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 θα γίνει σε **ομάδες έως 2 ατόμων**
3. Ημερομηνία παράδοσης: **Πέμπτη, 27/6/2024 στις 23:00**
4. **Ανεβάστε στο eclass τα ακόλουθα παραδοτέα σε μορφή zip:**
   Παραδοτέα: α) Κώδικας και β) Αναφορά με τις απαντήσεις, παρατηρήσεις, πειράματα, αποτελέσματα και οδηγίες χρήσης του κώδικα.
5. **ΠΡΟΣΟΧΗ: Δεν επιτρέπεται να χρησιμοποιήσετε έτοιμες υλοποιήσεις των αλγορίθμων που σας ζητούνται εκτός αν αναφέρεται ρητά.**
6. **Την εβδομάδα μετά την παράδοση θα γίνει προφορική εξέταση κάθε ομάδας πάνω στα παραδοτέα. Λεπτομέρειες θα ανακοινωθούν στο eclass.**

Θέμα 1: Αλγόριθμος Perceptron

Έστω ότι έχουμε τα ακόλουθα 2D δείγματα από 4 κλάσεις $\omega_1, \omega_2, \omega_3, \omega_4$.

	ω_1		ω_2		ω_3		ω_4	
Δείγμα	x_1	x_2	x_1	x_2	x_1	x_2	x_1	x_2
1	0.1	1.1	7.1	4.2	-3	-2.9	-2	-8.4
2	6.8	7.1	-1.4	-4.3	0.5	8.7	-8.9	0.2
3	-3.5	-4.1	4.5	0	2.9	2.1	-4.2	-7.7
4	2	2.7	6.3	1.6	-0.1	5.2	-8.5	-3.2
5	4.1	2.8	4.2	1.9	-4	2.2	-6.7	-4
6	3.1	5	1.4	-3.2	-1.3	3.7	-0.5	-9.2
7	-0.8	-1.3	2.4	-4	-3.4	6.2	-5.3	-6.7
8	0.9	1.2	2.5	-6.1	-4.1	3.4	-8.7	-6.4
9	5	6.4	8.4	3.7	-5.1	1.6	-7.1	-9.7
10	3.9	4	4.1	-2.2	1.9	5.1	-8	-6.3

Γράψτε ένα πρόγραμμα σε Python που να διαβάζει τα δείγματα των κλάσεων και να υλοποιεί τον αλγόριθμο Perceptron (στην batch μορφή του) για να κάνει ταξινόμηση ανάμεσα σε δύο κλάσεις. Πιο συγκεκριμένα:

a) Θα διαβάζει και θα σχεδιάζει στο ίδιο διάγραμμα τα παραπάνω δείγματα αλλά με διαφορετικό χρώμα για κάθε κλάση.

b) Θα ξεκινάει με μηδενικές τιμές για το διάνυσμα των βαρών $\boldsymbol{a} = \mathbf{0}$ και θα σημειώνει τον αριθμό των επαναλήψεων που απαιτούνται μέχρι να συγκλίνει ο αλγόριθμος.

c) Εφαρμόστε το πρόγραμμά σας διαδοχικά πρώτα για ταξινόμηση των δειγμάτων των κλάσεων ω_1 και ω_2, στη συνέχεια των κλάσεων ω_2 και ω_3 και τέλος των κλάσεων ω_3 και ω_4. Σε όλες τις περιπτώσεις σχεδιάστε τα δείγματα και το όριο απόφασης που βρίσκει ο αλγόριθμος. Εξηγήστε που μπορεί να οφείλεται η διαφορά στον αριθμό των επαναλήψεων που χρειάζεται για να συγκλίνει ο αλγόριθμος στις παραπάνω περιπτώσεις.

Θέμα 2: 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:

$$f(z) = \frac{1}{1 + 2e^{-z}}$$

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:

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

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 $\boldsymbol{\theta}$.

d) 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:

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

i) 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. Θα χρειαστεί να συμπληρώσετε κώδικα python στο αρχείο που σας δίνεται: ***`MyLogit.py`***. Χρησιμοποιήστε μόνο τα modules (`numpy`, `matplotlib`, `scipy.optimize`) σε όλη την άσκηση. Πιο συγκεκριμένα:

ii) Αρχικά διαβάστε και σχεδιάστε τα δείγματά σας από το αρχείο που σας δίνεται.

iii) Συμπληρώστε τον κώδικα της συνάρτησης `sigmoid` ώστε να υλοποιεί την σιγμοειδή συνάρτηση $f(z)$ της άσκησης.

iv) Συμπληρώστε τις συναρτήσεις `costFunction` και `gradient` έτσι ώστε να επιστρέφει το κόστος $J(\boldsymbol{\theta})$ (J) και την κλίση $\nabla J(\boldsymbol{\theta})$ (grad) όπως ορίζονται στο ερώτημα (α).

v) Η βελτιστοποίηση των παραμέτρων γίνεται με κώδικα που υπάρχει έτοιμος στην άσκηση. Εσείς απλά τρέξτε τον κώδικα και βρείτε το σύνορο απόφασης με την συνάρτηση. Επίσης συμπληρώστε τον κώδικα στην συνάρτηση `predict` για να προβλέψετε αν ο φοιτητής θα γίνει δεκτός με βάση

διάφορες τιμές βαθμών στις δύο εξετάσεις. Ελέγξτε το αποτέλεσμα για έναν φοιτητή που έγραψε στο πρώτο μάθημα 45 και στο δεύτερο 85. Ποια είναι η πιθανότητα να γίνει δεκτός;

Θέμα 3: Εκτίμηση Παραμέτρων με Maximum Likelihood

Έστω ότι έχουμε τα ακόλουθα 3D δείγματα από 3 κλάσεις $\omega_1, \omega_2, \omega_3$ τα οποία ακολουθούν Gaussian κατανομές σε όλες τις διαφορετικές διαστάσεις.

	$\boldsymbol{\omega_1}$			$\boldsymbol{\omega_2}$		
Δείγμα	$\boldsymbol{x_1}$	$\boldsymbol{x_2}$	$\boldsymbol{x_3}$	$\boldsymbol{x_1}$	$\boldsymbol{x_2}$	$\boldsymbol{x_3}$
1	0.42	-0.087	0.58	-0.4	0.58	0.089
2	-0.2	-3.3	-3.4	-0.31	0.27	-0.04
3	1.3	-0.32	1.7	0.38	0.055	-0.035
4	0.39	0.71	0.23	-0.15	0.53	0.011
5	-1.6	-5.3	-0.15	-0.35	0.47	0.034
6	-0.029	0.89	-4.7	0.17	0.69	0.1
7	-0.23	1.9	2.2	-0.011	0.55	-0.18
8	0.27	-0.3	-0.87	-0.27	0.61	0.12
9	-1.9	0.76	-2.1	-0.065	0.49	0.0012
10	0.87	-1	-2.6	-0.12	0.054	-0.063

a) Γράψτε ένα πρόγραμμα σε python το οποίο θα υπολογίζει τις τιμές μεγίστης πιθανοφάνειας $\hat{\mu}$ και $\hat{\sigma}^2$. Εφαρμόστε το πρόγραμμά σας ξεχωριστά για κάθε ένα από τα 3 χαρακτηριστικά x_i της κλάσης $\boldsymbol{\omega_1}$ στον παραπάνω πίνακα.

b) Θεωρώντας ότι ανά δύο τα χαρακτηριστικά της κλάσης $\boldsymbol{\omega_1}$ ακολουθούν 2D κανονική κατανομή $\boldsymbol{p(x) \sim \mathcal{N}(\mu, \Sigma)}, \boldsymbol{x} \in \mathbb{R}^2$, δημιουργήστε ένα πρόγραμμα που θα υπολογίζει τις παραμέτρους της κανονικής κατανομής με τη μέθοδο της μεγίστης πιθανοφάνειας. Εφαρμόστε το πρόγραμμά σας και στα 3 ζευγάρια χαρακτηριστικών που υπάρχουν.

c) Θεωρώντας ότι το συνολικό διάνυσμα χαρακτηριστικών της κλάσης $\boldsymbol{\omega_1}$ ακολουθεί 3D κανονική κατανομή $\boldsymbol{p(x) \sim \mathcal{N}(\mu, \Sigma)}$, $\boldsymbol{x} \in \mathbb{R}^3$, δημιουργήστε ένα πρόγραμμα που θα υπολογίζει τις παραμέτρους της κανονικής κατανομής με τη μέθοδο της μεγίστης πιθανοφάνειας. Εφαρμόστε και πάλι το πρόγραμμά σας στα δείγματα της κλάσης $\boldsymbol{\omega_1}$.

d) Υποθέστε ότι το τρισδιάστατο μοντέλο σας είναι διαχωρίσιμο, έτσι ώστε $\boldsymbol{\Sigma} = diag(\sigma_1^2, \sigma_2^2, \sigma_3^2)$. Γράψτε ένα πρόγραμμα που θα εκτιμάει με τη μέθοδο μεγίστης πιθανοφάνειας το μέσο $\widehat{\boldsymbol{\mu}}$ και τα διαγώνια στοιχεία του $\widehat{\boldsymbol{\Sigma}}$. Εφαρμόστε το πρόγραμμά σας στα δεδομένα της κλάσης $\boldsymbol{\omega_2}$.

e) Συγκρίνετε τα αποτελέσματά της εκτίμησης που βρήκατε για το διάνυσμα του μέσου $\widehat{\boldsymbol{\mu}} = [\hat{\mu}_1, \hat{\mu}_2, \hat{\mu}_3]$ και για τις διακυμάνσεις $\hat{\sigma}_i^2, i = 1, \cdots, 3$ σε κάθε μία από τις παραπάνω περιπτώσεις που εφαρμόσατε. Απαντήστε αν τα αποτελέσματα που βρήκατε είναι ίδια ή διαφορετικά και εξηγήστε γιατί.

Θέμα 4: Ομαδοποίηση (Clustering) με K-means και GMM

Σε αυτή την άσκηση θα υλοποιήσετε τον K-means clustering αλγόριθμο και θα τον χρησιμοποιήσετε για να συμπιέσετε μια εικόνα. Ο K-means είναι ένας επαναληπτικός αλγόριθμος 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. Στα πλαίσια της άσκησης θα χρειαστεί να συμπληρώσετε τον κώδικα που λείπει στο αρχείο `myImageCompression.py`. Πιο συγκεκριμένα, θα πρέπει να συμπληρώσετε τα επόμενα μέρη του προγράμματος:

a) Την συνάρτηση ***findClosestCentroids***

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***

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) Εμφανίστε και συγκρίνετε την συμπιεσμένη εικόνα. Μπορείτε να χρησιμοποιήσετε την εικόνα '**Fruit.png**' που σας δίνεται ή οποιαδήποτε δική σας. Δοκιμάστε να τρέξετε τον αλγόριθμο για διαφορετικό αριθμό κλάσεων/χρωμάτων K και κάντε παρατηρήσεις για το αποτέλεσμα που παράγεται (συμπιεσμένη εικόνα) σε συνάρτηση με τον όγκο δεδομένων που απαιτούνται για την συμπιεσμένη εικόνα.

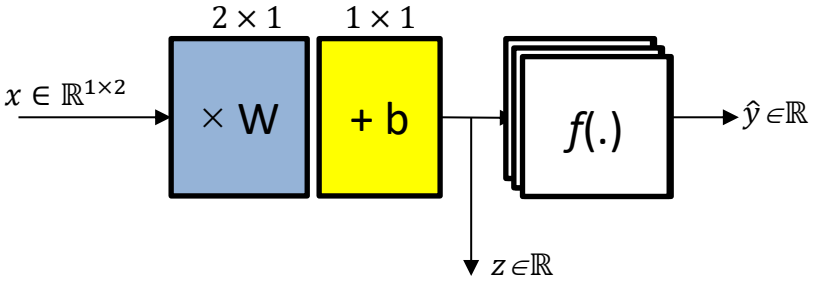
Θέμα 5: Νευρωνικά δίκτυα

Επιλέξτε ένα από τα δύο παρακάτω θέματα. Το πρώτο (**5α**) συνδυάζει θεωρία και υλοποίηση ενός απλού νευρωνικού δικτύου χωρίς τη χρήση έτοιμων προγραμμάτων, ενώ το δεύτερο (**5β**) είναι καθαρά εφαρμοσμένο και χρησιμοποιεί τη βιβλιοθήκη tensorflow/keras και πιο πολύπλοκα δίκτυα.

Θέμα 5α: 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).

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

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

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

β) Να δείξετε ότι

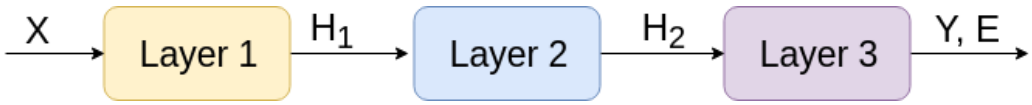
$$\frac{\partial J}{\partial z^{(i)}} = -y^{(i)} + \hat{y}^{(i)}, \qquad i \in batch$$

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:

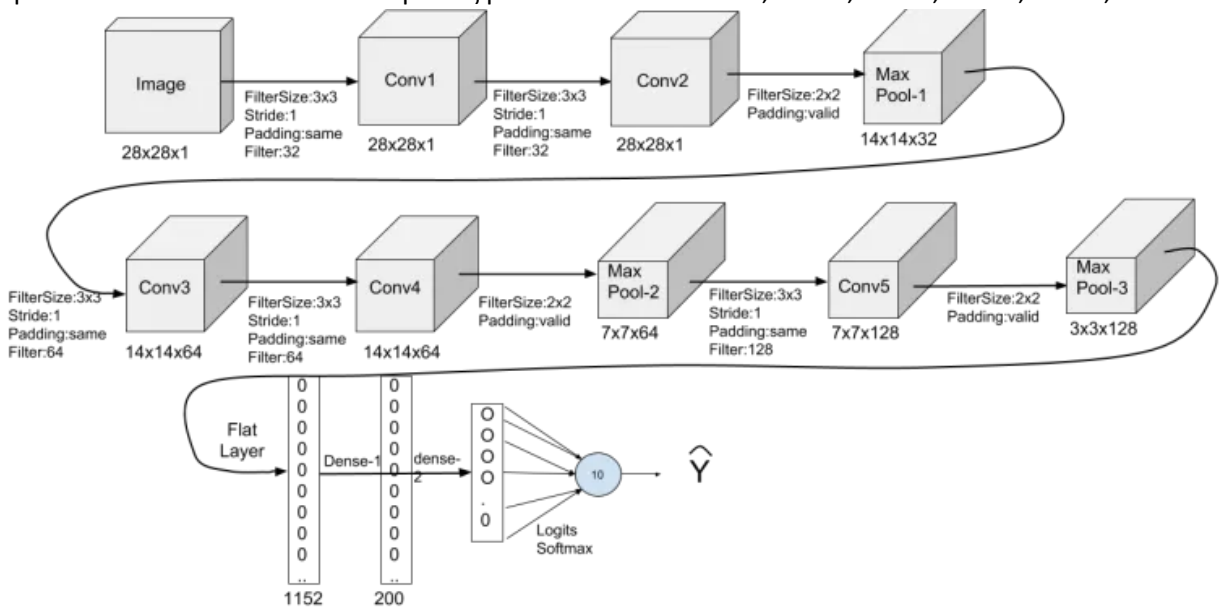
1) Learning Rate/Αριθμός epochs/Το batch size.
2) Χρησιμοποιήστε συνάρτηση ενεργοποίησης tanh και συνάρτηση σφάλματος MSE (Mean Square Error)
3) Διαφορετικές αρχιτεκτονικές δικτύου (π.χ. περισσότερα layers ή/και διαφορετικό αριθμό νευρώνων στο hidden layer.

Θέμα 5β: Convolutional Neural Networks for Image Recognition

Σε αυτή την άσκηση θα δοκιμάσουμε διάφορους ταξινομητές νευρωνικών δικτύων από την βιβλιοθήκη tensorflow/keras 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