ΚΟΝΤΟΓΙΑΝΝΗΣ ΑΝΔΡΕΑΣ 03115187
2η ΑΝΑΛΥΤΙΚΗ ΑΣΚΗΣΗ ΑΝΑΓΝΩΡΙΣΗ ΠΡΟΤΥΠΩΝ

Άσκηση 2.1: (Hidden Markov Models)
Ζητείται να χρησιμοποιηθεί ένα HMM για να αποκωδικοποιηθεί μια απλή ακολουθία DNA. Είναι γνωστό ότι μια ακολουθία DNA είναι μια σειρά από στοιχεία του συνόλου $\{A, C, G, T\}$. Ας υποθέσουμε ότι υπάρχει μία κρυμμένη κατάσταση S που ελέγχει τη δημιουργία της ακολουθίας DNA και έχει 2 πιθανές καταστάσεις $\{S_1, S_2\}$. Επίσης, δίνονται οι ακόλουθες πιθανότητες μετάβασης για το HMM λ:

$$P(S_1|S_1) = 0.8 \quad P(S_2|S_1) = 0.2 \quad P(S_1|S_2) = 0.2 \quad P(S_2|S_2) = 0.8$$

οι ακόλουθες πιθανότητες των παρατηρήσεων:

$$P(A|S_1) = 0.4 \quad P(C|S_1) = 0.1 \quad P(G|S_1) = 0.4 \quad P(T|S_1) = 0.1$$

$$P(A|S_2) = 0.1 \quad P(C|S_2) = 0.4 \quad P(G|S_2) = 0.1 \quad P(T|S_2) = 0.4$$

και οι ακόλουθες a-priori πιθανότητες:

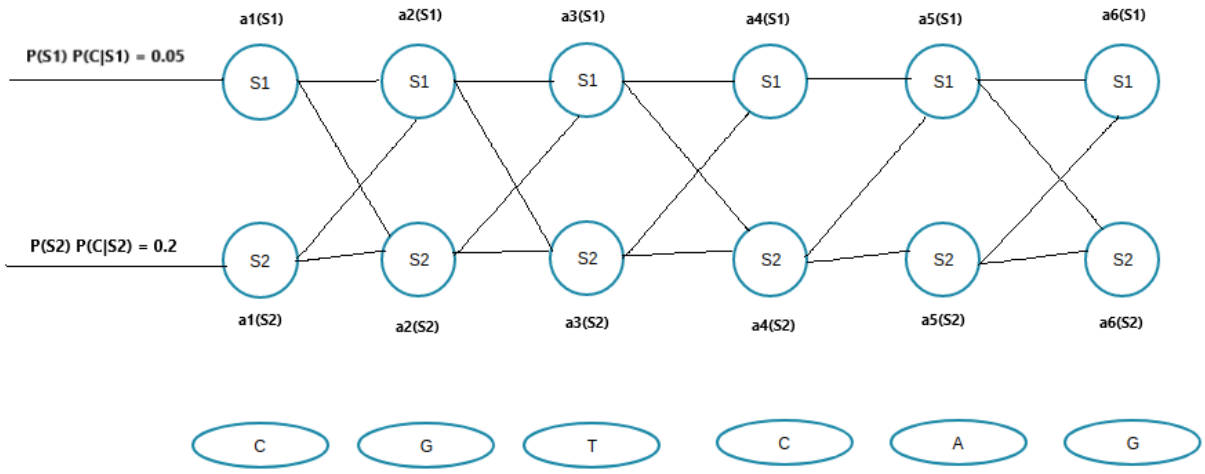
$$P(S_1) = 0.5 \quad P(S_2) = 0.5$$

Έστω ότι η παρατηρούμενη ακολουθία είναι $x = CGTCAG$. Υπολογίστε:
(α) Την πιθανότητα $P(x|\lambda)$ χρησιμοποιώντας τον forward αλγόριθμο.
(β) Τις εκ των υστέρων πιθανότητες $P(\pi_i = S_1|x, \lambda)$ για $i = 1, \ldots, 6$.
(γ) Το πιο πιθανό μονοπάτι κρυμμένων καταστάσεων χρησιμοποιώντας τον αλγόριθμο Viterbi.

Λύση

(α) Forward Probabilities

Θέλουμε να υπολογίσουμε την πιθανότητα P(x|λ) κάνοντας χρήση του forward αλγόριθμου, δεδομένου ότι η observed sequence είναι η CGTCAG. Παρακάτω, αναπαριστούμε το HMM model.



Μένει να υπολογίσουμε τις forward probabilities $a_i(S_i)$.

$$a_1(s_1) = 0.05,$$
$$a_1(s_2) = 0.2$$

$$a_2(S_1) = P(S_1|S_1) \cdot P(G|S_1) \cdot a_1(S_1) + P(S_1|S_2) \cdot P(G|S_1) \cdot a_1(S_2) = 0.032$$
$$a_2(S_2) = P(S_2|S_2) \cdot P(G|S_2) \cdot a_1(S_2) + P(S_2|S_1) \cdot P(G|S_2) \cdot a_1(S_1) = 0.017$$

$$a_3(S_1) = P(S_1|S_1) \cdot P(T|S_1) \cdot a_2(S_1) + P(S_1|S_2) \cdot P(T|S_1) \cdot a_2(S_2) = 0.029$$
$$a_3(S_2) = P(S_2|S_2) \cdot P(T|S_2) \cdot a_2(S_2) + P(S_2|S_1) \cdot P(T|S_2) \cdot a_2(S_1) = 0.008$$

$$a_4(S_1) = P(S_1|S_1) \cdot P(C|S_1) \cdot a_3(S_1) + P(S_1|S_2) \cdot P(C|S_1) \cdot a_3(S_2) = 0.00248$$
$$a_4(S_2) = P(S_2|S_2) \cdot P(C|S_2) \cdot a_3(S_2) + P(S_2|S_1) \cdot P(C|S_2) \cdot a_3(S_1) = 0.00488$$

$$a_5(S_1) = P(S_1|S_1) \cdot P(A|S_1) \cdot a_4(S_1) + P(S_1|S_2) \cdot P(A|S_1) \cdot a_4(S_2) = 0.0046976$$
$$a_5(S_2) = P(S_2|S_2) \cdot P(A|S_2) \cdot a_4(S_2) + P(S_2|S_1) \cdot P(A|S_2) \cdot a_4(S_1) = 0.00044$$

$$a_6(S_1) = P(S_1|S_1) \cdot P(G|S_1) \cdot a_5(S_1) + P(S_1|S_2) \cdot P(G|S_1) \cdot a_5(S_2) = 0.001538432$$
$$a_6(S_2) = P(S_2|S_2) \cdot P(G|S_2) \cdot a_5(S_2) + P(S_2|S_1) \cdot P(G|S_2) \cdot a_5(S_1) = 0.000129152$$

Έχουμε:

$P(\mathbf{x}|\lambda) = 0.001538432 + 0.000129152 = 0.001667584$, αποτέλεσμα που προέκυψε με κάποιες στρογγυλοποιήσεις.

(β) γ_i

Υπολογίζουμε πρώτα τα α_i, β_i και ύστερα τα γ_i.
Για το πρώτο state:
$\alpha_1 = [0.05, 0.65306122, 0.26605505, 0.12311558, 0.60137931, 0.83628137]$
Για το δεύτερο state:
$\alpha_2 = [0.2, 0.34693878, 0.73394495, 0.87688442, 0.39862069, 0.16371863]$

Για το πρώτο state:
$\beta_1 = [5.12546277, 0.68531469, 0.85191279, 2.29206088, 1.26748252, 1]$
Για το δεύτερο state:
$\beta_2 = [3.71863431, 1.59234883, 1.05368161, 0.81859317, 0.59646236, 1]$

Για το πρώτο state:
$\gamma_1 = [0.25627314, 0.44755245, 0.2266557, 0.2821884, 0.76223776, 0.83628137]$
Για το δεύτερο state:
$\gamma_2 = [0.74372686, 0.55244755, 0.7733443, 0.7178116, 0.23776224, 0.16371863]$

(γ) Viterbi algorithm

Αρχικοποιούμε $v_1(i) = \pi_i b_i$, $v_1(1) = 1/20$, $v_1(2) = 1/5$.

Επαναλαμβάνουμε την διαδικασία με $v_t(j) = max[v_{t-1}(i)a_{ij}b_j(x_t)]$. Λαμβάνουμε:

$v_2(1) = max(\frac{32}{2000}, \frac{32}{2000})$

$v_2(2) = max(\frac{1}{1000}, \frac{16}{1000})$

$v_3(1) = max(\frac{256}{200000}, \frac{32}{100000})$

$v_3(2) = max(\frac{256}{200000}, \frac{512}{100000})$

$v_4(1) = max(\frac{1024}{10000000}, \frac{1024}{10000000})$

$v_4(2) = max(\frac{1024}{10000000}, \frac{16384}{10000000})$

$v_5(1) = max(\frac{32768}{10^9}, \frac{131072}{10^9})$

$v_5(2) = max(\frac{2048}{10^9}, \frac{131072}{10^9})$

$v_6(1) = max(\frac{4194304}{10^{11}}, \frac{1018576}{10^{11}})$

$v_6(1) = max(\frac{262141}{10^{11}}, \frac{1018576}{10^{11}})$

Επομένως, η πιο πιθανή ακολουθία με τον αλγόριθμο Viterbi είναι η $2- > 2- > 2- > 2- > 1- > 1$, με πιθανότητα να συμβεί $\frac{4194304}{10^{11}}$.

Άσκηση 2.2: (Principal Component Analysis)
Ζητείται η εφαρμογή της Principal Component Analysis (PCA) πάνω στο ευρέως διαδεδομένο σύνολο δεδομένων κρίνων του Fisher, προκειμένου να μετασχηματιστούν τα δεδομένα σε ένα χώρο χαμηλότερων διαστάσεων. Τα δεδομένα αποτελούνται από 3 κλάσεις (για τους 3 διαφορετικούς τύπους κρίνου), καθεμιά από τις οποίες περιλαμβάνει 50 δείγματα. Τα δεδομένα περιγράφονται από 4 διαφορετικά χαρακτηριστικά:

- μήκος σεπάλων σε εκ.
- πλάτος σεπάλων σε εκ.
- μήκος πετάλων σε εκ.
- πλάτος πετάλων σε εκ.
- τύπος κρίνου (Iris Setosa/Iris Versicolour/Iris Virginica)

(α) Κατεβάστε το σύνολο δεδομένων που έχει ανεβεί στο mycourses (αρχείο `PCA.data`).
(β) Προεπεξεργαστείτε τα δεδομένα αφαιρώντας τη μέση τιμή και διαιρώντας με την τυπική απόκλιση του κάθε χαρακτηριστικού ξεχωριστά. Τα προκύπτοντα δεδομένα θα πρέπει να έχουν μέση τιμή 0 και διασπορά 1.
(γ) Υπολογίστε τον δειγματικό πίνακα συνδιασπορών $\Sigma = \frac{1}{m-1}\sum_{i=1}^{m} \mathbf{x}_i \mathbf{x}_i^\top$, όπου $\mathbf{x}_i$ είναι το i-στό δείγμα και m το πλήθος των δειγμάτων.
(δ) Παραγοντοποιήστε τον πίνακα συνδιασπορών κάνοντας χρήση του Singular Value Decomposition (SVD) και βρείτε τις αντίστοιχες ιδιοτιμές και ιδιοδιανύσματα. Προσέξτε εάν η συγκεκριμένη υλοποίηση του SVD δίνει τα αποτελέσματα με φθίνουσα ή αύξουσα σειρά ιδιοτιμών. Ο μετασχηματισμός SVD ενός πίνακα Σ είναι μια παραγοντοποίηση της μορφής $\Sigma = UDV^\top$. Για έναν συμμετρικό, θετικά ορισμένο πίνακα Σ, οι $U = V$ περιέχουν τα ιδιοδιανύσματα και D είναι ένας διαγώνιος πίνακας με τις αντίστοιχες ιδιοτιμές.
(ε) Προβάλετε τα δεδομένα πάνω στις δύο πρώτες κύριες συνιστώσες και σχεδιάστε τα αποτελέσματα που προκύπτουν.
(στ) Ποιος είναι ο ελάχιστος απαιτούμενος αριθμός από κύριες συνιστώσες ώστε να "ερμηνεύεται" το 95% της διασποράς των τιμών;

Λύση

(α) Διάβασμα δεδομένων

:
```
1  import pandas as pd
2  import numpy as np
3
4  # Read Data and take rows and columns
5  df = pd.read_csv('PCA.data', na_values = '?', header = None)
6  print("First 5 records:\n")
7  print(df.head())
8
9  # Drop the last column
10 y = np.array(df[4])
11 del df[4]
```

```
First 5 records:

     0    1    2    3           4
0  5.1  3.5  1.4  0.2  Iris-setosa
1  4.9  3.0  1.4  0.2  Iris-setosa
2  4.7  3.2  1.3  0.2  Iris-setosa
3  4.6  3.1  1.5  0.2  Iris-setosa
4  5.0  3.6  1.4  0.2  Iris-setosa
```

(β) Προεπεξεργασία δεδομένων

```
# Preprocessing
# (X - μ) / σ
# Standardization
data = np.array(df)
for i in range(4):
    mu = np.mean(data[:,i])
    sigma = np.sqrt(np.var(data[:,i]))
    data[:,i] = (data[:,i] - mu) / sigma
```

(γ) Υπολογισμός δειγματικού πίνακα συνδιασπορών

```
M = len(data)
cov = np.zeros((4,4))
for i in range(M):
    cov += np.matmul(np.array([data[i]]).transpose(), np.array([data[i]]))

cov = cov / (M - 1)
print('Δειγματικός πίνακας συνδιασπορών\n')
print(cov)

# print(np.cov(data, rowvar=False)) same result
```

```
Δειγματικός πίνακας συνδιασπορών

[[ 1.00671141 -0.11010327  0.87760486  0.82344326]
 [-0.11010327  1.00671141 -0.42333835 -0.358937  ]
 [ 0.87760486 -0.42333835  1.00671141  0.96921855]
 [ 0.82344326 -0.358937    0.96921855  1.00671141]]
```

(δ) Singular Value Decomposition (SVD)

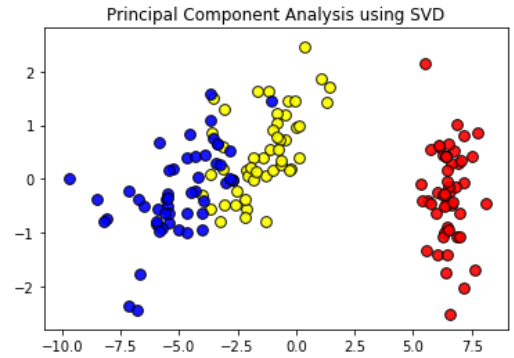
```
u, d, vh = np.linalg.svd(cov, full_matrices=True)
print(u)
print()
print(np.diag(d))
print()
print(vh)
```

```
[[-0.52237162 -0.37231836  0.72101681  0.26199559]
 [ 0.26335492 -0.92555649 -0.24203288 -0.12413481]
 [-0.58125401 -0.02109478 -0.14089226 -0.80115427]
 [-0.56561105 -0.06541577 -0.6338014   0.52354627]]

[[2.93035378 0.         0.         0.        ]
 [0.         0.92740362 0.         0.        ]
 [0.         0.         0.14834223 0.        ]
 [0.         0.         0.         0.02074601]]

[[-0.52237162  0.26335492 -0.58125401 -0.56561105]
 [-0.37231836 -0.92555649 -0.02109478 -0.06541577]
 [ 0.72101681 -0.24203288 -0.14089226 -0.6338014 ]
 [ 0.26199559 -0.12413481 -0.80115427  0.52354627]]
```

Παρατηρούμε πως ο αρχικός δειγματικός πίνακας συνδιασπορών ήταν συμμετρικός και θετικά ορισμένος. Αυτό έχει σαν αποτέλεσμα οι $U = V$ και να περιέχουν τα ιδιοδιανύσματα, ενώ ο D να είναι ένας διαγώνιος πίνακας με τις αντίστοιχες ιδιοτιμές. Στον D οι ιδιοτιμές εμφανίζονται σε φθίνουσα σειρά.

(ε) Principal Component Analysis (PCA)

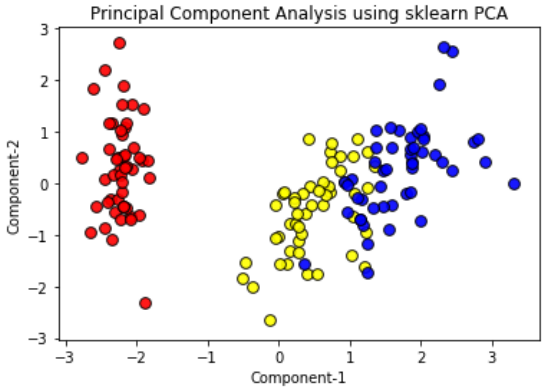
```
import matplotlib.pyplot as plt

u_new = u[:,0:2]
d = np.diag(d)
d_new = d[0:2,0:2]

b = np.dot(u_new, d_new)
dataPCA = np.dot(data, b)

# Plot
plt.title('Principal Component Analysis using SVD')
plt.scatter(dataPCA[0:50,0], dataPCA[0:50,1], color='red', edgecolors='k', s=60, alpha=0.9)
plt.scatter(dataPCA[50:100,0], dataPCA[50:100,1], color='yellow', edgecolors='k', s=60, alpha=0.9)
plt.scatter(dataPCA[100:150,0], dataPCA[100:150,1], color='blue', edgecolors='k', s=60, alpha=0.9)
plt.show()
```




```
# unique labels/types
print(pd.unique(y))
```

```
['Iris-setosa' 'Iris-versicolor' 'Iris-virginica']
```

```
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

pca = PCA(n_components=2)
dataPCA = pca.fit_transform(data)

# Plot
X0, X1 = dataPCA[:,0], dataPCA[:,1]
colors = {
    'Iris-setosa': 'red',
    'Iris-versicolor': 'yellow',
    'Iris-virginica': 'blue'
}
fig, ax = plt.subplots()
for i,label in enumerate(colors.keys()):
    ax.scatter(
        X0[y == label], X1[y == label],
        c=(colors[label]),
        s=60, alpha=0.9, edgecolors='k'
    )
ax.set_xlabel('Component-1')
ax.set_ylabel('Component-2')
ax.set_title('Principal Component Analysis using sklearn PCA')
plt.show()
```



(στ) PCA explained variation

```
print("Using PCA with SVD\n")

print('Explained variation for PCA(2): {}'.format(np.sum((d[0] + d[1]) / np.sum(d))))
print('Explained variation for PCA(1): {}'.format(d[0,0] / np.sum(d)))
```

```
Using PCA with SVD

Explained variation for PCA(2): 0.9580097536148198
Explained variation for PCA(1): 0.7277045209380135
```

```
print("Using sklearn PCA\n")
var2 = np.sum(pca.explained_variance_ratio_)
print('Explained variation for PCA(2): {}'.format(var2))

pca = PCA(n_components=1)
dataPCA = pca.fit_transform(data)
var1 = np.sum(pca.explained_variance_ratio_)
print('Explained variation for PCA(1): {}'.format(var1))
```

```
Using sklearn PCA

Explained variation for PCA(2): 0.9580097536148197
Explained variation for PCA(1): 0.7277045209380132
```

Παρατηρούμε ότι και με χρήση του SVD και με χρήση του sklearn, τα δυο PCA δίνουν ακριβώς τα ίδια αποτελέσματα, όπως και περιμέναμε. Βλέπουμε ακόμα ότι ο ελάχιστος αριθμός κύριων συνιστωσών που ερμηνεύεται τουλάχιστον το 95% της πληροφορίας (variance) των τιμών ειναι για n_components $= 2$.

Άσκηση 2.3: (Linear Discriminant Analysis)
Στο μάθημα είδαμε ότι η Linear Discriminant Analysis (LDA) βασίζεται στην ανάστροφη σχέση των μητρών (πινάκων) S_W και S_B:

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

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

όπου το ω_i αναπαριστά μια κλάση με μέση τιμή $\vec{\mu_i}$, $|Classes|$ είναι το πλήθος των κλάσεων και $\vec{\mu}$ είναι η μέση τιμή όλων των δειγμάτων.
(α) Δείξτε ότι στην περίπτωση διαχωρισμού δύο κλάσεων ω_1 και ω_2, ο πίνακας S_B μπορεί να γραφτεί στη μορφή $S_B = P(\omega_1)P(\omega_2)(\vec{\mu_2}-\vec{\mu_1})(\vec{\mu_2}-\vec{\mu_1})^T$.
(β) Βασιζόμενοι στο υποερώτημα **(α)**, να βρείτε το ιδιοδιάνυσμα του πίνακα $S_W^{-1}S_B$ και την ιδιοτιμή του.

Λύση

(α)

Για $\mathrm{K} = 2$, έχουμε:

$$\mathbf{S}_B = P(\omega_1)(\boldsymbol{\mu}_1 - \boldsymbol{\mu})(\boldsymbol{\mu}_1 - \boldsymbol{\mu})^{\mathrm{T}} + P(\omega_2)(\boldsymbol{\mu}_2 - \boldsymbol{\mu})(\boldsymbol{\mu}_2 - \boldsymbol{\mu})^{\mathrm{T}}$$

Γνωρίζουμε ότι το $\boldsymbol{\mu}$, ως ο μέσος όρος όλων των data points μπορεί να γραφεί:

$$\boldsymbol{\mu} = \frac{1}{N}(N_1\boldsymbol{\mu}_1 + \mathrm{N}_2\boldsymbol{\mu}_2) = P(\omega_1)\boldsymbol{\mu}_1 + \mathrm{P}(\omega_2)\boldsymbol{\mu}_2$$

Ακόμη, επειδή έχουμε μόνο δυο κατηγορίες (classes) θα έχουμε ότι $P(\omega_1) = 1 - P(\omega_2)$. Άρα, μπορούμε να γράψουμε:

$$\begin{aligned}
\mathbf{S}_B &= P(\omega_1)\left((1 - \mathrm{P}(\omega_1))\boldsymbol{\mu}_1 - \mathrm{P}(\omega_2)\boldsymbol{\mu}_2\right)\left((1 - \mathrm{P}(\omega_1))\boldsymbol{\mu}_1 - \mathrm{P}(\omega_2)\boldsymbol{\mu}\right)^T \\
&\quad + P(\omega_2)\left((1 - \mathrm{P}(\omega_2))\boldsymbol{\mu}_2 - \mathrm{P}(\omega_1)\boldsymbol{\mu}_1\right)\left((1 - \mathrm{P}(\omega_2))\boldsymbol{\mu}_2 - \mathrm{P}(\omega_1)\boldsymbol{\mu}_1\right)^T \\
&= P(\omega_1)\left(\mathrm{P}(\omega_2)\boldsymbol{\mu}_1 - \mathrm{P}(\omega_2)\boldsymbol{\mu}_2\right)\left(\mathrm{P}(\omega_2)\boldsymbol{\mu}_1 - \mathrm{P}(\omega_2)\boldsymbol{\mu}_2\right)^T + P(\omega_2)\left(\mathrm{P}(\omega_1)\boldsymbol{\mu}_2 - \mathrm{P}(\omega_1)\boldsymbol{\mu}_1\right)\left(\mathrm{P}(\omega_1)\boldsymbol{\mu}_2 - \mathrm{P}(\omega_1)\boldsymbol{\mu}_1\right)^T \\
&= P(\omega_1)P(\omega_2)(\mathrm{P}(\omega_1) + P(\omega_2))\left(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\right)\left(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\right)^T \\
&= P(\omega_1)P(\omega_2)\left(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1\right)\left(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1\right)^T
\end{aligned}$$

(β) Eigenvalue problem

Έχουμε:

$$\mathbf{S}_W^{-1}\mathbf{S}_B\mathbf{x} = \lambda\mathbf{x}$$

Αλλά

$$\mathbf{S}_B\mathbf{x} = (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T = a(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$$

Αυτό σημαίνει πως το παραπάνω διάνυσμα δείχνει πάντα προς την κατεύθυνση του $\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2$, αφού ο a είναι scalar. Έτσι, επιλύουμε το πρόβλημα των ιδιοτιμών απευθείας. Πιο αναλυτικά:

$$\mathbf{S}_W^{-1}\mathbf{S}_B\mathbf{x} = \mathbf{S}_W^{-1}[a(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)] = a[\mathbf{S}_W^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)]$$

όπου $\lambda = a$, η ιδιοτιμή, και $\mathbf{x} = \mathbf{S}_W^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$, το ιδιοδιάνυσμα.

Άσκηση 2.4: (Multilayer Perceptron)
Μας δίνονται 10 διανύσματα χαρακτηριστικών που προέρχονται από δύο κλάσεις ω_1 και ω_2:

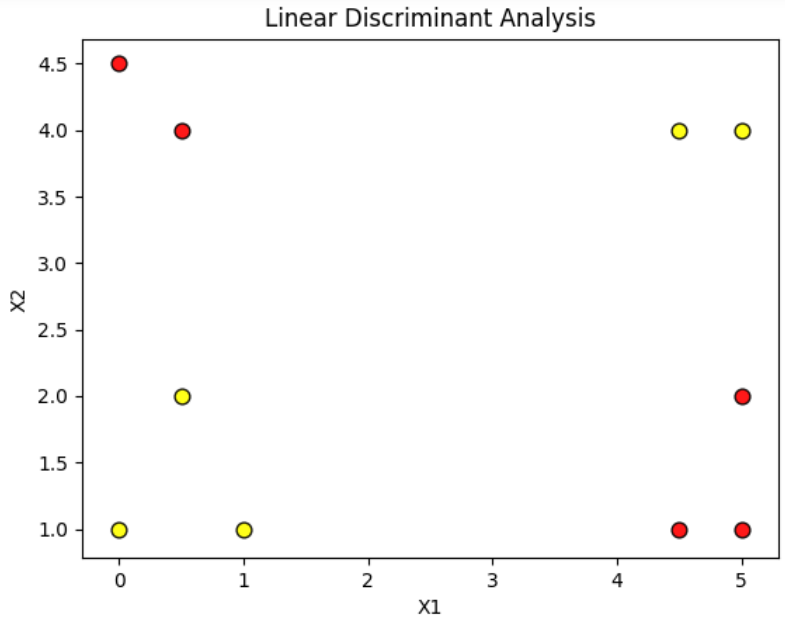
$$\omega_1 : [4.5, 1]^\top, [5, 2]^\top, [5, 1]^\top, [0, 4.5]^\top, [0.5, 4]^\top$$

$$\omega_2 : [0, 1]^\top, [0.5, 2]^\top, [5, 4]^\top, [4.5, 4]^\top, [1, 1]^\top$$

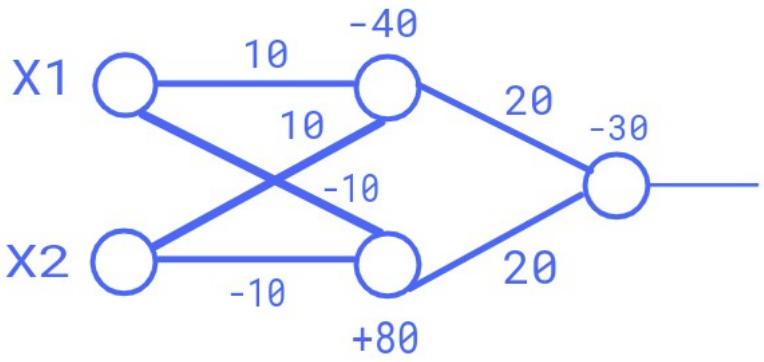
Ελέγξτε αν οι δύο κλάσεις είναι γραμμικά διαχωρίσιμες, και αν όχι, σχεδιάστε ένα κατάλληλο multilayer perceptron με τους κόμβους να έχουν βηματική συνάρτηση ενεργοποίησης (step)

Λύση

Ελέγχουμε αν τα δείγματα είναι γραμμικά διαχωρίσιμα. Από την παρακάτω γραφική παράσταση, βλέπουμε φανερά πως δεν είναι και, από την άλλη, συμπεραίνουμε ότι το πρόβλημά μας είναι παρόμοιο με αυτό της XOR.



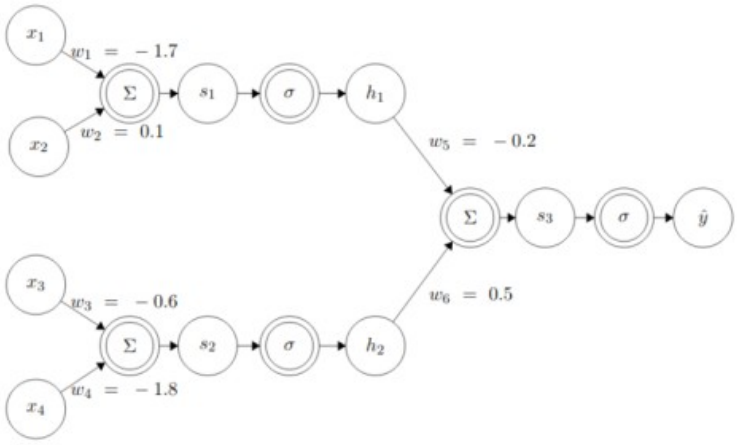
Παρακάτω, σχεδιάζουμε το MLP που επιλύει πλήρως το πρόβλημά μας.



Για το παραπάνω MLP, το οποίο χρησιμοποιεί για activation function την step function: $f(x) = 1, x > 0$ αλλίως -1, τρέχουμε όλα τα data points που έχουμε δεδομένα και δείχνουμε πως κατηγοριοποιούνται όλα σωστά.

Για το [4.5,1]: $step(4.5(10) + 1(10) - 40) = 1$, $step(4.5(-10) + 1(-10) + 80) = 1$, $step(1(20) + 1(20) - 30) = 1$, αρα ορθώς κατηγοριοποιείται στην πρώτη κατηγορία.
Για το [5,2]: $step(5(10) + 2(10) - 40) = 1$, $step(5(-10) + 2(-10) + 80) = 1$, $step(1(20) + 1(20) - 30) = 1$, αρα ορθώς κατηγοριοποιείται στην πρώτη κατηγορία.
Για το [5,1]: $step(5(10) + 1(10) - 40) = 1$, $step(5(-10) + 1(-10) + 80) = 1$, $step(1(20) + 1(20) - 30) = 1$, αρα ορθώς κατηγοριοποιείται στην πρώτη κατηγορία.
Ομοίως και τα για τα υπόλοιπα data points, όλα κατηγοριοποιούνται στις σωστές κατηγορίες.

Άσκηση 2.5: (Backpropagation)
Υποθέστε ότι έχουμε το ακόλουθο νευρωνικό δίκτυο. Οι κόμβοι που βρίσκονται μέσα σε μονό κύκλο υποδηλώνουν μεταβλητές (για παράδειγμα η x_1 είναι μια μεταβλητή εισόδου, h_1 είναι μια ενδιάμεση μεταβλητή, και $\hat{y}$ είναι μια μεταβλητή εξόδου). Οι κόμβοι που βρίσκονται μέσα σε διπλό κύκλο υποδηλώνουν συναρτήσεις (για παράδειγμα το $\sum$ υπολογίζει το άθροισμα των εισόδων του και η σ αναπαριστά τη συνάρτηση logistic $\sigma(x) = \frac{1}{1+e^{-x}}$.



Θεωρήστε ότι η συνάρτηση για το L2 loss δίνεται από τη σχέση $L(y, \hat{y}) = ||y - \hat{y}||_2^2$. Επίσης, μας δίνονται τα δεδομένα ενός δείγματος $(x_1, x_2, x_3, x_4) = (-0.7, 1.2, 1.1, -2)$ με τιμή για το πραγματικό του label ίση με 0.5. Χρησιμοποιήστε τον αλγόριθμο backpropagation για να υπολογίσετε τη μερική παράγωγο $\frac{\partial L}{\partial w_1}$.
Σημείωση: Το gradient για μια συνάρτηση L2 loss είναι ίσο με $2||y - \hat{y}||$.

Λύση

Αρχικά, θα εκτελέσουμε forward propagation για να βρούμε τα s_1, s_2, s_3 και το y:

$$s_1 = -1.7(-0.7) + 0.1(1.2) = 1.31$$
$$h_1 = \sigma(s_1) = 0.78751$$

$$s_2 = 1.1(-0.6) + (-2)(-1.8) = 2.94$$
$$h_2 = \sigma(s_2) = 0.94979$$

$$s_3 = h_1(-0.2) + h_2(0.5) = 0.3174$$
$$y = \sigma(s_3) = 0.5787$$

Στην συνέχεια, εκτελούμε το backpropagation για να βρούμε τα deltas, $\delta = \frac{\partial L}{\partial s}$, και τελικά το $\frac{\partial L}{\partial \omega_1}$:
Για το output unit:

$$\delta_1^{(3)} = 2||t - y|| \cdot \sigma'(s_3)$$

Για το hidden unit:

$$\delta_1^{(2)} = \sigma'(s_1) \cdot \omega_5 \cdot \delta_1^{(3)}$$

Τελικά, υπολογίζουμε το ζητούμενο derivative:

$$\frac{\partial L}{\partial \omega_1} = \delta_1^{(2)} \cdot x_1 = 8.99 \cdot 10^{-4}$$

Άσκηση 2.6: (Support Vector Machine)
Θεωρήστε το πρόβλημα του διαχωρισμού ενός συνόλου από διανύσματα εκπαίδευσης δύο κλάσεων. Τα δεδομένα εκπαίδευσης είναι της μορφής $\{(\mathbf{x}_i, y_i)\}$, όπου τα διανύσματα χαρακτηριστικών $\mathbf{x}_i \in \mathbb{R}^m$ και τα labels των κλάσεων $y_i \in \{-1, 1\}$.
Όπως είναι γνωστό, στην περίπτωση όπου τα δεδομένα εκπαίδευσης δεν είναι γραμμικώς διαχωρίσιμα (για παράδειγμα μέσω ενός κανόνα απόφασης $\text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$ για κάποια $\mathbf{w}, b$), τότε το πρόβλημα χρειάζεται να διατυπωθεί με χρήση slack variables $\{\xi_i\}$, $1 \leq i \leq n$. Έτσι, ο ταξινομητής SVM με το μεγαλύτερο περιθώριο αποκτάται μέσω της επίλυσης του δυϊκού προβλήματος:

$$L(\mathbf{w}, b, \alpha, \xi, \beta) = \frac{1}{2}\mathbf{w} \cdot \mathbf{w} + C\sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \alpha_i[(\mathbf{w} \cdot \mathbf{x}_i + b)y_i - (1 - \xi_i)] - \sum_{i=1}^{n} \beta_i \xi_i$$

όπου το C είναι μια σταθερά, $\alpha_i, \beta_i \geq 0$, $\forall i$ είναι οι πολλαπλασιαστές Lagrange, και $\xi_i \geq 0$ είναι οι slack variables.
(α) Υποθέστε ότι $n = 4$ και ότι το $\mathbf{x}$ είναι δύο διαστάσεων $< x_i^1, x_i^2 >:< 2, 2 >, < 2.5, 2.5 >, < 5, 5 >, < 7, 7 >$. Τώρα το SVM εκπαιδεύεται με βάση την παραπάνω εξίσωση. Να δείξετε ότι οποιαδήποτε labels y και αν έχουν τα τέσσερα δείγματα εκπαίδευσης, το βέλτιστο διάνυσμα

παραμέτρων $\hat{\mathbf{w}} = (\hat{w}^1, \hat{w}^2)$ έχει την ιδιότητα ότι $\hat{w}^1 = \hat{w}^2$.
(β) Θεωρήστε την εκπαίδευση ενός SVM με slack variables, αλλά δίχως την ύπαρξη του bias όρου ($b = 0$). Θα χρησιμοποιήσουμε έναν kernel $\mathbf{K}(\mathbf{u}, \mathbf{v})$ με την ιδιότητα ότι για δύο οποιαδήποτε σημεία $\mathbf{u}$ και $\mathbf{v}$ που ανήκουν στο σύνολο εκπαίδευσης ισχύει ότι $-1 < \mathbf{K}(\mathbf{u}, \mathbf{v}) < 1$. Επιπρόσθετα, $\mathbf{K}(\mathbf{u}, \mathbf{u}) < 1$. Να δείξετε ότι αν υπάρχουν n δείγματα στο σύνολο εκπαίδευσης και η σταθερά C επιλέγεται ούτως ώστε $C < \frac{1}{n-1}$, τότε όλες οι δυϊκές μεταβλητές α_i είναι μη μηδενικές (και άρα όλα τα δείγματα του συνόλου εκπαίδευσης αποτελούν support vectors).
(γ) Θεωρήστε τον εξής kernel:

$$\mathbf{K}(\mathbf{u}, \mathbf{v}) = \mathbf{u} \cdot \mathbf{v} + 4(\mathbf{u} \cdot \mathbf{v})^2$$

όπου τα διανύσματα $\mathbf{u}$ και $\mathbf{v}$ είναι δύο διαστάσεων. Ο kernel αυτός είναι ίσος με το εσωτερικό γινόμενο $\phi(\mathbf{u}) \cdot \phi(\mathbf{v})$ για κάποιο ορισμό της συνάρτησης ϕ. Ποια είναι η συνάρτηση αυτή ϕ;

Λύση

(α) Example

Έχουμε ότι $\boldsymbol{\varphi}(\mathbf{x}_n) = \mathbf{x}_n$, και άρα μπορούμε να γράψουμε:

$$\mathbf{w} = [w_1, w_2]$$

$$\mathbf{w} = \sum_{n=1}^{N} a_n t_n \mathbf{x}_n = [2a_1t_1 + 2.5a_2t_2 + 5a_3t_3 + 7a_4t_4, 2a_1t_1 + 2.5a_2t_2 + 5a_3t_3 + 7a_4t_4]$$

Άρα, λαμβάνουμε ότι $w_1 = w_2$, και άρα μπορούμε να γράψουμε

(β) Support Vectors

Έχουμε ότι:

$-1 < \mathrm{K}(\mathbf{u}, \mathbf{v}) < 1,$

$C < \frac{1}{N-1}$

και θέλουμε να ότι όλες οι δυϊκές μεταβλητές a_n είναι μη μηδενικές.
Υποθέτουμε πως υπάρχει κάποια δυϊκή μεταβλητή $a_j = 0$ και θέλουμε να οδηγηθούμε σε άτοπο. Λαμβάνοντας υπόψιν ότι $b_n \geq 0$ και ότι $a_n + b_n = C$, παίρνουμε ότι:

$a_n + b_n < \frac{1}{N-1} => a_n < \frac{1}{N-1}$, για κάθε $n \in [1, N]$.

Ακόμη, επειδή $a_j = 0$ τότε $a_j + b_j = C => b_j \neq 0$. Και επειδή $b_j \xi_j = 0 => \xi_j = 0$. Έτσι, μπορούμε να γράψουμε:

$$\xi_j = |t_j - y(\mathbf{x}_j)| = 0$$

$$=> \left| t_j - \sum_{n=1}^{N} a_n t_n K(\mathbf{x}_j, \mathbf{x}_n) \right| = 0$$

$$=> t_j = \sum_{n=1}^{N} a_n t_n K(\mathbf{x}_j, \mathbf{x}_n)$$

$$=> |t_j| = \left| \sum_{n=1}^{N} a_n t_n K(\mathbf{x}_j, \mathbf{x}_n) \right|$$

$$=> 1 = \left| \sum_{n=1}^{N} a_n t_n K(\mathbf{x}_j, \mathbf{x}_n) \right|$$

Άρκει να δείξουμε ότι η παραπάνω συνθήκη οδηγεί σε άτοπο. Από τριγωνική ανισότητα ισχύει ότι:

$$\left| \sum_{n=1}^{N} a_n t_n K(\mathbf{x}_j, \mathbf{x}_n) \right| \leq \sum_{n=1}^{N} |a_n t_n K(\mathbf{x}_j, \mathbf{x}_n)| = \sum_{n=1}^{N} |a_n||t_n||K(\mathbf{x}_j, \mathbf{x}_n)|$$

Έχουμε ακόμη ότι:

$$|a_n| < \frac{1}{N-1}$$

$$|\mathrm{K}(\mathbf{u}, \mathbf{v})| < 1$$

$$|t_n| = 1$$

Όμως, το παραπάνω άθροισμα περιέχει όλα τα μη αρνητικά a_n, και άρα αθροίζει συνολικά N-1 όρους. Άρα προκύπτει:

$$\left| \sum_{n=1}^{N} a_n t_n K(\mathbf{x}_j, \mathbf{x}_n) \right| \leq \sum_{n=1}^{N} |a_n t_n K(\mathbf{x}_j, \mathbf{x}_n)| = \sum_{n=1}^{N} |a_n||t_n||K(\mathbf{x}_j, \mathbf{x}_n)| < \frac{\mathrm{N}-1}{N-1} = 1$$

Με αυτόν τον τρόπο, οδηγούμαστε σε άτοπον και άρα δεν υπάρχει κάποια δυϊκή μεταβλητή $a_j = 0$, με αποτέλεσμα όλα τα data points να αποτελούν support vectors.

(γ) Kernel example

Έχουμε ότι:

$$\mathrm{K}(\mathbf{u}, \mathbf{v}) = \mathbf{uv} + 4(\mathbf{uv})^2$$

$$= x_1 y_1 + x_2 y_2 + 4x_1^2 y_1^2 + 8x_1 x_2 y_1 y_2 + 4x_2^2 y_2^2$$

$$= [x_1, x_2, 2x_1^2, \sqrt{8}x_1 x_2, 2x_2^2] \cdot [y_1, y_2, 2y_1^2, \sqrt{8}y_1 y_2, 2y_2^2]$$

$$= \phi(\mathbf{u}) \cdot \phi(\mathbf{v})$$

Άσκηση 2.7: (Logistic Regression)
Θεωρήστε το πρόβλημα logistic regression για ένα σύνολο δεδομένων $\{\boldsymbol{\phi}_n, t_n\}$, όπου $t_n \in \{0, 1\}$ και $\boldsymbol{\phi}_n = \boldsymbol{\phi}(\mathbf{x}_n)$ είναι οι κατηγορίες και οι συναρτήσεις βάσης, αντίστοιχα, για δείγματα $n = \{1, 2, \ldots, N\}$. Η συνάρτηση σφάλματος $E(\mathbf{w})$, η οποία αναφέρεται συνήθως και ως cross-entropy, ορίζεται ως:

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

όπου $\mathbf{w}$ είναι το διάνυσμα βαρών, $y_n = \sigma(\mathbf{w}^\top \boldsymbol{\phi}_n)$ η έξοδος του μοντέλου logistic regression στο διάνυσμα εισόδου $\mathbf{x}_n$, και $\sigma(a) = \frac{1}{1+\exp(-a)}$ η logistic sigmoid συνάρτηση.
(α) Να δείξετε ότι για ένα γραμμικώς διαχωρίσιμο σύνολο δεδομένων, η λύση μέγιστης πιθανοφάνειας για το μοντέλο logistic regression αντιστοιχεί στην εύρεση ενός διανύσματος $\mathbf{w}$, για το οποίο η επιφάνεια απόφασης $\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}) = 0$ διαχωρίζει τις κλάσεις, απειρίζοντας ταυτόχρονα το μέτρο του διανύσματος $\mathbf{w}$.

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

$$\mathbf{H} = [\boldsymbol{\Phi}^\top \mathbf{R} \boldsymbol{\Phi}]$$

όπου $\boldsymbol{\Phi}$ είναι ο πίνακας των χαρακτηριστικών και $\mathbf{R}$ είναι ένας διαγώνιος πίνακας με στοιχεία $y_n(1 - y_n)$.
Να δείξετε ότι η Hessian μήτρα $\mathbf{H}$ είναι θετικώς ορισμένη. Ως εκ τούτου, δείξτε ότι η συνάρτηση σφάλματος είναι κυρτή συνάρτηση του $\mathbf{w}$ και ότι έχει μοναδικό ελάχιστο.
(γ) Να γράψετε κώδικα που θα υλοποιεί τον iterative reweighted least squares (IWLS) αλγόριθμο για logistic regression. Χρησιμοποιώντας τον αλγόριθμο αυτό, να υπολογίσετε και να σχεδιάσετε τα διαχωριστικά επίπεδα απόφασης που αντιστοιχούν στο σύνολο δεδομένων του προβλήματος τριών κλάσεων που έχει ανεβεί στο mycourses (αρχείο `MLR.data`). Οι δύο πρώτες στήλες περιλαμβάνουν τα διανύσματα χαρακτηριστικών, ενώ η τρίτη την κλάση. Συγκρίνετε τα αποτελέσματα με εκείνα που θα προέκυπταν εάν εφαρμοζόταν ταξινόμηση με βάση τα ελάχιστα τετράγωνα, σχεδιάζοντας τα αντίστοιχα διαχωριστικά επίπεδα απόφασης.

Λύση

(α) Maximum Likelihood and the cross-entropy error function for a binary classification problem

Για ένα γραμμικά διαχωρίσιμο dataset $\{\boldsymbol{\phi}_n, t_n\}$ όπου $t_n \in \{0, 1\}$ και $\boldsymbol{\phi}_n = \boldsymbol{\phi}(\mathbf{x}_n)$, και ακόμα $p(C_i|\boldsymbol{\phi}) = y(\boldsymbol{\phi}) = \sigma(\mathbf{w}^T\boldsymbol{\phi})$, με σ(.) να είναι η logistic sigmoid function, η likelihood function μπορεί να γραφεί:

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

Ο αρνητικός λογάριθμος της παραπάνω συνάρτησης είναι η cross-entropy error function. Άρα η μεγιστοποίηση της likelihood function ανάγεται στην ελαχιστοποίηση της cross-entropy error function. Δεδομένου ότι:

$$\frac{d\sigma}{da} = \sigma(1 - \sigma)$$

όπου a η ανεξάρτητη μεταβλητή της logistic sigmoid.

Γνωριζούμε για τη logistic sigmoid ότι:

- αν $a > 0$ τότε $\sigma(a) > 0.5$,
- αν $a < 0$ τότε $\sigma(a) < 0.5$,
- αν $a = 0$ τότε $\sigma(a) = 0.5$

Οπότε η επιλογή της κλάσης C_i έναντι της C_j, με $i, j = \{0, 1\}$, $i \neq j$, κατά τη διαδικασία της πρόβλεψης, θα είναι όταν:

$$p(C_i|\phi) > p(C_j|\phi)$$

.

Έτσι, η μία θα επιλέγεται όταν $\sigma(\mathbf{w}^T\phi) > 0.5$ και η άλλη όταν το $\sigma(\mathbf{w}^T\phi) < 0.5$, πάντοτε με την αρχική μας προϋπόθεση ότι $p(C_i|\phi) = 1 - p(C_j|\phi)$. Άρα, το decision boundary θα είναι όταν:

$$\sigma(\mathbf{w}^T\phi) = 0.5$$

$$=> \mathbf{w}^T\phi = 0$$

Η likelihood προκειμένου να μεγιστοποιηθεί, δηλαδή να ελαχιστοποιηθεί η cross-entropy error function, πρέπει για τα δείγματα i που $t_i = 0$ το $\mathbf{w}^T\phi$ να τείνει στο $-\infty$, ενώ για τα δείγματα j που $t_j = 1$ το $\mathbf{w}^T\phi$ να τείνει στο $+\infty$, αφού πρέπει στην πρώτη περίπτωση να ελαχιστοποιείται το $1 - ln(y_i)$, ενώ στην δεύτερη να μεγιστοποιείται το $ln(y_j)$. Δεδομένου ότι μπορούμε να γράψουμε, αφού έχουμε να κάνουμε με Euclidean vectors:

$$\mathbf{w}^T\phi = ||\mathbf{w}^T|| \cdot ||\phi|| \cdot cos\theta,$$

όπου η θ, η γωνία των δυο παραπάνω διανυσμάτων, και το $||\phi||$ σταθερό, τότε και στις δυο περιπτώσεις το $||\mathbf{w}^T||$ πρέπει να τείνει στο $+\infty$.
Άμεσο συμπέρασμα αυτών είναι ότι αυτή η μέθοδος του maximum likelihood είναι αρκετά ευάλωτη στο overfitting για γραμμικώς διαχωρίσιμα δείγματα, κάτι που μπορεί να αποφευχθεί με χρήση regularization ή με εισαγωγή priors και μέθοδο MAP.

(β) Logistic Regression and Hessian Matrix

Θεωρούμε αυθαίρετο vector $\mathbf{u} \neq \mathbf{0}$ μεγέθους M, όπου M ο αριθμός των χαρακτηριστικών. Προκειμένου ο $\mathbf{H}$ να είναι θετικά ορισμένος (positive definite) αρκεί να δείξουμε ότι:

$$\mathbf{u}^T\mathbf{H}\mathbf{u} > 0$$

Έχουμε:

$$\mathbf{u}^T\mathbf{H}\mathbf{u} = \mathbf{u}^T[\sum_{n=1}^{N} y_n(1 - y_n)\boldsymbol{\varphi}_n\boldsymbol{\varphi}_n^T]\mathbf{u}$$

$$= \sum_{n=1}^{N} y_n(1 - y_n)(\boldsymbol{\varphi}_n^T\mathbf{u})^T(\boldsymbol{\varphi}_n^T\mathbf{u})$$

$$= \sum_{n=1}^{N} y_n(1 - y_n)c_n^2 > 0$$

αφού $c_n^2 > 0$, διότι θα έχει οπωσδήποτε τουλάχιστον μια μη μηδενική τιμή, και $y_n(1 - y_n) > 0$, διότι το y_i αποτελεί πιθανότητα. Άρα, ο $\mathbf{H}$ ειναι positive definite και άρα η error function είναι convex και έχει global minimum.

(γ) Iterative Reweighted Least Squares (IRLS) with Logistic Regression

```
# Read in the file
with open('MLR.data', 'r') as file :
    filedata = file.read()

# Replace the "" string with ","
filedata = filedata.replace(' ', ',')

# Write the file out again
with open('MLR.data', 'w') as file:
    file.write(filedata)
```

```python
import pandas as pd
import numpy as np

# Read Data and take rows and columns
df = pd.read_csv('MLR.data', na_values = '?', header = None)
print("First 5 records:\n")
print(df.head())

# Drop the last column
t = np.array(df[2])
del df[2]
X = np.array(df)
K = np.unique(t)
```

```
First 5 records:

          0         1  2
0 -3.605388  3.012493  0
1  8.941356  2.108067  1
2  4.161469 -2.959369  2
3  0.047052  4.166693  0
4  6.749173  1.589336  1
```

```python
from sklearn.base import BaseEstimator, ClassifierMixin
from numpy import dot, multiply, squeeze, zeros, ones, reshape, array, identity
import numpy as np

class LogisticRegression(BaseEstimator, ClassifierMixin):
    # Iterative Reweighted Least Squares (IRLS) for Multiclass Logistic Regression
    def __init__(self, K, maxiter=10):
        '''
            K: int, classes
            maxiter: int
        '''

        self.W = None
        self.K = K
        self.maxiter = maxiter


    @staticmethod
    def stable_softmax(x):
        '''
            x: list of scalars
        '''
        z = x - max(x)
        numerator = np.exp(z)
        denominator = np.sum(numerator)
        softmax = numerator / denominator

        return softmax


    @staticmethod
    def softmax(W, X, K):
        '''
            W: K x M
            X: N x M
            K: int
        '''

        N = X.shape[0]
        M = X.shape[1]
        softmax_all = zeros((N,K))

        for i,xn in enumerate(X):
            xn = reshape(xn, (M,1))  # M x 1

            A = []
            for k in range(K):
                a_nk = squeeze(dot( reshape(W[k], (1,M)), xn )) # scalar
                A.append(a_nk)

            softmax_all[i,:] = reshape(array(stable_softmax(A)), (1,K))

        return softmax_all  # N x K
```

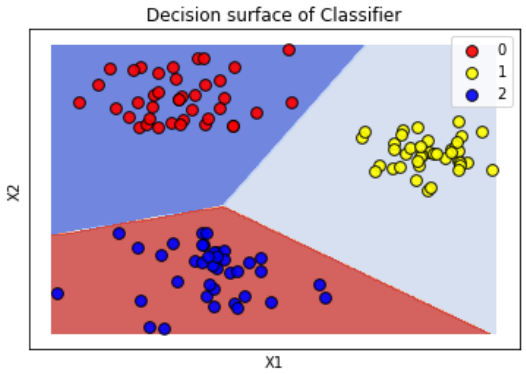
```
    def fit(self, X, t):
        '''
            X: N x M
            t: list of targets
        '''

        K = self.K
        X_t = X.transpose()

        N = X.shape[0] # n_samples
        M = X.shape[1] # n_features
        I = np.identity(K)  # K x K

        # 1-of-K coding scheme for target values
        T = zeros((N,K))   # N x K
        for i,k in enumerate(t):
            T[i][k] = 1

        # Initialize w, grad, hessian
        self.W = zeros(shape=(K,M))        # K x M
        grad = ones(shape=(K,M))      # K x M
        H = ones(shape=(K*M,K*M))     # K*M x K*M
         for iteration in range(self.maxiter):

             # Y = softmax for all n,k
             Y = softmax(W, X, K)  # N x K

             # gradient of cross-entropy error function
             grad = dot( X_t, Y - T ).transpose()   # K x M

             # Hessian Matrix: K*M x K*M
             for k in range(K):
                 for j in range(K):
                     Ikj = I[k][j]
                     # Diagonal R
                     R = zeros((N,N))
                     for n in range(N):
                         R[n][n] = Y[n][k] * (Ikj - Y[n][j])

                     H[k*M:k*M+M, j*M:j*M+M] = dot( dot(X_t, R), X )  # M x M

             # Newton-Raphson update
             self.W = reshape(self.W, (K*M,1))
             grad = reshape(grad, (K*M,1))
             Wnew = self.W - dot( np.linalg.pinv(H), grad )

            # keep preferred dimensions
            grad = reshape(grad, (K,M))
            self.W = reshape(Wnew, (K,M))

            continue

        return self


    def predict(self, X):
        # Logistic Regression predict on given test data
        solve = softmax(W, X, self.K)
        y_predict = np.argmax(solve, axis=1)

        return y_predict


    def score(self, X, y_truth):
        # Logistic Regression accuracy score on given test data
        y_predict = self.predict(X)

        return ( np.sum([y_predict == y_truth]) ) / ( len(y_predict) )
```

```
lr = LogisticRegression(K=3, maxiter=5)
lr.fit(X, t)
print("IRLS LogisticRegression Score:\t", lr.score(X, t) * 100, "%")
print("Final W:\n", lr.W)
```

```
IRLS LogisticRegression Score:   100.0 %
Final W:
 [[-0.35798298  1.48913729]
 [ 0.39032115  0.39589378]
 [-0.03233817 -1.88503107]]
```

```
import warnings
warnings.filterwarnings("ignore",category=DeprecationWarning)
warnings.filterwarnings("ignore",category=FutureWarning)

import numpy as np
import matplotlib.pyplot as plt

def plt_decision_boundaries(clf, X, y, labels, plot_labels):
    # Plot Decision boundaries for a 10 labels classification problem, supposed that labers are in range[0,3]
    '''
        clf:            classifier that has fit, predict and score methods
        X:              X_train
        y:              y_train
        labels:         an numpy array of labels
        plot_labels:    a list of the axis titles on the plot
    '''
    fig, ax = plt.subplots()
    # title for the plots
    title = ('Decision surface of Classifier')

    # Initializations
    X0, X1 = X[:,0], X[:,1]

    x_min, x_max = X0.min() - 0.2, X0.max() + 0.2
    y_min, y_max = X1.min() - 0.2, X1.max() + 0.2
    xx, yy = np.meshgrid(np.arange(x_min, x_max, .05),
                         np.arange(y_min, y_max, .05))

    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)
    out = ax.contourf(xx, yy, Z, cmap=plt.cm.coolwarm, alpha=0.8)

    colors = ['red', 'yellow', 'blue']

    for label in labels:
        ax.scatter(
            X0[y == label], X1[y == label],
            c=(colors[int(label)]), label=int(label),
            s=60, alpha=0.9, edgecolors='k'
        )

    ax.set_ylabel(plot_labels[1])
    ax.set_xlabel(plot_labels[0])
    ax.set_xticks(())
    ax.set_yticks(())
    ax.set_title(title)
    ax.legend()
    plt.show()
```

```
lr = LogisticRegression(K=3, maxiter=5)
lr.fit(X, t)
plt_decision_boundaries(lr, X, t, np.arange(3), ["X1", "X2"])
```



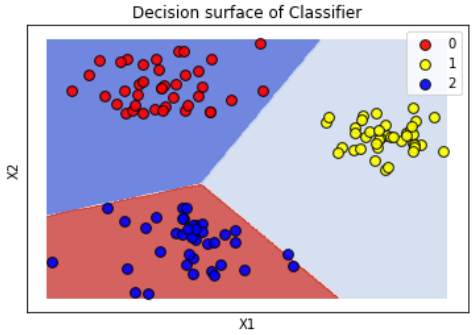
Linear Regression and Mean Squared Error cost function (Normal Equations method)

```
from sklearn.base import BaseEstimator, ClassifierMixin
from numpy import dot, multiply, squeeze, zeros, ones, reshape, array, identity
import numpy as np

class LinearRegression(BaseEstimator, ClassifierMixin):
    # Normal Equations for Multiclass Logistic Regression
    def __init__(self, K):
        '''
            K: int, classes
        '''

        self.Wml = None
        self.K = K


    def fit(self, X, t):
        '''
            X: N x M
            t: list of targets
        '''

        K = self.K
        N = X.shape[0] # n_samples
        M = X.shape[1] # n_features
        X_t = X.transpose()

        # 1-of-K coding scheme for target values
        T = zeros((N,K))   # N x K
        for i,k in enumerate(t):
            T[i][k] = 1

        # Mean Squared Error cost function - Maximum Likelihood solution
        pseudo_inv = dot( np.linalg.pinv(dot(X_t, X)), X_t ) # pseudo-inverse
        self.Wml = dot( pseudo_inv, T)

        return self


    def predict(self, X):
        # Linear Regression predict on given test data
        Y = dot( X, self.Wml )
        y_predict = np.argmax(Y, axis=1)

        return y_predict


    def score(self, X, y_truth):
        # Linear Regression accuracy score on given test data
        y_predict = self.predict(X)

        return ( np.sum([y_predict == y_truth]) ) / ( len(y_predict) )
```

```
lr = LinearRegression(K=3)
lr.fit(X, t)
print("Normal Equations LinearRegression Score:\t", lr.score(X, t) * 100, "%")
print("Wml:\n", lr.Wml)
```

```
Normal Equations LinearRegression Score:        99.16666666666667 %
Wml:
 [[-0.03973394  0.09163179  0.00441572]
 [ 0.15567397  0.0324988  -0.08580454]]
```

```
lr = LinearRegression(K=3)
lr.fit(X, t)
plt_decision_boundaries(lr, X, t, np.arange(3), ["X1", "X2"])
```



Παρατηρούμε πως και τα δυο μοντέλα δίνουν σχεδόν άριστα ποσοστά. Το Logistic Regression δίνει 100%, ενώ το Linear Regression δίνει 99.17%. Αυτό οφείλεται στην διαφορετική cost function καθώς και στην activation function που χρησιμοποιούν. Βλέπουμε πως αν και τα δείγματα είναι linearly seperable, η χρήση της softmax και της cross-entropy καταφέρνουν να μην κάνουν κανένα λάθος, τοποθετώντας τα δείγματα πιο κεντρικά στην αντίστοιχη περιοχή απόφασης, σε αντίθεση με τη χρήση των normal equations και της mean-squared που κάποια δείγματα τοποθετούνται πολύ κοντά στο desicion boundary, με αποτέλεσμα ένα να γίνεται misclassified.