ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ
ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

[THL 311]- Στατιστική Μοντελοποίηση & Αναγνώριση Προτύπων

Αναφορά 2ης εργαστηριακής άσκησης

Διδάσκων Θεωρίας: *Υπεύθυνος Εργαστηρίου:*

Ζερβάκης Μιχαήλ *Διακολουκάς Βασίλειος*

Ιωάννης Περίδης
A.M. 2018030069

1 Ιουλίου 2022

Περιεχόμενα & Οδηγίες Χρήσης:

Στην αναφορά αυτή, θα γίνει παρουσίαση και επεξήγηση των ασκήσεων 1,2,4,5 και 6 και των αποτελεσμάτων τους. Οι ασκήσεις 1,2 και 4 ,χρειάστηκαν να επιλυθούν στην MATLAB, ενώ οι ασκήσεις 5 και 6, υλοποιήθηκαν σε γλώσσα python. Οι κώδικες που υλοποιήθηκαν έχουν δωθεί σε 5 διαφορετικά αρχεία με ονόματα exercise2_1/2/4/5/6. Κάθε αρχείο MATLAB έχει μέσα πολλές συναρτήσεις και μόνο ένα εκτελέσιμο συνολικό αρχείο.m , με ονόματα My_logisticRegression για την άσκηση 1, ex2_2_regularizedLogisticRegression για την 2 και ex_4 για την 4. Στους φακέλους 5 και 6, περιέχονται τα αρχεία της python, με κατάληξη .py με ονόματα ex_2_5 για την άσκηση 5 και plots, fashion, και fashion_cnn για την 6.

Ο κώδικας είναι εμπλουτισμένος με σχόλια σε όλα τα σημεία για καλύτερη κατανόηση και μεγαλύτερη ευκολία ανάγνωσης. Για να τρέξετε τον κώδικα, πρέπει να παρακολουθείτε συνεχώς την κονσόλα για να βλέπετε τα αριθμητικά αποτελέσματα και τις γραφικές παρστάσεις στα figures. Μετά από τα αποτελέσματα του κάθε ερωτήματος η ροή του προγράμματος σταματάει με το pause και το πρόγραμμα περιμένει να πατήσετε ένα πλήκτρο για να συνεχίσει στα επόμενα.

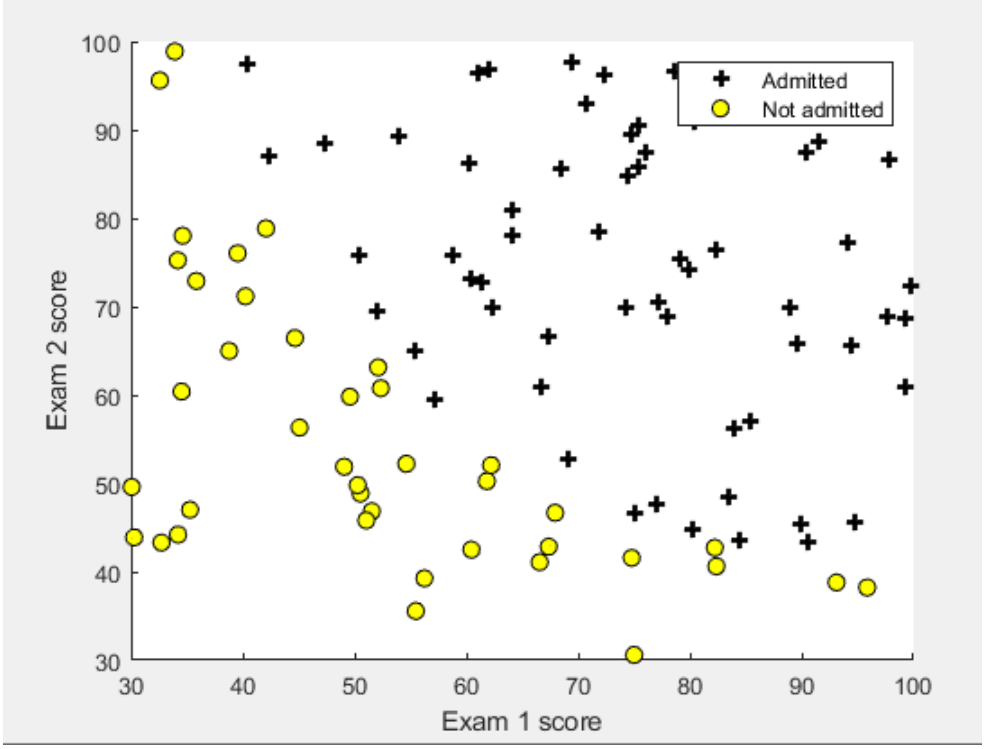
Πολλά υποερωτήματα από διάφορες ασκήσεις επιλύθηκαν με το χέρι στο χαρτί με μαθηματικό τρόπο και έχουν παραδοθεί σκαναρισμένες σε μορφή pdf σε ξεχωριστό φάκελο, με όνομα hand_written_exercises. Συγκεκριμένα, στο χαρτί λύθηκαν η άσκηση 1α, η 2α , ολόκληρες οι 3 και 4 και η 5 α,β,γ.

Θέμα 1: Λογιστική Παλινδρόμιση: Αναλυτική εύρεση κλίσης (Gradient).

Όπως προαναφέρθηκε το ερώτημα 1) α, υπολογίστηκε χειρόγραφα και θα βρεθεί στο αντίστοιχο pdf.Συγκεκριμένα, υπολογίστηκε με αναλυτικό τρόπο η εύρεση της κλίσης της συνάρτησης κόστους.

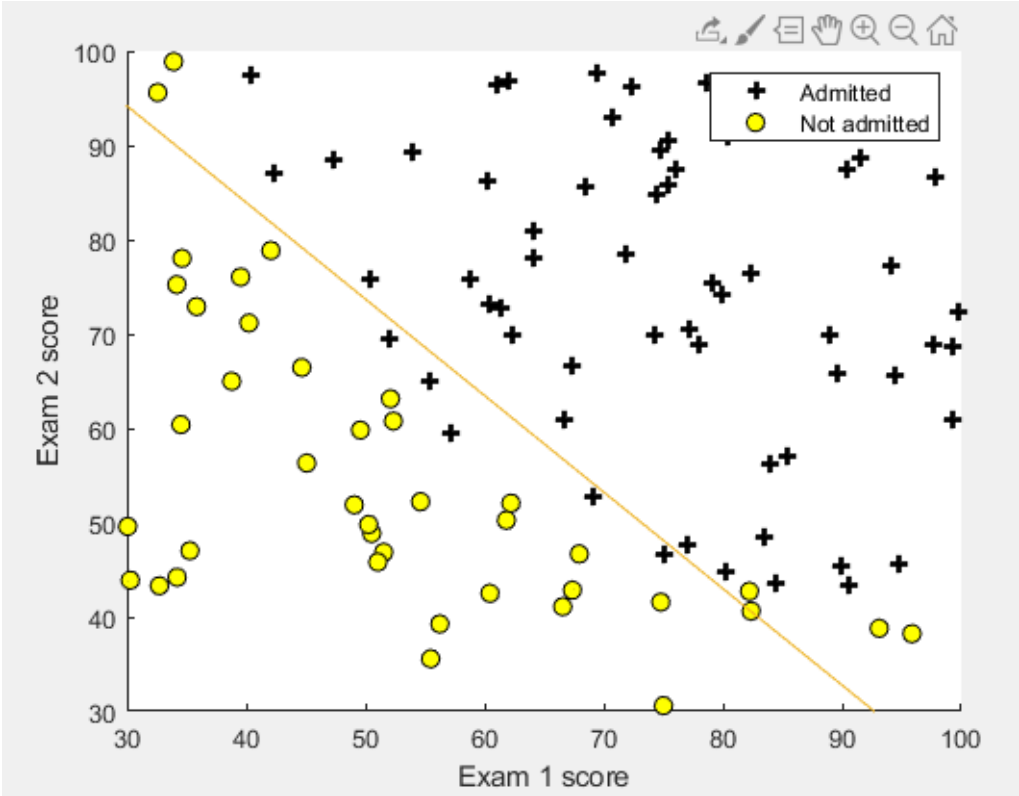
Στο ερώτημα 1)β μας ζητήθηκε η εφαρμογή της λογιστικής παλινδρόμισης πάνω σε ένα αρχείο εισόδου , που περιέχει βαθμούς φοιτητών. Σκοπός αυτού, ήταν η πρόβλεψη του αν θα γίνουν δεκτοί στο πανεπιστήμιο ή όχι.

Παρακάτω φαίνεται το dataset με τα αρχικά δεδομένα:



Συνεχίζοντας, ελαχιστοποιήθηκε η συνάρτηση κόστους cross entropy και υπολογίστηκαν οι παράμετροι θ. Έτσι, σχεδιάστηκε η ευθεία παλινδρόμισης , η οποία ορίζει τα δεδομένα. Για την πρόβλεψη του αν φοιτητής θα γίνει αποδεκτός ή όχι, με βάση τους βαθμούς του,χρησιμοποιήθηκαν η συνάρτηση sigmoid function και το θ που βρέθηκε.

Παρακάτω φαίνονται τα δεδομένα με την γρμαμή του ορίου απόφασης:



Παρατηρήθηκαν τ απαρακάτω αποτελέσματα:

```
Cost at initial theta (zeros): 0.693147
Gradient at initial theta (zeros):
 -0.100000
 -12.009217
 -11.262842
```

```
Cost at theta found by fminunc: 0.203498
theta:
 -25.161343
 0.206232
 0.201472
```

```
For a student with scores 45 and 85, we predict an admission probability of 0.776291

Train Accuracy: 89.000000
```

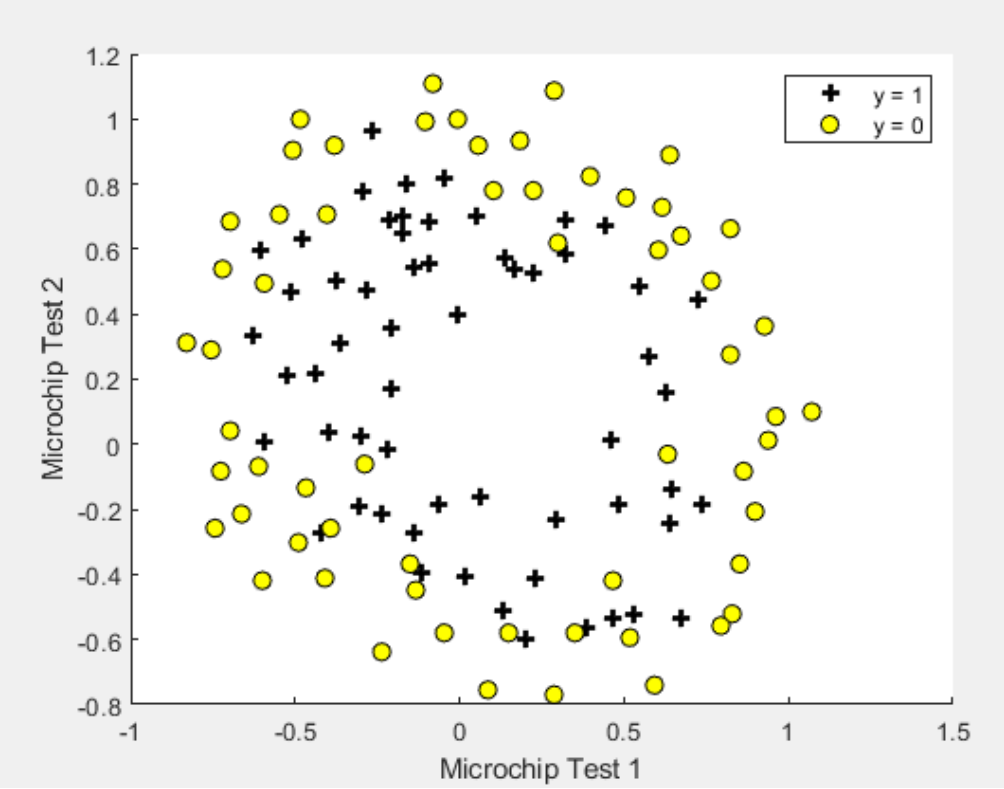
Αναλύοντας, φαίνεται πως το μοντέλο αυτό, πετυγχαίνει σωστή ταξινόμιση σε ποσοστό 89%, το οποίο είναι και αναμενόμενο από την θεωρία, αφού έχουμε γραμμικά διαχωρισμένα τα δεδομένα του dataset. Αυτό γίνεται ακόμα πιο αντιληπτό , παρατηρώντας την γραμμή ορίου διαχωρισμού, η οποία είναι τοποθετημένη με μεγάλη ακρίβεια.

Θέμα 2: Λογιστική Παλινδρόμιση με Ομαλοποίηση.

Αντίστοιχα με την 1, το ερώτημα 2) α, υπολογίστηκε χειρόγραφα και θα βρεθεί στο αντίστοιχο pdf.Συγκεκριμένα, υπολογίστηκε με αναλυτικό τρόπο η εύρεση της κλίσης της συνάρτησης κόστους.

Στο ερώτημα αυτό 2)β, μας ζητήθηκε η εφαρμογή της λογιστικής παλνδρόμισης με ομαλοποίηση, αυτή τη φορά σε δεδομένα εισόδου, τα οποία δεν είναι γραμμικώς διαχωρισμένα. Συγκεκριμένα, εφαρμόζεται πάνω σε 2 test από Microchips, με σκοπό την επίτευξη της quality analysis, δηλαδή μιας διαδικασίας επιβεβαίωσης ότι τα τσίπ δουλεύουν σωστά.

Παρακάτω φαίνεται το dataset με τα αρχικά δεδομένα:



Είναι φανερό και από την παραπάνω εικόνα, πως τα δεδομένα δεν είναι πλεόν γραμμικώς διαχωρισμένα, θα χρειαστεί λοιπόν, να προβληθούν σε έναν μεγαλύτερο χώρος διάστασης, με σκοπό να δημιουργηθεί ένα νέο σύνορο αποφάσεων. Για να συμβεί αυτό, θα πρέπει να προβληθούν, τα δεδομένα στον χώρο που θα οριστεί από όλους τους όρους των πολυωνύμων x1 και x2, μέχρι και τον 6ο βαθμό.

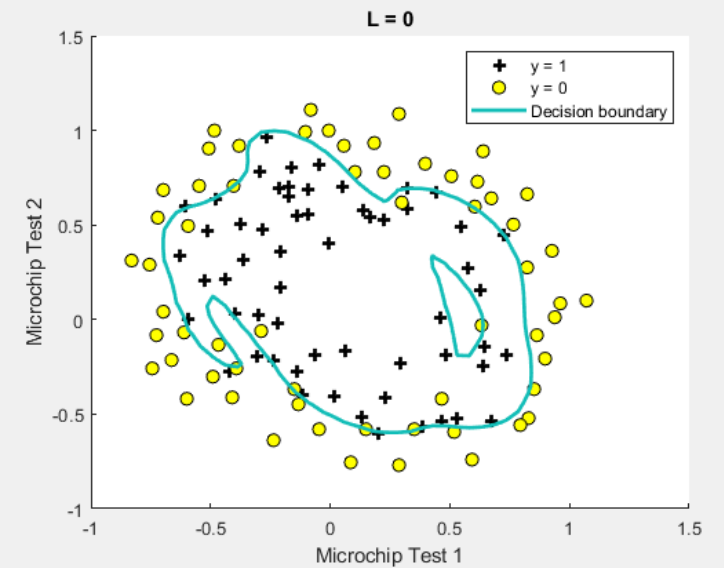
Συγκεκριμένα, χρησιμοποιήθηκαν οι παρακάτω σχέσεις για την δημιουργία του mapFeature:

$$P(x_1, x_2) = \sum_{i=0}^{6} \sum_{j=0}^{j} x_1^{i-j} x_2^{j} \quad \text{και}$$

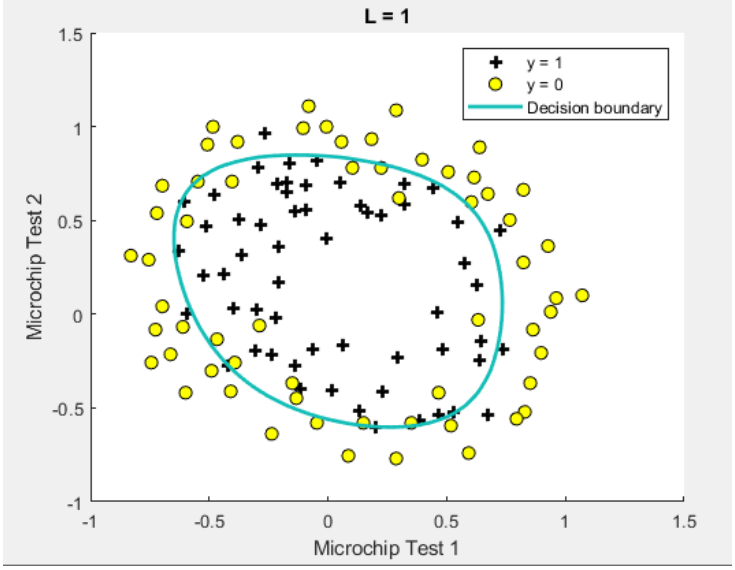
$$mapFeature(x) = [1, x_1, x_2, x_1^2, x_1 x_2, x_2^2, x_1^3, \ , \dots, x_1 x_2^5, x_2^6]^T$$

Ύστερα, με σκοπό την σχεδίαση και την εύρεση του καλύτερου σύνορου απόφασης, βελτιστοποιήθηκαν τα θ και καταγράφηκαν τα διαφορετικά σύνορα για πολλαπλές τιμές του λ.

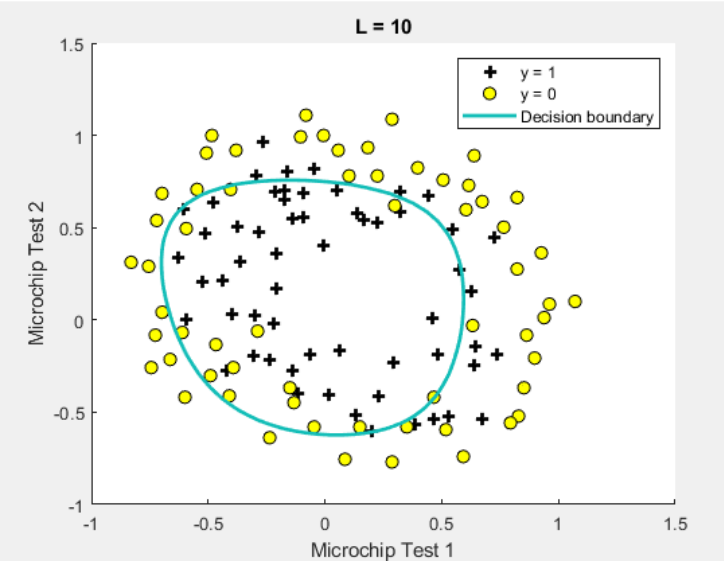
Decision Boundary : λ = 0
Accuracy = 88,983051%



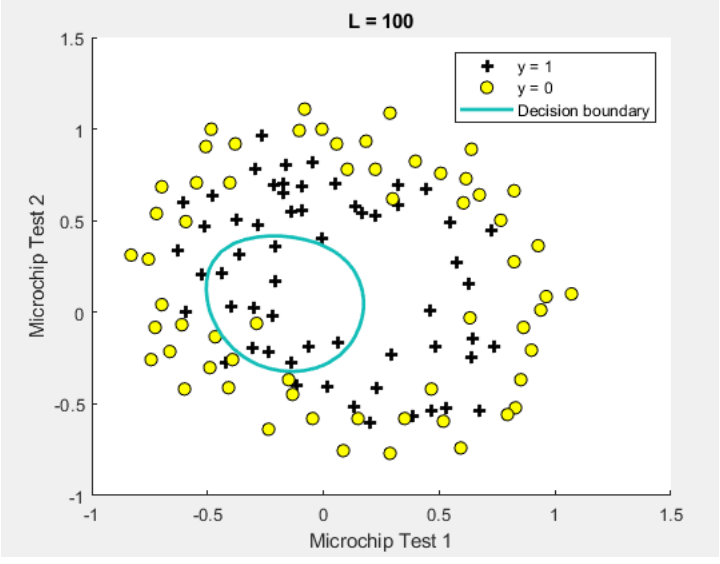
Decision Boundary : λ = 1
Accuracy = 82,203390%



Decision Boundary : λ = 10
Accuracy = 74,576271%



Decision Boundary : λ = 100
Accuracy = 60,169492%



Παρατηρώ τα εξής για κάθε διαφορετικό λ:

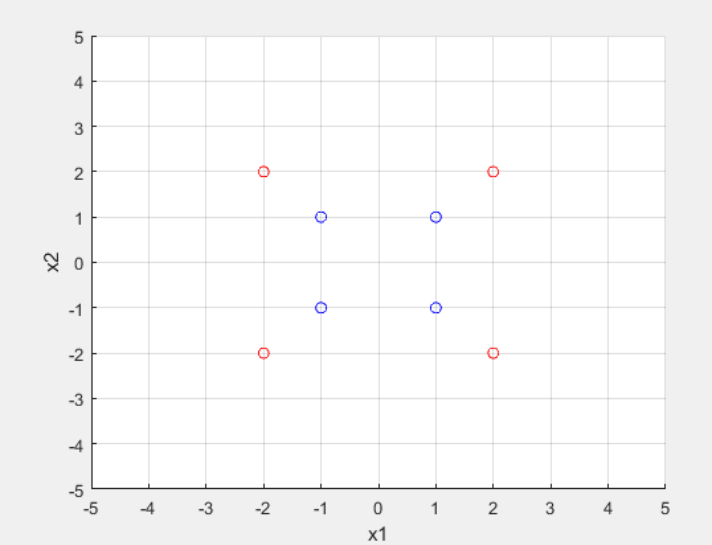
- Για **λ = 0**: Εδώ πετυχαίνω όπως είναι ξεκάθαρο και σχηματικά, το μεγαλύτερο accuracy, για το συγκεκριμένο τέστ εισόδων. Παρόλα αυτά, εμφανίζεται το φαινόμενο του overfitting, πάνω σε αυτό το συγκεκριμένο dataset. Δηλαδή, το σύνορο είναι εξαιρεστικά συγκεκριμένο και τέλεια εφαρμοσμένο στο training set. Αυτό έχει ώς αποτέλεσμα, να έχει πολύ κακή γενίκευση για όλα τα υπόλοιπα datasets, εκτός οτυ συγκεκριμένου.

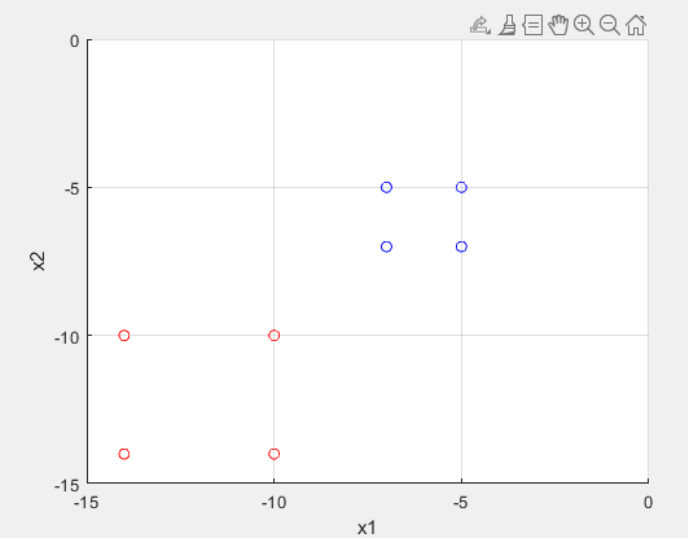
- Για **λ = 1 ή 10:** Για τις 2 τιμές αυτές, παρατηρώ πως έχω ένα αρκετά αξιόλογο accuracy, χρησιμοποιώντας 2 πολύ γενικά σύνορα απόφασης. Χωρίς δηλαδή να υπάρχουν, πολύ συγκεκριμένες μορφοποιήσεις για να επικαλυφθεί τέλεια το dataset. Επομένως, παρόλο που χάνω ένα μικρό ποσοστό αξιοπιστίας, πλέον δεν θα αντιμετωπίζω το πρόβλημα που αναφαίρθηκε παραπάνω, και θα μπορώ να έχω και αρκετά καλές γενικεύσεις σε άλλα datasets, εκτός του training.

- Για **λ = 100:** Για την τόσο μεγάλη τιμή αυτή, παρατηρούμε έντονα το αντίθετο φαινόμενο του overfitting, δηλαδή το underfitting. Αναλύοντας, το μοντέλο αυτό έχει μια πάρα πολύ κακή ακρίβεια , άρα δεν καταφέρνει επιτυχώς να μοντελοποιήσει το training set αυτό. Ακόμη, είναι φανερό πως δεν έχει και καμία πιθανότητα για να κάνει κάποια καλή γενίκευση.

Θέμα 4: Support Vector Machines (Maximum Likelihood).

Παρακάτω φαίνονται τα αποτελέσματα πριν και μετά την εφαρμογή του kernel trick:





Παρατηρείται πως πριν γίνει το kernel trick, δεν φαίνεται ξεκάθαρα ποιο είναι το γραμμικό επίπεδο διαχωρισμού. Αντιθέτως, αφού εφαρμοστεί γίνεται ξεκάθαρο. Όπως προαναφέρθηκε, η αναλυτική βελτιστοποίηση λύθηκε χειρόγραφα και είναι στο αντίστοιχο pdf.

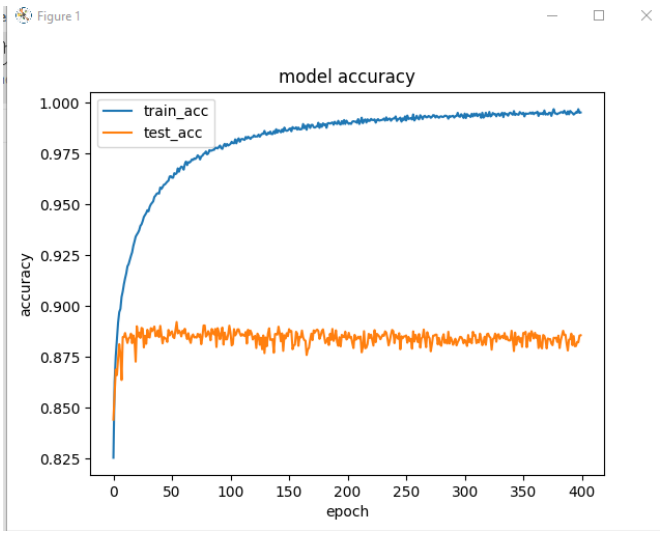
Θέμα 6: Convolutional Neural Networks for Image Recognition.

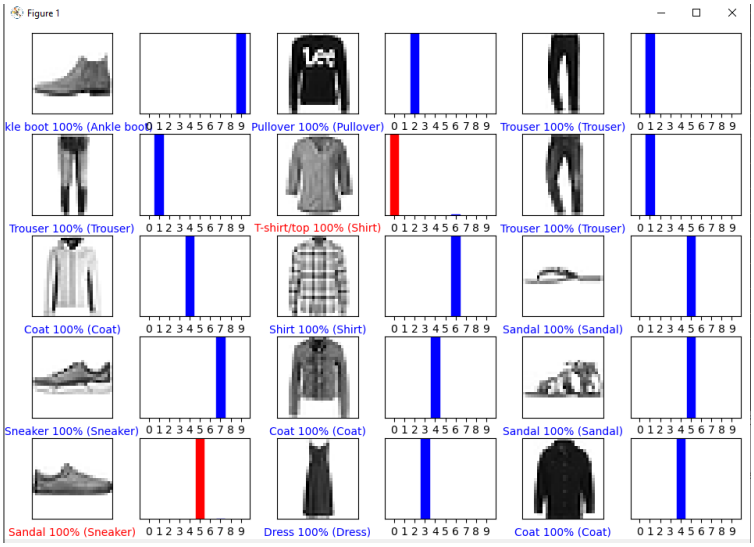
Στην άσκηση αυτή, θα αναλυθεί οι διαφορετικές αρχιτεκτονικές των ταξινομητών των Νευρωνικών δικτύων , χρησιμοποιώντας το σύνολο Fashion-MNIIST, του οποίου οι εικόνες, χωρίζονται σε 10 κατηγορίες που θα φανούν παρακάτω.

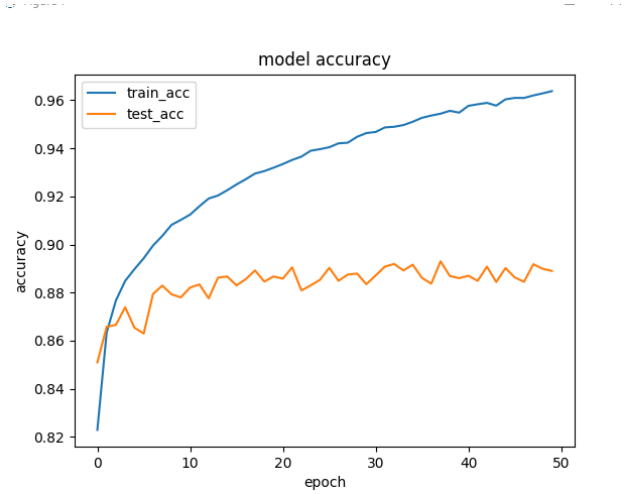
Ερώτημα 1)

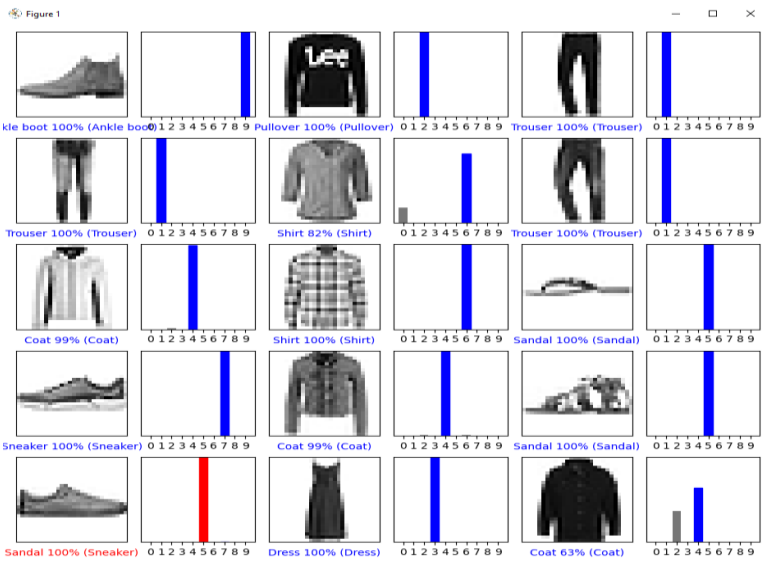
Αρχικά, χρησιμοποιήθηκε ο δοθέν κώδικας, που υλοποιεί έναν ταξινομητή με dense neural networks.

Παρακάτω φαίνονται οι γραφικές παραστάσεις του model accuracy για τιμές του epoch 400 και 50 αντίστοιχα, χρησιμοποιώντας αρχικά adam optimizer.Μαζί με κάθε εικόνα, δίνονται και τα αντίστοιχα αποτελέσματα για το σετ δεδομένων , για κάθε μια από τις εικόνες:









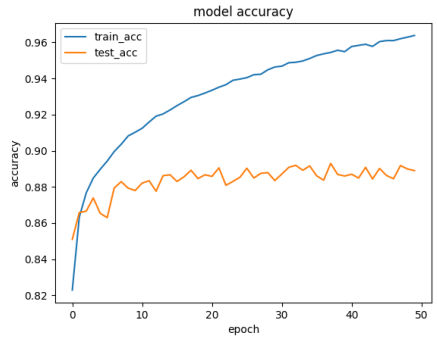
Παρατηρείται πως για αύξηση του epochs, από 50 σε 400, αυξάνεται φανερά το train_acc και μειώνεται ελάχιστα το test_acc. Το γεγονός αυτό οφείλεται στο ότι γνωρίζουμε πως όσο περισσότερο εκπαιδεύουμε ένα νευρωνικό δίκτυο, τόσο περισσότερο, γίνονται fit τα βάρη του στο training data set. Παρόλα αυτά, και στις δύο περιπτώσεις έχουμε εξαιρετικά καλά αποτελέσματα.

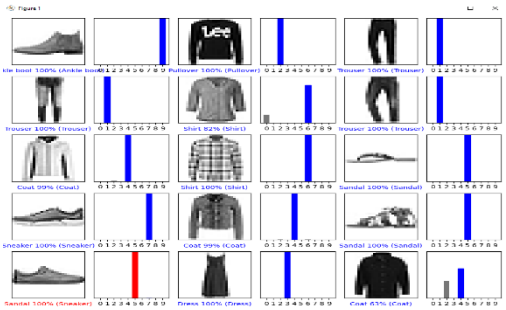
Ερώτημα 2)

Στο ερώτημα αυτό, δοκιμάστηκαν όλοι οι αλγόριθμοι βελτιστοποίησης και στο τέλος συγκρίθηκαν τα αποτελέσματά τους.

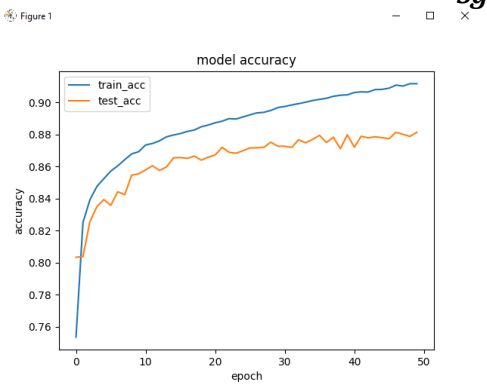
Παρακάτω φαίνονται οι γραφικές παραστάσεις των τιμών των accuracy για κάθε διαφορετικό optimizer, με 50 epochs. Αντίστοιχα φαίνονται και οι εικόνες από το data set:

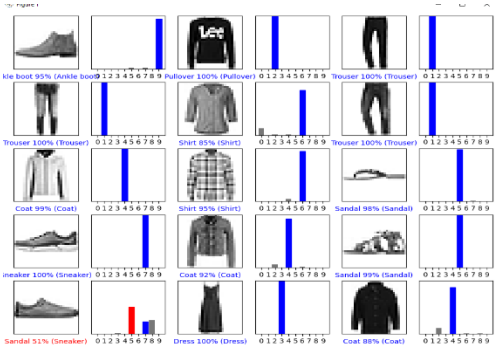
Adam:



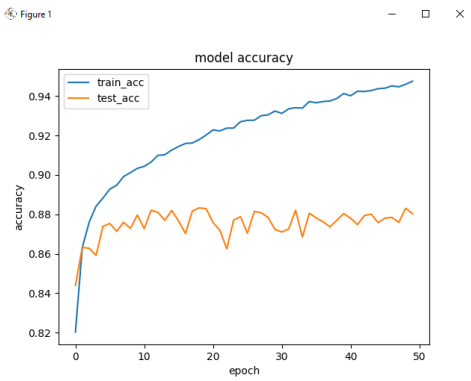


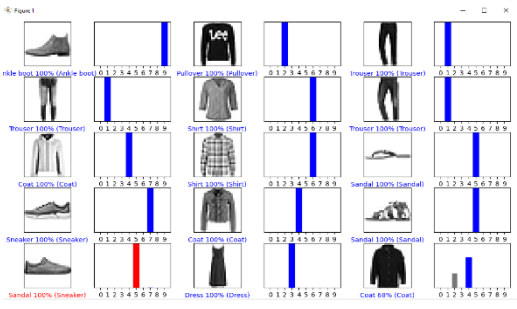
Sgd:



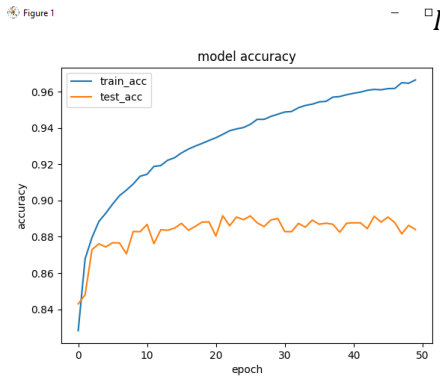


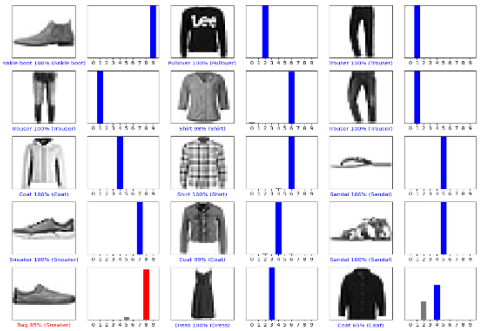
Rmsprop:



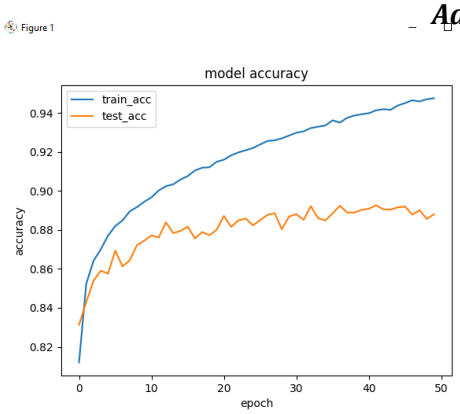


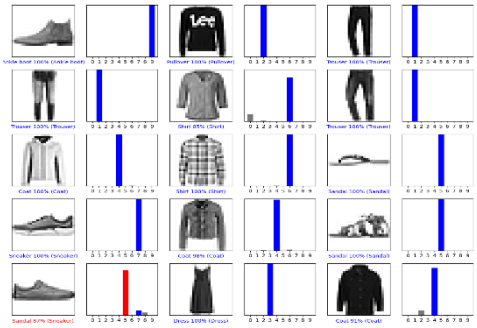
Nadam:



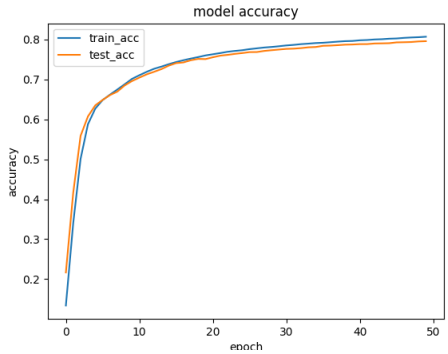


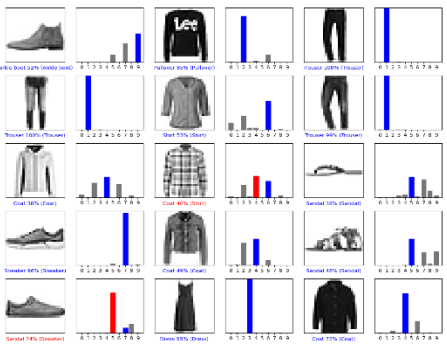
Adamax:





Ftrl:



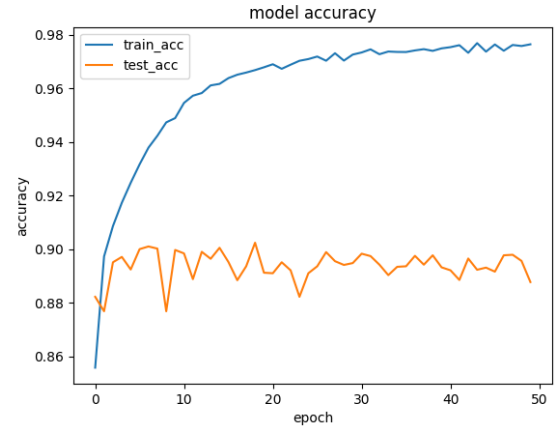


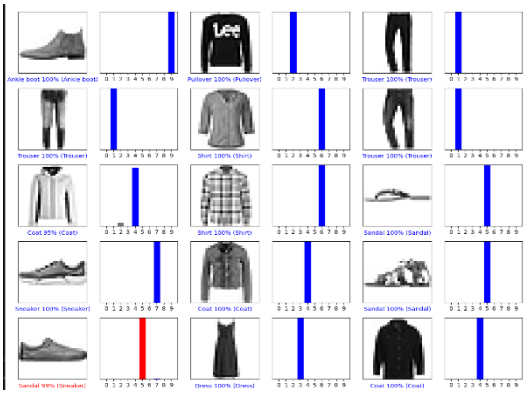
Παρατηρήθηκε από τις τιμές του train_acc και val_acc, πως οι 2 καλύτεροι και πιο αποδοτικοί αλγόριθμοι βελτιστοποίησης, είναι ο amdam και ο nadam, εφόσον αυτοί είχαν τις πιο υψηλές τιμές . Συγκεκριμένα, μεταξύ των δύο, ο adam είναι ο καλύτερος , γιατί ενώ το train_acc, του μοιάζει πολύ με εκείνο του nadam, έχει καλύτερο val_acc.

Ερώτημα 3)

Στο ερώτημα αυτό, έγινε αλλαγή της αρχιτεκτονικής του δικτύου, όπως ορίζει το σχήμα των προδιαγραφών στην εκφώνηση. Συγκεκριμένα, έγινε ανάλυση του προβλήματος της ταξινόμησης, χρησιμοποιώντας αυτή τη φορά Συνελικτικά Νευρωνικά Δίκτυα (CNNs).

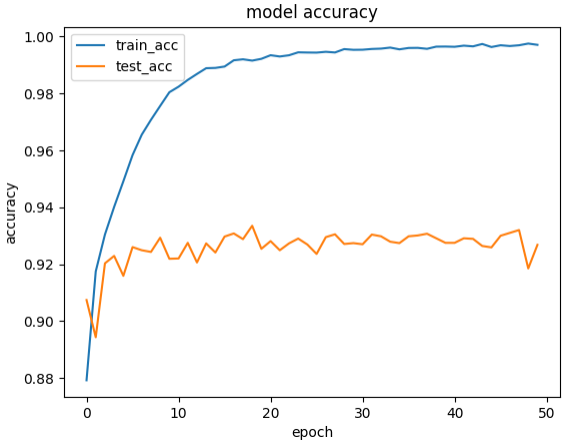
Παρακάτω φαίνεται η γραφική παράσταση του model accuracy του CNN και οι εικόνες του data-set:

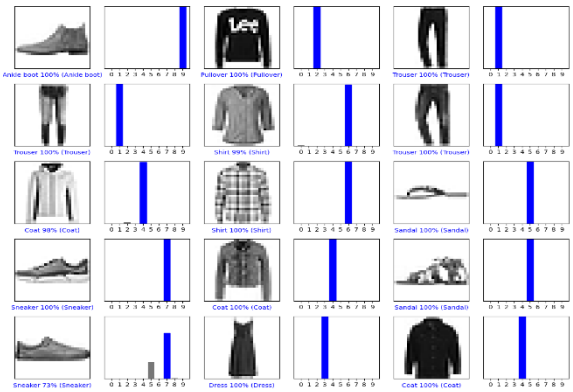




Ερώτημα 4)

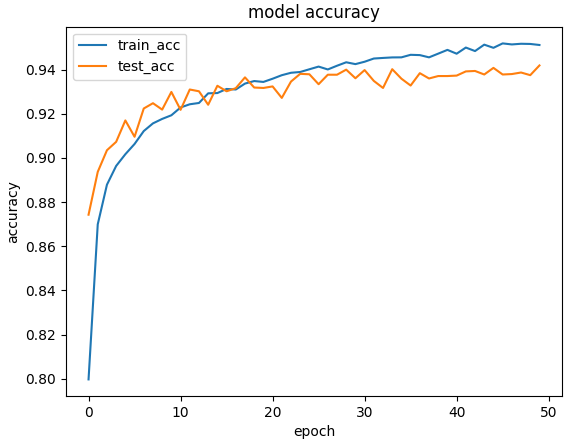
Παρακάτω φαίνεται η γραφική παράσταση του model accuracy του CNN και οι εικόνες του data-set, με προσθήκη Batch Normalization, μετά τα convs και το dense:

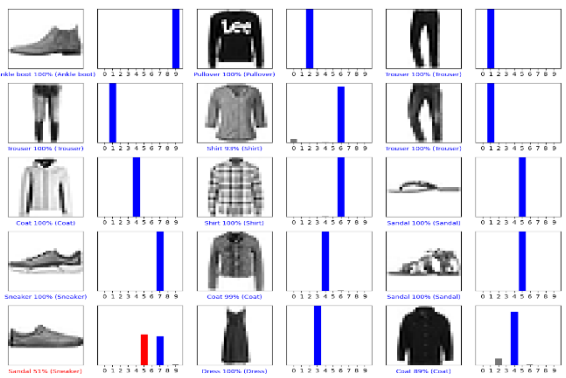




Ερώτημα 5)

Παρακάτω φαίνεται η γραφική παράσταση του model accuracy του CNN και οι εικόνες του data-set, με προσθήκη Dropout, μετά τα maxPoolings και πριν το dense:





Θέμα 5: Υλοποίηση ενός απλού νευρωνικού δικτύου.

Μέρος Α:

Η άσκηση αυτή, έχει λυθεί αναλυτικά στο χέρι, στο αντίστοιχο αρχείο pdf, (συγκεκριμένα, λύθηκαν τα ερωτήματα α,β,γ) παρόλα αυτά , γράφω κάποια πράγματα ξανά σαν συμπληρωματικά των λυμένων ερωτημάτων και γράφω και ολόκληρο και το ερώτημα δ. Δεν κατάφερα να ολοκληρώσω το μέρος β της άσκησης (οπότε το αρχείο exercise_5 , που χρειαζόταν τον κώδικα της Python, θα είναι κενό).

Στην άσκηση αυτή, μας ζητήθηκε να δημιουργηθεί ένα νευρωνικό δίκτυο, που θα χρησιμοποιηθεί για ταξινόμηση. Όπως δίνεται στην εκφώνηση σαν συνάρτηση ενεργοποίησης , μπορούμε να έχουμε την $f(z) = \frac{1}{1+e^{-z}}$.

Ενώ, η συνάρτηση κόστους που χρησιμοποιήθηκε, είναι ο μέσος όρος της cross entropy loss, πάνω σε batch , ίσο με B (δηλαδή B είναι το πλήθος των δειγμάτων), όπως φαίνεται παρακάτω:

$$J(Y, Y\hat{}\,; W, b) = \frac{1}{B}\sum(-y^{(i)} \ln\left(y\hat{}^{(i)}\right) - \left(1 - y^{(i)}\right) \ln\left(1 - y\hat{}^{(i)}\right))$$

Για το forward pass, υλοποιούμε την παρακάτω συνάρτηση:
$y\hat{}^{(i)} = f(x^{(i)}W + b)$

Ενώ, για το backward pass ,υλοποιούμε τις μερικές παραγώγους του J ως προς W και ως προς b:

$\frac{\partial J}{\partial W} = \frac{1}{B}\sum(y\hat{}^{(i)} - y^{(i)})x^{(i)r}$ και $\frac{\partial J}{\partial b} = \frac{1}{B}\sum(y\hat{}^{(i)} - y^{(i)})$