6. Μηχανική Μάθηση με Νευρωνικά Δίκτυα

***Σύνοψη**: Στο έκτο κεφάλαιο θα μελετήσουμε τους αλγορίθμους συστάσεων που βασίζονται σε τεχνητά νευρωνικά δίκτυα. Συγκεκριμένα, θα περιγράψουμε το **Multi-layer Perceptron - (MLP)**, το βασικό νευρωνικό δίκτυο, πρόσθιας τροφοδότησης και με πολλά στρώματα ανάλυσης. Επίσης, θα παρουσιάσουμε λεπτομερώς τα **Graph Convolutional Netwoks**, τα **Graph Neural Networks** και τα **Node Embeddings**. Τα **Convolutional Neural Networks** είναι ιδιαίτερα αποτελεσματικά στο **image processing and recognition** και, δεδομένου ότι μια εικόνα συνήθως απεικονίζεται μ' έναν δισδιάστατο πίνακα, αυτά τα τεχνητά νευρωνικά δίκτυα μπορούν κατά αναλογία να εφαρμοστούν και στα συστήματα συστάσεων, τα οποία συνίστανται κατά βάση επίσης από δύο διαστάσεις (χρήστες και στοιχεία). Τέλος, θα αναλύσουμε το **Recurrent Neural Network**, το οποίο επιτρέπει στο σύστημα συστάσεων με τη χρήση των δομικών μονάδων **LSTM** και **GRU** να "ξεχνάει" παλαιότερες χρονικά αλληλεπιδράσεις του χρήστη με τα στοιχεία, και γι' αυτό είναι χρήσιμο για συστάσεις στοιχείων των οποίων ο χρόνος ζωής είναι σχετικά μικρός (π.χ. ειδησεογραφικά άρθρα).*

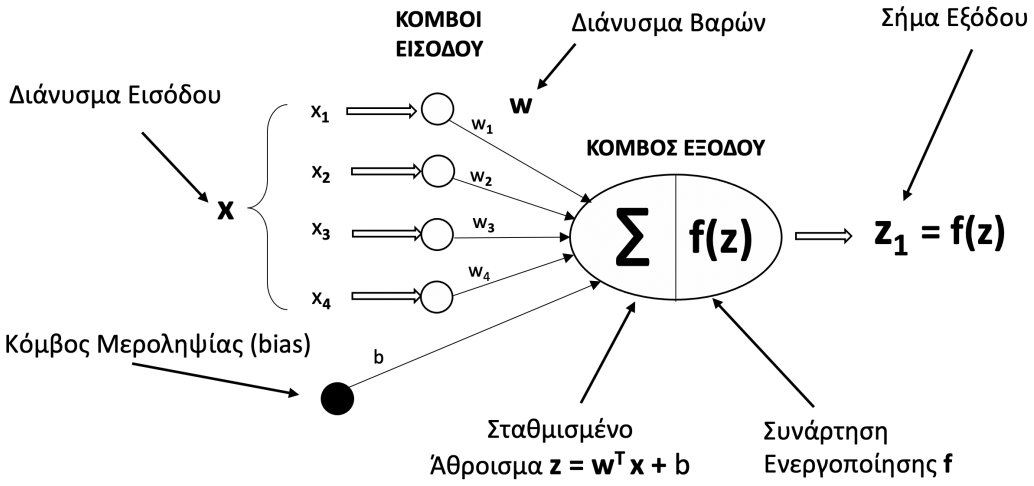
***Προαπαιτούμενη γνώση**: Προτείνεται η πρότερη μελέτη του πέμπτου κεφαλαίου, επειδή κάποιοι αλγόριθμοί του θα υλοποιηθούν με νευρωνικά δίκτυα στο παρόν κεφάλαιο.*

6.1 Νευρωνικά δίκτυα τύπου Perceptron

6.1.1 Περιγραφή ενός Single-layer Perceptron

Στον ανθρώπινο εγκέφαλο η νευρική σύναψη είναι ένας σύνδεσμος που επιτρέπει σ' έναν νευρώνα (ή αλλιώς νευρικό κύτταρο) να μεταδώσει ένα ηλεκτρικό ή χημικό σήμα σ' έναν άλλο νευρώνα, επιφέροντας κάποια μεταβολή της κατάστασης του δεύτερου. Στα βιολογικά συστήματα η **μάθηση** πραγματοποιείται μέσω της αύξησης της δύναμης (π.χ. του πάχους) των συνάψεων ως αποτέλεσμα της ενίσχυσής τους από την επανάληψη ίδιων εξωτερικών ερεθισμάτων.

Τα τεχνητά νευρωνικά δίκτυα, τώρα, προσομοιώνουν τον ανθρώπινο εγκέφαλο με τη χρήση νευρώνων (ή αλλιώς κόμβων) οι οποίοι συνδέονται μεταξύ τους μέσω συνδέσμων που ονομάζονται **νευρικές συνάψεις (synapses)**. Η πιο βασική υπολογιστική δομή στα *τεχνητά νευρωνικά δίκτυα* είναι το *αντίληπτρο* (*perceptron*), το οποίο περιέχει ένα σύνολο κόμβων εισόδου και έναν κόμβο εξόδου. Η αρχιτεκτονική δομή ενός *perceptron* φαίνεται στο Σχήμα 6.1.



Σχήμα 6.1: Παράδειγμα ενός Single-Layer Perceptron με τη χρήση bias.

Το *perceptron* λοιπόν δέχεται ένα σύνολο εισόδων, εκτελεί ορισμένους μαθηματικούς υπολογισμούς και δίνει μία έξοδο. Τα εισερχόμενα σήματα αναπαριστώνται ως διάνυσμα εισόδου:

$$\mathbf{x} = [x_1, x_2, \ldots, x_n], \quad x_i \in \mathbb{R}.$$

Οι νευρώνες που ανήκουν σε διαφορετικά **στρώματα ενός νευρωνικού δικτύου** (Neural Network Layers) συνδέονται μεταξύ τους μέσω *κατάλληλων συνδέσμων* (ή αλλιώς, *συνάψεων*) που αναπαριστώνται με τη χρήση κατευθυνόμενων ακμών ενός γράφου. Η ισχύς των *συνάψεων* εκφράζεται με τη χρήση *βαρών (weights)* επί των κατευθυνόμενων ακμών του γράφου. Τα *βάρη* πάνω στις κατευθυνόμενες ακμές που συνδέεουν έναν νευρώνα ενός επιπέδου μ' έναν νευρώνα του επόμενου επιπέδου ομαδοποιούνται σ' ένα *διάνυσμα βαρών* ή *σταθμίσεων*:

$$\mathbf{w} = [w_1, w_2, \ldots, w_n]^T, \quad w_i \in \mathbb{R}.$$

Ο πρώτος υπολογισμός που εκτελεί ένα *perceptron* είναι το **σταθμισμένο άθροισμα (weighted sum)**. Πιο συγκεκριμένα, πολλαπλασιάζει κάθε είσοδο με το αντίστοιχο βάρος της. Στη συνέχεια αθροίζονται όλες οι είσοδοι και προστίθεται ένας όρος που ονομάζεται **μεροληψία (bias)**, όπως φαίνεται στο κέντρο του Σχήματος 6.1 και μοντελοποιείται ως εσωτερικό γινόμενο:

$$\mathbf{z} = \sum_{i=1}^{n} w_i x_i = \mathbf{w}^T \mathbf{x} + b$$

Ο δεύτερος υπολογισμός που εκτελεί ο νευρώνας ονομάζεται *ενεργοποίηση*. Αυτό γίνεται παίρνοντας την έξοδο του σταθμισμένου αθροίσματος και περνώντας τη μέσα από μια *συνάρτηση ενεργοποίησης* f (*activation function* f), όπως φαίνεται στο δεξιό μέρος του Σχήματος 6.1.

$$z_1 = f(z) = f\left(\mathbf{w}^T\mathbf{x} + b\right) = f\left(\sum_{i=1}^{N} w_i x_i + b\right)$$

Η επιλογή της κατάλληλης συνάρτησης ενεργοποίησης f αποτελεί κρίσιμο μέρος του σχεδιασμού νευρωνικών δικτύων. Τονίζουμε ότι η πιο βασική συνάρτηση ενεργοποίησης $f(\cdot)$ είναι η *ταυτοτική* (*identity activation function*), η οποία όμως δεν παρέχει *μη γραμμικότητα* (*non-linearity*), και λειτουργεί μόνο για τις περιπτώσεις όπου τα δεδομένα μπορούν να διαχωριστούν σε δύο ομάδες με τη χρήση μια απλής γραμμής:

$$f(x) = x$$

Η *γραμμική συνάρτηση ενεργοποίησης* χρησιμοποιείται συχνά στον κόμβο εξόδου του νευρωνικού δικτύου, όταν η πρόβλεψή μας αφορά σ' έναν πραγματικό αριθμό. Για παράδειγμα, ένα σύστημα συστάσεων προσπαθεί να προβλέψει τη βαθμολογία που θα έβαζε ένας χρήστης σ' ένα στοιχείο. Σ' αυτήν την περίπτωση η πρόβλεψη της βαθμολογίας είναι μια μεταβλητή που εκφράζεται μ' έναν πραγματικό αριθμό, και συνεπώς είναι λογικό να χρησιμοποιηθεί η γραμμική συνάρτηση ενεργοποίησης. Ωστόσο, ο παραγόμενος αλγόριθμος του νευρωνικού δικτύου είναι ο ίδιος με την *παλινδρόμηση ελαχίστων τετραγώνων* (*least squares regression*).

Σημειώνουμε εδώ ότι το βασικό πλεονέκτημα των νευρωνικών δικτύων έναντι των άλλων μεθόδων πρόβλεψης είναι οι περιπτώσεις κατά τις οποίες τα πρώτα χρησιμοποιούν συναρτήσεις ενεργοποίησης που μπορούν να εκφράσουν μη γραμμικές (non-linear) συσχετίσεις μεταξύ των δεδομένων, κυρίως σε καταστάσεις όπου δεν διαχωρίζονται τα δεδομένα σε δύο ομάδες με τη χρήση μια απλής γραμμής. Τέτοιες κλασικές, μη γραμμικές συναρτήσεις ενεργοποίησης είναι η *σιγμοειδής συνάρτηση (sigmoid function)*και η *συνάρτηση υπερβολικής εφαπτομένης (tahn function)*:

$$f(x) = \frac{1}{1+e^{-x}} \qquad \text{(sigmoid function)}$$

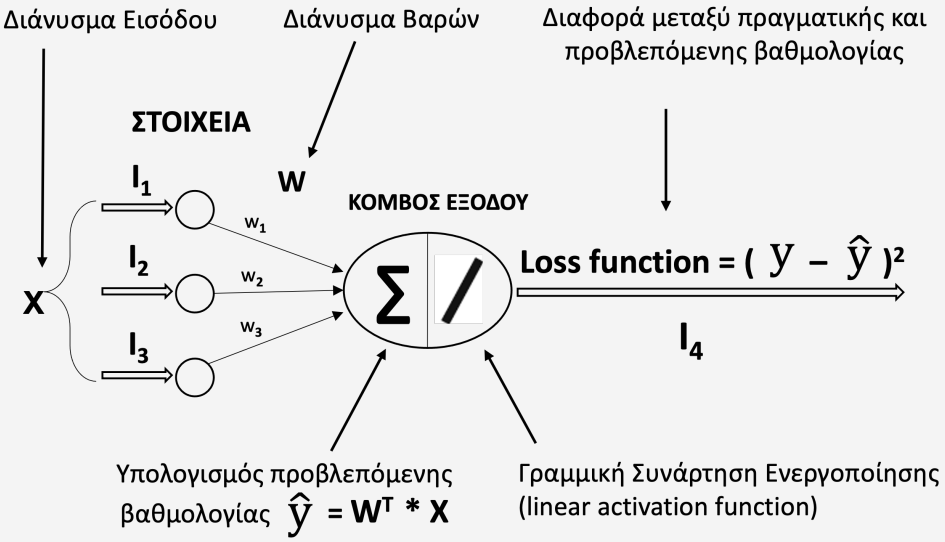
$$f(x) = \frac{e^{2x}-1}{e^{2x}+1} \qquad \text{(tanh function)}$$

Τονίζουμε ότι τα περισσότερα βασικά μοντέλα μηχανικής μάθησης μπορούν να υλοποιηθούν με απλές αρχιτεκτονικές νευρωνικών δικτύων. Παρακάτω θα "μοντελοποιήσουμε" τις δύο παραδοσιακές τεχνικές μηχανικής μάθησης (*γραμμική παλινδρόμηση* και *ανάλυση πίνακα μέσω αποσύνθεσης κατά UV*) ως αρχιτεκτονικές νευρωνικών δικτύων, ώστε να καταδείξουμε το πώς η **βαθιά μάθηση** (*deep learning*) -δηλαδή αυτή που έχει πολλά επίπεδα επεξεργασίας της πληροφορίας, άρα πολλά επίπεδα νευρώνων- μπορεί να αποτελέσει μια ειδική περίπτωση της παραδοσιακής μηχανικής μάθησης.

Παράδειγμα 6.1 Δίνεται ο πίνακας βαθμολογιών χρηστών-στοιχείων του Πίνακα 6.2 και μας ζητείται να προβλέψουμε τη βαθμολογία του χρήστη U_4 για το στοιχείο I_4 με τη χρήση ενός *single-layer* και ενός *multi-layer perceptron*. Η αρχιτεκτονική του νευρωνικού δικτύου που θα χρησιμοποιηθεί για την πρόβλεψη της τιμής με τη χρήση ενός *single-layer perceptron* εμφανίζεται στο Σχήμα 6.3. Το προαναφερθέν νευρωνικό δίκτυο (βλ. Σχήμα 6.3) μοντελοποιεί τον *αλγόριθμο συνεργατικού φιλτραρίσματος βάσει στοιχείου* (*item-based collaborative filtering*).

	I_1	I_2	I_3	I_4
U_1	4	1	1	4
U_2	1	4	2	0
U_3	2	1	4	5
U_4	1	4	1	?

Σχήμα 6.2: Πίνακας βαθμολογιών Χρήστη-Στοιχείου Α (4×4).



Σχήμα 6.3: Παράδειγμα ενός *Single-Layer Perceptron* για την υλοποίηση του αλγορίθμου *Item-based Collaborative Filtering* επί των δεδομένων του Παραδείγματος 6.1.

Αναλυτικότερα, το νευρωνικό δίκτυο του Σχήματος 6.3 έχει $m-1$ κόμβους εισόδου $\{I_1, I_2, I_3\}$ όπου m ο αριθμός των στοιχείων του Παραδείγματος 6.1. Οι βαθμολογίες των στοιχείων εισόδου χρησιμοποιούνται για την πρόβλεψη της βαθμολογίας του εναπομείναντος τέταρτου στοιχείου I_4, όπως φαίνεται στο Σχήμα 6.3. Η μεταβλητή $\hat{y}$ εκφράζει τη βαθμολογία που προβλέπει το νευρωνικό δίκτυο για τα στοιχεία, ενώ η μεταβλητή y αντιπροσωπεύει τις πραγματικές βαθμολογίες των στοιχείων. Ο υπολογισμός της προβλεπόμενης βαθμολογίας ενός στοιχείου $\hat{y}$ υπολογίζεται ως το

εσωτερικό γινόμενο του διανύσματος εισόδου $\mathbf{X}$ και του διανύσματος βαρών $\mathbf{W}$ του νευρωνικού δικτύου ως ακολούθως:

$$\hat{y} = \mathbf{W}^T \cdot \mathbf{X}$$

Όπως φαίνεται στον κόμβο εξόδου του Σχήματος 6.3, η *συνάρτηση ενεργοποίησης* του νευρωνικού δικτύου είναι η *γραμμική* ή αλλιώς *ταυτοτική συνάρτηση* $f(x) = x$. Επίσης, (βλ. Σχήμα 6.3) ο στόχος της *συνάρτησης απώλειας* (*loss function*) L είναι να ελαχιστοποιηθεί η διαφορά ή αλλιώς το σφάλμα (error ή e) μεταξύ των πραγματικών και των προβλεπόμενων βαθμολογιών των στοιχείων:

$$L = e^2 = \arg\min_{\mathbf{W}} \sum_{(\mathbf{X},y)\in\mathcal{D}} (y - \hat{y})^2$$

όπου $(\mathbf{X}, y)$ το ζεύγος των δεδομένων εκπαίδευσης $\mathcal{D}$ που εισάγονται στο νευρωνικό δίκτυο. Η διαφορά μεταξύ πραγματικής και προβλεπόμενης βαθμολογίας χρησιμοποιείται για την ενημέρωση του διανύσματος βαρών $\mathbf{W}$ με τρόπο παρόμοιο με τη διαδικασία βελτιστοποίησης που ακολουθείται στην *παλινδρόμηση ελαχίστων τετραγώνων*, και η οποία περιγράφηκε λεπτομερώς στο προηγούμενο κεφάλαιο. Πιο συγκεκριμένα, η βελτιστοποίηση των παραμέτρων του διανύσματος βαρών $\mathbf{W}$ είναι αντίστοιχη των ενημερώσεων που βασίζονται στη *μέθοδο της επικλινούς καθόδου* (*gradient descent*) όπως αυτή εφαρμόζεται στην *παλινδρόμηση ελαχίστων τετραγώνων*. Η μέθοδος αυτή (*gradient descent*) βρίσκει την κατεύθυνση προς την οποία θα πρέπει να αλλάξει το διάνυσμα βαρών $\mathbf{W}$, έτσι ώστε να ελαχιστοποιηθεί η *συνάρτηση σφάλματος* (*loss function* L). Σημειώστε ότι οι ενημερώσεις της *μεθόδου επικλινούς καθόδου* (*gradient descent*), γνωστής και ως **κανόνας δέλτα**, προσδιορίζονται με τον υπολογισμό της μερικής παραγώγου της *συνάρτησης απώλειας* e^2 ως προς το διάνυσμα βαρών $\mathbf{W}$. Σημειώνουμε ότι το σύνολο των μερικών παραγώγων που αφορά σ' όλες τις μεταβλητές της *συνάρτησης απώλειας* e^2 καλείται *Ανάδελτα* (*Gradient*). Συνεπώς, κάθε φορά που ένα ζεύγος δεδομένων εκπαίδευσης $(\mathbf{X}, y)$ εισάγεται στο νευρωνικό δίκτυο, η *κλίση* (*Ανάδελτα*) της *συνάρτησης απώλειας* μπορεί να υπολογιστεί ως εξής:

$$\frac{\partial e^2}{\partial \mathbf{W}} = -2 \cdot e \cdot \mathbf{X}$$

Κινούμενοι κάθε φορά στην αντίθετη κατεύθυνση της παραπάνω υπολογισθείσας μερικής παραγώγου, μπορούμε να ελαχιστοποιήσουμε τη *συνάρτηση απώλειας* e^2 με τον ακόλουθο κανόνα ενημέρωσής της:

$$\mathbf{W}^{t+1} = \mathbf{W}^t + 2 \cdot \eta \cdot e \cdot \mathbf{X}$$

όπου $\eta \in (0, 1]$ ο ρυθμός μάθησης και W^t η τιμή του διανύσματος βαρών στην t-οστή επανάληψη ή αλλιώς τη χρονική στιγμή t της εκτέλεσης του αλγορίθμου. Όπως φαίνεται στο Σχήμα 6.3 στην είσοδο του νευρωνικού δικτύου εισάγουμε σταδιακά στο διάνυσμα εισόδου $\mathbf{X}$ όλες τις παρατηρούμενες βαθμολογίες των στοιχείων $\{I_1, I_2, I_3\}$ του Σχήματος 6.2, δηλαδή τις ακόλουθες τριπλέτες:

$$\{4, 1, 1\}, \{1, 4, 2\}, \{2, 1, 4\} \text{ και } \{1, 4, 1\}.$$

Το νευρωνικό δίκτυο του Σχήματος 6.3 χρησιμοποιείται για την πρόβλεψη της βαθμολογίας του στοιχείου I_4. Ειδικότερα, μετά την εισαγωγή μιας τριπλέτας δεδομένων εκπαίδευσης στην είσοδο του νευρωνικού δικτύου -και η οποία αφορά στις βαθμολογίες του εκάστοτε υπό εξέταση χρήστη για τα στοιχεία $\{I_1, I_2, I_3\}$- το νευρωνικό δίκτυο προβλέπει μια βαθμολογία για το στοιχείο I_4. Εν συνεχεία, συγκρίνουμε τη βαθμολογία που προέβλεψε το νευρωνικό δίκτυο για το στοιχείο I_4 με την πραγματική βαθμολογία του στοιχείου, η οποία βρίσκεται στην 4η στήλη του πίνακα βαθμολογιών χρηστών και στοιχείων A (βλ. Σχήμα 6.2). Άρα, οι προβλέψεις των βαθμολογιών που προκύπτουν από το νευρωνικό δίκτυο χρησιμοποιούνται για τη δημιουργία ενός νέου πίνακα $\hat{A}$ με τις κατά προσέγγιση βαθμολογίες χρηστών-στοιχείων, ο οποίος συγκρίνεται με τον αρχικό πίνακα A. Εφόσον υπάρξει διαφορά ως προς τη βαθμολογία, τότε διορθώνουμε αναλόγως το διάνυσμα βαρών $\mathbf{W}$ με την **τεχνική της Οπίσθιας Διάδοσης (Back Propagation)**, η οποία διαδίδει ανάστροφα (από την έξοδο του νευρωνικού προς την είσοδό του) τις τιμές εκείνες που ελαχιστοποιούν το άθροισμα των τετραγωνικών σφαλμάτων μεταξύ των πραγματικών και των προβλεπόμενων βαθμολογιών που προβλέπει το νευρωνικό στην έξοδό του. Αυτή η προσέγγιση επαναλαμβάνεται επαναληπτικά μέχρι τη σύγκλιση των τιμών του πίνακα $\hat{\boldsymbol{A}} = \mathbf{W}^T \cdot \mathbf{X}$, ώστε αυτές να μη μεταβάλλονται άλλο. Σημειώστε ότι κάθε επανάληψη απαιτεί την εισαγωγή n δεδομένων εκπαίδευσης (n τριπλετών) στο νευρωνικό δίκτυο όπου n ο αριθμός των χρηστών. Στο παράδειγμά μας, σε κάθε επανάληψη εισάγονται στο νευρωνικό δίκτυο 4 τριπλέτες, δεδομένου ότι έχουμε 4 χρήστες που έχουν αξιολογήσει τα στοιχεία $\{I_1, I_2, I_3\}$ των οποίων οι βαθμολογίες χρησιμοποιούνται για να προβλέψουμε τη βαθμολογία του στοιχείου I_4.

Το κύριο μειονέκτημα του *perceptron* είναι ότι δεν μπορεί να χωρίσει μη γραμμικώς διαχωρίσιμα δεδομένα μιας και αποτελείται μόνο από το στρώμα των κόμβων εισόδου και έναν μόνο κόμβο εξόδου. Για να γίνει κατανοητό το παραπάνω μειονέκτημα θα παρουσιάσουμε το κάτωθι παράδειγμα:

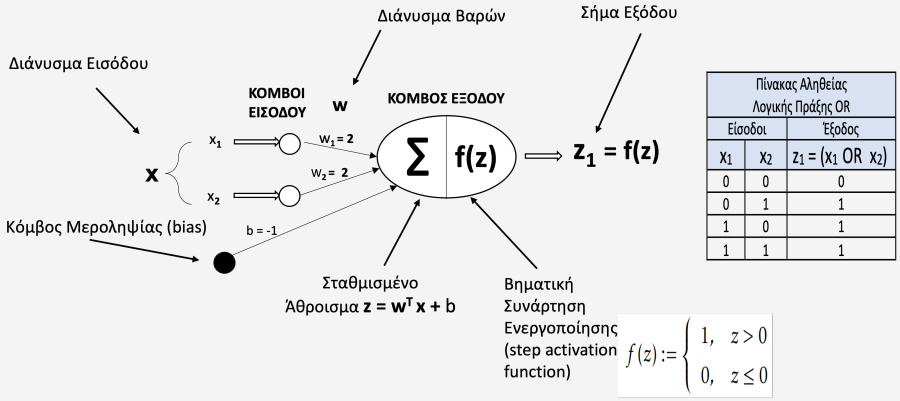
Παράδειγμα 6.2 Έστω, λοιπόν, ότι μας δίνεται το *perceptron* δύο εισόδων του Σχήματος 6.4 που υλοποιεί τη *λογική πράξη OR* χρησιμοποιώντας ως *συνάρτηση ενεργοποίησης* τη *βηματική συνάρτηση* (*step function*) με *κατώφλι* (*threshold*) $t = 0$. Οι τιμές εισόδου του *perceptron* μπορεί να είναι 0 (ψευδές) ή 1 (αληθές). Εάν λόγου χάρη το ζευγάρι των τιμών εισόδου του *perceptron* είναι $\{x_1 = 0, x_2 = 1\}$, τότε το σταθμισμένο άθροισμά του είναι:

$$z = w_1 * x_1 + w_2 * x_2 + b = 2 * 0 + 2 * 1 - 1 = 1$$

και η έξοδος της βηματικής συνάρτησής του είναι $f(z) = 1$, επειδή $z > 0$. Συνεπώς, η έξοδος του perceptron είναι 1 (Αληθές), γεγονός που είναι σύμφωνο με τον πίνακα αληθείας της λογικής πράξης OR όπως φαίνεται στο δεξιό μέρος του Σχήματος 6.4. Εναλλακτικά, εάν το ζευγάρι των τιμών εισόδου του *perceptron* είναι $\{x_1 = 0, x_2 = 0\}$, τότε το σταθμισμένο άθροισμά του είναι:

$$z = w_1 * x_1 + w_2 * x_2 + b = 2 * 0 + 2 * 0 - 1 = -1$$

και η έξοδος της *βηματικής συνάρτησης ενεργοποίησής* του είναι: $f(z) = -1$, επειδή $z < 0$. Ως εκ τούτου, η έξοδος του *perceptron* είναι -1 (Ψευδές), πράγμα που είναι σύμφωνο με τον *πίνακα αληθείας* της *λογικής πράξης OR* όπως φαίνεται στο δεξιό μέρος του Σχήματος 6.4.



Σχήμα 6.4: Παράδειγμα ενός *perceptron* δύο εισόδων που υλοποιεί τη *λογική πράξη OR*.

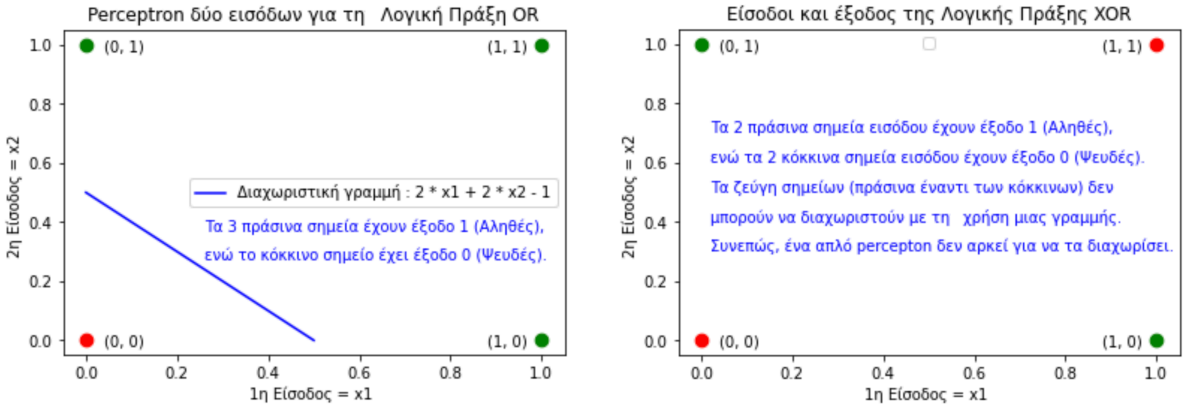
Θα πρέπει να υπογραμμίσουμε εδώ ότι το *perceptron* του Σχήματος 6.4 μπορεί να ξεχωρίζει τα σημεία ενός καρτεσιανού συστήματος αξόνων χ-y μεταξύ δύο ημιχώρων που διαχωρίζονται από ένα υπερεπίπεδο και το οποίο διαμορφώνεται από τις τιμές x των εισόδων του *perceptron* για τις οποίες ισχύει η παρακάτω ισότητα:

$$w_1 * x_1 + w_2 * x_2 + b = 0$$

Έτσι, όπως φαίνεται από τη μπλε διαχωριστική γραμμή του Σχήματος 6.5(α), το *perceptron* του παραδείγματος 6.2 διαχωρίζει τον χώρο του καρτεσιανού συστήματος αξόνων χ-y σε δύο ημιχώρους με την παρακάτω γραμμική εξίσωση:

$$2 * x_1 + 2 * x_2 - 1 = 0$$

Τονίζουμε, εδώ, ότι οι συντελεστές $w_1 = 2$, και $w_2 = 2$ είναι αυτοί που ευθύνονται για την κλίση της ευθείας και το b = -1 (bias) είναι η μετατόπισή της στο καρτεσιανό σύστημα αξόνων x-y. Επίσης, υπογραμμίζουμε ότι υπάρχουν πολλοί διαφορετικοί συνδυασμοί των παραπάνω συντελεστών που κάνουν τον διαχωρισμό των εισόδων και της εξόδου της *λογικής πράξης OR* σε ψευδείς και αληθείς τιμές. Αυτός άλλωστε είναι και ο βασικός στόχος που προσπαθεί να πετύχει η **μηχανική μάθηση (machine learning)**: να βρεθούν, δηλαδή, οι καταλληλότεροι συντελεστές βαρών w και ο κατάλληλος *συντελεστής μεροληψίας* b, ώστε να διαχωριστούν τα δεδομένα με τον βέλτιστο τρόπο. Ωστόσο, υπάρχουν ορισμένα προβλήματα τα οποία δεν μπορούν να επιλυθούν από ένα *perceptron*, και θα δώσουμε παρακάτω ένα σχετικό παράδειγμα.



(α) (β)

Σχήμα 6.5: Γεωμετρική αναπαράσταση των εισόδων και των εξόδων της *Λογικής Πράξης* (α) *OR* και (β) *XOR*.

Παράδειγμα 6.3 Έστω ότι μας ζητείται να υλοποιήσουμε την *πύλη XOR* με τη βοήθεια ενός *perceptron*, η οποία (πύλη) βγάζει έξοδο 1 (Αληθές) εφόσον η μία από τις εισόδους της είναι 1, ενώ η άλλη είσοδος είναι 0, σύμφωνα με τον παρακάτω *πίνακα αληθείας* της *λογικής πράξης XOR*:

Είσοδοι		Έξοδος
x_1	x_2	x_1 XOR x_2
0	0	0
0	1	1
1	0	1
1	1	0

Όπως φαίνεται στο Σχήμα 6.5(β), τα σημεία εισόδων της *λογικής πράξης XOR* είναι αδύνατον να διαχωριστούν με τη βοήθεια μιας γραμμής. Γι' αυτόν τον λόγο αναπτύχθηκαν νευρωνικά δίκτυα τα οποία έχουν περισσότερα ενδιάμεσα στρώματα νευρώνων, ώστε να μπορούν να διαχωρίσουν τα δεδομένα, τα οποία περιλαμβάνονται είτε σε *κυρτές περιοχές λύσεων* είτε και σε *μη-κυρτές*, και με τα οποία θα ασχοληθούμε λεπτομερώς στην επόμενη Ενότητα.[α]

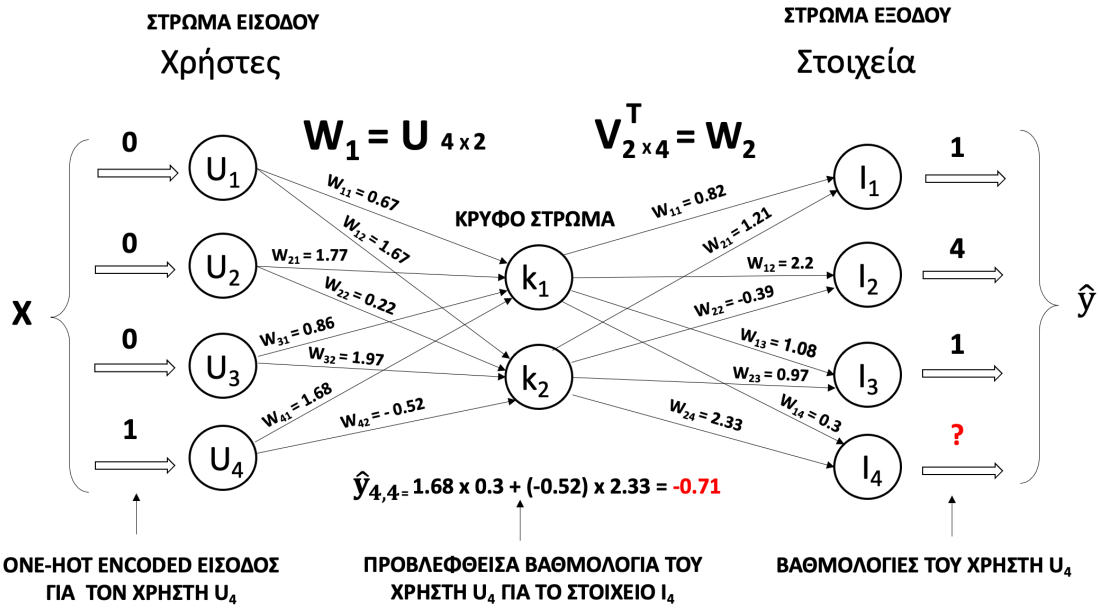
[α] *Κυρτές περιοχές λύσεων* είναι εκείνες που δεν διαθέτουν ένα ευθύγραμμο τμήμα που να έχει και τα δυο του άκρα μέσα στην περιοχή και κάποια σημεία του εκτός. Σε αντίθετη περίπτωση η περιοχή ονομάζεται *μη κυρτή*.

6.1.2 Παρουσίαση ενός Multi-layer Perceptron

Στα *νευρωνικά δίκτυα πολλαπλών στρωμάτων* οι νευρώνες είναι τοποθετημένοι σε στρώματα, εκ των οποίων αυτά της εισόδου και εξόδου διαχωρίζονται από μια άλλη ομάδα πρόσθετων ενδιά-

μεσων στρωμάτων που όμως βρίσκονται μαζί. Αυτά τα ενδιάμεσα στρώματα ονομάζονται *κρυφά*, επειδή οι υπολογισμοί που εκτελούνται σε αυτά δεν είναι ορατοί στον χρήστη. Τα *νευρωνικά δίκτυα πολλαπλών στρωμάτων* που ακολουθούν τη συγκεκριμένη αρχιτεκτονική ονομάζονται *Δίκτυα Πρόσθιας Τροφοδότησης* (**Feed Forward Neural Networks**), επειδή τα διαδοχικά στρώματα τροφοδοτούν το ένα το άλλο με κατεύθυνση προς τα εμπρός, από την είσοδο στην έξοδο. Η αρχιτεκτονική αυτών των δικτύων συνήθως προϋποθέτει ότι κάθε κόμβος ενός στρώματος συνδέεται με όλους τους κόμβους του επόμενου **(Fully Dense Network)**.

Το κύριο πλεονέκτημα των *νευρωνικών δικτύων πολλαπλών επιπέδων* είναι ότι παρέχουν τη δυνατότητα υπολογισμού πολύπλοκων μη γραμμικών συναρτήσεων κάτι που δεν είναι εύκολο με άλλες μεθόδους πρόβλεψης (π.χ. μέσω της *γραμμικής παλινδρόμησης*). Πιο συγκεκριμένα, στους νευρώνες κάθε διαφορετικού στρώματος ενός τέτοιου νευρωνικού δικτύου μπορούμε να εφαρμόσουμε εναλλακτικές *συναρτήσεις ενεργοποίησης* (π.χ. *Tanh, ReLU, Sigmoid* κ.λπ.) οι οποίες επιτρέπουν τον διαχωρισμό των δεδομένων με μη γραμμικό τρόπο. Ένα τέτοιο νευρωνικό δίκτυο τριών στρωμάτων και μ' ένα επιπλέον κρυφό στρώμα απεικονίζεται στο Σχήμα 6.6 βάσει των δεδομένων του Παραδείγματος 6.1. Το νευρωνικό δίκτυο του σχήματος 6.6 υλοποιεί τον *αλγόριθμο αποσύνθεσης πίνακα κατά UV*



Σχήμα 6.6: Παράδειγμα ενός *Multi-Layer Perceptron* για την υλοποίηση του *UV-decomposition* βάσει των δεδομένων του Παραδείγματος 6.1.

(*UV-decomposition*), τον οποίο περιγράψαμε αναλυτικά στο προηγούμενο κεφάλαιο. Συνοπτικά, η *αποσύνθεση κατά UV* εκφράζει τον αρχικό πίνακα βαθμολογίας χρήστη-στοιχείου A του τρέχοντος παραδείγματός μας ως το γινόμενο δύο άλλων πινάκων U και V όπως φαίνεται παρακάτω:

$$A \approx U \cdot V^{\top} \tag{6.1}$$

Γενικότερα, εάν έχουμε έναν πίνακα A με n γραμμές και m στήλες, μπορούμε να υπολογίσουμε δύο πίνακες: τον πίνακα U με n γραμμές και k στήλες, και τον πίνακα V με m γραμμές και k στήλες, έτσι ώστε το γινόμενο των δύο πινάκων $U \cdot V^\top$ να υπολογίζει κατά προσέγγιση τον αρχικό πίνακα A. Τονίζουμε ότι η μεταβλητή k ελέγχει τον **λανθάνοντα διανυσματικό χώρο (latent vector space)** και το μέγεθος των πινάκων U και V.

Στο νευρωνικό δίκτυο τριών στρωμάτων του Σχήματος 6.6 ο πίνακας U_{4x2} εκφράζει τα λανθάνοντα διανύσματα των χρηστών, και αντιπροσωπεύεται από τις ακμές με βάρη που συνδέουν τους νευρώνες του πρώτου στρώματος (εισόδου) με το κρυφό στρώμα. Ο δε πίνακας V^T_{2x4} εκφράζει τα **λανθάνοντα (latent)** διανύσματα των στοιχείων, και αντιπροσωπεύεται από τις κατευθυνόμενες ακμές με βάρη που συνδέουν τους νευρώνες του κρυφού στρώματος με το στρώμα εξόδου, όπως φαίνεται στο Σχήμα 6.6. Εφεξής, οι πίνακες U και V^T θα ονομάζονται *πίνακες διανυσμάτων βαρών* W_1 και W_2 αντίστοιχα όπως φαίνεται στο Σχήμα 6.6, προκειμένου να διατηρήσουμε την ορολογία των νευρωνικών δικτύων. Τέλος, παρατηρήστε ότι οι νευρώνες του κρυφού στρώματος στο Σχήμα 6.6 είναι δύο, που σημαίνει ότι δημιουργούμε έναν δισδιάστατο διανυσματικό χώρο (k=2) διατηρώντας μόνο τις δύο πρωτεύουσες διαστάσεις από τους U και V αντίστοιχα.

Στο νευρωνικό δίκτυο του Σχήματος 6.6 εισάγουμε το διάνυσμα εισόδου X, το οποίο είναι *one-hot encoded*, που σημαίνει ότι το σήμα εισόδου ισούται με τη μονάδα (1) μόνο για τον υπό εξέταση χρήστη, ενώ για τους υπόλοιπους είναι μηδέν (0). Επομένως, για τον χρήστη U_4 το διάνυσμα εισόδου X είναι {0,0,0,1}, όπως φαίνεται στο αριστερό μέρος του Σχήματος 6.6. Έπειτα, το *Multi-layer Perceptron (MLP)* του Σχήματος 6.6 προβάλλει το διάνυσμα εισόδου X του χρήστη U_4 στον δισδιάστατο διανυσματικό χώρο του κρυφού στρώματος. Πιο συγκεκριμένα, δοθέντος του αρχικού διανύσματος εισόδου X, το κρυφό στρώμα του *MLP* το αντιστοιχίζει με μια κρυφή αναπαράσταση $Z_1 \in \mathbb{R}^K$ μέσω της ακόλουθης *συνάρτησης ενεργοποίησης*:

$$\boldsymbol{Z}_1 = f\left(\boldsymbol{W}_1^T * X + \boldsymbol{b}_1\right) \tag{6.2}$$

όπου $\boldsymbol{W_1} \in \mathbb{R}^{n \times k}$ ένας πίνακας βαρών και $\boldsymbol{b_1} \in \mathbb{K}$ το *bias* (μεροληψία).

Σ' αυτό το σημείο σημειώνουμε ότι έχοντας ως δεδομένο ένα *MLP* με l στρώματα, για κάθε στρώμα του *MLP* το διάνυσμα εισόδου αναπαριστάται σ' ένα άλλο διάνυσμα του λανθάνοντος χώρου του επόμενου στρώματος. Η κατόπιν προκύπτουσα *λανθάνουσα διανυσματική αναπαράσταση* (*latent vector representation*) του επόμενου στρώματος απεικονίζεται ως εξής:

$$\begin{aligned} \boldsymbol{Z}_1 &= f\left(\boldsymbol{W}_1^T * X + \boldsymbol{b}_1\right) \\ \boldsymbol{Z}_2 &= f\left(\boldsymbol{W}_2^T * Z_1 + \boldsymbol{b}_2\right) \\ &\cdots \\ \boldsymbol{Z}_l &= f\left(\boldsymbol{W}_l^T * \boldsymbol{Z}_{l-1} + \boldsymbol{b}_l\right) \end{aligned} \tag{6.3}$$

Στο *MLP* του Σχήματος 6.6 έχουμε μόνο ένα κρυφό στρώμα και συνεπώς υπολογίζουμε δύο μόνο λανθάνοντα διανύσματα (το $\boldsymbol{Z}_1$ που αφορά στον χρήστη και το $\boldsymbol{Z}_2$ που αφορά στο στοιχείο):

$$\begin{aligned} \boldsymbol{Z}_1 &= f\left(\boldsymbol{W}_1^T * X + \boldsymbol{b}_1\right) \\ \boldsymbol{Z}_2 &= f\left(\boldsymbol{W}_2^T * Z_1 + \boldsymbol{b}_2\right) \end{aligned} \tag{6.4}$$

Για την αντιμετώπιση του προβλήματος της πρόβλεψης της βαθμολογίας ενός χρήστη για ένα στοιχείο χρησιμοποιούμε μια *αντικειμενική συνάρτηση* για τη συμπλήρωση των ελλειπουσών τιμών του αντίστοιχου πίνακα ως εξής:

$$\min_{\mathbf{u},\mathbf{v}} \sum_{y_{ij}\in\mathcal{R}} \left(y_{ij} - \mathbf{u}_i \cdot \mathbf{v}_j^T\right)^2 + \lambda_u \cdot \|\mathbf{u}_i\|^2 + \lambda_v \cdot \|\mathbf{v}_i\|^2 \tag{6.5}$$

όπου $\mathcal{R}$ το σύνολο των παρατηρούμενων βαθμολογιών, y_{ij} η βαθμολογία του χρήστη i για το στοιχείο j, $\mathbf{u}_i$ το λανθάνον διάνυσμα του χρήστη i και $\mathbf{v}_j$ το λανθάνον διάνυσμα του στοιχείου j, ενώ οι παράμετροι λ_u και λ_v "κανονικοποιούν" τη L^2 *νόρμα* των $\mathbf{u}$ και $\mathbf{v}$.

Τέλος, η έξοδος του νευρωνικού δικτύου είναι η πρόβλεψη των βαθμολογιών $\hat{y}$ ενός χρήστη για τα στοιχεία. Ακολούθως, συγκρίνουμε τις βαθμολογίες που προέβλεψε το νευρωνικό δίκτυο με τις πραγματικές βαθμολογίες του χρήστη U_4, οι οποίες βρίσκονται στην 4η γραμμή του πίνακα βαθμολογιών χρηστών-στοιχείων A όπως αυτός φαίνεται στο Σχήμα 6.2. Συμπερασματικά, οι προβλέψεις των βαθμολογιών που προκύπτουν από το νευρωνικό δίκτυο χρησιμοποιούνται για τη δημιουργία ενός νέου πίνακα $\hat{A} = U \cdot V^\top$ που κρατά τις κατά προσέγγιση βαθμολογίες χρηστών-στοιχείων, ο οποίος συγκρίνεται με τον αρχικό πίνακα A, και εφόσον υπάρχει διαφορά, τότε διορθώνουμε αναλόγως τα διανύσματα βαρών $\boldsymbol{W_1}$ και $\boldsymbol{W_2}$. Αυτή η προσέγγιση επαναλαμβάνεται μέχρι τη σύγκλιση των τιμών του πίνακα $\hat{A} = U \cdot V^\top$ σε σημείο που να μη μεταβάλλονται πια. Επομένως, το νευρωνικό δίκτυο διαμορφώνει ανάλογα τα διανύσματα βαρών $\boldsymbol{W_1}$ και $\boldsymbol{W_2}$ έτσι ώστε, παίρνοντας το εσωτερικό γινόμενό τους, να μπορεί να προβλέψει τις βαθμολογίες των χρηστών για τα στοιχεία. Για παράδειγμα (βλ. το κέντρο του Σχήματος 6.6), υπολογίζουμε τη βαθμολογία του χρήστη U_4 για το στοιχείο I_4 με τον ακόλουθο τρόπο:

$$\hat{y}_{4,4} = w_{4,1} \cdot w_{1,4} + w_{4,2} \cdot w_{2,4} = 1.68 \cdot 0.3 + (-0.52) \cdot 2.33 = -0.71 \tag{6.6}$$

Συνεπώς, προβλέπουμε ότι στον χρήστη U_4 δεν αρέσει το στοιχείο I_4, κάτι που είναι λογικό, διότι ο χρήστης U_4 έχει παρόμοιες βαθμολογίες με τον χρήστη U_2 όπως φαίνεται στο Σχήμα 6.2.

Δεδομένου, τώρα, ότι το νευρωνικό δίκτυο του Σχήματος 6.6 αποτελείται από ένα κρυφό στρώμα δύο νευρώνων, έχει τη δυνατότητα γι' αυτόν τον λόγο να διαχωρίσει τα δεδομένα του προβλήματός μας με τη βοήθεια μιας ανοιχτής, *κυρτής περιοχής λύσεων*. Το κρυφό λοιπόν στρώμα των δύο νευρώνων μπορεί να διαχωρίσει το καρτεσιανό επίπεδο αξόνων x-y με δύο τεμνόμενες ευθείες που οριοθετούν την προαναφερθείσα περιοχή. Εάν όμως υπήρχε και τρίτος νευρώνας στο κρυφό στρώμα, τότε θα υπήρχε και τρίτη ευθεία η οποία θα οριοθετούσε το καρτεσιανό επίπεδο αξόνων x-y με μεγαλύ-

τερη σαφήνεια, ορίζοντας μια κλειστή, *κυρτή (τριγωνική) περιοχή λύσεων*. Συνεπώς, πιο πολύπλοκα δίκτυα νευρώνων, τα οποία οργανώνονται σε πολλά στρώματα, είναι ικανά να επιλύσουν σύνθετα μη γραμμικώς διαχωρίσιμα προβλήματα.

6.2 Graph Convolutional Networks

Ένα **Συν-εξελικτικό Νευρωνικό Δίκτυο (Convolutional Neural Network - CNN)** είναι ένα perceptron πολλαπλών στρωμάτων που έχει σχεδιαστεί ειδικά για να αναγνωρίζει τα αντικείμενα από δισδιάστατες εικόνες. Το μοτίβο συνδεσιμότητας μεταξύ των νευρώνων ενός *CNN* είναι εμπνευσμένο από την οργάνωση του οπτικού φλοιού των οργανισμών, ο οποίος διαθέτει μια μικρή περιοχή κυττάρων που είναι ευαίσθητα μόνο σε συγκεκριμένες περιοχές του οπτικού πεδίου. Κατ' αναλογία, σ' ένα τεχνητό *συν-εξελικτικό στρώμα ενός CNN* κάθε νευρώνας λαμβάνει είσοδο μόνο από μια περιορισμένη περιοχή του προηγούμενου στρώματος, το οποίο και ονομάζεται **πεδίο υποδοχής του νευρώνα (the neuron's receptive field)**. Η ύπαρξη λοιπόν αυτού του πεδίου επιτρέπει την εξαγωγή τοπικών χαρακτηριστικών από την εικόνα. Μόλις όμως εξαχθεί ένα χαρακτηριστικό, η ακριβής θέση του έχει πια μικρότερη σημασία εφόσον διατηρείται κατά προσέγγιση η θέση του σε σχέση μ' άλλα χαρακτηριστικά. Τα βασικά υπολογιστικά στρώματα ενός *CNN* είναι δύο:

- Το **στρώμα συν-εξέλιξης (convolve layer)** χρησιμοποιεί ένα σύνολο από φίλτρα που εντοπίζουν την παρουσία συγκεκριμένων χαρακτηριστικών ή μοτίβων που υπάρχουν στην αυθεντική εικόνα που δίνεται στην είσοδο δημιουργώντας πολλαπλούς **δισδιάστατους χάρτες χαρακτηριστικών (2-dimensional feature maps)**, εντός των οποίων οι επιμέρους νευρώνες περιορίζονται στο να μοιράζονται παρόμοια διανύσματα βαρών.

- Κάθε *στρώμα συν-εξέλιξης* ακολουθείται από ένα **στρώμα δειγματοληψίας υποσυνόλου (sub-sampling or pooling layer)** με την οποία μειώνεται η ανάλυση του κάθε χάρτη χαρακτηριστικών. Ο λόγος ύπαρξης αυτού του επιπέδου είναι η σταδιακή μείωση του μεγέθους των διαστάσεων καθώς η είσοδος είναι μια "μεγάλη εικόνα", ενώ η έξοδος αποτελείται από μερικές μόνο κλάσεις εικονιδίων πολύ μικρότερης διαστατικότητας.

Το *CNN* έχει εφαρμοστεί με επιτυχία σε διάφορους τομείς επεξεργασίας και ανάλυσης δεδομένων, όπως η **Επεξεργασία Εικόνας (Image Processing)** και η **Επεξεργασία Φυσικής Γλώσσας (Natural Language Processing)**, επιτυγχάνοντας σημαντικές βελτιώσεις όσον αφορά στην παραγωγή αποτελεσματικών και σωστών προβλέψεων (π.χ. αναγνώριση εικόνας και προσώπου, αναγνώριση φωνής και ομιλίας κ.λπ.). Η μεγάλη αποτελεσματικότητα των *CNN* ιδίως στην επεξεργασία και αναγνώριση εικόνας οδήγησαν στην εφαρμογή τους και στην **Αναλυτική Γραφοδεδομένων (Graph Data Analytics)**. Αναπτύχθηκαν έτσι τα **Συν-εξελικτικά Δίκτυα Γράφων (Graph Convolutional Network - GCN)**, τα οποία αφορούν στην αρχιτεκτονική ενός νευρωνικού δικτύου που χρησιμοποιεί τον **πίνακα γειτνίασης (adjacency matrix)** των κόμβων ενός γράφου και τα

χαρακτηριστικά των κόμβων (node features) του, και εφαρμόζει τεχνικές παρακολούθησης της παράλληλης εξέλιξης των συνδέσμων και των χαρακτηριστικών των κόμβων του γράφου. Καθώς τα περισσότερα συστήματα συστάσεων "μοντελοποιούνται" ως διμερείς γράφοι, το νευρωνικό δίκτυο τύπου *GCN* έχει προσελκύσει το επιστημονικό ενδιαφέρον της κοινότητας των ευφυών συστημάτων συστάσεων. Στις επόμενες παραγράφους παρουσιάζονται συνοπτικά οι μέθοδοι συστάσεων με βάση το μοντέλο *GCN*.

Μία από τις σημαντικές μελέτες στα συστήματα συστάσεων βάσει του μοντέλου *GCN* είναι το ακόλουθο ερευνητικό άρθρο, [Ying et al., 2018], που περιγράφει τον *αλγόριθμο PinSage*, ο οποίος προτείνει στοιχεία σε χρήστες για μια διαδικτυακή υπηρεσία πολύ υψηλής κλίμακας δεδομένων (που σημαίνει εκατοντάδες εκατομμύρια χρηστών και στοιχείων), το γνωστό *Pinterest*. Το προτεινόμενο μοντέλο βασίζεται σε σύντομους, **τυχαίους** (δηλ. μη προκαθορισμένους) **"περιπάτους"** (*random walks*) στους κόμβους του διμερούς γράφου χρηστών και στοιχείων. Η συγκέντρωση των τελευταίων δημιουργεί λανθάνουσες αναπαραστάσεις των κόμβων ή αλλιώς **ενσωματώσεις κόμβων (node embeddings)** του γράφου σ' έναν μικρότερο (πιο περιορισμένο) διανυσματικό χώρο που ως εκ τούτου είναι και αντιπροσωπευτικότερος του κάθε υπό εξέταση κόμβου. Με τον όρο *ενσωμάτωση ενός κόμβου* εννοούμε την αναπαράστασή του σ' έναν μικρότερο διανυσματικό χώρο, γεγονός που επιτυγχάνεται εστιάζοντας στα κύρια χαρακτηριστικά του υπό εξέταση κόμβου και αφαιρώντας τυχόν "θορυβώδη" δεδομένα. Η εκπαίδευση του *νευρωνικού δικτύου GCN* του *αλγορίθμου PinSage* καθοδηγείται από μια **συνάρτηση απώλειας** (δηλ. μείωσης του σφάλματος, άρα της απώλειας, μεταξύ της πραγματικής τιμής και αυτής που προβλέπει ο αλγόριθμος) **για τη βελτιστοποίηση κατάταξης (ranking loss function)**, ώστε τα συναφή στοιχεία να τοποθετούνται όσο το δυνατόν πιο κοντά στον **χώρο ενσωμάτωσης (embedding space)**, και οι κόμβοι να αναπαριστώνται τόσο ως προς τη συνδεσμολογία τους με τους άλλους κόμβους όσο και ως προς τα χαρακτηριστικά που αυτοί έχουν. Για την αντιμετώπιση των ζητημάτων υψηλής κλιμάκωσης των δεδομένων χρησιμοποιούνται οι τεχνολογίες *Map Reduce* και *locality sensitive hashing* δεδομένου ότι το *Pinterest* είναι μια υπηρεσία που διαχειρίζεται καθημερινά εκατοντάδες εκατομμύρια χρήστες και στοιχεία (συνήθως φωτογραφίες).

Το πρόβλημα της σύστασης στοιχείων σε γράφους μεγάλης κλίμακας αντιμετωπίζεται και στην ακόλουθη ερευνητική εργασία, [Chen et al., 2020]. Οι συγγραφείς της λοιπόν αναθεώρησαν τη βασική αρχιτεκτονική του *GCN* αφαιρώντας τις *μη γραμμικές συναρτήσεις ενεργοποίησης* και προσθέτοντας περισσότερους συνδέσμους μεταξύ των *νευρώνων διαφορετικών στρωμάτων* [He et al., 2016]. Το κίνητρό τους ήταν η ανάγκη για μείωση της "αραιότητας" των δεδομένων στους μεγάλους γράφους στοιχείων-χρηστών και το γεγονός ότι τα μοντέλα που βασίζονται στην αρχιτεκτονική τύπου *GCN* δεν μπορούν εύκολα να μεγαλώσουν σε βάθος λόγω του **φαινομένου της υπερομαλοποίησης (oversmoothing effect)**.

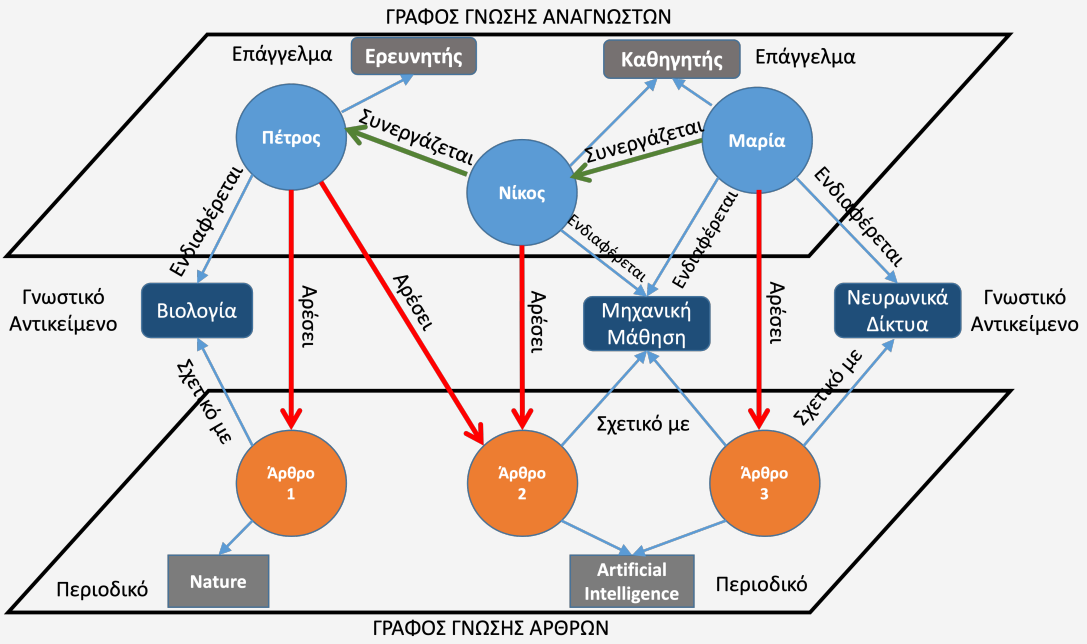
Το πρόβλημα της "αραιότητας" των δεδομένων σε γράφους χρηστών-στοιχείων αντιμετωπίζεται επίσης στην παρακάτω ερευνητική εργασία, [Feng et al., 2019]. Η εκεί προτεινόμενη μέθοδος

εμπλουτίζει τα δεδομένα βαθμολογιών των στοιχείων από τους χρήστες με επιπρόσθετες πηγές δεδομένων (π.χ. δημογραφικά στοιχεία των χρηστών ή χαρακτηριστικά των στοιχείων), οι οποίες ενσωματώνονται και οι δύο στον ίδιο διανυσματικό χώρο. Εν συνέχεια υπόκεινται σε επεξεργασία μέσω των *πολυωνυμικών φίλτρων του* **Chebyshev** *για γράφους* (*polyonymous Chebyshev graph filters*) για την εξαγωγή των βασικών χαρακτηριστικών χρηστών και στοιχείων. Τέλος, οι συγγραφείς εφαρμόζουν έναν *μηχανισμό εντοπισμού της εστίασης* **(attention mechanism)** των χρηστών, έτσι ώστε το *GCN* να δίνει έμφαση στους σημαντικούς γείτονες του υπό εξέταση χρήστη.

6.2.1 Αναπαράσταση Κόμβων μέσω ενός Νευρωνικού Δικτύου τύπου GCN

Στο σημείο αυτό θα περιγράψουμε τη βασική αρχιτεκτονική ενός νευρωνικού δικτύου τύπου *GCN* με τη βοήθεια ενός παραδείγματος.

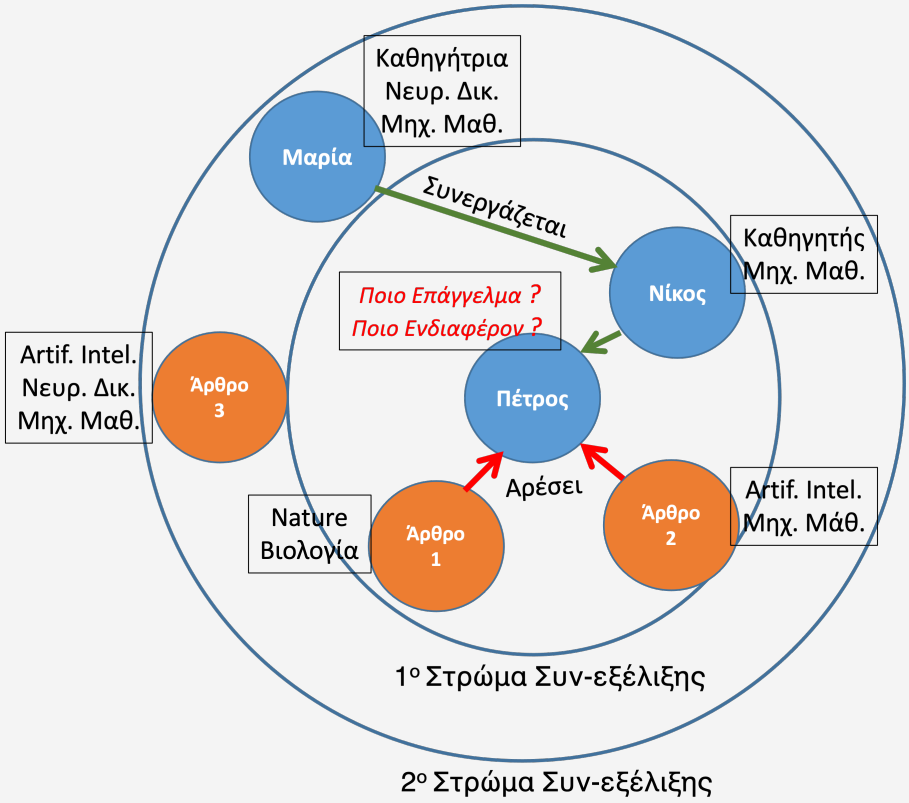
Παράδειγμα 6.4 Έστω ότι είμαστε οι διαχειριστές ενός ηλεκτρονικού εκδοτικού οίκου που θέλει να προσφέρει εξατομικευμένο περιεχόμενο στους αναγνώστες του βάσει των ερευνητικών προτιμήσεων τους και των συνεργασιών τους με άλλους αναγνώστες. Αρχικά το σύστημα συστάσεων μπορεί να ζητήσει από τους αναγνώστες ορισμένες πληροφορίες για κάποια ευρύτερα δημογραφικά τους χαρακτηριστικά (π.χ. επάγγελμα, ερευνητικά ενδιαφέροντα κ.λπ.) που μπορεί να παίξουν σημαντικό ρόλο στην επιλογή των προτεινόμενων άρθρων. Παραδείγματος χάρη, ένας αναγνώστης από το ακαδημαϊκό περιβάλλον τείνει να επιλέγει επιστημονικά άρθρα σχετικά με τα ατομικά ερευνητικά του ενδιαφέροντα, ενώ αναγνώστες που εργάζονται σε ερευνητικά κέντρα τείνουν να επιλέγουν άρθρα σχετικά με τα αντικείμενα που εξετάζονται εκεί. Ακολούθως, όπως ήδη αναφέραμε, το σύστημα συστάσεων καταγράφει τη διάδραση των αναγνωστών με τα άρθρα καθώς και τις ερευνητικές ή άλλες συνεργασίες μεταξύ των αναγνωστών ώστε να διαμορφώσει το προφίλ τους.
Επομένως, βάσει της εκφώνησης του Παραδείγματος 6.4 κατασκευάζουμε έναν διμερή γράφο χρηστών-στοιχείων (βλ. Σχήμα 6.7) όπου τα βασικά χαρακτηριστικά ενός αναγνώστη είναι το επάγγελμα και τα ερευνητικά του ενδιαφέροντα. Στο ίδιο Σχήμα (6.7), οι κόκκινες κατευθυνόμενες ακμές που ενώνουν τους αναγνώστες με τα άρθρα αποτυπώνουν το γεγονός ότι σ' έναν αναγνώστη αρέσει ένα συγκεκριμένο ερευνητικό άρθρο. Τέλος, στο Σχήμα 6.7 αποτυπώνονται με τη χρήση των πράσινων ακμών οι προαναφερθείσες συνεργασίες του κάθε αναγνώστη με τους άλλους. Για έναν νέο αναγνώστη, για τον οποίο εκ των πραγμάτων δεν υπάρχει ακόμη διάδραση με πολλά άρθρα, υπάρχει το πρόβλημα της **ψυχρής εκκίνησης (cold start)** του συστήματος συστάσεων, δεδομένου ότι αυτό χρειάζεται έναν ελάχιστο αριθμό αλληλεπιδράσεων του αναγνώστη με τα άρθρα προκειμένου οι συστάσεις του να είναι εύστοχες και ακριβείς. Σε μια τέτοια δυσχερή περίπτωση, το σύστημα συστάσεων θα μπορούσε να αναπαραστήσει το προφίλ ενός αναγνώστη με τη χρήση των προφίλ άλλων χρηστών ή άρθρων που έχουν *παρόμοια χαρακτηριστικά* (*node features*) μ' αυτόν.



Σχήμα 6.7: Γράφος Γνώσης Αναγνωστών και Άρθρων.

Έστω ότι μας ζητείται να προβλέψουμε βάσει των δεδομένων του Σχήματος 6.7 τα χαρακτηριστικά (επάγγελμα και ερευνητικά ενδιαφέροντα) του αναγνώστη Πέτρου, βασιζόμενοι στα άρθρα που του αρέσουν και τις συνεργασίες του μ' άλλους αναγνώστες. Συνεπώς, σχηματίζουμε έναν νέο γράφο, με σκοπό να προβλέψουμε τα χαρακτηριστικά του αναγνώστη Πέτρου, ο οποίος βρίσκεται στο κέντρο των *ομόκεντρων κύκλων/στρωμάτων συν-εξέλιξης* ενός νευρωνικού δικτύου τύπου *GCN* όπως φαίνεται στο Σχήμα 6.8. Ο γράφος είναι και αυτός διμερής, δηλαδή αποτελείται από δύο τύπους κόμβων (αναγνώστες και άρθρα). Όπως παρατηρούμε στο Σχήμα (6.8), τα δύο χαρακτηριστικά (επάγγελμα και ερευνητικά ενδιαφέροντα) του κάθε αναγνώστη και τα δύο χαρακτηριστικά (τίτλος περιοδικού και γνωστικό αντικείμενο) του κάθε άρθρου εμφανίζονται ως **χαρακτηριστικά των κόμβων (node features)** του γράφου. Όπως φαίνεται στο ίδιο Σχήμα (6.8), τα προτιμώμενα άρθρα και οι συνεργασίες του Πέτρου αναπαριστώνται με τη χρήση κατευθυνόμενων ακμών.

Υπάρχουν εδώ δύο διαφορετικοί τύποι ακμών: Ο πρώτος τύπος αφορά στις συνεργασίες του (με πράσινο χρώμα), ενώ ο δεύτερος αφορά στις προτιμήσεις (likes) του για συγκεκριμένα άρθρα (με κόκκινο χρώμα). Παρατηρήστε ότι όλες οι ακμές έχουν κατεύθυνση προς τον υπό εξέταση κόμβο (τον Πέτρο), διότι γι' αυτόν τον κόμβο επιθυμούμε να υπολογίσουμε τη λανθάνουσα αναπαράστασή του.



Σχήμα 6.8: *Ομόκεντρα στρώματα συν-εξέλιξης* του νευρωνικού δικτύου τύπου *GCN* για τη δημιουργία της λανθάνουσας αναπαράστασης του κόμβου "Πέτρος" βάσει των δεδομένων του Παραδείγματος 6.4.

Η βασική λειτουργία ενός *Graph Convolutional Network (GCN)* είναι να συγκεντρώνει πληροφορίες από τους κόμβους που γειτνιάζουν με τον εκάστοτε υπό εξέταση κόμβο και να τις προωθεί στο επόμενο στρώμα συν-εξέλιξης. Για να προβλέψουμε επομένως το λανθάνον διάνυσμα του αναγνώστη που ονομάζεται Πέτρος, εφαρμόζουμε ένα νευρωνικό δίκτυο τύπου *GCN* στον γράφο του Σχήματος 6.8 έτσι ώστε τα χαρακτηριστικά (ερευνητικά ενδιαφέροντα, επάγγελμα κ.λπ.) των γειτονικών κόμβων του υπό εξέταση κόμβου να διαδίδονται και να αθροίζονται σωρευτικά από στρώμα σε στρώμα, για να αναπαρασταθούν οι δικές του προτιμήσεις βάσει του ακόλουθου κανόνα συν-εξέλιξης:

$$h_i^{(l+1)} = \sigma\left(\sum_{r\in R}\sum_{j\in\mathcal{N}_i^r}\frac{1}{c_{i,r}}\cdot W_r^{(l)}\cdot h_j^{(l)}\right) \tag{6.7}$$

όπου $h_i^{(l+1)}$ η αναπαράσταση του υπό εξέταση κόμβου i, $h_j^{(l)}$ η λανθάνουσα αναπαράσταση του γειτονικού κόμβου j στο l-στρώμα του συν-εξελικτικού νευρωνικού δικτύου, $W_r^{(l)}$ ο πίνακας βαρών στο l-στρώμα σε σχέση με τις κατευθυνόμενες ακμές τύπου r, $\mathcal{N}_i^r$ το σύνολο των γειτονικών κόμβων του υπό εξέταση κόμβου i που συνδέονται μαζί του με τις κατευθυ-

νόμενες ακμές τύπου $r \in \mathcal{R}$, και $c_{i,r} = |\mathcal{N}_i^r|$ ο αριθμός των γειτόνων του κόμβου i ως προς τις κατευθυνόμενες ακμές τύπου r. Υπενθυμίζουμε ότι υπάρχουν δύο τύποι κατευθυνόμενων ακμών r στο παράδειγμά μας: οι ακμές συνεργασιών με άλλους αναγνώστες και αυτές που δηλώνουν την αρέσκεια του Πέτρου για ένα άρθρο. Τέλος, σ είναι μια *μη γραμμική συνάρτηση ενεργοποίησης* (π.χ. *σιγμοειδής συνάρτηση*).

Συνεπώς, η αναπαράσταση των προτιμήσεων του Πέτρου μέσω ενός νευρωνικού δικτύου *GCN* μπορεί να προκύψει από τη σωρευτική άθροιση πληροφοριών των γειτονικών κόμβων του με βάση είτε (α) τα δεδομένα αλληλεπίδρασής του με τα άρθρα είτε (β) τις συνεργασίες του μ' άλλους αναγνώστες. Για παράδειγμα, το λανθάνον διάνυσμα του υπό εξέταση αναγνώστη i βάσει της διάδρασής του με τα άρθρα έχει ως εξής:

$$\boldsymbol{h}_i = \sigma\left(\mathbf{W} \cdot \text{AGGREGATE}\left(\left\{\boldsymbol{q}_j, \forall j \in C(i)\right\}\right)\right) \tag{6.8}$$

όπου $C(i)$ το σύνολο των άρθρων που αρέσουν στον αναγνώστη i, και $\boldsymbol{q}_j$ το *διάνυσμα ενσωμάτωσης* (*item embedding*) ενός άρθρου j με το οποίο αλληλεπίδρασε ο αναγνώστης i. Η AGGREGATE είναι μια *γενικευμένη συνάρτηση σωρευτικής άθροισης των χαρακτηριστικών των γειτονικών κόμβων* (άρθρων ή αναγνωστών) του υπό εξέταση κόμβου i. Συνήθως για τη *συνάρτηση AGGREGATE* χρησιμοποιείται ο *τελεστής μέσου όρου*, ο οποίος αποτυπώνεται στην παρακάτω Εξίσωση:

$$\boldsymbol{h}_i = \sigma\left(\mathbf{W} \cdot \left\{\sum_{j \in C(i)} \frac{1}{C(i)} \cdot \boldsymbol{q}_j\right\}\right) \tag{6.9}$$

Εναλλακτικά, αντί του *τελεστή μέσου όρου* θα μπορούσαμε να χρησιμοποιήσουμε τον *τελεστή συνένωσης CONCATENATION*, ο οποίος απλά συνενώνει τα διανύσματα των κόμβων που αθροίζονται μαζί, όπως φαίνεται ακολούθως:

$$\boldsymbol{h}_i = \sigma\left(\mathbf{W} \cdot \left\{\sum_{j \in C(i)} \left[\, \boldsymbol{h}_i \parallel \boldsymbol{q}_j \,\right]\right\}\right) \tag{6.10}$$

όπου $\parallel$ ο τελεστής συνένωσης των δύο κάθε φορά διανυσμάτων που εκφράζουν τα χαρακτηριστικά των κόμβων που συνενώνονται.

Βασιζόμενοι στα δεδομένα του Σχήματος 6.8, έστω ότι θα χρησιμοποιήσουμε τον *τελεστή συνένωσης CONCATENATION*. Συνεπώς, δεδομένων των χαρακτηριστικών των κόμβων *Άρθρο 1* και *Άρθρο 2*, θα κάναμε την πρόβλεψη σχετικά με τα ερευνητικά ενδιαφέροντα του Πέτρου ότι αυτά θα αφορούν είτε στη *Βιολογία* είτε στη *Μηχανική Μάθηση*. Τονίζουμε ότι η παραπάνω διαδικασία σωρευτικής άθροισης θα πρέπει να ακολουθηθεί και για τις συνεργασίες του Πέτρου μ' άλλους αναγνώστες, οι οποίοι ανήκουν (και οι δύο) στον ακαδημαϊκό χώρο.

Επομένως, βάσει των προαναφερθεισών συνεργασιών του, αναμένουμε το επάγγελμα του Πέτρου να είναι Καθηγητής Πανεπιστημίου (βλ. Σχήμα 6.8).

Συνοψίζοντας, σ' αυτήν την ενότητα εξετάσαμε το πώς μπορούμε να αναπαραστήσουμε έναν κόμβο ενός γράφου σε μια *χαμηλής διαστατικότητας διανυσματική αναπαράσταση* (*node embed-ding*). Συγκεκριμένα, προβλέψαμε τη διανυσματική αναπαράσταση του αναγνώστη Πέτρου όπως φαίνεται στο Σχήμα 6.8. Σημειώστε ότι, ακολουθώντας μια παρόμοια διαδικασία νευρωνικού δικτύου τύπου *GCN*, μπορούμε να προβλέψουμε οποιαδήποτε άλλη αναπαράσταση κόμβου (είτε κάποιου αναγνώστη είτε κάποιου άρθρου).

6.2.2 Μηχανισμός Εστίασης σε Γράφο (Graph Attention Mechanism)

Στην παράγραφο αυτή θα επαναπροσδιορίσουμε τον κανόνα διάδοσης ενός *GCN* της Εξίσωσης 6.7 με την εφαρμογή ενός *μηχανισμού εστίασης*, ώστε μέσω αυτού του σταθερού συντελεστή -εδώ $c_{i,r}$- ο οποίος αντιπροσωπεύει τον αριθμό των γειτόνων του υπό εξέταση κόμβου i για τις ακμές τύπου r, να μην αποδίδουμε σε όλους τους κόμβους την ίδια σημασία.

Η βασική ιδέα πίσω από τον **Μηχανισμό Εστίασης σε Γράφο (Graph Attention Mechanism)** είναι να υπολογίζει δυναμικά για κάθε κόμβο τον συντελεστή $c_{i,r}$ της Εξίσωσης 6.7 και όχι απλά (να υπολογίζει) μια σταθερή τιμή του για όλους τους κόμβους. Άρα, μπορούμε να χρησιμοποιήσουμε τα χαρακτηριστικά των κόμβων για να καθορίσουμε τη "βαρύτητα" των πρώτων ως προς τον υπό εξέταση κόμβο. Ως εκ τούτου, κάποιοι γειτονικοί κόμβοι ενδεχομένως να είναι περισσότερο σημαντικοί από κάποιους άλλους για τον προσδιορισμό της αναπαράστασης του υπό εξέταση κόμβου, επειδή έχουν περισσότερα *κοινά χαρακτηριστικά* (*node features*) μ' αυτόν.

Ο νέος δυναμικός συντελεστής, λοιπόν, εφεξής θα συμβολίζεται ως a_{ij}, και θα υπολογίζεται με βάση τα κοινά χαρακτηριστικά ενός γειτονικού κόμβου j με τον υπό εξέταση κόμβο i, τα οποία στη συνέχεια θα περνούν σε μια *συνάρτηση εστίασης* (*attention function*) που θα υλοποιείται από ένα *νευρωνικό δίκτυο ενός επιπέδου* (*single-layer perceptron*) ως ακολούθως:

$$a_{ij} = \text{attention}\left(h_i, h_j\right)$$

Τώρα, για έναν υπό εξέταση κόμβο i, και προκειμένου να καταστήσουμε τους *συντελεστές προσοχής* a_{ij} εύκολα συγκρίσιμους μεταξύ των γειτόνων του, τους κανονικοποιούμε ως προς το άθροισμα των συντελεστών όλων των γειτονικών κόμβων j χρησιμοποιώντας τη *συνάρτηση softmax*, ώστε να προκύψει μια κατανομή πιθανοτήτων το άθροισμα των οποίων να είναι η μονάδα (1), ως ακολούθως:

$$a_{ij} = \frac{\exp\left(a_{ij}\right)}{\sum_{k \in \mathcal{N}_i} \exp\left(a_{ik}\right)}$$

Έτσι, ο *κανόνας συν-εξέλιξης και διάδοσης* της Εξίσωσης 6.7 διαμορφώνεται ως εξής:

$$h_i^{(l+1)} = \sigma\left(\sum_{r\in R}\sum_{j\in\mathcal{N}_i^r} a_{ij}\cdot W_r^{(l)}\cdot h_j^{(l)}\right) \tag{6.11}$$

6.2.3 Αναπαράσταση Κόμβου μέσω ενός GCN με τη χρήση Πινάκων Γραμμικής Άλγεβρας

Σ' αυτήν την υποενότητα θα ορίσουμε τον *κανόνα συν-εξέλιξης και διάδοσης της πληροφορίας* της Εξίσωσης 6.7 με τη χρήση πινάκων γραμμικής άλγεβρας. Βάσει λοιπόν του **πίνακα γειτνίασης (adjacency matrix)** $\mathbf{A} \in \{1,0\}^{n\times n}$ των κόμβων και του **πίνακα χαρακτηριστικών (feature matrix)** των κόμβων $\mathbf{H} \in \mathbb{R}^{n\times c}$ μπορούμε να εκφράσουμε τον *κανόνα συν-εξέλιξης της πληροφορίας* της Εξίσωσης 6.7 με τη μορφή ενός γινόμενου πινάκων ως ακολούθως:

$$\boldsymbol{H}^{(l+1)} = \sigma\left(\tilde{\boldsymbol{D}}^{-1}\cdot\tilde{\boldsymbol{A}}\cdot\boldsymbol{H}^{(l)}\cdot\boldsymbol{W}^{(l)}\right)$$

όπου $\tilde{\mathbf{A}} = \mathbf{A} + \boldsymbol{I}$, $\boldsymbol{I}$ ο *ταυτοτικός πίνακας*, $\tilde{D}$ ένας διαγώνιος **πίνακας βαθμού (degree matrix)** με $\tilde{D}_{i,i} = \sum_j \tilde{\mathbf{A}}_{i,j}$, $\mathbf{W} \in \mathbb{R}^{c\times c'}$ ένας πίνακας βαρών, σ μία *μη γραμμική συνάρτηση ενεργοποίησης* και $\mathbf{H} \in \mathbb{R}^{n\times c'}$ η *λανθάνουσα αναπαράσταση των κόμβων του γράφου*. Το σκεπτικό πίσω από την παραπάνω εξίσωση είναι ότι οι αρχικές αναπαραστάσεις $\mathbf{H}$ των κόμβων υπόκεινται σ' έναν γραμμικό μετασχηματισμό μέσα από τον πολλαπλασιασμό τους με τον πίνακα βαρών $\mathbf{W}$, και στη συνέχεια διαδίδονται στους γειτονικούς κόμβους μέσω του *πίνακα διάδοσης* $\tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}}$, ο οποίος είναι γνωστός και ως **πίνακας πιθανοτήτων μετάβασης (transition probability matrix)**.

Επιπρόσθετα, η **φασματική αποσύνθεση (spectral decomposition)** του *Laplacian πίνακα* $\mathbf{L}$ επιτρέπει την κατασκευή ενσωματώσεων κόμβων χαμηλής διαστατικότητας σε γράφους. Ο *πίνακας Laplacian* $\mathbf{L}$ είναι μια αναπαράσταση ενός γραφήματος σε πίνακα, όπως απεικονίζεται παρακάτω:

$$\boldsymbol{L} = \boldsymbol{I} - \boldsymbol{D}^{-\frac{1}{2}}\cdot\boldsymbol{A}\cdot\boldsymbol{D}^{-\frac{1}{2}}$$

Επίσης, μπορούμε να χρησιμοποιήσουμε έναν εναλλακτικό ορισμό του *Laplacian πίνακα* ως ακολούθως:

$$\boldsymbol{L} = \boldsymbol{D}^{-\frac{1}{2}}\cdot(\boldsymbol{D} - \boldsymbol{A})\cdot\boldsymbol{D}^{-\frac{1}{2}}$$

Ο παραπάνω πίνακας L είναι γνωστός ως *κανονικοποιημένος πίνακας Laplacian με θετικό/αρνητικό πρόσημο* (όπου signed= προσημασμένος κατ' άλλη εκδοχή)" (*signed normalized Laplacian matrix*). Συνεπώς, μπορούμε να εκφράσουμε τον *κανόνα συν-εξέλιξης και διάδοσης της πληροφορίας* της Εξίσωσης 6.7 με τη μορφή ενός γινόμενου πινάκων όπως παρακάτω:

$$\boldsymbol{H}^{(l+1)} = \sigma\left(\boldsymbol{L}\cdot\boldsymbol{H}^{(l)}\cdot\boldsymbol{W}^{(l)}\right)$$

όπου $\boldsymbol{L}$ ο *Laplacian πίνακας*, $\boldsymbol{H}$ ο πίνακας χαρακτηριστικών των κόμβων του γράφου, και $\boldsymbol{W}$ ο πίνακας βαρών.

6.2.4 Ορισμός Αντικειμενικής Συνάρτησης για την Πρόβλεψη Βαθμολογιών

Για τις ανάγκες ενός συστήματος συστάσεων η έξοδος του νευρωνικού δικτύου θα πρέπει να είναι η πρόβλεψη των βαθμολογιών $\hat{y}$ ενός αναγνώστη για τα άρθρα. Όπως εξηγήσαμε στην Ενότητα 6.1.2, με τη χρήση ενός *MLP* είναι δυνατόν να θεωρήσουμε την ενσωμάτωση του υπό εξέταση αναγνώστη και του υπό εξέταση άρθρου ως δύο ανεξάρτητες εισόδους και να τις προβλέπουμε μέσω *εσωτερικού γινόμενου* (*dot product*) ή απλής *συνένωσης* (*concatenation*). Μπορούμε επομένως να διοχετεύσουμε το συνενωμένο διάνυσμά τους σ' ένα *MLP* τριών επιπέδων, για να μάθουμε τη δυνητική συσχέτιση μεταξύ της ενσωμάτωσης αναγνώστη και της ενσωμάτωσης άρθρου, και για να κάνουμε πρόβλεψη βαθμολογιών, ως εξής:

$$\boldsymbol{z}_1 = \boldsymbol{h}_i \oplus \boldsymbol{q}_j \tag{6.12}$$

$$\boldsymbol{z}_l = \sigma\left(W_l^T \boldsymbol{z}_{l-1} + b_l\right) \tag{6.13}$$

$$\hat{y}_{ij} = \sigma\left(W^T \boldsymbol{z}_l + b\right) \tag{6.14}$$

όπου $\boldsymbol{h}_i$ η ενσωμάτωση του αναγνώστη, $\boldsymbol{q}_j$ η ενσωμάτωση του άρθρου, $\oplus$ ο *τελεστής συνένωσης* (*concatenation*), ενώ οι μεταβλητές W_l, b_l και σ συμβολίζουν τον πίνακα βαρών, τη *μεροληψία* και τη *συνάρτηση ενεργοποίησης* για το l-στρώμα αντίστοιχα. Τέλος, ορίζουμε ως *συνάρτηση απώλειας* τη *διασταυρούμενη εντροπία* (*cross-entropy loss*) της προβλεπόμενης βαθμολογίας $\hat{y}_{ij}$ ως ακολούθως:

$$L = -\sum_{(i,j)\in Y\cup Y^-} y_{ij} \cdot \log \hat{y}_{ij} + \left(1 - y_{ij}\right) \cdot \log\left(1 - \hat{y}_{ij}\right) \tag{6.15}$$

όπου $\hat{y}_{ij}$ η βαθμολογία που προβλέπουμε ότι ο υπό εξέταση αναγνώστης i θα έβαζε στο άρθρο j, ενώ y_{ij} η πραγματική βαθμολογία που ο ίδιος έχει ήδη βάλει. Τονίζουμε εδώ ότι το επαυξημένο σύνολο δεδομένων $Y \cup Y^-$ περιλαμβάνει εκτός από τα παρατηρούμενα ζεύγη αλληλεπίδρασης αναγνωστών-άρθρων Y και ζεύγη αλληλεπίδρασης αναγνωστών-άρθρων $\cup Y^-$ που δεν έχουν υπάρξει στην πραγματικότητα (μη παρατηρούμενα), έτσι ώστε να έχουμε *ισορροπία* (*balance*) ως προς την εκπαίδευση του μοντέλου πρόβλεψης ανάμεσα σε θετικές (παρατηρούμενες) και αρνητικές (μη παρατηρούμενες) αλληλεπιδράσεις μεταξύ αναγνωστών και άρθρων.

6.3 Ενσωματώσεις Γράφου (Graph Embeddings)

Οι **ενσωματώσεις γράφου (graph embeddings)** είναι εξελιγμένες μέθοδοι μείωσης της διαστατικότητας των χαρακτηριστικών των κόμβων ενός γράφου, οι οποίες έχουν οδηγήσει σε αποτελεσματικότερη **ανάλυση των γραφοδεδομένων (graph data analytics)**. Γι' αυτόν τον λόγο οι *ενσωματώσεις γράφου* έχουν χρησιμοποιηθεί στα συστήματα συστάσεων για τον **υπολογισμό της ομοιό-**

τητας (similarity inference) και την **πρόβλεψη συνδέσμων (link prediction)** μεταξύ χρηστών και στοιχείων ενός διμερούς γράφου. Λόγου χάρη, οι αλγόριθμοι *node2vec* και *DeepWalk* [Grover and Leskovec, 2016] μαθαίνουν διανυσματικές αναπαραστάσεις των κόμβων ενός γράφου με στόχο τον εντοπισμό "παρόμοιων" σημασιολογικά κόμβων. Πιο συγκεκριμένα, μαθαίνουν λανθάνουσες αναπαραστάσεις των κόμβων ενός γράφου μέσω της "διάσχισης" του γράφου με *τυχαίους* (μη προκαθορισμένους) *περιπάτους* (*random walks*). Οι δύο προαναφερθέντες αλγόριθμοι [Grover and Leskovec, 2016] ουσιαστικά εφαρμόζουν το *μοντέλο Skip-gram* για να μάθουν τις λανθάνουσες αναπαραστάσεις των κόμβων ενός γράφου (αντί για τις αναπαραστάσεις των λέξεων ενός δοθέντος κειμένου, όπου και αρχικά εφαρμόστηκε το προαναφερθέν μοντέλο, γνωστό στην παραπάνω εκδοχή του ως *word2vec*). Στην ίδια κατεύθυνση, ο *αλγόριθμος Metapath2vec* [Dong et al., 2017] επεκτείνει τους αλγορίθμους *DeepWalk* και *node2vec* καθώς εφαρμόζεται σε ετερογενείς γράφους εκτελώντας *τυχαίους περιπάτους* βασισμένους σε **μετα-μονοπάτια (meta paths)**.

Επιπλέον, υπάρχουν αρχιτεκτονικές *βαθέων νευρωνικών δικτύων*, όπως τα *νευρωνικά δίκτυα γράφων* (*GNNs*), τα *δίκτυα συν-εξέλιξης γράφων* (*GCNs*) κ.λπ., που διαδίδουν και σωρευτικά αθροίζουν επαναλαμβανόμενα τόσο τα *χαρακτηριστικά* (*node features*) όσο και τις *τοπολογικές πληροφορίες* (*node locality structure*) των κόμβων μέσα από τις τοπικές γειτονιές του υπό εξέταση κόμβου. Σκοπός τους είναι να "μάθουν" ένα λανθάνον διάνυσμα του προαναφερθέντος κόμβου που να είναι αντιπροσωπευτικότερο έναντι του αρχικού διανύσματος χαρακτηριστικών. Η μέθοδος *MCRec* [Hu et al., 2018], παραδείγματος χάρη, είναι μια επέκταση του *NCF* [He et al., 2017] και χρησιμοποιεί ως είσοδο τις ενσωματώσεις κόμβων που κατασκευάστηκαν μετά την εφαρμογή του αλγορίθμου *node2vec*, ο οποίος εφαρμόστηκε σε ακολουθίες κόμβων και, όπως προαναφέραμε, αποτελεί μια παραλλαγή του αλγορίθμου *word2vec* για γράφους. Τέλος, το *MP4Rec* [Ozsoy et al., 2020] είναι ένα *GNN* το οποίο είναι σε θέση να προσφέρει και ακριβείς αλλά και εξηγήσιμες συστάσεις. Για την εκμάθηση των τοπικών χαρακτηριστικών των κόμβων του γράφου χρησιμοποιεί *μετα-μονοπάτια* και τον αλγόριθμο *PathSim* [Yizhou et al., 2011].

6.4 Ανατροφοδοτούμενα Νευρωνικά Δίκτυα (Recurrent Neural Networks)

Ένα *ανατροφοδοτούμενο νευρωνικό δίκτυο* (*RNN*) διακρίνεται από ένα *νευρωνικό δίκτυο πρόσθιας τροφοδότησης* (π.χ. *MLP*) λόγω του ότι διαθέτει τουλάχιστον έναν βρόχο ανατροφοδότησης, δηλαδή μια νευρική σύναψη (ή αλλιώς, κατευθυνόμενη ακμή), η οποία συνδέεται όχι μ' έναν νευρώνα του επόμενου στρώματος του νευρωνικού δικτύου, αλλά μ' έναν νευρώνα του ίδιου ή προηγούμενου στρώματος. Βάσει του παραπάνω χαρακτηριστικού, η κύρια διαφορά μεταξύ ενός *CNN* και ενός *RNN* είναι η ικανότητα του δεύτερου να επεξεργάζεται χρονικώς εξαρτώμενα δεδομένα (π.χ. χρονοσειρές, ακολουθίες λέξεων κειμένου, συνεδρίες χρηστών κ.λπ.). Το βασικό λοιπόν στοιχείο ενός *RNN* είναι ότι υπάρχει μία είσοδος $\boldsymbol{x}_t$ σε κάθε χρονική στιγμή t και μία **κρυφή κατάσταση (hidden state)** $\boldsymbol{h}_t$, η οποία αλλάζει καθώς καταφθάνουν νέα στιγμιότυπα δεδομένων. Επίσης, σε

κάθε χρονική στιγμή υπάρχει και μια τιμή εξόδου $\hat{y}_t$. Ο όρος *recurrent*, εδώ "ανατροφοδοτούμενα", εκφράζει τη δυνατότητα που προσφέρει η αρχιτεκτονική των *RNNs* να εκτελούν την ίδια εργασία για κάθε στοιχείο μίας ακολουθίας με το αποτέλεσμα κάθε φορά να εξαρτάται από τους προηγούμενους υπολογισμούς. Επομένως, σ' αυτού του τύπου τα νευρωνικά δίκτυα οι πληροφορίες έχουν "μνήμη" επηρεαζόμενες από το παρελθόν.

Ωστόσο, στα *συστήματα συστάσεων που βασίζονται σε συνεδρίες χρηστών* (*session-based recommender systems*) έχουμε να κάνουμε κυρίως με *ανώνυμες συνεδρίες* (*user sessions*) και επομένως δεν είμαστε σε θέση να δημιουργήσουμε προφίλ χρηστών που να εκφράζουν τις αλληλεπιδράσεις τους με τα στοιχεία για μεγάλα χρονικά διαστήματα, επειδή αυτοί οι χρήστες δεν έχουν κάνει εγγραφή στην υπηρεσία συστάσεων. Γι' αυτόν τον λόγο οι συστάσεις αυτών των συστημάτων μοντελοποιούνται καλύτερα με *ανατροφοδοτούμενα νευρωνικά δίκτυα* ώστε να μπορεί το σύστημα να "ξεχνά" παρελθούσες αλληλεπιδράσεις των χρηστών με στοιχεία.

Ο Hidasi και οι λοιποί συν-συγγραφείς [Hidasi et al., 2015] παρουσίασαν ένα σύστημα συστάσεων, γνωστό ως *GRU4Rec*, το οποίο βασίζεται σε *Gated Recurrent Units* (*GRUs*). Αναλυτικότερα, μία μονάδα *GRU* είναι ένα απλό νευρωνικό δίκτυο που εμπεριέχει *συνδέσεις ανατροφοδότησης* (*feedback connections*) προκειμένου να μπορεί να "ξεχνά". Ο *αλγόριθμος GRU4Rec* μαθαίνει δηλαδή πότε και πώς να ενημερώνει την *κρυφή κατάσταση* (*hidden state*) της μονάδας *GRU* για να "λησμονεί" παλαιότερες αλληλεπιδράσεις των χρηστών με τα στοιχεία. Στην ίδια λογική, προτάθηκαν διάφορες αρχιτεκτονικές βελτιώσεις του *νευρωνικού δικτύου τύπου RNN* (βλ. [Quadrana et al., 2017]) που οδήγησαν σε αύξηση της αποτελεσματικότητας του αρχικού αλγορίθμου *GRU4Rec*. Επίσης, στην ερευνητική εργασία του De Souza Pereira Moreira και των λοιπών συν-συγγραφέων, [De Souza Pereira Moreira et al., 2018] και [Moreira et al., 2019], προτάθηκε ένα σύστημα συστάσεων με την ονομασία *CHAMELEON*, το οποίο χρησιμοποιεί ένα *νευρωνικό δίκτυο συν-εξέλιξης* (*CNN*) για την αξιοποίηση του περιεχομένου των άρθρων καθώς και ένα *νευρωνικό δίκτυο τύπου RNN* με μονάδες **μακράς και βραχυπρόθεσμης μνήμης (Long Short Term Memory - LSTM)** για τη διαδοχική επεξεργασία *δεδομένων χρήσης* (*usage data*) που αφορούν στη διάδραση αναγνωστών με ειδησεογραφικά άρθρα.

6.5 Ερωτήσεις Κεφαλαίου

1. Σχεδιάστε τα βασικά δομικά στοιχεία και περιγράψτε τη βασική λειτουργία ενός *νευρωνικού δικτύου ενός στρώματος τύπου Perceptron* (*Single-layer Perceptron*).

2. Να αναφέρετε κάποιες βασικές *συναρτήσεις ενεργοποίησης* που να μπορούν να χρησιμοποιηθούν σ' ένα νευρωνικό δίκτυο. Ποια είναι η διαφορά μεταξύ *γραμμικών* και *μη γραμμικών συναρτήσεων ενεργοποίησης*;

3. Σε τί χρησιμεύει η *μέθοδος οπίσθιας διάδοσης* (*back propagation*) στα νευρωνικά δίκτυα; Ποια είναι η σχέση της με τη μέθοδο της *επικλινούς καθόδου* (*gradient descent*);

4. Σχεδιάστε τα βασικά δομικά στοιχεία και περιγράψτε τη βασική λειτουργία ενός *νευρωνικού δικτύου πολλαπλών στρωμάτων τύπου Perceptron* (*Multi-layer Perceptron*).

5. Ποια είναι η εξίσωση για τον υπολογισμό των *λανθανόντων διανυσμάτων* σ' ένα *νευρωνικό δίκτυο πολλαπλών στρωμάτων τύπου Perceptron*; Περιγράψτε τη λειτουργία καθεμίας παραμέτρου της εξίσωσης ξεχωριστά.

6. Ποια είναι τα βασικά στρώματα ενός *συν-εξελικτικού νευρωνικού δικτύου* (*CNN*); Περιγράψτε τη λειτουργία του καθενός ξεχωριστά.

7. Ποια είναι η εξίσωση για τον υπολογισμό των *λανθανόντων διανυσμάτων* σ' ένα *Graph Convolutional Network*; Περιγράψτε τη λειτουργία καθεμίας παραμέτρου της εξίσωσης ξεχωριστά.

8. Περιγράψτε τον *Μηχανισμό Εστίασης σε Γράφο* (*Graph Attention Mechanism*). Ποια είναι η βασική λειτουργία που επιτελεί σ' ένα *νευρωνικό δίκτυο τύπου GCN*;

9. Περιγράψτε τη βασική λειτουργία ενός *ανατροφοδοτούμενου νευρωνικού δικτύου* (*Recurrent Neural Network*). Σε τί διαφέρει από ένα *δίκτυο πρόσθιας τροφοδότησης*;

6.6 Ασκήσεις Προγραμματισμού

Παραγοντοποίηση του πίνακα βαθμολογιών χρήστη-στοιχείου με τη βοήθεια ενός νευρωνικού δικτύου.

6.6.1 Άσκηση

Δίνεται ο πίνακας βαθμολογίας χρήστη-ταινίας **A** του Σχήματος 6.9, όπου οι στήλες M_{1-4} αντιπροσωπεύουν ταινίες, ενώ οι γραμμές U_{1-4} αφορούν σε χρήστες. Να υλοποιήσετε τον *αλγόριθμο παραγοντοποίησης πινάκων UV-decomposition* με τη βοήθεια ενός νευρωνικού δικτύου και να βρείτε τον *πλησιέστερο πίνακα προσέγγισης* $\hat{A}$ του αρχικού δοθέντος πίνακα A.

	M_1	M_2	M_3	M_4
U_1	4	1	1	4
U_2	1	4	2	0
U_3	2	1	4	5
U_4	1	4	1	?

Σχήμα 6.9: Πίνακας βαθμολογιών χρήστη-ταινιών **A** (4×4).

Λύση

Για το συγκεκριμένο παράδειγμα θα χρησιμοποιήσουμε τη *βιβλιοθήκη keras* (*pip install keras*), η οποία υλοποιεί διάφορους τύπους νευρωνικών δικτύων. Η *βιβλιοθήκη keras* θα τρέξει ως

frontend περιβάλλον, ενώ ως *backend* θα χρησιμοποιήσουμε τη *βιβλιοθήκη tensorflow* (*pip install tensorflow*), η οποία εμπεριέχει αλγορίθμους μηχανικής μάθησης. Συνεπώς, θα πρέπει να εγκατασταθούν οι παραπάνω βιβλιοθήκες στο *περιβάλλον* (*environment*) του *Anaconda* όπου θα τρέξουμε τον κώδικα της *Python*. Ακόμη, για να οπτικοποιήσουμε το μοντέλο του νευρωνικού δικτύου, θα πρέπει να εγκαταστήσουμε τη *βιβλιοθήκη graphviz* (*brew install graphviz*). Με τον ακόλουθο κώδικα κάνουμε *import* τις *βιβλιοθήκες keras, tensorflow,* και *graphviz*.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import warnings
import pydot
import keras
import graphviz
from sklearn.model_selection import train_test_split
from IPython.display import SVG
from tensorflow.keras.optimizers import SGD
from keras.utils.vis_utils import model_to_dot
from sklearn.metrics import mean_absolute_error
from keras.utils.vis_utils import model_to_dot
```

Στη συνέχεια, χωρίζουμε το *σύνολο δεδομένων* (*data set*) των 16 βαθμολογιών του Σχήματος 6.9 σε *train* (που περιέχει 12 βαθμολογίες) και σε *test* (που περιέχει 4 βαθμολογίες) *subsets*. Έπειτα, με βάση το *υποσύνολο των δεδομένων εκπαίδευσης* (*train subset*), θα προχωρήσουμε σε προβλέψεις βαθμολογιών για το *υποσύνολο των δεδομένων ελέγχου* (*test subset*).

```
dataset = pd.read_csv("u2.data",sep='\t',names="user_id,item_id
    ,rating".split(","))
dataset.user_id = dataset.user_id.astype('category').cat.codes.
    values
dataset.item_id = dataset.item_id.astype('category').cat.codes.
    values

print("Dataset \n")
print("User_id        Movie_id         Real_Rating")
for i in range(0,16):
```

```
    print(f"{str(dataset.user_id.iloc[i]+1) + chr(9) + chr(9) +
     str(dataset.item_id.iloc[i]+1) + chr(9) +  chr(9) +  str(
   dataset.rating.iloc[i])}")

Dataset

User_id         Movie_id         Real_Rating
    1               1               4
    1               2               1
    1               3               1
    1               4               4
    2               1               1
    2               2               4
    2               3               2
    2               4               0
    3               1               2
    3               2               1
    3               3               4
    3               4               5
    4               1               1
    4               2               4
    4               3               1
    4               4               0

train, test = train_test_split(dataset, test_size=0.2)
n_users, n_movies = len(dataset.user_id.unique()), len(dataset.
   item_id.unique())
print("\n train subset \n")
print("User_id        Movie_id         Real_Rating")
for i in range(0,12):
    print(f"{str(train.user_id.iloc[i]+1) + chr(9) + chr(9) +
   str(train.item_id.iloc[i]+1) + chr(9) +  chr(9) +  str(train
   .rating.iloc[i])}")

print("\n test subset \n")
print("User_id        Movie_id         Real_Rating")
```

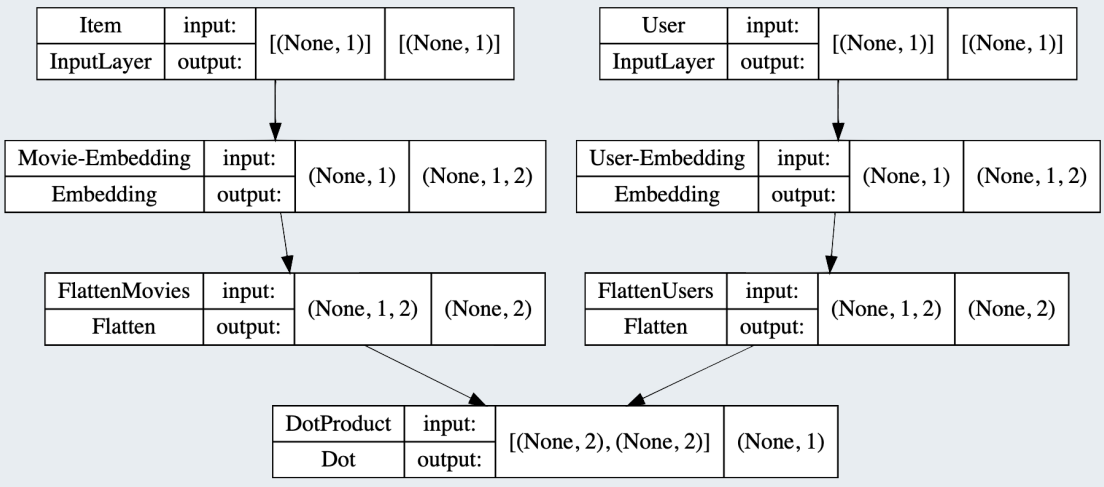
```
for i in range(0,4):
    print(f"{str(test.user_id.iloc[i]+1) + chr(9) + chr(9) +
   str(test.item_id.iloc[i]+1) + chr(9) +  chr(9) +  str(test.
   rating.iloc[i])}")

train subset
User_id         Movie_id         Real_Rating
    2               1               1
    3               4               5
    2               4               0
    3               3               4
    1               2               1
    1               4               4
    4               3               1
    4               2               4
    1               1               4
    2               3               2
    2               2               4
    3               2               1

 test subset
User_id         Movie_id         Real_Rating
    1               3               1
    4               1               1
    3               1               2
    4               4               0
```

Κατόπιν, προσπαθούμε να μάθουμε νέες (χαμηλής διαστατικότητας) αναπαραστάσεις χρηστών και ταινιών. Οι αναπαραστάσεις αυτές ονομάζονται και *ενσωματώσεις* (*embeddings*). Με τον παρακάτω κώδικα και με την *εντολή keras.layers.Embedding* δημιουργούμε τις *ενσωματώσεις* χρήστη (διάστασης 4 * 2) και ταινίας (διάστασης 4 * 2) αντίστοιχα. Έπειτα, θα συνδυάσουμε την *ενσωμάτωση* ενός χρήστη με κάθε *ενσωμάτωση* ταινίας χρησιμοποιώντας την πράξη του εσωτερικού γινόμενου ώστε να προβλέψουμε τη βαθμολογία ενός χρήστη για μια ταινία. Με άλλα λόγια, για την πρόβλεψη της βαθμολογίας κάθε ζεύγους χρήστη-ταινίας θα πάρουμε το εσωτερικό γινόμενο της αντίστοιχης *λανθάνουσας αναπαράστασης* του διανύσματος του χρήστη και του διανύσματος της ταινίας. Παραδείγματος χάρη, έστω ότι χρη-

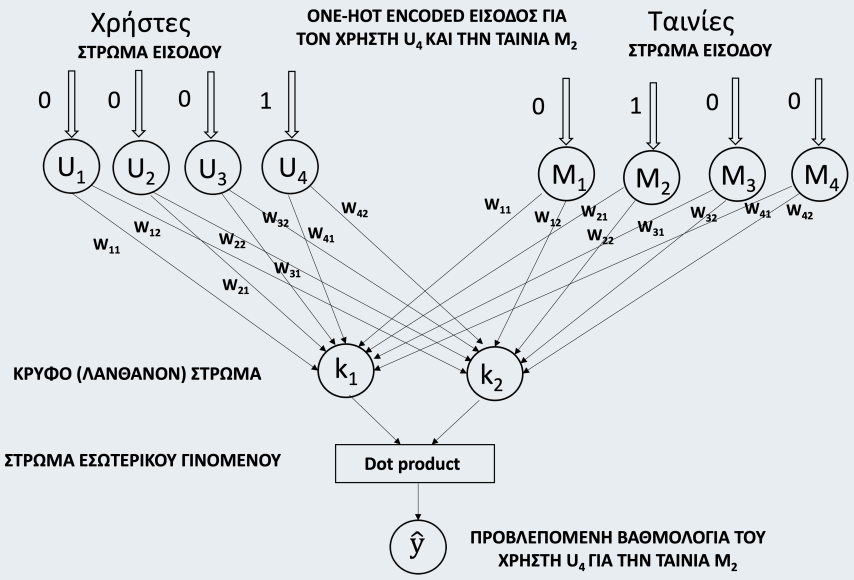
σιμοποιούμε $k = 2$ διαστάσεις για το λανθάνον διάνυσμα χρήστη και στοιχείου αντίστοιχα. Τα λανθάνοντα αυτά διανύσματα αντιστοιχούν λόγου χάρη στο πόσο αρέσει στον χρήστη η δράση και το ρομαντικό στοιχείο, ενώ το λανθάνον διάνυσμα της ταινίας στο πόση δράση και πόσο ρομαντικό στοιχείο υπάρχει στην ταινία (δηλαδή σε δύο διαστάσεις μόνο).

```
n_latent_factors_user = 2
n_latent_factors_movie = 2
movie_input = keras.layers.Input(shape=[1],name='Item')
movie_vec = keras.layers.Flatten(name='FlattenMovies')(keras.
    layers.Embedding(n_movies, n_latent_factors_movie,name='
    Movie-Embedding')(movie_input))
user_input = keras.layers.Input(shape=[1],name='User')
user_vec = keras.layers.Flatten(name='FlattenUsers')(keras.
    layers.Embedding(n_users, n_latent_factors_user,name='User-
    Embedding')(user_input))

prod = keras.layers.dot([movie_vec, user_vec], axes=1, name='
    DotProduct')
model = keras.Model([user_input, movie_input], prod)

SVG(model_to_dot(model, show_shapes=True, show_layer_names=
    True, rankdir='HB').create(prog='dot', format='svg'))
```

Με την εκτέλεση του παραπάνω κώδικα εμφανίζεται και μια σχηματική αναπαράσταση του νευρωνικού δικτύου του παραδείγματός μας που υλοποιεί την *παραγοντοποίηση πινάκων* (*matrix factorization*).



Το παραπάνω μοντέλο λαμβάνει δύο εισόδους: έναν χρήστη και μία ταινία. Για να εκφράσουμε έναν χρήστη, χρησιμοποιούμε 2 διαστάσεις. Δεδομένου ότι έχουμε 4 χρήστες στο παράδειγμά μας, θα δημιουργηθεί ένας πίνακας διαστάσεων 2 * 4 (πίνακας βαρών των χρηστών). Αντίστοιχα και για τις ταινίες θα δημιουργηθεί ένας πίνακας διαστάσεων 2 * 4 (πίνακας βαρών των ταινιών). Για να γίνει περισσότερο κατανοητή η δομή του νευρωνικού δικτύου, παραθέτουμε και το παρακάτω Σχήμα:



Για να δούμε πόσες είναι οι συνολικές παράμετροι που θέλουμε να μάθουμε στο μοντέλο μας, εκτελούμε τον παρακάτω κώδικα:

```
model.summary()
```

```
Model: "model_1"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
Item (InputLayer)               [(None, 1)]          0           []

User (InputLayer)               [(None, 1)]          0           []

Movie-Embedding (Embedding)     (None, 1, 2)         8           ['Item[0][0]']

User-Embedding (Embedding)      (None, 1, 2)         8           ['User[0][0]']

FlattenMovies (Flatten)         (None, 2)            0           ['Movie-Embedding[0][0]']

FlattenUsers (Flatten)          (None, 2)            0           ['User-Embedding[0][0]']

DotProduct (Dot)                (None, 1)            0           ['FlattenMovies[0][0]',
                                                                 'FlattenUsers[0][0]']

==================================================================================================
Total params: 16
Trainable params: 16
Non-trainable params: 0
```

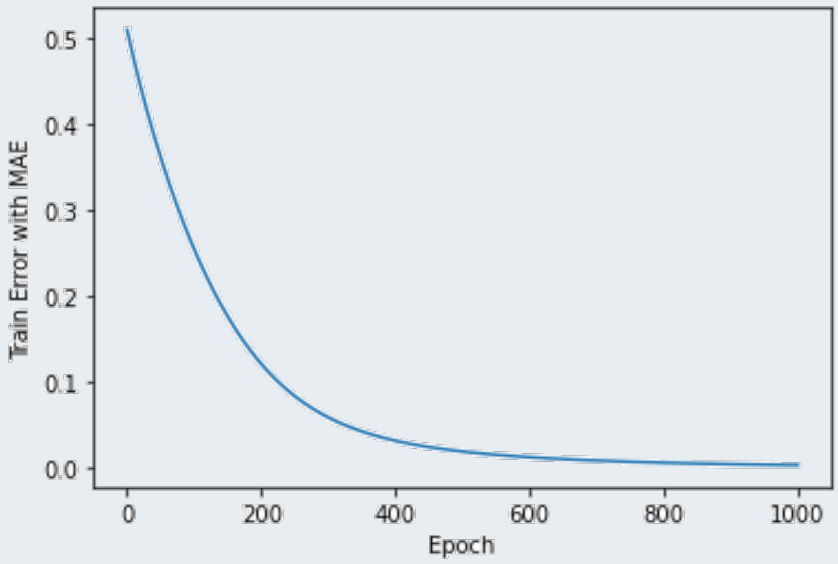
Όπως φαίνεται στην παραπάνω εικόνα, έχουμε 16 παραμέτρους για να εκπαιδεύσουμε στο μοντέλο πρόβλεψης. Από αυτές, οι 2*4 = 8 αφορούν στους χρήστες, ενώ οι 2*4 = 8 στις ταινίες, και αντιστοιχούν κατά αυτήν τη σειρά στα στοιχεία των πινάκων βαρών για τους χρήστες και για τις ταινίες.

Για τις ανάγκες του παραδείγματός μας και με σκοπό να εκπαιδεύσουμε το μοντέλο μας, θα το τρέξουμε για 1000 επαναλήψεις ή αλλιώς *εποχές* (*epochs*) με τη χρήση του παρακάτω κώδικα:

```
model.compile('SGD', 'mean_squared_error')
history = model.fit([train.user_id, train.item_id], train.
    rating, epochs=1000, verbose=1)

y_hat = np.round(model.predict([test.user_id, test.item_id]),0)
y_true = test.rating

pd.Series(history.history['loss']).plot()
plt.xlabel("Epoch")
plt.ylabel("Train Error with MAE")

Epoch 1/1000
1/1 [==============================] - 0s 428ms/step - loss:
    0.5089
Epoch 2/1000
1/1 [==============================] - 0s 4ms/step - loss:
    0.5054
Epoch 3/1000
1/1 [==============================] - 0s 3ms/step - loss:
    0.5018
Epoch 4/1000
1/1 [==============================] - 0s 3ms/step - loss:
    0.4983
Epoch 5/1000
1/1 [==============================] - 0s 4ms/step - loss:
....
.....

1/1 [==============================] - 0s 2ms/step - loss:
    0.0028
```

```
Epoch 999/1000
1/1 [==============================] - 0s 3ms/step - loss:
    0.0028
Epoch 1000/1000
1/1 [==============================] - 0s 2ms/step - loss:
    0.0028
```

Όπως φαίνεται από την πρώτη σειρά του παραπάνω κώδικα, η βελτιστοποίηση του μοντέλου μας θα γίνει με τη μέθοδο *Stochastic Gradient Descent* (*SGD*), και η μετρική που θα χρησιμοποιηθεί για να υπολογίσει τη διαφορά μεταξύ της προβλεπόμενης βαθμολογίας του χρήστη (y_hat) για μια ταινία έναντι της πραγματικής του βαθμολογίας (y_true) είναι το *Μέσο Απόλυτο Σφάλμα* (*Mean Absolute Error - MAE*). Αρκεί, τώρα, να δούμε το παρακάτω Σχήμα για να διαπιστώσουμε ότι το *MAE* μειώνεται δραστικά (0.0028 στα δεδομένα εκπαίδευσης) μετά από τις 1000 επαναλήψεις (*εποχές - epoch*).



Ακολούθως, με τον παρακάτω κώδικα "εκτυπώνουμε" για κάθε ζευγάρι χρήστη-ταινίας την πραγματική και την προβλεπόμενη βαθμολογία για το *train* και για το *test subset*:

```
predictions = model.predict([train.user_id.head(12), train.
    item_id.head(12)])
print("\n train subset \n")
print("User_id        Movie_id        Real_Rating      [
    Predicted_Rating]")
for i in range(0,12):
    print(f"{str(train.user_id.iloc[i]+1) + chr(9) + chr(9) +
```

```
    str(train.item_id.iloc[i]+1) + chr(9) +  chr(9) +  str(train
    .rating.iloc[i]) + chr(9) + chr(9) + str(predictions[i])  }"
    )

predictions = model.predict([test.user_id.head(4), test.item_id
    .head(4)])
print("\n train subset \n")
print("User_id       Movie_id       Real_Rating     [
    Predicted_Rating]")
for i in range(0,4):
    print(f"{str(test.user_id.iloc[i]+1) + chr(9) + chr(9) +
    str(test.item_id.iloc[i]+1) + chr(9) +  chr(9) +  str(test.
    rating.iloc[i]) + chr(9) + chr(9) + str(predictions[i])  }")
```

```
 train subset

User_id       Movie_id       Real_Rating     [Predicted_Rating]
2               4              0               [0.21990767]
4               4              0               [0.05071604]
1               4              4               [4.2736845]
2               1              1               [0.6660578]
4               3              1               [1.047406]
3               4              5               [4.68342]
3               1              2               [2.6894011]
4               2              4               [3.381989]
3               3              4               [3.9802878]
1               1              4               [3.3085825]
1               2              1               [1.6772952]
3               2              1               [0.5219314]

 test subset

User_id       Movie_id       Real_Rating     [Predicted_Rating]
2               2              4               [0.7832108]
2               3              2               [0.4125717]
1               3              1               [3.9893165]
4               1              1               [2.4312117]
```

Τέλος, ακολουθεί ο κώδικας που εμφανίζει τις *λανθάνουσες αναπαραστάσεις* των χρηστών, των ταινιών καθώς και τον τελικό πίνακα με τις κατά προσέγγιση προβλεπόμενες βαθμολογίες των χρηστών για ταινίες.

```
movie_embedding_learnt = model.get_layer(name='Movie-Embedding').get_weights()[0]
print("\n Ο πίνακας βαρών διάστασης 4*2 με τις ενσωματώσεις των ταινιών \n")
print(pd.DataFrame(movie_embedding_learnt))
print("\n Ο πίνακας βαρών διάστασης 4*2 με τις ενσωματώσεις των χρηστών \n")
user_embedding_learnt = model.get_layer(name='User-Embedding').get_weights()[0]
print(pd.DataFrame(user_embedding_learnt))
print("\n Ο πίνακας βαθμολογίας χρηστώνταινιών- 4*4 που προβλέπει το μοντέλο μας \n")
print(np.around(np.asmatrix(user_embedding_learnt)*np.asmatrix(movie_embedding_learnt).T,2))
```

```
 Ο πίνακας βαρών διάστασης 4*2 με τις ενσωματώσεις των ταινιών

          0         1
0  1.518744  0.657337
1  1.671875 -0.485499
2  1.040667  1.510207
3  0.665299  2.025937

 Ο πίνακας βαρών διάστασης 4*2 με τις ενσωματώσεις των χρηστών

          0         1
0  1.475147  1.625061
1  0.456455 -0.041349
2  0.897869  2.016879
3  1.853398 -0.583605

 Ο πίνακας βαθμολογίας χρηστών-ταινιών 4*4 που προβλέπει το μοντέλο μας

[[3.31 1.68 3.99 4.27]
 [0.67 0.78 0.41 0.22]
 [2.69 0.52 3.98 4.68]
 [2.43 3.38 1.05 0.05]]
```

Όπως βλέπουμε λοιπόν στο παραπάνω Σχήμα, η πρόβλεψη της βαθμολογίας του χρήστη 4 για την ταινία 4 είναι 0.05. Αυτή η πρόβλεψη είναι λογική και αναμενόμενη, δεδομένου ότι η βαθμολογική συμπεριφορά του χρήστη 4 είναι παρόμοια με αυτήν του χρήστη 2 και ο δεύτερος έχει βαθμολογήσει την ταινία 4 με μηδέν (0) όπως φαίνεται στο Σχήμα 6.9.

Βιβλιογραφία

L. Chen, L. Wu, R. Hong, K. Zhang, and M. Wang. Revisiting graph based collaborative filtering: A linear residual graph convolutional network approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 27--34, 2020.

G. De Souza Pereira Moreira, F. Ferreira, and A. M. da Cunha. News session-based recommendations using deep neural networks. In *Proceedings of the 3rd Workshop on Deep Learning for Recommender Systems*, pages 15--23. ACM, 2018.

Y. Dong, V. Chawla, and A. Swami. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 135--144. ACM, 2017.

C. Feng, Z. Liu, S. Lin, and T. Quek. Attention-based graph convolutional network for recommendation system. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7560--7564. IEEE, 2019.

A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855--864. ACM, 2016.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770--778, 2016.

X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, pages 173--182. International World Wide Web Conferences Steering Committee, 2017.

B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk. Session-based recommendations with recurrent neural networks. *CoRR*, abs/1511.06939, 2015. URL `http://arxiv.org/abs/1511.06939`.

B. Hu, C. Shi, W. X. Zhao, and P. S. Yu. Leveraging meta-path based context for top-n recommendation with a neural co-attention model. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1531--1540. ACM, 2018.

P. Moreira, D. Jannach, and A. M. da Cunha. Contextual hybrid session-based news recommendation with recurrent neural networks. *arXiv preprint arXiv:1904.10367*, 2019.

M. G. Ozsoy, D. O'Reilly-Morgan, P. Symeonidis, E. Z. Tragos, N. Hurley, B. Smyth, and A. Lawlor. Mp4rec: Explainable and accurate top-n recommendations in heterogeneous information networks. *IEEE Access*, 8:181835--181847, 2020.

M. Quadrana, A. Karatzoglou, B. Hidasi, and P. Cremonesi. Personalizing session-based recommendations with hierarchical recurrent neural networks. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, RecSys '17, pages 130--137, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4652-8. 10.1145/3109859.3109896. URL `http://doi.acm.org/10.1145/3109859.3109896`.

R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 974--983, 2018.

S. Yizhou, H. Jiawei, Y. Xifeng, Y. Philip S., and W. Tianyi. Pathsim: Meta path-based top-k similarity search in heterogenuos information networks. In *Proceedings of the VLDB Endowment (VLDB'2011)*, Seattle, Washigton, 2011.