ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

Σχολή ΗΜΜΥ

Ομάδα 5 – Αγγελόπουλος Δημήτριος 2020030038

Στατιστική Μοντελοποίηση και Αναγνώριση Προτύπων (ΤΗΛ311)

2η Σειρά Ασκήσεων

Θέμα 1

Σε αυτό το θέμα θα βρούμε μία εκτίμηση των $y^{(i)} \in \{0,1\}$ με βάση ενός συνόλου δεδομένων $D = \{(\underline{x}^{(1)}, y^{(1)}), (\underline{x}^{(2)}, y^{(2)}), \dots, (\underline{x}^{(m)}, y^{(m)})\}$ χρησιμοποιώντας την συνάρτηση λογιστικής παλινδρόμησης.

$$f(z) = \frac{1}{1+e^{-z}}$$

Η εκτίμηση δίνεται από τον τύπο $\hat{y} = h_\theta(\underline{x}) = f(\underline{\theta}^T \underline{x})$. Σκοπός είναι να βελτιστοποιήσουμε την εκτίμηση μας ορίζοντας μία συνάρτηση κόστους και στην συνέχεια ελαχιστοποιώντας την. Η συνάρτηση κόστους που θα χρησιμοποιήσουμε θα είναι η cross-entropy :

$$J(\underline{\theta}) = \frac{1}{m}\sum_{i=1}^{m}\left(-y^{(i)}\ln(\hat{y}^{(i)}) - (1-y^{(i)})\ln(1-\hat{y}^{(i)})\right)$$

Για ελαχιστοποίηση θα βρούμε το gradient* της

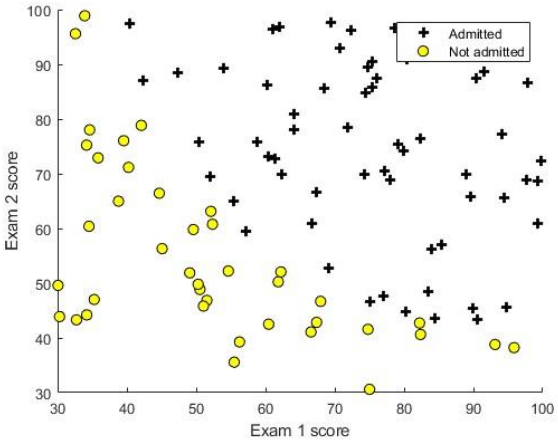
$$\nabla_{\underline{\theta}} J = \frac{1}{m}\sum_{i=1}^{m}\left(\left(-y^{(i)} + h_\theta(\underline{x}^{(i)})\right)\underline{x}^{(i)}\right)$$

Επομένως για το j-οστό στοιχείο του $\underline{\theta}$ η παράγωγος θα είναι

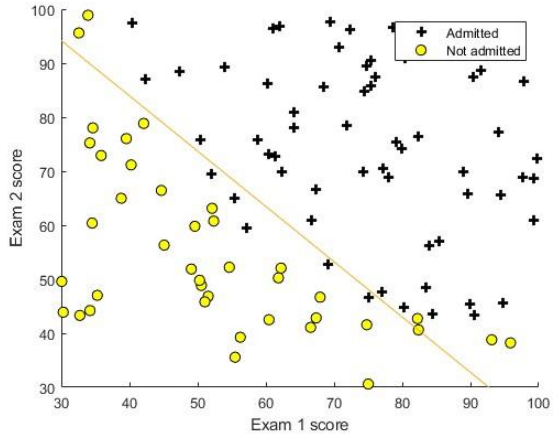
$$\frac{\partial J}{\partial \theta_j} = \frac{1}{m}\sum_{i=1}^{m}\left(-y^{(i)} + h_\theta(\underline{x}^{(i)})\right)x_j^{(i)}$$

Θα χρησιμοποιήσουμε την λογιστική παλινδρόμηση για να ελέγξουμε αν ένας φοιτητής θα γίνει δεκτός σε ένα πανεπιστήμιο με βάση τους βαθμούς του σε δύο εξετάσεις.

Αρχικά έχουμε τα εξής δεδομένα



Στην συνέχεια θα υλοποιήσουμε τον regressor μας συμπληρώνοντας αρχικά τον αντίστοιχο κώδικα για την sigmoid συνάρτηση. Υστέρα υλοποιούμε την συνάρτηση κόστους καθώς και την κλίση της, που υπολογίσαμε στην προηγούμενη σελίδα. Τέλος βρίσκουμε το σύνορο απόφασης και το σχεδιάζουμε στα παραπάνω δεδομένα.



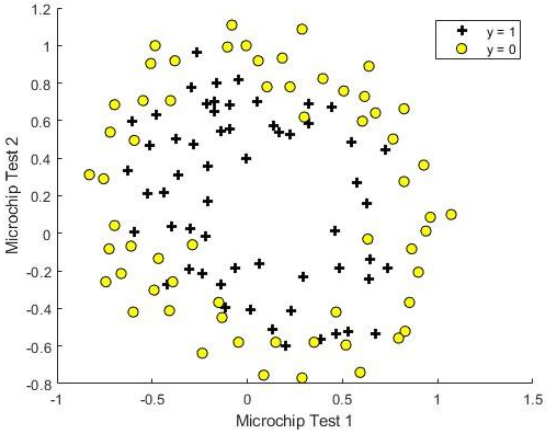
Η ακρίβεια του logistic regression είναι 89% επομένως το μοντέλο μας είναι αρκετά έμπιστο, όσον αφορά την πρόβλεψη να γίνει αποδεκτός ένας φοιτητής. Αυτό προκύπτει από το γεγονός ότι η πρόβλεψη μας βασίζεται σε γραμμικό σύνορο απόφασης και τα δεδομένα μας είναι κατά προσέγγιση γραμμικώς διαχωρίσιμα.

*Οι πράξεις επισυνάπτονται σε ξεχωριστό pdf στο zip αρχείο.

Θέμα 2

Σε αυτήν την άσκηση θα εφαρμόσουμε λογιστική παλινδρόμηση όπως και στην προηγούμενη, απλά θα προσθέσουμε έναν όρο ομαλοποίησης στην συνάρτηση κόστους, για να προβλέψουμε αν τα μικροτσίπ περνούν έναν έλεγχο ποιότητας.

Αρχικά θα τυπώσουμε τα δεδομένα μας :



Στην συνέχεια θα απεικονίσουμε τα δεδομένα αυτά σε χώρο μεγαλύτερης διάστασης ώστε να μπορέσουν να διαχωριστούν με μεγαλύτερη ακρίβεια, με βάση το ακόλουθο mapping

$$P(x_1, x_2) = \sum_{i=1}^{6} \sum_{j=1}^{i} x_1^{i-j} x_2^{j}$$

Αφού συμπληρώσουμε τον αντίστοιχο κώδικα χρησιμοποιούμε την ομαλοποιημένη cross entropy συνάρτηση κόστους

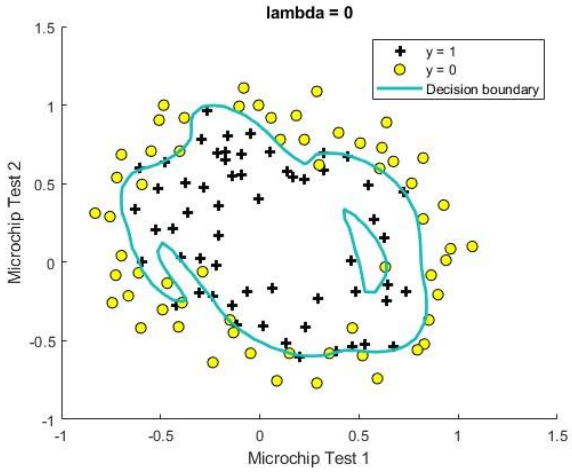
$$J(\underline{\theta}) = \frac{1}{m} \sum_{i=1}^{m} \left(-y^{(i)} \ln\left(\hat{y}^{(i)}\right) - \left(1 - y^{(i)}\right) \ln\left(1 - \hat{y}^{(i)}\right) \right) + \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2$$

Η παράγωγος ως προς το j-οστό στοιχείο του gradient J ως προς $\underline{\theta}$ είναι

$$\frac{\partial J}{\partial \theta_j} = \frac{1}{m} \sum_{i=1}^{m} \left(-y^{(i)} + h_\theta\left(\underline{x}^{(i)}\right) \right) x_j^{(i)} + \frac{\lambda}{m} \theta_j$$

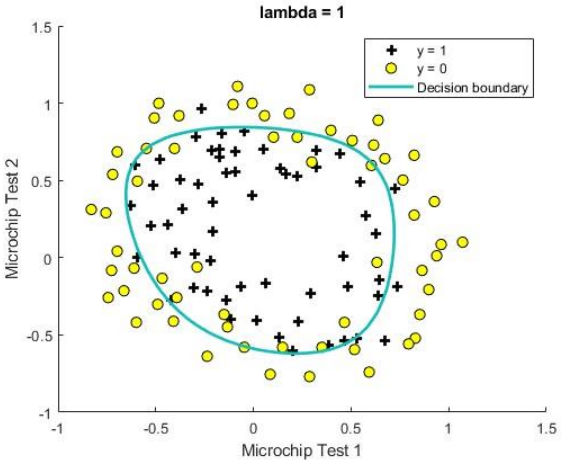
Αφού γράψουμε τον αντίστοιχο κώδικα για την συνάρτηση κόστους και την κλίση της θα εφαρμόσουμε την παλινδρόμηση για διαφορετικά λ.

- $\lambda = 0$



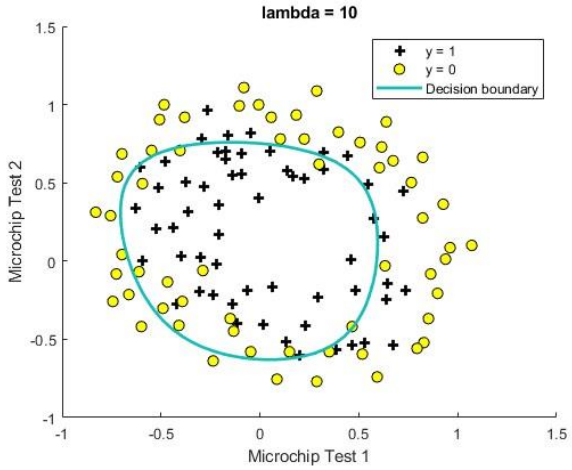
Train Accuracy: 88.89%

- $\lambda = 1$



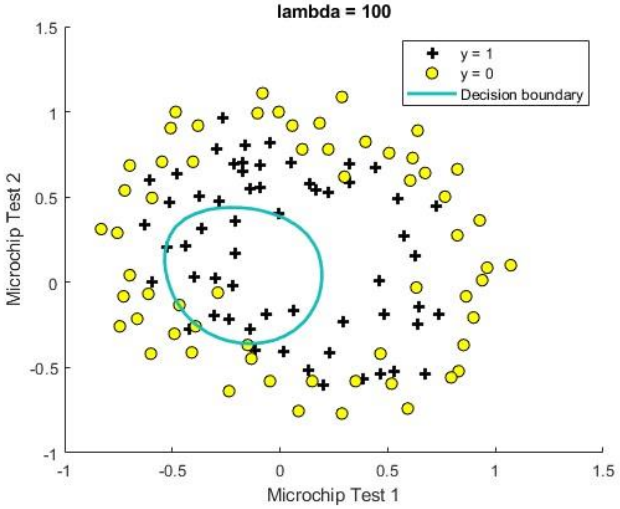
Train Accuracy: 83.05%

- $\lambda = 10$



Train Accuracy: 74.57%

- $\lambda = 100$



Train Accuracy: 60.16%

Προσθέτοντας τον όρο κανονικοποίησης μετατρέπουμε την συνάρτηση κόστους σε Lagrangian, δηλαδή προσθέτουμε L2 constraints τα οποία δεν επιτρέπουν την ελαχιστοποίηση της στο πραγματικό ελάχιστο της. Όσο μικρότερο το λ τόσο πιο κοντά βρισκόμαστε στο πραγματικό ελάχιστο και τόσο μεγαλύτερη ακρίβεια πετυχαίνουμε. Ο λόγος που χρησιμοποιούμε constraints είναι καθώς ένα πραγματικό πρόβλημα μπορεί να μην επιτρέπει την χρήση του πραγματικού ελαχίστου (πχ overfitting) καθιστώντας ως βέλτιστη την λύση που ικανοποιεί ταυτόχρονα τα constraints και ελαχιστοποιεί την J.

Θέμα 3

Για data record $D = \{x_1, \dots, x_n\}$ τα δείγματα παράγονται ανεξάρτητα και ακολουθούν την κατανομή Poisson με λ > 0.

$$p(x|\lambda) = \frac{\lambda^x e^{-\lambda}}{x!}, x = 0,1,2, \dots \ , \lambda > 0$$

Θα βρούμε μία εκτίμηση του λ με βάση το κριτήριο Maximum Likelihood (ML)

$$\hat{\lambda}_{ML} \rightarrow \max p(D|\lambda)$$

Η μεγιστοποίηση του likelihood ισοδυναμεί με την μεγιστοποίηση* του φυσικού λογαρίθμου της.

$$\hat{\lambda}_{ML} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Στην συνέχεια θα ακολουθήσουμε την ίδια διαδικασία ικανοποιώντας το κριτήριο Maximum a-posteriori (MAP) στο οποίο λαμβάνουμε υπόψη και την prior πιθανότητα.

$$\hat{\lambda}_{MAP} \rightarrow \max \ p(\lambda|D) \leftrightarrow \max \left(p(\lambda)p(D|\lambda)\right)$$

Καθώς

$p(\lambda|D) = \frac{p(\lambda)p(D|\lambda)}{p(D)}$ με $p(\lambda) = e^{-\lambda}$

Παίρνοντας τον λογάριθμο και παραλογίζοντας έχουμε ότι :

$$\hat{\lambda}_{MAP} = \frac{1}{n+1}\sum_{i=1}^{n} x_i$$

*Οι πράξεις επισυνάπτονται σε ξεχωριστό pdf στο zip αρχείο.

Θέμα 4

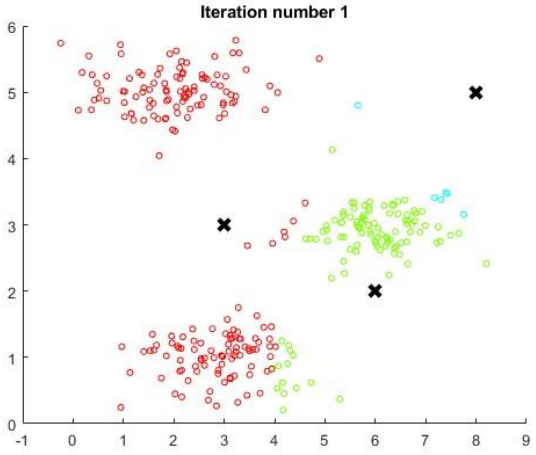
Σε αυτή την άσκηση θα εφαρμόσουμε τον αλγόριθμο K-means για clustering αρχικά 2D δεδομένων $X = \{\underline{x}^{(1)}, \underline{x}^{(2)}, \dots, \underline{x}^{(m)}\}$.

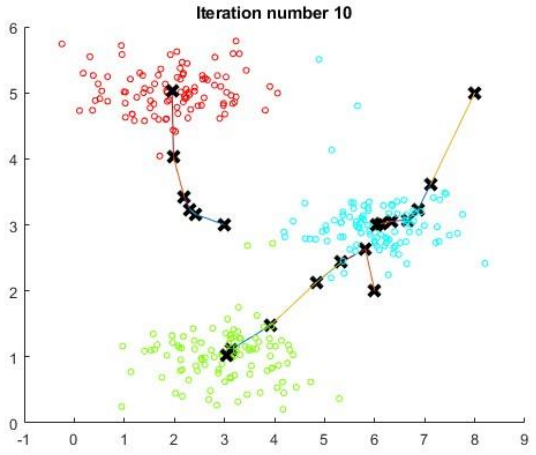
Η ομαδοποίηση εφαρμόζεται στα K κέντρα που διαλέγουμε και ελαχιστοποιεί την ευκλείδεια απόσταση $\left\|\underline{x}^{(i)} - \underline{\mu}_j\right\|^2 \forall\, j = 1, 2, \dots, K$.

Αρχικοποιούμε τυχαία τα κέντρα και βρίσκουμε τα πλησιέστερα για κάθε δείγμα $\underline{x}^{(i)}$. Στην συνέχεια τα υπολογίζουμε με βάση τον τύπο :

$$\underline{\mu}_k = \frac{1}{|C_k|} \sum_{i \in C_k} \underline{x}^{(i)}$$

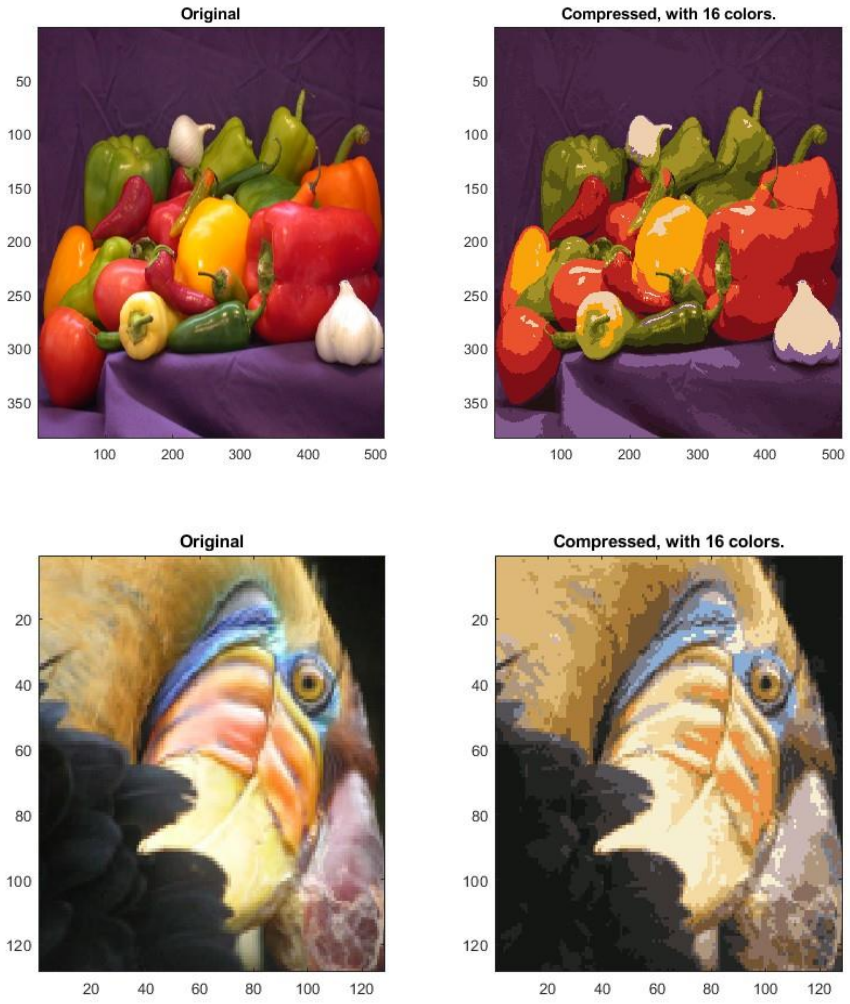
Στο πρώτο μέρος διαλέγουμε K = 3 και τρέχουμε τον αλγόριθμο 10 φορές. Στις παρακάτω εικόνες φαίνονται οι ομαδοποιήσεις για την τυχαία αρχικοποίηση και την διαδρομή του αλγορίθμου μετά τις 10 επαναλήψεις.





Παρατηρούμε ότι ο αλγόριθμος φαίνεται να έχει κατηγοριοποιήσει σε σωστά clusters τα δεδομένα καθώς βλέπουμε ότι τα means συγκλίνουν σε λογικές τιμές.

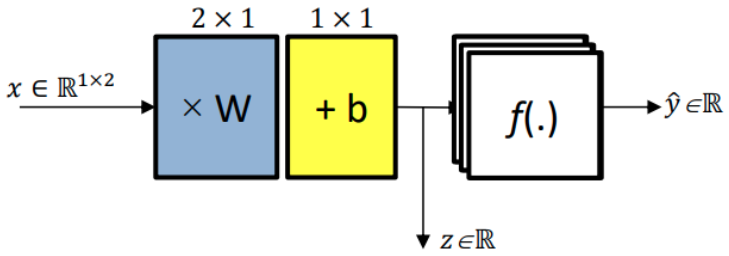
Τώρα θα εφαρμόσουμε τον αλγόριθμο για συμπίεση εικόνων σε 16 χρώματα. Θα το εφαρμόσουμε και στις 2 εικόνες που επισυνάπτονται και θα έχουμε :



Πράγματι το clustering έχει γίνει σωστά, με λίγα σφάλματα όπως στην πρώτη εικόνα κάτω δεξιά στο μωβ χρώμα. Αν θέλουμε καλύτερη ανάκτηση πληροφορίας θα πρέπει να αυξήσουμε τον αριθμό των χρωμάτων δηλαδή των μέσων αλλά και των επαναλήψεων. Αυτό βέβαια θα οδηγήσει σε λιγότερο καλή συμπίεση.

Θέμα 5

Σε αυτό το θέμα θα υλοποιήσουμε ένα απλό Νευρωνικό Δίκτυο (NN) χρησιμοποιώντας μόνο numpy. Προτού όμως γίνει αυτό πρέπει να το αναλύσουμε μαθηματικά.



Μέρος Α

Η συνάρτηση ενεργοποίησης είναι η σιγμοειδής συνάρτηση και η συνάρτηση κόστους η cross-entropy. Για να έχουμε αριθμητική ευστάθεια θα υπολογίζουμε την cross-entropy ανά ένα σύνολο Β που ονομάζεται Batch

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

Αρχικά θέτουμε $z = \underline{x}W + b$ και καταλήγουμε* σε

$$J(y,\hat{y};W,b) = \frac{1}{B}\sum_i \left(z^{(i)} - z^{(i)}\, y^{(i)} - \ln(1+e^{-z^{(i)}}) \right)$$

Στην συνέχεια υπολογίζουμε* την κλίση της συνάρτησης κόστους ως προς $z^{(i)}$

$$\frac{\partial J}{\partial z^{(i)}} = -y^{(i)} + \hat{y}^{(i)}, \qquad i \in Batch$$

Τέλος με τον κανόνα της αλυσίδας υπολογίζουμε ότι

$$\frac{\partial J}{\partial W} = \left(-y^{(i)} + \hat{y}^{(i)}\right)\underline{x}^{(i)T}, \qquad \frac{\partial J}{\partial b} = -y^{(i)} + \hat{y}^{(i)}, \qquad i \in Batch$$

Για να βελτιστοποιήσουμε ένα Νευρωνικό Δίκτυο πρέπει να υπολογίσουμε την κλήση της συνάρτησης κόστους ως προς κάθε παράμετρο ανά layer και στην συνέχεια να εφαρμόσουμε stochastic approximation για να ανανεώσουμε τις παραμέτρους.

Ο backpropagation βασίζεται στον κανόνα της αλυσίδας για να μεταφέρει την παράγωγο ως προς κάθε παράμετρο προς τα πίσω επίπεδα του NN. Θα πάρουμε ως παράδειγμα το τελευταίο επίπεδο και θα έχουμε

$$\begin{bmatrix} W_3^{(i+1)} \\ b_3^{(i+1)} \end{bmatrix} = \begin{bmatrix} W_3^{(i)} \\ b_3^{(i)} \end{bmatrix} - \eta\left(-y^{(i)} + \hat{y}^{(i)}\right)\begin{bmatrix} \underline{x}^{(i)T} \\ 1 \end{bmatrix}$$

Μέρος Β

Σε αυτό το μέρος της άσκησης θα γράψουμε κώδικα για την δημιουργία κάθε βασικού χαρακτηριστικού ενός fully connected Νευρωνικού Δικτύου έτσι ώστε να δοκιμάσουμε διάφορες αρχιτεκτονικές για classification του Digits-MNIST dataset.

1.

Διακρίνουμε τις επόμενες περιπτώσεις

Cases	Learning Rate	Epochs	Batch size	Ratio	MSE
1	0.1	100	128	0.8	0.0219
2	0.01	150	128	0.8	0.033
3	0.1	100	64	0.85	0.0205
4	0.1	200	64	0.9	0.0206
5	0.3	50	128	0.15	nan

Παρατηρούμε ότι για την αρχιτεκτονική αυτή όσο και να πειράξουμε τις υπερπαραμέτρους το μέσο τετραγωνικό σφάλμα δεν μειώνεται πιο πολύ από ότι παρατηρούμε. Ταυτόχρονα για κατάλληλη επιλογή των παραμέτρων η ακρίβεια δεν βελτιώνεται σημαντικά. Τέλος παρατηρούμε ότι αν επιλέξουμε πολύ μεγάλο learning rate τότε το NN δεν συγκλίνει σε ελάχιστο αλλά σε μία τελείως διαφορετική μη βέλτιστη λύση και το μοντέλο μας συναντάει τιμές που δεν μπορεί να διαχειριστεί.

2.

Στην συνέχεια θα χρησιμοποιήσουμε για συνάρτηση κόστους την MSE και συνάρτηση ενεργοποίησης την υπερβολική εφαπτομένη.

Cases	Learning Rate	Epochs	Batch size	Ratio	MSE
1	0.1	100	128	0.55	0.0701
2	0.1	150	64	0.6	0.0677
3	0.1	200	64	0.7	0.052

Αυτό το μοντέλο για τις αρχικές υπερπαραμέτρους έχει πολύ χειρότερη απόδοση. Παρόλα αυτά με αύξηση των epochs και μείωση του Batch size παρατηρούμε ότι αποκτά μεγαλύτερη ακρίβεια η οποία όμως είναι λιγότερο καλή από την αντίστοιχη του μοντέλου του (1) .

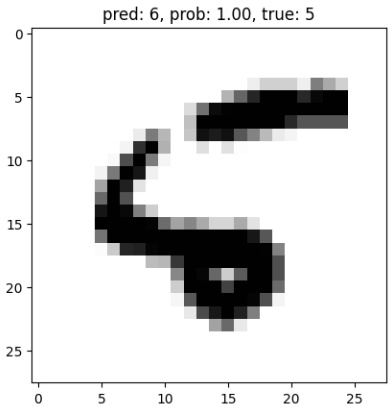
Αυτό σημαίνει ότι για το classification των ψηφίων η cross-entropy σε συνδυασμό με την sigmoid αποδίδουν καλύτερα σε σχέση με την MSE και την tanh.

3.

Τέλος θα χρησιμοποιήσουμε την cross-entropy και sigmoid όπως στο πρώτο πείραμα χρησιμοποιώντας αρχικά περισσότερους νευρώνες και στην συνέχεια προσθέτοντας ένα ακόμα επίπεδο ίδιου αριθμού νευρώνων.

Cases	Layers	Ratio	MSE
1	1 (100 neurons)	0.85	0.0147
2	2 (100 neurons)	0.85	0.0222

Για τις αρχικές τιμές των υπερπαραμέτρων παρατηρούμε ότι το μοντέλο μας είναι αρκετά καλύτερο από το αντίστοιχο του ερωτήματος 1. Ενώ ταυτόχρονα βλέπουμε ότι με την προσθήκη παραπάνω επιπέδων δεν αλλάζει το accuracy. Επιπλέον παρατηρούμε ότι το δίκτυο σε κάθε περίπτωση για κάθε αρχιτεκτονική κάνει πάντοτε λάθος την κατηγοριοποίηση στο εξής ψηφίο.



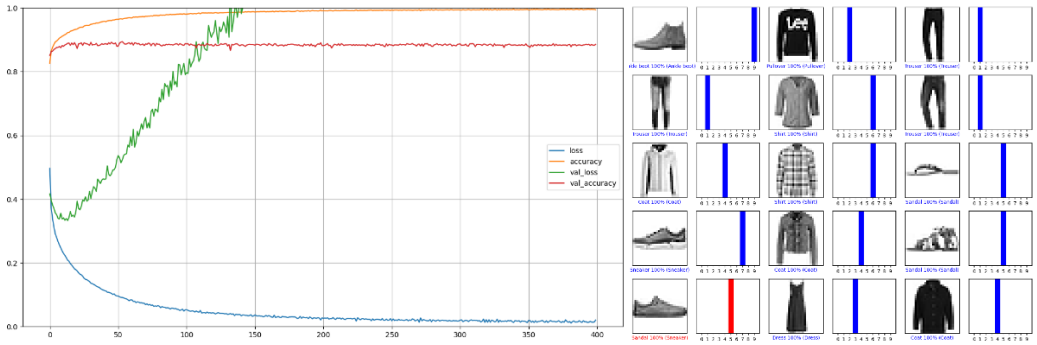
Λογικό καθώς το συγκεκριμένο δεν έχει τα συνηθισμένα χαρακτηριστικά ενός ψηφίου “5”.

Θέμα 6

Σε αυτήν την άσκηση θα χρησιμοποιήσουμε την βιβλιοθήκη tensorflow/keras για να δημιουργήσουμε μερικούς classifiers που θα κατηγοριοποιούν τα δεδομένα Fashion-MNIST.

1.

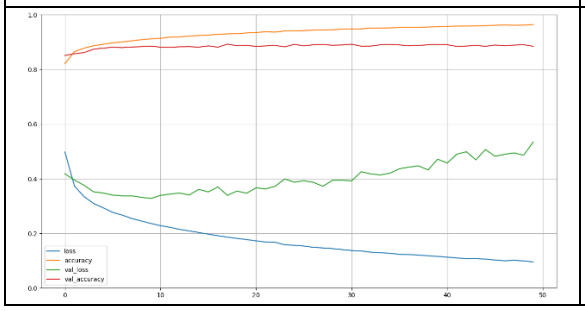
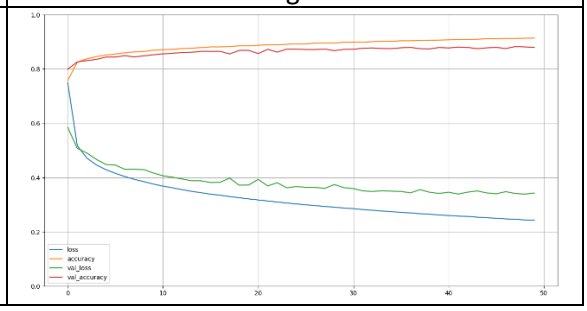
Αρχικά τρέχουμε το πρόγραμμα με αρχιτεκτονική ενός μόνο επιπέδου fully-connected νευρωνικού δικτύου για 400 epochs και καταγράφουμε κάθε τιμή που αφορά το accuracy και το loss σε ένα γράφημα.

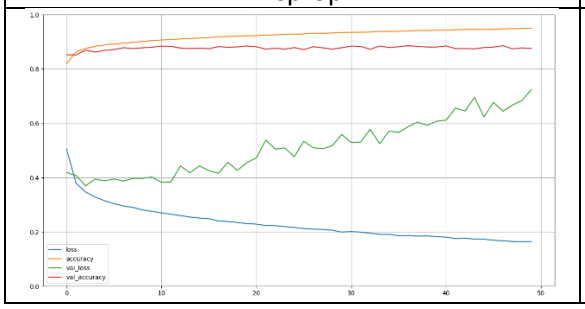
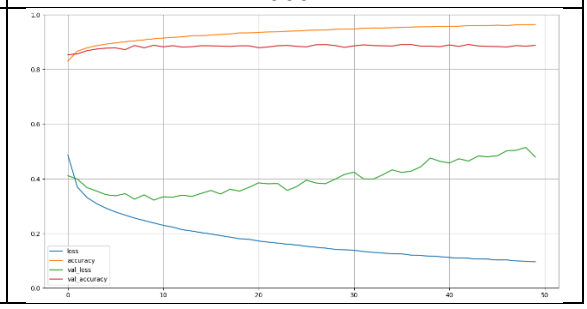


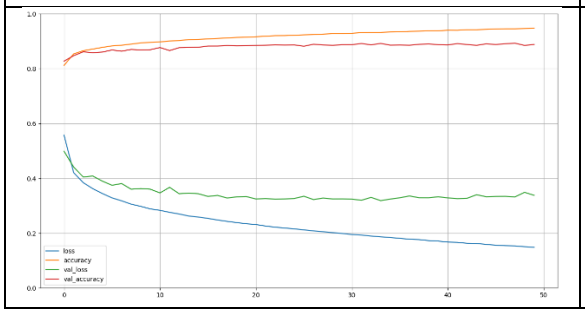
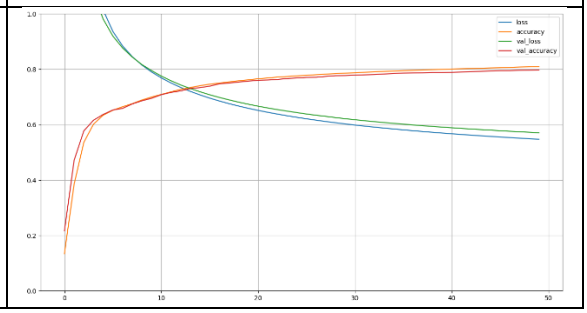
Παρατηρούμε ότι το validation accuracy συγκλίνει σε τιμή λίγο μικρότερη του 0,9 ενώ το Training accuracy συγκλίνει στο 1. Λογικό καθώς το πρώτο προκύπτει από δεδομένα πάνω στα οποία το NN δεν έχει εκπαιδευτεί. Ταυτόχρονα παρατηρούμε ότι το validation loss αυξάνεται πάρα πολύ. Αυτό συμβαίνει γιατί υπάρχει overfitting. Στόχος μας θα είναι να μειώσουμε το overfitting και να αυξήσουμε το validation accuracy.

2.

Στην συνέχεια θα συγκρίνουμε διάφορους optimizers και θα δούμε ποιος έχει το μεγαλύτερο accuracy (μειώνουμε τα epochs σε 50).

adam	sgd
	

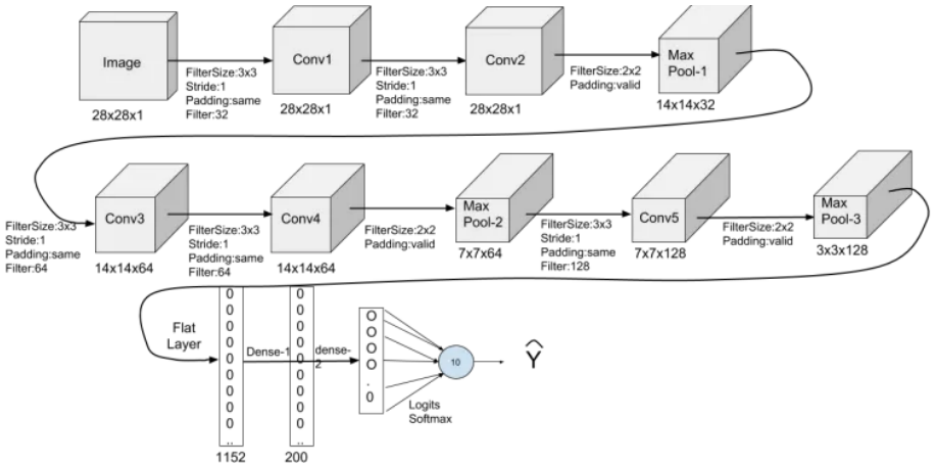
rmsprop	nadam
	

adammax	ftrl
	

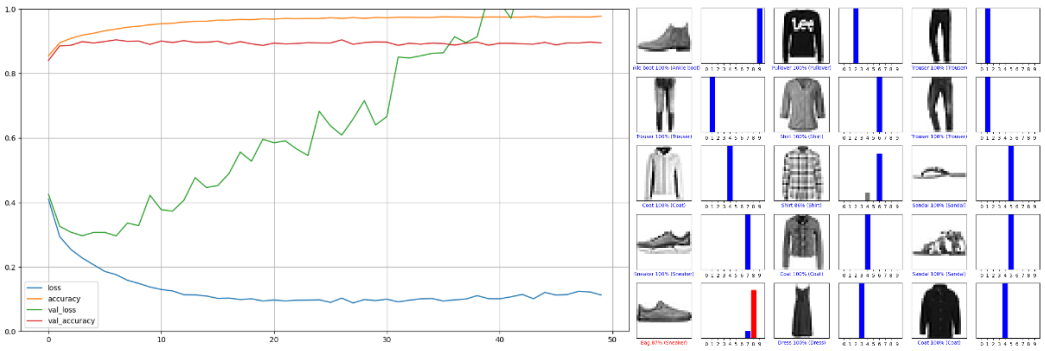
Βλέπουμε ότι οι optimizers που προσδίδουν το καλύτερο validation accuracy είναι οι adam, nadam, adammax. Ταυτόχρονα αυτοί που δίνουν το λιγότερο overfitting είναι οι sgd και ftrl.

3.

Θα αλλάξουμε την αρχιτεκτονική του δικτύου από ένα dense NN σε ένα Convolutional Neural Network (CNN). Αρχικά θα πρέπει να μετασχηματίσουμε τις διαστάσεις των εικόνων όπως μας περιγράφεται, καθώς οι είσοδοι του CNN πρέπει να είναι tensors.



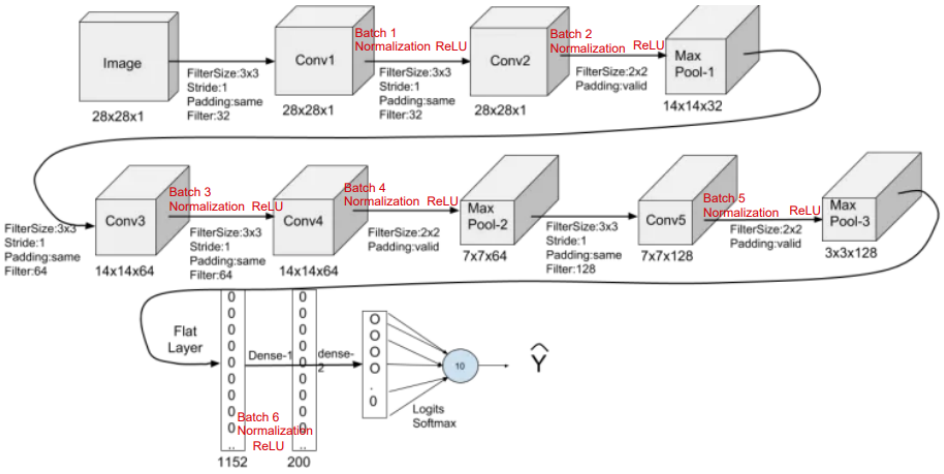
Τρέχουμε το πρόγραμμα που υλοποιεί την παραπάνω αρχιτεκτονική για 50 epochs και optimizer adam και έχουμε



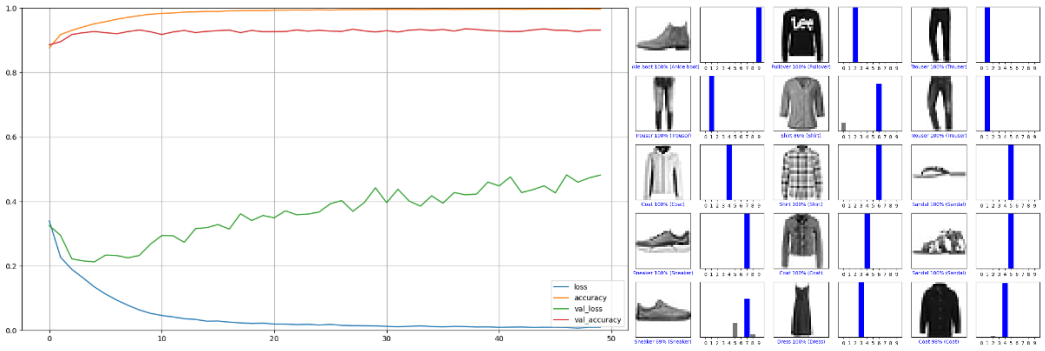
Παρατηρούμε ότι παρά των παραπάνω επιπέδων συνελίξεων που βάλαμε έχουμε ακόμα ταχύτερο overfitting και όχι σημαντική βελτίωση στο validation accuracy.

4.

Στην συνέχεια προσθέτουμε batch normalization και συναρτήσεις ενεργοποίησης ReLU διαδοχικά αμέσως μετά κάθε επίπεδο συνέλιξης και μετά του 1ου dense επιπέδου όπως φαίνεται παρακάτω.



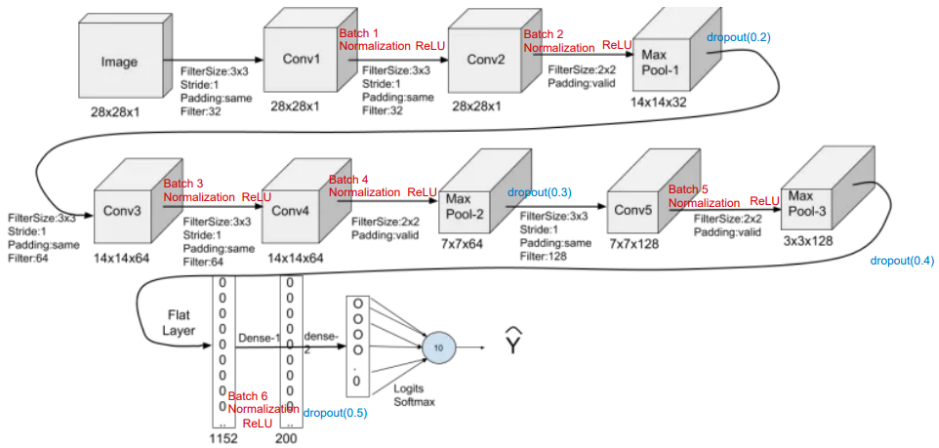
Αφού προσθέσουμε τα αντίστοιχα επίπεδα στο πρόγραμμα, το τρέχουμε και παίρνουμε τις παρακάτω γραφικές παραστάσεις .



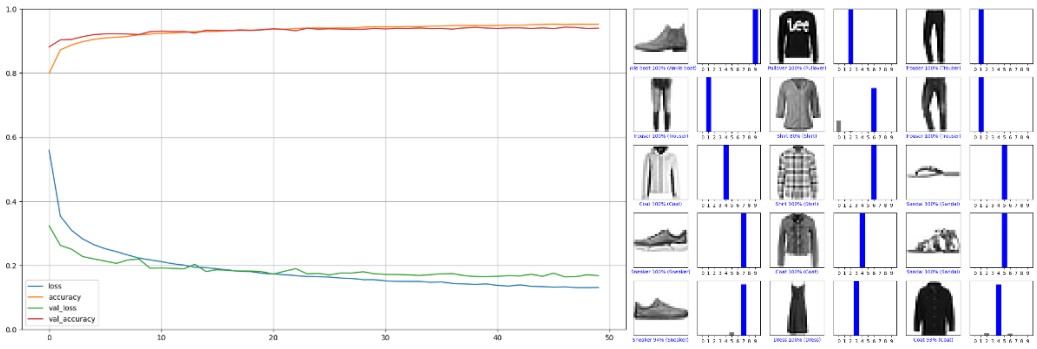
Κατευθείαν βλέπουμε ότι το overfitting είναι πολύ μικρότερο, δεν παύει όμως να υπάρχει. Παράλληλα το validation accuracy αυξήθηκε περισσότερο.

5.

Τέλος προσθέτουμε μερικά dropout layers αμέσως μετά κάθε max-pooling layer και του 2ου dense layer όπως φαίνεται στην παρακάτω διάταξη.



Προσθέτουμε στο πρόγραμμα και αυτά τα επίπεδα και έχουμε.



Το dropout regularization εμποδίζει το CNN να βασιστεί αποκλειστικά στην αναγνώριση συγκεκριμενών προτύπων. Με αυτόν τον τρόπο το learning γίνεται πιο σωστά και το overfitting παύει να υπάρχει. Ταυτόχρονα το validation accuracy ταυτίζεται σχεδόν με το training accuracy. Όλες οι παραπάνω αλλαγές καθιστούν το μοντέλο μας ιδανικό για classification καθώς γνωρίζουμε ότι έχει σταθερά καλή ακρίβεια για κάθε νέο δεδομένο που θα κατηγοριοποιήσει.

** Κάθε αρχείο .py έτρεξε μέσω του colab με ρύθμιση με Hardware Accelerator την GPU στο Runtime