ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«ΑΝΑΠΤΥΞΗ ΑΛΓΟΡΙΘΜΩΝ ΒΑΘΙΑΣ ΜΑΘΗΣΗΣ ΓΙΑ ΔΕΔΟΜΕΝΑ ΠΟΛΛΑΠΛΩΝ ΕΤΙΚΕΤΩΝ»

(Deep Learning Algorithms for Multi-Label Data)

«ΓΚΑΝΙΑΣ ΕΥΡΙΠΙΔΗΣ»

ΑΕΜ: 1866 **Κατεύθυνση**: Πληροφοριακά Συστήματα

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ:

Τσουμάκας Γρηγόριος, ΕΠΙΚΟΥΡΟΣ ΚΑΘΗΓΗΤΗΣ

ΘΕΣΣΑΛΟΝΙΚΗ 2013

ΠΕΡΙΛΗΨΗ

Η περιοχή των δεδομένων πολλαπλών ετικετών, έχει προσελκύσει το ενδιαφέρον πολλών ερευνητών που ασχολούνται με προβλήματα μάθησης με επίβλεψη, για δύο βασικούς λόγους. Πρώτον, αυτού του είδους τα δεδομένα ανακύπτουν σε μια πληθώρα εφαρμογών, όπως η σημασιολογική δεικτοδότηση εγγράφων, μουσικής και βίντεο, η πρόβλεψη της λειτουργίας των πρωτεϊνών, η ιατρική διάγνωση, η ανακάλυψη φαρμάκων και η κατηγοριοποίηση ερωτημάτων μηχανών αναζήτησης. Δεύτερον, τα δεδομένα πολλαπλών ετικετών παρουσιάζουν ενδιαφέρουσες ερευνητικές προκλήσεις όπως η αξιοποίηση των συσχετίσεων μεταξύ των ετικετών και η κλιμάκωση σε μεγάλο αριθμό ετικετών.

Χαρακτηριστικό των δεδομένων πολλαπλών ετικετών είναι ότι κάθε αντικείμενο ενδιαφέροντος χαρακτηρίζεται με μία ή παραπάνω ετικέτες από ένα σύνολο ετικετών και σκοπός δεν είναι μια απλή ταξινόμηση (classification) ενός αντικειμένου σε μία από τις ετικέτες, αλλά μια κατάταξη (ranking) των ετικετών ως προς τη συνάφειά τους με το αντικείμενο ή μια διχοτόμηση του συνόλου των ετικετών σε εκείνες που σχετίζονται με το αντικείμενο και σε εκείνες που δεν σχετίζονται με αυτό (multi-label classification). Στην εργασία αυτή μελετάμε αυτή την περιοχή και εφαρμόζουμε μια τεχνική βαθιάς μάθησης (deep learning), εμπνευσμένη από τη θεωρία που εξηγεί πώς το μυαλό αναγνωρίζει τα πρότυπα. Τεχνολογικές εταιρίες, οι οποίες έχουν εφαρμόσει αυτή την τεχνική σε διάφορους τομείς όπως η τεχνητή όραση, η αναγνώριση ομιλίας και ο προσδιορισμός των πολλά υποσχόμενων νέων μορίων για το σχεδιασμό φαρμάκων, αναφέρουν σημαντικά κέρδη σε ακρίβεια πρόβλεψης.

Παρουσιάζουμε ένα σχετικά πρόσφατο μοντέλο νευρωνικών δικτύων, τα δίκτυα πεποίθησης μεγάλου βάθους. Αυτά τα νευρωνικά δίκτυα μπορούν να χειριστούν δεδομένα πολλαπλών ετικετών χωρίς να χρειάζονται μετασχηματισμό πριν ή μετά την εκπαίδευση. Αυτός ο μετασχηματισμός είναι ανεπιθύμητος, γιατί συνήθως μεγαλώνει το σύνολο των δεδομένων. Σκοπός μας είναι να καταφέρουμε να φτιάξουμε ένα μοντέλο που θα χειρίζεται δεδομένα πολλαπλών ετικετών, χρησιμοποιώντας τεχνικές βαθιάς μάθησης. Στα πλαίσια αυτής της εργασίας, καταφέραμε να δείξουμε ότι αυτές οι τεχνικές – και συγκεκριμένα η τεχνική βαθιάς μάθησης με δίκτυα πεποίθησης – δίνει καλύτερα αποτελέσματα από άλλες αντίστοιχες στα περισσότερα από τα είδη προβλημάτων που παρουσιάστηκαν.

ABSTRACT

Many scientists, who specialize in supervised learning, are interested in multi-label data for two main reasons. First of all, this type of data arising at a variety of applications such as semantic indexing of documents, music and video, protein function prediction, medical diagnosis, drug discovery and search engine queries clustering. Secondly, multi-label data perform interesting research challenges like the use of correlations between labels and scaling to a large number of labels.

In multi-label data, every subject of interest is characterized by one or more labels from a label set and the purpose is not a simple classification of an instance, but a label ranking for their relevance to the subject or a bipartition of them to these that are relevant and those which are not (multi-label classification). In this project, we investigate this field and we apply a deep learning method, inspired by the theory that explains how the brain recognizes patterns. Technology companies are reporting startling gains in fields as diverse as computer vision, speech recognition and the identification of promising new molecules for designing drugs.

We present a fairly recent kind of neural network, the deep belief network, which handles multi-label data without transforming them before or after the training. This transformation is undesirable, because it causes the dataset to be bigger. Our aim is to build a model that can handle multi-label data using deep learning techniques. In this project, we showed that these techniques – specifically the deep learning using belief networks – have better results in the most of the encountered subjects of interest.

ΕΥΧΑΡΙΣΤΙΕΣ

Πριν την παρουσίαση της παρούσας εργασίας, αισθάνομαι την υποχρέωση να ευχαριστήσω ορισμένους από τους ανθρώπους που γνώρισα, συνεργάστηκα μαζί τους και έπαιξαν πολύ σημαντικό ρόλο στην πραγματοποίησή της.

Πρώτο από όλους, θέλω να ευχαριστήσω τον κύριο Γρηγόριο Τσουμάκα, για την πολύτιμη βοήθεια που μου προσέφερε κατά την εκπόνηση αυτής της εργασίας. Τον ευχαριστώ που ήταν δίπλα μου και με βοήθησε σε ότι χρειάστηκα, με τις γνώσεις, τις συμβουλές και την εμπειρία του. Ευχαριστώ, επίσης, για το χρόνο που διέθεσε για αυτήν την εργασία και για την εμπιστοσύνη που μου έδειξε για να μου την αναθέσει.

Θέλω, επίσης, να ευχαριστήσω τον κύριο Αναστάσιο Τέφα, για τις πληροφορίες που μου έδωσε για το γνωστικό αντικείμενο των νευρωνικών δικτύων. Ευχαριστώ για τη διάλεξη που έγινε πάνω στο θέμα των νευρωνικών δικτύων βαθιάς μάθησης στα πλαίσια του μαθήματος “Υπολογιστική Νοημοσύνη”, η οποία δεν ήταν στο πρόγραμμα να γίνει.

Τέλος, θέλω να ευχαριστήσω συγγενείς και φίλους που με βοήθησαν και με στήριξαν ο καθένας με τρόπο του και υπομείνανε την πολύωρη ενασχόλησή μου με το αντικείμενο.

29 Ιουνίου 2013

Ευριπίδης Γκανιάς

ΠΕΡΙΕΧΟΜΕΝΑ

ΛΙΣΤΑ ΣΧΗΜΑΤΩΝ

ΛΙΣΤΑ ΠΙΝΑΚΩΝ

ΚΕΦΑΛΑΙΟ 1: ΕΙΣΑΓΩΓΗ

ΕΙΣΑΓΩΓΗ

Αντικείμενο της παρούσας εργασίας είναι η μελέτη των βαθιών (νευρωνικών) δικτύων πεποίθησης και η διερεύνηση της καταλληλότητας τους για μάθηση από δεδομένα πολλαπλών ετικετών. Πιο συγκεκριμένα, παρουσιάζουμε τον τρόπο με τον οποίο δομούνται αυτά τα δίκτυα, πως εκπαιδεύονται και που στηρίζεται η φιλοσοφία τους. Επίσης, πειραματιζόμαστε με τις παραμέτρους τους και παρατηρούμε τη συμπεριφορά τους με διάφορα σύνολα δεδομένων πολλαπλών ετικετών. Στη συνέχεια περιγράφονται τα υπόλοιπα κεφάλαια από τα οποία αποτελείται η εργασία.

Στο Κεφάλαιο 2 παρουσιάζουμε το γνωστικό υπόβαθρο της εργασίας στις περιοχές των νευρωνικών δικτύων και της μάθησης από δεδομένα πολλαπλών ετικετών. Απαντάμε στις ερωτήσεις του τύπου: τι είναι τα νευρωνικά δίκτυα, τι η στοχαστική μηχανική και πως αυτά συνδυάζονται; Μιλάμε συνοπτικά για τη δομή του γενικού μοντέλου των νευρωνικών δικτύων, τη φύση των νευρώνων του, τις συναρτήσεις ενεργοποίησης, τη στοχαστική μηχανική και πως αυτή επηρεάζει τον τρόπο μάθησης των νευρωνικών δικτύων. Επίσης, αναλύουμε το στοχαστικό μοντέλο ενός νευρώνα, περιγράφουμε την τεχνική της προσομοιωμένης ανόπτησης και παρέχουμε μια βάση για τις μηχανές Boltzmann. Στη συνέχεια, εξετάζουμε τις μηχανές Boltzmann και τα βαθιά δίκτυα πεποίθησης (Deep Belief Networks – DBNs), τα οποία αποτελούνται από περιορισμένες μηχανές Boltzmann (Restricted Boltzmann Machines – RBMs). Τέλος, παρουσιάζουμε το αντικείμενο της μάθησης από δεδομένα πολλαπλών ετικετών και αναφέρουμε κάποιες ιδιαίτερες μετρικές αξιολόγησης και τεχνικές μάθησης για τέτοιου είδους δεδομένα..

Στο Κεφάλαιο 3, παρουσιάζουμε ένα μοντέλο νευρωνικού δικτύου, το οποίο χειρίζεται δεδομένα πολλών ετικετών. Εκπαιδεύεται με τέτοιου είδους δεδομένα και μπορεί να κάνει προβλέψεις για πολλαπλές ετικέτες. Αναλύουμε την αρχιτεκτονική και τους αλγορίθμους εκπαίδευσης και πρόβλεψης που χρησιμοποιεί. Στο ίδιο κεφάλαιο, υπάρχει μια σύντομη περιγραφή της υλοποίησης και των πακέτων που χρησιμοποιήθηκαν κατά τη συγγραφή του κώδικα.

Στο Κεφάλαιο 4, παραθέτουμε τα αποτελέσματα από κάποια πειράματα που εκτελέστηκαν. Για τα πειράματα, δοκιμάσαμε ενδεικτικά διάφορους τύπους δεδομένων, για να δούμε πως συμπεριφέρονται τα DBNs σε αυτά. Τα δεδομένα των πειραμάτων διαφέρουν ως προς τον τύπο, το πλήθος των χαρακτηριστικών, των ετικετών και το μέγεθος του συνόλου των παραδειγμάτων. Επίσης, σε αυτό το κεφάλαιο, περιγράφονται περιληπτικά μερικά σχετικά σύνολα δεδομένων πολλαπλών ετικετών, από τα οποία κάποια χρησιμοποιήθηκαν στα πειράματα.

Στο Κεφάλαιο 5, παρουσιάζονται τα συμπεράσματα που προέκυψαν από αυτή την εργασία. Αναφέρονται οι δυσκολίες που αντιμετωπίσαμε και πως αυτές ξεπεράστηκαν, καθώς και συμβουλές για παρόμοιες εργασίες. Επίσης, ανασκοπούμε στο τι μάθαμε κατά τη διάρκεια της υλοποίησης και της εκτέλεσης των παραδειγμάτων. Στο τέλος αυτού του κεφαλαίου αναφέρουμε πιθανές επεκτάσεις της εργασίας αυτής, καθώς και πώς αυτές θα μπορούσαν να γίνουν.

Στο Παράρτημα Ι, υπάρχουν κάποια διαγράμματα UML που περιγράφουν τις συνδέσεις των κλάσεων και πώς αυτές αλληλεπιδρούν μεταξύ τους για να δημιουργήσουν το τελικό αποτέλεσμα.

ΚΕΦΑΛΑΙΟ 2: ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ

ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ

Στο κεφάλαιο αυτό θα περιγράψουμε τι είναι ένα νευρωνικό δίκτυο, πως εκπαιδεύεται, καθώς και τα απαραίτητα εργαλεία που χρειάζονται για αυτό. Θα εξηγήσουμε τον τρόπο με τον οποίο δουλεύουν καθώς και το πώς ξεκίνησαν.

Αναλύουμε το γενικό μοντέλο των νευρώνων, καθώς και το στοχαστικό το οποίο χρησιμοποιείται σε μια οικογένεια νευρωνικών δικτύων όπως οι μηχανές Boltzmann. Η οικογένεια νευρωνικών δικτύων στην οποία ανήκουν οι μηχανές Boltzmann είναι οι στοχαστικές μηχανές, που βασίζονται στη στατιστική μηχανική. Η θεωρία των νευρωνικών δικτύων χρησιμοποιεί τη λογική και τους όρους της στατιστικής μηχανικής για να περιγράψει νευρωνικά δίκτυα αυτής της οικογένειας.

Στη συνέχεια θα περιγραφούν τα Δίκτυα Πεποίθησης Μεγάλου Βάθους τα οποία χρησιμοποιούν μια παραλλαγή των μηχανών Boltzmann, τις περιορισμένες μηχανές Boltzmann, με σκοπό να μειώσουν το χρόνο εκπαίδευσης, αλλά να κρατήσουν τα καλά χαρακτηριστικά των μηχανών Boltzmann.

2.1 ΤΕΧΝΗΤΑ ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ

Ο ανθρώπινος εγκέφαλος είναι ένας εξαιρετικά πολύπλοκος, μη γραμμικός, παράλληλος υπολογιστής, ο οποίος έχει τη δυνατότητα να οργανώνει τα δομικά του στοιχεία (νευρώνες) με τρόπο ώστε να εκτελούν συγκεκριμένους υπολογισμούς, με πολλαπλάσια ταχύτητα από αυτή του γρηγορότερου ψηφιακού υπολογιστή στον κόσμο (Haykin, 2010).

Πώς, όμως, ο ανθρώπινος εγκέφαλος καταφέρνει τόσο πολύπλοκα πράγματα όπως η όραση, η ομιλία, η ακοή, κτλ; Κατά τη γέννηση ο εγκέφαλος έχει ήδη κάποια δομή και μπορεί να κατασκευάσει δικούς του κανόνες συμπεριφοράς χρησιμοποιώντας την «εμπειρία». Με την πάροδο του χρόνου, η εμπειρία συσσωρεύεται – κυρίως τα δύο πρώτα χρόνια – και βοηθά στην ανάπτυξη του εγκεφάλου.

Μια βασική ιδιότητα του νευρικού συστήματος είναι ότι προσαρμόζεται ανάλογα με το περιβάλλον του. Αυτή την ιδέα είχαν οι Warren McCulloch και Walter Pitts (1943) οι οποίοι προσπάθησαν να εξηγήσουν πως μπορεί να λειτουργούν οι νευρώνες του ανθρώπινου εγκέφαλου, προσομοιώνοντάς τους σε ένα ηλεκτρικό κύκλωμα. Στην πλέον γενική μορφή του, ένα «Τεχνητό Νευρωνικό Δίκτυο» (ΤΝΔ) είναι μια μηχανή σχεδιασμένη ώστε να μοντελοποιεί τον τρόπο με τον οποίο ο εγκέφαλος εκτελεί μια συγκεκριμένη λειτουργία.

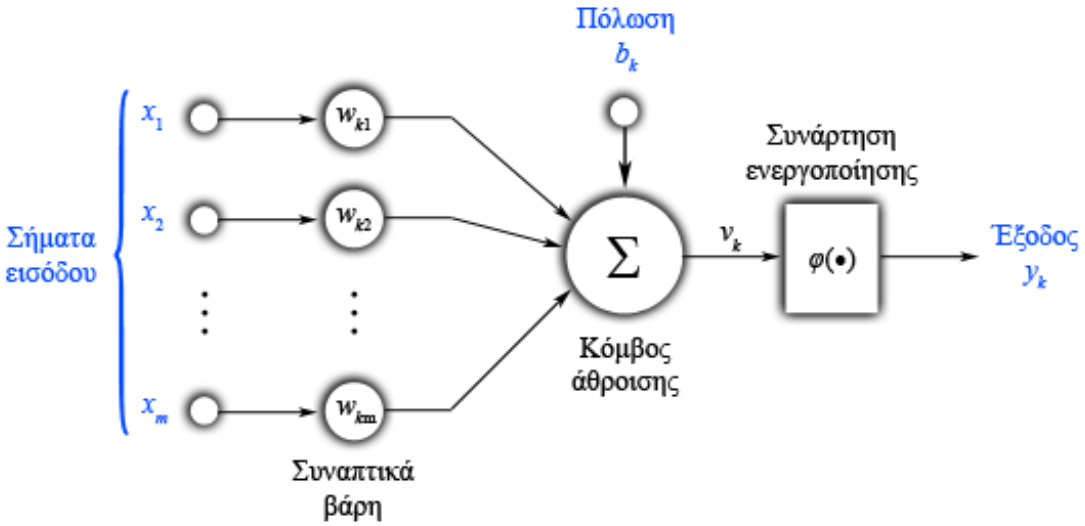
2.1.1 Μοντέλο Νευρώνων

Οι νευρώνες αποτελούν ξεχωριστές μονάδες επεξεργασίας πληροφορίας, η σύνδεση των οποίων δημιουργεί ένα νευρωνικό δίκτυο. Ο γενικός τύπος νευρώνα αποτελείται από τρία βασικά στοιχεία:

1. Ένα σύνολο από *συνάψεις* (ή *διασυνδέσεις*), η καθεμία από τις οποίες έχει δικό της βάρος. Στην πράξη, ένα σήμα εισόδου x_j που περνάει από τη σύναψη j για να φτάσει στον νευρώνα k πολλαπλασιάζεται με το βάρος w_{kj}
2. Έναν *αθροιστή (adder)*, ο οποίος αθροίζει όλα τα σήματα εισόδου πολλαπλασιασμένα με τα αντίστοιχα βάρη των συνάψεων.

3. Μια *συνάρτηση ενεργοποίησης (activation function)* για τον περιορισμό του πλάτους του σήματος εξόδου ενός νευρώνα.

Στο Σχ. 2.1.1 οι συνάψεις του νευρώνα είναι τα βέλη από τα σήματα εισόδου στον κόμβο άθροισης (αθροιστής). Παρατηρούμε ότι υπάρχει μια σταθερά πόλωσης b_k, η οποία είναι μια εξωτερική παράμετρος του νευρώνα k.



Σχήμα 2.1.1 Μη γραμμικό μοντέλο νευρώνα

Μπορούμε να περιγράψουμε μαθηματικά το μοντέλο του νευρώνα που απεικονίζεται στο Σχ. 2.1.1 με ένα ζεύγος εξισώσεων:

$$v_k = \sum_{j=1}^{m} w_{kj}\, x_j + b_k$$

και

$$y_k = \varphi(v_k)$$

όπου $x_1, x_2, \ldots, x_m$ είναι το διάνυσμα εισόδου, $w_{k1}, w_{k2}, \ldots, w_{km}$ τα αντίστοιχα συναπτικά βάρη του νευρώνα k, v_k το τοπικό πεδίο του νευρώνα, b_k η πόλωση, $\varphi(\cdot)$ η συνάρτηση ενεργοποίησης και y_k η έξοδος του νευρώνα.

Με άλλα λόγια, ένας νευρώνας συμβολίζει μια συνάρτηση m μεταβλητών, όπου m το μέγεθος του διανύσματος εισόδου. Ανάλογα με το αν η πόλωση b_k είναι θετική ή αρνητική η συνάρτηση αυτή μετακινείται παράλληλα στο υπερεπίπεδο. Αν η πόλωση δεν υπήρχε, τότε η συνάρτηση θα περνούσε αναγκαστικά από την αρχή των αξόνων. Πολλές φορές, για ευκολία στις πράξεις, εντάσσουμε την πόλωση στο διάνυσμα εισόδου ως $x_0 = +1$, προσθέτουμε το συναπτικό βάρος w_{k0} και η εξίσωση για τον υπολογισμό του τοπικού πεδίου γίνεται:

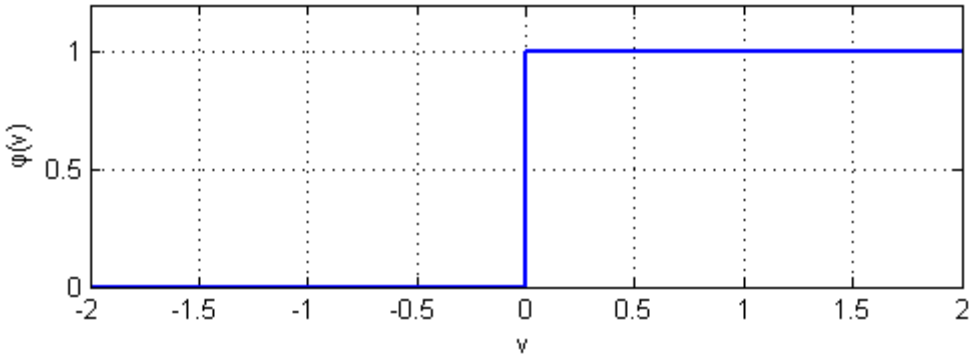
$$v_k = \sum_{j=0}^{m} w_{kj}\, x_j$$

Το συναπτικό βάρος w_{k0} πρέπει να είναι ίσο με την πόλωση b_k. Το παραπάνω μοντέλο νευρώνα ονομάζεται μοντέλο McCulloch–Pitts, προς τιμήν του πρωτοποριακού τους έργου.

2.1.1.1 Συνάρτηση Ενεργοποίησης

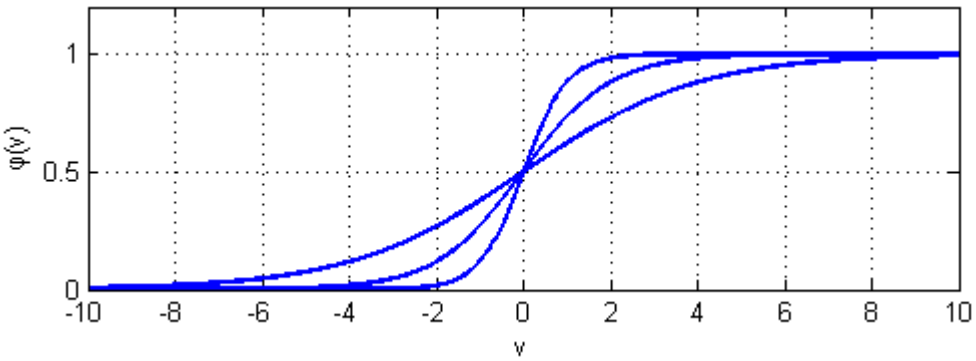
Η συνάρτηση ενεργοποίησης $\varphi(v)$ ορίζει την έξοδο ενός νευρώνα δεδομένου του τοπικού πεδίου v. Υπάρχουν δύο βασικοί τύποι συναρτήσεων ενεργοποίησης:

1. Συνάρτηση Κατωφλίου (Threshold Function): $\varphi(v) = \begin{cases} 1 \text{ εάν } v \geq a \\ 0 \text{ εάν } v < a \end{cases}$
   Σε αυτό το μοντέλο, η έξοδος ενός νευρώνα λαμβάνει τιμή 1 εάν το τοπικό πεδίο του είναι μη αρνητικό, ενώ 0 σε διαφορετική περίπτωση. Η τιμή κατωφλίου a ορίζεται από το χρήστη – μια συχνή τιμή είναι το 0.



Σχήμα 2.1.2 Γραφική αναπαράσταση της συνάρτησης κατωφλίου

2. Σιγμοειδής Συνάρτηση (Sigmoid Function):
   Είναι η πλέον κοινή μορφή συνάρτησης ενεργοποίησης που χρησιμοποιείται στα νευρωνικά δίκτυα λόγω της συνέχειάς της. Οι τιμές της εξόδου αυτής της συνάρτησης κυμαίνονται στο διάστημα από 0 ως 1 όπως και στη συνάρτηση κατωφλίου, με τη διαφορά ότι είναι διαφορίσιμη (πολύ σημαντικό χαρακτηριστικό της θεωρίας των νευρωνικών δικτύων). Η παράμερος a καθορίζει το πόσο απότομη θα είναι η κλίση της συνάρτησης. Οι σημαντικότερες σιγμοειδής συναρτήσεις είναι η λογιστική $\varphi(v) = \frac{1}{1+e^{-av}}$ και η υπερβολική εφαπτομένη $\varphi(v) = \frac{1-e^{-2v}}{1+e^{-2v}}$.



Σχήμα 2.1.3 Γραφική αναπαράσταση της λογιστικής συνάρτησης για μεταβαλλόμενη παράμετρο της κλίσης α (0.5, 1, 2). Όσο η παράμετρος α τείνει στο άπειρο, τόσο περισσότερο η γραφικής της αναπαράσταση μοιάζει με τη συνάρτηση κατωφλίου.

2.1.1.2 Στοχαστικό Μοντέλο Νευρώνα

Η συμπεριφορά του μοντέλου του νευρώνα που περιγράψαμε παραπάνω είναι καθορισμένη έτσι ώστε κάθε φορά που εισάγουμε μια συγκεκριμένη είσοδο να μας δίνει την ίδια έξοδο. Αυτό το μοντέλο νευρώνα είναι ντετερμινιστικό και χρησιμοποιείται σε μια μεγάλη οικογένεια νευρωνικών δικτύων. Ορισμένες φορές, όμως, είναι επιθυμητό να χρησιμοποιήσουμε ένα στοχαστικό μοντέλο για να βασίσουμε την ανάλυση του νευρωνικού μας δικτύου. Σε αυτές τις περιπτώσεις μπορούμε να μετατρέψουμε το μοντέλο McCulloch–Pitts σε στοχαστικό μοντέλο χρησιμοποιώντας τη συνάρτηση ενεργοποίησης με πιθανοκρατικό τρόπο.

Στο στοχαστικό μοντέλο, οι επιτρεπτές καταστάσεις ενός νευρώνα είναι δύο: "ενεργός" και "ανενεργός". Η απόφαση για τη μετάβαση από την κατάσταση "ανενεργός" στην κατάσταση "ενεργός" είναι πιθανοκρατική. Αν η κατάσταση του νευρώνα k συμβολίζεται με y_k και η πιθανότητα ενεργοποίησης συμβολίζεται με $P(v_k)$, όπου v_k είναι το τοπικό πεδίο του νευρώνα, τότε το μοντέλο αυτό μπορεί να περιγραφεί μαθηματικά ως εξής:

$$y_k = \begin{cases} \text{ενεργός με πιθανότητα } P(v_k) \\ \text{ανενεργός με πιθανότητα } 1 - P(v_k) \end{cases}$$

Μια ενδεικτική σιγμοειδής συνάρτηση για την $P(v_k)$ είναι η:

$$P(v_k) = \frac{1}{1 + e^{-v_k/T}}$$

Όπου T είναι μια ψευδοθερμοκρασία (μέγεθος δανεισμένο από τη θερμοδυναμική) η οποία χρησιμοποιείται για τον έλεγχο της στάθμης του θορύβου – στο δικό μας πεδίο την αβεβαιότητα αναφορικά με την ενεργοποίηση του νευρώνα. Όταν το $T \to 0$, ο στοχαστικός νευρώνας μετατρέπεται σε μια ντετερμινιστική μορφή – χωρίς θόρυβο – δηλαδή στο μοντέλο McCulloch–Pitts.

2.2 ΣΤΑΤΙΣΤΙΚΗ ΜΗΧΑΝΙΚΗ

Ο αριθμός των βαθμών ελευθερίας ενός συστήματος όπως τα νευρωνικά δίκτυα – όταν μιλάμε για "μεγάλα" δεδομένα – είναι τεράστιος, γεγονός που καθιστά αναγκαία τη χρήση πιθανοκρατικών μεθόδων για την περιγραφή τους. Πολλές φορές, για να γίνει πιο κατανοητός ο τρόπος λειτουργίας ενός πεδίου, χρησιμοποιούμε όρους από κάποιο άλλο επιστημονικό πεδίο. Αυτό ακριβώς κάνει η στατιστική μηχανική.

Η στατιστική μηχανική είναι ο κλάδος που μελετά τις ιδιότητες ισορροπίας συστημάτων μεγάλης κλίμακας, τα οποία αποτελούνται από στοιχεία υποκείμενα στους νόμους της μηχανικής που ισχύουν στη μικροσκοπική κλίμακα. Ο κύριος στόχος της στατιστικής μηχανικής είναι ο προσδιορισμός των θερμοδυναμικών ιδιοτήτων μεγάλων σε μέγεθος δεδομένων, ξεκινώντας όμως από την περιγραφή της κίνησης μικροσκοπικών στοιχείων όπως τα άτομα και τα ηλεκτρόνια. Στη θεωρία της στατιστικής μηχανικής, η έννοια της εντροπίας παίζει πολύ σημαντικό ρόλο. Η εντροπία ενός συστήματος συμβολίζει το πόσο τάξη υπάρχει σε αυτό – όσο μεγαλύτερη η εντροπία, τόσο λιγότερη τάξη υπάρχει στο σύστημα.

Θεωρούμε ένα φυσικό σύστημα με πολλούς βαθμούς ελευθερίας, το οποίο μπορεί να βρίσκεται σε οποιαδήποτε συγκεκριμένη κατάσταση – από ένα μεγάλο αριθμό πιθανών καταστάσεων. Η πιθανότητα πραγματοποίησης της κατάστασης i ενός στοχαστικού συστήματος συμβολίζεται με p_i και έχει τις εξής ιδιότητες:

$$p_i \geq 0 \ για \ όλα \ τα \ i$$

και

$$\sum_i p_i = 1$$

Επίσης η ενέργεια του συστήματος όταν αυτό βρίσκεται στην κατάσταση i συμβολίζεται με E_i. Η στατιστική μηχανική μας λέει ότι όταν το σύστημα είναι σε θερμική ισορροπία με το περιβάλλον του, τότε η κατάσταση i πραγματοποιείται με πιθανότητα:

$$p_i = \frac{1}{Z} e^{-\frac{E_i}{T}}$$

όπου T είναι ένα μέγεθος ψευδοθερμοκρασίας το οποίο ελέγχει τις θερμικές διακυμάνσεις του συστήματος, και το μέγεθος κανονικοποίησης Z είναι η συνάρτηση διαμέρισης. Η κατανομή πιθανοτήτων της παραπάνω εξίσωσης αποκαλείται κανονική κατανομή ή κατανομή Gibbs. Χρησιμοποιώντας τις δύο τελευταίες εξισώσεις, μπορούμε να δούμε ότι:

$$Z = \sum_i e^{-\frac{E_i}{T}}$$

Σύμφωνα με την κατανομή Gibbs, είναι πιο πιθανό να πραγματοποιηθούν οι καταστάσεις χαμηλής ενέργειας από αυτές που διακρίνονται από υψηλή ενέργεια. Επίσης, η κατανομή πιθανοτήτων επικεντρώνεται σε ένα μικρότερο υποσύνολο καταστάσεων χαμηλής ενέργειας, καθώς η θερμοκρασία μειώνεται.

Η ελεύθερη ενέργεια (free energy) ενός φυσικού συστήματος ορίζεται βάσει της συνάρτησης διαμέρισης Z

$$F = -T \ \log Z$$

Η μέση ενέργεια (average energy) του συστήματος ορίζεται ως εξής:

$$\bar{E} = \sum_i p_i E_i$$

Αν $H = -\sum_i p_i \log p_i$ είναι η εντροπία του συστήματος τότε μπορούμε να αναδιατυπώσουμε τη συνάρτηση της ελεύθερης ενέργειας σε συνάρτηση με τη μέση ενέργεια ως εξής:

$$\boldsymbol{F = \bar{E} - TH}$$

Στατιστική Φυσική	Συνδυαστική Βελτιστοποίηση
Δείγματα	Στιγμιότυπο προβλήματος
Κατάσταση	Διαμόρφωση
Ενέργεια	Συνάρτηση κόστους
Θερμοκρασία	Παράμετρος ελέγχου
Ενέργεια τελικής κατάστασης	Ελάχιστο κόστος
Διαμόρφωση τελικής κατάστασης	Βέλτιστη διαμόρφωση

Πίνακας 2.2.1 Αντιστοιχεία μεταξύ Στατιστικής Φυσικής και Συνδυαστικής Βελτιστοποίησης

2.2.1 Προσομοιωμένη Ανόπτηση

Σε πολλές περιπτώσεις στα νευρωνικά δίκτυα προσπαθούμε να βελτιστοποιήσουμε μια συνάρτηση κόστους – όπως στα Multi-layer Perceptron (MLP) – ή ενέργειας. Στις περισσότερες περιπτώσεις χρησιμοποιούμε ντετερμινιστικές τεχνικές όπως η κλίση της συνάρτησης της ενέργειας, χρησιμοποιώντας την παράγωγο.

Αυτές οι τεχνικές όμως παγιδεύονται σε τοπικά ελάχιστα. Χρειαζόμαστε μια τεχνική που να έχει τη δυνατότητα να ξεπεράσει τα τοπικά ελάχιστα. Χρειαζόμαστε δηλαδή μια τεχνική, η οποία αντί σε κάθε βήμα της να προχωράει πάντα προς την κατεύθυνση που ελαχιστοποιεί τη συνάρτηση της ενέργειας, να προχωράει προς τα εκεί τις περισσότερες φορές. Μια τέτοια τεχνική είναι η προσομοιωμένη ανόπτηση.

Η τεχνική αυτή προέρχεται από τα μεταλλουργεία, όπου για να φτάσουμε ένα μέταλλο στην κατάσταση ελάχιστης ενέργειας (κρυσταλλική δομή), το θερμαίνουμε κατάλληλα σε μια αρχική θερμοκρασία, ώστε τα άτομα να κινούνται τυχαία, και στη συνέχεια το ψήχουμε σταδιακά μέχρι να φτάσει στην ελάχιστη ενέργεια. Πώς ξέρουμε όμως κάθε πότε πρέπει να το ψύξουμε; Σε κάθε στάδιο ψύξης, το αφήνουμε να φτάσει σε θερμική ισορροπία.

Αντίστοιχη κατάσταση επικρατεί και στα νευρωνικά δίκτυα. Έχουμε ένα μεγάλο πλήθος νευρώνων που με τις παραμέτρους τους διαμορφώνουν τη συνάρτηση της ενέργειας. Μπορούμε να ξεφύγουμε από τοπικά ελάχιστα, αφού επιτρέπονται κινήσεις προς καταστάσεις που αυξάνουν την ενέργεια. Σε υψηλές θερμοκρασίες βρίσκουμε τα γενικά χαρακτηριστικά της συνάρτησης της ενέργειας αφήνοντας το σύστημα να κάνει περισσότερες "ελεύθερες" κινήσεις. Όσο ελαττώνεται όμως η θερμοκρασία, αυτές οι κινήσεις μειώνονται με αποτέλεσμα να ανιχνεύονται περισσότερες λεπτομέρειες στις χαμηλές θερμοκρασίες.

Πώς μπορούμε, όμως, να προσομοιώσουμε αυτή τη διαδικασία σε ένα νευρωνικό δίκτυο; Χρησιμοποιεί το συνδυασμό δύο σχετιζόμενων στοιχείων:

1. Ένα χρονοδιάγραμμα που καθορίζει το ρυθμό με τον οποίο μειώνεται η θερμοκρασία.
2. Έναν αλγόριθμο που βρίσκει την κατανομή της κατάστασης ισορροπίας σε κάθε νέα θερμοκρασία που προβλέπει το διάγραμμα, χρησιμοποιώντας την τελική κατάσταση του συστήματος στην προηγούμενη θερμοκρασία ως σημείο έναρξης για την επόμενη θερμοκρασία.

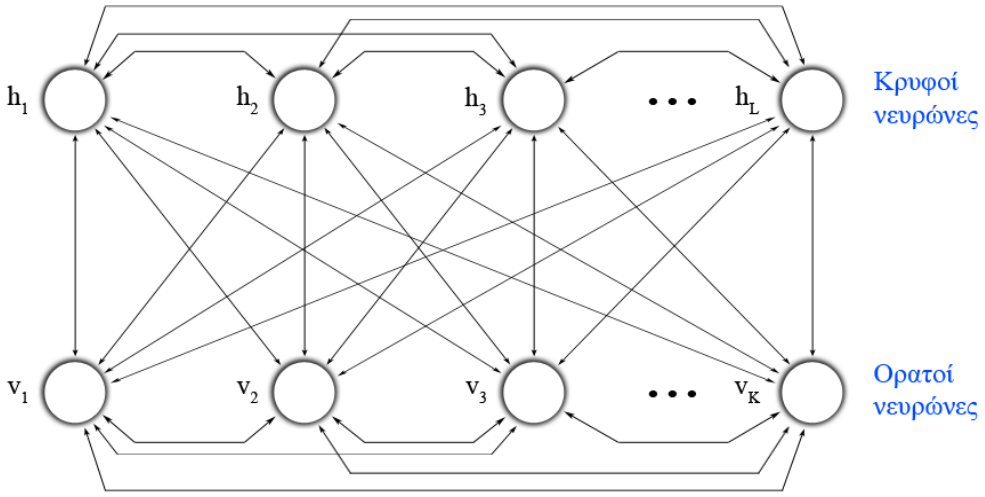
2.2.1.1 Χρονοδιάγραμμα Ανόπτησης

Σε αυτό το κομμάτι πρέπει να βρούμε έναν τρόπο να μειώνουμε τη θερμοκρασία συνεχώς, έτσι ώστε να φτάσουμε στην "κρυσταλλική" μας δομή. Το *χρονοδιάγραμμα ανόπτησης* καθορίζει μια πεπερασμένη ακολουθία τιμών της θερμοκρασίας και έναν πεπερασμένο αριθμό μεταβάσεων που επιχειρούνται σε κάθε τιμή. Οι παράμετροι ενδιαφέροντος είναι τρεις:

1. Η *αρχική τιμή της θερμοκρασίας* T_0, η οποία πρέπει να οριστεί όσο υψηλή πρέπει, ώστε να μπορούν να είναι αποδεκτές όλες οι προτεινόμενες μεταβάσεις.
2. Η *μείωση της θερμοκρασίας* a, είναι μια παράμετρος η οποία μειώνει σταδιακά τη θερμοκρασία. Ουσιαστικά είναι η διαδικασία ψύξης, η οποία εκτελείται εκθετικά:
$$T_k = aT_{k-1}, \qquad k = 1, 2, \dots$$
Όπου a μια θετική σταθερά μικρότερη από το 1, αλλά πολύ κοντά σε αυτό. Σε κάθε θερμοκρασία επιχειρούνται αρκετές μεταβάσεις, έτσι ώστε να έχουμε κατά μέσο όρο 10 αποδεκτές μεταβάσεις σε κάθε πείραμα.
3. Η *τελική τιμή της θερμοκρασίας*, είναι η θερμοκρασία στη οποία έχει σταματήσει η ανόπτηση. Αν σε τρεις διαδοχικές θερμοκρασίες δεν επιτευχθεί ο επιθυμητός αριθμός αποδεκτών μεταβάσεων, η ανόπτηση σταματά και το σύστημα σταθεροποιείται.

2.2.2 Μηχανή Boltzmann

Η μηχανή Boltzmann (Boltzmann Machine – BM) είναι μια στοχαστική, δυαδική μηχανή η οποία χρησιμοποιεί την τεχνολογία της προσομοιωμένης ανόπτησης που περιγράφηκε παραπάνω και μπορεί να χρησιμοποιηθεί για την ανακάλυψη υποκείμενων κανονικοτήτων των δεδομένων εκπαίδευσης. Με τον όρο στοχαστική μηχανή εννοούμε ένα νευρωνικό δίκτυο με νευρώνες όμοιους με αυτούς του στοχαστικού μοντέλου. Οι καταστάσεις των νευρώνων αυτού του νευρωνικού δικτύου είναι δύο: «ενεργός» και «ανενεργός». Μαθηματικά συνήθως εκφράζονται ως -1 και 1 ή 0 και 1, όπου η υψηλότερη τιμή σημαίνει «ενεργός» νευρώνας ενώ η χαμηλότερη «ανενεργός». Η συναπτικές συνδέσεις μεταξύ των νευρώνων είναι συμμετρικές και το νευρωνικό δίκτυο είναι πλήρως συνδεδεμένο. Στο Σχ. 2.2.1 φαίνεται η δομή ενός τέτοιου νευρωνικού δικτύου.



Σχήμα 2.2.1 Αρχιτεκτονικός γράφος μιας μηχανής Boltzmann. Όλες οι συνάψεις είναι συμμετρικές.

Βλέπουμε ότι μια μηχανή Boltzmann αποτελείται από δύο λειτουργικές ομάδες στοχαστικών νευρώνων:

- Τους *ορατούς νευρώνες* (*εισόδου* και *εξόδου*), οι οποίοι παρέχουν έναν ενδιάμεσο μηχανισμό ώστε το δίκτυο να έχει τη δυνατότητα να επικοινωνεί με το περιβάλλον στο οποίο λειτουργεί (interface)
- Τους *κρυφούς νευρώνες*, οι οποίοι χρησιμοποιούνται για να εξηγήσουν τους υποκείμενους περιορισμούς που περιέχονται στα διανύσματα εισόδου. Αυτή η εργασία επιτυγχάνεται καταγράφοντας στατιστικές συσχετίσεις υψηλότερης τάξης στα διανύσματα πρόσδεσης.

Οι ορατοί νευρώνες χωρίζονται σε νευρώνες εισόδου και εξόδου, το πλήθος των οποίων επιλέγει ο κατασκευαστής της μηχανής.

Η εκπαίδευση γίνεται σε δύο φάσεις. Στην *πρώτη φάση*, οι καταστάσεις των ορατών νευρώνων (εισόδου και εξόδου) κλειδώνουν, ενώ οι καταστάσεις των κρυφών νευρώνων μεταβάλλονται χρησιμοποιώντας την τεχνική της προσομοιωμένης ανόπτησης. Στη *δεύτερη φάση*, κλειδώνουν μόνο οι καταστάσεις των νευρώνων εισόδου, ενώ οι κρυφοί νευρώνες και οι νευρώνες εξόδου μεταβάλλονται με το ίδιο τρόπο.

2.2.2.1 Αλγόριθμος Μάθησης Boltzmann

Ο αλγόριθμος εκπαίδευσης μια μηχανής Boltzmann φαίνεται αναλυτικά παρακάτω:

Φάση 0: Αρχικοποίηση

Δώσε στα βάρη w_{ji} τυχαίες, ομοιόμορφα κατανεμημένες τιμές στο διάστημα $[-\varepsilon, +\varepsilon]$ όπου το ε συνήθως είναι 0.5 ή 1.

Φάση 1: Αύξηση των βαρών

Βήμα 1. Κλείδωσε τους ορατούς νευρώνες στις αντίστοιχες σωστές τιμές τους σύμφωνα με τα δεδομένα εκπαίδευσης.

Βήμα 2. Άφησε το δίκτυο να «τρέξει» ελεύθερα. Υπολόγισε την μεταβολή της ενέργειας της κατάστασης j σύμφωνα με τον τύπο

$$\Delta E_j = \sum_{i \neq j} w_{ji}\xi_i - \theta_j$$

Και στη συνέχεια πήδηξε σε μια κατάσταση χαμηλότερης ενέργειας σύμφωνα με τον πιθανοτικό κανόνα

$$\xi_j = \begin{cases} +1 \ με \ πιθανότητα \ p_j \\ -1 \ με \ πιθανότητα \ 1 - p_j \end{cases}$$

όπου

$$p_j = \frac{1}{1 + e^{-\Delta E_j / T}}$$

Μείωσε αργά τη θερμοκρασία T (δηλαδή εκτέλεσε προσομοιωμένη ανόπτηση για μια πεπερασμένη ακολουθία μειωμένων θερμοκρασιών) μέχρις ότου η έξοδος να πάει στη μόνιμη κατάσταση – τη θερμική ισορροπία.

Βήμα 3. Αύξησε το συναπτικό βάρος μεταξύ δύο οποιονδήποτε νευρώνων των οποίων η κατάσταση είναι «ενεργός». Για να γίνει αυτό, χρειαζόμαστε την τελική θερμοκρασία του προηγούμενου βήματος, τις συσχετίσεις:

$$\rho_{ji}^{+} = \langle \xi_j \xi_i \rangle^{+}, \qquad j \neq i, \qquad j = 1, 2, \dots, N$$

όπου το «+» αναφέρεται στη συνθήκη κλειδώματος και την ανανέωση των βαρών σύμφωνα με τον κανόνα:

$$\Delta w_{ji} = +\gamma \rho_{ji}^{+}, \qquad j \neq i$$

όπου γ είναι η παράμετρος του ρυθμού μάθησης.

Φάση 2: Μείωση των βαρών

Βήμα 1. Κλείδωσε μόνο τους νευρώνες εισόδου, αφήνοντας τους νευρώνες εξόδου και τους κρυφούς νευρώνες να «τρέχουν» ελεύθεροι.

Βήμα 2. Άφησε το δίκτυο να πάει σε θερμική ισορροπία όπως στο Βήμα 2 της Φάσης 1.

Βήμα 3. Υπολόγισε (εκτίμησε) τις συσχετίσεις

$$\rho_{ji}^{-} = \langle \xi_j \xi_i \rangle^{-}, \qquad j \neq i, \qquad j = 1, 2, \dots, N$$

όπου το «–» αναφέρεται στη συνθήκη κατά την οποία οι κρυμμένοι νευρώνες και οι νευρώνες εξόδου «τρέχουν» ελεύθερα. Μείωσε τα βάρη μεταξύ δύο οποιονδήποτε νευρώνων οι οποίοι είναι σε κατάσταση «ενεργός», με βάση τον κανόνα:

$$\Delta w_{ji} = -\gamma \rho_{ji}^{-}, \qquad j \neq i$$

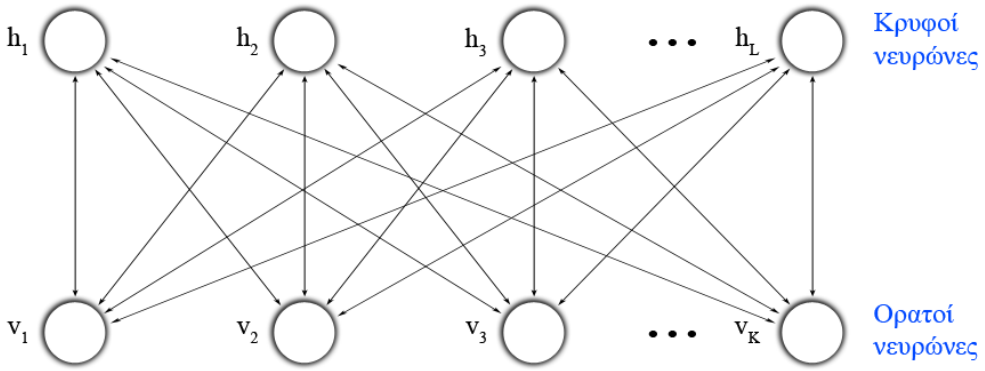
Επαναλαμβάνουμε τις φάσεις 1 και 2 μέχρι να επιτευχθεί σύγκλιση – μέχρι, δηλαδή, τα βάρη να φτάσουν στη μόνιμη κατάσταση.

2.2.3 Περιορισμένη Μηχανή Boltzmann

Μια υποκατηγορία των μηχανών Boltzmann θα χρησιμοποιηθούν σε επόμενο κεφάλαιο για τη δημιουργία ενός δικτύου πεποίθησης (Belief Network). Η δομή αυτών των μηχανών πρωτοπαρουσιάστηκε από τον Smolensky (1986)· στην εργασία του αποκαλούνται "harmonium". Αργότερα το harmonium μετονομάστηκε σε Περιορισμένη Μηχανή Boltzmann (Restricted Boltzmann Machine - RBM) λόγο της διαφοράς τους από τις κλασικές μηχανές Boltzmann ως προς τη συνδεσιμότητα των νευρώνων τους. Οι νευρώνες των RBM είναι πλήρως συνδεδεμένοι ανάμεσα στα επίπεδα, αλλά όχι μέσα στο ίδιο επίπεδο· δηλαδή, σε κάθε επίπεδο οι τιμές των νευρώνων είναι ανεξάρτητες μεταξύ τους.

Ένα RBM μαθαίνει ένα επίπεδο από κρυφά χαρακτηριστικά μεταβλητών (νευρώνες $h_1,\ h_2,\ h_3, \dots h_L$ στο Σχ. 2.2.2), τα οποία μπορεί να είναι λιγότερα σε αριθμό από αυτά του πραγματικού συνόλου χαρακτηριστικών αλλά πιο ισχυρά, καθώς παρέχουν συμπαγή αναπαράσταση των υποκείμενων μοτίβων και της δομής της εισόδου. Είναι ισχυρότερο από την ομαδοποίηση, αφού χρησιμοποιώντας L κρυφούς νευρώνες ένα RBM μπορεί να συλλάβει 2^L περιοχές χώρων εισόδου. Μια συνηθισμένη ομαδοποίηση απαιτεί $O(2^L)$ παραμέτρους και παραδείγματα για να συλλάβει τόσο μεγάλη πολυπλοκότητα.

Μπορούμε να φανταστούμε το RBM σαν ένα διμερή γράφο με δύο σύνολα κορυφών: ορατές και κρυφές, όπως φαίνεται στο Σχ. 2.2.2, όπου $h_i, i = 1, 2, 3, \dots, L$ είναι οι κρυφές κορυφές και $v_j, j = 1, 2, 3, \dots, K$ οι ορατές κορυφές (όπως αυτές των δεδομένων εκπαίδευσης). Οι πληροφορίες ρέουν προς όλες τις κατευθύνσεις ανάμεσα στις ομάδες κορυφών h και v, και τα βάρη δεν είναι κατευθυνόμενα.



Σχήμα 2.2.2 Αρχιτεκτονικός γράφος ενός RBM. Κάθε ακμή έχει το αντίστοιχο βάρος της w_{ji}.

Το RBM εκπαιδεύεται μέσω της μετρικής *contrastive divergence*, που είναι όμοια με τη *gradient descent* αλλά δεν ακολουθεί τη λογαριθμική πιθανότητα. Οι κρυφοί νευρώνες και τα βάρη ενημερώνονται εναλλακτικά.

Με ένα πίνακα $L \times N$ ο οποίος αντιπροσωπεύει τις εισόδους **V**, ένα πίνακα $K \times N$ που αντιπροσωπεύει τις εξόδους **H**, και ένα πίνακα $L \times K$ που αντιπροσωπεύει τα βάρη **W**, διεξάγεται το ακόλουθο:

1. Βήμα ενημέρωσης. Για κάθε κρυφό νευρώνα $i = 1, \dots, K$:
    1. Υπολόγισε την ενέργεια ενεργοποίησης a_k:
    $$a_i = \sum_{j=1}^{L} w_{ji}\, x_j$$
    2. Ανάθεσε την τιμή 1 στο h_i με πιθανότητα $\sigma(a_i)$ (αλλιώς την τιμή 0) όπου $\sigma(\cdot)$ είναι η σιγμοειδής συνάρτηση.
    3. Για κάθε ακμή, υπολόγισε τη θετική της ενέργεια (αν και τα δύο άκρα της είναι ενεργά):

$$e_{ji}^{+} = x_j h_i$$

2. Βήμα ανακατασκευής. Επανάλαβε το βήμα 1, αποκτώντας την κατάσταση για κάθε ορατό νευρώνα $j = 1, \ldots, d$. Μετά ενημέρωσε τους κρυφούς νευρώνες και υπολόγισε την αρνητική ενέργεια e_{jk}^{-} της κάθε ακμής με τον ίδιο τρόπο.
3. Ενημέρωσε το βάρος της κάθε ακμής:

$$w_{jk} \leftarrow w_{jk} + a(e_{jk}^{+} - e_{jk}^{-})$$

όπου α είναι ο ρυθμός μάθησης και $(e_{jk}^{+} - e_{jk}^{-})$ είναι η *contrastive divergence*.

Παρατηρούμε ότι η διαδικασία μάθησης ενός RBM δεν υφίσταται διαδικασία ανόπτησης. Έτσι ο αλγόριθμος μάθησης δεν χρησιμοποιεί τη θερμοκρασία ως παράμετρο. Παρόλα αυτά, η μάθηση ενός RBM έχει ελάχιστες διαφορές από τη μάθηση μιας κλασικής μηχανής Boltzmann. Επίσης, παρατηρούμε ότι η ενημέρωση της κάθε κρυφής κατάστασης γίνεται ανεξάρτητα από τις υπόλοιπες. Αυτό μας επιτρέπει να ενημερώνουμε όλες τις κρυφές καταστάσεις παράλληλα. Το ίδιο ισχύει και τις ορατές στην επόμενη φάση.

2.2.4 Δίκτυα Πεποίθησης Μεγάλου Βάθους

Η εκπαίδευση των μηχανών Boltzmann αποδείχτηκε χρονοβόρα διαδικασία λόγο του χρόνου που χρειάζεται μια τέτοια μηχανή για να έρθει σε κατανομή ισορροπίας, πράγμα το οποίο είναι συχνότερο όταν οι ορατοί νευρώνες είναι κλειδωμένοι. Επίσης, παρατηρήθηκε ότι η κατάσταση επιβαρύνεται όταν το πλήθος των κρυφών νευρώνων είναι μεγάλο.

Παρόλα αυτά, το ενδιαφέρον για μια μηχανή που έχει τη δυνατότητα να μαθαίνει κατανομές πιθανοτήτων από δυαδικά διανύσματα, όπως μια κλασική μηχανή Boltzmann δεν χάθηκε ποτέ. Έτσι, το 2006, ο Hinton κ.ά., επινόησαν ένα νέο τρόπο μάθησης, ο οποίος αφενός να έχει τις δυνατότητες μιας μηχανής Boltzmann και αφετέρου θα μπορεί να μπορεί να εκτελεί τις ακόλουθες λειτουργίες:

- Να αγνοεί την αρνητική φάση της μηχανής Boltzmann, η οποία είναι υπεύθυνη για τον αυξημένο χρόνο της εκπαίδευσης και να χρησιμοποιεί ένα άλλο μέσο για τον έλεγχο στη διαδικασία της μάθησης.
- Να λειτουργεί αποτελεσματικά σε πυκνά συνδεδεμένα δίκτυα.

Τα δίκτυα αυτά ονομάστηκαν Δίκτυα Πεποίθησης Μεγάλου Βάθους (Deep Belief Networks – DBNs) και είναι βασισμένα στη δομή των RBM. Τα DBN αποτελούνται από πολλά επίπεδα, το καθένα από τα οποία είναι ένα RBM (Σχ. 2.2.3).

Η εκπαίδευση ενός δικτύου DBN διεξάγεται επίπεδο προς επίπεδο, ως εξής (Hinton, κ.ά., 2006· Hinton, 2007):

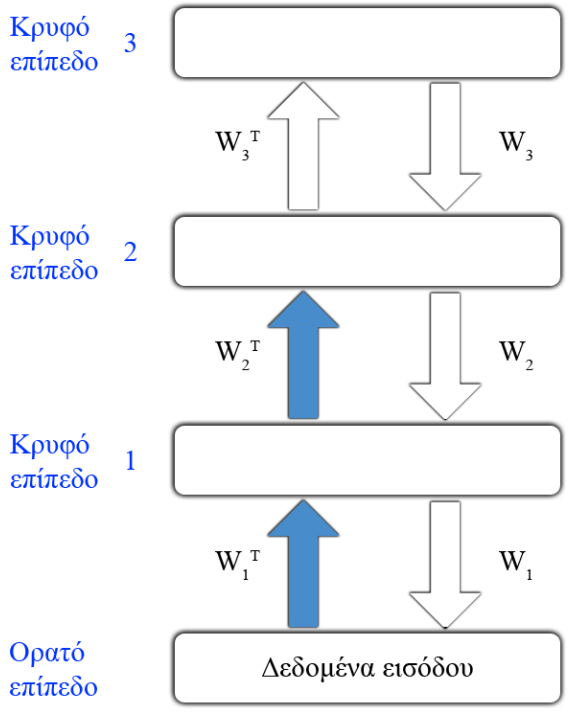
1. Μια πρώτη μηχανή RBM εκπαιδεύεται άμεσα στα δεδομένα εισόδου, δίνοντας έτσι στους στοχαστικούς νευρώνες του κρυφού επιπέδου της μηχανής RBM τη δυνατότητα να καταγράψουν τα σημαντικά χαρακτηριστικά των δεδομένων εισόδου. Από εδώ και στο εξής θα αποκαλούμε αυτό το κρυφό επίπεδο *πρώτο κρυφό επίπεδο* του δικτύου DBN
2. Οι ενεργοποιήσεις που σχετίζονται με τα χαρακτηριστικά στα οποία έχει εκπαιδευτεί η μηχανή αντιμετωπίζονται στη συνέχεια ως "δεδομένα εισόδου", τα οποία χρησιμοποιούνται για την εκπαίδευση μιας δεύτερης μηχανής RBM.

Θα μπορούσαμε να πούμε ότι η διαδικασία μάθησης που μόλις περιγράψαμε μαθαίνει τα "χαρακτηριστικά των χαρακτηριστικών".

3. Η διαδικασία μάθησης, με την οποία το δίκτυο μαθαίνει τα χαρακτηριστικά των χαρακτηριστικών, συνεχίζεται μέχρι να εκπαιδευτεί ένας προκαθορισμένος αριθμός κρυφών επιπέδων του δικτύου DBN.

Είναι σημαντικό να αναφερθεί ότι κάθε φορά που προστίθεται ένα νέο επίπεδο χαρακτηριστικών στο δίκτυο, βελτιώνεται το (μεταβαλλόμενο) κάτω όριο λογαριθμικής συνάρτησης πιθανοφάνειας των αρχικών δεδομένων εκπαίδευσης (Hinton κ.ά., 2006). Επίσης, ο Hinton πρότεινε η εκπαίδευση των RBM εσωτερικά του DBN να γίνεται σε δύο εποχές, καθώς αυτές είναι αρκετές να δώσουν ένα πολύ καλό – αλλά όχι τέλειο – αποτέλεσμα γλιτώνοντας πολύ χρόνο συγκριτικά με τον αριθμό των εποχών που χρειάζονται για να έρθει το RBM σε θερμική ισορροπία.

Στο Σχ. 2.2.3 αναπαριστάται ένα DBN αφού εκπαιδευτούν τρία κρυφά επίπεδά του. Τα βέλη με κατεύθυνση προς τα πάνω δείχνουν τα βάρη που υπολογίζονται ως αποτέλεσμα της μάθησης των "χαρακτηριστικών των χαρακτηριστικών". Τα βάρη αυτά βοηθούν στον υπολογισμό των δυαδικών τιμών των χαρακτηριστικών σε κάθε κρυφό επίπεδο του δικτύου όταν εφαρμόζεται ένα διάνυσμα δεδομένων στο ορατό επίπεδο.



Σχήμα 2.2.3 Αρχιτεκτονική δομή ενός DBN αφού εκπαιδευτούν τρία κρυφά επίπεδά του. Το κάθε κρυφό επίπεδο είναι το αντίστοιχο κρυφό επίπεδο κάποιου RBM.

Τι κάνει όμως τα DBN τόσο ξεχωριστά; Τα DBN λειτουργούν με δύο τρόπους μετά την εκπαίδευσή τους:

1. *Ταξινόμηση*: Εφαρμόζοντας τα δεδομένα στο κατώτερο επίπεδό τους, μπορούν να σου δώσουν μια ή περισσότερες ετικέτες (ανάλογα με τη χρήση).

2. *Παραγωγή δεδομένων*: Εφαρμόζοντας μία ή περισσότερες ετικέτες στο κορυφαίο επίπεδο, μπορούν να παράγουν δεδομένα όμοια με αυτά των δεδομένων εκπαίδευσης. Λόγο των στοχαστικών νευρώνων τους κάθε φορά θα παράγουν διαφορετικά δεδομένα που θα σχετίζονται με τις ετικέτες που διάλεξε ο χρήστης.

Αυτό συμβαίνει καθώς ανάμεσα στα δύο τελευταία επίπεδα (επίπεδα 2 και 3 στο Σχ. 2.2.3), οι συνδέσεις και προς τις δύο κατευθύνσεις – δηλαδή οι συνάψεις της "κορυφαίας" μηχανής RBM – ανήκουν στο *παραγωγικό μοντέλο*. Στο Σχ. 2.2.3 το παραγωγικό μοντέλο απεικονίζεται με λευκά βέλη. Αυτό κάνει την κορυφαία μηχανή RBM να παίζει το ρόλο μιας *διμερούς συσχετιστικής μνήμης*.

Πιο συγκεκριμένα, κατά την εκπαίδευση "από κάτω προς τα πάνω", η κορυφαία RBM μαθαίνει από το κρυφό επίπεδο της αμέσως προηγούμενής της. Κατά την παραγωγή δεδομένων "από πάνω προς τα κάτω", η κορυφαία RBM παίρνει το ρόλο ενός εναρκτήριου μηχανισμού της παραγωγικής μοντελοποίησης.

Για τη λειτουργία της μάθησης δεν απαιτείται η παραγωγή των δεδομένων, πράγμα πολύ θετικό αφού η παραγωγή δεδομένων είναι μια χρονοβόρα διαδικασία, κυρίως επειδή η κορυφαία RBM πρέπει να φτάσει σε κατάσταση ισορροπίας.

2.3 ΔΕΔΟΜΕΝΑ ΠΟΛΛΑΠΛΩΝ ΕΤΙΚΕΤΩΝ

Ένα μεγάλο ερευνητικό κομμάτι της μάθησης με επίβλεψη ασχολείται με την ταξινόμηση δεδομένων μιας ετικέτας, όπου τα δεδομένα εκπαίδευσης συνοδεύονται με μια μόνο ετικέτα λ από ένα σύνολο L που αποτελείται από πολλές διαφορετικές ετικέτες. Σε πολλές εφαρμογές, όμως, τα δεδομένα εκπαίδευσης αντιστοιχούν σε παραπάνω από μία ετικέτες, οι οποίες αποτελούν ένα σύνολο Y ετικετών για το οποίο ισχύει η σχέση $Y \subseteq L$. Αυτά τα δεδομένα ονομάζονται *δεδομένα πολλαπλών ετικετών* (Multi-label Data).

Δεδομένα κειμένου, όπως τα έγγραφα και οι ιστοσελίδες, είναι χαρακτηριστικά παραδείγματα *multi-label* δεδομένων αφού συνήθως συνδέονται με περισσότερες από μία ετικέτες. Για παράδειγμα, ένα άρθρο εφημερίδας που μιλάει για τις αντιδράσεις της Χριστιανικής εκκλησίας σχετικά με την προβολή της ταινίας «Κώδικας Da Vinci» μπορεί να σημειωθεί με την ετικέτα "Θρησκεία" αλλά και με την "Ταινίες".

2.3.1 Μάθηση

Υπάρχουν δύο μοντέλα αναπαράστασης των ετικετών για να καταφέρουμε να εκπαιδεύσουμε με επίβλεψη ένα σύστημα με *multi-label* δεδομένα: η *multi-label ταξινόμηση* (*multi-label classification* – MLC) και η *κατάταξη ετικετών* (*label ranking* – LR).

Το MLC μοντέλο ασχολείται με τη μάθηση ενός μοντέλου που παράγει μια δυαδική αναπαράσταση του συνόλου των ετικετών η οποία τις κατατάσσει σε σχετικές και μη σχετικές ως προς το στιγμιότυπο των δεδομένων εισόδου. Από την άλλη, το LR μοντέλο ασχολείται με τη μάθηση ενός μοντέλου που παράγει μια κατάταξη των ετικετών του συνόλου σύμφωνα με τη σχετικότητά τους ως προς το στιγμιότυπο των δεδομένων εισόδου. Το μοντέλο LR μπορεί να εκπαιδευτεί με δεδομένα μίας ετικέτας, το ίδιο καλά με αυτά των πολλών ετικετών.

Στην πραγματικότητα, αυτό που χρειαζόμαστε είναι ένα μοντέλο που θα κατατάσσει τις ετικέτες σε σχετικές και μη σχετικές σύμφωνα με το στιγμιότυπο των δεδομένων εισόδου, αλλά παράλληλα θα ταξινομεί τις σχετικές σύμφωνα με το πόσο σχετικές είναι. Ένα τέτοιο μοντέλο ονομάζεται *multi-label κατάταξη* (*multi-label ranking* – MLR) και αποτελεί μια πολύ ενδιαφέρουσα γενίκευση των μοντέλων MLC και LR.

Παράδειγμα	Χαρακτηριστικά	Σύνολο Ετικετών
1	$\mathbf{x}_1$	$\{\lambda_1, \lambda_4\}$
2	$\mathbf{x}_2$	$\{\lambda_3, \lambda_4\}$
3	$\mathbf{x}_3$	$\{\lambda_1\}$
4	$\mathbf{x}_4$	$\{\lambda_2, \lambda_3, \lambda_4\}$

Πίνακας 2.3.1 Παράδειγμα multi-label συνόλου δεδομένων

Οι αλγόριθμοι που μπορούν να χρησιμοποιηθούν για τη διαχείριση multi-label προβλημάτων μπορούν χωριστούν σε δύο ομάδες: *μετασχηματισμός προβλήματος* (*problem transformation*) και *προσαρμογή αλγορίθμου* (*algorithm adaptation*). Για την επεξήγηση των αλγορίθμων αυτών θα χρησιμοποιήσουμε την εξής ορολογία: $L = \{\lambda_j : j = 1, \dots, q\}$ είναι το σύνολο των ξεχωριστών ετικετών σε μια multi-label

διεργασία ενώ $D = \{(\mathbf{x}_i, \mathbf{Y}_i), i = 1, \ldots, m\}$ είναι ένα multi-label σύνολο δεδομένων εκπαίδευσης, όπου $\mathbf{x}_i$ είναι το διάνυσμα των χαρακτηριστικών και $\mathbf{Y}_i \subseteq L$ το σύνολο των ετικετών που αντιστοιχούν στο παράδειγμα i.

2.3.1.1 Μέθοδοι Μετασχηματισμού Προβλήματος

Οι αλγόριθμοι της ομάδας αυτής, μετατρέπουν το σύνολο δεδομένων εκπαίδευσης από multi-label σε single-label και δίνουν τη δυνατότητα στο χειριστή του προβλήματος να τρέξει έναν single-label αλγόριθμο εκπαίδευσης για την επίλυσή του. Αφού εκπαιδεύσουμε τη μηχανή με το παραγόμενο από το μετασχηματισμό σύνολο δεδομένων εκπαίδευσης, περνώντας από το σύστημα το σύνολο δεδομένων δοκιμής ως έξοδο θα πάρουμε την πιθανότητα του κάθε παραδείγματος του συνόλου να σχετίζεται με μια ετικέτα. Αυτό μπορεί χρησιμοποιηθεί για τη δημιουργία της κατάταξης (LR) των ετικετών.

Υπάρχουν διάφορες απλές μέθοδοι μετασχηματισμού δεδομένων, όπως οι μέθοδοι *copy*, *copy-weight*, η οικογένεια μεθόδων *select* (*select-max*, *select-min*, *select-random*, *ignore*), η *label powerset* (LP) και η *binary relevance* (BR), καθώς και πιο σύνθετες, όπως οι *pruned problem transformation* (PPT) (Read, 2088), *random k-labelsets* (RAkEL) (Tsoumakas και Vlahavas, 2007), *ranking by pairwise comparison* (RPC) (Hüllermeier et. al, 2008) και η INSDIF (Zhang και Zhou, 2007). Για το δεδομένα του Πίνακα 2.3.1, ο Πίνακας 2.3.2 δείχνει τους μετασχηματισμούς κάποιων από τις παραπάνω μεθόδους.

Παρ.	Ετικέτα	Βάρος
1a	λ_1	0,50
1b	λ_4	0,50
2a	λ_3	0,50
2b	λ_4	0,50
3	λ_1	1,00
4a	λ_2	0,33
4b	λ_3	0,33
4c	λ_4	0,33

(a)

Παρ.	Ετικέτα
1	λ_4
2	λ_4
3	λ_1
4	λ_4

(b)

Παρ.	Ετικέτα
1	λ_1
2	λ_3
3	λ_1
4	λ_2

(c)

Παρ.	Ετικέτα
3	λ_1

(d)

Πίνακας 2.3.2 Μετασχηματισμός των δεδομένων του Πίνακα 2.3.1 χρησιμοποιώντας τις μεθόδους (a) *copy-weight*, (b) *select-max*, (c) *select-min* και (d) *ignore*.

Η πιο δημοφιλής μέθοδος μετασχηματισμού προβλήματος είναι η BR. Αυτή μετατρέπει τα δεδομένα εκπαίδευσης σε $q = |L|$ single-label δεδομένα εκπαίδευσης D_{λ_j}, $j = 1, \ldots, q$, ένα για κάθε ετικέτα του συνόλου L. Το κάθε D_{λ_j} σύνολο δεδομένων αποτελείται από όλα τα παραδείγματα του πρωτότυπου συνόλου, αλλά έχει μια θετική ετικέτα αν το σύνολο περιέχει την ετικέτα λ_j αλλιώς έχει αρνητική. Έτσι δημιουργούμε μια μηχανή για το κάθε σύνολο. Όταν θέλουμε να προβλέψουμε τις ετικέτες ενός καινούριου παραδείγματος, το περνάμε από όλες τις μηχανές και παίρνουμε την ένωση των θετικών προβλέψεων ως multi-label έξοδο.

Σε αυτή την εργασία θα εστιάσουμε περισσότερο σε μεθόδους προσαρμογής του αλγορίθμου, οπότε δεν θα αναλυθούν περισσότεροι οι μέθοδοι μετατροπής του προβλήματος.

2.3.1.2 Μέθοδοι Προσαρμογής Αλγορίθμου

Πολλές φορές δεν μπορούμε ή δεν θέλουμε να πειράξουμε τα δεδομένα, οπότε χρειαζόμαστε κάποιον αλγόριθμο που να εκπαιδεύει απευθείας ένα σύστημα με multi-label δεδομένα. Αυτήν ακριβώς τη δουλειά κάνουν οι τεχνικές προσαρμογής αλγορίθμου, αφού παίρνουν έναν ήδη υπάρχων αλγόριθμο και τον τροποποιούν ώστε να δουλεύει αποτελεσματικά με δεδομένα πολλών ετικετών χωρίς να χρειάζεται να εφαρμόσει κάποιο φίλτρο στα δεδομένα ή να τα επεξεργαστεί. Αναφορικά κάποιες προσαρμογές έχουν γίνει στους αλγορίθμους: C4.5, AdaBoost (AdaBoost.MH και AdaBoost.MR), *k*NN (π.χ. ML-*k*NN), κ.ά. Στη συνέχεια θα ασχοληθούμε κυρίως με προσαρμογές αλγορίθμων εκπαίδευσης νευρωνικών δικτύων.

Οι αλγόριθμοι BP-MLL και BP-MIP (Zhang και Zhou, 2006) είναι προσαρμογές του γνωστού αλγορίθμου back-propagation με σκοπό την υποστήριξη multi-label δεδομένων. Η σημαντικότερη αλλαγή που έγινε σε αυτούς τους αλγορίθμους είναι η εφαρμογή μια νέας συνάρτησης κόστους, η οποία λαμβάνει υπόψη περισσότερες από μία ετικέτες. Το αποτέλεσμα των αλγορίθμων αυτών είναι τύπου MLC.

Ο αλγόριθμος MMP (Crammer και Singer, 2003) είναι μια οικογένεια αλγορίθμων για LR χρήση σε multi-label δεδομένα που εκπαιδεύεται online[1] και χρησιμοποιούν το μοντέλο McCulloch-Pitts. Το δίκτυο MMP δεσμεύει ένα perceptron για κάθε ετικέτα (μέθοδος BR), αλλά η ανανέωση των βαρών γίνεται με τέτοιο τρόπο ώστε να επιτευχθεί όσο το δυνατόν σωστότερη κατάταξη των ετικετών.

Μια άλλη παραλλαγή των MMP είναι τα MLPP (Loza Mencía και Fürnkranz, 2008), τα οποία λειτουργούν όμοια με τα προηγούμενα, αλλά αυτή τη φορά δεσμεύουν ένα perceptron για κάθε ζευγάρι ετικετών – δηλαδή αν το πλήθος των ετικετών είναι n, δεσμεύουν $\frac{n(n-1)}{2}$ perceptrons.

Ο αλγόριθμος CMLPP (Loza Mencía και Fürnkranz, 2008) είναι η εξέλιξη του MLPP, όπου προστίθεται μια τεχνητή ετικέτα κατά την εκπαίδευση. Ο σκοπός της ετικέτας αυτής είναι να διαχωρίζει αυτόματα τις υπόλοιπες ετικέτες σε σχετικές και μη, δίνοντας μια MLR μορφή στην έξοδο του συστήματος.

Τέλος, ο αλγόριθμος ML-RBF (Zhang, 2009) εκπαιδεύει ένα RBF νευρωνικό δίκτυο, χρησιμοποιώντας τον αλγόριθμο k-means για τον υπολογισμό των κέντρων, καθώς και τη μέθοδο *copy* για το μετασχηματισμό των δεδομένων εκπαίδευσης. Η έξοδος του συστήματος είναι ένα LR όπου εφαρμόζεται ένα κατώφλι (συνήθως το μηδέν) για να δώσει το τελικό MLR αποτέλεσμα.

2.3.2 Μετρικές Αξιολόγησης Multi-label Δεδομένων

Οι μετρικές αξιολόγησης για συστήματα που εκπαιδεύονται με multi-label δεδομένα είναι διαφορετικές από αυτών που εκπαιδεύονται με single-label. Σε αυτό το

[1] online μάθηση: καθώς τρέχουμε παραδείγματα στο σύστημα για να πάρουμε ένα αποτέλεσμα, το σύστημα χρησιμοποιεί τα δεδομένα αυτά για να βελτιώσει ταυτόχρονα την εκπαίδευσή του.

κεφάλαιο θα αναφερθούν κάποιες μετρικές αξιολόγησης που έχουν προταθεί στο παρελθόν για multi-label δεδομένα, οι οποίες έχουν εφαρμογή σε μεθόδους που διαχωρίζουν δυαδικά τις ετικέτες (σχετικές και όχι) και σε μεθόδους που δημιουργούν μια κατάταξη των ετικετών.

Για την επεξήγηση των μετρικών αυτών θα χρησιμοποιηθεί η ίδια ορολογία με αυτή του Κεφαλαίου 2.3.1. Επιπλέον, προστίθεται ο όρος Z_i, ο οποίος είναι η πρόβλεψη μιας MLC μεθόδου δίνοντας ένα παράδειγμα $\mathbf{x}_i$, ενώ $r_i(\lambda)$ είναι η πρόβλεψη-κατάταξη μιας μεθόδου LR, όπου η πιο σχετική ετικέτα παίρνει την τιμή 1 ενώ η λιγότερο σχετική την τιμή q.

2.3.2.1 Δυαδικός Διαχωρισμός Ετικετών

Μερικές από τις μετρικές που αξιολογούν MLC δεδομένα υπολογίζονται βασιζόμενες στο μέσο όρο της διαφοράς του πραγματικού συνόλου ετικετών με αυτό που πρόβλεψε το σύστημα, ενώ άλλες είναι βασισμένες στις διαφορές των ετικετών μία προς μία. Οι πρώτες ονομάζονται μετρικές *βασισμένες στο παράδειγμα* (*example-based*) ενώ οι τελευταίες *βασισμένες στην ετικέτα* (*label-based*).

Στη συνέχεια θα αναφερθούν έξι μετρικές "βασισμένες στο παράδειγμα": *hamming-loss*, *classification accuracy*, *precision*, *recall*, F_1 (η αρμονική μέση τιμή των μετρικών precision και recall) και *accuracy*. Η μετρική *Hamming-loss* προσδιορίζεται ως εξής:

$$\text{Hamming} - \text{Loss} = \frac{1}{m}\sum_{i=1}^{m}\frac{|Y_i \,\Delta\, Z_i|}{M}$$

όπου Δ είναι η αντίστοιχη πράξη XOR της λογικής Boole. Η μετρική *classification accuracy* ή αλλιώς *subset accuracy* είναι:

$$\text{ClassificationAccuracy} = \frac{1}{m}\sum_{i=1}^{m} I(Z_i = Y_i)$$

Όπου $I(\text{true}) = 1$ και $I(\text{false}) = 0$. Οι υπόλοιπες προσδιορίζονται παρακάτω:

$$Precision = \frac{1}{m}\sum_{i=1}^{m}\frac{|Y_i \cap Z_i|}{|Z_i|} \qquad Recall = \frac{1}{m}\sum_{i=1}^{m}\frac{|Y_i \cap Z_i|}{|Y_i|}$$

$$F_1 = \frac{1}{m}\sum_{i=1}^{m}\frac{2|Y_i \cap Z_i|}{|Z_i|+|Y_i|} \qquad Accuracy = \frac{1}{m}\sum_{i=1}^{m}\frac{|Y_i \cap Z_i|}{|Y_i \cup Z_i|}$$

Στις μετρικές "βασισμένες στην ετικέτα" μπορούν να εφαρμοστούν όλες οι παραπάνω τεχνικές, για τις οποίες ο υπολογισμός του μέσου όρου όλων των ετικετών μπορεί να επιτευχθεί με τη χρήση δύο τεχνικών: *macro-averaging* και *micro-averaging*. Αν, λοιπόν, $B(tp, tn, fp, fn)$ είναι μια μετρική "βασισμένη στο παράδειγμα", με tp, tn, fp και fn να είναι το πλήθος των θετικών true, αρνητικών true, θετικών false και αρνητικών false αντίστοιχα, οι macro-averaging και micro-averaging εκδοχές της B υπολογίζονται ως εξής:

$$B_{macro} = \frac{1}{q}\sum_{\lambda=1}^{q} B(tp_\lambda, fp_\lambda, tn_\lambda, fn_\lambda)$$

$$B_{micro} = B\left(\sum_{\lambda=1}^{q} tp_\lambda, \sum_{\lambda=1}^{q} fp_\lambda, \sum_{\lambda=1}^{q} tn_\lambda, \sum_{\lambda=1}^{q} fn_\lambda\right)$$

όπου tp_λ, tn_λ, fp_λ και fn_λ είναι τα αντίστοιχα πλήθη θετικών και αρνητικών true ή false για την ετικέτα λ μετά τη δυαδική αξιολόγηση. Έχει παρατηρηθεί ότι σε μερικές περιπτώσεις το αποτέλεσμα του macro-averaging είναι το ίδιο με του micro-averaging.

2.3.2.2 Κατάταξη ετικετών

Η μετρική *one-error* υπολογίζει πόσες φορές η πρώτη στη κατάταξη ετικέτα δεν βρίσκεται στο σύνολο των ετικετών του παραδείγματος:

$$1 - \text{Error} = \frac{1}{m}\sum_{i=1}^{m} \delta\left(\arg\min_{\lambda \in L} r_i(\lambda)\right) \quad \text{όπου} \quad \delta(\lambda) = \begin{cases} 1 \text{ αν } \lambda \notin Y_i \\ 0 \quad \text{αλλιώς} \end{cases}$$

Η μετρική αξιολόγησης *coverage* υπολογίζει, κατά μέσο όρο, πόσο πολύ πρέπει να κατεβούμε στη λίστα της κατάταξης για να καλύψουμε όλες τις σχετικές με το παράδειγμα ετικέτες:

$$\text{Cov} = \frac{1}{m}\sum_{i=1}^{m} \max_{\lambda \in Y_i} r_i(\lambda) - 1$$

Η μετρική *ranking loss* εκφράζει το πλήθος των φορών που μια μη σχετική ετικέτα κατατάχθηκε υψηλότερα από μια σχετική:

$$\text{R} - \text{Loss} = \frac{1}{m}\sum_{i=1}^{m} \frac{1}{|Y_i||\bar{Y}_i|} |\{(\lambda_a, \lambda_b) : r_i(\lambda_a) > r_i(\lambda_b), (\lambda_a, \lambda_b) \in Y_i \times \bar{Y}_i\}|$$

Όπου $\bar{Y}_i$ είναι το συμπληρωματικό σύνολο του Y_i ως προς το L.

Η μετρική *average precision* αξιολογεί το μέσο ποσοστό των ετικετών που έχουν καταταχθεί πάνω από μια ετικέτα $\lambda \in Y_i$, οι οποίες όμως βρίσκονται στο σύνολο Y_i:

$$\text{AvgPrec} = \frac{1}{m}\sum_{i=1}^{m} \frac{1}{|Y_i|} \sum_{\lambda \in Y_i} \frac{|\{\lambda' \in Y_i : r_i(\lambda') \leq r_i(\lambda)\}|}{r_i(\lambda)}$$

Η μετρική *hierarchical loss* είναι μια παραλλαγή της *Hamming-loss* η οποία λαμβάνει υπόψη μια ήδη υπάρχουσα ιεραρχία των ετικετών. Εξετάζει τις προβλέψεις των ετικετών σε συνάρτηση με την ιεραρχία και κάθε φορά που βρίσκει μια λάθος πρόβλεψη, το υποδένδρο που έχει ως ρίζα αυτόν τον κόμβο δε λαμβάνεται υπόψη στον υπολογισμό της απώλειας. Αν $\text{anc}(\lambda)$ είναι το σύνολο των προγόνων του κόμβου λ, ο τύπος της *hierarchical loss* είναι:

$$H - Loss = \frac{1}{m}\sum_{i=1}^{m} |\{\lambda : \lambda \in Y_i \,\Delta\, Z_i, \text{anc}(\lambda) \cap (Y_i \,\Delta\, Z_i) = 0\}|$$

ΚΕΦΑΛΑΙΟ 3: ΔΙΚΤΥΑ ΠΕΠΟΙΘΗΣΗΣ ΜΕΓΑΛΟΥ ΒΑΘΟΥΣ ΠΟΛΛΑΠΛΩΝ ΕΤΙΚΕΤΩΝ

ΔΙΚΤΥΑ ΠΕΠΟΙΘΗΣΗΣ ΜΕΓΑΛΟΥ ΒΑΘΟΥΣ ΠΟΛΛΑΠΛΩΝ ΕΤΙΚΕΤΩΝ

Σε αυτό το κεφάλαιο, θα εστιάσουμε στη δομή του νευρωνικού δικτύου που υλοποιήθηκε σε αυτή την εργασία, πως αυτό μπορεί να εκπαιδευτεί με multi-label σύνολα δεδομένων και πώς η έξοδός του προσαρμόζεται σύμφωνα με τις επιθυμητές ετικέτες. Θα περιγράψουμε την αρχιτεκτονική του νευρωνικού δικτύου και τους αλγορίθμους εκπαίδευσης και δοκιμής. Θα αναφερθούμε, επίσης, σε βιβλιοθήκες που μπορούν να χρησιμοποιηθούν για την υλοποίηση τέτοιων αλγορίθμων. Τέλος, θα εξηγήσουμε τον κώδικα ο οποίος υλοποιήθηκε, τις βιβλιοθήκες που χρησιμοποιήθηκαν, καθώς και το λόγο για τον οποίο επιλέχθηκαν αυτές. Επίσης, θα αναφερθούμε σύντομα σε άλλες βιβλιοθήκες που θα μπορούσαν να χρησιμοποιηθούν.

3.1 Η ΓΕΝΙΚΗ ΙΔΕΑ ΤΟΥ ΑΛΓΟΡΙΘΜΟΥ

Ο τρόπος με τον οποίο λειτουργεί και εκπαιδεύεται ένα DBN αναλύθηκε στο δεύτερο κεφάλαιο. Σε αυτό το κεφάλαιο, θα επεκτείνουμε αυτό το μοντέλο έτσι ώστε να εκπαιδεύεται με επίβλεψη.

Αρχικά, η μηχανή εκπαιδεύεται χωρίς επίβλεψη, κρατώντας όλη την χρήσιμη πληροφορία στα βάρη των συνάψεων των νευρώνων της. Όταν θέλουμε να κάνουμε μια πρόβλεψη χρησιμοποιώντας ένα τέτοιο δίκτυο, βάζοντας στο κατώτερο επίπεδο τα χαρακτηριστικά ενός στιγμιότυπου, μας δίνει στο κορυφαίο επίπεδο ένα διάνυσμα, το οποίο έχει κρατήσει όλη την πληροφορία του στιγμιότυπου αυτού. Αυτό το διάνυσμα μπορεί να έχει διαφορετικό αριθμό διαστάσεων από το αρχικό.

Έως τώρα έχουμε πετύχει τη μείωση των διαστάσεων κρατώντας την περισσότερη δυνατή πληροφορία. Σε αυτό το σημείο, έχουμε μετασχηματίσει, μη γραμμικά, τα δεδομένα εισόδου. Εφαρμόζοντας ένα γραμμικό ταξινομητή στο νέο διάνυσμα, σε συνδυασμό με επιβλεπόμενη μάθηση, η έξοδος του νευρωνικού δικτύου θα συμμορφωθεί με τις ετικέτες των δεδομένων εκπαίδευσης.

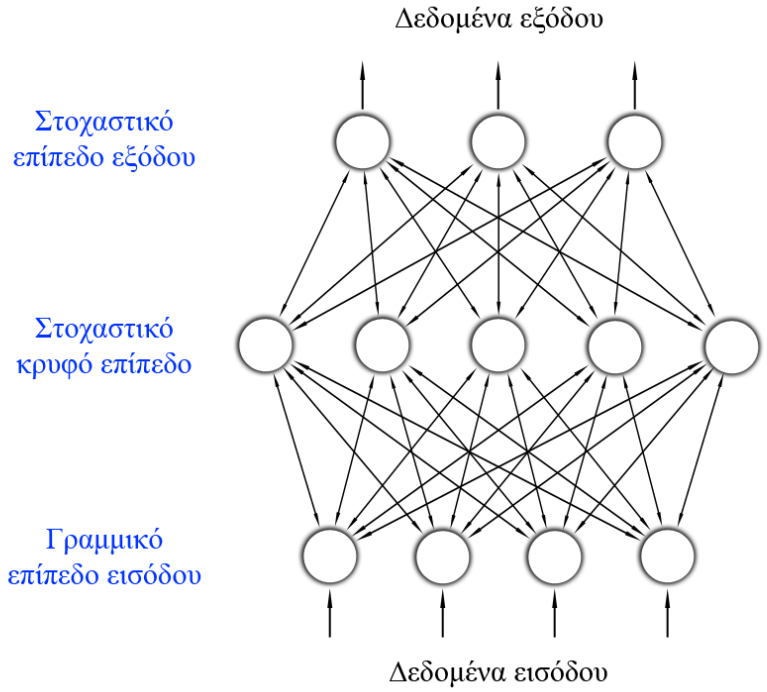
Στα παρακάτω υποκεφάλαια εξηγείται το μοντέλο που υλοποιήθηκε και χρησιμοποιήθηκε στην παρούσα εργασία.

3.1.1 Η Αρχιτεκτονική του Μοντέλου

Η αρχιτεκτονική του μοντέλου χαρακτηρίζεται από το πλήθος των επιπέδων του νευρωνικού δικτύου, το πλήθος των νευρώνων σε κάθε επίπεδο καθώς και τον τύπο των νευρώνων κάθε τέτοιου επιπέδου. Η αρχιτεκτονική του μοντέλου αυτής της εργασίας έχει ως εξής:

1. Το επίπεδο εισόδου είναι γραμμικό. Δηλαδή, αποτελείται από στοχαστικούς νευρώνες, οι οποίοι παίρνουν ως είσοδο πραγματικές τιμές οι οποίες πολλαπλασιάζονται με τα βάρη και αθροίζονται μεταξύ τους. Η έξοδός τους μετασχηματίζεται από μια γραμμική συνάρτηση ενεργοποίησης (π.χ. Λογιστική, βλ Κεφάλαιο 2.1.1.1) και μας δίνει οποιαδήποτε πραγματική τιμή σε κάποιο διάσημα (π.χ. στο διάστημα $[0,1]$). Το πλήθος των νευρώνων αυτού του επιπέδου ορίζεται από το χρήστη και είναι ίσο με το πλήθος των χαρακτηριστικών του προβλήματος.

2. Τα ενδιάμεσα (κρυφά) επίπεδα είναι δυαδικά. Οι νευρώνες τους είναι παρόμοιοι με αυτούς του επιπέδου εισόδου, αλλά η συνάρτηση ενεργοποίησης είναι η συνάρτηση κατωφλίου (βλ. Κεφάλαιο 2.1.1.1). Το πλήθος των κρυφών επιπέδων, καθώς και το πλήθος των νευρώνων σε αυτά καθορίζεται από το χρήστη και συνήθως είναι ανάλογα των χαρακτηριστικών και του πλήθους των ετικετών. Τις περισσότερες φορές υπάρχει μόνο ένα κρυφό επίπεδο για οικονομία χώρου και χρόνου.
3. Το επίπεδο εξόδου είναι παρόμοιο με ένα κρυφό επίπεδο, το οποίο αποτελείται από τόσους νευρώνες όσο και το πλήθος των ετικετών.



Σχήμα 3.1.1 Ένα παράδειγμα της αρχιτεκτονικής ενός δικτύου πεποίθησης μεγάλου βάθους.

Ο τρόπος με τον οποίο λειτουργούν οι στοχαστικοί νευρώνες, καθώς και η αρχιτεκτονική τους, περιγράφονται στο Κεφάλαιο 2.1.1.2. Οι δυαδικοί στοχαστικοί νευρώνες που χρησιμοποιούμε, ακολουθούν το βασικό πρότυπο των βαθιών δικτύων πεποίθησης (Hinton, et. al, 2006) και γι αυτό επιλέχθηκαν ανάμεσα στους υπόλοιπους τύπους νευρώνων.

Στο σχήμα 3.1.1 φαίνεται ένα παράδειγμα της δομής ενός δικτύου, όπως αυτό που περιγράψαμε παραπάνω. Το δίκτυο αυτό αποτελείται από δύο RBMs, ένα που συνδέει το επίπεδο εισόδου με το κρυφό επίπεδο και ένα που συνδέει το κρυφό με το εξόδου.

3.1.2 Ο Αλγόριθμος Μάθησης

Ο αλγόριθμος εκπαίδευσης ταιριάζει με τα πρότυπα του DBN αλγορίθμου, προσθέτοντας ένα επιπλέον επίπεδο στο τέλος, στο οποίο εφαρμόζεται ο γραμμικός ταξινομητής back-propagation – όπως στα MLP. Χάριν απλότητας, θα εξηγήσουμε τον αλγόριθμο χρησιμοποιώντας ένα συγκεκριμένο παράδειγμα. Έστω οι διαστάσεις του διανύσματος εισόδου (χαρακτηριστικά) είναι τέσσερεις, οι ετικέτες είναι τρεις και

το πλήθος των νευρώνων του κρυφού επιπέδου είναι πέντε (βλ. Σχήμα 3.1.1). Ο αλγόριθμος έχει ως εξής:

1. Αρχικά δημιουργείται ένα DBN όπως ακριβώς περιγράφεται στο προηγούμενο κεφάλαιο. Η δομή του (για το συγκεκριμένο παράδειγμα) θα περιέχει δύο RBM:
    a. Το πρώτο RBM, αποτελείται από ένα ορατό επίπεδο με τέσσερεις γραμμικούς νευρώνες και ένα κρυφό επίπεδο με πέντε δυαδικούς-στοχαστικούς.
    b. Το δεύτερο, αποτελείται από ένα ορατό επίπεδο με πέντε δυαδικούς-στοχαστικούς νευρώνες και ένα κρυφό επίπεδο με τρεις δυαδικούς-στοχαστικούς.
2. Σε πρώτη φάση, η εκπαίδευση αφορά όλα τα RBM εκτός του τελευταίου. Στο παράδειγμά μας, αφορά μόνο το πρώτο RBM. Εφαρμόζεται, λοιπόν, ένας άπληστος αλγόριθμος για να εκπαιδεύσει τα πρώτα RBM το ένα μετά το άλλο (βλ. Κεφάλαιο 2.2.2 και 2.2.3) για ένα πλήθος εποχών ορισμένο από τον χρήστη. Για παράδειγμα, αν ο χρήστης έχει ορίσει 20 συνολικές εποχές και 2 ατομικές, τότε το κάθε RBM θα εκπαιδευτεί για 2 εποχές και αυτό θα γίνει για 20 επαναλήψεις.
3. Αφού εκπαιδευτούν τα πρώτα νευρωνικά δίκτυα, εφαρμόζεται ο αλγόριθμος back-propagation για να διαδώσει το σφάλμα προς τα πίσω, διορθώνοντας τα βάρη στις συνάψεις των νευρώνων. Με αυτόν τον τρόπο, κατευθύνουμε το νευρωνικό μας δίκτυο να δίνει στην έξοδο τις ετικέτες που θέλουμε.

Η πρώτη φάση, έχει ως σκοπό να αρχικοποιήσει τα βάρη των συνάψεων ώστε να πατήσει πάνω ο αλγόριθμος back-propagation και να δώσει καλύτερα αποτελέσματα. Αυτό επιτυγχάνεται, καθώς οι νευρώνες είναι στοχαστικοί και η πρώτη φάση της εκπαίδευσης γίνεται με την τεχνική της προσομοιωμένης ανόπτησης (βλ. Κεφάλαιο 2.2.1.1). Η τεχνική αυτή δεν χρησιμοποιεί την παράγωγο για να βρει την κλίση της συνάρτησης και το κόστος εκπαίδευσης μπορεί να αυξηθεί στοχαστικά. Έτσι, μπορούμε να ξεφύγουμε από τοπικά ελάχιστα πριν εφαρμόσουμε μάθηση με επίβλεψη.

Αν εξαιρέσουμε την πρώτη φάση του αλγορίθμου, μοιάζει πολύ με τον αλγόριθμο ADALINE που εφαρμόζεται στα MLP. Η διαφορά τους είναι στους στοχαστικούς νευρώνες και στην πρώτη φάση της εκπαίδευσης, η οποία εγγυάται πολύ καλύτερα αποτελέσματα.

3.1.3 Ο Αλγόριθμος Πρόβλεψης

Αφού ολοκληρωθεί η εκπαίδευση του νευρωνικού δικτύου με τον τρόπο που εξηγήσαμε στο προηγούμενο κεφάλαιο, μπορεί δίνοντάς του ένα στιγμιότυπο του προβλήματος, για το οποίο εκπαιδεύτηκε, να κάνει μια πρόβλεψη για τις ετικέτες οι οποίες είναι σχετικές. Κάνοντας feed forward τα δεδομένα στο δίκτυο, θα πάρουμε από τους νευρώνες εξόδου κάποιες πραγματικές τιμές στο διάστημα $[0,1]$ (μία σε κάθε νευρώνα εξόδου). Από αυτές τις τιμές μπορούμε να βγάλουμε μια κατάταξη (LR) ή μια δυαδική αναπαράσταση (bipartition – MLC) χρησιμοποιώντας μια τιμή κατωφλίου.

Η τιμή κατωφλίου υπολογίζεται αυτόματα χρησιμοποιώντας το σύνολο των δεδομένων εκπαίδευσης, αλλά μπορεί και ο χρήστης να δώσει μια δική του. Αυτό επιτυγχάνεται με κάποια εργαλεία που περιγράφονται στη συνέχεια αυτού του

κεφαλαίου. Ο χρήστης μπορεί να επιλέξει το μοντέλο αναπαράστασης των ετικετών (LR ή MLC) σύμφωνα με το πρόβλημα που καλείται να λύσει. Οι ετικέτες μπορούν πολύ εύκολα να μετατραπούν σε οποιαδήποτε μοντέλο αναπαράστασης επιθυμούμε.

3.2 ΥΛΟΠΟΙΗΣΗ

Το παραπάνω μοντέλο, οι αλγόριθμοι εκπαίδευσης και δοκιμής υλοποιήθηκαν χρησιμοποιώντας τη γλώσσα προγραμματισμού JAVA. Τα εργαλεία που χρησιμοποιήθηκαν είναι τα Mulan, jaRBM και Weka και περιγράφονται στο Κεφάλαιο 3.3. Τα δυναμικά και στατικά διαγράμματα UML της υλοποίησης βρίσκονται σε Παράρτημα Ι. Παρακάτω περιγράφονται οι κλάσεις του μοντέλου, των αλγορίθμων εκπαίδευσης και δοκιμής.

3.2.1 Κλάση Μοντέλου

Το μοντέλο του νευρωνικού δικτύου αντιστοιχεί στη κλάση $DeepBeliefNetwork$. Η κλάση αυτή επεκτείνει την κλάση $RBMNet$ (του εργαλείου jaRBM) και υιοθετεί τα πρότυπα $NeuralNet$ (του εργαλείου Mulan) και $Cloneable$. Αυτή η κλάση αφορά μόνο το μοντέλο και περιέχει μεθόδους για το χειρισμό των στιγμιότυπών της. Παρακάτω περιγράφονται οι μέθοδοι που περιέχει.

- public $DeepBeliefNetwork$(int[] numUnitPerInstance): Ο constructor της κλάσης. Παίρνει ως είσοδο έναν πίνακα με το πλήθος των νευρώνων σε κάθε επίπεδο και δημιουργεί το DBN με τυχαία βάρη στις συνάψεις. Η παράμετρος του constructor πρέπει να περιέχει πληροφορία για τουλάχιστον τρία επίπεδα.
- public $DeepBeliefNetwork$(RBMNet rbmnet): Δημιουργεί ένα DBN χρησιμοποιώντας ένα $RBMNet$ αντικείμενο. Αντιγράφει ουσιαστικά τα δίκτυα RBM από το αντικείμενο της παραμέτρου και τα τοποθετεί στον εαυτό του. Η παράμετρος δεν πρέπει να είναι $null$.
- public int $getNetInputSize$(): Επιστρέφει το πλήθος των νευρώνων εισόδου.
- public int $getNetOutputSize$(): Επιστέφει το πλήθος των νευρώνων εξόδου.
- $public\ int\ getLayersCount$(): Επιστρέφει το πλήθος των επιπέδων του νευρωνικού δικτύου.
- public List < Neuron > $getLayersUnits$(int layerIndex): Επιστρέφει τους νευρώνες ενός συγκεκριμένου επιπέδου σε μορφή λίστας νευρώνων. Η τιμή της παραμέτρου πρέπει να αντιστοιχεί σε κάποιο επίπεδο που υπάρχει στο νευρωνικό δίκτυο.
- public double[] $getOutput$(): Επιστρέφει την έξοδο του νευρωνικού δικτύου για την τελευταία είσοδο που έκανε feed forward.
- public double[] $feedForward$(double[] input): Κάνει feed forward ένα διάνυσμα εισόδου και επιστρέφει την έξοδο του νευρωνικού δικτύου. Το διάνυσμα εισόδου πρέπει να έχει ίδιες διαστάσεις με αυτές του πρώτου επιπέδου του νευρωνικού δικτύου και να μην έχει την τιμή $null$.
- public double[] $feedForward$(double[] input, int lastRBM): Κάνει feed forward ένα διάνυσμα εισόδου και επιστρέφει την έξοδο ενός συγκεκριμένου RBM. Σταματάει δηλαδή το feed forward στο RBM που ορίζει ο χρήστης. Η θέση του RBM που ορίζει ο χρήστης πρέπει αν υπάρχει στο νευρωνικό δίκτυο, δηλαδή να είναι μικρότερη του πλήθους των RBM και μεγαλύτερη του μηδέν. Επίσης, το διάνυσμα εισόδου και το πρώτο επίπεδο του νευρωνικού δικτύου πρέπει να έχουν ίδιες διαστάσεις.
- public double[] $feedBackward$(double[] input): Κάνει feed backward ένα διάνυσμα από ετικέτες και επιστρέφει ένα διάνυσμα χαρακτηριστικών. Το

διάνυσμα εισόδου πρέπει να έχει τις ίδιες διαστάσεις με αυτές του τελευταίου επιπέδου του νευρωνικού δικτύου και να μην έχει την τιμή $null$.

- public double[] $feedBackward$(double[] input,int fromRBM): Κάνει feed backward ένα διάνυσμα από κρυφά χαρακτηριστικά ενός συγκεκριμένου επιπέδου και επιστρέφει ένα διάνυσμα χαρακτηριστικών του πρώτου επιπέδου. Το διάνυσμα εισόδου πρέπει να έχει τις ίδιες διαστάσεις με αυτές του κρυφού επιπέδου του $fromRBM$ στο νευρωνικό δίκτυο και να μην έχει την τιμή $null$.
- public boolean[] $generateFeedForwardUnits$(double[] input): Κάνει feed forward ένα διάνυσμα εισόδου και επιστρέφει ένα bipartition τν ετικετών.
- public boolean[] $generateFeedBackwardUnits$(double[] input): Κάνει feed backward ένα διάνυσμα ετικετών και επιστρέφει το bipartition ενός διανύσματος εισόδου.
- public void $reset$(): Αρχικοποιεί το DBN. Σβήνει ότι έχει μάθει και ορίζει τα βάρη των συνάψεων σε τυχαίες τιμές.
- public DeepBeliefNetwork $clone$(): Επιστρέφει ένα πιστό αντίγραφο του νευρωνικού δικτύου.
- public String $toString$(): επιστρέφει τα βάρη των συνάψεων των νευρώνων για κάθε RBM σε μορφή συμβολοσειράς.

3.2.2 Κλάσεις Εκπαίδευσης και Πρόβλεψης

Για τη διαδικασία της εκπαίδευσης έχουν δημιουργηθεί τρεις κλάσεις ελέγχου. Η $Learner$, η $DBLearner$ και η $LearningManager$, η καθεμιά με το δικό της ρόλο. Η πρώτη είναι μια abstract κλάση, η οποία ταυτίζεται με τη δομή ενός εκπαιδευτή οντοτήτων όπως τα $DeepBeliefNetworks$. Περιέχει μεθόδους για τον ορισμό των παραμέτρων μάθησης και abstract μεθόδους για να υλοποιηθούν από τις υποκλάσεις της. Επεκτείνει την κλάση $MultiLabelLearnerBase$ από τη βιβλιοθήκη Mulan. Στο πλαίσιο αυτής της εργασίας η μόνη κλάση που υλοποιεί αυτό το πρότυπο είναι η $DBLearner$, αλλά σε κάποια επέκταση του προγράμματος θα μπορούσαμε να δημιουργήσουμε έναν $AutoEncoderLearner$, που εκπαιδεύει τέτοιες δομές με διαφορετικό τρόπο (auto-encoders). Η $DBLearner$ περιέχει τις απαραίτητες μεθόδους για την εκπαίδευση ενός νευρωνικού δικτύου βαθιάς μάθησης, την πρόβλεψη των ετικετών ενός στιγμιότυπου και υλοποιεί της abstract μεθόδους της $Learner$. Τέλος, οι κλάσεις $LearningManager$ και $PredictionManager$ διαχειρίζονται την εκπαίδευση του δικτύου και την πρόβλεψη ετικετών, χρησιμοποιώντας τις παραμέτρους του χρήστη. Στη συνέχεια περιγράφονται αυτές οι τρεις κλάσεις.

3.2.2.1 Εκπαιδευτής

Πρόκειται για ένα πρότυπο κλάσης ελέγχου, το οποίο υλοποιεί κάποιες μεθόδους, ενώ κάποιες άλλες τις αφήνει για τις κλάσεις που το υλοποιούν. Επεκτείνει το πρότυπο $MultiLabelLearnerBase$ (του πακέτου Mulan), με σκοπό να υπάρχει συμβατότητα μέσω του εργαλείου. Η κλάση περιέχει μεθόδους που ορίζουν τις απαραίτητες παραμέτρους για την εκπαίδευση.

- public $Learner$(): Ο constructor της κλάσης, ο οποίος αρχικοποιεί τα πεδία της κλάσης στις προκαθορισμένες τιμές τους.

- public void *setDBN*(DeepBeliefNetwork dbn): Αντικαθιστά το τρέχον νευρωνικό δίκτυο με το νευρωνικό της παραμέτρου. Η παράμετρος δεν πρέπει να είναι *null*. Όταν το νευρωνικό δίκτυο δεν έχει οριστεί ξανά, ενημερώνει την κλάση ότι αυτό ορίστηκε και είναι έτοιμο να χρησιμοποιηθεί.
- public void *setEpochs*(int epochs): Ορίζει το πλήθος των (συνολικών) επαναλήψεων του αλγορίθμου εκπαίδευσης. Η προκαθορισμένη τιμή τους είναι 20 και η παράμετρος πρέπει να είναι θετική.
- public void *setEpochsPerLayer*(int epochsPerLayer): Αυτή η μέθοδος ορίζει το πλήθος των εποχών για την ξεχωριστή εκπαίδευση του κάθε RBM. Η παράμετρος της μεθόδου πρέπει να είναι θετική. Η προκαθορισμένη τιμή της είναι 2.
- public void *setMomentum*(double momentum): Η μέθοδος που ορίζει την ορμή του αλγορίθμου εκπαίδευσης. Η ορμή δείχνει πόσο απότομες θα είναι οι κινήσεις του πάνω στο υπερεπίπεδο, κατά την αναζήτηση της βέλτιστης λύσης. Μια πολύ μικρή ορμή οδηγεί τον αλγόριθμο σε μικρά και σταθερά βήματα, ενώ μια μεγάλη τον ωθεί να ξεπεράσει τα τοπικά ακρότατα. Η καλύτερη λύση είναι κάπου στη μέση, οπότε η προκαθορισμένη τιμή του είναι στο 0.6. Η τιμή της παραμέτρου αυτής πρέπει να κυμαίνεται ανάμεσα στο μηδέν και στο ένα.
- public void *setLearningRate*(double learningRate): Ορίζει το ρυθμό μάθησης του αλγορίθμου εκπαίδευσης. Ο ρυθμός μάθησης αφορά το πόσο γρήγορα μαθαίνει το νευρωνικό δίκτυο. Ένας μεγάλος ρυθμός μάθησης μπορεί να φέρει αρνητικά αποτελέσματα στη μάθηση, αφού μπορεί να ωθήσει τον αλγόριθμο να μη συγκλίνει ποτέ. Ένας μικρός ρυθμός μάθησης μπορεί να κάνει τον αλγόριθμο να συγκλίνει πολύ αργά, με αποτέλεσμα να χρειαστεί πολλές εποχές για να συγκλίνει. Η τιμή του ρυθμού μάθησης πρέπει να βρίσκεται στο διάστημα $[0, 1]$. Η προκαθορισμένη τιμή του είναι 0.05.
- public void *setWeightCost*(double weightCost): Ορίζει τη συχνότητα με την οποία αλλάζουν τα βάρη κατά τη διαδικασία της εκπαίδευσης. Ορίζει, δηλαδή, το πόσο μεγάλη είναι η διαφορά των βαρών μετά από κάθε επανάληψη του αλγορίθμου. Η τιμή της πρέπει να είναι ανάμεσα στο μηδέν και στο ένα. Η προκαθορισμένη τιμή της είναι 0.00001.
- public void *setThreshold*(double threshold): Η μέθοδος που ορίζει την τιμή κατωφλίου. Η τιμή κατωφλίου δεν χρειάζεται κατά την εκπαίδευση, αλλά είναι απαραίτητη για να οριστεί η έξοδος του νευρωνικού δικτύου κατά την πρόβλεψη. Αυτή πρέπει να είναι ένας πραγματικός αριθμός ανάμεσα στο μηδέν και το ένα. Η προκαθορισμένη τιμή της είναι -1 που σημαίνει ότι δεν έχει οριστεί. Αν ο χρήστης δεν ορίσει αυτή την τιμή, θα υπολογιστεί αυτόματα κατά την εκπαίδευση του νευρωνικού δικτύου, με μια μέθοδο που περιγράφεται στη συνέχεια.
- public DeepBeliefNetwork *getLearnedDBN*(): Επιστρέφει το DBN αν αυτό έχει εκπαιδευτεί.
- public int *getEpochs*(): Επιστρέφει το πλήθος των εποχών εκπαίδευσης του DBN.
- public int *getEpochsPerLayer*(): Επιστρέφει το πλήθος των εποχών εκπαίδευσης του κάθε RBM ξεχωριστά.
- public double *getMomentum*(): Επιστρέφει την τιμή της ορμής του αλγορίθμου.

- public double *getLearningRate*(): Επιστρέφει την τιμή του ρυθμού μάθησης του αλγορίθμου.
- public double *getWeightCost*(): Επιστρέφει τη συχνότητα με την οποία αλλάζουν τα βάρη μετά από κάθε επανάληψη.
- public double *getThreshold*(): Επιστρέφει την τιμή κατωφλίου, αν αυτή έχει οριστεί ή ενημερώνει ότι δεν έχει οριστεί.
- public boolean *isSetDBN*(): Δίνει καταφατική απάντηση αν το νευρωνικό δίκτυο έχει οριστεί, ενώ αρνητική αν όχι.
- public MultiLabelOutput[] *makePrediction*(Instances instances): Δημιουργεί πολλαπλές προβλέψεις για τα παραδείγματα του ορίσματος χρησιμοποιώντας την αντίστοιχη μέθοδο για πρόβλεψη ενός στιγμιότυπου. Επιστρέφει την απάντηση σε ένα πίνακα προβλέψεων.
- protected ThresholdFunction *buildThresholdFunction*(Instances instances) : Η μέθοδος που δημιουργεί μια συνάρτηση κατωφλίου χρησιμοποιώντας ένα σύνολο στιγμιότυπων. Η συνάρτηση αυτή μπορεί αργότερα να μας δώσει την τιμή κατωφλίου, αν αυτή δεν έχει οριστεί από το χρήστη.

3.2.2.2 DBN Εκπαιδευτής

Ο εκπαιδευτής DBN νευρωνικών δικτύων είναι μια κλάση ελέγχου η οποία επεκτείνει την κλάση *Learner*. Ειδικεύει στην εκπαίδευση δικτύων τέτοιου τύπου χρησιμοποιώντας τους κατάλληλους αλγορίθμους. Η ίδια κλάση μπορεί να κάνει προβλέψεις χρησιμοποιώντας το νευρωνικό δίκτυο που εκπαίδευσε. Υλοποιεί τις abstract μεθόδους των κλάσεων *Learner* και *MultiLabelLearnerBase*. Στη συνέχεια γίνεται μια σύντομη περιγραφή των μεθόδων αυτής της κλάσης.

- void *buildInternat*(MultiLabelInstance mli): Υλοποιεί την abstract μέθοδο της κλάσης MultiLabelLearnerBase. Μετατρέπει τα δεδομένα εισόδου σε πίνακες double αριθμών και τα χωρίζει σε δεδομένα εισόδου και εξόδου. Εκπαιδεύει το νευρωνικό δίκτυο μη επιβλεπόμενα χρησιμοποιώντας τη μέθοδο *buildUnsupervised*. Στη συνέχεια πραγματοποιεί εκπαίδευση με επίβλεψη, χρησιμοποιώντας τη συνάρτηση *buildSupervised*.
- void *buildUnsupervised*(double[][] dataset, double lr): Δημιουργεί έναν άπληστο εκπαιδευτή *GreedyLearner* (χρησιμοποιώντας το εργαλείο jaRBM) και εκπαιδεύει χωρίς επίβλεψη τα πρώτα RBM, για τόσες εποχές όσες έχει ορίσει ο χρήστης (βλ. Κεφάλαιο 3.1.2). Μετά τη μη επιβλεπόμενη εκπαίδευση, αποθηκεύει το εκπαιδευμένο νευρωνικό δίκτυο στη θέση του παλιού.
- void *buildSupervised*(double[][]dataset, double[][] targets, double lr): Δημιουργεί έναν εκπαιδευτή για να εφαρμόσει εκπαίδευση back-propagation χρησιμοποιώντας αυτή τη φορά τις ετικέτες των στιγμιότυπων και μεταφέροντας προς τα πίσω το σφάλμα. Μετά την επιβλεπόμενη εκπαίδευση, αποθηκεύει το εκπαιδευμένο νευρωνικό δίκτυο στη θέση του παλιού.
- MultiLabelOutput *makePredictionInternal*(Instance instance): Δημιουργεί μια πρόβλεψη για το στιγμιότυπο εισόδου. Το στιγμιότυπο αυτό πρέπει να είναι όμοιο με αυτά που εκπαιδεύτηκε το νευρωνικό δίκτυο. Αν τα χαρακτηριστικά του είναι περισσότερα, θεωρούμε ότι κάποια από αυτά είναι οι ετικέτες, οπότε τις αφαιρούμε. Αφού πάρουμε τα χαρακτηριστικά που χρειαζόμαστε, τα κάνουμε feed forward στο νευρωνικό δίκτυο και παίρνουμε τις πιθανότητες της εξόδου. Στη συνέχεια αν δεν έχει οριστεί τιμή κατωφλίου,

τη δημιουργούμε χρησιμοποιώντας τη συνάρτηση που φτιάξαμε στη μέθοδο *buildInternal*. Δημιουργούμε ένα bipartition της πρόβλεψης χρησιμοποιώντας την τιμή κατωφλίου. Τέλος, κατασκευάζουμε ένα αντικείμενο της κλάσης *MultiLabelOutput* χρησιμοποιώντας τις πιθανότητες και τη δυαδική ακολουθία ψηφίων και το επιστρέφουμε ως πρόβλεψη.

- TechnicalInformation *getTechnicalInformation*(): Η μέθοδος που επιστρέφει ένα αντικείμενο τύπου *TechnicalInformation* (του εργαλείου Weka). Το αντικείμενο αυτό περιέχει πληροφορίες για το τεχνικό υπόβαθρο τις κλάσης αυτή, σχετικές εργασίες, και άλλες πληροφορίες.
- String *globalInfo*(): Επιστρέφει μια περιγραφή του εκπαιδευτή σε μορφή συμβολοσειράς.

3.2.2.3 Διαχειριστής Μάθησης

Ο ρόλος αυτής της κλάσης ελέγχου, είναι να χειριστεί τις παραμέτρους που έδωσε ο χρήστης, να τις περάσει στον εκπαιδευτή και να ξεκινήσει τη διαδικασία της μάθησης. Οι μέθοδοι αυτής της κλάσης περιγράφονται στη συνέχεια του κεφαλαίου.

- public *LearningManager*(): Ο constructor της κλάσης δημιουργεί έναν νέο εκπαιδευτή.
- public void *SetLearnerParameters*(OptionsHandler oh): Δέχεται ως παράμετρο ένα χειριστή επιλογών και μετατρέπει τις επιλογές του χρήστη σε παραμέτρους του εσωτερικού εκπαιδευτή. Αν δεν κληθεί αυτή η μέθοδος, η διαδικασία της μάθησης δεν μπορεί να ξεκινήσει.
- public long *startLearning*(): Η μέθοδος αυτή μπορεί να κληθεί μόνο αν έχουν οριστεί οι παράμετροι του εκπαιδευτή. Δίνει εντολή στον εκπαιδευτή να εκπαιδεύσει το νευρωνικό δίκτυο και επιστρέφει το συνολικό χρόνο εκπαίδευσης σε milliseconds.
- public Learner *getLearner*(): Επιστρέφει τον εσωτερικό εκπαιδευτή της κλάσης.

3.2.2.4 Διαχειριστής Πρόβλεψης

Ακόμα μία κλάση ελέγχου, ο ρόλος της οποίας είναι να διαχειριστεί τη διαδικασία της πρόβλεψης. Αποτελείται από μόνο μία μέθοδο η οποία περιγράφεται στη συνέχεια.

- public long *TestData*(DBLearner learner, MultiLabelInstances mli): Χρησιμοποιεί τη δομή *Evaluator* (του πακέτου Mulan) για να πάρει κάποιες μετρικές για την απόδοση σύμφωνα με τα δεδομένα της παραμέτρου. Τυπώνει τις τιμές των μετρικών στην οθόνη και επιστρέφει το χρόνο εκτέλεσης της δοκιμής σε milliseconds.

3.2.3 Άλλες Κλάσεις

Έχουν υλοποιηθεί κάποιες βοηθητικές κλάσεις, για την ομαλότερη λειτουργία του προγράμματος. Αυτές είναι οι *DBNFile*, *OptionsHandler*, *FlowManager* και *Converter*, η καθεμία για διαφορετικό σκοπό. Η *DBNFile* διαχειρίζεται την αποθήκευση και ανάκτηση ενός νευρωνικού δικτύου DBN στο δίσκο. Η *OptionsHandler* χειρίζεται τις επιλογές του χρήστη, οι οποίες εισάγονται κατά την εντολή για εκτέλεση του προγράμματος. Η *FlowManager*, διαχειρίζεται τη ροή των δεδομένων και τη σειρά με την οποία εκτελεστούν οι εντολές (περιέχει τη μέθοδο

main). Η κλάση *Converter* είναι υπεύθυνη για τις μετατροπές των δεδομένων από έναν τύπο σε κάποιον άλλο. Στη συνέχεια του κεφαλαίου περιγράφονται αναλυτικότερα αυτές οι κλάσεις.

3.2.3.1 Διαχειριστής DBN Αρχείων

Διαχειρίζεται τα αρχεία των νευρωνικών δικτύων. Περιέχει μόνο στατικές μεθόδους, οι οποίες αποσκοπούν στην αποθήκευση νευρωνικών δικτύων στο δίσκο ή ανάκτηση στην ανάκτησή τους από αυτόν. Οι μέθοδοι αυτές περιγράφονται στη συνέχεια.

- public DeepBeliefNetwork *GetFromFile*(String filename): Η μέθοδος που διαβάζει από το δίσκο ένα αρχείο που έχει αποθηκευμένο ένα DBN νευρωνικό δίκτυο. Χρησιμοποιεί την αντίστοιχη μέθοδο της κλάσης RBMNetFile του πακέτου jaRBM.
- public void *SaveToFile*(DeepBeliefNetwork someDBN): Αποθηκεύει στο δίσκο το DBN νευρωνικό δίκτυο του ορίσματος σε ένα αρχείο που η διεύθυνσή του είναι προκαθορισμένη. Η διεύθυνση αυτή είναι η "DATA/mydbn.dbn". Στην πραγματικότητα, χρησιμοποιείται η μέθοδος *SaveToFile*(someDBN, filename), με παράμετρο την προκαθορισμένη διεύθυνση.
- public void *SaveToFile*(DeepBeliefNetwork someDBN, String filename): Αποθηκεύει στο σκληρό δίσκο το νευρωνικό δίκτυο της πρώτης παραμέτρου, στη διεύθυνση του αρχείου που δίνεται στην δεύτερη παράμετρο. Αυτό υλοποιείται στο πακέτο jaRBM για δίκτυα τύπου RBMNet και εδώ χρησιμοποιείται για DBN νευρωνικά δίκτυα.

3.2.3.2 Αποκωδικοποιητής Επιλογών

Συλλέγει τις επιλογές του χρήστη και τις μετατρέπει στη μορφή που τις χρειάζεται το πρόγραμμα. Ορίζει ένα σύνολο επιλογών για το χρήστη, ο οποίος χρησιμοποιώντας αυτές μπορεί να αλλάξει τις παραμέτρους της εκπαίδευσης. Οποιαδήποτε επιλογή δεν ανήκει σε αυτό το σύνολο, δεν μπορεί να την επεξεργαστεί. Οι επιλογές αυτές φαίνονται στον πίνακα 3.2.1. Ο χρήστης δεν είναι υποχρεωμένος να ορίσει όλες αυτές τις επιλογές, εφόσον υπάρχει αρχική τιμή γι αυτές. Στη συνέχεια περιγράφονται οι μέθοδοι της κλάσης αυτής.

- public *OptionsHandler*(): Ο constructor της κλάσης δημιουργεί μια δομή που περιέχει όλες τις δυνατές επιλογές του χρήστη.
- public void *setArgs*(String[] args): Δέχεται ως όρισμα ένα πίνακα συμβολοσειρών με τις επιλογές του χρήστη και τις αποθηκεύει σε κατάλληλες μεταβλητές, ώστε να μπορεί αργότερα να τις δώσει όταν αυτές ζητηθούν. Χρησιμοποιεί τη μέθοδο *readArg*(arg) για να βρει τον τύπο της κάθε επιλογής του πίνακα.
- private void *readArg*(String arg): Ανάλογα με τον τύπο του ορίσματος, το αποθηκεύει, μετατρέποντάς το κατάλληλα, σε τοπικές μεταβλητές.
- public static MultiLabelInstances *GetData*(String path): Επιστρέφει τα δεδομένα του αρχείου του ορίσματος σε *MultiLabelInstances* μορφή.
- public static String *GetFileName*(String path): Δέχεται ως όρισμα ένα μονοπάτι που οδηγεί σε ένα αρχείο και επιστρέφει το όνομα αυτού του αρχείου.

επιλογή	Επεξήγηση
-path	Το μονοπάτι που βρίσκονται τα αρχεία των δεδομένων εκπαίδευσης, δοκιμής και πληροφοριών.
-train	Το όνομα του ARFF αρχείου με τα δεδομένα εκπαίδευσης.
-test	Το όνομα του ARFF αρχείου με τα δεδομένα δοκιμής.
-load	Το όνομα του αρχείου που είναι αποθηκευμένο το νευρωνικό δίκτυο που θέλουμε να ανακτήσουμε.
-info	Το όνομα του αρχείου XML με τις πληροφορίες του συνόλου δεδομένων.
-targets	Το πλήθος των ετικετών. Αν είναι ορισμένη η επιλογή "info" τότε αυτή η επιλογή δεν χρησιμοποιείται.
-be	Το πλήθος των επαναλήψεων της εκπαίδευσης του DBN κατά την επιβλεπόμενη μάθηση.
-ge	Το πλήθος των επαναλήψεων της εκπαίδευσης του κάθε RBM κατά την μη επιβλεπόμενη μάθηση.
-m	Η τιμή του momentum (ορμή).
-lr	Η τιμή του ρυθμού μάθησης.
-wcost	Η τιμή του weight cost (πόσο πολύ αλλάζουν τα βάρη σε κάθε επανάληψη).
-l	Το πλήθος των νευρώνων σε κάθε επίπεδο του νευρωνικού δικτύου.
-threshold	Η τιμή κατωφλίου.
-debug	Επιλέγεται αν θέλουμε να εμφανίζεται στην οθόνη η πρόοδος της εκπαίδευσης.

Πίνακας 3.2.1 Οι επιλογές του χρήστη και οι επεξηγήσεις τους

- public String $getTrainPath()$: Επιστρέφει τη θέση του αρχείου με τα δεδομένα εκπαίδευσης στο δίσκο, αν αυτή έχει οριστεί.
- public String $getTestPath()$: Επιστρέφει τη θέση του αρχείου με τα δεδομένα δοκιμής στο δίσκο. Αν αυτή δεν έχει οριστεί, επιστρέφει τη θέση του αρχείου με τα δεδομένα εκπαίδευσης.
- public MultiLabelInstances $getTrainSet()$: Επιστρέφει τα δεδομένα εκπαίδευσης σε μορφή $MultiLabelInstances$ αντικειμένου.
- public MultiLabelInstances $getTestSet()$: Επιστρέφει τα δεδομένα δοκιμής σε μορφή $MultiLabelInstances$ αντικειμένου. Αν το αρχείο τους δεν έχει οριστεί επιστρέφει τα δεδομένα εκπαίδευσης.
- public DeepBeliefNetwork $getNN()$: Αν ο χρήστης έχει ορίσει το νευρωνικό δίκτυο δίνοντας ως παράμετρο ένα αρχείο που το περιέχει, τότε επιστρέφει αυτό το νευρωνικό δίκτυο. Αν δεν το έχει ορίσει ο χρήστης, τότε δημιουργεί ένα καινούργιο χρησιμοποιώντας το πλήθος των νευρώνων σε κάθε επίπεδο και επιστρέφει αυτό.
- public int[] $getLayers()$: Επιστρέφει το πλήθος των νευρώνων σε κάθε επίπεδο, σε μορφή πίνακα ακεραίων, αν αυτό έχει οριστεί από το χρήστη.
- public int $getBPEpochs()$: Επιστρέφει το πλήθος των εποχών που θα εκπαιδεύεται το DBN κατά την εκπαίδευση με επίβλεψη, αν αυτό έχει οριστεί. Αν όχι επιστρέφει την προκαθορισμένη τιμή, δηλαδή την τιμή 10.
- public int $getGreedyEpochs()$: Επιστρέφει το πλήθος των εποχών που θα εκπαιδεύεται το κάθε RBM ξεχωριστά κατά την εκπαίδευση χωρίς επίβλεψη,

αν ο χρήστης το έχει ορίσει. Σε διαφορετική περίπτωση επιστρέφει την τιμή 100 (προκαθορισμένη τιμή).

- public double *getMomentum*(): Επιστρέφει την τιμή της ορμής του αλγορίθμου εκπαίδευσης. Η προκαθορισμένη τιμή είναι 0.6.
- public double *getLearningRate*(): Επιστρέφει την τιμή του ρυθμού μάθησης για τον αλγόριθμο εκπαίδευσης, εφόσον αυτή έχει οριστεί από το χρήστη. Διαφορετικά, επιστρέφει την προκαθορισμένη τιμή 0.05.
- public double *getWeightCost*(): Επιστρέφει τη συχνότητα αλλαγής των βαρών των συνάψεων σε κάθε επανάληψη. Αν ο χρήστης δεν έχει ορίσει αυτή την τιμή, επιστρέφει την τιμή 0.00001, η οποία είναι και η προκαθορισμένη.
- public double *getThreshold*(): Επιστρέφει την τιμή κατωφλίου. Η προκαθορισμένη τιμή είναι -1, που σημαίνει ότι δεν έχει οριστεί από το χρήστη.
- public boolean *getDebug*(): Ενημερώνει αν θα τυπώνεται η πρόοδος της εκπαίδευσης. Αν ο χρήστης δεν έχει επιλέξει αυτή την επιλογή, επιστρέφει αρνητική απάντηση. Σε διαφορετική περίπτωση, επιστρέφει θετική.

3.2.3.3 Μετατροπέας

Η κλάση του μετατροπέα *Converter* είναι μια βοηθητική κλάση με στατικές μεθόδους, που βοηθάει στη μετατροπή των αντικειμένων, του εργαλείου Mulan, που παριστάνουν δεδομένα σε μονοδιάστατα και πολυδιάστατα διανύσματα. Στη συνέχεια περιγράφονται οι στατικές μέθοδοι αυτής της κλάσης.

- *MultiLabelOutputArrayToBipartitionArray*(MultiLabelOutput[] Z): Μετατρέπει έναν πίνακα αντικειμένων της κλάσης *MultiLabelOutput* του πακέτου Mulan σε ένα δυσδιάστατο πίνακα λογικών τιμών, χρησιμοποιώντας την εσωτερική τιμή κατωφλίου.
- *MultiLabelOutputArrayToDoubleMatrix*(MultiLabelOutput[] Z): Μετατρέπει έναν πίνακα αντικειμένων της κλάσης *MultiLabelOutput* σε ένα δυσδιάστατο πίνακα πραγματικών τιμών.
- *MultiLabelOutputToBipartition*(MultiLabelOutput z): Μετατρέπει ένα αντικείμενο της κλάσης *MultiLabelOutput* σε ένα μονοδιάστατο πίνακα λογικών τιμών, χρησιμοποιώντας την εσωτερική τιμή κατωφλίου.
- *MultiLabelOutputToDoubleArray*(MultiLabelOutput z): Μετατρέπει ένα αντικείμενο της κλάσης *MultiLabelOutput* σε ένα μονοδιάστατο πίνακα πραγματικών τιμών.
- *InstancesToBipartitionArray*(Instances instances, int[] indices): Μετατρέπει ένα αντικείμενο της κλάσης *Instances* του πακέτου Weka σε ένα δυσδιάστατο πίνακα λογικών τιμών, χρησιμοποιώντας την εσωτερική τιμή κατωφλίου. Ο πίνακας που επιστρέφεται περιέχει μόνο τα χαρακτηριστικά των οποίων οι θέσεις βρίσκονται στον πίνακα του ορίσματος.
- *InstancesToDoubleMatrix*(Instances instances, int[] indices): Μετατρέπει ένα αντικείμενο της κλάσης *Instances* σε ένα δυσδιάστατο πίνακα πραγματικών τιμών. Ο επιστρεφόμενος πίνακας περιέχει μόνο τα χαρακτηριστικά των οποίων οι θέσεις βρίσκονται στον πίνακα του ορίσματος.
- *InstanceToBipartition*(Instance instance, int[] indices): Μετατρέπει ένα αντικείμενο της κλάσης *Instance* του πακέτου Weka σε ένα μονοδιάστατο πίνακα λογικών τιμών, χρησιμοποιώντας την εσωτερική τιμή κατωφλίου. Ο

πίνακας που επιστρέφεται περιέχει μόνο τα χαρακτηριστικά που ο πίνακας του ορίσματος υποδεικνύει τις θέσεις τους στο διάνυσμα εισόδου.

- *InstanceToDoubleArray*(Instance instance, int[]indices): Μετατρέπει ένα αντικείμενο της κλάσης *Instance* σε ένα πίνακα πραγματικών τιμών μίας διάστασης. Ο νέος πίνακας περιέχει μόνο τα χαρακτηριστικά που υποδεικνύει τις θέσεις τους ο πίνακας του ορίσματος.
- *SetThreshold*(double threshold): Ορίζει μια νέα τιμή στην τιμή κατωφλίου. Η προκαθορισμένη τιμή είναι 0.45.

3.2.3.4 Διαχειριστής Ροής

Πρόκειται για την κλάση που ρυθμίζει τη σειρά με την οποία θα εκτελούνται οι εντολές, αφού περιέχει τη μέθοδο *main*. Δέχεται τα ορίσματα του χρήστη και χρησιμοποιεί τις υπόλοιπες κλάσεις για να καταφέρει την εκτέλεση όλων των απαραίτητων διεργασιών. Παρακάτω περιγράφονται οι μέθοδοι της κλάσης αυτής.

- public static OptionsHandler *ParseArguments*(String[] args): Δέχεται ως παράμετρο τις επιλογές του χρήστη. Δημιουργεί έναν *OptionsHandler*, ο οποίος θα διαβάσει τις επιλογές του χρήστη και θα τις μετατρέψει σε διαχειρίσιμες τιμές (βλ. Κεφάλαιο 3.2.3.2).
- public static Stirng *FormatMilliseconds*(long milliseconds): Μετατρέπει ένα πλήθος milliseconds σε συμβολοσειρά που δείχνει τις ώρες, τα λεπτά, τα δευτερόλεπτα και τα δέκατα του δευτερολέπτου.
- public static void *main*(String[] args): Η μέθοδος που εκτελείται κατά την εκκίνηση του προγράμματος. Μετατρέπει την είσοδο του χρήστη σε ένα χειριστή *OptionsHandler* μέσω της μεθόδου που περιγράφηκε παραπάνω. Δημιουργεί έναν *LearningManager*, του δίνει τις επιλογές του χρήστη και ξεκινάει τη διαδικασία της μάθησης. Όταν αυτή τελειώσει, τυπώνει το χρόνο εκτέλεσης της διαδικασίας αυτής και αποθηκεύει στο εκπαιδευμένο νευρωνικό δίκτυο στο δίσκο, στη θέση "DATA/filename.dbn" όπου το filename είναι αντίστοιχο με αυτό των δεδομένων εκπαίδευσης. Τέλος, δοκιμάζει τα δεδομένα δοκιμής, χρησιμοποιώντας έναν *PredictionManager* και τυπώνει τις μετρικές αξιολόγησης και το χρόνο δοκιμής.

3.3 ΕΡΓΑΛΕΙΑ ΠΟΥ ΧΡΗΣΙΜΟΠΟΙΗΘΗΚΑΝ

Πριν αρχίσουμε να υλοποιούμε την εργασία, έγινε έρευνα για τα εργαλεία που θα μπορούσαμε να χρησιμοποιήσουμε, τα καλύτερα από τα οποία φαίνονται στον πίνακα 3.3.1. Αρχικά, δοκιμάσαμε το πακέτο Theano στη γλώσσα Python. Η εμπειρία μας για αυτή τη βιβλιοθήκη περιγράφεται σύντομα στην ενότητα 3.3.1. Το εργαλείο που χρησιμοποιήσαμε τελικά είναι το jaRBM, το οποίο είναι μια JAVA βιβλιοθήκη για RBM νευρωνικά δίκτυα. Επίσης, χρησιμοποιήθηκαν οι JAVA βιβλιοθήκες Mulan και Weka, οι οποίες περιγράφονται στη συνέχεια του κεφαλαίου.

Εργαλείο	Γλώσσα Προγραμματισμού	GPU	Περιγραφή
Theano	Python	Ναι	Βιβλιοθήκη συμβολικών εκφράσεων σε Python χρησιμοποιώντας CPU/GPU
mPoT	Python	Ναι	Χρησιμοποιεί τα εργαλεία CUDAMat και gnumpy για να εκπαιδεύει μοντέλα φυσικών φωτογραφιών
DeepLearnToolbox	MATLAB	Όχι	Ένα εργαλείο για τη χρήση τεχνικών Βαθιάς Μάθησης
Deep Belief Networks	MATLAB	Όχι	Κώδικας για την εκπαίδευση DBN νευρωνικών δικτύων
matrbm	MATLAB	Όχι	Απλοποιημένη μορφή κώδικα για τη εκπαίδευση DBN νευρωνικών δικτύων
jaRBM	JAVA	Όχι	Μια JAVA βιβλιοθήκη για RBM νευρωνικά δίκτυα
Cuda-Convnet	C++/CUDA	Ναι	Μια εφαρμογή που περιέχει αλγορίθμους μάθησης για feed-forward νευρωνικά δίκτυα .

Πίνακας 3.3.1 Σχετικά εργαλεία, που μπορούν να χρησιμοποιηθούν για τη δημιουργία κώδικα για νευρωνικά δίκτυα πεποίθησης μεγάλου βάθους.

3.3.1 Η βιβλιοθήκη Theano

Η βιβλιοθήκη συμβολικών εκφράσεων Theano, ήταν η πρώτη που επιλέχθηκε για την υλοποίηση της εργασίας, αφού μπορεί να χρησιμοποιήσει τους πυρήνες της κάρτας γραφικών για να κάνει πιο γρήγορους υπολογισμούς. Επίσης, οι εντολές της και η λογική, έμοιαζε αρκετά με τον τρόπο που δουλεύει το MATLAB, απλά δούλευε σε Python. Το κύριο πρόβλημα που δημιουργήθηκε με τη βιβλιοθήκη αυτή ήταν, ότι η είσοδος του νευρωνικού δικτύου έπρεπε αναγκαστικά να είναι διανύσματα με δυαδικές τιμές, ενώ τα δεδομένα εκπαίδευσης που είχαμε στη διάθεσή μας ήταν διανύσματα πραγματικών αριθμών. Επίσης, υπήρχε πρόβλημα με τον τύπο των αρχείων, καθώς δεν υπήρχε αποδοτικός κώδικας για ανάγνωση αρχείων τύπου ARFF (Attribute-Relation File Format), συμπιεσμένα με την sparse τεχνική. Όλα αυτά τα προβλήματα που προέκυψαν, μας απέτρεψαν από το να τη χρησιμοποιήσουμε για την υλοποίηση των νευρωνικών δικτύων μεγάλου βάθους και του αλγορίθμου εκπαίδευσής τους.

3.3.2 Η βιβλιοθήκη jaRBM

Η βιβλιοθήκη jaRBM περιέχει κλάσεις και μεθόδους, που βοηθάνε πολύ στη συγγραφή του κώδικα. Περιέχει την κλάση *RBMImpl*, η οποία υλοποιεί την αρχιτεκτονική και τις λειτουργίες ενός RBM νευρωνικού δικτύου όπως περιγράφονται στο Κεφάλαιο 2.2.3.

Η κλάση *RBMNet* αντιπροσωπεύει μια στοίβα από RBM νευρωνικά δίκτυα, τα οποία συνδέονται για να δημιουργήσουν τη δομή. Η κλάση αυτή προσφέρει τη δομή ενός δικτύου βαθιάς μάθησης, ενώ οι λειτουργίες του δεν προσφέρονται. Αυτό δίνει μεγαλύτερη ευχέρεια στο χρήστη να τα χρησιμοποιήσει όπως αυτός θέλει. Στην εργασία αυτή χρησιμοποιούμε τα *RBMNet* για να φτιάξουμε DBN νευρωνικά δίκτυα (βλ. Κεφάλαιο 3.2.1). Μπορούμε όμως να δημιουργήσουμε και auto-encoders χρησιμοποιώντας την ίδια κλάση.

Η κλάση *RBMNetFile* δίνει τη δυνατότητα αποθήκευσης του RBM δικτύου στο δίσκο και ανάκτησής του από αυτόν. Αυτό είναι πολύ σημαντικό, γιατί μπορούμε να κρατήσουμε τις ρυθμίσεις και τα βάρη των συνάψεων του νευρωνικού δικτύου στο δίσκο και αργότερα να τα ανακτήσουμε χωρίς να χρειάζεται να το εκπαιδεύσουμε ξανά. Στο Κεφάλαιο 3.2.3.4 αναφέρεται το κομμάτι της εργασίας που η κλάση αυτή μας φάνηκε χρήσιμη.

Η κλάση *RBMNetLearner* χρησιμοποιείται για την εκπαίδευση του νευρωνικού δικτύου. Πρόκειται για μια abstract κλάση η οποία περιέχει ένα πρότυπο μεθόδων για τις υποκλάσεις της. Αποτελείται από τρία πεδία: το *RBMNet* με το οποίο θα ασχοληθεί, έναν *BatchDatasourceReader* ο οποίος είναι ο χειριστής του συνόλου δεδομένων, και το *batchSize* (το πλήθος των διανυσμάτων που θα διαβάζει κάθε φορά από το αρχείο). Στην πραγματικότητα, δεν μπορούμε να έχουμε αντικείμενα της κλάσης αυτής, οπότε χρησιμοποιούνται μόνο οι υποκλάσεις της.

Η κλάση *GreedyLearner*, εκπαιδεύει το δίκτυο των RBM άπληστα, ενώ ο *StochasticMetaDescentLearner* επιβλεπόμενα τροφοδοτώντας προς τα πίσω το σφάλμα (βλ. Κεφάλαιο 3.2.2.2 μέθοδος *buildInternat*). Οι δύο αυτές κλάσεις επεκτείνουν την κλάση *RBMNetLearner* και κληρονομούν τις μεθόδους και τα πεδία της.

Η κλάση *BatchDatasourceReader*, χρησιμεύει στην ανάγνωση, από το σκληρό δίσκο, των αρχείων με τα δεδομένα. Μπορεί όμως να πάρει ως είσοδο και έναν δισδιάστατο πίνακα και να τον χρησιμοποιήσει στη θέση ενός αρχείου. Αυτός ο τρόπος χρήσης μας φάνηκε χρήσιμος σε αυτή την εργασία, καθώς μπορούσαμε πολύ εύκολα να πάρουμε τα δεδομένα σε μορφή πίνακα πραγματικών αριθμών.

3.3.3 Η βιβλιοθήκη Mulan

Αυτή η βιβλιοθήκη περιέχει αλγορίθμους μηχανικής μάθησης και μεθόδους αξιολόγησης της μάθησης για σύνολα multi-label δεδομένων. Έχουμε βασιστεί πάνω στα πρότυπα αυτής της βιβλιοθήκης, χρησιμοποιώντας abstract κλάσεις και μεθόδους για πλήρη συμβατότητα. Περιέχει την κλάση *MultiLabelInstances*, που αντιπροσωπεύει το σύνολο των δεδομένων εισόδου και την κλάση *MultiLabelOutput*, η οποία δίνει στα δεδομένα εξόδου μια δομή και τους προσθέτει ιδιότητες για την ευκολότερη μετατροπή τους.

Η κλάση *MultiLabelLearnerBase*, δίνει μια δομή στην κλάση του εκπαιδευτή. Περιέχει μεθόδους για την ομαλή λειτουργία της κλάσης που την επεκτείνει. Η κλάση *Learner* που υλοποιήσαμε κληρονομεί αυτή την κλάση, με σκοπό την ομοιόμορφη δομή και την πληρότητα των λειτουργιών της.

Η κλάση *Evaluator*, χρησιμοποιώντας έναν εκπαιδευτή που επεκτείνει την κλάση *MultiLabelLearnerBase* και ένα σύνολο δεδομένων τύπου *MultiLabelInstances*, μπορεί να δημιουργήσει μια αξιολόγηση σε μορφή *Evaluation*. Η αξιολόγηση αυτή περιέχει τις τιμές των μετρικών αξιολόγησης multi-label προβλέψεων, όπως αυτές του Κεφαλαίου 2.3.2, οι οποίες έχουν τη δυνατότητα τα εμφανιστούν στην έξοδο του προγράμματος.

3.3.4 Η βιβλιοθήκη Weka

Η βιβλιοθήκη Mulan χρησιμοποιεί κλάσεις και μεθόδους αυτού του πακέτου, οπότε, σχεδόν αναγκαστικά, θα έπρεπε να το χρησιμοποιήσουμε κι εμείς. Πιο συγκεκριμένα, περιέχει τις κλάσεις *Instance* και *Instances*, τα οποία είναι απαραίτητα για την αναπαράσταση του συνόλου των δεδομένων. Επίσης, περιέχει απαραίτητες μεθόδους, όπως τη μέθοδο που διαβάζει αρχεία δεδομένων τύπου ARFF.

Δίνει τη δυνατότητα να συμπεριφέρεσαι σε sparse συμπιεσμένα δεδομένα σαν να είναι dense. Τα δεδομένα αυτά, διαβάζονται από κάποιο αρχείο ARFF που είναι συμπιεσμένο σε μορφή sparse. Αυτή η μορφή συμπίεσης, χρησιμοποιείται όταν το σύνολο δεδομένων έχει πολύ αραιά χαρακτηριστικά, συμπεριλαμβάνοντας πολλά μηδενικά. Αφαιρεί όλα τα μηδενικά και προσθέτει, ως δείκτη, τη θέση του κάθε μη μηδενικού αριθμού πριν από αυτόν. Με αυτόν τον τρόπο, διπλασιάζουμε το μέγεθος του κάθε διανύσματος, αλλά δεν συμπεριλαμβάνουμε τα μηδενικά, οπότε αν το μεγαλύτερο μέρος του διανύσματος αποτελείται από μηδέν έχουμε συμπίεση.

ΚΕΦΑΛΑΙΟ 4: ΑΝΑΛΥΣΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ

ΑΝΑΛΥΣΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ

Σε αυτό το κεφάλαιο, θα συγκρίνουμε τα αποτελέσματα που μας δίνει το νευρωνικό δίκτυο που περιγράψαμε παραπάνω με άλλους αλγορίθμους νευρωνικών δικτύων. Θα σχολιάσουμε το γεγονός ότι αυτό δίνει (ή όχι) καλά αποτελέσματα και θα προβληματιστούμε με αυτό. Θα αναφερθούμε σύντομα στα σύνολα των δεδομένων που χρησιμοποιήθηκαν και θα παρατηρήσουμε τη συμπεριφορά των DBNs για κάθε ένα από αυτά, εστιάζοντας στον τύπο του κάθε συνόλου.

4.1 ΣΥΝΟΛΑ ΔΕΔΟΜΕΝΩΝ

Τα σύνολα των δεδομένων (datasets) επιλέχθηκαν με βάση τον τύπο των δεδομένων, το πλήθος των χαρακτηριστικών, των ετικετών και των παραδειγμάτων. Σκοπός μας είναι να δείξουμε τη δυνατότητα του νευρωνικού δικτύου να κάνει προβλέψεις ανεξάρτητα με τον τύπο των δεδομένων και να συγκρίνουμε την απόδοσή του με άλλα νευρωνικά δίκτυα. Μερικές γενικές πληροφορίες των συνόλων δεδομένων συνοψίζονται στον Πίνακα 4.1.1.

Όνομα	Πλήθος χαρ.	Πλήθος ετικετών	Πλήθος παρ.	cardinality	Τύπος
Emotions	72	6	593	1.869	Music
Scene	294	6	2 407	1.074	Image
Mediamill	120	101	43 907	4.376	Video
Enron	1 001	53	1 702	3.378	Text
Medical	1 449	45	978	1.245	Text
Genbase	1 186	27	662	1.252	Biology
Yeast	103	14	2 417	4.237	Biology

Πίνακας 4.1.1 Κάποια βασικά χαρακτηριστικά των συνόλων δεδομένων με τα οποία εκπαιδεύτηκε το νευρωνικό δίκτυο που υλοποιήσαμε.

Στην πρώτη στήλη του Πίνακα 4.1.1 φαίνεται το όνομα του κάθε συνόλου δεδομένων, όπως αυτό είναι γνωστό. Οι επόμενες τρεις στήλες, είναι το πλήθος των χαρακτηριστικών, των ετικετών και των παραδειγμάτων αντίστοιχα για κάθε σύνολο δεδομένων. Η στήλη *"cardinality"* δείχνει πόσες, κατά μέσο όρο, ετικέτες αντιστοιχούν σε κάθε παράδειγμα του συνόλου δεδομένων, ενώ η στήλη *"τύπος"* την κατηγορία προβλημάτων στην οποία ανήκει το κάθε σύνολο.

Το σύνολο δεδομένων *emotions* (Trohidis et al., 2008) αποτελείται από 593 τραγούδια τύπου κλασικής μουσικής, reggae, rock, pop, hip-hop, techno και jazz. Από το κάθε τραγούδι, έχουν βγει 72 χαρακτηριστικά, τα οποία έχουν σχέση με την αλλαγή της περιόδου του κάθε κομματιού και το τέμπο του. Οι ετικέτες του είναι 6 και η κάθε μία αντιπροσωπεύει ένα συγκεκριμένο συναίσθημα.

Το σύνολο *scene* (Boutell et al., 2004) χαρακτηρίζεται από 294 τιμές, οι οποίες αντιπροσωπεύουν μια εικόνα. Οι 6 ετικέτες περιγράφουν το αντικείμενο της εικόνας (π.χ. ένα τοπίο αποτελούμενο από μια παραλία και ένα ηλιοβασίλεμα). Το σύνολο αυτό έχει 2407 παραδείγματα εικόνων με τις περιγραφές τους.

Το *mediamill* σύνολο δεδομένων αποτελείται από 43907 αποσπάσματα από βίντεο, τα οποία χαρακτηρίζονται από 120 πραγματικές τιμές. Οι 101 ετικέτες αυτών των βίντεο αντιστοιχούν στο θέμα του. Δηλαδή, περιγράφουν το αντικείμενο για το οποίο μιλάει (π.χ. αθλητικά, πολιτική, κ.α.).

Το σύνολο *enron* (Read, 2004), περιέχει πληροφορίες για 1702 e-mail. Οι πληροφορίες αυτές συνοψίζονται σε 1001 χαρακτηριστικά που αντιπροσωπεύουν το κείμενο του μηνύματος και 53 ετικέτες που η κάθε μία είναι ένας φάκελος του λογαριασμού. Σκοπός αυτού του συνόλου δεδομένων είναι να καταχωρίσουμε το e-mail σε σωστούς φακέλους (πιθανώς περισσότερους από έναν).

Το σύνολο δεδομένων *medical*, αποτελείται από 1449 κλινικές αναφορές των 978 χαρακτηριστικών, οι οποίες συνδέονται με 45 ετικέτες. Η κάθε ετικέτα αντιπροσωπεύει έναν κωδικό διάγνωσης, ενώ τα χαρακτηριστικά συμβολίζουν την περιγραφή των συμπτωμάτων μιας ασθένειας. Στόχος είναι να αντιστοιχίσουμε τα συμπτώματα με έναν ή περισσότερους κωδικούς κάποιας ασθένειας.

Στο *genbase* (Diplaris et al., 2005), κάθε παράδειγμα περιέχει 1186 χαρακτηριστικά, τα οποία χαρακτηρίζουν μια πρωτεΐνη και 27 ετικέτες οι οποίες είναι ένα σύνολο πρωτεϊνών. Αποτελείται από 662 τέτοια παραδείγματα, όπου το καθένα περιγράφει μια σύνθεση πρωτεϊνών.

Το σύνολο δεδομένων *yeast* (Elisseeff και Weston, 2002), περιέχει παραδείγματα που περιγράφουν γονίδια, τα οποία συνδέονται με 103 χαρακτηριστικά. Συνολικά υπάρχουν 2417 γονίδια και 14 πιθανές ετικέτες για το καθένα.

4.2 ΠΕΙΡΑΜΑΤΑ

Για τη δοκιμή των παραπάνω συνόλων δεδομένων, πειραματιστήκαμε με τις παραμέτρους του εκπαιδευτή του νευρωνικού δικτύου. Συγκεκριμένα, δοκιμάσαμε διάφορες τιμές της τιμής του learning rate και της τιμής του momentum. Θεωρούμε πάντα δύο επίπεδα, για να συνδυάσουμε ταχύτητα και καλά αποτελέσματα. Όσο περισσότερα επίπεδα έχουμε, τόσο καλύτερα αποτελέσματα θα πάρουμε (Hinton, 2006), όμως θα δείξουμε ότι ακόμα και με δύο επίπεδα βάθος τα αποτελέσματα είναι καλύτερα από τους άλλους αλγορίθμους. Οι νευρώνες στα επίπεδα παρατάσσονται έτσι ώστε να δημιουργούν ένα κωνικό σχήμα, το οποίο τις περισσότερες φορές δίνει καλά αποτελέσματα (Hinton, 2006).

Συγκρίναμε το νευρωνικό δίκτυο DBN με άλλους αλγορίθμους εκπαίδευσης multi-label δεδομένων, οι οποίοι στηρίζονται και πάλι σε νευρωνικά δίκτυα. Πιο συγκεκριμένα, τα αποτελέσματα των BP-MLL και MMP αλγορίθμων εκπαίδευσης είναι αυτά που μας προσέγγισαν το ενδιαφέρον.

Στη συνέχεια, θα αναλύσουμε τα αποτελέσματα των τριών νευρωνικών δικτύων, θα τα συγκρίνουμε και θα δείξουμε τους πειραματισμούς που κάναμε με τις παραμέτρους ώστε να πετύχουμε καλύτερα αποτελέσματα. Θα δούμε ότι για κάθε ξεχωριστό τύπο dataset έχουμε διαφορετικά αποτελέσματα.

4.2.1 Πειράματα με Σύνολα Δεδομένων Ήχου

Το πρώτο σύνολο δεδομένων που δοκιμάσαμε ήταν το σύνολο emotions που περιγράφηκε παραπάνω. Δημιουργήσαμε ένα σύνολο πέντε τιμών για την επιλογή του momentum και ένα σύνολο επτά τιμών για τη επιλογή του learning rate. Το πλήθος των χαρακτηριστικών αυτού του συνόλου είναι 72 και το πλήθος των ετικετών 6, οπότε το DBN που δημιουργήσαμε έχει τη δομή $[72, 36, 15, 6]$, όπου κάθε αριθμός αντιστοιχεί στο πλήθος των νευρώνων του κάθε επιπέδου. Στους πίνακες 4.2.1 και 4.2.2 φαίνονται τα πειράματα που κάναμε με τις τιμές των learning rate και momentum στην προσπάθειά μας να πετύχουμε καλύτερα αποτελέσματα.

momentum	Hamming-Loss	F Measure	Example Based Accuracy	R-Loss
0.01	0.2244	0.6054	0.5206	0.2001
0.05	**0.2219**	0.6050	0.5359	0.2104
0.10	0.2261	0.6310	0.5417	0.1890
0.15	0.2236	**0.6398**	**0.5491**	**0.1848**
0.20	0.2434	0.5938	0.5128	0.2148

Πίνακας 4.2.1 Μετρικές αξιολόγησης (Hamming Loss, F1, Accuracy, R-Loss) για τις διάφορες τιμές του momentum στο πείραμα ήχου emotions.

Παρατηρούμε ότι για momentum ίσο με 0.05 έχουμε το καλύτερο αποτέλεσμα για τη μετρική αξιολόγησης Hamming-Loss, ενώ για momentum ίσο με 0.15 έχουμε τα καλύτερα αποτελέσματα για τις υπόλοιπες μετρικές. Τα πειράματα για την εύρεση της καλύτερης τιμής του momentum έγιναν με learning rate ίσο με 0.1. Για την τιμή του learning rate, πειραματιστήκαμε με τις τιμές του, έχοντας σταθερό το momentum στο

0.1. Τα καλύτερα αποτελέσματα που είχαμε σε όλες τις μετρικές αξιολόγησης ήταν για learning rate ίσο με 0.01 (βλ. Πίνακα 4.2.2).

Learning rate	Hamming-Loss	F Measure	Example Based Accuracy	R-Loss
0.001	0.2219	0.6050	0.5359	0.2104
0.005	0.2318	0.6206	0.5375	0.1984
0.010	**0.2129**	**0.6589**	**0.5718**	**0.1714**
0.050	0.2475	0.6219	0.5342	0.2043
0.100	0.2219	0.6050	0.5359	0.2104
0.150	0.2186	0.6386	0.5538	0.1829
0.200	0.3399	0.6225	0.5075	0.2054

Πίνακας 4.2.2 Μετρικές αξιολόγησης (Hamming Loss, F1, Accuracy, R-Loss) για τις διάφορες τιμές του learning rate στο πείραμα ήχου emotions.

Έχοντας επιλέξει τις παραμέτρους για την εκπαίδευση του DBN εκπαιδεύουμε και δοκιμάζουμε τους αλγορίθμους BP-MLL και MMP αφήνοντας τις παραμέτρους όπως έχουν οριστεί από default στη βιβλιοθήκη Mulan. Τα αποτελέσματα των τριών νευρωνικών δικτύων συνοψίζονται στον Πίνακα 4.2.3.

Νευρωνικό Δίκτυο	Hamming-Loss	F Measure	Example Based Accuracy	R-Loss	Χρόνος Εκπαίδευσης (Λ:ΔΔ.δ)
DBN	**0.2129**	**0.6589**	**0.5718**	**0.1714**	4:41.0
BP-MLL	0.3375	0.6338	0.4983	0.2040	0:02.4
MMP	-	-	-	0.3376	**0:00.1**

Πίνακας 4.2.3 Μετρικές αξιολόγησης για τρία διαφορετικά νευρωνικά δίκτυα που κάνουν προβλέψεις σε multi-label δεδομένα. Οι τιμές αναφέρονται στο dataset emotions.

Παρατηρούμε ότι το νευρωνικό δίκτυο DBN δίνει τα καλύτερα αποτελέσματα σε όλες τις μετρικές αξιολόγησης. Το δίκτυο MMP δεν μας έδωσε μετρικές MLC αλλά μας έδωσε μια LR μετρική, η οποία το βάζει τελευταίο στην κατάταξη. Το μόνο μειονέκτημα του DBN είναι ο χρόνος εκπαίδευσης, ο οποίος είναι πολύ μεγαλύτερος από αυτόν στους άλλους δύο αλγορίθμους. Το καλύτερο χρόνο (δηλαδή το μικρότερο) τον είχε το δίκτυο MMP, το οποίο εκπαιδεύτηκε σε ένα δέκατο του δευτερολέπτου.

4.2.2 Πειράματα με Σύνολα Δεδομένων Εικόνας

Το σύνολο δεδομένων *scene* είναι ένα σύνολο δεδομένων εικόνας. Σε αυτή τη ενότητα θα ασχοληθούμε με αυτό το σύνολο δεδομένων, όπου η διάταξη των νευρώνων του νευρωνικού δικτύου γίνεται σύμφωνα με τον πίνακα $[294, 147, 59, 6]$, όπου ο κάθε αριθμός αντιπροσωπεύει το πλήθος των νευρώνων σε κάθε επίπεδο του νευρωνικού δικτύου. Αρχικά πειραματιστήκαμε με τις τιμές των παραμέτρων momentum και learning rate, όπου δημιουργήσαμε ένα σύνολο πέντε τιμών για την

πρώτη και πέντε για τη δεύτερη. Στους πίνακες 4.2.4 και 4.2.5 φαίνονται τα πειράματα με τις παραμέτρους αυτές.

Κρατώντας σταθερή την τιμή του learning rate στο 0.1, αλλάζαμε την τιμή του momentum για να βρούμε την καλύτερη δυνατή. Στον Πίνακα 4.2.4 παρατηρούμε ότι για momentum ίσο με 0.1 έχουμε τα καλύτερα αποτελέσματα για τις μετρικές αξιολόγησης F1 και Accuracy, ενώ για momentum ίσο με 0.01 έχουμε τις καλύτερες τιμές στις μετρικές Hamming-Loss και Ranking-Loss.

Παρατηρούμε ότι για την τιμή 0.1 οι μετρικές που μας δείχνουν την ακρίβεια του αλγορίθμου είναι οι καλύτερες, ενώ για την τιμή 0.01 καλύτερες είναι οι μετρικές που δείχνουν την απώλεια. Παρακάτω θα δούμε ότι όποια από τις δύο τιμές και να επιλέξουμε, το αποτέλεσμα σε σχέση με τους άλλους αλγορίθμους είναι το ίδιο. Εμείς θα επιλέξουμε την τιμή 0.1 για το momentum, γιατί η διαφορά των τιμών της Hamming-Loss και Ranking-Loss είναι μικρή σε σχέση με την 0.01, ενώ το αντίθετο δεν ισχύει.

momentum	Hamming-Loss	F Measure	Example Based Accuracy	R-Loss
0.01	**0.2942**	0.1752	0.1716	**0.4847**
0.05	0.2982	0.1948	0.1851	0.4861
0.10	0.3064	**0.2136**	**0.1973**	0.4861
0.15	0.2965	0.1953	0.1855	0.4861
0.20	0.2981	0.1919	0.1831	0.4861

Πίνακας 4.2.4 Μετρικές αξιολόγησης (Hamming Loss, F1, Accuracy, R-Loss) για τις διάφορες τιμές του momentum στο πείραμα εικόνας scene.

Στον Πίνακα 4.2.5 έχουμε τα αποτελέσματα του learning rate. Εκεί φαίνεται ότι για learning rate ίσο με 0.001 το DBN έχει την μικρότερη απώλεια σύμφωνα με τις μετρικές Hamming-Loss και Ranking-Loss. Επίσης, έχει την καλύτερη ακρίβεια για τιμή 0.05 σύμφωνα με την μετρική F1, ενώ για 0.01 σύμφωνα με τη μετρική Accuracy.

Learning rate	Hamming-Loss	F Measure	Example Based Accuracy	R-Loss
0.001	**0.2569**	0.2106	0.1752	**0.4846**
0.005	0.2854	**0.2154**	0.1786	0.4861
0.010	0.3064	0.2136	**0.1973**	0.4861
0.050	0.3005	0.2015	0.1959	0.4856
0.100	0.2908	0.1962	0.1968	0.4861

Πίνακας 4.2.5 Μετρικές αξιολόγησης (Hamming Loss, F1, Accuracy, R-Loss) για τις διάφορες τιμές του learning rate στο πείραμα εικόνας scene.

Στον Πίνακα 4.2.6 συγκρίνουμε τα τρία νευρωνικά δίκτυα. Το DBN μας έδωσε σαφώς καλύτερα αποτελέσματα από το BP-MLL σε όλες τις μετρικές αξιολόγησης, αλλά ο χρόνος εκπαίδευσης είναι απαγορευτικός σε σχέση με τους άλλους δύο. Το MMP έδωσε τα καλύτερα τη μικρότερη απώλεια, σύμφωνα με τη μετρική Ranking-Loss και ο χρόνος εκπαίδευσης που χρειάστηκε ήταν πολύ λιγότερος.

Νευρωνικό Δίκτυο	Hamming-Loss	F Measure	Example Based Accuracy	R-Loss	Χρόνος Εκπαίδευσης (ΛΛ:ΔΔ.δ)
DBN	**0.3064**	**0.2136**	**0.1973**	0.4861	89:22.3
BP-MLL	0.4599	0.1379	0.0868	0.5000	00:28.0
MMP	-	-	-	**0.2122**	**00:00.4**

Πίνακας 4.2.6 Μετρικές αξιολόγησης για τρία διαφορετικά νευρωνικά δίκτυα που κάνουν προβλέψεις σε multi-label δεδομένα. Οι τιμές αναφέρονται στο dataset scene.

Παρατηρούμε ότι σε αυτό το σύνολο δεδομένων το νευρωνικό μας δίκτυο δεν έδωσε τα αναμενόμενα αποτελέσματα. Αυτό συνέβη λόγο της συνάρτησης που χρησιμοποιούμε για να βρούμε το κατώφλι. Σύμφωνα με τα πειράματα που κάναμε, υπάρχουν μετρικές τις οποίες δεν τις αναφέρουμε, όπως η micro-average AUC και η macro-average AUC, οι οποίες αφορούν την ακρίβεια σύμφωνα με την κατάταξη των ετικετών και μας δίνουν τις τιμές 0.7068 και 0.8260 αντίστοιχα.

4.2.3 Πειράματα με Βιολογικά Σύνολα Δεδομένων

Το σύνολο δεδομένων *yeast* είναι ένα σύνολο δεδομένων βιολογίας. Σε αυτή τη ενότητα θα ασχοληθούμε με αυτό το σύνολο δεδομένων, όπου η διάταξη των νευρώνων του νευρωνικού δικτύου γίνεται σύμφωνα με τον πίνακα $[103, 52, 21, 14]$, όπου ο κάθε αριθμός αντιπροσωπεύει το πλήθος των νευρώνων σε κάθε επίπεδο του νευρωνικού δικτύου. Αρχικά πειραματιστήκαμε με τις τιμές των παραμέτρων momentum και learning rate, όπου δημιουργήσαμε ένα σύνολο πέντε τιμών για την πρώτη και επτά για τη δεύτερη. Στους πίνακες 4.2.7 και 4.2.8 φαίνονται τα πειράματα με τις παραμέτρους αυτές.

momentum	Hamming-Loss	F Measure	Example Based Accuracy	R-Loss
0.01	**0.2225**	**0.6336**	**0.5198**	**0.1785**
0.05	0.2477	0.6247	0.5045	0.1822
0.10	0.2444	0.6261	0.5078	0.1838
0.15	0.2377	0.6283	0.5118	0.1824
0.20	0.2503	0.6272	0.5078	0.1863

Πίνακας 4.2.7 Μετρικές αξιολόγησης (Hamming Loss, F1, Accuracy, R-Loss) για τις διάφορες τιμές του momentum στο πείραμα βιολογίας yeast.

Παρατηρούμε ότι, αυτή τη φορά, για momentum ίσο με 0.01 έχουμε τις καλύτερες τιμές όλων των μετρικών αξιολόγησης. Για τα πειράματα του momentum κρατήσαμε το learning rate σταθερό και ίσο με 0.1 και αλλάζαμε τις τιμές του momentum. Για τα πειράματα του learning rate κρατήσαμε σταθερό το momentum και πειραματιστήκαμε με την τιμή του learning rate.

Όπως φαίνεται στον πίνακα 4.2.8, η τιμή 0.005 για το learning rate δίνει την καλύτερη ακρίβεια σύμφωνα με τις μετρικές αξιολόγησης F1 και Accuracy, ενώ για ρυθμό μάθησης ίσο με 0.15 έχουμε τη λιγότερη απώλεια σύμφωνα με τη μετρική αξιολόγησης Hamming-Loss και για τιμή 0.1 τη λιγότερη απώλεια σύμφωνα με τη μετρική Ranking-Loss. Εμείς επιλέξαμε την τιμή 0.005 για το learning rate, καθώς η

διαφορά στις τιμές των Hamming-Loss και Ranking-Loss ανάμεσα στα πειράματα δεν έχουν μεγάλη διαφορά.

Learning rate	Hamming-Loss	F Measure	Example Based Accuracy	R-Loss
0.001	0.2304	0.6426	0.5262	0.1840
0.005	0.2263	**0.6440**	**0.5307**	0.1856
0.010	0.2270	0.6433	0.5298	0.1877
0.050	0.2371	0.6350	0.5220	0.1869
0.100	0.2444	0.6261	0.5078	**0.1838**
0.150	**0.2184**	0.6302	0.5141	0.1880
0.200	0.2660	0.6128	0.4807	0.1890

Πίνακας 4.2.8 Μετρικές αξιολόγησης (Hamming Loss, F1, Accuracy, R-Loss) για τις διάφορες τιμές του learning rate στο πείραμα βιολογίας yeast.

Νευρωνικό Δίκτυο	Hamming-Loss	F Measure	Example Based Accuracy	R-Loss	Χρόνος Εκπαίδευσης (ΛΛ:ΔΔ.δ)
DBN	**0.2304**	**0.6426**	**0.5307**	**0.1856**	25:56.2
BP-MLL	0.4386	0.5407	0.3901	0.3447	00:21.3
MMP	-	-	-	0.4810	**00:00.8**

Πίνακας 4.2.9 Μετρικές αξιολόγησης για τρία διαφορετικά νευρωνικά δίκτυα που κάνουν προβλέψεις σε multi-label δεδομένα. Οι τιμές αναφέρονται στο dataset yeast.

Σε σύγκριση με τα άλλα δύο νευρωνικά δίκτυα, το DBN δίνει καλύτερα αποτελέσματα σε προβλήματα βιολογικών δεδομένων σε όλες τις μετρικές αξιολόγησης. Ο Πίνακας 4.2.9 δείχνει ότι το νευρωνικό δίκτυο MMP, ενώ έχει τον καλύτερο χρόνο εκπαίδευσης, δίνει τα χειρότερα αποτελέσματα, ενώ το BP-MLL δίνει κάτι ενδιάμεσο, όπως έκανε και στα προηγούμενα πειράματα. Το νευρωνικό δίκτυο DBN, έχει πολύ καλύτερη απόδοση από τα άλλα δύο, αλλά και πάλι έχει το μεγαλύτερο χρόνο εκπαίδευσης. Παρόλα αυτά, στην παρούσα εργασία μας ενδιαφέρει περισσότερο η απόδοση του νευρωνικού δικτύου μετά την εκπαίδευση και λιγότερο ο χρόνος εκπαίδευσης.

ΚΕΦΑΛΑΙΟ 5: ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΜΕΛΛΟΝΤΙΚΕΣ ΕΠΕΚΤΑΣΕΙΣ

ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΜΕΛΛΟΝΤΙΚΕΣ ΕΠΕΚΤΑΣΕΙΣ

Σε αυτό το κεφάλαιο, μιλάμε για κάποια γενικά συμπεράσματα σχετικά με τα νευρωνικά δίκτυα μεγάλου βάθους που αναλύσαμε και συζητήσαμε σε προηγούμενα κεφάλαια. Τα συμπεράσματα, αφορούν κυρίως θέματα σχεδίασης και αποτελεσμάτων, καθώς και συνδυασμό των δύο. Θα αναφερθούμε, επίσης, σε δυσκολίες που παρουσιάστηκαν και κάποιες επεκτάσεις που μπορούν να γίνουν, είτε από εμάς είτε κάποιον άλλο που θα ήθελε να συνεχίσει αυτή την εργασία.

5.1 ΣΥΜΠΕΡΑΣΜΑΤΑ

Στα πλαίσια αυτής της εργασίας, εξετάσαμε τα νευρωνικά δίκτυα πεποίθησης μεγάλου βάθους. Είδαμε τη δομή, τον αλγόριθμο εκπαίδευσής τους και τον τρόπο με τον οποίο λειτουργούν. Καταφέραμε να συνδυάσουμε διαφορετικούς τομείς και να εξηγήσουμε πώς αυτοί μπορούν να μας δώσουν μια πολύ δυνατή μηχανή. Δοκιμάσαμε να εκπαιδεύσουμε αυτά να νευρωνικά δίκτυα με διάφορα σύνολα multi-label δεδομένων και δείξαμε ότι μπορούν να ανταπεξέλθουν στα περισσότερα από αυτά.

Παρατηρώντας τη δομή και τον τρόπο εκπαίδευσής τους, είδαμε ότι μπορούν να ξεπεράσουν τοπικά ελάχιστα και να αποφύγουν τα προβλήματα που συναντάνε τα νευρωνικά δίκτυα που στηρίζονται στην επιβλεπόμενη μάθηση και μόνο. Οι ιδιότητές τους είναι εντυπωσιακές, αφού έχουν τη δυνατότητα να κάνουν προβλέψεις και χρησιμοποιώντας αυτές να παράγουν καινούρια δεδομένα. Αυτό γίνεται λόγο του τελευταίου RBM, το οποίο λειτουργεί σαν μια συσχετιστική μνήμη.

Τα πειράματα που κάναμε στο τέταρτο κεφάλαιο, μας έδειξαν ότι μπορούν να ανταγωνιστούν τους καλύτερους αλγορίθμους που ειδικεύονται σε multi-label δεδομένα και να δώσουν μια ικανοποιητική λύση σε τέτοιου είδους προβλήματα.

Η τεχνολογία αυτή έχει ήδη τεθεί σε χρήση σε υπηρεσίες όπως η εικονική προσωπική βοηθός της Apple, η οποία βασίζεται στην υπηρεσία αναγνώρισης ομιλίας και στην υπηρεσία Street View της Google, η οποία χρησιμοποιεί μηχανική όραση για να προσδιορίσει συγκεκριμένες διευθύνσεις.

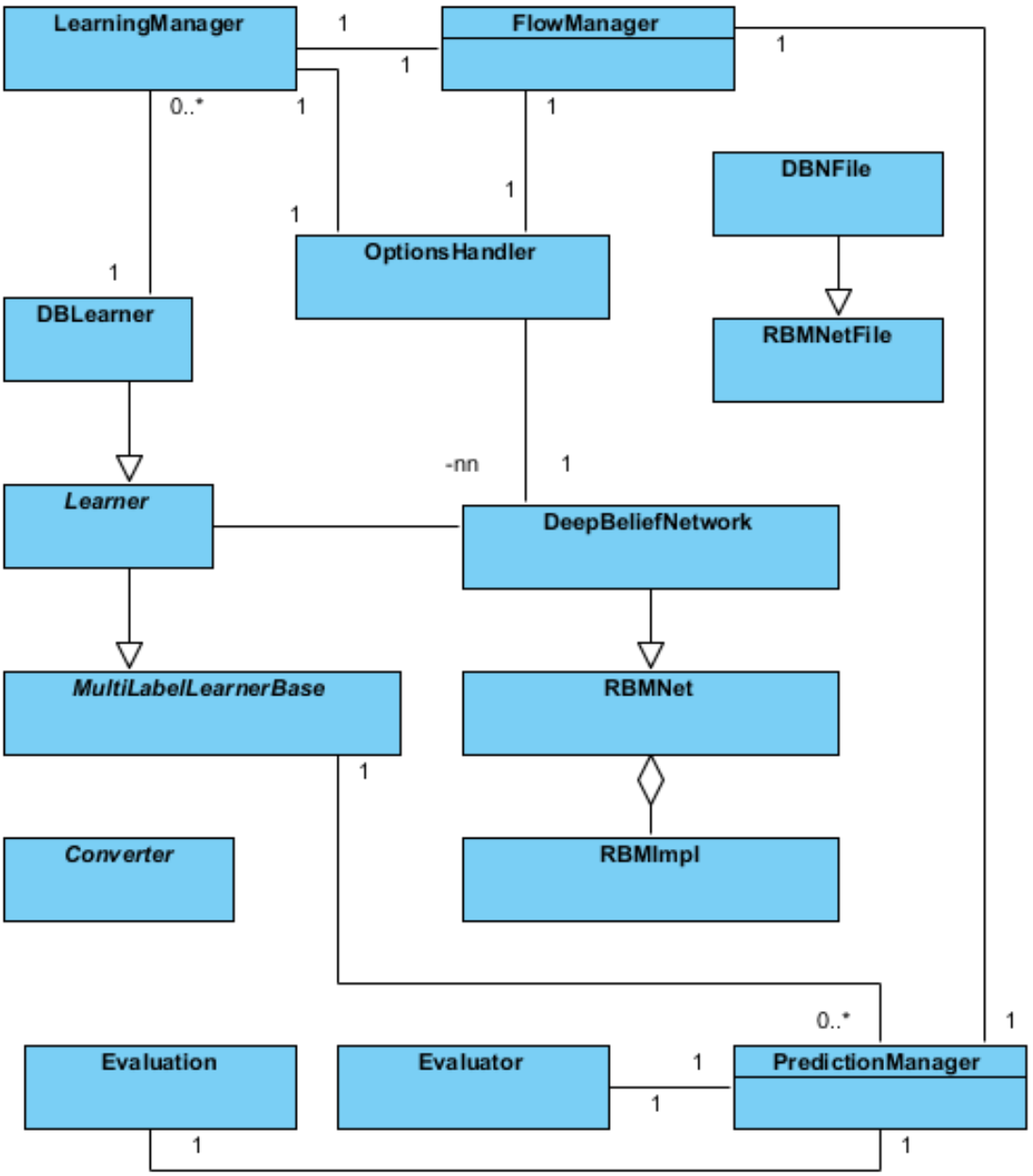
5.2 ΜΕΛΛΟΝΤΙΚΗ ΕΡΓΑΣΙΑ

Μελετώντας όλα αυτά που αναφέρονται στην παρούσα εργασία, σκεφτήκαμε κάποιες επεκτάσεις που θα μπορούσαν να γίνουν στα πλαίσια κάποιας εργασίας επέκτασης. Αρχικά, ξεκινήσαμε να ενσωματώσουμε μια τεχνική που θα χρησιμοποιούσε πυρήνες της GPU αντί της CPU, όμως λόγο κάποιων τεχνικών δυσκολιών αυτό ήταν αδύνατον. Σκεφτήκαμε, λοιπόν, ότι αφού η εργασία αυτή γράφτηκε σε JAVA, θα μπορούσε να χρησιμοποιήσει κάποιον πολυνηματικό (multi-thread) μηχανισμό σε JAVA ή την τεχνική Map-Reduce για την εκπαίδευση του νευρωνικού δικτύου. Αυτό είναι εφικτό λόγο της φύσης της εκπαίδευσης των RBM (βλ. Κεφάλαια 2.2.2, 2.2.3).

Μια δεύτερη επέκταση, θα μπορούσε να είναι η δημιουργία των κλάσεων RBM και $RBMLearne$, με σκοπό την διόρθωση λαθών του κώδικα της βιβλιοθήκης $jaRBM$. Αυτό θα ήταν μια καλή λύση στο πρόβλημα που δημιουργήθηκε στο Κεφάλαιο 4. Ο νέος κώδικας θα μπορούσε να αντικαταστήσει τη βιβλιοθήκη $jaRBM$ και να ενσωματωθεί στην εργασία, προσθέτοντας μια καλύτερη βιβλιοθήκη σε JAVA, πράγμα πολύ χρήσιμο αφού στο συγκεκριμένο τομέα, οι λύσεις που υπάρχουν σε JAVA είναι ελάχιστες.

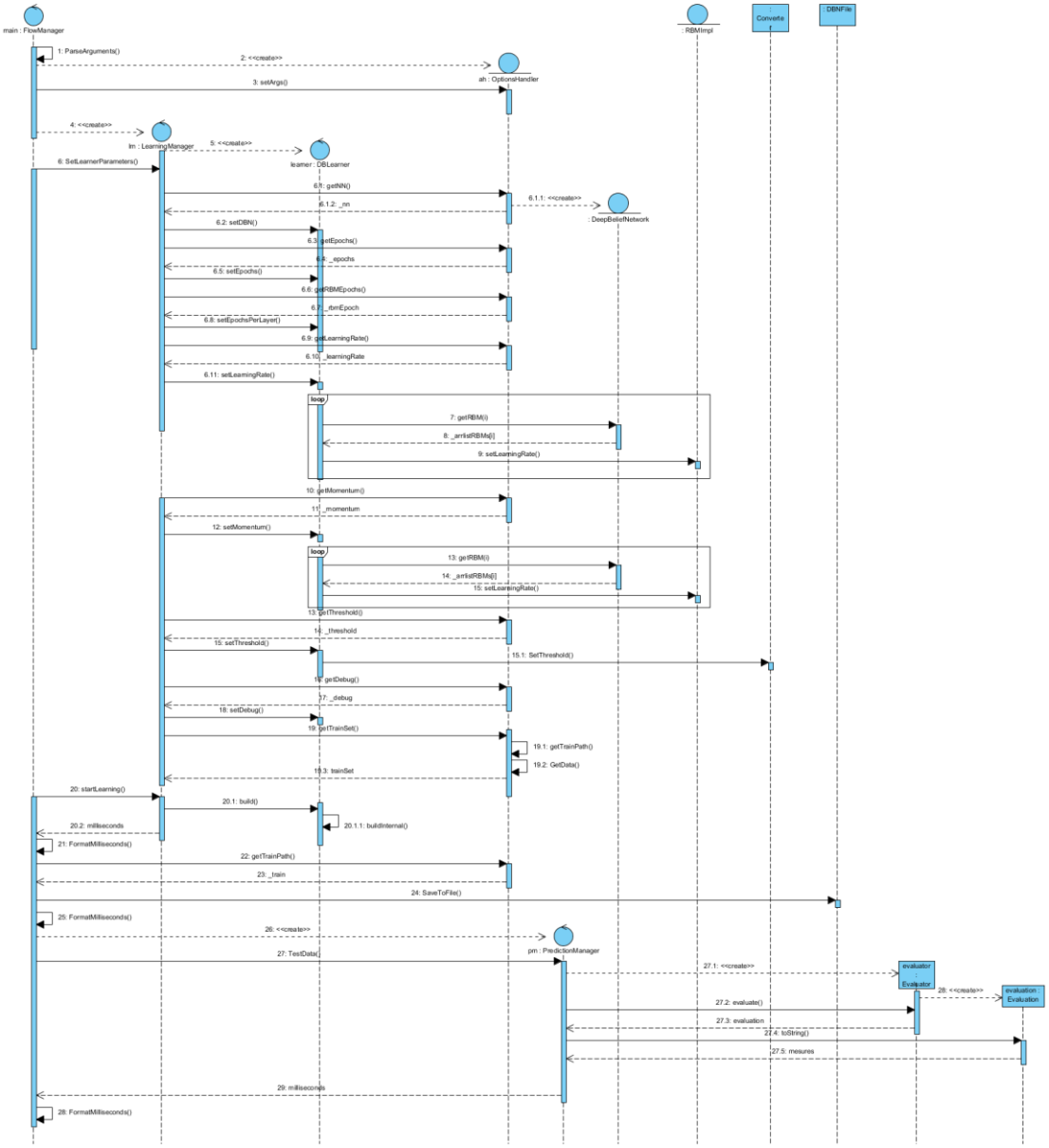
Όταν καταφέρουμε να διορθώσουμε τους αλγορίθμους εκπαίδευσης, οι κλάσεις της παρούσας εργασίας θα μπορούσαν να ενσωματωθούν στην ανοιχτή βιβλιοθήκη για αλγορίθμους μηχανικής μάθησης Mulan. Εφόσον οι κλάσεις του προγράμματος είναι πλήρως συμβατές με τα πρότυπα του Mulan, θα ήταν πολύ εύκολη η προσθήκη τους σε αυτήν.

ΠΑΡΑΡΤΗΜΑ Ι: ΔΙΑΓΡΑΜΜΑΤΑ UML



Σχήμα 5.2.1 Το διάγραμμα κλάσεων του προγράμματος που υλοποιήθηκε, σύμφωνα με τα πρότυπα της UML.

Το παραπάνω διάγραμμα υπάρχει με περισσότερη λεπτομέρεια στο internet. Λόγο μεγέθους δεν θα ήταν εύκολο να διαβαστεί αν το χρησιμοποιούσαμε εδώ. Μπορείτε να βρείτε το λεπτομερές διάγραμμα κλάσεων εδώ.



Σχήμα 5.2.2 Το διάγραμμα ακολουθίας του προγράμματος που υλοποιήθηκε, σύμφωνα με το πρότυπο της UML.

Το παραπάνω διάγραμμα περιγράφει τη βασική ροή της διαδικασίας εκπαίδευσης και πρόβλεψης του προγράμματος που υλοποιήθηκε. Μπορείτε να βρείτε το λεπτομερές διάγραμμα ακολουθίας εδώ.

ΠΑΡΑΡΤΗΜΑ ΙΙ: ΑΝΑΦΟΡΕΣ

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] D. H. Achley, G. E. Hinton, T. J. Sejnowski. *A learning algorithm for Boltzmann machines*. 1985.

[2] W. Chen, J. Yan, B. Zhang, Z. Chen, Q. Yang, *Document transformation for multi-label feature selection in text categorization*. Proc. 7th IEEE International Conference on Data Mining, Los Alamitos, CA, USA, IEEE Computer Society. 2007.

[3] K. Crammer, Y. Singer, *A Family of Additive Online Algorithms for Category Ranking*. School of Computer Science & Engineering, Hebrew University, Israel.

[4] A. Elisseeff, J. Weston, A *kernel method for multi-labeled classification*. Advances in Neural Information Processing Systems 14. 2002.

[5] J. Frunkraz, E. Hullermeiler, E. Loza Mencia, K. Brinker, *Multilabel classification via calibrated label ranking*. Machine Learning. 2008.

[6] S. Godbole, S. Sarawagi, *Discriminative methods for multi-labeled classification*. In: Proceedings of the 8th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2044). 2004.

[7] G. E. Hinton. *Training products of experts by minimizing contrastive divergence*. 2000.

[8] G. E. Hinton, R. R. Salakhutdinov. *Reducing the dimensionality of data with neural networks*. 2006.

[9] S. Haykin, *Νευρωνικά Δίκτυα και Μηχανική Μάθηση, 3η Έκδοση*. Εκδώσεις Παπασωτηρίου, Αθήνα 2010.

[10] E. Hullermeier, J. Furnkraz, W. Cheng, K. Brinker, *Label ranking by learning pairwise preferences*. Artificial Intelligence. 2008.

[11] E. Loza Mencia, J. Furnkranz, *Efficient Multi-label Classification Algorithms for Large-Scale Problems in the Legal Domain*. Knowledge Engineering Group, Technische Universitat Darmstadt.

[12] E. Loza Mencia, J. Furnkranz, *Pairwise learning of multilabel classifications with perseptrons*. 2008 IEEE International Joint Conference on Networks (IJCNN-08). Hong Kong, 2008.

[13] J. Read, *A pruned problem transformation method for multi-label classification*. Proc. 2008 New Zealand Computer Science Research Student Conference (NZCSRS 2008). 2008.

[14] J. Read, F. P. Cruz, *Deep Belief Networks for Multi-label Classification*. Department of Signal Theory and Communications Universidad Carlos III, Madrid 2003.

[15] M. A. Renzato, M. Szummer, *Semi-supervised Learning of Compact Document Representations with Deep Networks*. Courant Institute, New York University – USA, Microsoft Research Cambridge – UK.

[16] G. Tsoumakas, I. Katakis, I. Vlahavas, *Mining Multi-label Data*. Dept. of Informatics, Aristotle University of Thessaloniki.

[17] G. Tsoumakas, I. Vlahavas, *Random k-Labelsets for Multi-Label Classification*. IEEE Transactions on Knowledge and Data Engineering (TKDE 2010). 2010.

[18] G. Tur, L. Deg, D. Hakkani-Tür, X. He, *Towards Deeper Understanding: Deep Convex Networks for Semantic Utterance Classification*. Microsoft Speech Labs, Microsoft Research.

[19] M. L. Zhang, *ML-RBF: RBF Neural Networks for Multi-Label Learning*. Hohai University, China, 2009.

[20] M. L. Zhang, Z. H. Zhou, Multi-label learning by instance differentiation. Proceeding of the Twenty-Second AAAI Conference on Artificial Intelligence. Vancouver, British Columbia, Canada, AAAI Press, 2007.

[21] M. L. Zhang, Z. H. Zhou, *Neural Networks for Multi-Instance Learning*. National Laboratory for Novel Software Technology, Nanjing University, China.

[22] M. L. Zhang, Z. H. Zhou, Senior Member, *Multi-label Neural Networks with Applications to Functional Genomics and Text Categorization*. IEEE Transactions on Knowledge and Data Engineering.

WEB SITES

[1] 2007 NIPS tutorial on: Deep Belief Nets, Geoffrey Hinton, Canadian Institute for Advanced Research & Department of Computer Science, University of Toronto, http://www.cs.toronto.edu/~hinton/nipstutorial/nipstut3.pdf (τελευταία επίσκεψη 9 Ιουνίου 2013)

[2] Deep Learning, http://deeplearning.net/ (τελευταία επίσκεψη 22 Φεβρουαρίου 2013)

[3] Hinton's Neural Network Simulation (Generative): Ένας προσομοιωτής με live αποτελέσματα από ένα DBN: http://www.cs.toronto.edu/~hinton/adi/ (τελευταία επίσκεψη 17 Ιουνίου 2013)

[4] jaRBM: A Java library for Restricted Boltzmann Machines, http://sourceforge.net/projects/jarbm/ (τελευταία επίσκεψη 20 Ιουνίου 2013)

[5] Mulan: A Java Library for Multi-Label Learning, http://mulan.sourceforge.net/

[6] theano 0.6rc3 documentation, http://deeplearning.net/software/theano/ (τελευταία επίσκεψη 15 Φεβρουαρίου 2013)

[7] Η σελίδα του εργαλείου mPoT (Marc'Aurelio Ranzato et. al, 2010), http://www.cs.toronto.edu/~ranzato/publications/mPoT/mPoT.html (τελευταία επίσκεψη 20 Ιουνίου 2013)

[8] DeepLearnToolbox: Ένα εργαλείο για βαθιά μάθηση σε Matlab/Octave, https://github.com/rasmusbergpalm/DeepLearnToolbox (τελευταία επίσκεψη 20 Ιουνίου 2013)

[9] Εκπαίδευση ενός Auto-encoder ή άλλου ταξινομητή στα δεδομένα της βάσης MNIST, http://www.cs.toronto.edu/~hinton/MatlabForSciencePaper.html (τελευταία επίσκεψη 20 Ιουνίου 2013)

[10] Κώδικας για την εκπαίδευση RBM και DBN νευρωνικά δίκτυα σε Matlab με το εργαλείο matrbm, https://code.google.com/p/matrbm/ (τελευταία επίσκεψη 20 Ιουνίου 2013)

[11] Υψηλής απόδοσης C++/CUDA εφαρμογή συνελικτικών νευρωνικών δικτύων, https://code.google.com/p/cuda-convnet/ (τελευταία επίσκεψη 20 Ιουνίου 2013)

ΠΑΡΑΡΤΗΜΑ ΙΙΙ: ΑΚΡΩΝΥΜΑ

ΕΛΛΗΝΙΚΟΙ ΟΡΟΙ

Ακρώνυμο	Επεξήγηση
ΤΝΔ	Τεχνητό Νευρωνικό Δίκτυο

ΑΓΓΛΙΚΟΙ ΟΡΟΙ

Ακρώνυμο	Επεξήγηση
ADALINE	Adaptive Linear Neuron
ANN	Artificial Neural Network
ARFF	Attribute-Relation File Format
BM	Boltzmann Machine
BP	Back-Propagation
BP-MIP	Back-Propagation for Multi-Instance Problems
BP-MLL	Back-Propagation Multi-Label Learning
BR	Binary Relevance
CLR	Calibrated Label Ranking
CMLPP	Calibrated Multi-Label Pairwise Perceptron
CPU	Central Processing Unit
DBN	Deep Belief Network
GPU	Graphics Processing Unit
LP	Label Powerset
LR	Label Ranking
MLC	Multi-Label Classification
MLP	Multi-Layer Perceptron
MLPP	Multi-label Pairwise Perceptron
MLR	Multi-Label Ranking
MMP	Multi-class Multi-label Perceptron
Mulan	Multi-Label Learning
NN	Neural Network
PPT	Pruned Problem Transformation
RA*k*EL	RAndom k-Labelsets
RBF	Radius Basis Function

Ακρώνυμο	Επεξήγηση
RBM	Restricted Boltzmann Machine
RPC	Ranking by Pairwise Comparison
SVM	Support Vector Machine

ΠΑΡΑΡΤΗΜΑ IV: ΓΛΩΣΣΑΡΙΟ

ΓΛΩΣΣΑΡΙΟ

Όρος	Επεξήγηση
Artificial intelligence	Τεχνητή νοημοσύνη
Artificial neural network	Τεχνητό νευρωνικό δίκτυο
Back-propagation	Διάδοση προς τα πίσω
Bipartition	Δυαδικός διαχωρισμός
Boltzmann Machine	Μηχανή Boltzmann
Classification	Ταξινόμηση
Contrastive divergence	Αντιπαραβολική απόκλιση
Dataset	Σύνολο δεδομένων
Deep Belief Network	Δίκτυο Πεποίθησης Μεγάλου Βάθους
Epochs	Επαναλήψεις ενός αλγορίθμου μάθησης (εποχές)
Feed forward	Τροφοδότηση προς τα εμπρός
Gradient descent	Κεκλιμένη κάθοδος
Learning rate	Ρυθμός μάθησης
Momentum	Ορμή
Multi-label data	Δεδομένα πολλών Ετικετών
Multi-thread	Πολυνηματικό
Machine learning	Μηχανική μάθηση
Ranking	Κατάταξη
Restricted Boltzmann Machine	Περιορισμένη Μηχανή Boltzmann
Simulation annealing	Προσομοιωμένη ανόπτηση