Στατιστική Μοντελοποίηση και Αναγνώριση Προτύπων

Φυλλάδιο Ασκήσεων 2

Νίκου Γεώργιος Νεκτάριος 2016030125

Θέμα 1

α)

$$h_\theta(x^{(i)}) = f(\theta_j x_j^{(i)}) = \frac{1}{1+e^{-\theta_j x_j^{(i)}}},$$

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{\partial}{\partial \theta_j}(\frac{1}{m}\sum_{i=1}^{m}(-y^{(i)}\ln(f(\theta_j x_j^{(i)})) - (1-y^{(i)}) * \ln(1-f(\theta_j x_j^{(i)}))))$$

$$= \frac{\partial}{\partial \theta_j}(\frac{1}{m}\sum_{i=1}^{m}(-y^{(i)}\ln(f(\theta_j x_j^{(i)})) - (1-y^{(i)}) * \ln(1-f(\theta_j x_j^{(i)}))))$$

$$= \frac{1}{m}\sum_{i=1}^{m}(-y^{(i)} * \frac{1}{f(\theta_j x_j^{(i)})} * (f(\theta_j x_j^{(i)}))' - (1-y^{(i)}) * (\frac{1}{1-f(\theta_j x_j^{(i)})} * (1-f(\theta_j x_j^{(i)}))'))$$

$$= \frac{1}{m}\sum_{i=1}^{m}(-y^{(i)} * \frac{(f(\theta_j x_j^{(i)}))'}{f(\theta_j x_j^{(i)})} + (1-y^{(i)}) * \frac{(f(\theta_j x_j^{(i)}))'}{1-f(\theta_j x_j^{(i)})})$$

$$= \frac{1}{m}\sum_{i=1}^{m}(-y^{(i)} * \frac{x_j^{(i)} e^{-\theta_j x_j^{(i)}}}{1+e^{-\theta_j x_j^{(i)}}} + (1-y^{(i)}) * \frac{x_j^{(i)}}{1+e^{-\theta_j x_j^{(i)}}})$$

$$= \frac{1}{m}\sum_{i=1}^{m}(x_j^{(i)} * h_\theta(x^{(i)}))[-y^{(i)} e^{-\theta_j x_j^{(i)}} + 1 - y^{(i)}])$$

$$= \frac{1}{m}\sum_{i=1}^{m}(x_j^{(i)} * h_\theta(x^{(i)}))[-y^{(i)}(1+e^{-\theta_j x_j^{(i)}}) + 1])$$

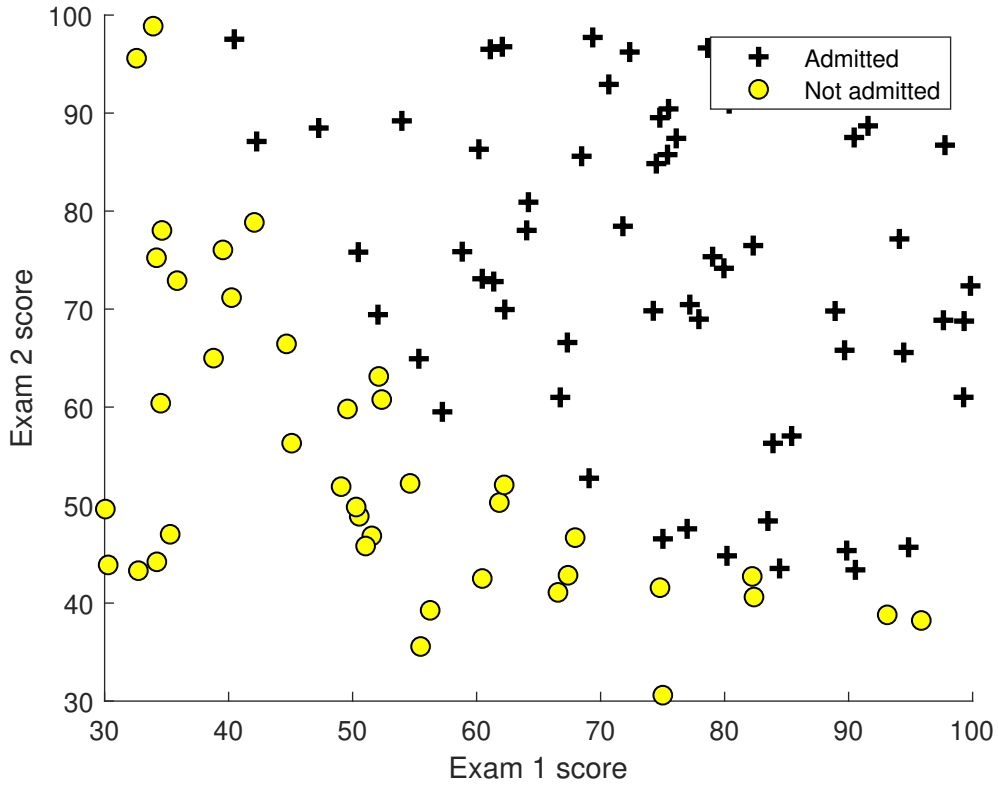
$$= \frac{1}{m}\sum_{i=1}^{m}(x_j^{(i)} * h_\theta(x^{(i)}))[-y^{(i)}\frac{1}{h_\theta(x^{(i)})} + 1])$$

$$= \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$

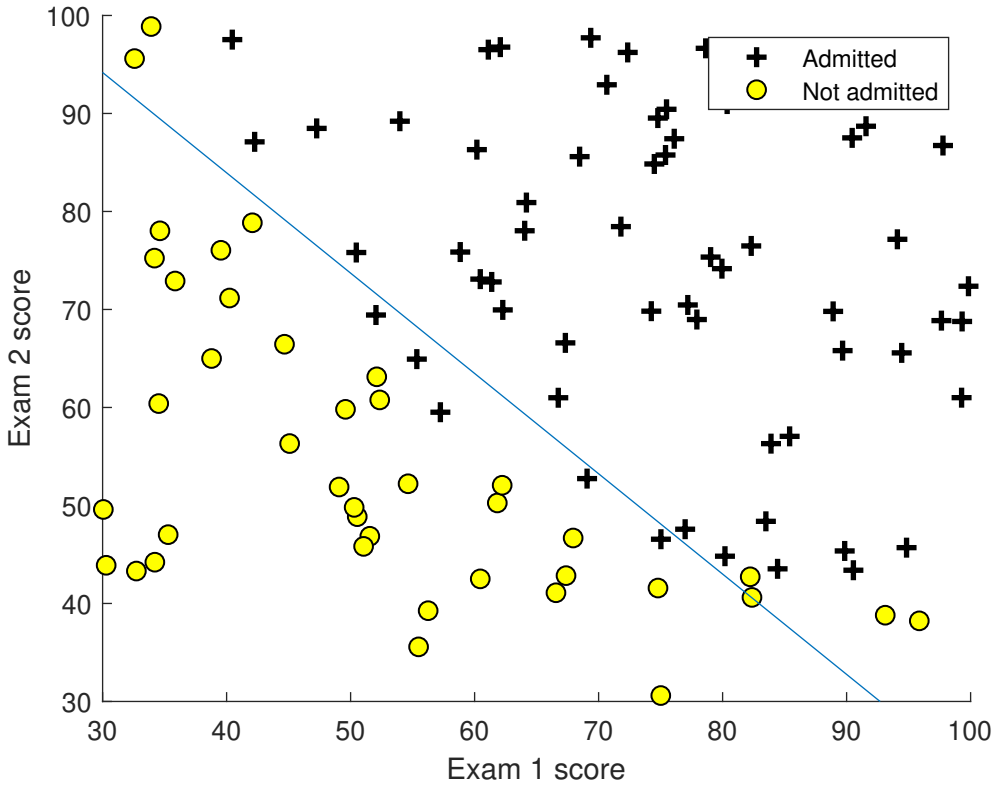
b)

Στο β ερώτημα αυτής της άσκησης εισάγονται δεδομένα φοιτητών από 2 διαφορετικές εξετάσεις και πρέπει να προβλέψουμε αν ο κάθε φοιτητής θα γίνει

δεκτός στο πανεπιστήμιο μέσω λογιστικής παλινδρόμισης. Παρακάτω απεικονίζονται οι βαθμοί των φοιτητών καθώς και το αν έχουν γίνει δεκτοί ή όχι, με σταυρό και κύκλο αντίστοιχα.



Χρησιμοποιώντας την σιγμοειδή συνάρτηση κάνουμε λογιστική παλινδρόμηση και βρίσκουμε το σύνορο απόφασης το οποίο διαχωρίζει τις δύο κλάσεις. Επιπλέον η συνάρτηση εφαρμόζει element-wise την σιγμοειδή όταν η είσοδος είναι πίνακας ενώ για τις υπόλοιπες περιπτώσεις εφαρμόζεται κανονικά.



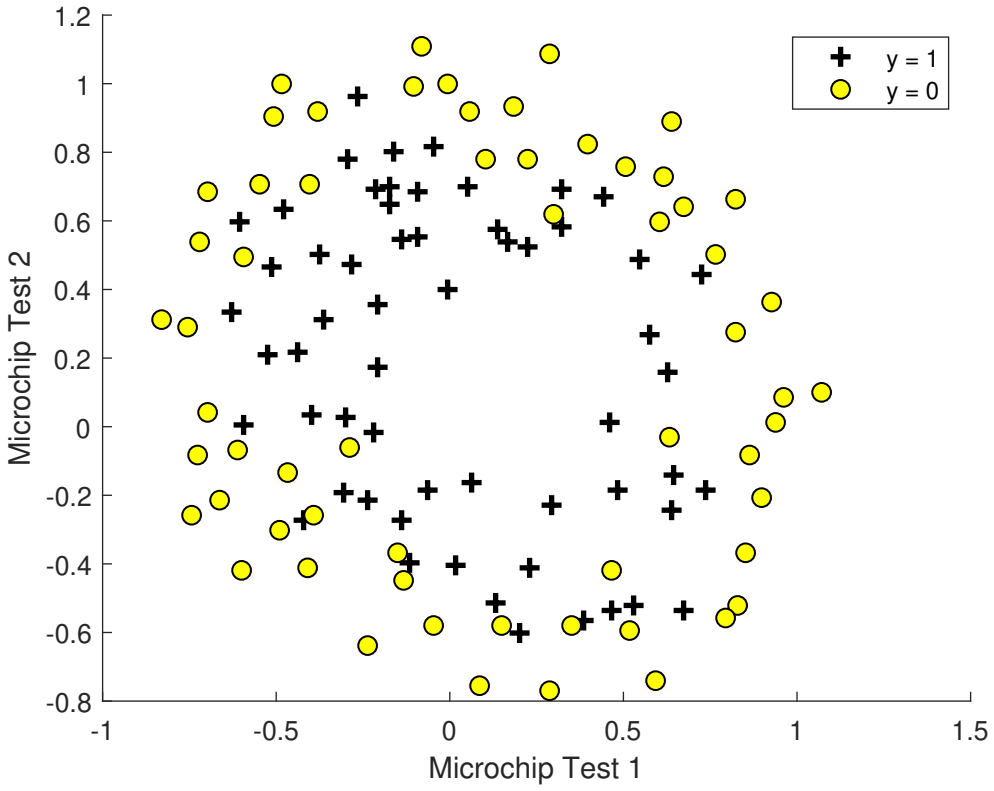
Μέσω της σιγμοειδής συνάρτησης υπολογίζουμε την πιθανότητα να δεχθεί το πανεπιστήμιο έναν φοιτητή που έχει σκορ 45 και 85 στα τεστ 1 και 2. Η πιθανότητα υπολογίζεται στο 77,62%. Τέλος με την συνάρτηση predict, καλώντας την σιγμοειδή συνάρτηση με threshold 0.5, υπολογίζουμε την ακρίβεια της λογιστικής παλινδρόμησης σε σχέση με τις αληθινές τιμές εισαγωγής.

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

Train Accuracy: 89.000000
```

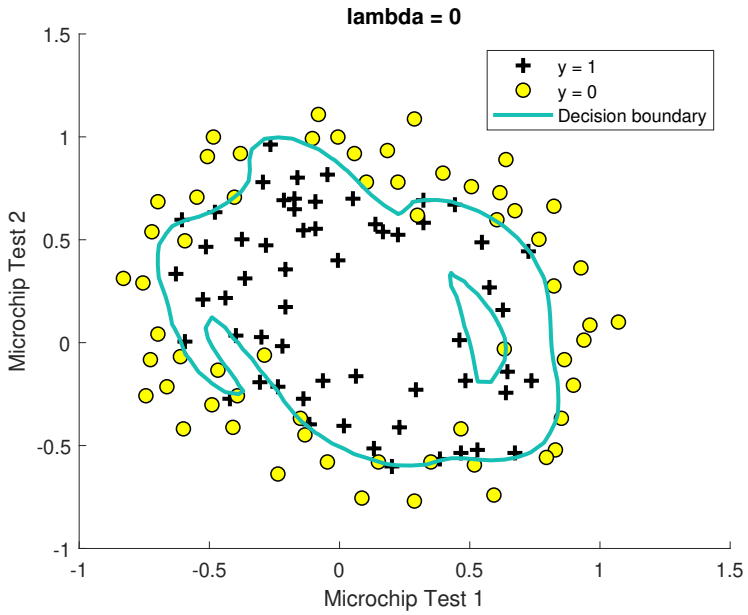
Θέμα 2

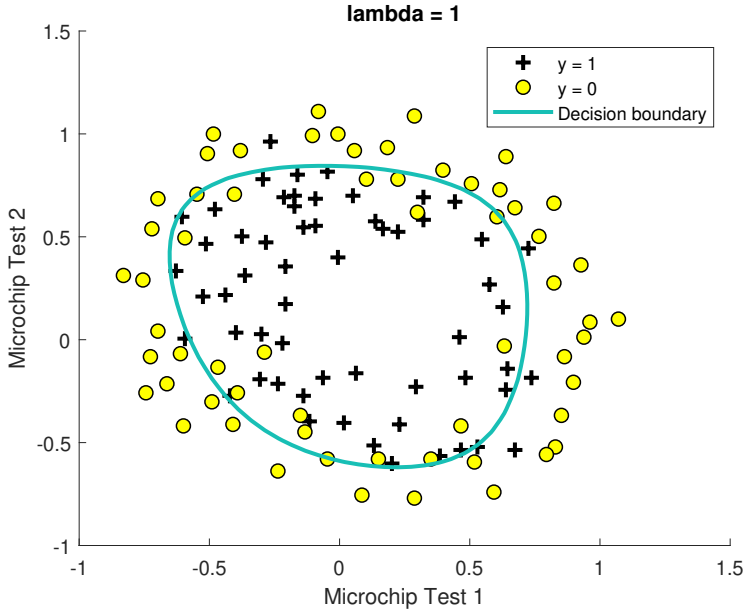
Στην δεύτερη άσκηση με την βοήθεια ομαλοποιημένης λογιστικής παλινδρόμησης πρέπει να προβλέψουμε με βάση 2 μετρικές αν τα μικροτσίπ περνάνε τον έλεγχο

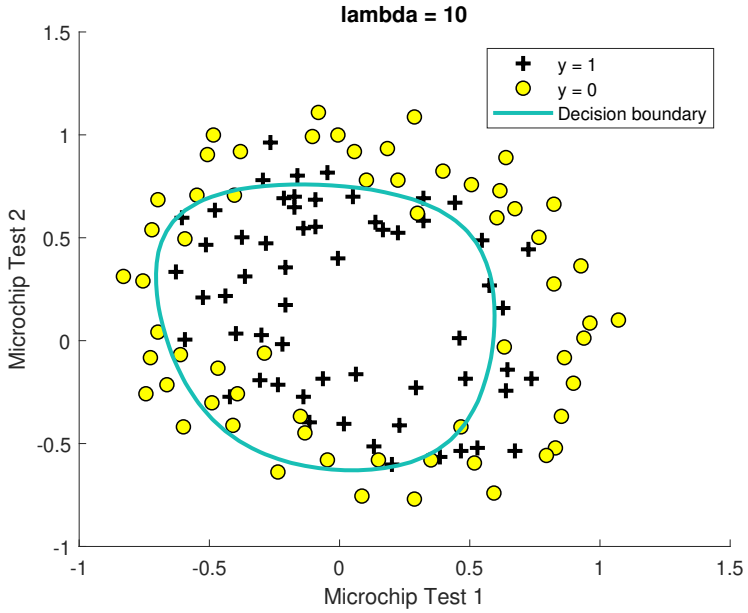
ποιότητας. Στο παρακάτω σχήμα φαίνονται τα αποτελέσματα των 2 τέστ για τα μικροτσίπ και το αν περνάνε τον έλεγχο ποιότητας, με σταυρό τα τσίπ που γίνονται αποδεκτά και κύκλο όσα απορρίπτονται.

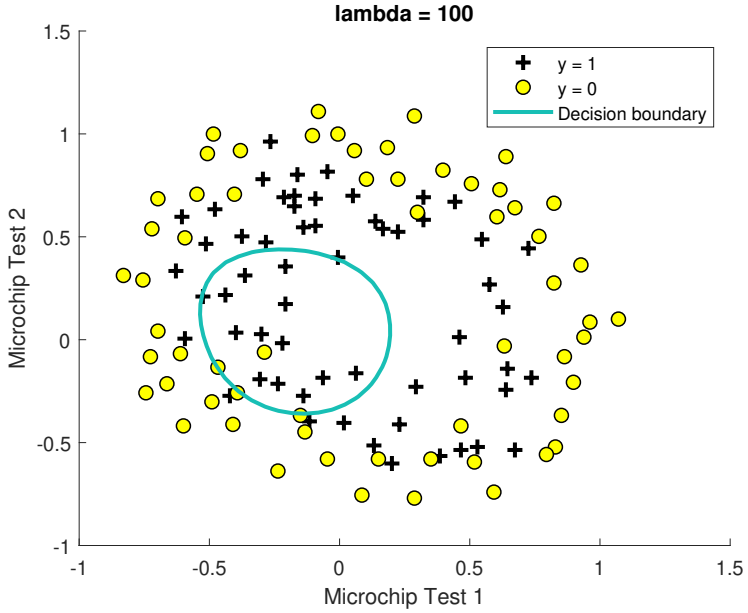


Όμως τα δεδομένα των 2 κλάσεων δεν διαχωρίζονται γραμμικά επομένως μετατρέπουμε τα δεδομένα σε χώρο μεγαλύτερης διάστασης για να διαχωριστούν ευκολότερα με την λογιστική παλινδρόμηση. Στην συνάρτηση mapFeature απεικονίζω τα γινόμενα πολυωνύμων x_1, x_2 μέχρι 6ο βαθμό. Η διαφορά με την λογιστική παλινδρόμηση της προηγούμενης άσκησης είναι η ομαλοποίηση της συνάρτησης κόστους και του gradient του σφάλματοςκαι την με βάση το λ και τις ϑ παραμέτρους. Στα επόμενα σχήματα απεικονίζονται τα δεδομένα και ο διαχωρισμός των κλάσεων για λ = 0,1,10,100.


lambda = 0
y = 1
y = 0
Decision boundary
Microchip Test 2
Microchip Test 1
1.5
1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1
1.5


lambda = 1
y = 1
y = 0
Decision boundary
Microchip Test 2
Microchip Test 1
1.5
1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1
1.5





Για $\lambda = 1$ έχουμε αρκετά καλό διαχωρισμό των κλάσεων. Για $\lambda = 10{,}100$ παρατηρώ underfitting ιδιαίτερα για λάμδα=100 όπου υπάρχει πολύ κακός δι-

αχωρισμός. Τέλος για λ = 0 υπάρχει overfitting στα training δεδομένα και θα έχει κακό generalization σε τυχαία δεδομένα. Το train accuracy για κάθε λάμδα είναι το εξής:
lambda = 0 : Train Accuracy: 88.983051
lambda = 1 : Train Accuracy: 81.355932
lambda = 10 : Train Accuracy: 74.576271
lambda = 100 : Train Accuracy: 60.169492

Θέμα 3

$$P(D|\lambda) = P(D|\lambda) = P(x_1, x_2, ..., x_n|\lambda) = \prod_{k=1}^{n} P(x_k|\lambda)$$

$$L_x(\lambda) = L(\lambda) = \ln\left(P(D|\lambda)\right) = \ln\left(\prod_{k=1}^{n} P(x_k|\lambda)\right) = \sum_{k=1}^{n} \ln\left(\frac{\lambda^{x_k} e^{-\lambda}}{x_k!}\right)$$

$$= \sum_{k=1}^{n} (\ln\left(\lambda^{x_k} e^{-\lambda}\right) - \ln\left(x_k!\right)) = \sum_{k=1}^{n} (\ln\left(\lambda^{x_k}\right) - \lambda - \ln\left(x_k!\right))$$

$$= -\lambda + \sum_{k=1}^{n} (x_k * \ln\left(\lambda\right) - \ln\left(x_k!\right))$$

$$\hat{\lambda}_{MLE} = \arg\max_{\lambda} P(D|\lambda) = \arg\max_{\lambda} L(\lambda)$$

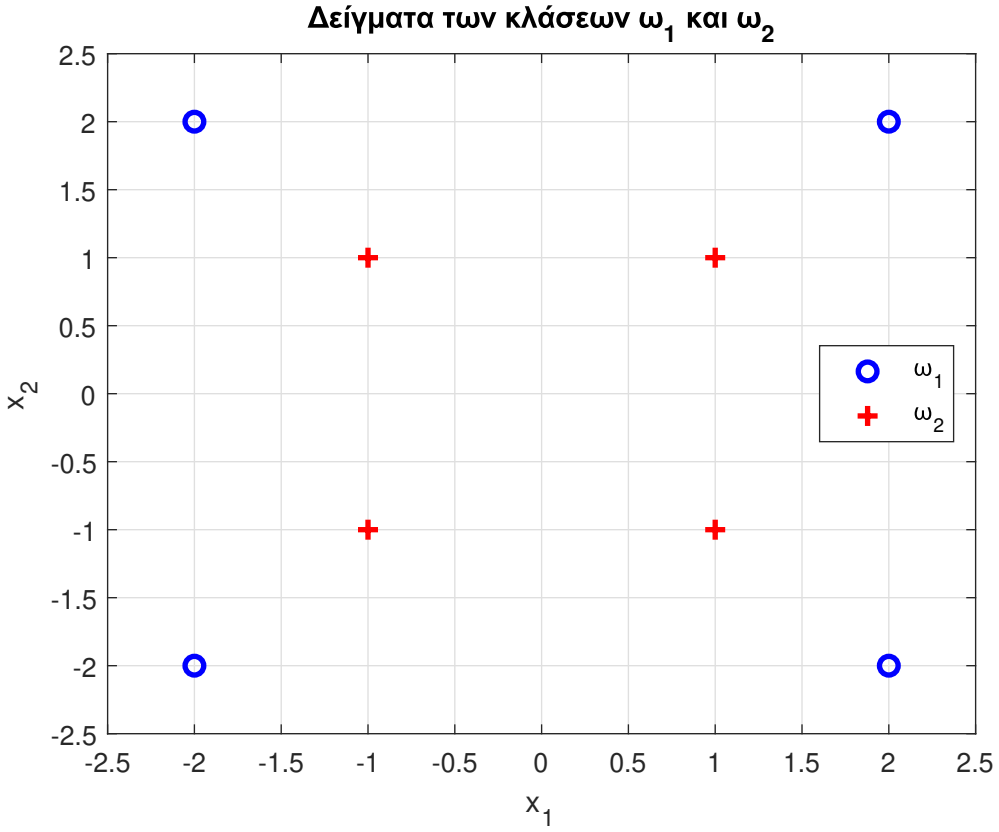
$$\frac{dL(\lambda)}{d\lambda} = 0 \Leftrightarrow \frac{d}{d\lambda}(-\lambda + \sum_{k=1}^{n} (x_k * \ln\left(\lambda\right) - \ln\left(x_k!\right))) = 0$$

$$\Leftrightarrow -1 + \frac{1}{\lambda} \sum_{k=1}^{n} x_k = 0 \Leftrightarrow$$

$$\hat{\lambda}_{MLE} = \sum_{k=1}^{n} x_k$$

Θέμα 4

1.



Τα support vectors ονομάζονται τα δείγματα που βρίσκονται πιο κοντά στην γραμμή διαχωρισμού. Στην συγκεκριμένη περίπτωση είναι όλα τα δείγματα εκπαίδευσης και των δύο κλάσεων καθώς έχουν ίση απόσταση από τα δείγματα της άλλης κλάσης και ακολούθως από την γραμμή διαχωρισμού. Τα δείγματα δεν είναι γραμμικά διαχωρίσιμα και τα δείγματα βρίσκονται διατεταγμένα με τέτοιο τρόπο που δεν μπορεί να εφαρμοστεί ένα soft margin. Επομένως δεν υπάρχει κάποιο γραμμικό υπερεπίπεδο διαχωρισμού που να διαχωρίζει τα δεδομένα των δύο κλάσεων και πρέπει να γίνει μετασχηματισμός των δεδομένων.

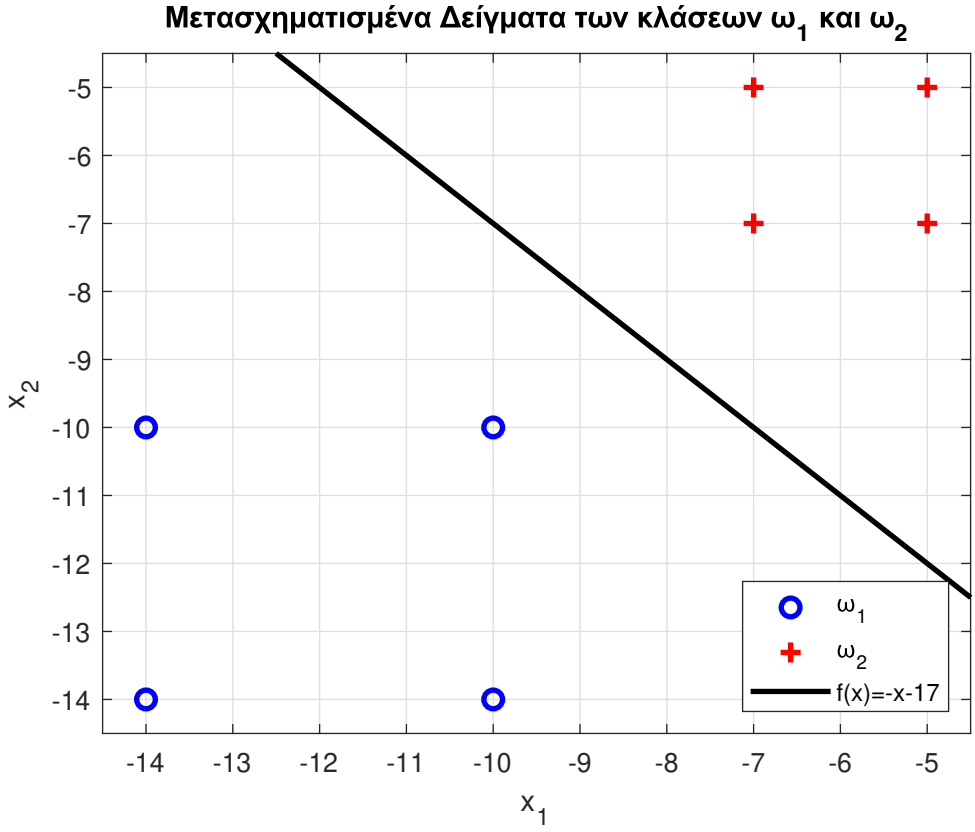
2.

$$\Phi(x) = x - ||x||^2 - 4$$
$$\omega_1 : ||x||^2 = x_1^2 + x_2^2 = 2^2 + 2^2 = 8$$
$$\omega_2 : ||x||^2 = x_1^2 + x_2^2 = 1^2 + 1^2 = 2$$

Άρα τα μετασχηματισμένα δισδιάστα δείγματα x=[x1,x2] για τις δύο κλάσεις είναι:

$$\omega_1 : x^+ = [-10, -10]^T, [-10, -14]^T, [-14, -14]^T, [-14, -10]^T$$
$$\omega_2 : x^- = [-5, -5]^T, [-5, -7]^T, [-7, -7]^T, [-7, -5]^T$$

Σαν γραμμικό επίπεδο διαχωρισμού θα πρότεινα την ευθεία που χωρίζει κάθετα τα 2 εγγύτερα δείγματα των κλάσεων, δηλαδή των δειγματων $\{\omega_1 : [-10, -10]^T, \omega_2 : [-7, -7]^T\}$. Η ευθεία που διαισθητικά προτείνω να διαχωρίζει τις κλάσεις είναι η $g(x) = -x - 17$



3.
Τα support vectors είναι τα δείγματα $\{\omega_1 : [-10,-10]^T, \omega_2 : [-7,-7]^T\}$. Πρέπει να σχειδάσω ένα υπερεπίπεδο διαχωρισμού που έχει την γενικότερη μορφή: $g(x) = w^T x + w_0 = 0$.
Η τιμή g(x) για τα support vectors x^+ για την κλάση ω_1 και x^- για την κλάση ω_2 να γίνεται $g(x^+) =< w * x^+ > +w_0 = 1, g(x^-) =< w * x^- > +w_0 = -1$.
Θα πρέπει επίσης να ισχύει ο περιορισμός
$w^T x^+ + w_0 \geq 1, x^+ \in \omega_1$
$w^T x^- + w_0 \leq 1, x^+ \in \omega_1$

Minimize: $J(w, w_0) = \frac{1}{2}(w_1^2 + w_2^2)$
Subject to: $f_i(w, w_0) = y_i(w^T x_i + w_0 - 1) \geq 0$, όπου $y_i = 1$ για την ω_1 και $y_i = -1$ για την ω_2.
$L(w, w_0, \lambda) = J(w, w_0) - \sum_{i=1}^{N} \lambda_i [y_i(w^T x_i + w_0) - 1]$

Οι εξισώσεις περιορισμού για τα support vectors είναι:

$$[-10,-10]^T : 1(-10w_1 - 10w_2 + w_0) \geq 1 \Leftrightarrow -w_1 - w_2 \geq \frac{1-w_0}{10}$$
$$[-7,-7]^T : -1(-7w_1 - 7w_2 + w_0) \geq 1 \Leftrightarrow w_1 + w_2 \geq \frac{1+w_0}{7}$$

Η συνάρτηση θα έχει 2 πολλαπλασιαστές $\lambda = [\lambda_1, \lambda_2]$:

$$L(w, w_0, \lambda) = \frac{w_1^2 + w_2^2}{2} - \lambda_1(-w_1 - w_2 + \frac{-1+w_0}{10}) - \lambda_2(w_1 + w_2 + \frac{-1-w_0}{7})$$

Οι συνθήκες KKT είναι:

$$\frac{\partial L(w, w_0, \lambda)}{\partial w_1} = 0 \Leftrightarrow w_1 = \lambda_2 - \lambda_1 \tag{1}$$
$$\frac{\partial L(w, w_0, \lambda)}{\partial w_2} = 0 \Leftrightarrow w_2 = \lambda_2 - \lambda_1 \tag{2}$$
$$\frac{\partial L(w, w_0, \lambda)}{\partial w_0} = 0 \Leftrightarrow \lambda_2 - \lambda_1 = 0 \Leftrightarrow \lambda_2 = \lambda_1 \tag{3}$$
$$\lambda_1(-w_1 - w_2 + \frac{-1+w_0}{10}) = 0 \tag{4}$$
$$\lambda_2(w_1 + w_2 + \frac{-1-w_0}{7}) = 0 \tag{5}$$
$$\lambda_1, \lambda_2 \geq 0 \tag{6}$$

$$(1,2) \xrightarrow{3} w_1 = w_2 = 0$$
$$(4) = (5) \rightarrow \lambda_1(\frac{-1+w_0}{10}) = \lambda_2(\frac{-1-w_0}{7})$$
$$\Leftrightarrow \frac{-1+w_0}{10} = \frac{-1-w_0}{7}$$
$$\Leftrightarrow 17w_0 = -17$$
$$w_0 = -1$$

Άρα η βέλτιστη γραμμή διαχωρισμού είναι η $g(x) = w_0 = -1$

Θέμα 5

Μέρος Α

α)

$$J(Y,\hat{Y};W,b) = \frac{1}{B}\sum_i(-y^{(i)} * \ln(f(z^{(i)}) - (1-y^{(i)}) * \ln(1-f(z^{(i)}))$$

$$= \frac{1}{B}\sum_i(-y^{(i)} * \ln(\frac{1}{1+e^{-z^{(i)}}}) - (1-y^{(i)}) * \ln(1-\frac{1}{1+e^{-z^{(i)}}}))$$

$$= \frac{1}{B}\sum_i(-y^{(i)} * -\ln(1+e^{-z^{(i)}}) - (1-y^{(i)}) * \ln(\frac{e^{-z^{(i)}}}{1+e^{-z^{(i)}}}))$$

$$= \frac{1}{B}\sum_i(y^{(i)} * \ln(1+e^{-z^{(i)}}) + (y^{(i)}-1) * (-z^{(i)} - \ln(1+e^{-z^{(i)}}))$$

$$= \frac{1}{B}\sum_i(y^{(i)} * \ln(1+e^{-z^{(i)}}) - y^{(i)} * z^{(i)} - y^{(i)} * \ln(1+e^{-z^{(i)}}) + z^{(i)} + \ln(1+e^{-z^{(i)}}))$$

$$= \frac{1}{B}\sum_i(y^{(i)} * (\ln(1+e^{-z^{(i)}}) - z^{(i)} - \ln(1+e^{-z^{(i)}})) + z^{(i)} + \ln(1+e^{-z^{(i)}}))$$

$$= \frac{1}{B}\sum_i(-y^{(i)} * z^{(i)} + z^{(i)} + \ln(1+e^{-z^{(i)}}))$$

β)

$$\begin{aligned}\frac{\partial J}{\partial z^{(i)}} &= \frac{\partial}{\partial z^{(i)}}(-y^{(i)} * z^{(i)} + z^{(i)} + \ln(1 + e^{-z^{(i)}})) \\ &= -y^{(i)} + 1 + \frac{-e^{-z^{(i)}}}{1 + e^{-z^{(i)}}} \\ &= -y^{(i)} + \frac{1 + e^{-z^{(i)}} - e^{-z^{(i)}}}{1 + e^{-z^{(i)}}} \\ &= -y^{(i)} + \frac{1 + e^{-z^{(i)}} - e^{-z^{(i)}}}{1 + e^{-z^{(i)}}} \\ &= -y^{(i)} + \frac{1}{1 + e^{-z^{(i)}}} \\ &= -y^{(i)} + f(z^{(i)}) \\ &= -y^{(i)} + \hat{y}^{(i)} \ , i \in batch\end{aligned}$$

γ)

$$\begin{aligned}\frac{\partial J}{\partial W} = \sum_i (\frac{\partial J}{\partial z^{(i)}} \frac{\partial z^{(i)}}{\partial W}) &= \sum_i ((-y^{(i)} + \hat{y}^{(i)}) * \frac{\partial}{\partial W}(x^{(i)} W + b)) \\ &= \sum_i ((-y^{(i)} + \hat{y}^{(i)}) * x^T)\end{aligned}$$

$$\begin{aligned}\frac{\partial J}{\partial b} = \sum_i (\frac{\partial J}{\partial z^{(i)}} \frac{\partial z^{(i)}}{\partial b}) &= \sum_i ((-y^{(i)} + \hat{y}^{(i)}) * \frac{\partial}{\partial b}(x^{(i)} W + b)) \\ &= \sum_i (-y^{(i)} + \hat{y}^{(i)})\end{aligned}$$

δ)

Γνωρίζοντας το σφάλμα στην έξοδο του δικτύου, υπολογίζεται η παράγωγος του σφάλματος σχέση με την έξοδο ($\frac{\partial E}{\partial Y}$) και ακολούθως τα βάρη και το bias

στο Layer 3. Για το backpropagate η έξοδος του Layer 3 γίνεται είσοδος για το Layer 2 και με τον κανόνα αλυσίδας υπολογίζω το $\frac{\partial E}{\partial H_2} = \frac{\partial E}{\partial Y} * \frac{\partial Y}{\partial H_2}$. Μέσω του κανόνα αλυσίδας ενημερώνονται τα βάρη και το bias στο Layer 2 με βάση το σφάλμα και παρομοίως γινεται η ίδια διαδικασία backpropagate/ενημέρωσης για το το Layer 1.
Για το layer 3 τα βάρη και το bias μεταβάλλονται ως εξής:

$$E = (-y^{(i)} * \ln(\hat{y}^{(i)}) - (1 - y^{(i)}) * \ln(1 - \hat{y}^{(i)})$$

$$\Phi(z) = \frac{1}{1 + e^{-}z}, d\Phi(z) = (1 - z) * z$$

$$\frac{\partial E}{\partial Y} = E \odot \Phi'(z_3)$$

$$\frac{\partial E}{\partial w_3} = \frac{\partial E}{\partial Y} * y_2^T$$

$$\frac{\partial E}{\partial b_3} = \frac{\partial E}{\partial Y}$$

$$w_3 = w_3 - p * \frac{\partial E}{\partial w_3}$$

$$b_3 = b_3 - p * \frac{\partial E}{\partial b_3}$$

Μέρος Β

Για το δεύτερο μέρος της άσκησης συμπλήρωσα τον κώδικα ώστε να φτιαχτεί ένα νευρωνικό δίκτυο το οποίο αναγνωρίζει και κατηγοριοποιεί χειρόγραφα ψηφία από την βάση δειγμάτων MNIST. Η αρχική υλοποίηση με learning rate = 0.1, batch size=128 και training epochs = 100 είχε accuracy ίσο με ratio: 0.85 και mse: 0.0218. Παρακάτω βρίσκονται τα αποτελέσματα των πειραματισμών για διάφορες τιμές σε παραμέτρους και αρχιτεκτονικές του νευρωνικού δικτύου.
1)

Παράμετροι νευρωνικού	accuracy	MSE
Αρχικό	0.85	0.0218
learning rate = 0.01	0.7	0.0409
learning rate = 0.3	0.85	0.0247
learning rate = 0.5	0.80	0.0298
training epochs = 20	0.65	0.0386
training epochs = 50	0.85	0.0238
training epochs = 200	0.75	0.0324
training epochs = 500	0.75	0.0393
batch size = 8	0.75	0.0471
batch size = 24	0.80	0.0347
batch size = 256	0.65	0.0453
batch size = 1024	0.15	0.1468

Αρχικά παρατηρώ ότι για τις αρχικές παραμέτρους έχουμε βέλτιστα αποτελέσματα. Για το learning rate η μείωση του για τα ίδια epochs δίνει αρκετά χειρότερα αποτελέσματα. Ενώ με την αύξηση του έχουμε ελάχιστα χαμηλότερο accuracy πιθανώς λόγω overfitting. Αντίστοιχα για training epochs=20,200,500 παράγονται χειρότερα αποτελέσματα λόγω underfitting/overfitting ενώ για training epochs = 50 έχουμε παρόμοιο accuracy με την αρχική αρχιτεκτονική. Τέλος όσο αυξάνεται το batch size υπάρχει σημαντική αύξηση στο error ενώ για μικρότερο batch size υπάρχει μια μικρή αύξηση.

2)Κάνω τα πειράματα με τις αρχικές τιμές σε learning rate, training epochs, batch size αλλά με διαφορετικούς συνδυασμούς activation function και error

Activation function, error	accuracy	MSE
Sigmoid + cross entropy	0.85	0.0218
Sigmoid + mse	0.20	0.16
Tanh + cross entropy	0.85	0.0251
Tanh + mse	0.00	0.1977

Από τους παραπάνω συνδυασμούς συμπεραίνω ότι το mean square error δίνει πολύ κακά αποτελέσματα για την συγκεκριμένη χρήση ενώ η χρήση του tanh σαν activation function δεν επηρεάζει ιδιαίτερα.

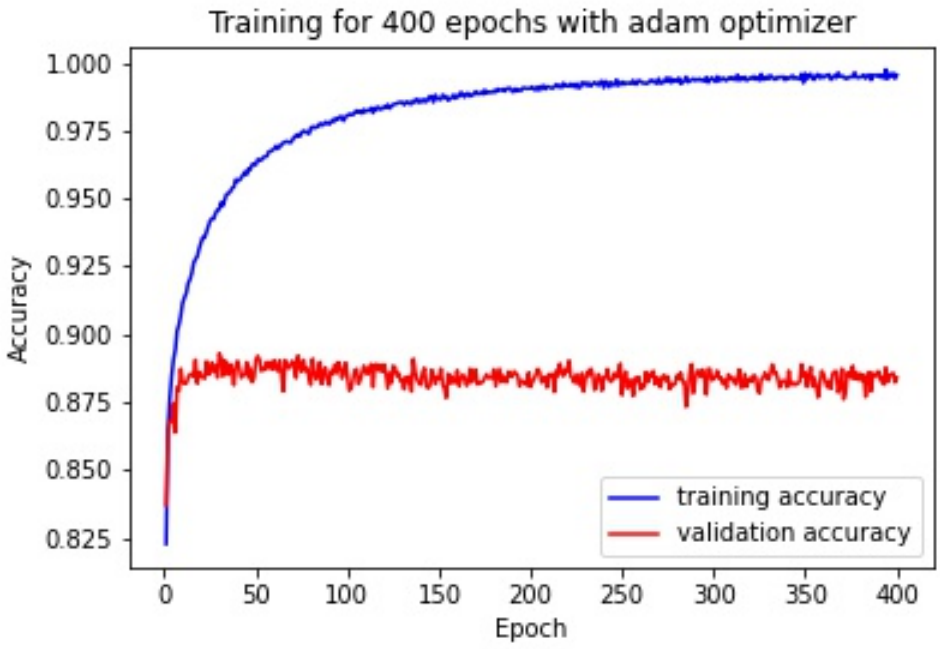
3) Παρακάτω είναι οι δοκιμές που πραγματοποίησα με διάφορες αρχιτεκτονικές δικτύου

Αρχιτεκτονική νευρωνικού	accuracy	MSE
Αρχικό	0.85	0.0218
2 layers	0.75	0.0285
2 layers with tanh activation func	0.85	0.0310
4 Dense Layers with Sigmoid and Tanh activation funcs	0.75	0.0418
1 layer with 100 neurons	0.75	0.0360
2 layers with 200 neurons, sigmoid and tanh activation functions	0.40	0.1183
4 layers with 300 neurons,sigmoid and tanh activation functions	0.10	0.18
4 layers with 300 neurons tanh activation function and mse error	0.15	0.17

Από τα παραπάνω αποτελέσματα συμπεραίνω ότι η αύξηση των νευρώνων/layers δεν παράγει απαραίτητα πιο βέλτιστα αποτελέσματα. Γενικότερα φαίνεται από το error ότι περισσότερο επηρεάζει αρνητικά η αύξηση των νευρώνων σε σχέση με την αύξηση των layers.

Θέμα 6

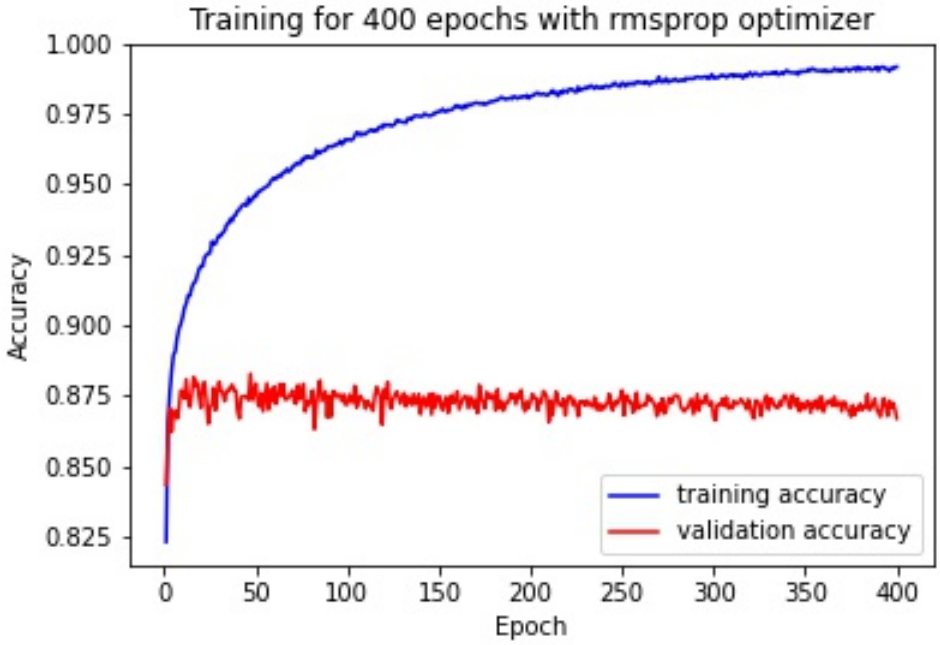
1.

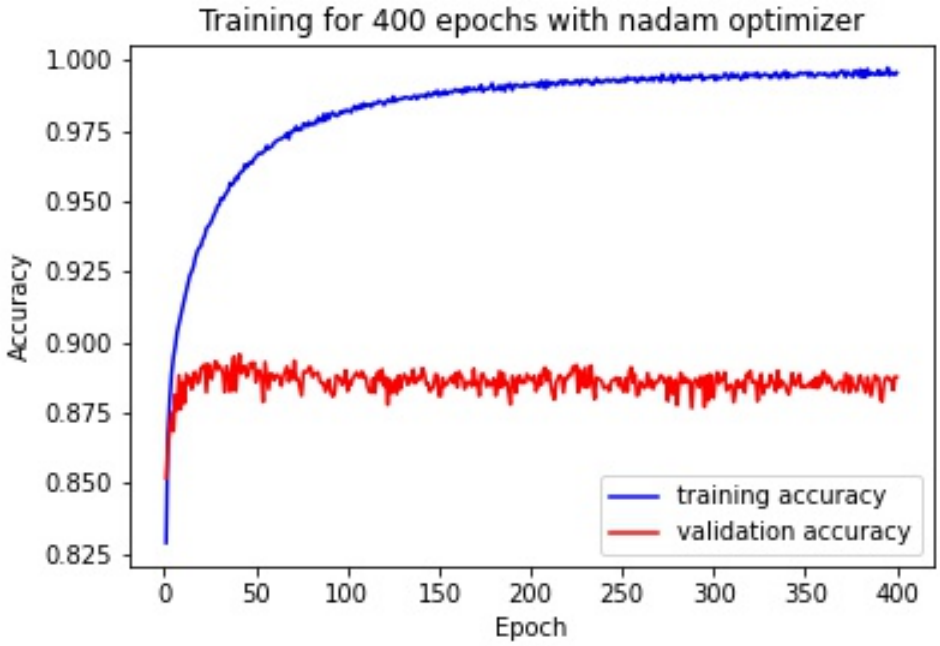


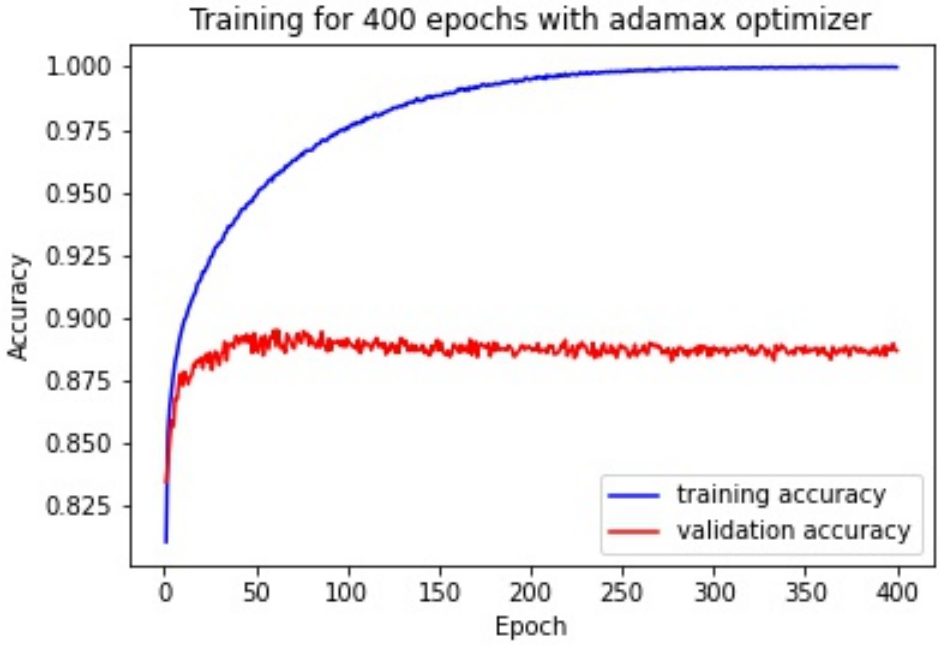
Το training accuracy ξεκινάει στο 0,83 στο epoch 1 και αυξάνεται σε κάθε epoch μέχρι μια τιμή μεγαλύτερη του 0,99 στο epoch 400. Το validation accuracy ξεκινάει στο 0,85 στο epoch 1 ακολουθεί μια μικρή άυξηση και σταθεροποιείται γρήγορα περίπου στο 0,875. Άρα συμπεραίνεται ότι το νευρωνικό δίκτυο προσαρμόζει τις μεταβλητές του πάνω στα δεδομένα του training αλλά δεν βελτιώνεται για τυχαία δεδομένα.

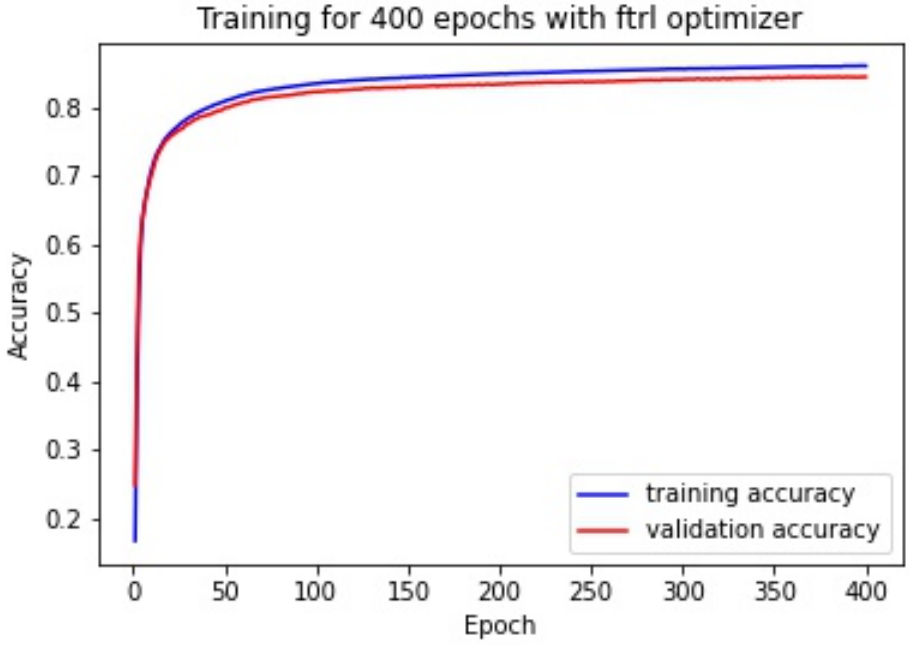
2.


Training for 400 epochs with sgd optimizer
1.00
0.95
0.90
0.85
0.80
0.75
Accuracy
0
50
100
150
200
250
300
350
400
Epoch
training accuracy
validation accuracy


Training for 400 epochs with rmsprop optimizer
1.000
0.975
0.950
0.925
0.900
0.875
0.850
0.825
Accuracy
0
50
100
150
200
250
300
350
400
Epoch
training accuracy
validation accuracy


Training for 400 epochs with nadam optimizer
1.000
0.975
0.950
0.925
0.900
0.875
0.850
0.825
Accuracy
0
50
100
150
200
250
300
350
400
Epoch
training accuracy
validation accuracy


Training for 400 epochs with adamax optimizer
Accuracy
Epoch
training accuracy
validation accuracy


Training for 400 epochs with ftrl optimizer
Accuracy
Epoch
training accuracy
validation accuracy

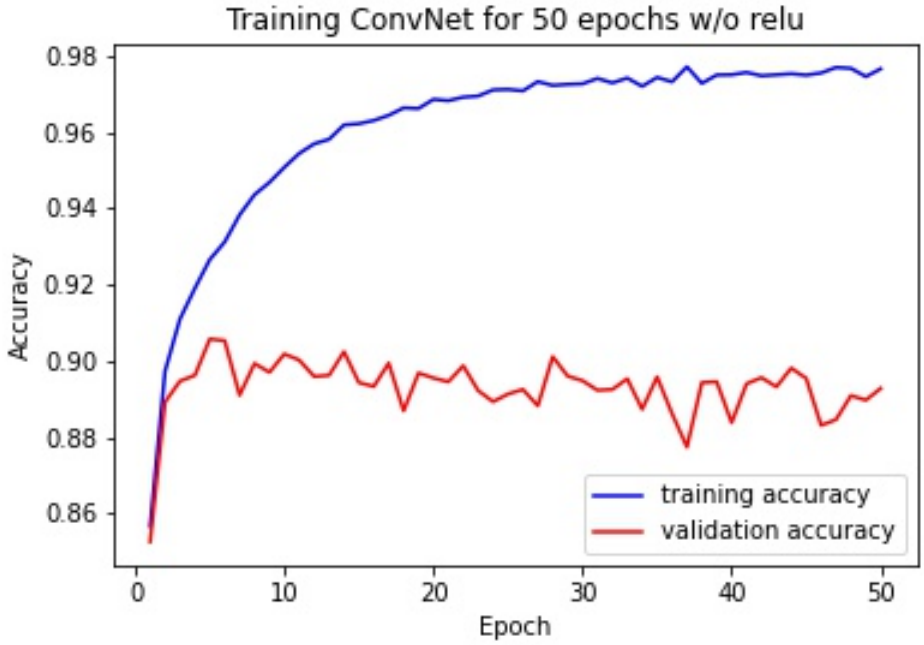
Το καλύτερο accuracy στο testing/validation το δίνει ο adamax με 0,8880, ενώ το χειρότερο έχει ο ftrl optimizer ο οποίος μάλιστα παρουσιάζει χαμηλό accuracy και στο training.

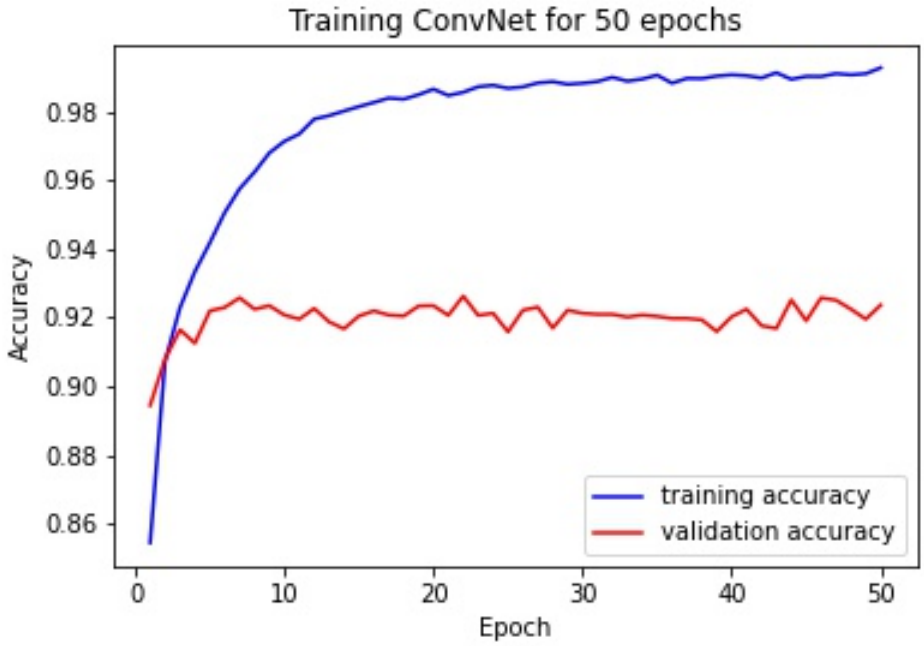
3.

Για αυτό το ερώτημα κατασκεύασα το νευρωνικό δίκτυο όπως ορίζει το σχήμα της άσκησης με την παρακάτω αρχιτεκτονική

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 28, 28, 32)        320

 conv2d_1 (Conv2D)           (None, 28, 28, 32)        9248

 max_pooling2d (MaxPooling2D  (None, 14, 14, 32)       0
 )

 conv2d_2 (Conv2D)           (None, 14, 14, 64)        18496

 conv2d_3 (Conv2D)           (None, 14, 14, 64)        36928

 max_pooling2d_1 (MaxPooling  (None, 7, 7, 64)         0
 2D)

 conv2d_4 (Conv2D)           (None, 7, 7, 128)         73856

 max_pooling2d_2 (MaxPooling  (None, 3, 3, 128)        0
 2D)

 flatten (Flatten)           (None, 1152)              0

 dense (Dense)               (None, 200)               230600

 dense_1 (Dense)             (None, 10)                2010

=================================================================
Total params: 371,458
Trainable params: 371,458
Non-trainable params: 0
```

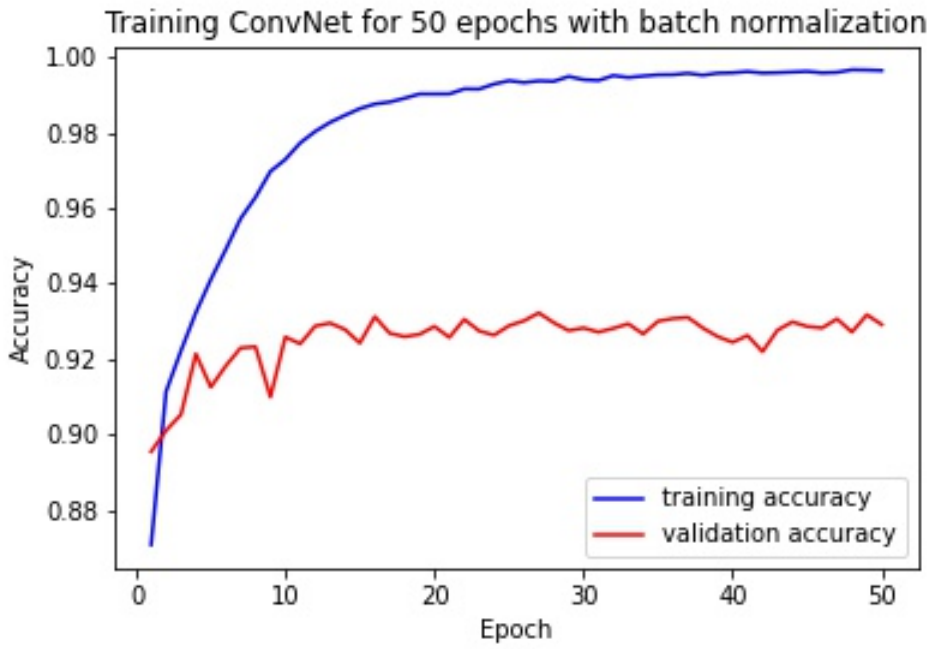
Σε αυτή την αρχιτεκτονική δοκίμασα υλοποίηση χωρίς activation function και υλοποίηση με ReLU activation function.


Training ConvNet for 50 epochs w/o relu
Accuracy
Epoch
training accuracy
validation accuracy
0.98
0.96
0.94
0.92
0.90
0.88
0.86
0
10
20
30
40
50

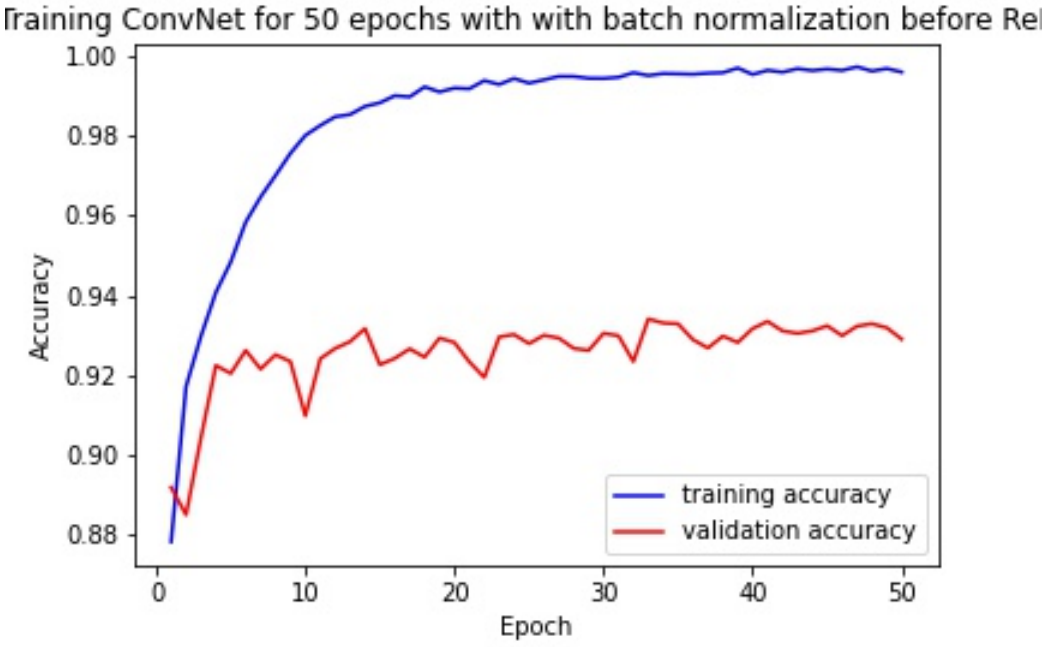

Training ConvNet for 50 epochs
Accuracy
Epoch
training accuracy
validation accuracy
0.98
0.96
0.94
0.92
0.90
0.88
0.86
0
10
20
30
40
50

Συγκρίνοντας τις δύο υλοποιήσεις συμπεραίνω ότι η έλλειψη activation function οδηγεί σε χειρότερο accuracy, παρόμοιο με αυτή με το ένα layer, και μεγαλύτερες διακυμάνσεις σε σχέση με την υλοποίηση με ReLU. Για την δεύτερη υλοποίηση με activation function παρατηρώ ότι υπάρχει σημαντική βελτίωση σε σχέση με την προηγούμενη αρχιτεκτονική με ένα layer. Συγκεκριμένα με τον adamax optimizer η προηγούμενη αρχιτεκτονική είχε accuracy 0,8880 ενώ η παρούσα αρχιτεκτονική πέτυχε accuracy 0,923. Βέβαια με την πιο περίπλοκη αρχιτεκτονική υπάρχει μεγαλύτερη πολυπλοκότητα άρα και μεγαλύτερο κόστος σε χρόνο.

4.

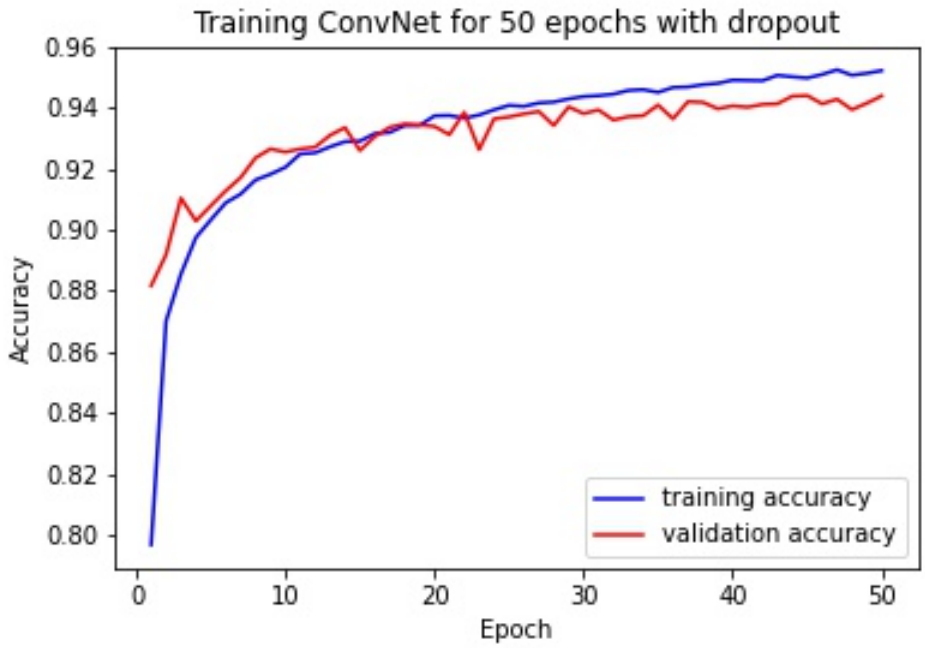


Με την προσθήκη batch normalization μετά τα conv layers καθώς και το ένα dense layer στο τέλος παρατηρείται μια μικρή άυξηση στο accuracy σε σχέση με τον προηγούμενο σχεδιασμό, δηλαδή πετυχαίνει accuracy 0,929.



Εναλλάσσοντας το Batch Normalization με το activation function παρατηρώ παρόμοιο accuracy 0,928 στο testing. Ωστόσω υπάρχουν και μεγαλύτερες διακυμάνσεις στο validation accuracy, ειδικά στα πρώτα epochs, σε σχέση με τον προηγούμενο σχεδιασμό επομένως θα προτιμούσα τον πρώτο.

5.



Τέλος με την προσθήκη dropout, δηλαδή την τυχαία απενεργοποίηση νευρώνων για την αποφυγή overfitting, πετυχαίνονται εντυπωσιακά αποτελέσματα. Το testing accuracy φτάνει στο 0,9439 φτάνοντας πολύ κοντά στο training accuracy, το οποίο μάλιστα στα αρχικά epochs το ξεπερνάει. Ένα σημαντικό στοιχείο είναι ότι το training accuracy φτάνει μέχρι το 0,95 ενώ σε προηγούμενες υλοποιήσεις ξεπερνούσε το 0,99. Αυτο σημαίνει ότι η προσθήκη του dropout περιορίζει σημαντικά το overfitting.