Αναγνώριση Προτύπων - 2η Σειρά Ασκήσεων

Ακαδημαϊκό Έτος 2021-2022

Ονοματεπώνυμο: Χρήστος Δημόπουλος

Αρ. Μητρώου: 031 17 037

e-mail: chrisdim1999@gmail.com

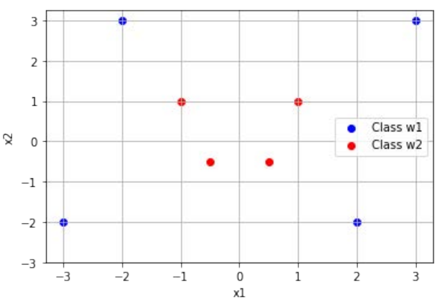
## Άσκηση 2.1: (SVM)

Μας δίνονται $N = 8$ διανύσματα χαρακτηριστικών $\tilde{x}_n = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ που προέρχονται από δύο κλάσεις $w_1$ και $w_2$:

$$w_1: \left\{ \begin{pmatrix} 3 \\ 3 \end{pmatrix}, \begin{pmatrix} 2 \\ -2 \end{pmatrix}, \begin{pmatrix} -3 \\ -2 \end{pmatrix}, \begin{pmatrix} -2 \\ 3 \end{pmatrix} \right\}, \quad z_1 = z_2 = z_3 = z_4 = -1$$

$$w_2: \left\{ \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 0.5 \\ -0.5 \end{pmatrix}, \begin{pmatrix} -0.5 \\ -0.5 \end{pmatrix}, \begin{pmatrix} -1 \\ 1 \end{pmatrix} \right\}, \quad z_5 = z_6 = z_7 = z_8 = +1$$

a) Αρχικά, σχεδιάζοντας τα παραπάνω σημεία σε ένα γράφημα, παρατηρούμε ότι οι δύο κλάσεις $w_1$ και $w_2$ δεν είναι γραμμικά διαχωρίσιμες.




Γι' αυτόν τον λόγο, μετατρέπουμε τα διανύσματα $\tilde{x}_n$ σε έναν χώρο υψηλότερων διαστάσεων, $y_n = \varphi(\tilde{x}_n)$, χρησιμοποιώντας την εξής μορφή φ-functions 2ης τάξης:

$$\varphi(x_1, x_2) = \left[ 1,\ x_1,\ x_2,\ \frac{x_1^2 + x_2^2 - 5}{3} \right]^T$$

Το διάνυσμα βαρών $W$ είναι, επίσης, επαυξημένο κατά $w_0$ ($W = [w_0\ \tilde{w}]^T$) και καλούμαστε να ελαχιστοποιήσουμε το μήκος του, υπό τους περιορισμούς $z_n W^T y_n \geq 1$, β) Κατά την εκπαίδευση ενός SVM ταξινομητή, θέλουμε να ελαχιστοποιήσουμε το μέτρο $\|w\|$, υπό τους περιορισμούς $z_n w^T y_n \geq 1$, $n=1,\dots,8$. Για τον σκοπό αυτό, ορίζουμε το συναρτησιακό:

$$\mathcal{L}(w,a) = \frac{1}{2}\|w\|^2 - \sum_{k=1}^{8} a_k [z_k w^T y_k - 1]$$

το οποίο θέλουμε να ελαχιστοποιήσουμε ως προς το διάνυσμα βαρών $w$ και να μεγιστοποιήσουμε ως προς τους άγνωστους πολλαπλασιαστές $a_k \geq 0$. Το παραπάνω πρόβλημα είναι ισοδύναμο με τη μεγιστοποίηση ως προς τα $a$ της ποσότητας:

$$\mathcal{L}(a) = \sum_{k=1}^{8} a_k - \frac{1}{2}\sum_{i=1}^{8}\sum_{j=1}^{8} a_i a_j z_i z_j y_i^T y_j$$

υπό τους περιορισμούς:

$$\sum_{k=1}^{8} z_k a_k = 0 \quad \text{και} \quad a_k \geq 0, \; k=1,\dots,8 \qquad (\mathrm{I}).$$

Το άνωθι πρόβλημα βελτιστοποίησης, μπορεί να επιλυθεί με πολλαπλασιαστές Lagrange, θέτοντας το συναρτησιακό:

$$Q(a,\lambda) = \mathcal{L}(a) + \lambda \sum_{k=1}^{8} z_k a_k$$

και εξισώνοντας τις παραγώγους του ως προς $a_i$, $i=1,\dots,8$ και $\lambda$ ίσες με μηδέν.

Στην πράξη, χρησιμοποιώντας πακέτα βελτιστοποίησης της python βρίσκουμε:

$$\boxed{\begin{array}{llll} a_1 = 0, & a_2 = 0.1536, & a_3 = 0, & a_4 = 0.0934 \\ a_5 = 0.1665, & a_6 = 0.017, & a_7 = 0.0175, & a_8 = 0.0459 \end{array}}$$

Η παραπάνω λύση είναι δεκτή, αφού $a_i \geq 0$, $i=1,\dots,8$ και επιπλέον ισχύει $\sum_{i=1}^{8} z_i a_i = 0$.

γ) Αρχικά, προσδιορίζουμε το μη-επαυξημένο διάνυσμα βαρών ως:

$$\tilde{w}_r = \sum_{k=1}^{8} a_k z_k \tilde{y}_{k,r} \Rightarrow \tilde{w} = [-5\times10^{-5}, \ 0.2221, -0.6668]$$

Για την εύρεση του bias $w_0$, επιλέγουμε ένα οποιοδήποτε δείγμα που λειτουργεί ως support vector (δηλαδή $a_i > 0$). Στην προκειμένη, χρησιμοποιούμε το διάνυσμα $y_2 = [1 \ 2 \ -2 \ 1]^T$, για το οποίο βρήκαμε $a_2 = 0.1536 > 0$:

$$z_2 [w_0 \ \ \tilde{w}] \begin{bmatrix} 1 \\ \tilde{y}_2 \end{bmatrix} - 1 = 0$$

$$\Rightarrow z_2 \cdot (w_0 + \tilde{w} \cdot \tilde{y}_2) - 1 = 0 \Rightarrow \boxed{w_0 = \frac{1}{z_2} - \tilde{w}\tilde{y}_2}$$

Απ' όπου βρίσκουμε $w_0 = 0.1112$.

Τελικώς $\boxed{w = [0.1112, -5\times10^{-5}, \ 0.2221, -0.6668]^T}$

Ελέγχουμε τους περιορισμούς:

- $z_1 w^T y_1 = 2.11 \geq 1$
- $z_2 w^T y_2 = 1 \geq 1$
- $z_3 w^T y_3 = 2.11 \geq 1$
- $z_4 w^T y_4 \simeq 1 \geq 1$
- $z_5 w^T y_5 \simeq 1 \geq 1$
- $z_6 w^T y_6 \simeq 1 \geq 1$
- $z_7 w^T y_7 \simeq 1 \geq 1$
- $z_8 w^T y_8 \simeq 1 \geq 1$

δ) Το περιθώριο του ταξινομητή δίνεται ως:

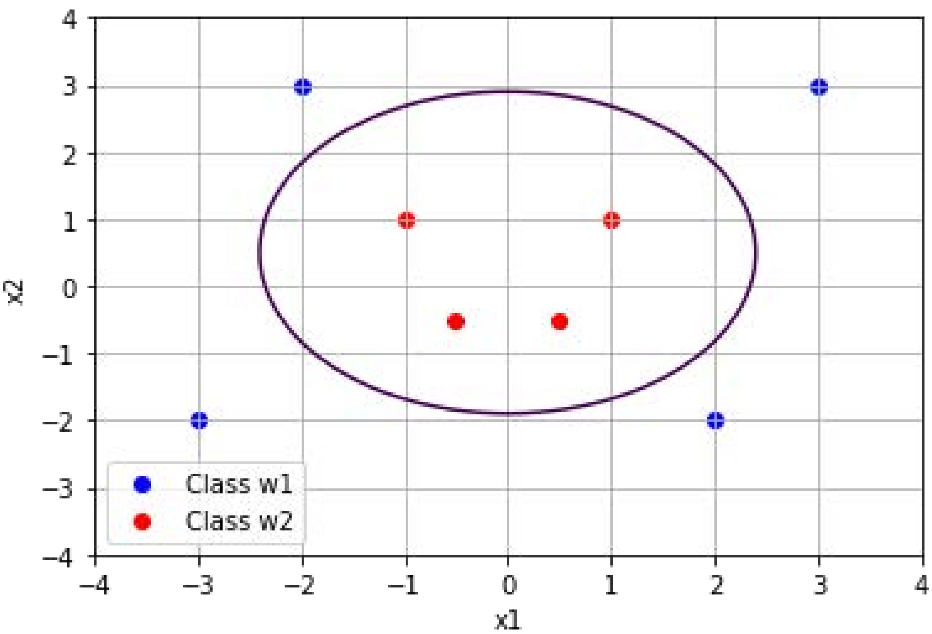
$$B = \frac{1}{\|\tilde{w}\|} \Rightarrow \boxed{B = 1.423}$$

ε) Η συνάρτηση διαχωρισμού:

$$g(x_1,x_2)=0 \Rightarrow w^T\varphi(x_1,x_2)=0 \Rightarrow \begin{bmatrix} w_0 & \tilde{w} \end{bmatrix} \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \frac{x_1^2+x_2^2-5}{3} \end{bmatrix} = 0$$

$$\Rightarrow w_0 + w_1 x_1 + w_2 x_2 + w_3 \cdot \frac{x_1^2+x_2^2-5}{3} = 0$$

απεικονίζεται παρακάτω μαζί με τα 8 αρχικά σημεία-δείγματα:




στ) Support Vectors είναι τα διανύσματα $y_i$ για τα οποία βρήκαμε ότι ισχύει αυστηρώς $a_i > 0$, δηλαδή τα διανύσματα $y_2, y_4, y_5, y_6, y_7, y_8$ (όλα πλην των $y_1, y_3$).

ζ) Τα νέα δείγματα $x' = \begin{pmatrix} \sqrt{2} \\ \sqrt{2} \end{pmatrix}$ και $x'' = \begin{pmatrix} -2 \\ 3 \end{pmatrix}$ ταξινομούνται στις κατηγορίες $w_2$ και $w_1$ αντίστοιχα, καθώς:

- $w^T \cdot \begin{bmatrix} 1 \\ \sqrt{2} \\ \sqrt{2} \\ \frac{2+2-5}{3} \end{bmatrix} = 0.6475 > 0 \rightsquigarrow$ Class $w_2$.

- $w^T \cdot \begin{bmatrix} 1 \\ -2 \\ 3 \\ \frac{4+9-5}{3} \end{bmatrix} = -1 < 0 \rightsquigarrow$ Class $w_1$.

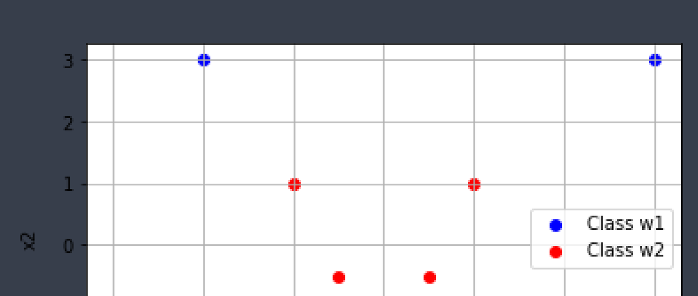
# Άσκηση 2.1: SVM

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import minimize
```

## Ερώτημα (α)

Αρχικά, σχεδιάζουμε σε γράφημα τα σημεία εισόδου και παρατηρούμε ότι δεν είναι γραμμικά διαχωρίσιμα. Για τον λόγο αυτό, μετατρέπουμε τα διανύσματα $\tilde{x}_n$ σε έναν χώρο υψηλότερων διαστάσεων, $y_n = ϕ(\tilde{x}_n)$, χρησιμοποιώντας την εξής μορφή ϕ-functions 2ης τάξης: $$ϕ(x_1, x_2) = \Big[1 \;\; \tilde{y}\Big] = \Big[1\;\; x_1\;\; x_2\;\; \frac{x_1^2+x_2^2-5}{3}\Big]^T$$

```python
# Plot Points of two Classes
W1 = [(3,3),(2,-2),(-3,-2),(-2,3)]          # w1 class
W2 = [(1,1),(0.5,-0.5),(-0.5,-0.5),(-1,1)]  # w2 class

def coords(data):
    xs, ys = [], []
    for point in data:
        xs.append(point[0])
        ys.append(point[1])
    return xs, ys

W1_x, W1_y = coords(W1)
plt.figure()
plt.grid()
plt.scatter(W1_x, W1_y, color='b', label = 'Class w1')

W2_x, W2_y = coords(W2)
plt.scatter(W2_x, W2_y, color='r', label = 'Class w2')

N = 3
plt.xticks(np.arange(-N, N+1, 1))
plt.yticks(np.arange(-N, N+1, 1))
plt.xlabel('x1')
plt.ylabel('x2')
plt.legend()
plt.show()
```




```python
# Map to Higher Dimensionality
def fi_function(x):
  return [1, x[0], x[1], ((x[0])**2 + (x[1])**2 - 5)/3]

new_w1 = []
new_w2 = []
print('W1 Class')
for sample in W1:
  new_w1.append(np.array(fi_function(sample)))
  print(fi_function(sample))

print('\nW2 Class')
for sample in W2:
  new_w2.append(np.array(fi_function(sample)))
  print(fi_function(sample))
```

```
W1 Class
[1, 3, 3, 4.333333333333333]
[1, 2, -2, 1.0]
[1, -3, -2, 2.6666666666666665]
[1, -2, 3, 2.6666666666666665]

W2 Class
[1, 1, 1, -1.0]
[1, 0.5, -0.5, -1.5]
[1, -0.5, -0.5, -1.5]
[1, -1, 1, -1.0]
```

## Ερώτημα (β)

Ακολούθως, με βάση το υποκεφάλαιο 5.11.1 [2], [5, SVM] προβαίνουμε στη μεγιστοποίηση του παρακάτω συναρτησιακού:
$$L({\alpha}) = \sum_{k=1}^{N}\alpha_i - \frac{1}{2}\sum_{k,j}^{N}\alpha_k\alpha_jz_kz_j\mathbf{y}_j^t\mathbf{y}_k$$

υπό τους περιοσισμούς:

$$\sum_{k=1}^{N}z_k\alpha_k = 0 \qquad \text{και} \qquad \alpha_k \geq 0, k = 1,\dots,N$$

```python
# Collec all samples in the new dimensional space
y = new_w1 + new_w2
z = [-1 for i in range(len(new_w1))] + [1 for i in range(len(new_w2))]
```

```python
def L(a): # Cost Function
  result = 0
  for i in range(len(z)):
    result += a[i]

  for i in range(len(z)):
    for j in range(len(z)):
      result -= 0.5*a[i]*a[j]*z[i]*z[j]*np.dot(y[i],y[j])

  return -result
```

```python
# Constrains on a_i
cons = ({'type': 'eq', 'fun': lambda a:  np.dot(z,a)},
        {'type': 'ineq', 'fun': lambda a: a[0]},
        {'type': 'ineq', 'fun': lambda a: a[1]},
        {'type': 'ineq', 'fun': lambda a: a[2]},
        {'type': 'ineq', 'fun': lambda a: a[3]},
        {'type': 'ineq', 'fun': lambda a: a[4]},
        {'type': 'ineq', 'fun': lambda a: a[5]},
        {'type': 'ineq', 'fun': lambda a: a[6]},
        {'type': 'ineq', 'fun': lambda a: a[7]}
        )
```

```python
res = minimize(L, (0,0,0,0,0,0,0,0),
               constraints=cons)
```

```python
res
```

```
     fun: -0.2469135792544758
     jac: array([ 1.22247892,  0.11110054,  1.22219016,  0.11125482, -0.11112959,
       -0.11106048, -0.11104704, -0.11110269])
 message: 'Optimization terminated successfully.'
    nfev: 63
     nit: 6
    njev: 6
  status: 0
 success: True
       x: array([-1.05270438e-17,  1.53551012e-01, -3.44416410e-16,  9.33680919e-02,
        1.66516910e-01,  1.69953160e-02,  1.74894126e-02,  4.59174647e-02])
```

```python
# Optimal a_i
a_opt = np.round(res.x,4)
print(a_opt) # a_i >= 0 for each i
```

```
[-0.      0.1536 -0.      0.0934  0.1665  0.017   0.0175  0.0459]
```

```python
0 == np.round(np.dot(z, a_opt),3) # constraints satisfied
```

```
True
```

Παρατηρούμε ότι ικανοποιούνται οι περιορισμοί του προβλήματος βελιστοποίησης.

## Ερώτημα (γ)

Το ζητούμενο διάνυσμα βαρών δίνεται από τη σχέση: $$\tilde{w} = \sum_{i=1}^{N}a_iz_i\tilde{y_i}$$

```python
weights = []
for j in range(1,len(y[1])):
  w = 0
  for i in range(len(z)):
      w += a_opt[i] * z[i] * y[i][j]
  weights.append(w)
```

```python
print(weights)
```

```
[-4.999999999997368e-05, 0.22214999999999996, -0.6668166666666666]
```

Ακολούθως, προσδιορίζουμε το bias $w_0$ χρησιμοποιώντας το support vector $y_2$ (για το οποίο ισχύει $α_2 > 0$) στην ακόλουθη σχέση: $$z_{2}(\tilde{w}\cdot \tilde{y_{2}} + w_0) - 1 = 0$$ από την οποία προκύπτει: $$w_0 = \frac{1}{z_{2}}-\tilde{w}\cdot \tilde{y}_{2}$$

```python
w0 = (1/z[1]) - np.dot(weights, y[1][1:]) # sample y2 is a Support Vector since a2 > 0
```

```python
# Total Weight Vector with bias
weights = [w0] + weights
print(weights)
```

```
[0.11121666666666652, -4.999999999997368e-05, 0.22214999999999996, -0.6668166666666666]
```

Έπειτα, για το διάνυσμα w που βρήκαμε, ελέγχουμε ότι ισχύουν οι προϋποθέσεις $z_n\mathbf{w}^{T}\mathbf{y}_n \geq 1 \;\;\; (n=1,\dots,N)$:

```python
for i in range(len(z)):
  print(z[i]*np.dot(weights, y[i])>=1)
```

```
True
True
True
True
True
True
True
True
```

## Ερώτημα (δ)

Το margin $\beta$ δίνεται από τον τύπο: $\beta = \frac{1}{|\tilde{w}|}$

```python
b = 1/(np.linalg.norm(weights[1:]))
print('Margin b =', b)
```

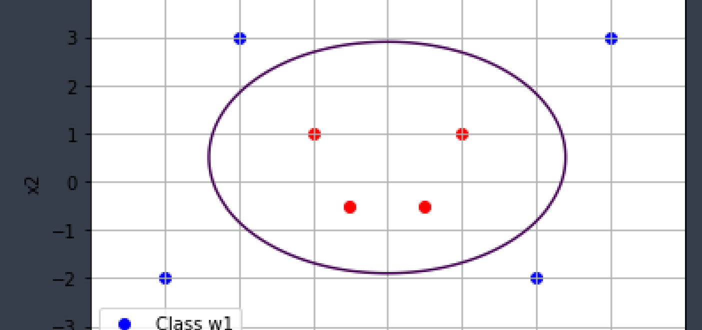
```
Margin b = 1.4227830509919526
```

## Ερώτημα (ε)

Βρίσκουμε τη συνάρτηση διαχωρισμού $g(x_1,x_2) = \mathbf{w}^T\phi(x_1,x_2)$ στον αρχικό χώρο $x_1-x_2$ και σχεδιάζουμε την καμπύλη $g(x_1,x_2)=0$ μαζί με τα 8 αρχικά σημεία.

```python
x1 = np.linspace(-4, 4, 500)
x2 = np.linspace(-4, 4, 500)
X, Y = np.meshgrid(x1, x2)
g = weights[0] + weights[1]*X + weights[2]*Y + weights[3]*(X**2+Y**2-5)/3
```

```python
plt.figure()
plt.grid()
plt.scatter(W1_x, W1_y, color='b', label = 'Class w1')
plt.scatter(W2_x, W2_y, color='r', label = 'Class w2')
plt.xlabel('x1')
plt.ylabel('x2')
plt.contour(X, Y, g, levels=[0])
plt.legend()
plt.show()
```




## Ερώτημα (στ)

Τα Support Vectors είναι τα σημεία με συντελεστή Lagrange αυστηρώς θετικό ($a_i > 0$), δηλαδή όλα εκτός από τα $x_1$ και $x_3$.

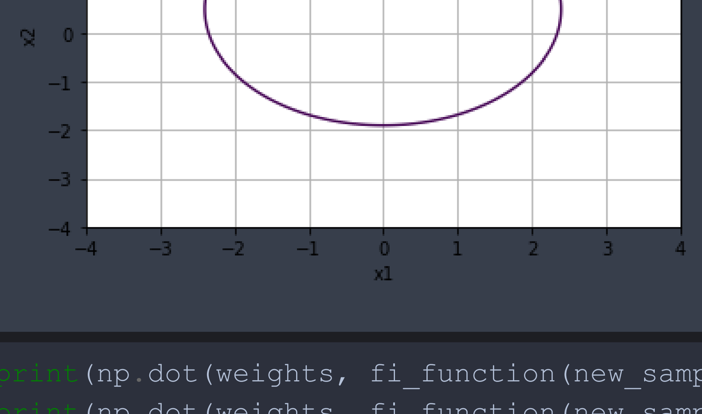
```python
for i in range(len(y)):
  if (a_opt[i]>0):
    print('Sample y'+str(i+1)+' is a Support Vector, since a'+str(i+1)+' = '+str(a_opt[i])+
' > 0')
  else:
    print('Sample y'+str(i+1)+' is NOT a Support Vector, since a'+str(i+1)+' = 0')
```

```
Sample y1 is NOT a Support Vector, since a1 = 0
Sample y2 is a Support Vector, since a2 = 0.1536 > 0
Sample y3 is NOT a Support Vector, since a3 = 0
Sample y4 is a Support Vector, since a4 = 0.0934 > 0
Sample y5 is a Support Vector, since a5 = 0.1665 > 0
Sample y6 is a Support Vector, since a6 = 0.017 > 0
Sample y7 is a Support Vector, since a7 = 0.0175 > 0
Sample y8 is a Support Vector, since a8 = 0.0459 > 0
```

## Ερώτημα (ζ)

```python
new_samples = [(np.sqrt(2), np.sqrt(2)), (-2, 3)]
```

```python
plt.figure()
plt.grid()
plt.scatter(new_samples[0][0], new_samples[0][1], color='r', label = 'Class w2')
plt.scatter(new_samples[1][0], new_samples[1][1], color='b', label = 'Class w1')
plt.xlabel('x1')
plt.ylabel('x2')
plt.contour(X, Y, g, levels=[0])
plt.legend()
plt.show()
```




```python
print(np.dot(weights, fi_function(new_samples[0]))) # > 0 --> Class w2
print(np.dot(weights, fi_function(new_samples[1]))) # < 0 --> Class w1
```

```
0.6475857210919529
-1.0004111111111111
```

Συνεπώς, ο SVM Classifier ταξινομεί το $\Big(\sqrt{2} \;\; \sqrt{2}\Big)^T$ στην κλάση $\omega_2$ και το $\Big(-2 \;\; 3\Big)^T$ στην κλάση $\omega_1$.

Άσκηση 2.2 : (HMM)

Δίνεται HMM $\lambda = (A, B, \pi)$ με τρεις καταστάσεις $1, 2, 3$ και δύο τύπους παρατηρή-
-σεων, H και T.

· Πίνακας Μεταβάσεων : $A = \begin{pmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{pmatrix}$

· Πιθανότητες παρατηρήσεων B :

| P(o\|q) | | | |
|---|---|---|---|
| o/q | 1 | 2 | 3 |
| H | 0.5 | 0.75 | 0.25 |
| T | 0.5 | 0.25 | 0.75 |

· a-priori πιθανότητες : $\pi_1 = \pi_2 = \pi_3 = 1/3$.

· Παρατηρούμενη ακολουθία $O = (H, T, H)$.

1) Forward Algorithm: Ορίζουμε $a_t(i) = P(o_0 o_1 \dots o_t \mid q_t = i; \lambda)$ (forward probability).

· Initialisation $(t=0)$: $a_0(i) = \pi_i b_i(O_0 = H)$

$$a_0(1) = \frac{1}{3} \cdot 0.5 = \frac{1}{6}$$

$$a_0(2) = \frac{1}{3} \cdot 0.75 = \frac{1}{4}$$

$$a_0(3) = \frac{1}{3} \cdot 0.25 = 1/12$$

· Iterative Update : $a_{t+1}(j) = \left[ \sum_{i=1}^{N} a_t(i) \cdot a_{ij} \right] \cdot b_j(o_{t+1})$

▫ Για $t=1$: $O_1 = T$

$$a_1(1) = \frac{1}{3} \cdot \frac{1}{3} \cdot (0.5 + 0.75 + 0.25) \cdot 0.5 = \frac{1}{3} \cdot \frac{1}{\cancel{3}} \cdot \frac{\cancel{3}}{2} \cdot \frac{1}{2} = \frac{1}{12}.$$

$$a_1(2) = \frac{1}{3} \cdot \frac{1}{3} (0.5 + 0.75 + 0.25) \cdot 0.25 = \frac{1}{3} \cdot \frac{1}{\cancel{3}} \cdot \frac{\cancel{3}}{2} \cdot \frac{1}{4} = \frac{1}{24}$$

$$a_1(3) = \frac{1}{3} \cdot \frac{1}{3} \cdot (0.5 + 0.75 + 0.25) \cdot 0.75 = \frac{1}{\cancel{3}} \cdot \frac{1}{\cancel{3}} \cdot \frac{\cancel{3}}{2} \cdot \frac{\cancel{3}}{4} = \frac{1}{8}.$$

- Για $t=2$: $O_2 = H$

$$a_2(1) = \frac{1}{3}\cdot\Big(\underbrace{\frac{1}{12} + \frac{1}{24} + \frac{1}{8}}_{6/24 = \frac{1}{4}}\Big)\cdot 0.5 = \frac{1}{3}\cdot\frac{1}{4}\cdot\frac{1}{2} = \frac{1}{24}$$

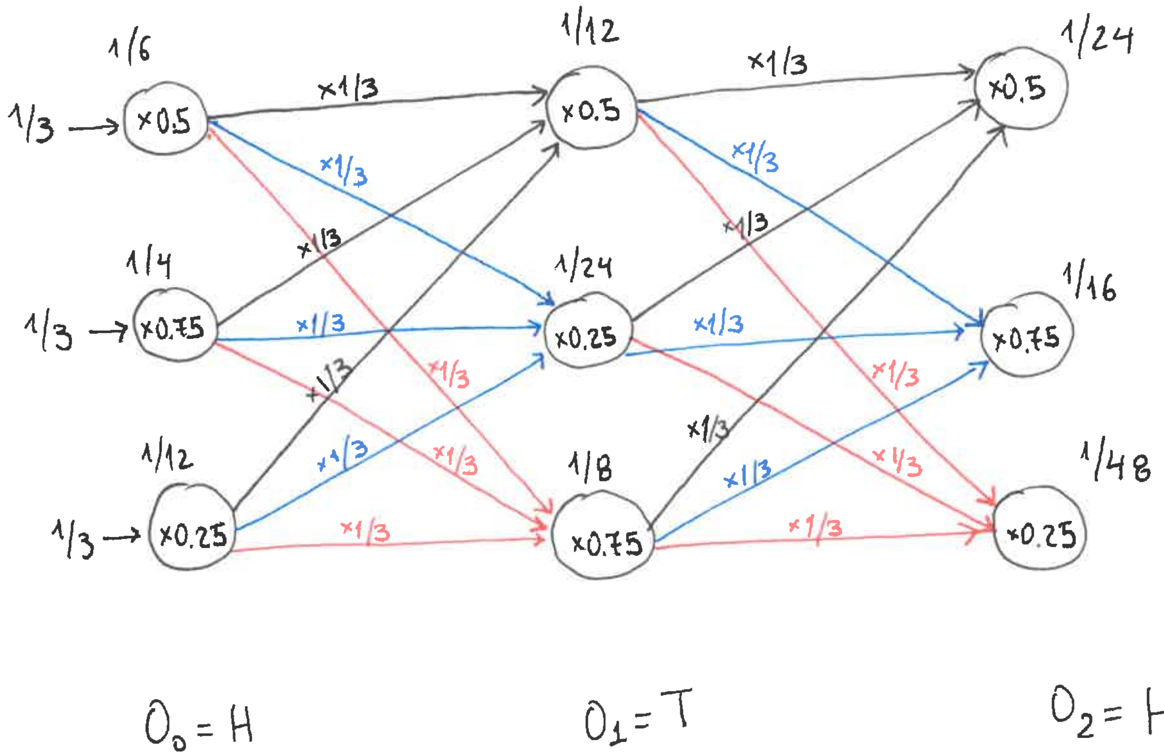
$$a_2(2) = \frac{1}{3}\cdot\Big(\frac{1}{4}\Big)\cdot 0.75 = \frac{1}{\cancel{3}}\cdot\frac{1}{4}\cdot\frac{\cancel{3}}{4} = \frac{1}{16}$$

$$a_2(3) = \frac{1}{3}\cdot\frac{1}{4}\cdot 0.25 = \frac{1}{3}\cdot\frac{1}{4}\cdot\frac{1}{4} = \frac{1}{48}$$

Τελικώς:

$$P(O = (H,T,H)\,|\,\lambda) = \sum_{i=1}^{3} a_2(3) = \frac{1}{24} + \frac{1}{16} + \frac{1}{48} = \frac{2+3+1}{48} = \frac{6}{48} = \frac{1}{8}$$

$$= \underline{0.125}.$$



2) Backward Algorithm: Ορίζουμε $\beta_t(i) = P(O_{t+1}, O_{t+2}, \ldots, O_T \mid q_t = i ; \lambda)$

· Initialization $(t = T = 3)$: $\beta_T(i) = 1 \quad , 1 \leq i \leq N = 3$

$$\beta_3(1) = \beta_3(2) = \beta_3(3) = 1$$

· Iterative Process: $\beta_t(i) = \sum_{j=1}^{N} a_{ij}\, b_j(O_{t+1})\, \beta_{t+1}(j) \, , \; 1 \leq i \leq N = 3 \, , 1 \leq t \leq T = 3$

□ Για $t = 2$: $O_3 = H$

$$\beta_2(1) = \frac{1}{3} \cdot 1 \cdot (0.5 + 0.75 + 0.25) = \frac{1}{\cancel{3}} \cdot \frac{\cancel{3}}{2} = \frac{1}{2}$$

$$\beta_2(2) = \frac{1}{3} \cdot 1 \left(0.5 + 0.75 + 0.25\right) = \frac{1}{2}$$

$$\beta_2(3) = \frac{1}{3} \cdot 1 \left(0.5 + 0.75 + 0.25\right) = \frac{1}{2}$$

□ Για $t = 1$: $O_2 = T$

$$\beta_1(1) = \frac{1}{3} \cdot \frac{1}{2} \cdot \left(0.5 + 0.25 + 0.75\right) = \frac{1}{\cancel{3}} \cdot \frac{1}{2} \cdot \frac{\cancel{3}}{2} = \frac{1}{4}$$

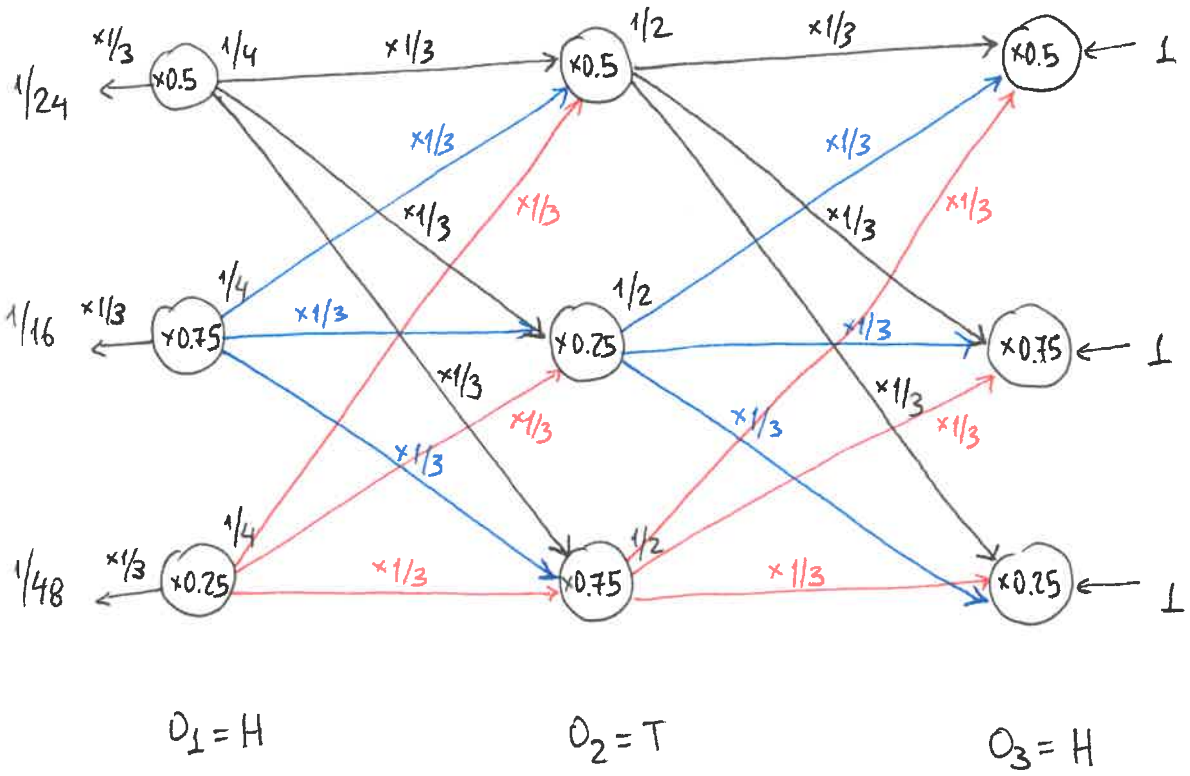
$$\beta_1(2) = \frac{1}{3} \cdot \frac{1}{2} \cdot \frac{3}{2} = \frac{1}{4}$$

$$\beta_1(3) = \frac{1}{3} \cdot \frac{1}{2} \cdot \frac{3}{2} = \frac{1}{4}$$

Τελικώς: $(O_1 = H)$

$$P(O = (H,T,H) \mid \lambda) = \sum_{j=1}^{3} \pi_j\, b_j(O_1)\, \beta_1(j) = \frac{1}{3} \times 0.5 \times \frac{1}{4} + \frac{1}{3} \times 0.75 \times \frac{1}{4} + \frac{1}{3} \times 0.25 \times \frac{1}{4}$$

$$= \frac{1}{\cancel{3}} \times \frac{1}{4} \times \frac{\cancel{3}}{2} = \frac{1}{8} = 0.125$$




3) <u>Αλγόριθμος Viterbi</u>:

· <u>Initialisation</u>: $\delta_0(i) = \pi_i b_i(O_0 = H)$

$\delta_0(1) = 1/6$ , $\delta_0(2) = \frac{1}{4}$ , $\delta_0(3) = \frac{1}{12}$

· <u>Iterative Process</u>: $\delta_{t+1}(j) = \max\limits_i \left(\delta_t(i) a_{ij}\right) \cdot b_j(O_{t+1})$

▫ <u>Για t = 1</u>: $O_1 = T$

$\delta_1(1) = \frac{1}{4} \cdot \frac{1}{3} \cdot 0.5 = \frac{1}{24}$ , $S_1(1) = 2$

$\delta_1(2) = \frac{1}{4} \cdot \frac{1}{3} \cdot 0.25 = \frac{1}{48}$ , $S_1(2) = 2$

$\delta_1(3) = \frac{1}{4} \cdot \frac{1}{3} \cdot 0.75 = \frac{1}{16}$ , $S_1(3) = 2$

▫ <u>Για t = 2</u>: $O_2 = H$

$\delta_2(1) = \frac{1}{16} \cdot \frac{1}{3} \cdot 0.5 = \frac{1}{96}$ , $S_2(1) = 3$

$\delta_2(2) = \frac{1}{16} \cdot \frac{1}{3} \cdot 0.75 = \frac{1}{64}$ , $S_2(2) = 3$

$\delta_2(3) = \frac{1}{16} \cdot \frac{1}{3} \cdot 0.25 = \frac{1}{192}$ , $S_2(3) = 3$

Συνεπώς, η πιο πιθανή ακολουθία καταστάσεων είναι η $2 \rightarrow 3 \rightarrow 2$ με πιθανότητα

$$\delta_2(2) = \frac{1}{64} = 0.015625.$$

4) Viterbi Training:

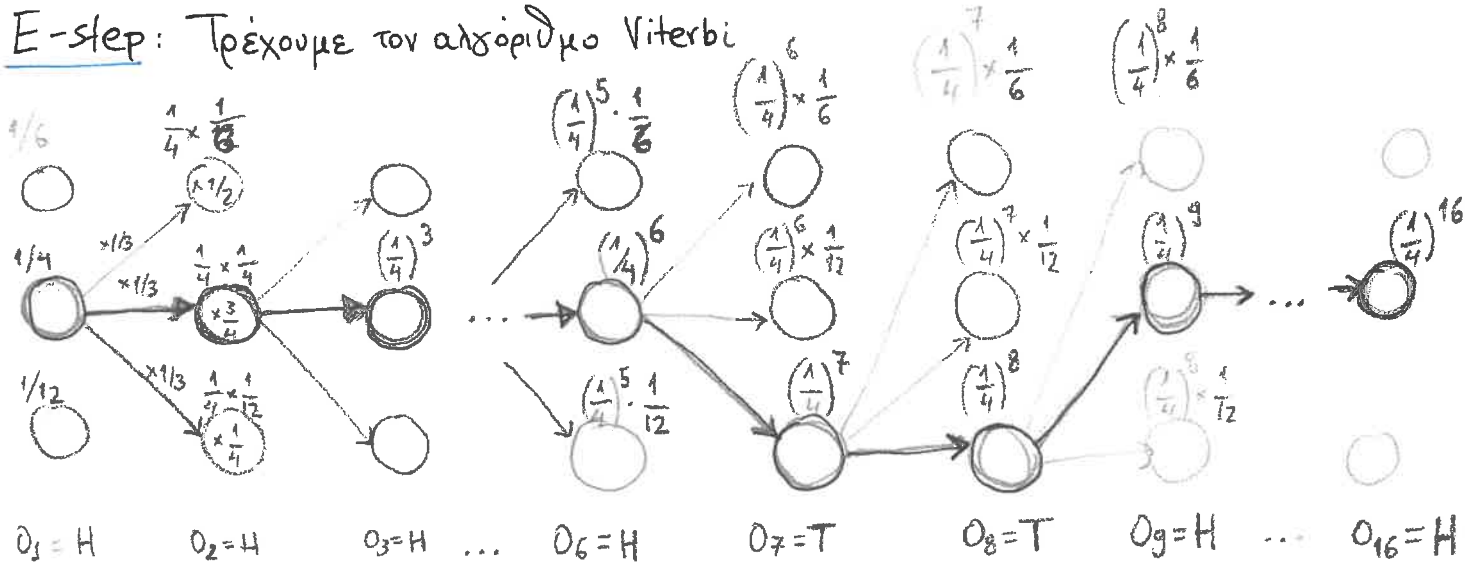
Παρατηρούμενη ακολουθία: $O =$ H H H H H H T T H H H H H H H H

• $1^{\eta}$ Επανάληψη:

Αρχικοποιούμε το μοντέλο HMM με βάση τα δεδομένα της άσκησης, δηλαδή:

$$\lambda^{(0)} = (A, B, \pi)$$

E-step: Τρέχουμε τον αλγόριθμο Viterbi




Η πιο πιθανή ακολουθία καταστάσεων είναι:

2 2 2 2 2 2 3 3 2 2 2 2 2 2 2 2

με πιθανότητα $(1/4)^{16}$.

M-step: Ενημερώνουμε τις ακόλουθες τιμές:

$$a_{22} = \frac{\#(2\rightarrow 2)}{\#(2\rightarrow *)} = \frac{12}{13}, \quad a_{23} = \frac{\#(2\rightarrow 3)}{\#(2\rightarrow *)} = \frac{1}{13}, \quad a_{21} = 0.$$

$$a_{33} = \frac{\#(3\rightarrow 3)}{\#(3\rightarrow *)} = \frac{1}{2}, \quad a_{32} = \frac{\#(3\rightarrow 2)}{\#(3\rightarrow *)} = \frac{1}{2}, \quad a_{31} = 0.$$

Επιπλέον:

$$P(O=H \mid q=2) = 1 \quad , \quad P(O=T \mid q=2) = 0$$

$$P(O=T \mid q=3) = 1 \quad , \quad P(O=H \mid q=3) = 0$$

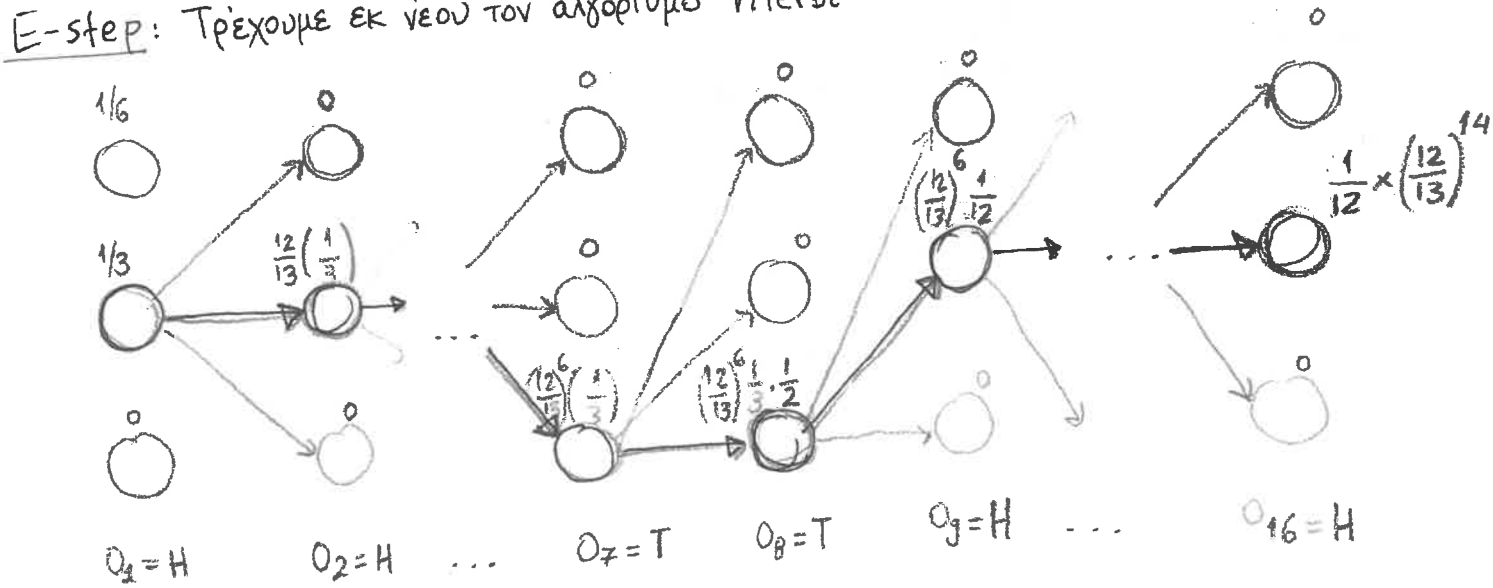
## 2η Επανάληψη:

Οι ανανεωμένες παράμετροι του HMM θα είναι $\lambda^{(1)} = (A^{(1)}, B^{(1)}, \pi)$, όπου:

$$A^{(1)} = \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 0 & \frac{12}{13} & \frac{1}{13} \\ 0 & 1/2 & 1/2 \end{bmatrix}$$ και $B^{(1)}$:

| P(O\|q) | | | |
|---|---|---|---|
| O\|q | 1 | 2 | 3 |
| H | 0.5 | 1 | 0 |
| T | 0.5 | 0 | 1 |

E-step: Τρέχουμε εκ νέου τον αλγόριθμο Viterbi




Η πιο πιθανή ακολουθία καταστάσεων είναι και πάλι η:

$$2 \to 2 \to 2 \to 2 \to 2 \to 2 \to 3 \to 3 \to 2 \to 2 \to 2 \to 2 \to 2 \to 2 \to 2 \to 2$$

με πιθανότητα $\cancel{\left(\frac{1}{3}\right)^{16}} \cdot \frac{1}{12} \times \left(\frac{12}{13}\right)^{14}$.

Ως εκ τούτου, οι παράμετροι του μοντέλου δεν ανανεώνονται και ο αλγόριθμος pseudo-EM έχει συγκλίνει με $\lambda = (A^{(1)}, B^{(1)}, \pi)$.

5) Ο forward-backward αλγόριθμος χρησιμοποιεί κάθε πιθανό μονοπάτι καταστάσεων για την εκπαίδευση του HMM, σε αντίθεση με τον pseudo-EM αλγόριθμο, ο οποίος χρησιμοποιεί το πιο πιθανό μονοπάτι καταστάσεων και μόνο. Το πιθανότερο μονοπάτι προσδιορίζεται με χρήση του αλγορίθμου Viterbi με αποδοτικό τρόπο, πολυπλοκότητας $O(N \times T)$, όπου N: number of hidden states,
T: number of ~~possible~~ observations

Συνεπώς:

- Ο forward-backward αλγόριθμος εκπαίδευσης είναι πιο αργός, αλλά δίνει πιο ακριβή αποτελέσματα στην εκπαίδευση του HMM, σε αντίθεση με τον pseudo-EM (Viterbi training), ο οποίος είναι πιο γρήγορος, αλλά λιγότερο ακριβής.

# Άσκηση 2.3: (CART)

| SATISFACTION | WORD ACCURACY | TASK_COMPLETION | TASK_DURATION |
|---|---|---|---|
| Y | 100 | Y | 3 |
| Y | 100 | Y | 2 |
| Y | 90 | N | 4 |
| N | 95 | N | 2 |
| N | 80 | Y | 5 |
| Y | 85 | Y | 5 |
| Y | 80 | Y | 1 |
| N | 85 | N | 3 |
| Y | 95 | N | 4 |

1) Οι δύο κατηγορίες του προβλήματος ταξινόμησης είναι οι εξής:

$\omega_1$: Τα δείγματα-χρήστες που έμειναν ευχαριστημένοι από το σύστημα διαλόγου, δηλαδή SATISFACTION = Y

$\omega_2$: τα δείγματα-χρήστες που δεν έμειναν ευχαριστημένοι από το σύστημα διαλόγου, δηλαδή SATISFACTION = N

Οι παράμετροι του δέντρου απόφασης είναι ερωτήσεις-κόμβοι της εξής μορφής:

TASK_COMPLETION = Y — Yes / No

WORD_ACCURACY > α — Yes / No

TASK_DURATION — Yes / No

όπου $\alpha, \beta \in \mathbb{R}$.

2) Ελέγχουμε τις ερωτήσεις του δέντρου απόφασης με βάση το ποιές δίνουν την ελάχιστη εντροπία:

- Ερώτηση Task_Completion = Y:

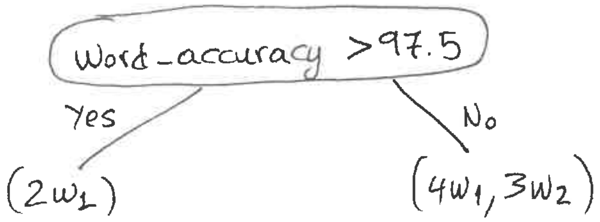

$$i_E(N_L) = -\left(\frac{4}{5}\log_2\left(\frac{4}{5}\right) + \frac{1}{5}\log_2\left(\frac{1}{5}\right)\right) = 0.722$$

$$i_E(N_R) = -(-0.5\log_2 0.5 + 0.5\log_2 0.5) = 1$$

$$i_E(\text{Task-completion} = Y) = \frac{5}{9}\, i_E(N_L) + \frac{4}{9}\, i_E(N_R) = 0.8455$$

- Ερώτηση Word_accuracy > a:

$\triangleright$ $a = \dfrac{95+100}{2} = 97.5$

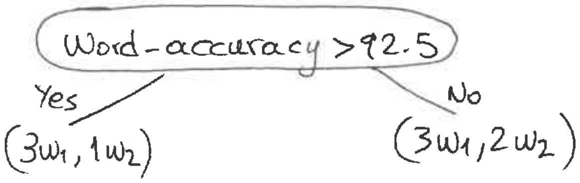



$$i_E(N_L) = 0$$

$$i_E(N_R) = -\left(\frac{4}{7}\log_2\left(\frac{4}{7}\right) + \frac{3}{7}\log_2\left(\frac{3}{7}\right)\right)$$

$$i_E(\text{Word-acc} > 97.5) = 0 + \frac{7}{9}\, i_E(N_R) = \underline{0.7663}$$

$\triangleright$ $a = \dfrac{90+95}{2} = 92.5$

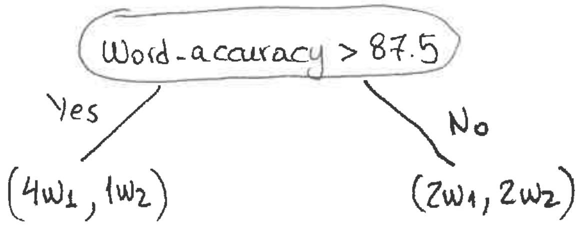



$$i_E(N_L) = -\left(\frac{3}{4}\log_2\left(\frac{3}{4}\right) + \frac{1}{4}\log_2\left(\frac{1}{4}\right)\right)$$

$$i_E(N_R) = -\left(\frac{3}{5}\log_2\left(\frac{3}{5}\right) + \frac{2}{5}\log_2\left(\frac{2}{5}\right)\right)$$

$$i_E(\text{Word-acc} > 92.5) = \frac{4}{9}\, i_E(N_L) + \frac{5}{9}\, i_E(N_R) = 0.9$$

$\triangleright$ $a = \dfrac{85+90}{2} = 87.5$




$$i_E(\text{Word-acc} > 87.5) = 0.8455$$

▹ $a = \frac{80+85}{2} = 82.5$

Word-accuracy > 82.5

Yes: $(5w_1, 2w_2)$ — No: $(1w_1, 1w_2)$

$i_\epsilon(N_L) = -\left(\frac{5}{7}\log_2\left(\frac{5}{7}\right) + \frac{2}{7}\log_2\left(\frac{2}{7}\right)\right)$

$i_\epsilon(N_R) = -\left(0.5\log_2(0.5) \times 2\right) = 1$

$i_\epsilon(\text{Word-acc} > 82.5) = \frac{7}{9} i_\epsilon(N_L) + \frac{2}{9} i_\epsilon(N_R) = 0.8935$

· Ερώτηση Task_Duration > a :

▹ $a = 4.5$

Task_Duration > 4.5

Yes: $(1w_1, 1w_2)$ — No: $(5w_1, 2w_2)$

$i_\epsilon(\text{Task\_Duration} > 4.5) = 0.8935$

▹ $a = 3.5$ :

Task_Duration > 3.5

Yes: $(3w_1, 1w_2)$ — No: $(3w_1, 2w_2)$

$i_\epsilon(\text{Task\_Duration} > 3.5) = 0.9$

▹ $a = 2.5$ :

Task_Duration > 2.5

Yes: $(4w_1, 2w_2)$ — No: $(2w_1, 1w_2)$

$i_\epsilon(\text{Task\_Duration} > 2.5) = 0.9183$

▹ $a = 1.5$ :

Task_Duration > 1.5

Yes: $(5w_1, 3w_2)$ — No: $(1w_1)$

$i_\epsilon(\text{Task\_Duration} > 1.5) = 0.8484$

Συνεπώς, η χαμηλότερη εντροπία δίνεται για την ερώτηση Word-accuracy > 97.5 ($i_\epsilon = 0.7663$), γι' αυτό και την επιλέγουμε ως τη ρίζα του δέντρου απόφασης.

Τώρα ακολουθούμε την ίδια διαδικασία, ώστε να βρούμε τη λιγότερο impure ερώτηση για τα δείγματα $(4w_1, 3w_2)$ του δεξιού κλαδιού (όπου Word-acc $\leq 97.5$).

· Ερώτηση Task_completion = Y :

Task_completion = Y

Yes: $(2w_1, 1w_2)$ No: $(2w_1, 2w_2)$

· $i_\epsilon(N_L) = -\left(\frac{2}{3}\log_2\left(\frac{2}{3}\right) + \frac{1}{3}\log_2\left(\frac{1}{3}\right)\right) = 0.918$

$i_\epsilon(N_R) = -2\left(0.5\log_2 0.5\right) = 1$

$i(\text{task\_compl.} = Y) = \frac{3}{7} i_\epsilon(N_L) + \frac{4}{7} i_\epsilon(N_R)$

$= 0.9648$

· Ερώτηση Word_accuracy > a :

▷ $a = \frac{90+95}{2} = 92.5$ :

Word_accuracy > 92.5

Yes: $(1w_1, 1w_2)$ No: $(3w_1, 2w_2)$

· $i_\epsilon(N_L) = 1$

$i_\epsilon(N_R) = -\left[\frac{3}{5}\log_2\left(\frac{3}{5}\right) + \frac{2}{5}\log_2\left(\frac{2}{5}\right)\right]$

$= 0.971$

$i_\epsilon(\text{word\_accuracy} > 92.5) = \frac{2}{7} i_\epsilon(N_L) + \frac{5}{7} i_\epsilon(N_R) = 0.9792$

▷ $a = \frac{85+90}{2} = 87.5$ :

Word_accuracy > 87.5

Yes: $(2w_1, 1w_2)$ No: $(2w_1, 2w_2)$

$i_\epsilon(\text{Word\_acc} > 87.5) = 0.9648$

▷ $a = \frac{80+85}{2} = 82.5$ :

Word_accuracy > 82.5

Yes: $(3w_1, 2w_2)$ No: $(1w_1, 1w_2)$

$i_\epsilon(\text{Word\_acc} > 82.5) = 0.971$

· Ερώτηση Task_Duration > a :

▷ $a = 4.5$ :

Task_Duration > 4.5

Yes: $(1w_1, 1w_2)$ No: $(3w_1, 2w_2)$

$i_\epsilon(\text{T.D.} > 4.5) = 0.971$

▷ $a = 3.5$ :

Task_Duration > 3.5

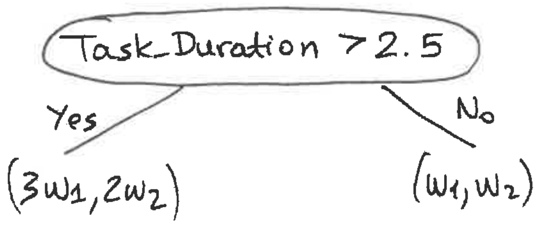
Yes: $(3w_1, 1w_2)$ No: $(1w_1, 2w_2)$

$i_\epsilon(N_L) = -\left(\frac{3}{4}\log\frac{3}{4} + \frac{1}{4}\log\frac{1}{4}\right) = 0.811$

$i_\epsilon(N_R) = -\left(\frac{1}{3}\log\frac{1}{3} + \frac{2}{3}\log\frac{2}{3}\right) = 0.918$

$i_\epsilon(\text{Task\_Duration} > 3.5) = \frac{4}{7} i_\epsilon(N_L) + \frac{3}{7} i_\epsilon(N_R) = 0.8568$

▹ $a = 2.5$ :

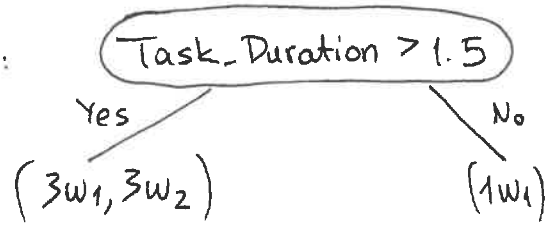



$i_E(N_L) = 0.971$

$i_E(N_R) = 1$

$i_E(T.D > 2.5) = \frac{5}{7} \times 0.971 + \frac{2}{7} = 0.9792$

▹ $a = 1.5$ :



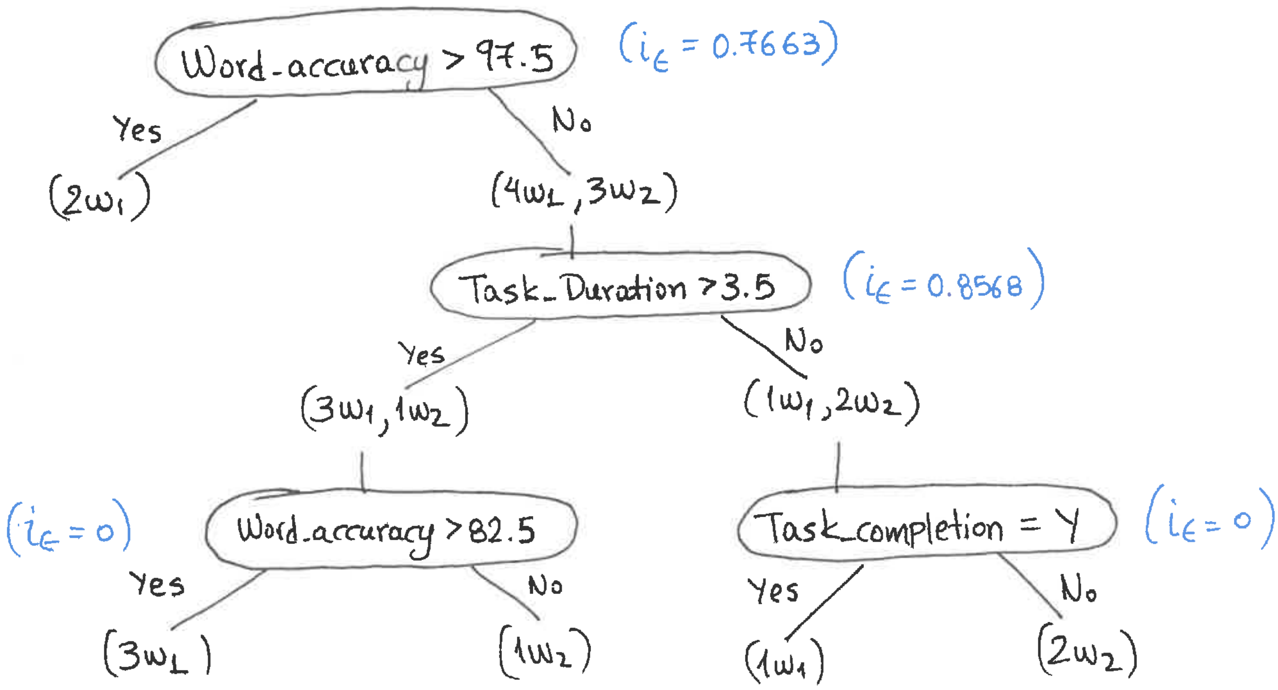

$i_E(N_L) = -2 \times 0.5 \log_2 0.5 = 1$

$i_E(N_R) = 0$

$i_E(T.D. > 1.5) = \frac{6}{7} \times 1 = 0.8571$

Συνεπώς, η χαμηλότερη εντροπία -άρα και το υψηλότερο purity- δίνεται από την ερώτηση Task_Duration > 3.5 ($i_E = 0.8568$), γι'αυτό και την εισάγουμε στο δέντρο απόφασης.

Έπειτα, για το αριστερό κλαδί μπορούμε να χρησιμοποιήσουμε την ερώτηση Word-accuracy > 82.5 που θα διαχωρίσει πλήρως τα δείγματα ($i_E = 0$), ενώ για το δεξί κλαδί, μπορούμε να διαχωρίσουμε πλήρως τα δείγματα με την ερώτηση Task_completion = Y ($i_E = 0$). Το δέντρο απόφασης, λοιπόν, θα είναι το εξής:



3) Για να ταξινομήσουμε τις ερωτήσεις του δέντρου ως προς τη σημαντικότητά τους, χρησιμοποιούμε τη μετρική Gini Importance [1] :

$$ni_j = w_j C_j - w_{left(j)} C_{left(j)} - w_{right(j)} C_{right(j)}$$

όπου $w_j$ : weighted number of samples reaching node $j$

$C_j$ : impurity value of node $j$

Επομένως:

- $ni\,(Word\_acc > 97.5) = 1 \cdot 0.7663 - \frac{2}{9} \cdot 0 - \frac{7}{9} \cdot 0.8568 = 0.0999$
- $ni\,(Task\_Duration > 3.5) = \frac{7}{9} \cdot 0.8568 - \frac{4}{7} \cdot 0 - \frac{3}{7} \cdot 0 = 0.6664$
- $ni\,(Word\_accuracy > 82.5) = ni\,(Task\_completion = Y) = 0$

Δηλαδή, η σειρά σημαντικότητας των ερωτήσεων είναι :

$(Task\_Duration > 3.5) > (Word\_acc > 97.5) > (Task\_Compl. = Y) = (Word\_acc > 82.5)$

Η σημαντικότητα των features, ορίζεται ως :

$$fi_i = \frac{\sum_{j:\,node\ j\ splits\ on\ feature\ i} ni_j}{\sum_{k \in all\ nodes} ni_k}$$

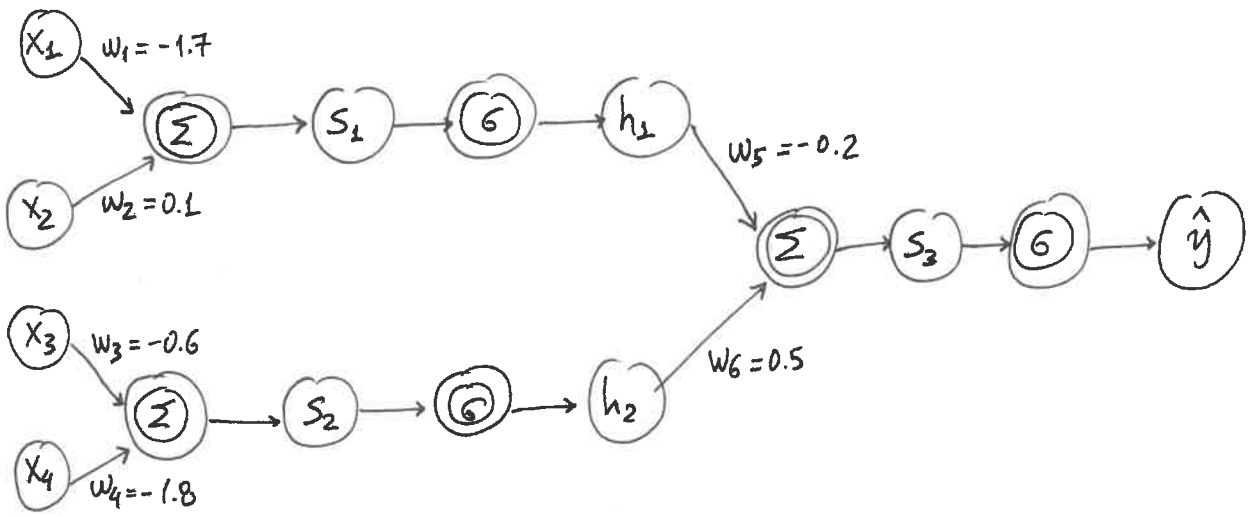
Άρα: $fi\,(Word\_acc) = \frac{0.0999 + 0}{0.0999 + 0.664 + 0 + 0} = \frac{0.0999}{0.7639} = 0.1307$

$fi\,(Task\_Duration) = \frac{0.664}{0.7639} = 0.8692$

$fi\,(Task\_completion) = 0$

Συνεπώς: $(Task\_Duration) > (Word\_accuracy) > (Task\_completion)$.

# Άσκηση 2.4 : (MLP Backpropagation)




Activation Function: $\sigma(x) = \frac{1}{1+e^{-x}}$ (logistic)

Loss Function: $L(y,\hat{y}) = \| y - \hat{y} \|_2^2$

Input Vector: $(x_1, x_2, x_3, x_4) = (-0.5, 1.4, 0.9, -3)$

True Label: $y = 0.5$

## Forward Pass:

- $s_1 = w_1 x_1 + w_2 x_2 \Rightarrow s_1 = 0.99$

  $h_1 = \frac{1}{1+e^{-0.99}} \Rightarrow h_1 = 0.729$

- $s_2 = w_3 x_3 + w_4 x_4 \Rightarrow s_2 = 4.86$

  $h_2 = \frac{1}{1+e^{-4.86}} \Rightarrow h_2 = 0.9923$

- $s_3 = w_5 h_1 + w_6 h_2 \Rightarrow s_3 = 0.35$

  $\hat{y} = \sigma(s_3) = \frac{1}{1+e^{-0.35}} \Rightarrow \hat{y} = 0.5867$

Συνεπώς: $L(y,\hat{y}) = \| y - \hat{y} \|^2 \Rightarrow \boxed{L(y,\hat{y}) = 0.007517}$

# Backward Pass:

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \tilde{y}}{\partial s_3} \cdot \frac{\partial s_3}{\partial h_1} \cdot \frac{\partial h_1}{\partial s_1} \cdot \frac{\partial s_1}{\partial w_1} \qquad \text{(chain rule)}$$

Όπου:

- $\frac{\partial L}{\partial \hat{y}} = +2\|y - \hat{y}\| = 0.086699 \times 2 = 0.1728$
- $\frac{\partial \tilde{y}}{\partial s_3} = \frac{\partial}{\partial s_3} \cdot \left[\frac{1}{1 + e^{-s_3}}\right] = \hat{y}(1 - \hat{y}) = 0.5867 \cdot (1 - 0.5867) = 0.242483$
- $\frac{\partial s_3}{\partial h_1} = \frac{\partial}{\partial h_1} \cdot \left[w_5 h_1 + w_6 h_2\right] = w_5 = -0.2$
- $\frac{\partial h_1}{\partial s_1} = \frac{\partial}{\partial s_1} \cdot \left[\frac{1}{1 + e^{-s_1}}\right] = h_1(1 - h_1) = 0.19752$
- $\frac{\partial s_1}{\partial w_1} = \frac{\partial}{\partial w_1} \cdot \left[w_1 x_1 + w_2 x_2\right] = x_1 = -0.5$

Συνολικά:

$$\frac{\partial L}{\partial w_1} = 2 * 0.086699 * 0.242483 * (-0.2) * (0.19752) * (-0.5)$$

$$\Rightarrow \boxed{\frac{\partial L}{\partial w_1} = 8,3048 \times 10^{-4}}$$

και το ανανεομένο βάρος θα είναι:

$$w_1' = w_1 + n \frac{\partial L}{\partial w_1}.$$

Άσκηση 2.5: (KLT-PCA)

1) Υποθέτουμε τυχαία διανύσματα $x \in \mathbb{C}^d$. Ο αντίστοιχος χώρος Hilbert είναι εφοδιασμένος με το εσωτερικό γινόμενο:

$$<x, y> = \mathcal{E}\{y^H \cdot x\}$$

Θέλουμε να ορίσουμε τον PCA μετασχηματισμό με βάση την ελαχιστοποίηση ενός λάθους προβολής. Για τον σκοπό αυτό, εισάγουμε ένα πλήρες ορθοκανονικό σύνολο από d-διάστατα διανύσματα βάσης $\{u_i\}$, με $i=1,2,\dots,d$ για τα οποία:

$$<u_i, u_j> = \mathcal{E}\{u_i^H u_j\} = \delta_{ij} = \begin{cases} 1, & \text{if } i=j \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

Καθένα από τα τυχαία διανύσματα $x_n \in \mathbb{C}^d$, λοιπόν, μπορεί να γραφεί ως γραμμικός συνδυασμός των διανυσμάτων βάσης:

$$x_n = \sum_{i=1}^{d} y_{ni} \cdot u_i \quad (2)$$

όπου οι συντελεστές $y_{ni}$ θα είναι διαφορετικοί για κάθε σημείο $x_n$. Κάθε τέτοιος συντελεστής, μάλιστα, θα είναι η προβολή του σημείου $x_n$ πάνω στο διάνυσμα βάσης $u_i$, δηλαδή το εσωτερικό γινόμενό τους:

$$y_{ni} = <x_n, u_i> = \mathcal{E}\{x_n^H u_i\}$$

Οπότε, η (2) μπορεί να γραφεί:

$$x_n = \sum_{i=1}^{d} <x_n, u_i> \cdot u_i \quad (3)$$

Στόχος μας είναι να προσεγγίσουμε τα διανύσματα $x_n$ με αναπαραστάσεις p-διαστάσεων, όπου $p<d$, δηλαδή να λάβουμε τις προβολές τους σε έναν υποχώρο χαμηλότερης διαστατικότητας. Χωρίς βλάβη της γενικότητας, ο p-διάστατος αυτός χώρος Hilbert, μπορεί να αναπαρασταθεί με τα p πρώτα διανύσματα βάσης. Η προσέγγιση του $x_n$ θα είναι:

$$\tilde{x}_n = \sum_{i=1}^{p} z_{ni} u_i + \sum_{i=p+1}^{d} b_i u_i \quad (4)$$

Ορίζουμε, στη συνέχεια, τη συνάρτηση σφάλματος λόγω μείωσης διαστατικότητας ως:

$$J = \mathcal{E}\{\|x - \tilde{x}\|^2\} \overset{(4)}{=} \mathcal{E}\{\|x - \sum_{i=1}^{p} z_i u_i - \sum_{i=p+1}^{d} b_i u_i\|^2\} \quad (5)$$

το οποίο θέλουμε να ελαχιστοποιήσουμε ως προς τα $z_i, b_i$. Χρησιμοποιώντας τις σχέσεις ορθοκανονικότητας (1) έχουμε:

$$\nabla_{z_i} J = 0 \Rightarrow \boxed{z_j = \langle x, u_j \rangle}, \quad j = 1, 2, \ldots, p$$

$$\nabla_{b_i} J = 0 \Rightarrow \boxed{b_j = \langle \bar{x}, u_j \rangle}, \quad j = p+1, \ldots, d$$

, όπου $\bar{x} = \mathcal{E}\{x\}$ η μέση τιμή των τυχαίων διανυσμάτων $x_n \in \mathbb{C}^d$.

Αντικαθιστώντας στη σχέση (5):

$$J = \mathcal{E}\{\|x - \sum_{i=1}^{p} \langle x, u_i \rangle u_i - \sum_{i=p+1}^{d} \langle \bar{x}, u_i \rangle u_i\|^2\} \overset{(3)}{=}$$

$$= \mathcal{E}\{\|\sum_{i=1}^{d} \langle x, u_i \rangle u_i - \sum_{i=1}^{p} \langle x, u_i \rangle u_i - \sum_{i=p+1}^{d} \langle \bar{x}, u_i \rangle u_i\|^2\}$$

$$= \mathcal{E}\{\|\sum_{i=p+1}^{d} (\langle x, u_i \rangle - \langle \bar{x}, u_i \rangle) \cdot u_i\|^2\} \quad (6)$$

λαμβάνοντας υπόψη ότι ο covariance matrix για τυχαία διανύσματα στον χώρο $\mathbb{C}^d$ είναι:

$$S = \mathcal{E}\{(x - \bar{x})(x - \bar{x})^H\}$$

η σχέση (6) μπορεί να γραφεί ως:

$$J = \sum_{i=p+1}^{d} u_i^H S u_i \quad (7)$$

Θέλουμε να ελαχιστοποιήσουμε την παραπάνω ποσότητα ως προς $u_i$, υπό τους περιορισμούς ορθοκανονικότητας (1). Για τον σκοπό αυτό, ορίζουμε το Lagrangian συναρτησιακό:

$$\tilde{J} = \sum_{i=p+1}^{d} u_i^H S u_i + \sum_{i=p+1}^{d} \lambda_i (-\langle u_i, u_i \rangle + 1) \quad (8)$$

Έχοντας ολοκληρώσει τον φορμαλισμό του προβλήματος για τυχαία διανύσματα, προβαίνουμε στη λύση με Μαθηματική Επαγωγή:

- Βάση Επαγωγής $(p=1)$:

Έστω, δηλαδή, ότι μετασχηματίζουμε τα διανύσματα $x_n \in \mathbb{C}^d$ σε 1-διάστατες αναπαραστάσεις.

Τότε, η (8) γράφεται:

$$\tilde{J}_1 = \sum_{i=2}^{d} u_i^H S u_i + \sum_{i=2}^{d} \lambda_i (-\langle u_i, u_i \rangle + 1)$$

Θέτοντας την παράγωγο ως προς $u_i$ ίση με μηδέν:

$$\cancel{2} S u_i - \lambda_i \cdot \cancel{2} u_i = 0 \Rightarrow S u_i = \lambda_i u_i$$

Δηλαδή, τα $u_i$ μπορούν να είναι οποιαδήποτε από τα ιδιοδιανύσματα του μητρώου $S$ με αντίστοιχη ιδιοτιμή $\lambda_i$. Αν θεωρήσουμε ότι ο πίνακας $S$ έχει ιδιοτιμές $\lambda_1 > \lambda_2 > \dots > \lambda_d$ με αντίστοιχα ιδιοδιανύσματα $e_1, e_2, \dots, e_d$, τότε για να ελαχιστοποιήσουμε τη συνάρτηση $J$, ορίζουμε ως $u_i$ τα ιδιοδιανύσματα του $S$ που αντιστοιχούν στις $d-1$ μικρότερες ιδιοτιμές. Με άλλα λόγια, προβάλλουμε ~~το~~ τα $x_n \in \mathbb{C}^d$ στη διεύθυνση του ιδιοδιανύσματος $e_1$ (με την μεγαλύτερη ιδιοτιμή $\lambda_1$) και το σφάλμα θα είναι:

$$\mathring{J}_1 = \sum_{i=2}^{d} e_i^H S e_i = \sum_{i=2}^{d} \lambda_i$$

- Επαγωγική Υπόθεση $(p := p) < d$

Υποθέτουμε ότι τα $u_i$ είναι τα ιδιοδιανύσματα $e_i$ του πίνακα $S$ που αντιστοιχούν στις $d-p$ μικρότερες ιδιοτιμές (ισοδύναμως προβάλλουμε τα $x_n$ στον υποχώρο που ορίζουν τα ιδιοδιανύσματα του $S$ που αντιστοιχούν στις $p$ μεγαλύτερες ιδιοτιμές). Τότε:

$$\mathring{J}_p = \sum_{i=p+1}^{d} e_i^H S e_i = \sum_{i=p+1}^{d} \lambda_i$$

□ Βήμα Επαγωγής $(p : p+1) < d$

Θέλουμε να ελαχιστοποιήσουμε το συναρτησιακό:

$$\tilde{J}_{p+1} = \sum_{i=p+2}^{d} u_i^H S u_i + \sum_{i=p+2}^{d} \lambda_i (1 - \langle u_i, u_i \rangle)$$

ως προς τα $u_i$. Παραγωγίζοντας ως προς $u_i$ και θέτοντας με μηδέν:

$$S u_i = \lambda u_i$$

Δηλαδή, τα $u_i$ είναι οποιαδήποτε από τα ιδιοδιανύσματα του $S$ με ιδιοτιμές $\lambda_i$. Αν αντικαταστήσουμε στην $J$, τότε προκύπτει:

$$J_{p+1} = \sum_{i=p+2}^{d} e_i^H S e_i = \sum_{i=p+2}^{d} \lambda_i \quad \text{(για κάποια ιδιοδιανύσματα } e_i\text{)}$$

Οπότε, για να ελαχιστοποιείται η $J$ επιλέγουμε ως $u_i$ τα ιδιοδιανύσματα $e_i$ του πίνακα $S$ που αντιστοιχούν στις $d-(p+1) = d-p-1$ μικρότερες ιδιοτιμές. Αν θεωρήσουμε, λοιπόν, ότι $\lambda_1 > \lambda_2 > \ldots \lambda_d$, θα ισχύει:

$$J_{p+1} = \sum_{i=p+2}^{d} \lambda_i = \left(\sum_{i=p+1}^{d} \lambda_i\right) - \lambda_{p+1}$$

$$\Rightarrow \boxed{J_{p+1} = J_p - \lambda_{p+1}}$$

2) Το συναρτησιακό προς ελαχιστοποίηση που ορίσαμε στο προηγούμενο ερώτημα ήταν :

$$\tilde{J} = \sum_{i=p+1}^{d} u_i^H S u_i + \sum_{i=p+1}^{d} \lambda_i (1 - \langle u_i, u_i \rangle) \quad (2.1)$$

με παραμέτρους τα $u_i$. Αν ορίσουμε τον πίνακα $U$, διαστάσεων $d \times (d-p)$, του οποίου οι στήλες είναι τα $u_i$ και τον πίνακα $H$ ως τον διαγώνιο πίνακα με τους Lagrangian συντελεστές $\lambda_i$, και επίσης μετονομάσουμε τον πίνακα συνδιασποράς $S$ σε $R$, τότε η παραπάνω εξίσωση γράφεται :

$$\tilde{J} = \text{trace}\{U^H R U\} + \text{trace}\{H(I - U^H U)\} \quad (2.2)$$

Δείξαμε ότι παραγωγίζοντας το $\tilde{J}$ ως προς $u_i$ και θέτοντας με μηδέν, προκύπτουν εξισώσεις $R u_i = \lambda_i u_i$. Βάζοντας τες όλες μαζί ως μια εξίσωση, προκύπτει :

$$\boxed{RU = UH} \quad (2.3)$$

η οποία είδαμε ότι δίνει ως λύση, ο πίνακας $U$ να έχει ως στήλες ιδιοδιανύσματα του μητρώου συνδιασποράς $R$ και το μητρώο $H$ να είναι διαγώνιος πίνακας με τις αντίστοιχες ιδιοτιμές.

Για να αποκτήσουμε μια γενική λύση, θεωρούμε ότι ο $H$ μπορεί να είναι ένας συμμετρικός Hermitian πίνακας. Τότε, σύμφωνα με το Φασματικό Θεώρημα [2] μπορεί να παραγοντοποιηθεί ως :

$$H = F D F^H$$

όπου $F$ unitrary matrix και $D$ diagonal matrix.

Λαμβάνοντας υπόψη ότι τα ιδιοδιανύσματα του $H$ είναι orthogonal, προβαίνουμε στη φασματική του διάσπαση :

$$H = (f_1, \dots, f_n) \begin{pmatrix} \lambda_1 & & \\ & \lambda_2 & \\ & & \ddots \\ & & & \lambda_n \end{pmatrix} \begin{pmatrix} f_1^H \\ \vdots \\ f_n^H \end{pmatrix} = (\lambda_1 f_1, \dots, \lambda_n f_n) \begin{pmatrix} f_1^H \\ \vdots \\ f_n^H \end{pmatrix} = \sum_{i=1}^{n} \lambda_i f_i f_i^H$$

με $n = d-p$. Σημειώνεται ότι τα $f_i f_i^H$ είναι rank-one πίνακες.

Αντικαθιστώντας, λοιπόν, στην (2.2) έχουμε:

$$\tilde{J} = \text{trace}\{U^H R U\} + \text{trace}\left\{\sum_{i=1}^{d-p} \lambda_i f_i f_i^H \cdot (I - U^H U)\right\}$$

Παραγωγίζοντας ως προς ένα εκ των $u_i$ και θέτοντας με μηδέν έχουμε:

$$\cancel{2} R u_i - \lambda_i f_i f_i^H \cancel{2} u_i = 0 \Rightarrow \boxed{R u_i = (f_i f_i^H) \cdot \lambda_i u_i} \quad (2.4)$$

πρόκειται, δηλαδή, για ένα πρόβλημα γενικευμένων ιδιοδιανυσμάτων. Όμως, ο
και τα $f_i$ orthogonal
πίνακας $f_i f_i^H$ είναι rank-one, οπότε η λύση της (2.4) θα είναι ίδια με τη λύση της εξίσωσης $R u_i = \lambda_i u_i$ (απλό ιδιοδιάνυσμα).
του μητρώου $R$

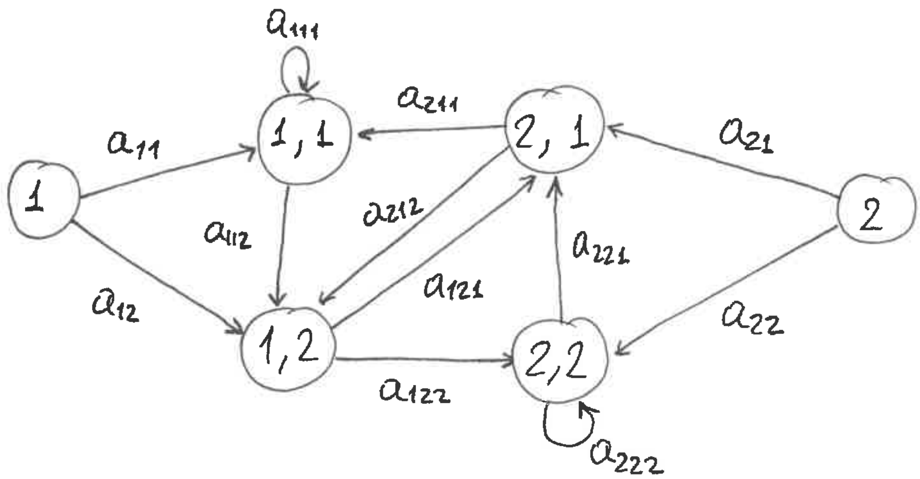
Συνεπώς, η γενική λύση είναι ίδια με την ειδική, σύμφωνα με την οποία τα $u_i$ είναι ιδιοδιανύσματα του μητρώου $R$.

## Άσκηση 2.6 : (Graphical Models)

Δίνεται Markov model με μνήμη 2 (MM-2) με δύο καταστάσεις 1 και 2. Ισχύει η σχέση ανεξαρτησίας:

$$p(q_t \mid q_{t-1}, q_{t-2}, q_{t-3}, \dots) = p(q_t \mid q_{t-1}, q_{t-2})$$

1) Bayesian Net:




2) Παρατηρούμε την ακόλουθη σειρά από καταστάσεις:

$$Seq = (1,1,1,2,1,2,2,2,2,1,2,2,2,2,1,1,1,1,2)$$

Δίνεται ότι: $p(q_0=1) = \pi_1 = 0.5$

$p(q_0=2) = \pi_2 = 0.5$

Με Maximum Likelihood έχουμε:

(δίνεται)

- $a_{11} = P(q_1=1 \mid q_0=1) = \frac{1}{4} = 0.25$ · $\cancel{a_{21} = P(q_1=1 \mid q_0=2) = 0}$
- $\cancel{a_{12} = P(q_1=2 \mid q_0=1) = 0}$ · $\cancel{a_{22} = P(q_1=2 \mid q_0=2) = 0}$

- $a_{111} = P(q_t=1 \mid q_{t-1}=1, q_{t-2}=1) = \dfrac{\#(11 \to 1)}{\#(11 \to *)} = \dfrac{3}{5} = 0.6$
- $a_{112} = P(q_t=2 \mid q_{t-1}=1, q_{t-2}=1) = \dfrac{\#(11 \to 2)}{\#(11 \to *)} = \dfrac{2}{5} = 0.4$

$$\cdot\ a_{121} = P(q_t=1 \mid q_{t-1}=2, q_{t-2}=1) = \frac{\#(12\to 1)}{\#(12\to *)} = \frac{1}{2} = 0.5$$

$$\cdot\ a_{122} = P(q_t=2 \mid q_{t-1}=2, q_{t-2}=1) = \frac{\#(12\to 2)}{\#(12\to *)} = 0.5$$

$$\cdot\ a_{211} = P(q_t=1 \mid q_{t-1}=1, q_{t-2}=2) = \frac{\#(21\to 1)}{\#(21\to *)} = \frac{1}{3} = 0.333\ldots$$

$$\cdot\ a_{212} = P(q_t=2 \mid q_{t-1}=1, q_{t-2}=2) = \frac{\#(21\to 2)}{\#(21\to *)} = \frac{2}{3} = 0.666\ldots$$

$$\cdot\ a_{221} = P(q_t=1 \mid q_{t-1}=2, q_{t-2}=2) = \frac{\#(22\to 1)}{\#(22\to *)} = \frac{2}{6} = \frac{1}{3} = 0.333\ldots$$

$$\cdot\ a_{222} = P(q_t=2 \mid q_{t-1}=2, q_{t-2}=2) = \frac{\#(22\to 2)}{\#(22\to *)} = \frac{4}{6} = \frac{2}{3} = 0.666\ldots$$

3) $$P(q_1=1, q_2=1, \ldots, q_{10}=1, q_{11}=2 \mid q_0=1) = \frac{P(\overbrace{11\ldots1}^{11\text{ times}}2)}{P(q_0=1)} = \frac{\cancel{\pi_1}\cdot a_{11}\cdot a_{111}^{9}\cdot a_{112}}{\cancel{\pi_1}}$$

$$= \frac{1}{4}(0.6)^9 \cdot 0.4 = \underline{0.0040311} \times \frac{1}{4} \simeq 0.00100775$$

4) Θεωρούμε μοντέλο Markov με μνήμη 1 (MM-1):

(α') $a_{11} = \frac{\#(1\to 1)}{\#(1\to *)} = \frac{2+3}{9} = \frac{5}{9}$ $\qquad a_{21} = \frac{\#(2\to 1)}{\#(2\to *)} = \frac{3}{9} = \frac{1}{3}$

$a_{12} = \frac{\#(1\to 2)}{\#(1\to *)} = \frac{4}{9}$ $\qquad a_{22} = \frac{\#(2\to 2)}{\#(2\to *)} = \frac{6}{9} = \frac{2}{3}$

Συνεπώς, ο πίνακας μετάβασης είναι $A = \begin{bmatrix} 5/9 & 4/9 \\ 1/3 & 2/3 \end{bmatrix}$.

(β) $$P(q_1=1, \ldots, q_{10}=1, q_{11}=2 \mid q_0=1) = \frac{\cancel{\pi_1}\cdot a_{11}^{10}\cdot(1-a_{11})}{\cancel{\pi_1}} = \left(\frac{5}{9}\right)^{10}\cdot\frac{4}{9}$$

$= \underline{0.0012448} > 0.00100775$ λόγω της υπόθεσης στο MM-2 ότι $a_{11} = \frac{1}{4} = 0.25$.

# Άσκηση 2.7: (Linear Discriminant Analysis)

Δίνονται τα μητρώα:

$$S_W = \sum_{i=1}^{|Classes|} \mathbb{E}_{x|x\in\omega_i}\left[(\vec{x}-\vec{\mu})(\vec{x}-\vec{\mu})^T\right]$$

$$S_B = \sum_{i=1}^{|Classes|} P(\omega_i)\,(\vec{\mu_i}-\vec{\mu})(\vec{\mu_i}-\vec{\mu})^T$$

όπου $\omega_i$ : i-οστη κλάση με μέση τιμή $\vec{\mu_i}$

$|Classes|$ : το πλήθος των κλάσεων

$\vec{\mu}$ : η μέση τιμή όλων των δειγμάτων

a) Έστω η περίπτωση όπου $|Classes| = 2$. Τότε:

$$S_B = P(\omega_1)\cdot(\vec{\mu_1}-\vec{\mu})(\vec{\mu_1}-\vec{\mu})^T + P(\omega_2)(\vec{\mu_2}-\vec{\mu})(\vec{\mu_2}-\vec{\mu})^T$$

Λαμβάνοντας υπόψη ότι $\vec{\mu} = P(\omega_1)\cdot\vec{\mu_1} + P(\omega_2)\vec{\mu_2}$ και $P(\omega_1) = 1 - P(\omega_2)$ :

$$S_B = P(\omega_1)\cdot\left(\vec{\mu_1} - P(\omega_1)\vec{\mu_1} - P(\omega_2)\vec{\mu_2}\right)\cdot\left(\vec{\mu_1} - P(\omega_1)\vec{\mu_1} - P(\omega_2)\vec{\mu_2}\right) +$$

$$+ P(\omega_2)\cdot\left(\vec{\mu_2} - P(\omega_1)\vec{\mu_1} - P(\omega_2)\vec{\mu_2}\right)\cdot\left(\vec{\mu_2} - P(\omega_1)\vec{\mu_1} - P(\omega_2)\vec{\mu_2}\right)$$

$$= P(\omega_1)\cdot\left[(1-P(\omega_1))\cdot\vec{\mu_1} - P(\omega_2)\vec{\mu_2}\right]\left[\vec{\mu_1}\cdot(1-P(\omega_1)) - P(\omega_2)\vec{\mu_2}\right]^T +$$

$$+ P(\omega_2)\cdot\left[-P(\omega_1)\vec{\mu_1} + (1-P(\omega_2))\vec{\mu_2}\right]\cdot\left[(1-P(\omega_2))\vec{\mu_2} - P(\omega_1)\vec{\mu_1}\right]^T$$

$$= P(\omega_1)\cdot P(\omega_2)^2\cdot(\vec{\mu_1}-\vec{\mu_2})(\vec{\mu_1}-\vec{\mu_2})^T + P(\omega_2)\,P^2(\omega_1)\cdot(\vec{\mu_2}-\vec{\mu_1})\cdot(\vec{\mu_2}-\vec{\mu_1})^T$$

$$= P(\omega_1)\cdot P(\omega_2)\cdot(\vec{\mu_1}-\vec{\mu_2})(\vec{\mu_1}-\vec{\mu_2})^T\cdot\cancel{\left(P(\omega_1)+P(\omega_2)\right)}^{\,1}$$

$$= P(\omega_1)P(\omega_2)\cdot(\vec{\mu_1}-\vec{\mu_2})(\vec{\mu_1}-\vec{\mu_2})^T$$

β) Το πρόβλημα που καλούμαστε να λύσουμε είναι το εξής:

$$S_w^{-1} S_B \vec{v} = \lambda \vec{v} \quad , \quad \vec{v} \neq \vec{0}$$

Όμως:

$$S_w^{-1} S_B \vec{v} = S_w^{-1} P(\omega_1) P(\omega_2) (\vec{\mu_1} - \vec{\mu_2}) \cdot \underbrace{(\vec{\mu_1} - \vec{\mu_2})^T \cdot \vec{v}}_{\text{βαθμωτό}}$$

$$= \underbrace{P(\omega_1) P(\omega_2) (\vec{\mu_1} - \vec{\mu_2})^T \vec{v}}_{\substack{\lambda \in \mathbb{C} \\ \text{βαθμωτό}}} \cdot S_w^{-1} (\vec{\mu_1} - \vec{\mu_2})$$

Η ποσότητα $S_w^{-1} S_B \vec{v}$ έχει την ίδια διεύθυνση με το διάνυσμα $e = S_w^{-1}(\vec{\mu_1} - \vec{\mu_2})$, για κάθε $\vec{v} \neq \vec{0}$. Το ιδιοδιάνυσμα, λοιπόν, του μητρώου $S_w^{-1} S_B$ θα είναι το:

$$\boxed{e = S_w^{-1} (\vec{\mu_1} - \vec{\mu_2})}$$

ενώ η αντίστοιχη ιδιοτιμή θα είναι:

$$\lambda = P(\omega_1) P(\omega_2) (\vec{\mu_1} - \vec{\mu_2})^T \cdot \vec{e}$$

$$\Rightarrow \boxed{\lambda = P(\omega_1) P(\omega_2) (\vec{\mu_1} - \vec{\mu_2})^T \cdot S_w^{-1} (\vec{\mu_1} - \vec{\mu_2})}$$

# Άσκηση 2.8 : (ICA)

α) Έστω τυχαίες ανεξ. μεταβλητές $x, y$ μη μηδενικές μέσες τιμές. Τότε:

- $$\mathbb{E}[(x+y)^4] = \mathbb{E}[x^4 + 4x^3y + 6x^2y^2 + 4xy^3 + y^4]$$

$$= \mathbb{E}[x^4] + 4\mathbb{E}[x^3y] + 6\mathbb{E}[x^2y^2] + 4\mathbb{E}[xy^3] + \mathbb{E}[y^4]$$

Λόγω ανεξαρτησίας, εν γένει ισχύει $\mathbb{E}[x \cdot y] = \mathbb{E}[x]\mathbb{E}[y]$. Οπότε:

$$\mathbb{E}[(x+y)^4] = \mathbb{E}[x^4] + 4\mathbb{E}[x^3]\cdot\cancelto{0}{\mathbb{E}[y]} + 6\mathbb{E}[x^2y^2] + 4\cancelto{0}{\mathbb{E}[x]}\cdot\mathbb{E}[y^3] + \mathbb{E}[y^4]$$

$$= \mathbb{E}[x^4] + 6\mathbb{E}[x^2]\mathbb{E}[y^2] + \mathbb{E}[y^4]$$

- $$\mathbb{E}[(x+y)^2] = \mathbb{E}[x^2 + 2xy + y^2] = \mathbb{E}[x^2] + 2\mathbb{E}[xy] + \mathbb{E}[y^2]$$

$$\overset{x,y \text{ ανεξ.}}{=} \mathbb{E}[x^2] + 2\cancelto{0}{\mathbb{E}[x]\cdot\mathbb{E}[y]} + \mathbb{E}[y^2] = \mathbb{E}[x^2] + \mathbb{E}[y^2].$$

Επομένως:

$$\mathrm{kurt}(x+y) = \mathbb{E}[(x+y)^4] - 3\left(\mathbb{E}[(x+y)^2]\right)^2$$

$$= \mathbb{E}[x^4] + \cancel{6\mathbb{E}[x^2]\mathbb{E}[y^2]} + \mathbb{E}[y^4] - 3\cdot\left(\mathbb{E}^2[x^2] + \mathbb{E}^2[y^2] + \cancel{2\mathbb{E}[x^2]\mathbb{E}[y^2]}\right)$$

$$= \underbrace{\mathbb{E}[x^4] - 3\left(\mathbb{E}[x^2]\right)^2}_{\mathrm{kurt}(x)} + \underbrace{\mathbb{E}[y^4] - 3\left(\mathbb{E}[y^2]\right)^2}_{\mathrm{kurt}(y)}$$

$$\Rightarrow \boxed{\mathrm{kurt}(x+y) = \mathrm{kurt}(x) + \mathrm{kurt}(y)}$$

β1) Θεωρούμε N στατιστικώς ανεξάρτητες Τ.Μ. $s_i$, με μηδενικές μέσες τιμές, μοναδιαίες διασπορές και κυρτώσεις με τιμές $a_i$, όπου $-a \leq a_i \leq a$, για κάποια άγνωστη αλλά σταθερή τιμή $a$. Δηλαδή:

$$\mathbb{E}[s_i] = 0, \quad Var(s_i) = 1, \quad kurt(s_i) = a_i, \quad \text{για } i = 1, 2, \ldots, N.$$

Αναμειγνύουμε τις παραπάνω Τ.Μ. μέσω σταθερών βαρών $w_i$ ως εξής:

$$x := \sum_{i=1}^{N} w_i s_i$$

Τότε:

$$E[x] = \mathbb{E}\left[\sum_i w_i s_i\right] = \sum_{i=1}^{N} w_i \cancelto{0}{\mathbb{E}[s_i]} = 0$$

Έπειτα:

$$Var(x) = \mathbb{E}[x^2] - \cancelto{0}{(\mathbb{E}[x])^2} = \mathbb{E}\left[\left(\sum_{i=1}^{N} w_i s_i\right)^2\right]$$

$$= \mathbb{E}\left[\sum_{i=1}^{N} w_i^2 s_i^2 + \sum_{i=1}^{N} \sum_{\substack{j=1,\\ j\neq i}}^{N} w_i w_j s_i s_j\right]$$

$$= \sum_{i=1}^{N} w_i^2 \mathbb{E}[s_i^2] + \sum_{i=1}^{N} \sum_{\substack{j=1,\\ j\neq i}}^{N} w_i w_j \cancel{\mathbb{E}[s_i] \cdot \mathbb{E}[s_j]}$$

, διότι $s_i, s_j$ ανεξάρτητα εφόσον $i \neq j$

$$= \sum_{i=1}^{N} w_i^2 \mathbb{E}[s_i^2]$$

Όμως $Var(s_i) = \mathbb{E}[s_i^2] - \cancelto{0}{(\mathbb{E}[s_i])^2} \Rightarrow \mathbb{E}[s_i^2] = 1$.

Άρα: $Var(x) = \sum_{i=1}^{N} w_i^2$. Για να έχει, λοιπόν, η τ.μ. $x$ μοναδιαία διασπορά πρέπει να ισχύει:

$$\boxed{\sum_{i=1}^{N} w_i^2 = 1}$$

B2) Θεωρούμε ένα ομοιόμορφα σταθμισμένο μείγμα, δηλαδή $w_i = w_j \ \forall i,j$, N τυχαίων μεταβλητών. Θεωρούμε, επίσης, ότι το μείγμα έχει μοναδιαία διασπορά, οπότε από την απαίτηση του (B1):

$$\sum_{i=1}^{N} w_i^2 = 1 \overset{w_i=w_j=w}{\underset{\forall i,j}{\Longrightarrow}} \sum_{i=1}^{N} w^2 = 1 \Rightarrow N \cdot w^2 = 1 \Rightarrow \boxed{w^2 = \frac{1}{N}}$$

Επιπλέον, ισχύει:

$$\mathrm{kurt}(ax) = E[(ax)^4] - 3(E[(ax)^2])^2 = a^4 \cdot \left[E[x^4] - 3(E[x^2])^2\right]$$
$$= a^4 \cdot \mathrm{kurt}(x), \quad \forall a \in \mathbb{R}.$$

Οπότε, σύμφωνα με το παραπάνω και την ιδιότητα του ερωτήματος (a) έχουμε:

$$\mathrm{kurt}(x) = \mathrm{kurt}\left(\sum_{i=1}^{N} w s_i\right) = w^4 \cdot \mathrm{kurt}\left(\sum_{i=1}^{N} s_i\right) \overset{(a)}{=} w^4 \sum_{i=1}^{N} \mathrm{kurt}(s_i)$$

$$\overset{w^2 = 1/N}{=} \frac{1}{N^2} \sum_{i=1}^{N} \mathrm{kurt}(s_i) = \frac{1}{N^2} \sum_{i=1}^{N} a_i$$

Οι επιμέρους κυρτώσεις των τ.μ. $s_i$, όμως, δηλαδή οι τιμές $a_i$ είναι φραγμένες εντός του διαστήματος $[-a, a]$, όπου a μια άγνωστη αλλά σταθερή τιμή. $\underline{\circ}s$

Εκ τούτου:

$$\boxed{\lim_{N\to+\infty} \frac{1}{N^2} \sum_{i=1}^{N} a_i = 0}, \quad \text{καθώς} \quad \lim_{N\to+\infty} \sum_{i=1}^{N} a_i \in \mathbb{R}.$$

Δηλαδή $\mathrm{kurt}(x) \xrightarrow{N\to+\infty} 0$.

## Άσκηση 2.9: (Logistic Regression)

Θεωρούμε το πρόβλημα logistic regression για ένα σύνολο δεδομένων $\{\varphi_n, t_n\}$, όπου $t_n \in \{0,1\}$ και $\varphi_n = \varphi(x_n)$ είναι οι κατηγορίες και οι συναρτήσεις βάσης, αντίστοιχα, για δείγματα $n = 1, 2, \dots, N$.

Η έξοδος του LR μοντέλου στο διάνυσμα εισόδου $x_n$ δηλώνει την a-posterior πιθανότητα το δείγμα να ανήκει στην κατηγορία $C_1$, δηλαδή:

$$y_n = p(C_1|\varphi_n) = \sigma(w^T \varphi_n)$$

και $p(C_2|\varphi_n) = 1 - P(C_1|\varphi_n) = 1 - \sigma(w^T\varphi_n)$

όπου $w$ : το διάνυσμα βαρών και $\sigma(x) = \dfrac{1}{1+e^{-x}}$ η σιγμοειδής συνάρτηση.

Τότε, η συνάρτηση πιθανοφάνειας γράφεται:

$$p(t|w) = \prod_{n=1}^{N} y_n^{t_n} \cdot \{1-y_n\}^{1-t_n}$$

από την οποία, προκύπτει το log-likelihood ή cross-entropy error:

$$E(w) = -\ln p(t|w) = -\sum_{n=1}^{N} \{t_n \ln y_n + (1-t_n)\ln(1-y_n)\}$$

a) Σε πρώτο στάδιο, προσδιορίζουμε την παράγωγο του cross-entropy σφάλματος ως προς τα βάρη $w$:

$$\nabla_w E(w) = -\nabla_w \sum_{n=1}^{N} \{t_n \ln y_n + (1-t_n)\ln(1-y_n)\}$$

$$= -\sum_{n=1}^{N} \frac{\partial}{\partial y_n}\left\{t_n \ln y_n + (1-t_n)\ln(1-y_n)\right\} \frac{\partial y_n}{\partial (w^T\varphi_n)} \cdot \frac{\partial (w^T\varphi_n)}{\partial w} \quad \text{(chain rule)}$$

$$= -\sum_{n=1}^{N} \left(\frac{t_n}{y_n} - \frac{1-t_n}{1-y_n}\right) \cdot y_n(1-y_n) \cdot \varphi_n \quad \left(\text{δεδομένου } \frac{\partial \sigma(x)}{\partial x} = \sigma(x)(1-\sigma(x))\right)$$

$$\Rightarrow \boxed{\nabla_w E(w) = \sum_{n=1}^{N} (y_n - t_n)\cdot \varphi_n}$$

Δεδομένου ότι τα target values $t_n$ παίρνουν τιμές $\{0, 1\}$, το likelihood θα λαμβάνει τη μέγιστη τιμή του για:

$$y = 0.5 \Rightarrow \sigma(w^T\varphi) = 0.5 \Rightarrow \frac{1}{1+e^{-w^T\varphi}} = 0.5 \Rightarrow \cancel{0.5} = \cancel{0.5}\, e^{-w^T\varphi}$$

$$\Rightarrow e^{-w^T\varphi} = 1 \Rightarrow \boxed{w^T\varphi = 0}$$

Με άλλα, λόγια, η διαχωριστική επιφάνεια θα είναι η $w^T\varphi(x) = 0$, έτσι τα δείγματα $x_n$ που ανήκουν στην κατηγορία $C_1$ θα έχουν $w^T\varphi(x_n) > 0$, ενώ τα δείγματα $x_m$ που ανήκουν στην κατηγορία $C_2$ θα έχουν $w^T\varphi(x_m) < 0$.

Ταυτόχρονα, απειρίζοντας το μέτρο του διανύσματος βαρών $w$, δηλαδή για $|w| \to +\infty$, θα έχουμε: $w^T\varphi(x_n) \to +\infty$, $w^T\varphi(x_m) \to -\infty$

- $$\lim_{|w|\to\infty} P(C_1|\varphi(x_n)) = \lim_{|w|\to\infty} \sigma(w^T\varphi x_n) = \lim_{|w|\to\infty} \frac{1}{1+e^{-w^T\varphi(x_n)}} = \frac{1}{1+0} = 1.$$
- $$\lim_{|w|\to\infty} P(C_2|\varphi(x_m)) = \lim_{|w|\to\infty} \left[1 - P(C_1|\varphi(x_m))\right] = \lim_{|w|\to\infty}\Bigg[1 - \underbrace{\cancel{\frac{1}{1+e^{-w^T\varphi(x_m)}}}}_{\to 0}\Bigg] = 1.$$

Επομένως, για $|w| \to +\infty$, καθένας από τους όρους $y_n = P(C_1|\varphi(x_n))$ και $1 - y_n = P(C_2|\varphi(x_n))$ θα λάβει τη μέγιστη τιμή του, δηλαδή $1$, οδηγώντας στη μεγιστοποίηση του likelihood $p(t|w) = \prod_{n=1}^{N} y_n^{t_n} \cdot (1-y_n)^{t_n}$.

Η παραπάνω προσέγγιση, ωστόσο, θα επιφέρει τεράστιο overfitting για γραμμικά διαχωρίσιμα datasets, καθώς η sigmoid function θα γίνει απείρως steep στο feature space, αντίστοιχως με την Heaviside step function. [3].

β) Η Hessian μήτρα για το logistic regression δίνεται από τη σχέση:

$$H = [\Phi^T R \Phi]$$

όπου $\Phi$ είναι ο πίνακας των χαρακτηριστικών και $R$ είναι ένας διαγώνιος πίνακας με στοιχεία $y_n(1-y_n)$.

Για να είναι ο πίνακας $H$ θετικά ορισμένος, πρέπει $\forall u \neq 0 \in \mathbb{R}^n/\{0\}$ να ισχύει:

$$u^T H u > 0$$

Ωστόσο, για τον διαγώνιο πίνακα $R$ έχουμε:

$$\det(R) = tr(R) = \sum_{n=1}^{N} y_n(1-y_n)$$

ενώ $0 < y_n < 1 \Rightarrow 0 < 1-y_n < 1$ οπότε και $0 < (1-y_n)y_n < 1$.

Συνεπώς:

$$u^T H u = u^T \Phi^T R \Phi u = u^T \sum_{n=1}^{N} \varphi_n y_n(1-y_n) \varphi_n^T \cdot u =$$

$$= \sum_{n=1}^{N} y_n(1-y_n) \cdot (\varphi_n^T \cdot u)^T (\varphi_n^T \cdot u) = \sum_{n=1}^{N} y_n(1-y_n) \cdot b_n^2 \quad , \text{ όπου } b_n = \varphi_n^T \cdot u$$

με $u \in \mathbb{R}^n/\{0\}$.

Αν υποθέσουμε ότι υπάρχει τουλάχιστον ένα μη-μηδενικό χαρακτηριστικό $\varphi_n \neq 0$, ώστε $b_n \neq 0$, τότε λόγω του ότι $0 < y_n(1-y_n) < 1$, τελικώς:

$$u^T H u = \sum_{n=1}^{N} y_n(1-y_n) \cdot b_n^2 > 0 \quad , \text{ για κάθε } u \in \mathbb{R}^n/\{0\}$$

και το μητρώο $H$ θα είναι θετικά ορισμένο.

$(H = \nabla\nabla E)$

Έχοντας, λοιπόν, $H > 0$, εύκολα προκύπτει ότι η $2^η$ παράγωγος της συνάρτησης σφάλματος $E$ ως προς τα βάρη $w$ είναι παντού θετική. Ως εκ τούτου, η $1^η$ παράγωγος της $E$ θα είναι γνησίως αύξουσα και θα έχει το πολύ ένα σημείο μηδενισμού, στο οποίο θα αντιστοιχεί το μοναδικό ελάχιστο της $E$. Ισοδυνάμως, η συνάρτηση $E$ θα είναι κυρτή.

# Άσκηση 2.9(γ): Iterative Reweighted Least Squares

Σε πρώτο στάδιο, θα υλοποιήσουμε τον αλγόριθμο iterative reweighted least squares (IWLS) για multiclass logistic regression.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import OneHotEncoder
from sklearn.metrics import accuracy_score
```

Αρχικά, διαβάζουμε τα δεδομένα εισόδου που αποτελούνται από διανύσματα δύο διαστάσεων, x1 και x2, τα οποία κατηγοριοποιούνται σε τρεις κλάσεις: 0, 1 και 2. Στα διανύσματα εισόδου X προσαυξάνουμε την διάσταση κατά ένα, ώστε x[0] = 1, προκειμένου να διευκολυνθούμε στις πράξεις με το bias των decision surfaces.

Επιπλέον, ορίζουμε:

- $\mathrm{N}$: τον αριθμό των δειγμάτων εισόδου. Στην προκειμένη περίπτωση, έχουμε 120 δείγματα.
- $\mathrm{M}$: τον αριθμό των χαρακτηριστικών κάθε δείγματος. Στην προκειμένη περίπτωση, έχουμε 3 χαρακτηριστικά, με το πρώτο να είναι πάντα μονάδα για το bias. -$K$: τον αριθμό των κλάσεων του προβλήματος, στην προκειμένη περίπτωση 3.

```
# Read Data
f = open('MLR.data',"r")
lines = f.readlines()
x1 = []
x2 = []
y = []
for x in lines:
    x1.append(float(x.split(' ')[0]))
    x2.append(float(x.split(' ')[1]))
    y.append(int(x.split(' ')[2][0]))
f.close()
```

```
X = np.column_stack((np.ones(len(x1)),x1,x2)) # concatenate 1 for bias multiplication
y = np.array(y)
print(X.shape)
N, M = X.shape
K = len(np.unique(y)) # 3 classes
```

```
(120, 3)
```

Έπειτα, για να ορίσουμε τις a-posteriori πιθανότητες του Multiclass Logistic Regression Classifier, χρησιμοποιούμε τη συνάρτηση softmax:

$$p(C_k|\varphi_n) = y_{nk}(\varphi_n) = \frac{exp^{w_k^T \varphi_n}}{\sum_{j=1}^{K} exp^{w_j^T \varphi_n}}$$

όπου $\varphi_n$: η συνάρτηση βάσης πάνω στο δείγμα $\mathrm{X}_n$.

```
def softmax(x, w, Class):
  numerator = np.exp(np.dot(w[Class,:], x))
  denominator = 0.
  for i in range(np.shape(w)[0]):
    denominator += np.exp(np.dot(w[i,:], x))
  return numerator/denominator
```

```
def softmax_all(X, w, K = 3):
  N = X.shape[0]
  Y = np.zeros((N,K)) # samples x classes = N x K
  for j in range(K):
      for n in range(N):
        Y[n,j] = softmax(X[n], w, Class = j)
  return Y
```

```
w = np.random.rand(K, M) # weight vectors KxM - one for each class
```

Έπειτα, χρησιμοποιούμε ένα 1-of-K coding scheme, με βάση το οποίο το target vector $t_n$ για ένα διάνυσμα χαρακτηριστικών $\varphi_n$ το οποίο ανήκει στην κλάση $C_k$ είναι ένα δυαδικό διάνυσμα με μηδέν όλα τα στοιχείο του, εκτός του k-οστού στοιχείου που είναι μονάδα.

```
# One-Hot Encode Labels
enc = OneHotEncoder()
enc.fit(np.array(y).reshape(len(y), 1))
T = enc.transform(np.array(y).reshape(len(y), 1)).toarray() # T: y one-hot encoded N x K
```

Ακολούθως, ορίζουμε τη συνάρτηση σφάλματος που λειτουργεί ως μετρική της επίδοσης του Logistic Regression Classifier. Η εν λόγω συνάρτηση $E(w)$, γνωστή και ως cross-entropy, είναι στην ουσία των log-likelihood των δειγμάτων εισόδου και ορίζεται ως:

$$E(w_1, \ldots, w_K) = -lnp(T|w_1, \ldots, w_K) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk} lny_{nk}$$

```
# Log-Likelihood Error Function
def log_likelihood(x, y_enc, w, classes = 3):
  E = 0

  for n in range(x.shape[0]):
    for k in range(classes):
      ynk = softmax(x[n], w, Class = k)
      E -= y_enc[n,k]*np.log(ynk)

  return E
```

```
w = np.random.rand(K, M)
print('Random Weights: Log-Likelihood E = ', log_likelihood(X, T, w))
```

```
Random Weights: Log-Likelihood E =  127.04051855009493
```

Για την εκτέλεση του IRLS αλγορίθμου, ωστόσο, χρειαζόμαστε την **Hessian μήτρα** της συνάρτησης σφάλματος E. Η μήτρα αυτή είναι διαστάσεων $KM * KM$ και αποτελείται από επαναλαμβανόμενα μπλοκ διαστάσεων $M * M$, με το μπλοκ j,k να δίνεται ως:

$$\nabla_{w_k}\nabla_{w_j} E(w) = \sum_{n=1}^{N} y_{nk}(I_{kj} - y_{nj})\varphi_n \varphi_n^T = \Phi^{\mathrm{T}} R \Phi$$

$R$: διαγώνιος πίνακας με στοιχεία $y_{nk}(I_{kj} - y_{nj})$ \ $\mathrm{I}$: ο μοναδιαίος πίνακας

```
def Hessian(X, w, K = 3):
  N, M = X.shape
  H = np.zeros((K*M,K*M))
  I = np.eye(K)
  Y = softmax_all(X, w)
  for k in range(K):
    for j in range(K):
        # Diagonal R
        R = np.zeros((N,N))
        for n in range(N):
            R[n][n] = Y[n][k] * (I[k,j] - Y[n][j])

        H[k*M:k*M+M, j*M:j*M+M] = np.dot( np.dot(X.T, R), X )

  return H # KM x KM
```

Ομοίως, προσδιορίζουμε και την πρώτη παράγωγο της συνάρτησης E ως:

$$\nabla_{w_j} E(w) = \sum_{n=1}^{N} (y_{nj} - t_{nj})\varphi_n$$

```
def GradientE(X, w, K = 3):
  N, M = X.shape
  grad_E = np.zeros((K,M)) # K x M
  for j in range(K):
    for n in range(N):
      ynj = softmax(X[n],w, Class = j)
      grad_E[j,:] += (ynj - T[n,j])*X[n]

  return grad_E
```

Στο σημείο αυτό, τρέχουμε τον αλγόριθμο IRLS για Multiclass LR. Ο αλγόριθμος τρέχει επαναληπτικά (εδώ για 20 επαναλήψεις), και τα βάρη ανανεώνονται ως εξής:

$$w^{new} = w^{old} - H^{-1}\nabla_w E$$

$$\rightarrow w^{new} = w^{old} - (\Phi^{\mathrm{T}} R \Phi)^{-1}\Phi^{\mathrm{T}}(y - t)$$

```
# IRLS Algorithm
w = np.ones((K, M)) # initialize with ones
total_error = [] # Log-Likelihood Function

for iteration in range(20):
  gradE = GradientE(X, w) # K, M
  H = Hessian(X, w) # KM x KM

  # Newton-Raphson update
  w = np.reshape(w,(K*M,1))
  gradE = np.reshape(gradE, (K*M,1))
  w -= np.dot(np.linalg.pinv(H), gradE)
  w = np.reshape(w, (K,M))

  total_error.append(log_likelihood(X,T,w)) # collect loss
```

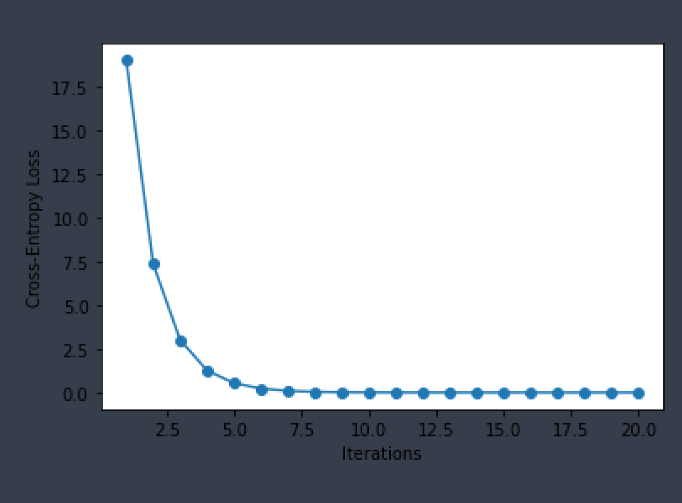
```
print('Weights After 20 Iterations of IRLS:')
print(w)
```

```
Weights After 20 Iterations of IRLS:
[[ -0.55576221  -6.52705988   2.66298619]
 [-12.35573972   1.31499084  -4.16030782]
 [ 15.04372858  -4.48760453  -8.97792324]]
```

```
print('Cross-Entropy Error after 20 Iterations of IRLS:', log_likelihood(X,T,w))
```

```
Cross-Entropy Error after 20 Iterations of IRLS: 2.864611554360134e-07
```

```
plt.plot([i for i in range(1,21)], total_error, '-o')
plt.xlabel('Iterations')
plt.ylabel('Cross-Entropy Loss')
plt.show()
```

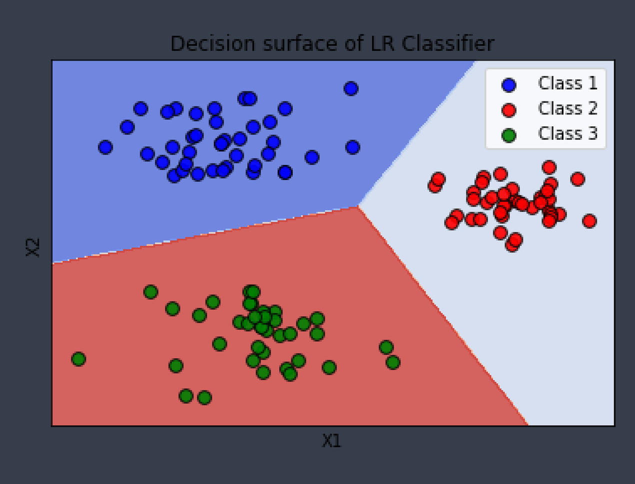



Ακολούθως, βλέπουμε τις επιφάνειες διαχωρισμού που χωρίζουν τα δείγματα στις τρεις κατηγορίες, με βάση τις ανανεωμένες τιμές των βαρών w μετά από 20 επαναλήψεις του IRLS αλγορίθμου.

```
# Function that predicts output class, given weights w
def predict(X, w):
  return np.argmax(softmax_all(X,w), axis = 1)
```

```
# Function to plot decision Surfaces (from python helper lab)
def plot_clf(w, X, y, labels, clf = 'IRLS'):
    fig, ax = plt.subplots()
    # title for the plots
    title = ('Decision surface of LR Classifier')
    # Set-up grid for plotting.
    X0, X1, X2 = X[:,0], X[:,1], X[:,2]

    x_min, x_max = X1.min() - 1, X1.max() + 1
    y_min, y_max = X2.min() - 1, X2.max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, .05),
                         np.arange(y_min, y_max, .05))

    if clf == 'IRLS':
      Z = predict(np.column_stack((np.ones(len(xx.ravel())), np.c_[xx.ravel(), yy.ravel
()])), w)
    else:
      Z = np.argmax(np.dot(np.column_stack((np.ones(len(xx.ravel())), np.c_[xx.ravel(), yy.
ravel()])),w), axis=1)
    Z = Z.reshape(xx.shape)
    out = ax.contourf(xx, yy, Z, cmap=plt.cm.coolwarm, alpha=0.8)

    zeros = ax.scatter(
        X1[y == 0], X2[y == 0],
        c='blue', label=labels[0],
        s=60, alpha=0.9, edgecolors='k')
    ones = ax.scatter(
        X1[y == 1], X2[y == 1],
        c='red', label=labels[1],
        s=60, alpha=0.9, edgecolors='k')
    twos = ax.scatter(
        X1[y == 2], X2[y == 2],
        c='green', label=labels[2],
        s=60, alpha=0.9, edgecolors='k')

    ax.set_ylabel('X2')
    ax.set_xlabel('X1')
    ax.set_xticks(())
    ax.set_yticks(())
    ax.set_title(title)
    ax.legend()
    plt.show()
```

```
plot_clf(w, X, y, labels=['Class 1', 'Class 2', 'Class 3'])
```




Παρατηρεί κανείς ότι οι τελικές ευθείες που ορίζουν τα βάρη w χωρίζουν τέλεια τα δείγματα εισόδου στις τρεις κλάσεις. Η επιτυχία αυτή μπορεί να ποσοτικοποιηθεί και με βάση την μετρική accuracy:

```
def score(X, w, y):
  y_pred = predict(X,w)
  acc = accuracy_score(y_pred, y)
  return acc

accuracy = score(X,w,y)
print('IRLS Logistic Regression Accuracy = '+str(100*accuracy)+'%')
```

```
IRLS Logistic Regression Accuracy = 100.0%
```

Παρατηρούμε ότι έχουμε ποσοστό ευστοχίας 100%.

## Linear Regression with Sum-of-Squares Loss Function

Στο πρόβλημα γραμμικής πρόβλεψης με συνάρτηση κόστους την Sum-of-Squares ο Newton-Raphson τύπος ανανέωσης βαρών παίρνει σε μόλις ένα βήμα την μορφή:

$$w^{new} = (\Phi^{\mathrm{T}}\Phi)^{-1}\Phi^{\mathrm{T}} t$$

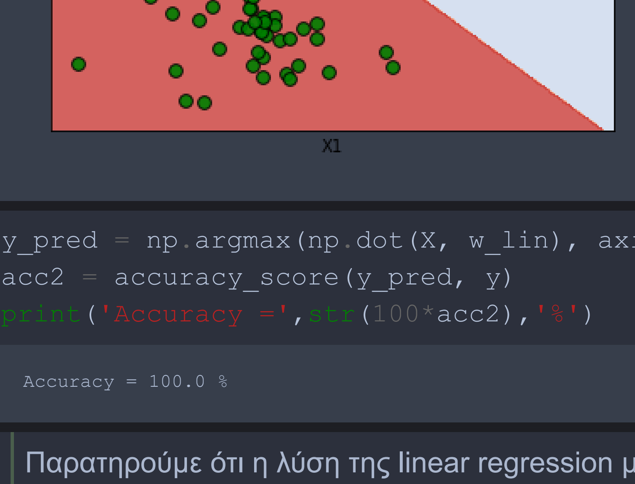
την οποία αναγνωρίζουμε ως τη κλασική λύση ελαχίστων τετραγώνων.

```
pseudo_inv = np.linalg.pinv(X)
w_lin = np.dot(pseudo_inv,T)
print('Weights of Linear Regression with S.o.S Function:')
print(w_lin)
```

```
Weights of Linear Regression with S.o.S Function:
[[ 0.30015285  0.12967629  0.57017086]
 [-0.05663662  0.08432925 -0.02769264]
 [ 0.12494785  0.01922407 -0.14417192]]
```

Παρουσιάζουμε στον χώρο τις τρεις ευθείες διαχωρισμού για τα άνωθι βάρη, καθώς και το ποσοστό ευστοχίας του ταξινομητή και την τιμή της συνάρτησης σφάλματος cross-entropy.

```
plot_clf(w_lin, X, y, labels=['Class 1', 'Class 2', 'Class 3'], clf='Linear Regression')
```

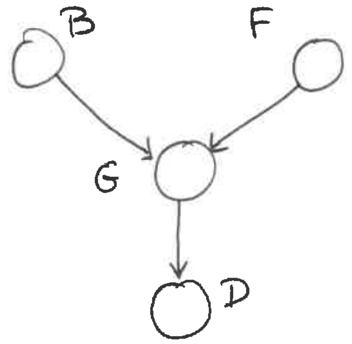



```
y_pred = np.argmax(np.dot(X, w_lin), axis=1)
acc2 = accuracy_score(y_pred, y)
print('Accuracy =',str(100*acc2),'%')
```

```
Accuracy = 100.0 %
```

Παρατηρούμε ότι η λύση της linear regression με συνάρτηση κόστους την Sum of Squares δίνει εξίσου ικανοποιητικά αποτελέσματα με τον αλγόριθμο IRLS (Accuracy = 100%). Οι ευθείες διαχωρισμού καταφέρνουν να χωρίσουν τα δείγματα στις 3 κλάσεις πλήρως.

# Άσκηση 2.10: (Conditional Independence)




Battery B
Fuel Tank F
Gauge G
Driver D

| B | 0 | 1 |
|---|---|---|
| P(B) | 0.05 | 0.95 |

| F | 0 | 1 |
|---|---|---|
| P(F) | 0.2 | 0.8 |

$P(G|BF)$:

| BF\G | 0 | 1 |
|---|---|---|
| 0 0 | 0.8 | 0.2 |
| 0 1 | 0.75 | 0.25 |
| 1 0 | 0.7 | 0.3 |
| 1 1 | 0.05 | 0.95 |

$P(D|G)$:

| G\D | 0 | 1 |
|---|---|---|
| 0 | 0.8 | 0.2 |
| 1 | 0.2 | 0.8 |

$\boxed{\text{Παρατηρούμε ότι } D=0}$

1.) Πιθανότητα το ντεπόζιτο F να είναι άδειο $(F=0)$, δεδομένου της παρατήρησης $D=0$, δηλαδή:

$$P(F=0 \mid D=0) = \frac{P(F=0, D=0)}{P(D=0)} = \frac{\sum_{B,G} P(B,G,F=0,D=0)}{\sum_{B,F,G} P(B,F,G,D=0)}$$

Επομένως:

| B | G | F | D | P(F) | P(B) | P(G/BF) | P(D/G) | P(BFGD) |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0.2 | 0.05 | 0.8 | 0.8 | 0.0064 |
| 0 | 1 | 0 | 0 | 0.2 | 0.05 | 0.2 | 0.2 | 0.0004 |
| 1 | 0 | 0 | 0 | 0.2 | 0.95 | 0.7 | 0.8 | 0.1064 |
| 1 | 1 | 0 | 0 | 0.2 | 0.95 | 0.3 | 0.2 | 0.0114 (+) |
| | | | | | | | | 0.1246 |

| B | F | G | D | P(B) | P(F) | P(G\|BF) | P(D\|G) | P(BFGD) |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0.05 | 0.2 | 0.8 | 0.8 | 0.0064 |
| 0 | 0 | 1 | 0 | 0.05 | 0.2 | 0.2 | 0.2 | 0.0004 |
| 0 | 1 | 0 | 0 | 0.05 | 0.8 | 0.75 | 0.8 | 0.024 |
| 0 | 1 | 1 | 0 | 0.05 | 0.8 | 0.25 | 0.2 | 0.002 |
| 1 | 0 | 0 | 0 | 0.95 | 0.2 | 0.7 | 0.8 | 0.1064 |
| 1 | 0 | 1 | 0 | 0.95 | 0.2 | 0.3 | 0.2 | 0.0114 |
| 1 | 1 | 0 | 0 | 0.95 | 0.8 | 0.05 | 0.8 | 0.0304 |
| 1 | 1 | 1 | 0 | 0.95 | 0.8 | 0.95 | 0.2 | 0.1444 (+) |
| | | | | | | | | 0.3254 |

Τελικώς $P(F=0|D=0) = \dfrac{0.1246}{0.3254} = 38.29\%$

2) Ζητάμε την πιθανότητα $P(F=0|D=0, B=0) = \dfrac{P(F=0, D=0, B=0)}{P(D=0, B=0)}$

| B | F | G | D | P(B) | P(F) | P(G\|BF) | P(D\|G) | P(BFGD) |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0.05 | 0.2 | 0.8 | 0.8 | 0.0064 |
| 0 | 0 | 1 | 0 | 0.05 | 0.2 | 0.2 | 0.2 | 0.0004 (+) |
| | | | | | | | | 0.0068 |

| B | F | G | D | P(B) | P(F) | P(G\|BF) | P(D\|G) | P(BFGD) |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0.05 | 0.2 | 0.8 | 0.8 | 0.0064 |
| 0 | 0 | 1 | 0 | 0.05 | 0.2 | 0.2 | 0.2 | 0.0004 |
| 0 | 1 | 0 | 0 | 0.05 | 0.8 | 0.75 | 0.8 | 0.024 |
| 0 | 1 | 1 | 0 | 0.05 | 0.8 | 0.25 | 0.2 | 0.002 (+) |
| | | | | | | | | 0.0328 |

Τελικώς:

$$P(F=0|D=0,B=0) = \frac{0.0068}{0.0328} = 0.2073 \text{ ή } 20.73\% < 38.29\%$$

Παρατηρούμε ότι η άνωθι πιθανότητα είναι μικρότερη από αυτήν που υπολογίσαμε στο υποερώτημα (1), γεγονός που οφείλεται στο explaining away, δηλαδή το μοτίβο συλλογιστικής σύμφωνα με το οποίο αν προκαθορίσουμε την πιθανή αιτία ενός παρατηρούμενο γεγονότος, τότε ~~οι~~ πιθανότητες που αφορούν εναλλακτικές πιθανές αιτίες μειώνονται.

Στην προκειμένη περίπτωση, για το παρατηρούμενο γεγονός $D=0$ προκαθορίζουμε ως αιτία την άδεια μπαταρία $(B=0)$ και ως αποτέλεσμα μειώνεται η πιθανότητα της αιτίας το άδειο ντεπόζιτο $(F=0)$.

# References

[1] https://towardsdatascience.com/the-mathematics-of-decision-trees-random-forest-and-feature-importance-in-scikit-learn-and-spark-f2861df67e3

[2] https://sites.millersville.edu/bikenaga/linear-algebra/spectral-theorem/spectral-theorem.html

[3] C. M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006. Ch. 4.3.2, 4.3.3, 4.3.4