ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

Τμήμα Ηλεκτρολόγων Μηχ. Και Μηχ. Υπολογιστών

Στατιστική Μοντελοποίηση και Αναγνώριση Προτύπων (ΤΗΛ311)

Φυλλάδιο Ασκήσεων 2

Οδηγίες:

1. Σας παρακαλώ να σεβαστείτε τον παρακάτω κώδικα τιμής τον οποίον θα θεωρηθεί ότι προσυπογράφετε μαζί με τη συμμετοχή σας στο μάθημα και τις εργασίες του:
   a) *Οι απαντήσεις στις εργασίες, τα quiz και τις εξετάσεις, ο κώδικας και γενικά οτιδήποτε αφορά τις εργασίες θα είναι προϊόν δικής μου δουλειάς.*
   b) *Δεν θα διαθέσω κώδικα, απαντήσεις και εργασίες μου σε κανέναν άλλο.*
   c) *Δεν θα εμπλακώ σε άλλες ενέργειες με τις οποίες ανέντιμα θα βελτιώνω τα αποτελέσματα μου ή ανέντιμα θα αλλάζω τα αποτελέσματα άλλων.*
2. Η εργασία αφορά ομάδες μέχρι 2 ατόμων
3. Ημερομηνία παράδοσης: **Δευτέρα, 10/7/2023 στις 23:00**
4. **Παραδοτέα:** α) Κώδικας και β) Αναφορά με τις απαντήσεις, παρατηρήσεις, πειράματα, αποτελέσματα και οδηγίες χρήσης του κώδικα.

Θέμα 1: Λογιστική Παλινδρόμηση: Αναλυτική εύρεση κλίσης (Gradient)

Υποθέτουμε ότι έχουμε ένα σύνολο m δεδομένων $\{(\boldsymbol{x}^{(1)}, y^{(1)}), (\boldsymbol{x}^{(2)}, y^{(2)}), \dots, (\boldsymbol{x}^{(m)}, y^{(m)})\}$, όπου $\boldsymbol{x}^{(i)} \in \mathbb{R}^{n\times 1}$ είναι τα διανύσματα χαρακτηριστικών και $y^{(i)} \in \{0,1\}$ ορίζουν την κλάση κάθε δείγματος (labels). Θέλουμε να προβλέψουμε τις τιμές των $y^{(i)}$ από τις αντίστοιχες τιμές $\boldsymbol{x}^{(i)}$, $i \in \{1,2,\dots,m\}$, χρησιμοποιώντας την συνάρτηση της λογιστικής παλινδρόμησης, η οποία ορίζεται ως εξής

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = f(\boldsymbol{\theta}^T \boldsymbol{x})$$

όπου $\boldsymbol{\theta} = [\theta_1, \theta_2, \cdots, \theta_n]^T$ είναι οι παράμετροι του γραμμικού μοντέλου και $f()$ είναι η λογιστική συνάρτηση που ορίζεται ως:

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

Έστω $\hat{y}^{(i)} = h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})$ η εκτίμηση της λογιστικής συνάρτησης για το $y^{(i)}$. Σύμφωνα με τη θεωρία, στην περίπτωση της λογιστικής παλινδρόμησης, μπορούμε να υπολογίσουμε το σφάλμα με βάση τη συνάρτηση κόστους/σφάλματος (loss function), που ονομάζεται cross-entropy, και ορίζεται ως εξής:

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

Αντικαθιστώντας το $\hat{y}^{(i)}$ έχουμε:

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

Για τη βελτιστοποίηση του σφάλματος χρειάζεται να υπολογίσουμε την κλίση (gradient) του σφάλματος $J(\boldsymbol{\theta})$ η οποία θα είναι ένα διάνυσμα ίσης διάστασης με το $\boldsymbol{\theta}$.

a) Αν θ_j και $x_j^{(i)}$ είναι η j συνιστώσα των διανυσμάτων $\boldsymbol{\theta} = [\theta_1, \theta_2, \dots, \theta_n]^T$ και $\boldsymbol{x}^{(i)} = \left[x_1^{(i)}, x_2^{(i)}, \dots x_n^{(i)}\right]^T$ αντίστοιχα, να δείξετε ότι το j-στοιχείο της κλίσης του σφάλματος είναι:

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \theta_j} = \frac{1}{m}\sum\nolimits_{i=1}^{m}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)x_j^{(i)}$$

b) Θα χρησιμοποιήσουμε την λογιστική παλινδρόμηση για να προβλέψουμε αν ένας φοιτητής θα γίνει δεκτός σε ένα πανεπιστήμιο με βάση τους βαθμούς του σε δύο εξετάσεις. Στο αρχείο «exam_scores_data1.txt» υπάρχουν δεδομένα από παλαιότερες αιτήσεις φοιτητών στη μορφή ***«Exam1Score, Exam2Score, [0: απόρριψη, 1: αποδοχή]»***, που θα χρησιμοποιηθούν ως δεδομένα εκμάθησης της λογιστικής παλινδρόμησης. Για να διαβάσετε τα δεδομένα τρέξτε την εντολή:

```
data = load('exam_scores_data1.txt');
```

Θα χρειαστεί να συμπληρώσετε κώδικα ώστε να τρέξει το κύριο αρχείο της άσκησης: ***My_logisticRegression.m***

i) Αρχικά δείτε τα δεδομένα με την συνάρτηση `plotData.m`

ii) Υλοποιήστε την σιγμοειδή συνάρτηση $f(z)$ στο αρχείο `sigmoid.m`. Αν η είσοδος z είναι πίνακας, η sigmoid θα πρέπει να εφαρμόζει την $f(z)$ σε κάθε στοιχείο του πίνακα.

iii) Υλοποιήστε την συνάρτηση *`costFunction`* στο αρχείο *`costFunction.m`* έτσι ώστε να επιστρέφει το κόστος $J(\boldsymbol{\theta})$ (J) και την κλίση $\nabla J(\boldsymbol{\theta})$ (grad) όπως ορίζονται στο ερώτημα (α) (***Σημείωση****: Χρησιμοποιώντας ως αρχικές τιμές του* $\boldsymbol{\theta} = \mathbf{0}$*, το κόστος θα πρέπει να βγαίνει περίπου J=0.693 και η κλίση περίπου [-0.1, -12.009217, -11.262842]*).

iv) Η βελτιστοποίση των παραμέτρων γίνεται με κώδικα που υπάρχει έτοιμος στην άσκηση `My_logisticRegression.m`. Εσείς απλά τρέξτε τον κώδικα και βρείτε το σύνορο απόφασης με την συνάρτηση `plotDecisionBoundary.m`. Επίσης τρέξτε την συνάρτηση `predict.m` για να προβλέψετε αν ο φοιτητής θα γίνει δεκτός με βάση διάφορες τιμές βαθμών στις δύο εξετάσεις.

Θέμα 2: Λογιστική Παλινδρόμηση με Ομαλοποίηση

Σε αυτή την άσκηση, θα εφαρμόσουμε ομαλοποιημένη λογιστική παλινδρόμηση για να προβλέψουμε αν τα μικροτσίπ από μια μονάδα κατασκευής περνούν τον έλεγχο ποιότητα (QA). Κατά τη διάρκεια της QA, κάθε μικροτσίπ περνάει από διάφορες δοκιμές για να εξασφαλιστεί ότι λειτουργεί σωστά. Υποθέτουμε ότι υπάρχουν τα αποτελέσματα δύο διαφορετικών δοκιμών για ορισμένα μικροτσίπ. Από αυτά τα αποτελέσματα θα πρέπει να καθορίσετε αν τα μικροτσίπ θα γίνουν αποδεκτά ή θα απορριφθούν. Θα χρησιμοποιήσουμε δεδομένα προηγούμενων δοκιμών για να δημιουργήσουμε ένα μοντέλο λογιστικής παλινδρόμησης. Θα πρέπει να συμπληρώσετε τον απαραίτητο κώδικα στα αρχεία matlab που βρίσκονται στον φάκελο *`exercise2_2`* ώστε να τρέξει η άσκηση από το κύριο αρχείο: *`ex2_2_regularizedLogisticRegression.m`*. Συγκεκριμένα:

a) Δείτε τα δεδομένα με την συνάρτηση `plotData.m`

b) Απεικονίστε τα δεδομένα σε χώρο μεγαλύτερης διάστασης όπου μπορούν να διαχωριστούν ευκολότερα με την λογιστική παλινδρόμηση. Συμπληρώστε την συνάρτηση `mapFeature.m` που απεικονίζει τα χαρακτηριστικά σε όλους τους όρους πολυωνύμων x_1 και x_2 βαθμού μέχρι και 6.

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

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

c) Η ομαλοποιημένη συνάρτηση κόστους δίνεται από την σχέση:

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

d) Αν θ_j και $x_j^{(i)}$ είναι η j συνιστώσα των διανυσμάτων $\theta = [\theta_1, \theta_2, \dots, \theta_n]^T$ και

$x^{(i)} = \left[x_1^{(i)}, x_2^{(i)}, \dots x_n^{(i)}\right]^T$ αντίστοιχα, να δείξετε ότι το j-στοιχείο της κλίσης του σφάλματος είναι:

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

e) Συμπληρώστε την συνάρτηση `costFunctionReg.m`

f) Ο κώδικας για την βελτιστοποίηση των παραμέτρων υπάρχει στην άσκηση. Να βρείτε τα σύνορα απόφασης για διάφορες τιμές της παραμέτρου λ (π.χ. για λ =0, 1, 10, 100)

Θέμα 3: Εκτίμηση Παραμέτρων με Maximum Likelihood και MAP

Υποθέστε ότι n δείγματα $D = \{x_1, \dots, x_n\}$ παράγονται ανεξάρτητα από μια κατανομή Poisson με παράμετρο λ:

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

A) Βρείτε τον εκτιμητή μεγίστης πιθανοφάνειας (MLE) της παραμέτρου $\hat{\lambda}_{ML}$.

B) Έστω ότι η εκ των προτέρων πιθανότητα της παραμέτρου λ είναι

$$p(\lambda) = e^{-\lambda}$$

Βρείτε την τιμή της παραμέτρου $\hat{\lambda}_{MAP}$ (Maximum A-Posteriori) που μεγιστοποιεί την πιθανότητα $p(\lambda|D)$

Θέμα 4: Ομαδοποίηση (Clustering) με K-means

Σε αυτή την άσκηση θα υλοποιήσετε τον K-means clustering αλγόριθμο και θα τον χρησιμοποιήσετε για να συμπιέσετε μια εικόνα. Για να καταλάβετε πως λειτουργεί ο K-means, θα αρχίσετε με ένα δισδιάστατο (2D) σύνολο δειγμάτων $X = \{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}, x^{(i)} \in \mathbb{R}^2$ που βρίσκονται στο αρχείο 'ex4data1.mat'. Ο K-means είναι μια επαναληπτική διαδικασία που ξεκινάει με κάποια αρχικοποίηση για τα κέντρα των *K* κλάσεων (π.χ. τυχαίες τιμές), και στην συνέχεια βελτιώνει την αρχική του επιλογή τοποθετώντας τα δείγματα στην κλάση με την μικρότερη απόσταση και ξαναϋπολογίζοντας τα κέντρα των κλάσεων.

Ο παρακάτω κώδικας Matlab συνοψίζει τον αλγόριθμο K-means:

```
% Initialize centroids
centroids = kMeansInitCentroids(X, K);
for iter = 1:iterations
    % Cluster assignment step: Assign each data point to the
    % closest centroid. idx(i) corresponds to cˆ(i), the index
    % of the centroid assigned to example i
    idx = findClosestCentroids(X, centroids);
    % Move centroid step: Compute means based on centroid
    % assignments
    centroids = computeMeans(X, idx, K);
end
```

Στα πλαίσια της άσκησης θα χρειαστεί να συμπληρώσετε τον κώδικα που λείπει και να τρέξετε το Matlab/Octave script ex24_kmeans.m. Πιο συγκεκριμένα, θα πρέπει να συμπληρώσετε τα επόμενα μέρη του προγράμματος:

a) Την συνάρτηση ***findClosestCentroids.m***

η οποία θα βρίσκει τα $c^{(i)} := j$ που ελαχιστοποιούν την απόσταση $\left\|x^{(i)} - \mu_j\right\|^2$, όπου $c^{(i)}$ είναι ο δείκτης του κοντινότερου κέντρου στο $x^{(i)}$, και μ_j είναι το διάνυσμα τιμών (συντεταγμένων) του κέντρου j. Το $c^{(i)}$ αντιστοιχεί στο `idx(i)` του παραπάνω κώδικα.

b) Την συνάρτηση ***computeCentroids.m***

η οποία θα υπολογίζει τα κεντροειδή των κλάσεων

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

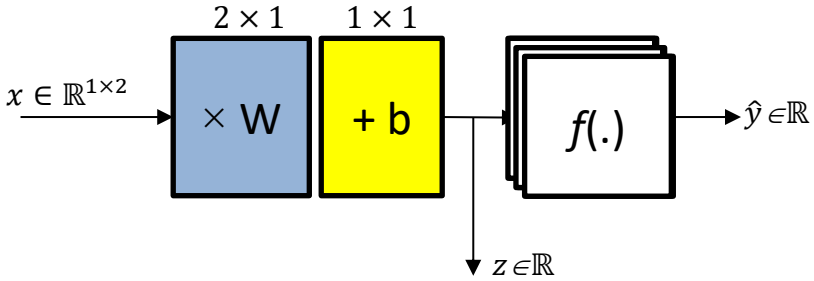
όπου C_k είναι το σύνολο των παραδειγμάτων που έχουν αντιστοιχηθεί στην κλάση k.

c) Συμπληρώστε την εντολή `X_recovered = ….` στο Matlab/Octave script ex24_kmean.m και δείτε την συμπιεσμένη εικόνα για διάφορες τιμές του K (αριθμός κλάσεων/χρωμάτων). Μπορείτε να χρησιμοποιήσετε είτε την εικόνα '**peppers.png**' είτε την '**bird_small.png**'. Δοκιμάστε να τρέξετε τον αλγόριθμο για διαφορετικό αριθμό κλάσεων/χρωμάτων και κάντε παρατηρήσεις για το αποτέλεσμα που παράγεται (συμπιεσμένη εικόνα) σε συνάρτηση με τον όγκο δεδομένων που απαιτούνται για την συμπιεσμένη εικόνα.

Θέμα 5: Υλοποίηση ενός απλού νευρωνικού δικτύου.

Σ' αυτή την άσκηση ο στόχος είναι να υλοποιηθεί και εκπαιδευτεί ένα νευρωνικό δίκτυο **χωρίς την χρήση έτοιμων προγραμμάτων/βιβλιοθηκών**. Η υλοποίηση θα γίνει σε python χρησιμοποιώντας numpy.

Μέρος Α: Έστω ότι σας δίνονται τα δεδομένα εκπαίδευσης $D = \{(x^{(1)}, y^{(1)}), \dots, (x^{(N)}, y^{(N)})\}$ με $x^{(i)} \in \mathbb{R}^{1\times 2}$ και $y^{(i)} \in \mathbb{R}$.



Το νευρωνικό δίκτυο όπως φαίνεται πιο πάνω ορίζεται από την σχέση:

$$\hat{y}^{(i)} = f(x^{(i)}W + b)$$

όπου $x^{(i)} \in \mathbb{R}^{1\times 2}$ είναι ένα δείγμα, $W \in \mathbb{R}^{2\times 1}$, $b \in \mathbb{R}^{1\times 1}$, και

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

Το σφάλμα ανάμεσα στην πρόβλεψη $\hat{y}^{(i)}$ του νευρωνικού δικτύου και στην τιμή $y^{(i)}$ που αντιστοιχεί στα δεδομένα εισόδου $x^{(i)}$, το μετράμε με την συνάρτηση cross-entropy

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

Για να έχουμε αριθμητική ευστάθεια, συνήθως υπολογίζουμε την cross-entropy για ένα σύνολο B δειγμάτων και παίρνουμε τον μέσο όρο. (Το σύνολο B δειγμάτων ονομάζεται batch).

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

α) Αν θέσουμε $z^{(i)} = x^{(i)}W + b$, οπότε $\hat{y}^{(i)} = f(z^{(i)})$, να δείξετε ότι

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

β) Να δείξετε ότι

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

γ) Με τον κανόνα της αλυσίδας να υπολογίσετε τις μερικές παραγώγους

$$\frac{\partial J}{\partial W}, \frac{\partial J}{\partial b}$$

Ο κανόνας παραγώγισης με τον πίνακα W είναι: $\frac{\partial (xW)}{\partial W} = x^T$

δ) Περιγράψτε σύντομα πως γίνεται οπισθοδιάδοση (Back Propagation) του σφάλματος και πως ενημερώνονται οι παράμετροι ενός νευρωνικού δικτύου πολλαπλών επιπέδων όπως το παρακάτω, στο οποίο χρησιμοποιούμε την cross-entropy σαν συνάρτηση σφάλματος (loss function) και την λογιστική συνάρτηση ως συνάρτηση ενεργοποίησης. Γράψτε τις σχέσεις που χρησιμοποιούνται για να μεταβληθούν τα βάρη και το bias σε ένα μόνο βήμα του αλγορίθμου και για ένα μόνο layer.

Μέρος Β: Να συμπληρώσετε τον κώδικα της άσκησης σε python και να εκπαιδεύσετε το νευρωνικό δίκτυο χρησιμοποιώντας δείγματα από την βάση δειγμάτων MNIST. Τα δείγματα είναι εικόνες μεγέθους 28x28 που απεικονίζουν χειρόγραφα ψηφία αριθμών από το 0 έως το 9. Ο στόχος είναι το δίκτυο σας να μπορεί να αναγνωρίζει την κλάση στην οποία ανήκει κάθε δείγμα ξεχωριστά. Αρχικά σας δίνεται μια αρχιτεκτονική ενός fully connected (dense) δικτύου αποτελούμενο από νευρώνες με συνάρτηση ενεργοποίησης την λογιστική συνάρτηση (Sigmoid):

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

Στο επόμενο επίπεδο θα χρησιμοποιηθεί η συνάρτηση ενεργοποίησης SoftMax.

Αφού συμπληρώσετε τον κώδικα δοκιμάστε να πειραματιστείτε με τα ακόλουθα και σημειώστε στην αναφορά σας τις παρατηρήσεις σας:

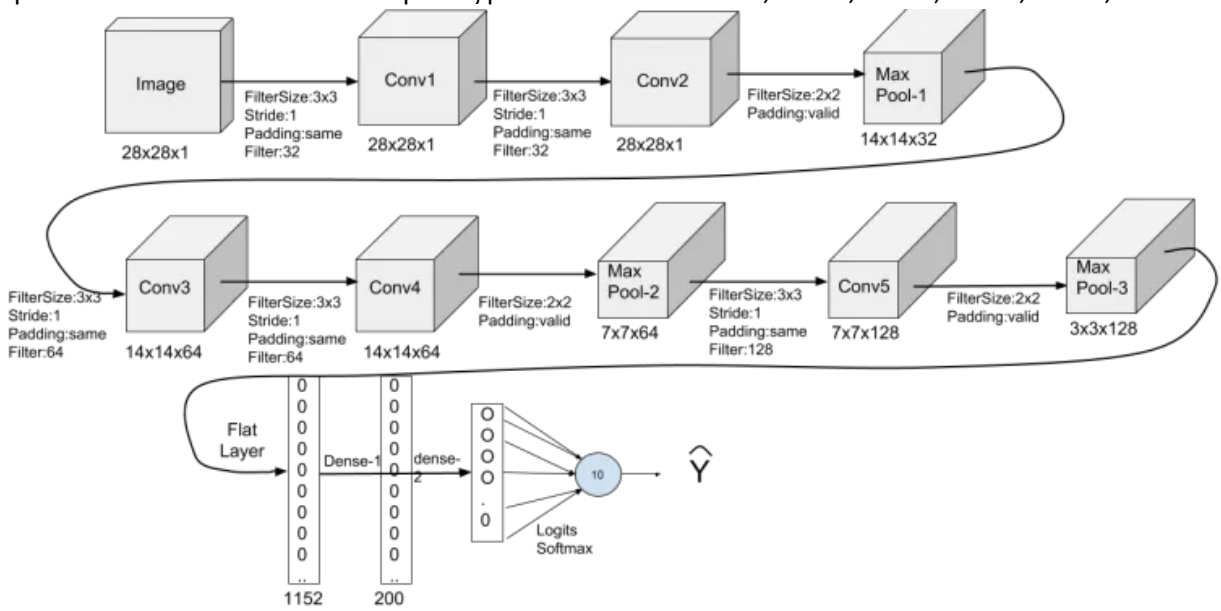
1) Learning Rate/Αριθμός epochs/To batch size.
2) Χρησιμοποιήστε συνάρτηση ενεργοποίησης tanh και συνάρτηση σφάλματος MSE (Mean Square Error)
3) Διαφορετικές αρχιτεκτονικές δικτύου (π.χ. περισσότερα layers ή/και διαφορετικό αριθμό νευρώνων στο hidden layer.

Θέμα 6: Convolutional Neural Networks for Image Recognition

Σε αυτή την άσκηση θα δοκιμάσουμε διάφορους την βιβλιοθήκη tensorflow/keras ταξινομητές νευρωνικών δικτύων χρησιμοποιώντας το σύνολο δεδομένων Fashion-MNIST [1]. Οι εικόνες του Fashion-MNIST χωρίζονται σε 10 κατηγορίες {'T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat', 'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot'}.

Στα αρχεία της άσκησης θα βρείτε τον κώδικα fashion.py που υλοποιεί έναν ταξινομητή με dense neural networks. Σε όλες τις παρακάτω ερωτήσεις τρέξτε το πρόγραμμα και δημιουργήστε ένα γράφημα με τις τιμές της ακρίβειας (accuracy) στα δεδομένα εκπαίδευσης (training data) και στα δεδομένα επικύρωσης (validation data). Τι παρατηρείτε;

1. Τρέξτε το πρόγραμμα με την αρχιτεκτονική που δίνεται και epoch = 400. Τι παρατηρείτε με το training και validation accuracy;
2. Δοκιμάστε τους αλγορίθμους βελτιστοποίησης adam, sgd, rmsprop, nadam, adamax, και ftrl. Ποιος δίνει το καλύτερο accuracy;
3. Αλλάξτε την αρχιτεκτονική του δικτύου ώστε να υλοποιεί το παρακάτω σχήμα. Θα χρειαστεί να μετασχηματίσετε τις διαστάσεις των δεδομένων σας από #images × height × width σε #images × height × width × #channels, όπου #images είναι ο αριθμός των εικόνων και #channels=1 είναι ο αριθμός των καναλιών. Επίσης αλλάξτε το epoch από 400 σε 50 για να τελειώνει πιο γρήγορα η εκπαίδευση και χρησιμοποιήστε τον adam optimizer.
4. Προσθέστε Batch Normalization αμέσως μετά τα επίπεδα conv1, conv2, conv3, conv4, conv5, dense1.



   Δοκιμάστε να τοποθετήσετε το Batch Normalization πριν την συνάρτηση ReLU (Δηλαδή Conv χωρίς ReLU -> BatchNormalization -> ReLU).
5. Προσθέστε dropout(0.2) αμέσως μετά το πρώτο maxPooling, dropout(0.3) αμέσως μετά το δεύτερο maxPooling, dropout(0.4) αμέσως μετά το τρίτο maxPooling και dropout(0.5) αμέσως πριν το τελευταίο dense(10) layer.

1. Han Xiao, Kashif Rasul, Roland Vollgraf, Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms, arXiv:1708.07747v1