Μηχανική Μάθηση

7ο Εξάμηνο 2022 – 2023

Assignment 2 – Solutions

Ζαρίφης Στέλιος – el20435

Email: el20435@mail.ntua.gr

Contents

Άσκηση 2.1

Έστω η κλάση H_{rec}^n των παράλληλων στους άξονες υπερ – παραλληλογράμμων του $\mathbb{R}^n$

$$H_{rec}^n = \left\{h_{(a_1,a_2,\dots,a_n,b_1,b_2,\dots,b_n)} : a_1 \le b_1, a_2 \le b_2, \dots, a_n \le b_n\right\},$$

$$h_{(a_1,a_2,\dots,a_n,b_1,b_2,\dots,b_n)}(x_1, x_2, \dots, x_n) = \begin{cases} 1, a_1 \le x_1 \le b_1, a_2 \le x_2 \le b_2, \dots, a_n \le x_n \le b_n \\ 0, else \end{cases}$$

Αν αποδείξουμε ότι η VC – διάσταση ($VC_{dim}(H_{rec}^n)$) της κλάσης είναι πεπερασμένη, τότε η τελευταία είναι PAC – εκπαιδεύσιμη.

Θα δείξουμε ότι $VC_{dim}(H_{rec}^n) < 2n + 1$.

Θεωρούμε ένα σύνολο με $2n + 1$ στοιχεία. Τα $2n$ σχηματίζουν ένα υπερπολύγωνο στον $\mathbb{R}^n$ και 1 βρίσκεται στο εσωτερικό του.

> Αυτή είναι η μόνη περίπτωση γιατί αν βρισκόταν στο εξωτερικό του, θα θεωρούσαμε εκείνο ως κορυφή του υπερπολυγώνου κι έτσι κάποιο άλλο θα ήταν στο εσωτερικό του.

Δείξαμε λοιπόν ότι $y_1 = y_2 = \cdots = y_{2n} = 1$ και $y_{2n+1} = 0$. Άρα δεν υπάρχει ταξινομητής στην κλάση H_{rec}^n που ταξινομεί τα στοιχεία του συγκεκριμένου συνόλου σωστά. Συνεπώς αποδείξαμε ότι

$$VC_{dim}(H_{rec}^n) < 2n + 1$$

Τελικά, δείξαμε και ότι η κλάση είναι PAC – εκπαιδεύσιμη, αφού

$$VC_{dim}(H_{rec}^n) < 2n + 1 < \infty$$

Άσκηση 2.2

Ερώτημα α

Έχουμε τους εξής περιορισμούς για τα u_{ij}

$$\sum_{j=1}^{m} u_{ij} = 1, \forall i \in \{1, 2, \dots, N\}$$

Θεωρούμε τη συνάρτηση κόστους

$$J_q(\theta, U) = \sum_{i=1}^{N}\sum_{j=1}^{m} u_{ij}^q d(x_i, \theta_j)$$

Μπορούμε να σχηματίσουμε τη Lagrange Function

$$\mathcal{L}(x, \theta, U, \lambda) = \sum_{i=1}^{N}\sum_{j=1}^{m} u_{ij}^q d(x_i, \theta_j) + \lambda^T u_j = \sum_{i=1}^{N}\sum_{j=1}^{m} u_{ij}^q d(x_i, \theta_j) + \sum_{i=1}^{N} \lambda_i \left(1 - \sum_{j=1}^{m} u_{ij}\right)$$

Minimum ως προς u_{kl} έχουμε όταν

$$\frac{\partial}{\partial u_{kl}} \mathcal{L}(x, \theta, U, \lambda) = 0 \Rightarrow q u_{kl}^{q-1} d(x_k, \theta_l) - \lambda_k = 0 \Rightarrow u_{kl} = \left(\frac{\lambda_k}{q d(x_k, \theta_l)}\right)^{\frac{1}{q-1}}, \begin{cases} \forall k \in \{1, 2, \dots, N\} \\ \forall l \in \{1, 2, \dots, m\} \end{cases} \quad (1)$$

Από τους περιορισμούς μπορούμε να δείξουμε

$$\sum_{j=1}^{m} u_{ij} = 1 \Rightarrow \sum_{j=1}^{m} \left(\frac{\lambda_i}{q d(x_i, \theta_j)}\right)^{\frac{1}{q-1}} = 1 \Rightarrow \lambda_i^{\frac{1}{q-1}} \sum_{j=1}^{m} \left(\frac{1}{q d(x_i, \theta_j)}\right)^{\frac{1}{q-1}} = 1 \Rightarrow$$

$$\Rightarrow \lambda_i^{-\frac{1}{q-1}} = \sum_{j=1}^{m} \left(\frac{1}{q d(x_i, \theta_j)}\right)^{\frac{1}{q-1}} \Rightarrow \lambda_i = \left[\sum_{j=1}^{m} \left(\frac{1}{q d(x_i, \theta_j)}\right)^{\frac{1}{q-1}}\right]^{1-q} \quad (2)$$

Συνεπώς από τις $(1, 2)$ έχουμε

$$u_{ij} = \left(\frac{\left[\sum_{k=1}^{m} \left(\frac{1}{q d(x_i, \theta_k)}\right)^{\frac{1}{q-1}}\right]^{1-q}}{q d(x_i, \theta_j)}\right)^{\frac{1}{q-1}} = \frac{\sum_{k=1}^{m} \left(\frac{1}{q d(x_i, \theta_k)}\right)^{-\frac{1}{q-1}}}{q d(x_i, \theta_j)^{\frac{1}{q-1}}} = \frac{1}{\sum_{k=1}^{m} \left(\frac{d(x_i, \theta_j)}{d(x_i, \theta_k)}\right)^{\frac{1}{q-1}}}$$

Για το δεύτερο ζητούμενο έχουμε

$$J(\theta, U) = \sum_{i=1}^{N}\sum_{j=1}^{m} u_{ij}^{q} d(x_i, \theta_j)$$

$$\hat{\theta} = \arg\min_{\theta} J(\theta, U)$$

Ελαχιστοποίηση όταν $\nabla_{\theta}\big(J(\theta, U)\big) = 0$. Ως προς την l συνιστώσα του $\boldsymbol{\theta}$ είναι

$$\frac{\partial J(\theta, U)}{\partial \theta_l} = 0 \xrightarrow{d(x_i,\theta_j)=\|x_i-\theta_j\|^2} \frac{\partial}{\partial \theta_l}\left[\sum_{i=1}^{N}\sum_{j=1}^{m}\left(\frac{1}{\sum_{k=1}^{m}\left(\frac{\|x_i-\theta_j\|^2}{\|x_i-\theta_k\|^2}\right)^{-\frac{1}{q-1}}}\right)\|x_i-\theta_j\|^2\right] = 0 \Rightarrow$$

$$\Rightarrow \sum_{i=1}^{N}\left(\frac{1}{\sum_{k=1}^{m}\left(\frac{-2(x_i-\theta_l)}{\|x_i-\theta_k\|^2}\right)^{-\frac{1}{q-1}}}\right)[-2(x_i-\theta_l)] = 0 \Rightarrow \frac{\partial}{\partial \theta_l}\left[\sum_{i=1}^{N}\sum_{j=1}^{m} u_{ij}^{q}\|x_i-\theta_j\|^2\right] = 0 \Rightarrow$$

$$\Rightarrow \sum_{i=1}^{N} u_{il}^{q}[-2(x_i-\theta_l)] = 0 \Rightarrow \sum_{i=1}^{N} u_{il}^{q} x_i = \sum_{i=1}^{N} u_{il}^{q}\theta_l \Rightarrow \theta_l = \frac{\sum_{i=1}^{N} u_{il}^{q} x_i}{\sum_{i=1}^{N} u_{il}^{q}}$$

Καταλήξαμε, λοιπόν, στις σχέσεις

$$u_{ij} = \frac{1}{\sum_{k=1}^{m}\left(\frac{d(x_i,\theta_j)}{d(x_i,\theta_k)}\right)^{\frac{1}{q-1}}} \text{ και } \theta_j = \frac{\sum_{i=1}^{N} u_{ij}^{q} x_i}{\sum_{i=1}^{N} u_{ij}^{q}}$$

Ερώτημα β

Χρησιμοποιούμε τις κλάσεις numpy και matplotlib:

```
import numpy as np
import matplotlib.pyplot as plt
```

Ορίζουμε τα μέσα διανύσματα και τους πίνακες συμμεταβλητότητας:

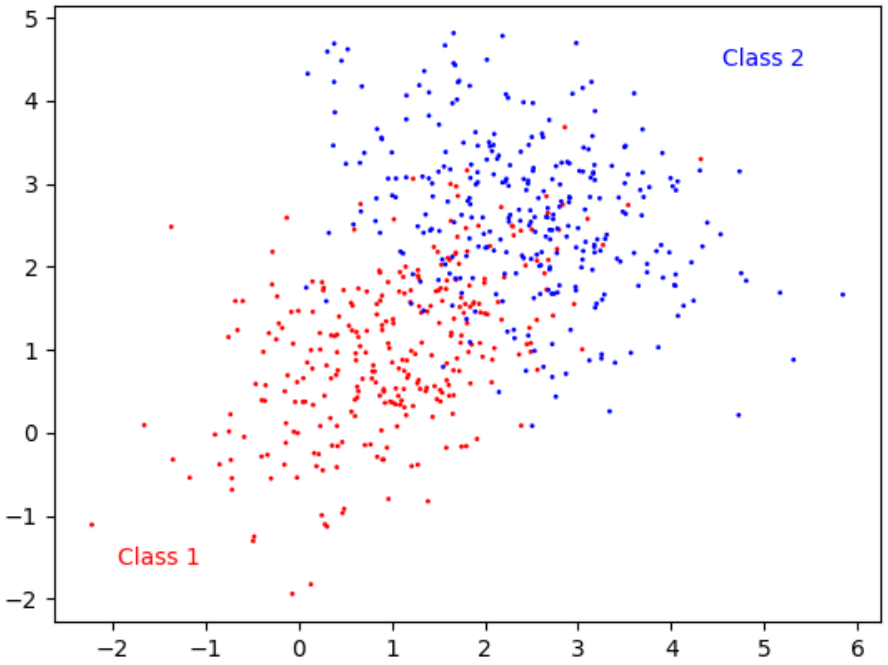
```
u1 = np.array([1, 1])
u2 = np.array([2.5, 2.5])
S1 = np.array([[1, 0.5], [0.5, 1]])
S2 = np.array([[1, -0.4], [-0.4, 1]])
```

Δημιουργούμε τα ζητούμενα datasets:

```
np.random.seed(42)
data1 = np.random.multivariate_normal(mean = u1, cov = S1, size = 300)
data2 = np.random.multivariate_normal(mean = u2, cov = S2, size = 300)
```

Σχεδιάζουμε τα δεδομένα σε κοινό διάγραμμα:

```
data = np.concatenate((data1, data2))

plt.scatter(data[:, 0], data[:, 1],
            c = ["r" if i < 300 else "b" for i in range(600)], s = 1)

plt.text(-1.5, -1.5, "Class 1", ha = "center", va = "center", color = "r")
plt.text(5, 4.5, "Class 2", ha = "center", va = "center", color = "b")

plt.show()
```



Ερώτημα γ

```
def kmeans(data, k, epsilon=0.0001, max_iter=100):
    # Initialize the cluster centers randomly
    centers = data[np.random.choice(np.arange(len(data)),
                  size=k, replace=False)]

    for _ in range(max_iter):
        # Assign each point to the nearest cluster center
        distances = np.linalg.norm(data[:, None, :] - centers,
                                   axis=2)
        labels = np.argmin(distances, axis=1)

        # Update the cluster centers
        new_centers = np.array([data[labels == i].mean(axis=0)
                                for i in range(k)])

        # Check for convergence
        if np.allclose(centers, new_centers, atol=epsilon):
            break

        centers = new_centers

    # Split the data into two datasets based on the cluster labels
    data1 = data[labels == 0]
    data2 = data[labels == 1]

    return labels, centers, data1, data2
```

Αρχικοποιούμε τα κέντρα των clusters με τυχαίο τρόπο.

Βρίσκουμε τη νόρμα για κάθε απόσταση σημείου του dataset από τα τρέχοντα κέντρα και αποδίδουμε κάθε σημείο στο κοντινότερό του κέντρο.

Υπολογίζουμε τα νέα κέντρα ως τους μέσους των ομάδων σημείων που μόλις αποδώσαμε.

Σταματάμε την αναζήτηση αν το κριτήριο παύσης ικανοποιήθηκε (δηλαδή αν οι αποστάσεις όλων των σημείων από το αντίστοιχό τους κέντρο είναι μικρότερες του ε).

Ανανεώνουμε τα κέντρα και συνεχίζουμε.

Επιστρέφουμε τα τελευταία κέντρα που αποδώσαμε καθώς και τα τελικά clusters

Ερώτημα δ

```
def d(p1, p2):
    return (p1[0]-p2[0])**2 + (p1[1]-p2[1])**2

def fuzzy_cmeans(data, epsilon=0.0001, max_iter=100):
    indices = random.sample(range(n), 2)
    u_fz = np.zeros([n,2])

    # Initialize random centers same as k-means
    centers_fz = data[indices,:] + 1e-7
    newcenters = np.zeros([2,2])

    # Stoping criterion
    e = 1
    iter_fz = 0
    iterations = 0
    while e >= epsilon:
        # Clusters
        for i in range(n):
            dn = d(data[i], centers_fz[0])
            dd = (1/d(data[i], centers_fz[0])) +
                 (1/d(data[i], centers_fz[1]))
            u_fz[i,0] = 1/ (dn*dd)
            u_fz[i][1] = 1 - u_fz[i][0]
            newcenters[0] += data[i] * (u_fz[i][0]**2)
            newcenters[1] += data[i] * (u_fz[i][1]**2)

        # Centers
        newcenters[0]= newcenters[0]/np.sum((u_fz[:,0])**2)
        newcenters[1]= newcenters[1]/np.sum((u_fz[:,1])**2)
        e = np.sqrt(d(centers_fz[0],newcenters[0])) +
            np.sqrt(d(centers_fz[1],newcenters[1]))
        centers_fz = newcenters
        newcenters = np.zeros([2,2])
        iter_fz += 1

        # Assign each point to the nearest cluster center
        distances = np.linalg.norm(data[:, None, :] - centers_fz, axis=2)
        labels = np.argmin(distances, axis=1)
        iterations += 1

    # Split the data into two datasets based on the cluster labels
    if (np.linalg.norm(centers_fz[0]) <= np.linalg.norm(centers_fz[1])):
        cluster1 = data[labels == 1]
        cluster2 = data[labels == 0]
    else:
        cluster1 = data[labels == 0]
        cluster2 = data[labels == 1]

    return labels, centers_fz, cluster1, cluster2
```

Αρχικοποιούμε τα κέντρα των clusters με τυχαίο τρόπο.

Βρίσκουμε τη νόρμα για κάθε απόσταση σημείου του dataset από τα τρέχοντα κέντρα και ανανεώνουμε τις συναρτήσεις συμμετοχής και τα κέντρα.

Υπολογίζουμε τα νέα κέντρα ως τους μέσους των κέντρων που μόλις βρήκαμε.

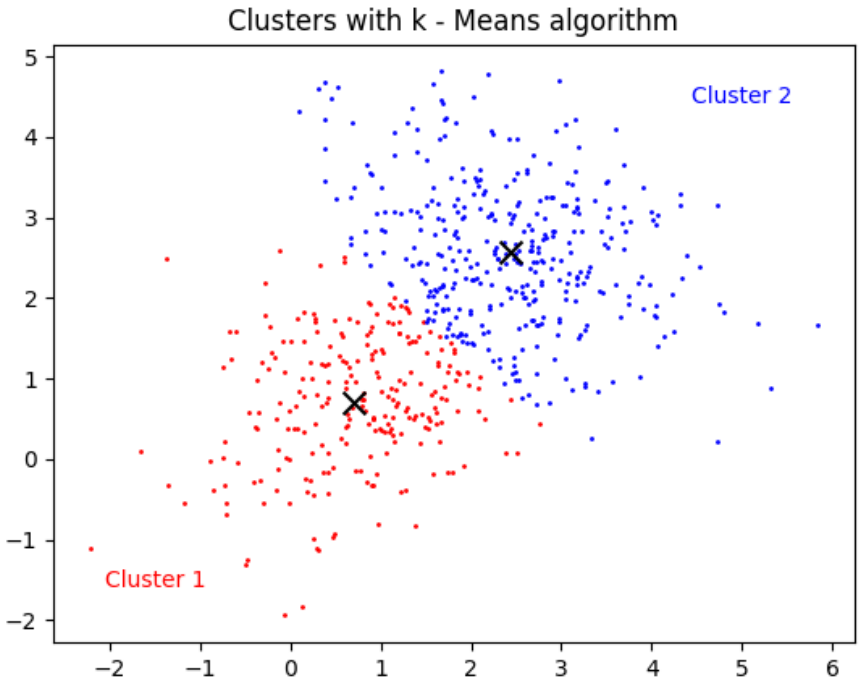
Σταματάμε την αναζήτηση αν το κριτήριο παύσης ικανοποιήθηκε (δηλαδή αν οι αποστάσεις όλων των σημείων από το αντίστοιχό τους κέντρο είναι μικρότερες του ε).

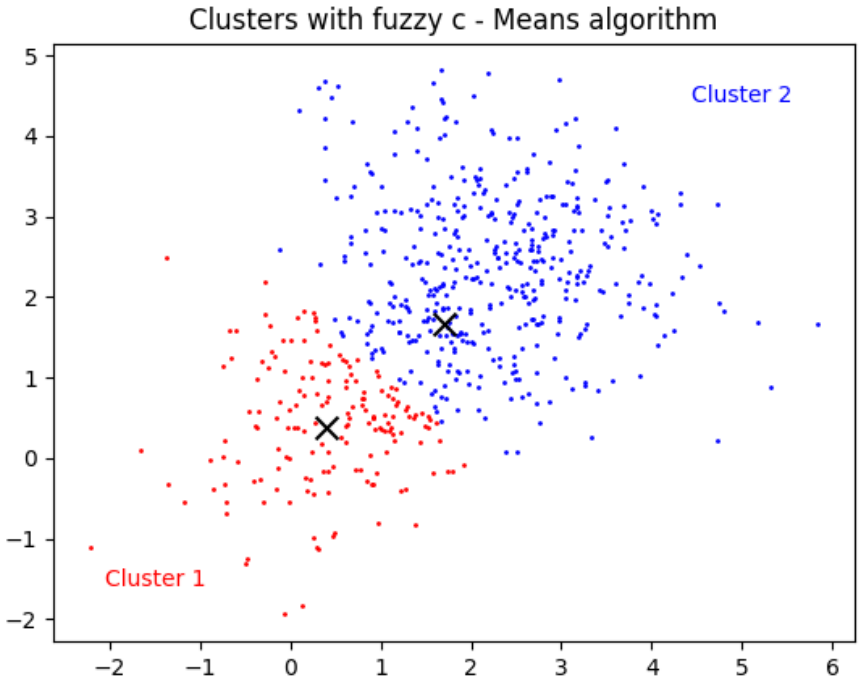
Αποδίδουμε τα σημεία στις ομάδες βάσει απόστασης.

Επιστρέφουμε τα τελευταία κέντρα που αποδώσαμε καθώς και τα τελικά clusters

Ερώτημα ε

Τυπώνουμε τα διαγράμματα των clusters που δημιούργησαν οι αλγόριθμοι $k - means$ και $fuzzy\ c - means$:





Υποερώτημα i

Χρησιμοποιήσαμε ένα μετρητή σε κάθε αλγόριθμο και τους τρέξαμε 10000 φορές με τυχαία δείγματα ως εισόδους. Μετρήσαμε ότι ο αλγόριθμος $k-means$ χρειάζεται 6.8681 επαναλήψεις ενώ ο $fuzzy\ c-means$ 16.5841, κατά μέσο όρο. Παρατηρούμε ότι ο $kmeans$ είναι ~ 2.41 φορές ταχύτερος, για τέτοιου είδους datasets.

Υποερώτημα ii

Όπως φαίνεται καθαρά και στα διαγράμματα, τα εκτιμόμενα κέντρα που προκύπτουν από τον αλγόριθμο $k-means$ είναι πολύ πιο κοντά στα πραγματικά από εκείνα του αλγορίθμου $fuzzy\ c-means$. Τα τελευταία τείνουν να συγκλίνουν μεταξύ τους.

Υποερώτημα iii

```
correctK = 0
for point in clusteredData1:
      if (point in data1):
            correctK += 1

for point in clusteredData2:
      if (point in data2):
            correctK += 1

print("K-means accuracy:", correctK/600)
```

Προσθέτοντας αυτές τις γραμμές στον αλγόριθμο $k-means$ παρακολουθούμε πόσα σημεία ταξινομήθηκαν σωστά. Παρατηρήσαμε accuracy περίπου 85%.

```
correctC = 0
for point in clusteredData1:
      if (point in data1):
            correctC += 1

for point in clusteredData2:
      if (point in data2):
            correctC += 1

print("C-means accuracy:", correctC/600)
```

Προσθέτοντας αυτές τις γραμμές στον αλγόριθμο $fuzzy\ c-means$ παρακολουθούμε πόσα σημεία ταξινομήθηκαν σωστά. Παρατηρήσαμε accuracy περίπου 75%.

Άρα παρατηρήσαμε καλύτερη επίδοση του αλγορίθμου $k-means$ σε σχέση με τον $fuzzy\ c-means$, τόσο χρονικά (αριθμός επαναλήψεων), όσο και σε accuracy.

Άσκηση 2.3

Ερώτημα α

Θεωρούμε δύο σημεία στο $\mathbb{R}^l$, $\boldsymbol{x} = [x_1, x_2, \dots, x_l]^T$ και $\boldsymbol{y} = [y_1, y_2, \dots, y_l]^T$. Έστω $|x_i - y_i| = \max_{j=1,2,\dots,l}\{|x_j - y_j|\}$. Ορίζουμε την απόστασή τους ως εξής

$$d_n(\boldsymbol{x}, \boldsymbol{y}) = |x_i - y_i| + \frac{1}{l/2+1}\sum_{\substack{j=1 \\ j\neq i}}^{l}|x_j - y_j|$$

Οι ιδιότητες που χρειάζεται να πληροί η απόσταση $d_n(x, y)$ για να είναι μετρική στο $\mathbb{R}^l \times \mathbb{R}^l$ είναι:

1. Μη Αρνητική: $d_n(\boldsymbol{x}, \boldsymbol{y}) \geq 0, \forall x, y \in \mathbb{R}^l$
2. Συμμετρία: $d_n(\boldsymbol{x}, \boldsymbol{y}) = d_n(\boldsymbol{y}, \boldsymbol{x}), \forall x, y \in \mathbb{R}^l$
3. Τριγωνική Ανισότητα: $d_n(\boldsymbol{x}, \boldsymbol{z}) \leq d_n(\boldsymbol{x}, \boldsymbol{y}) + d_n(\boldsymbol{y}, \boldsymbol{z})$

Ιδιότητα 1

Αφού $|x_i - y_i| = \max_{j=1,2,\dots,l}\{|x_j - y_j|\} \geq 0$ και η $d_n(\boldsymbol{x}, \boldsymbol{y})$ είναι άθροισμα τέτοιων όρων, θα ισχύει πάντα

$$d_n(\boldsymbol{x}, \boldsymbol{y}) \geq 0, \forall x, y \in \mathbb{R}^l$$

Ιδιότητα 2

Αφού $|x_i - y_i| = |y_i - x_i|, \forall x_i, y_i \in \mathbb{R}^l$, τότε μπορούμε να γράψουμε

$$d_n(\boldsymbol{x}, \boldsymbol{y}) = |x_i - y_i| + \frac{1}{l/2+1}\sum_{\substack{j=1 \\ j\neq i}}^{l}|x_j - y_j| = |y_i - x_i| + \frac{1}{l/2+1}\sum_{\substack{j=1 \\ j\neq i}}^{l}|y_j - x_j| = d_n(\boldsymbol{y}, \boldsymbol{x})$$

Ιδιότητα 3

$$d_n(\boldsymbol{x}, \boldsymbol{z}) = |x_i - z_i| + \frac{1}{l/2+1}\sum_{\substack{j=1 \\ j\neq i}}^{l}|x_j - z_j| = |x_i - y_i + y_i - z_i| + \frac{1}{l/2+1}\sum_{\substack{j=1 \\ j\neq i}}^{l}|x_j - y_j + y_j - z_j| =$$

Αλλά όπως ορίστηκε η $|x_i - y_i|$, είναι μετρική, άρα ισχύει η τριγωνική ανισότητα, δηλαδή

$$|x_i - y_i + y_i - z_i| \leq |x_i - y_i| + |y_i - z_i|$$

Άρα

$$d_n(\boldsymbol{x},\boldsymbol{z}) \le |x_i - y_i| + |y_i - z_i| + \frac{1}{l/2+1}\sum_{\substack{j=1\\ j\neq i}}^{l}\left(|x_j - y_j| + |y_j - z_j|\right) \Rightarrow$$

$$\Rightarrow d_n(\boldsymbol{x},\boldsymbol{z}) \le |x_i - y_i| + \frac{1}{l/2+1}\sum_{\substack{j=1\\ j\neq i}}^{l}|x_j - y_j| + |y_i - z_i| + \frac{1}{l/2+1}\sum_{\substack{j=1\\ j\neq i}}^{l}|y_j - z_j| \Rightarrow$$

$$d_n(\boldsymbol{x},\boldsymbol{z}) \le d_n(\boldsymbol{x},\boldsymbol{y}) + d_n(\boldsymbol{y},\boldsymbol{z})$$

Ερώτημα β

Θεωρούμε το σύνολο προτύπων

$$X = \{(0,3), (1.4,2.6), (-1.5,3.4), (-0.2,-0.4), (1,-1), (2,-1.5), (2.6,-1.8), (3,-2)\}$$

Έχουμε πίνακα προτύπων

$$D(X) = \left[\left(x_1^i, x_2^i\right)\right] = \begin{bmatrix} 0 & 3 \\ 1.4 & 2.6 \\ -1.5 & 3.4 \\ -0.2 & -0.4 \\ 1 & -1 \\ 2 & -1.5 \\ 2.6 & -1.8 \\ 3 & -2 \end{bmatrix}$$

Και πίνακα εγγύτητας

$$P(X) = \left[d_n\left(\boldsymbol{x}^i, \boldsymbol{x}^j\right)\right] = \begin{bmatrix} 0 & 1.6 & 1.7 & 3.5 & 4.5 & 5.5 & 6.1 & 6.5 \\ 1.6 & 0 & 3.3 & 3.8 & 3.8 & 4.4 & 5 & 5.4 \\ 1.7 & 3.3 & 0 & 4.45 & 5.65 & 6.65 & 7.25 & 7.65 \\ 3.5 & 3.8 & 4.45 & 0 & 1.5 & 2.75 & 3.5 & 4 \\ 4.5 & 3.8 & 5.65 & 1.5 & 0 & 1.25 & 2 & 2.5 \\ 5.5 & 4.4 & 6.65 & 2.75 & 1.25 & 0 & 0.75 & 1.25 \\ 6.1 & 5 & 7.25 & 3.5 & 2 & 0.75 & 0 & 0.5 \\ 6.5 & 5.4 & 7.65 & 4 & 2.5 & 1.25 & 0.5 & 0 \end{bmatrix}$$

Ερώτημα γ

Ο αλγόριθμος απλού δεσμού έχει ως εξής:

Αρχικοποιούμε $P_0 = P(X)$

Επαναλαμβάνουμε με $i = 1$ ως 6 φορές (μέχρι ο P να μείνει με 1 στοιχείο):

Εντοπίζουμε το μικρότερο στοιχείο (p_i, p_j) του P_{i-1}, εκτός της διαγωνίου του. Τα αντίστοιχα x_i, x_j είναι το ζεύγος των κοντινότερων σημείων. Τα ομαδοποιούμε συμβολίζοντας με ενα «Π» ύψους όσο το $P[p_i, p_j]$, όπως φαίνεται στο σχήμα.

Ανανεώνουμε τη νέα απόσταση των υπολοίπων στοιχείων από το Cluster που φτιάξαμε ως την ελάχιστη με κάποιο που ανήκει στο Cluster

Δηλαδή έχουμε

	P1	P2	P3	P4	P5	P6	P7	P8
P1	0							
P2	1.6	0						
P3	1.7	3.3	0					
P4	3.5	3.8	4.45	0				
P5	4.5	3.8	5.65	1.5	0			
P6	5.5	4.4	6.65	2.75	1.25	0		
P7	6.1	5	7.25	3.5	2	0.75	0	
P8	6.5	5.4	7.65	4	2.5	1.25	0.5	0

	P1	P2	P3	P4	P5	P6	P7, P8
P1	0						
P2	1.6	0					
P3	1.7	3.3	0				
P4	3.5	3.8	4.45	0			
P5	4.5	3.8	5.65	1.5	0		
P6	5.5	4.4	6.65	2.75	1.25	0	
P7, P8	6.1	5	7.25	3.5	2	0.75	0

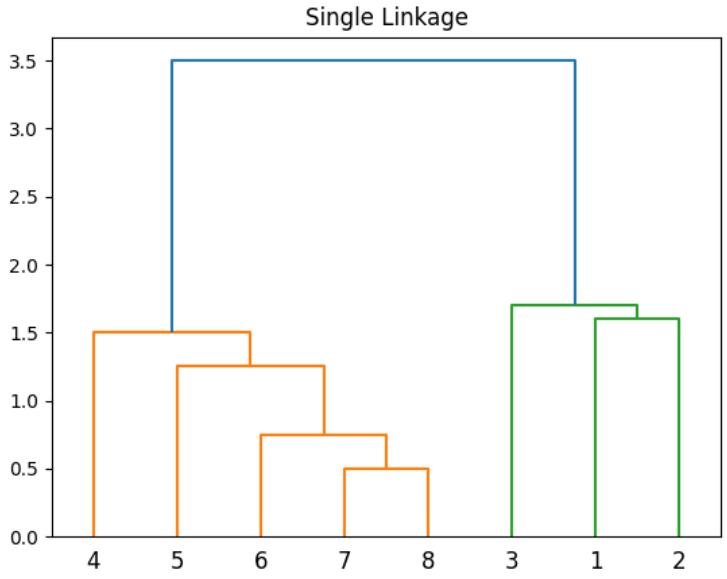
	P1	P2	P3	P4	P5	P6, P7, P8
P1	0					
P2	1.6	0				
P3	1.7	3.3	0			
P4	3.5	3.8	4.45	0		
P5	4.5	3.8	5.65	1.5	0	
P6, P7, P8	5.5	4.4	6.65	2.75	1.25	0

	P1	P2	P3	P4	P5, P6, P7, P8
P1	0				
P2	1.6	0			
P3	1.7	3.3	0		
P4	3.5	3.8	4.45	0	
P5, P6, P7, P8	4.5	3.8	5.65	1.5	0

	P1	P2	P3	P4, P5, P6, P7, P8
P1	0			
P2	1.6	0		
P3	1.7	3.3	0	
P4, P5, P6, P7, P8	3.5	3.8	4.45	0

	P1, P2	P3	P4, P5, P6, P7, P8
P1, P2	0		
P3	1.7	0	
P4, P5, P6, P7, P8	3.5	4.45	0

	P1, P2, P3	P4, P5, P6, P7, P8
P1, P2, P3	0	
P4, P5, P6, P7, P8	3.5	0



Ερώτημα δ

Ο αλγόριθμος πλήρους δεσμού έχει ως εξής:

Αρχικοποιούμε $P_0 = P(X)$

Επαναλαμβάνουμε με $i = 1$ ως 6 φορές (μέχρι ο P να μείνει με 1 στοιχείο):

> Εντοπίζουμε το μικρότερο στοιχείο (p_i, p_j) του P_{i-1}, εκτός της διαγωνίου του. Τα αντίστοιχα x_i, x_j είναι το ζεύγος των κοντινότερων σημείων. Τα ομαδοποιούμε συμβολίζοντας με ενα «Π» ύψους όσο το $P[p_i, p_j]$, όπως φαίνεται στο σχήμα.
>
> Ανανεώνουμε τη νέα απόσταση των υπολοίπων στοιχείων από το Cluster που φτιάξαμε ως την μέγιστη με κάποιο που ανήκει στο Cluster

Δηλαδή έχουμε

	P1	P2	P3	P4	P5	P6	P7	P8
P1	0							
P2	1.6	0						
P3	1.7	3.3	0					
P4	3.5	3.8	4.45	0				
P5	4.5	3.8	5.65	1.5	0			
P6	5.5	4.4	6.65	2.75	1.25	0		
P7	6.1	5	7.25	3.5	2	0.75	0	
P8	6.5	5.4	7.65	4	2.5	1.25	0.5	0

	P1	P2	P3	P4	P5	P6	P7, P8
P1	0						
P2	1.6	0					
P3	1.7	3.3	0				
P4	3.5	3.8	4.45	0			
P5	4.5	3.8	5.65	1.5	0		
P6	5.5	4.4	6.65	2.75	1.25	0	
P7, P8	6.5	5.4	7.65	4	2.5	1.25	0

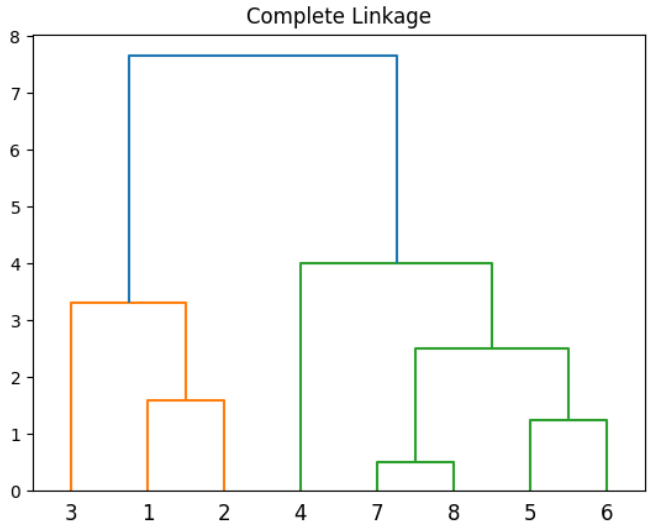
	P1	P2	P3	P4	P5	P6, P7, P8
P1	0					
P2	1.6	0				
P3	1.7	3.3	0			
P4	3.5	3.8	4.45	0		
P5, P6	5.5	4.4	6.65	2.75	0	
P7, P8	6.5	5.4	7.65	4	2.5	0

	P1, P2	P3	P4	P5, P6	P7, P8
P1, P2	0				
P3	3.3	0			
P4	3.8	4.45	0		
P5, P6	5.5	6.65	2.75	0	
P7, P8	6.5	7.65	4	2.5	0

	P1, P2	P3	P4	P5, P6, P7, P8
P1, P2	0			
P3	3.3	0		
P4	3.8	4.45	0	
P5, P6, P7, P8	6.5	7.65	4	0

	P1, P2, P3	P4	P5, P6, P7, P8
P1, P2, P3	0		
P4	4.45	0	
P5, P6, P7, P8	7.65	4	0

	P1, P2, P3	P4, P5, P6, P7, P8
P1, P2, P3	0	
P4, P5, P6, P7, P8	7.65	0



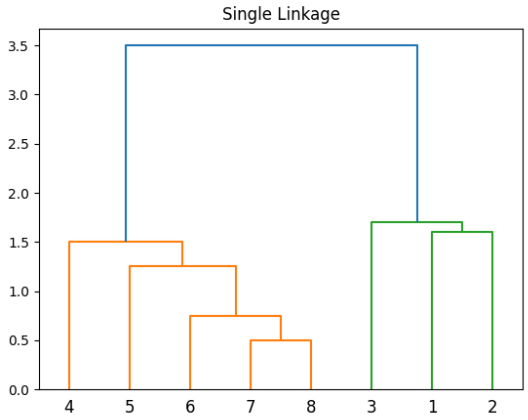
*Παρατήρηση: Οι αλγόριθμοι αυτοί δεν καταλήγουν πάντα στο ίδιο δενδρόγραμμα. Για παράδειγμα στον αλγόριθμο πλήρους δεσμού, θα μπορούσαμε να είχαμε επιλέξει το άλλο στοιχείο με απόσταση 1.25 και θα καταλήγαμε σε λίγο διαφορετική δομή κοντά στα φύλλα.

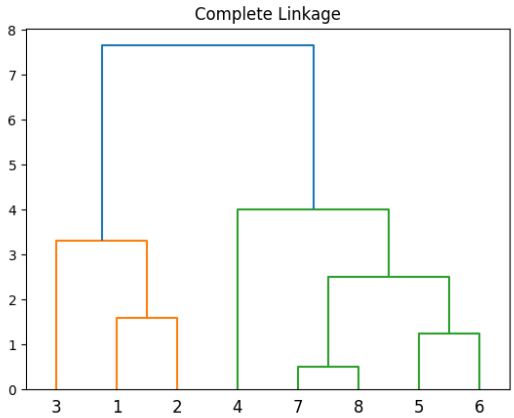
	P1	P2	P3	P4	P5	P6	P7, P8
P1	0						
P2	1.6	0					
P3	1.7	3.3	0				
P4	3.5	3.8	4.45	0			
P5	4.5	3.8	5.65	1.5	0		
P6	5.5	4.4	6.65	2.75	1.25	0	
P7, P8	6.5	5.4	7.65	4	2.5	1.25	0

Ερώτημα ε

Μπορούμε να παρατηρήσουμε ότι ο αλγόριθμος απλού δεσμού αναδεικνύει τις επιμήκεις ομάδες πρώτα ενώ ο πλήρους δεσμού τις συμπαγείς πρώτα. Από τα διαγράμματα που παρήχθησαν εκτελώντας τους ιεραρχικούς αλγορίθμους κρίνουμε ότι πρέπει να ορίσουμε την ομαδοποίηση «τεμαχίζοντας» τα διαγράμματα σε ύψος 2 για την πρώτη περίπτωση και σε ύψος 4 για τη δύτερη. Αυτό διότι οι ομάδες που θα δημιουργηθούν απέχουν μεταξύ τους τη μεγαλύτερη απόσταση σε σχέση με τις ομάδες που θα προέκυπταν σε άλλη ομαδοποίηση. Έτσι καταλήγουμε με τις εξής ομάδες:

1. Single Linkage: $\{P_1, P_2, P_3\}$ $\kappa\alpha\iota$ $\{P_4, P_5, P_6, P_7, P_8\}$
2. Complete Linkage: $\{P_1, P_2, P_3\}$ $\kappa\alpha\iota$ $\{P_4\}$ $\kappa\alpha\iota$ $\{P_5, P_6, P_7, P_8\}$





Άσκηση 2.4

Θεωρούμε την τυχαία ακολουθία $x_n \in \mathbb{R}^d, n = 1, 2, \dots, N$ με $\mathbb{E}[x_n] = 0$, μοναδιαίο $e \in \mathbb{R}^d$ και σταθερές a_n. Προσεγγίζουμε κάθε x_n με την ποσότητα $a_n e$ επιδιώκοντας ελάχιστο μέσο τετραγωνικό λάθος

$$J(a_1, a_2, \dots, a_n, e) = \frac{1}{N}\sum_{n=1}^{N} \|x_n - a_n e\|^2$$

Ερώτημα α

Απλοποιούμε τη συνάρτηση κόστους

$$J(a_1, a_2, \dots, a_n, e) = \frac{1}{N}\sum_{n=1}^{N} \|x_n - a_n e\|^2 = \frac{1}{N}\sum_{n=1}^{N} (x_n - a_n e)^T (x_n - a_n e) =$$
$$= \frac{1}{N}\sum_{n=1}^{N} (x_n^T - a_n e^T)(x_n - a_n e) = \frac{1}{N}\sum_{n=1}^{N} (x_n^T x_n - a_n e^T x_n - a_n x_n^T e + a_n^2 e^T e)$$

Επειδή τα e, x_n είναι διανύσματα στήλες, τότε το εσωτερικό τους γινόμενο ορίζεται ως

$$\langle e, x_n \rangle = \langle x_n, e \rangle = e^T x_n = x_n^T e$$

Επίσης επειδή το e είναι μοναδιαίο, είναι $e^T e = \|e\|^2 = 1$

Από αυτές τις σχέσεις, η J γίνεται

$$J(a_1, a_2, \dots, a_n, e) = \frac{1}{N}\sum_{n=1}^{N} (x_n^T x_n - a_n e^T x_n - a_n x_n^T e + a_n^2 e^T e) = \frac{1}{N}\sum_{n=1}^{N} (x_n^T x_n - 2a_n x_n^T e + a_n^2)$$

Ελάχιστο τετραγωνικό σφάλμα (ως προς τη σταθερά a_n που «ελέγχουμε») έχουμε όταν

$$\frac{\partial J}{\partial a_n} = 0 \Rightarrow \frac{\partial}{\partial a_n}\left[\frac{1}{N}\sum_{n=1}^{N} (x_n^T x_n - 2a_n x_n^T e + a_n^2)\right] = 0 \Rightarrow \frac{1}{N}(-2x_n^T e + 2a_n) = 0 \Rightarrow$$
$$\Rightarrow a_n = x_n^T e = \langle x_n, e \rangle$$

Επειδή

$$\frac{\partial^2 J}{\partial a_n^2} = \frac{2}{N}$$

Η J είναι κυρτή, άρα το ελάχιστο που βρήκαμε είναι μοναδικό.

Ερώτημα β

Με αντικατάσταση $a_n = x_n^T e$ στη συνάρτηση κόστους, έχουμε

$$J_1(e) = \frac{1}{N}\sum_{n=1}^{N}(x_n^T x_n - 2x_n^T e x_n^T e + x_n^T e x_n^T e) = \frac{1}{N}\sum_{n=1}^{N}(x_n^T x_n - x_n^T e x_n^T e) \xrightarrow{x_n^T e = e^T x_n}$$

$$\Rightarrow J_1(e) = \frac{1}{N}\sum_{n=1}^{N}(x_n^T x_n - e^T x_n x_n^T e) = \frac{1}{N}\sum_{n=1}^{N} x_n^T x_n - \frac{1}{N}\sum_{n=1}^{N} e^T x_n x_n^T e = \frac{1}{N}\sum_{n=1}^{N} x_n^T x_n - e^T \frac{1}{N}\sum_{n=1}^{N} x_n x_n^T e$$

Όμως, $x_n^T x_n = \|x_n\|^2$ και ο εμπειρικός πίνακας αυτοσυσχέτισης είναι

$$R_x = \frac{1}{N}\sum_{n=1}^{N} x_n x_n^T$$

Άρα είναι

$$J_1(e) = -e^T R_x e + \frac{1}{N}\sum_{n=1}^{N}\|x_n\|^2$$

Ερώτημα γ

Αξιοποιούμε το γεγονός ότι $\|e\|^2 = e^T e = 1$ και έχουμε

$$J_1(e) = -e^T R_x e + \frac{1}{N}\sum_{n=1}^{N}\|x_n\|^2$$

$$Περιορισμοί\ e^T e - 1 = 0$$

Αφού τα x_n ως είσοδοι, δε μεταβάλλονται, η ποσότητα $J_1(e)$ ελαχιστοποιείται όταν ελαχιστοποιείται το

$$-e^T R_x e$$

Θεωρούμε μια νέα συνάρτηση κόστους με τους πολλαπλασιαστές Lagrange

$$\mathcal{L}(e) = -e^T R_x e + \lambda(e^T e - 1)$$

Κρίσιμα σημεία:

$$\frac{\partial \mathcal{L}}{\partial e} = 0 \Rightarrow -(R_x + R_x^T)e + 2\lambda e = 0 \xrightarrow{R_x^T = R_x} 2R_x e = 2\lambda e \Rightarrow R_x e = \lambda e$$

Δείξαμε δηλαδή ότι έχουμε κρίσιμα σημεία ακριβώς όταν το διάνυσμα e ικανοποιεί την εξίσωση ιδιοτιμών – ιδιοδιανυσμάτων του πίνακα R_x, δηλαδή όταν το e είναι ιδιοδιάνυσμα του R_x. Εύκολα μπορούμε να δούμε ότι η τετραγωνική μορφή $-e^T R_x e$ ελαχιστοποιείται όταν επιλέξουμε το e να αντιστοιχεί στη μέγιστη ιδιοτιμή του R_x.

Παρατήρηση: Εφόσον $\|e\|^2 = e^T e = 1$, μπορούμε να γράψουμε

$$J_1(e) = -\frac{e^T R_x e}{e^T e} + \frac{1}{N}\sum_{n=1}^{N}\|x_n\|^2$$

Βλέπουμε ότι η ποσότητα $e^T R_x e/(e^T e)$ που πρέπει να μεγιστοποιηθεί είναι ένα πηλίκο Rayleigh και έχει την ιδιότητα να μεγιστοποείται όταν το διάνυσμα γίνεται η μέγιστη ιδιοτιμή του R_x.

Ερώτημα δ

Αν Η $J_1(e)$ είναι η συνάρτηση κόστους που ελαχιστοποιείται για την εύρεση των κύριων συνιστωσών του συνόλου δεδομένων $X = \{x_1, x_2, \dots, x_n\}$. Ο πίνακας R_x είναι ο πίνακας συνδιακύμανσης του συνόλου X και το e είναι ένα μοναδιαίο διάνυσμα που αντιπροσωπεύει την κύρια συνιστώσα του. Έτσι, ο όρος $e^T R_x e$ εκφράζει τη διακύμανση των δεδομένων κατά μήκος της κατεύθυνσης της κύριας συνιστώσας. Ο στόχος της PCA είναι να βρεθούν τα ιδιοδιανύσματα του πίνακα συνδιακύμανσης R_x που αντιστοιχούν στις μεγαλύτερες ιδιοτιμές, καθώς αυτά θα είναι οι κύριες συνιστώσες του συνόλου δεδομένων.

Ερώτημα ε

Εφαρμόζοντας την SVD στον πίνακα $X_{N\times d}$ που για γραμμές του έχει τα x_i^T, έχουμε

$$X = U\Sigma V^T$$

Όπου $U_{N\times N}$ πίνακας αριστερών singular vectors, $\Sigma_{N\times d}$ διαγώνιος πίνακας singular values και $V_{d\times d}^T$ πίνακας δεξιών singular vectors.

Αν, όπως δείξαμε προηγουμένως, θεωρήσουμε ότι οι κύριες συνιστώσες του X είναι τα ιδιοδιανύσματα του πίνακα διακύμανσης R_x, οι κύριες συνιστώσες των δεδομένων είναι οι στήλες του πίνακα V και οι αντίστοιχες ιδιοτιμές δίνονται από τα διαγώνια στοιχεία του πίνακα Σ.

Στην PCA αναζητούμε τα ιδιοδιανύσματα του πίνακα διακύμανσης R_x που αντιστοιχούν στις μεγαλύτερες ιδιοτιμές μέσω της εύρεσης των δεξιών μοναδιαίων διανυσμάτων του X που αντιστοιχούν στις μεγαλύτερες singular values. Δηλαδή, οι κύριες συνιστώσες των δεδομένων δίνονται από τις στήλες του πίνακα V που αντιστοιχούν στις μεγαλύτερες singular values του πίνακα Σ που αντιιστοιχούν στις ιδιοτιμές του R_x.

Άσκηση 2.5

2.5.0 Import necessary libraries

```
import pandas as pd
import numpy as np
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
```

*2.5.α Load **PCA_iris_data.csv***

```
iris_data = pd.read_csv("PCA_iris_data")
iris_data
```

```
     5.1  3.5  1.4  0.2     Iris-setosa
0    4.9  3.0  1.4  0.2     Iris-setosa
1    4.7  3.2  1.3  0.2     Iris-setosa
2    4.6  3.1  1.5  0.2     Iris-setosa
3    5.0  3.6  1.4  0.2     Iris-setosa
4    5.4  3.9  1.7  0.4     Iris-setosa
..   ...  ...  ...  ...             ...
144  6.7  3.0  5.2  2.3  Iris-virginica
145  6.3  2.5  5.0  1.9  Iris-virginica
146  6.5  3.0  5.2  2.0  Iris-virginica
147  6.2  3.4  5.4  2.3  Iris-virginica
148  5.9  3.0  5.1  1.8  Iris-virginica

[149 rows x 5 columns]
```

2.5.β Normalize data

```
# calculate the mean and standard deviation for each column
iris_data_mean = iris_data.iloc[:,:-1].mean()
iris_data_std = iris_data.iloc[:,:-1].std()

# subtract the mean and divide by the standard deviation for each column
iris_data.iloc[:,:-1] = (iris_data.iloc[:,:-1] - iris_data_mean) /
iris_data_std

iris_data
```

```
          5.1       3.5       1.4       0.2     Iris-setosa
0   -1.144495 -0.117663 -1.349413 -1.320609     Iris-setosa
1   -1.385868  0.343699 -1.406243 -1.320609     Iris-setosa
2   -1.506555  0.113018 -1.292584 -1.320609     Iris-setosa
3   -1.023809  1.266424 -1.349413 -1.320609     Iris-setosa
4   -0.541064  1.958467 -1.178925 -1.057898     Iris-setosa
..        ...       ...       ...       ...             ...
144  1.027859 -0.117663  0.810106  1.437859  Iris-virginica
145  0.545114 -1.271068  0.696447  0.912437  Iris-virginica
146  0.786486 -0.117663  0.810106  1.043792  Iris-virginica
147  0.424427  0.805062  0.923765  1.437859  Iris-virginica
```

```
148  0.062368 -0.117663  0.753276  0.781081  Iris-virginica

[149 rows x 5 columns]
```

2.5.γ Calculate sample covariance matrix

```
# Assuming iris_data_standardized is your DataFrame
N = iris_data.shape[0] - 1

# calculate the sample covariance matrix
cov_matrix = (1/N)*np.dot(iris_data.iloc[:,:-1].T, iris_data.iloc[:,:-1])

cov_matrix
```

```
array([[ 1.        , -0.10378415,  0.87128294,  0.81697087],
       [-0.10378415,  1.        , -0.41521773, -0.35073314],
       [ 0.87128294, -0.41521773,  1.        ,  0.9623143 ],
       [ 0.81697087, -0.35073314,  0.9623143 ,  1.        ]])
```

2.5.δ Factorize sample covariance matrix using SVD

```
# factorize the covariance matrix using SVD
U, s, V_T = np.linalg.svd(cov_matrix)

# calculate the eigenvectors and eigenvalues
eigenvalues = s**2
eigenvectors = V_T.T

print("Eigenvalues:\n", eigenvalues)
print("\nCorresponding eigenvectors:\n", eigenvectors)
```

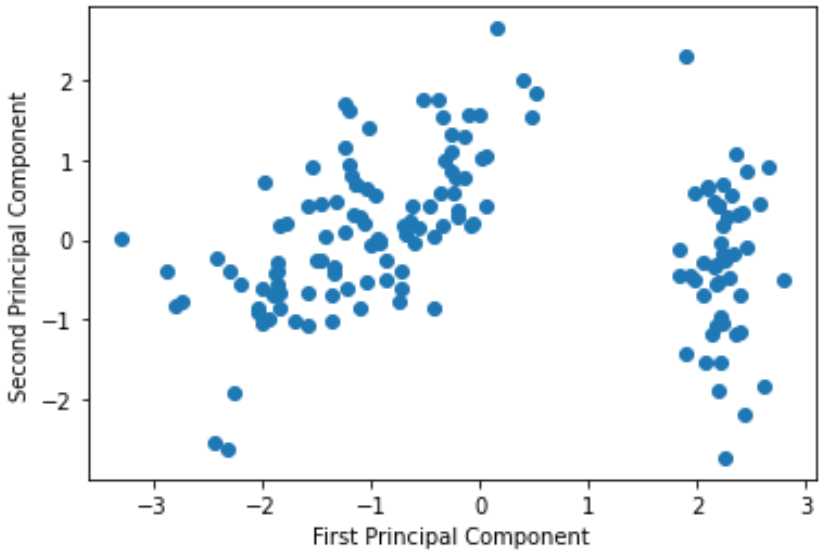
```
Eigenvalues:
 [8.43787078e+00 8.57283911e-01 2.20405565e-02 4.34379376e-04]

Corresponding eigenvectors:
 [[-0.52308496 -0.36956962  0.72154279  0.26301409]
  [ 0.25956935 -0.92681168 -0.2411952  -0.12437342]
  [-0.58184289 -0.01912775 -0.13962963 -0.80099722]
  [-0.56609604 -0.06381646 -0.63380158  0.52321917]]
```

Αυτή η εφαρμογή επιστρέφει τις ιδιοτιμές με φθίνουσα σειρά. Αυτό είναι χρήσιμο διότι η σειρά σημαντικότητας των συνιστωσών είναι σαφής!

2.5.ε Project on the first 2 principal components

```
# project data onto first two principal components
PCs = eigenvectors[:, :2]
projected_data = np.dot(iris_data.iloc[:,:-1], PCs)
```

```
# plot the projected data
plt.scatter(projected_data[:, 0], projected_data[:, 1])
plt.xlabel("First Principal Component")
plt.ylabel("Second Principal Component")
plt.show()
```



Τα δεδομένα `projected_data` είναι η προβολή των αρχικών σε χώρο χαμηλότερης διάστασης (2D).

Το διάγραμμα δίνει μια οπτική αναπαράσταση του τρόπου με τον οποίο τα δεδομένα κατανέμονται στον χώρο των 2 διαστάσεων καιοπτικοποιεί πόσο καλά τα δεδομένα προβάλλονται στις δύο πρώτες κύριες συνιστώσες.

Προσοχή: Πληροφορία πάντοτε χάνεται, αφού κρατάμε μόνο κάποια από τα πρώτα ιδιοδιανύσματα τα οποία κωδικοποιούν μέρος της πληροφορίας

$$C = U\Sigma V^T = \begin{pmatrix} u_{1,1} & u_{1,2} & \cdots & u_{1,n} \\ u_{2,1} & u_{2,2} & \cdots & u_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ u_{m,1} & u_{m,2} & \cdots & u_{m,n} \end{pmatrix} \begin{pmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_n^2 \end{pmatrix} \begin{pmatrix} u_{1,1} & u_{1,2} & \cdots & u_{1,n} \\ u_{2,1} & u_{2,2} & \cdots & u_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ u_{m,1} & u_{m,2} & \cdots & u_{m,n} \end{pmatrix}^T \Longrightarrow$$

$$\Longrightarrow C = \begin{pmatrix} | & | & & | \\ | & | & & | \\ u_1 & u_2 & \cdots & u_n \\ | & | & & | \\ | & | & & | \end{pmatrix} \begin{pmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_n^2 \end{pmatrix} \begin{pmatrix} — & — & u_1^T & — & — \\ — & — & u_2^T & — & — \\ & & \vdots & & \\ — & — & u_n^T & — & — \end{pmatrix}$$

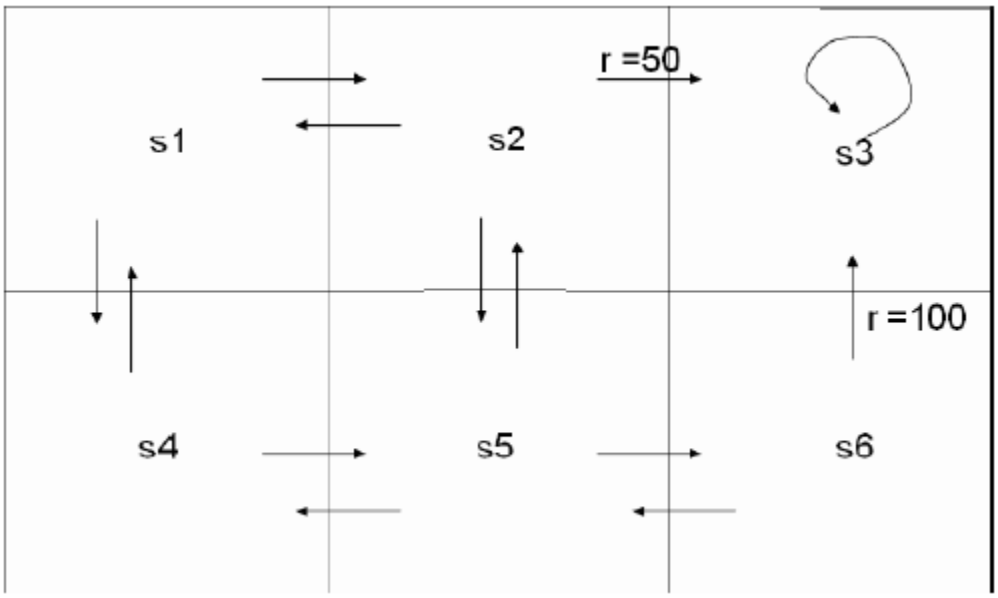
2.5.στ Project on the first 2 principal components

```
# calculate the explained variance by each principal component
explained_variance = eigenvalues / sum(eigenvalues)

# calculate the cumulative explained variance
cumulative_explained_variance = np.cumsum(explained_variance)

# find the number of components needed to explain 95% of the variance
num_components = np.where(cumulative_explained_variance >= 0.95)[0][0] + 1
print(num_components, " components explain about ",
"{:.4f}".format(cumulative_explained_variance[1]), "% of the variance", sep =
"")
```

```
2 components explain about 0.9976% of the variance
```

Το παραπάνω notebook μπορεί να βρεθεί και εδώ.

Άσκηση 2.6



Ερώτημα α

Δεδομένου ότι η μετακίνηση γίνεται με ντετερμινιστικό τρόπο, μπορεί να γίνει η εξής απλοποίηση:

$$v_\pi(s) = \sum_a \pi(a,s) \sum_{r,s'} p(r,s'|s,a)[r + \gamma v_\pi(s')] \Rightarrow v_\pi(s) = r + \gamma v_\pi(s')$$

Μέσω δυναμικού προγραμματισμού, μπορούμε να υπολογίσουμε τη $v^*(s)$:

$$v_{i+1}(s) = \max_{s'}\{r + \gamma v_i(s')\}$$

Τα βήματα που ακολουθούμε είναι τα εξής:

$v_0(s_1) = 0$
$v_0(s_2) = 0$
$v_0(s_3) = 0$
$v_0(s_4) = 0$
$v_0(s_5) = 0$
$v_0(s_6) = 0$

$v_1(s_1) = 0$
$v_1(s_2) = \max\{(0 + 0.8 \cdot 0), (50 + 0.8 \cdot 0), (0 + 0.8 \cdot 0)\} = 50$
$v_1(s_3) = 0$
$v_1(s_4) = 0$
$v_1(s_5) = \max\{(0 + 0.8 \cdot 50), (0 + 0.8 \cdot 0), (0 + 0.8 \cdot 0)\} = 40$
$v_1(s_6) = \max\{(100 + 0.8 \cdot 0), (0 + 0.8 \cdot 40)\} = 100$

$v_2(s_1) = \max\{(0 + 0.8 \cdot 50), (0 + 0.8 \cdot 0)\} = 40$
$v_2(s_2) = \max\{(0 + 0.8 \cdot 40), (50 + 0.8 \cdot 0), (0 + 0.8 \cdot 40)\} = 50$
$v_2(s_3) = 0$
$v_2(s_4) = \max\{(0 + 0.8 \cdot 40), (0 + 0.8 \cdot 40)\} = 32$
$v_2(s_5) = \max\{(0 + 0.8 \cdot 50), (0 + 0.8 \cdot 32), (0 + 0.8 \cdot 100)\} = 80$
$v_2(s_6) = \max\{(100 + 0.8 \cdot 0), (0 + 0.8 \cdot 80)\} = 100$

$v_3(s_1) = \max\{(0 + 0.8 \cdot 50), (0 + 0.8 \cdot 32)\} = 40$
$v_3(s_2) = \max\{(0 + 0.8 \cdot 40), (50 + 0.8 \cdot 0), (80 + 0.8 \cdot 40)\} = 64$
$v_3(s_3) = 0$
$v_3(s_4) = \max\{(0 + 0.8 \cdot 40), (0 + 0.8 \cdot 80)\} = 64$
$v_3(s_5) = \max\{(0 + 0.8 \cdot 64), (0 + 0.8 \cdot 64), (0 + 0.8 \cdot 100)\} = 80$
$v_3(s_6) = \max\{(0 + 0.8 \cdot 80), (100 + 0.8 \cdot 0)\} = 100$

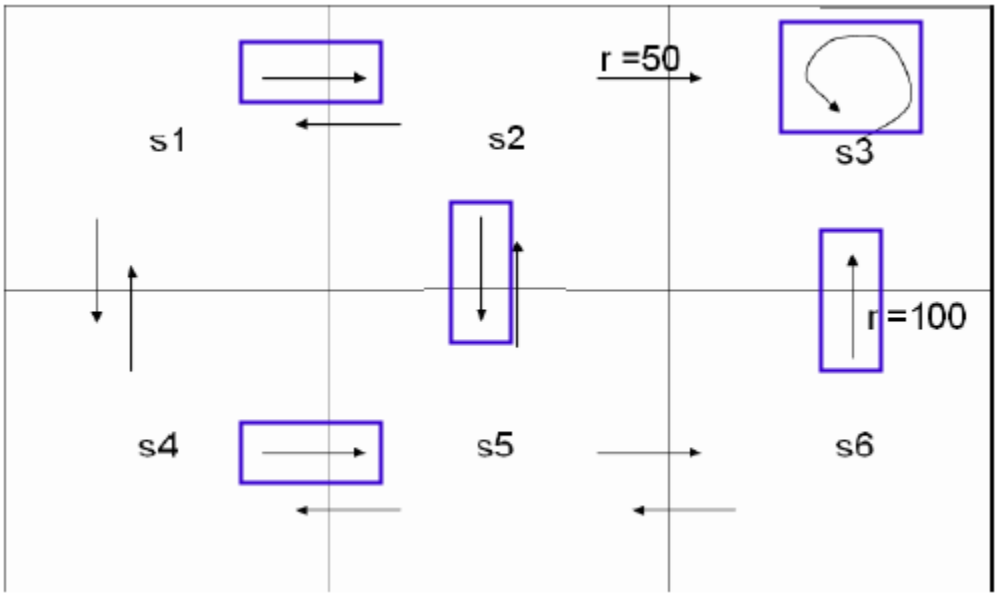
$v_4(s_1) = \max\{(0 + 0.8 \cdot 64), (0 + 0.8 \cdot 64)\} = 51.2$
$v_4(s_2) = \max\{(0 + 0.8 \cdot 51.2), (50 + 0.8 \cdot 0), (80 + 0.8 \cdot 0)\} = 64$
$v_4(s_3) = 0$
$v_4(s_4) = \max\{(0 + 0.8 \cdot 51.2), (0 + 0.8 \cdot 80)\} = 64$
$v_4(s_5) = \max\{(0 + 0.8 \cdot 64), (0 + 0.8 \cdot 64), (0 + 0.8 \cdot 100)\} = 80$
$v_4(s_6) = \max\{(0 + 0.8 \cdot 80), (100 + 0.8 \cdot 0)\} = 100$

Στο επόμενο βήμα οι τιμές $v_5(s_i)$ είναι ίδιες με τις $v_4(s_i)$. Άρα ο αλγόριθμος τερμάτησε.

Έχουμε: $v^*(s_1) = 51.2, v^*(s_2) = 64, v^*(s_3) = 0, v^*(s_4) = 64, v^*(s_5) = 80, v^*(s_6) = 100$

Ερώτημα β

Σημειώνουμε με πλαίσια τις ενέργειες που αντιστοιχούν σε βέλτιστη πολιτική:



Ισοπαλία υπήρξε κατά την επιλογή ενέργειας από την s_1: $v^*(s_2) = v^*(s_4) = 64$. Επιλέξαμε τυχαία την πολιτική που μας πηγαίνει στην s_2.

Ερώτημα γ

Το βέλτιστο μονοπάτι από οποιαδήποτε κατάσταση στην s_3 έχει μήκος το πολύ 4. Για κάθε κατάσταση γνωρίζουμε τη βέλτιστη πολιτική ύστερα από τόσες επαναλήψεις όσες είναι οι συνεχόμενες βέλτιστες πολιτικές ως το s_3, δηλαδή το μήκος του βέλτιστου μονοπατιού. Συνδυάζοντας τις δύο προτάσεις, χρειάζονται το πολύ 4 επαναλήψεις.

Ερώτημα δ

Είναι δυνατό. Πολλαπλασιάζοντας όλες τις ανταμοιβές με τον ίδιο αριθμό, κλιμακώνοντάς τες δηλαδή, δεν αλλάζει τίποτα στον τρόπο που το ρομπότ θα αποφασίσει να κινηθεί στο διδιάστατο χώρο. Έστω ότι τις δεκαπλασιάζουμε:

$$v^*(s_1) = 512, v^*(s_2) = 640, v^*(s_3) = 0, v^*(s_4) = 640, v^*(s_5) = 800, v^*(s_6) = 1000$$

Ενώ μεταβλήθηκαν, εύκολα βλέπουμε πως η βέλτιστη πολιτική π^* παραμένει ίδια.