ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Τμήμα Διδακτικής της Τεχνολογίας και Ψηφιακών Συστημάτων

ΣΥΣΤΗΜΑΤΑ ΑΝΙΧΝΕΥΣΗΣ ΠΑΡΕΙΣΦΡΗΣΕΩΝ ΜΕ ΧΡΗΣΗ ΕΞΕΛΙΚΤΙΚΩΝ ΝΕΥΡΩΝΙΚΩΝ ΔΙΚΤΥΩΝ

ΜΙΧΑΗΛΙΔΗΣ ΕΜΜΑΝΟΥΗΛ

Η εργασία υποβάλλεται στο πλαίσιο της απόκτησης Μεταπτυχιακού Διπλώματος Σπουδών στην Διδακτική της Τεχνολογίας και Ψηφιακών Συστημάτων

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ:

ΚΑΤΣΙΚΑΣ ΣΩΚΡΑΤΗΣ

Περίληψη

Ένα από τα μεγαλύτερα προβλήματα που καλούνται να αντιμετωπίσουν τα σύγχρονα πληροφοριακά συστήματα είναι η αντιμετώπιση νέων και άγνωστων απειλών εναντίον της ασφάλειάς τους. Πρόκειται συνήθως για παραλλαγές γνωστών τύπων απειλών που είναι όμως δύσκολο να αναγνωριστούν από τους μηχανισμούς ασφαλείας. Η δυσκολία αντιμετώπισης συνίσταται στην έλλειψη γνώσης για τα ακριβή χαρακτηριστικά τους. Η Τεχνητή Νοημοσύνη διαθέτει μηχανισμούς που έχουν τη δυνατότητα όχι μόνο να αναγνωρίζουν και να ταξινομούν πρότυπα απειλών αλλά και να γενικεύουν, αντιμετωπίζοντας έτσι τις διάφορες παραλλαγές που υπάρχουν. Τέτοιοι μηχανισμοί είναι τα Νευρωνικά Δίκτυα.

Στην εργασία αυτή μοντελοποιείται η μηχανή ανάλυσης και αναγνώρισης επιθέσεων ενός Συστήματος Ανίχνευσης Παρεισφρήσεων από ένα Εξελικτικό Νευρωνικό δίκτυο, με σκοπό να αξιολογηθεί η ικανότητά του να αναγνωρίσει νέες κυρίως και άγνωστες επιθέσεις στο περιβάλλον ενός δικτύου. Το Εξελικτικό Νευρωνικό Δίκτυο εκπαιδεύεται με δεδομένα που χρησιμοποιήθηκαν στον διεθνή διαγωνισμό KDD cup ‘99, και αξιολογείται με βάση τα αποτελέσματα του ίδιου διαγωνισμού. Τα πειράματα που πραγματοποιήθηκαν έδωσαν ιδιαίτερα θετικά αποτελέσματα στην αναγνώριση κυρίως επιθέσεων τύπου DoS και Probe.

Λέξεις κλειδιά:

Συστήματα Ανίχνευσης Παρεισφρήσεων, Intrusion Detection Systems, NIDS, Δικτυακά Συστήματα Ανίχνευσης Παρεισφρήσεων, Anomaly detection, Εντοπισμός ανωμαλιών συστήματος, Παρείσφρηση, Ανίχνευση Παρεισφρήσεων, Νευρωνικά Δίκτυα, Εξελικτικά Νευρωνικά Δίκτυα, Βελτιστοποίηση Νευρωνικών Δικτύων με Σμήνη Σωματιδίων.

Ευχαριστίες

Θερμές ευχαριστίες στον Καθηγητή Σωκράτη Κάτσικα για τις πολύτιμες συμβουλές καθ' όλη τη διάρκεια διερεύνησης του θέματος και εκπόνησης της εργασίας.

Ευχαριστίες οφείλω στου συνεξεταστές, τον Λέκτορα Χρήστο Ξενάκη και το Λέκτορα Ιωάννη Παραβάντη για την βήθειά τους στην ολοκλήρωση της εργασίας.

Θέλω να ευχαριστήσω επίσης τον Επίκουρο Καθηγητή του ΑΤΕΙ Καλαμάτας Ευστράτιο Γεωργόπουλο, για την βοήθειά του σχετικά με τα Νευρωνικά Δίκτυα.

Ευχαριστίες προς τα παιδιά μου Μαρία και Φώτη για την υπομονή τους, ιδιαίτερα όμως προς τη σύζυγό μου Δέσποινα για τη μεγάλη της στήριξη όλο αυτό τον καιρό.

Κατάλογος Πινάκων

Κατάλογος Σχημάτων

Συντομογραφίες

IDS	Intrusion Detection Systems
PSO	Particle Swarm Optimion
KDD	International Knowledge Discovery and Data Mining Tools Competition
CERT/CC	Computer Emergency Response Team/ Coordination Center
NN	Neural Networks
ENN	Evolutionary Neural Network
R2L	Remote to Local
U2R	User to Remote

1. ΓΕΝΙΚΗ ΕΙΣΑΓΩΓΗ

Η έκρηξη του διαδικτύου τις τελευταίες δύο δεκαετίες δημιούργησε νέες προοπτικές σε τομείς όπως το εμπόριο, η οικονομία, οι μεταφορές, η βιομηχανία και η υγεία. Όμως ταυτόχρονα αυξήθηκαν και οι απαιτήσεις για περισσότερη ασφάλεια στις συναλλαγές και στη διακίνηση των προϊόντων και των υπηρεσιών. Η βιομηχανία της πληροφορικής ανταποκρίθηκε στις απαιτήσεις αυτές, δημιουργώντας σύγχρονα και ασφαλή λειτουργικά συστήματα, αντιικά προγράμματα, firewalls, συστήματα ανίχνευσης παρεισφρήσεων κλπ. Όμως, παρ' όλα αυτά, καθημερινά γινόμαστε μάρτυρες παραβίασης ασφάλειας συστημάτων, δικτυακών επιθέσεων, διάδοσης καταστροφικών ιών, υποκλοπών προσωπικών δεδομένων κλπ.

Η απάντηση στο τι συμβαίνει μπορεί να δοθεί με τη βοήθεια της ιατρικής: παρά τα μεγάλα άλματα που έχουν γίνει στην ιατρική, συνεχίζουμε να αρρωσταίνουμε. Αυτό γιατί ενόσω δημιουργούνται φάρμακα, νέες ασθένειες εμφανίζονται συνεχώς, που συνήθως είναι βελτιωμένες εκδοχές παλαιών. Τα σύγχρονα πληροφορικά συστήματα αντιμετωπίζουν ακριβώς αυτό το πρόβλημα: «νέες ασθένειες». Νέες απειλές με την μορφή ιών, Trojan Horses, νέες εξελιγμένες τεχνικές από intruders και hackers, ο κατάλογος είναι μεγάλος.

Σε γενικές γραμμές υπάρχει τεχνολογική κάλυψη-ας μη ξεχνάμε και τον παράγοντα άνθρωπο στην υπόθεση ασφάλεια- για να αντιμετωπιστούν οι διάφορες απειλές αφού εντοπιστούν και αναλυθούν, όμως τότε συνήθως το κακό έχει ήδη γίνει.

Αν δεχτούμε ότι έχουν εντοπιστεί όλα τα είδη απειλών που μπορεί να υπάρξουν και ότι κάθε νέα απειλή είναι παραλλαγή μιας παλιάς και αφήσουμε έξω τον ανθρώπινο παράγοντα στην υπόθεση ασφάλεια τότε η λύση θα μπορούσε είναι η δημιουργία ενός έξυπνου μηχανισμού που να μπορεί να εντοπίζει πρότυπα ή γενικεύσεις απειλών.

Έχει εκδηλωθεί μεγάλο ερευνητικό ενδιαφέρον τα τελευταία χρόνια προς αυτή την κατεύθυνση. Αναζητούνται τρόποι έτσι ώστε τα Συστήματα Ανίχνευσης Παρεισφρήσεων- οι κατ' εξοχήν μηχανισμοί εντοπισμού απειλών-να είναι σε θέση να αναγνωρίζουν τις νέες απειλές.

Μέχρι στιγμής έχουν αναπτυχθεί δύο φιλοσοφίες ως προς την αναγνώριση απειλών . Η μια ανιχνεύει τις απειλές ή την παραβίαση της ασφάλειας με άμεσο τρόπο. Τα συστήματα ασφαλείας είναι εφοδιασμένα με πληροφορίες για ήδη υπάρχουσες και γνωστές απειλές και απλά ανατρέχουν σε αυτές για να αναγνωρίσουν την ύπαρξη τους. Αυτός ο τρόπος όμως έχει μια σοβαρή αδυναμία. Ενώ είναι αποτελεσματικός στην αντιμετώπιση γνωστών απειλών, δεν μπορεί να αναγνωρίσει νέες επιθέσεις για τις οποίες δεν υπάρχει καμία προηγούμενη πληροφορία.

Ο άλλος τρόπος είναι να μοντελοποιηθεί η κανονική λειτουργία του πληροφοριακού συστήματος. Ως κανονική λειτουργία θεωρείται οποιαδήποτε λειτουργία πραγματοποιείται σε απολύτως ελεγχόμενο περιβάλλον όπου ισχύουν οι τρεις θεμελιώδεις κανόνες ασφάλειας: ακεραιότητα, εμπιστευτικότητα και διαθεσιμότητα. Έτσι, κάθε «διαταραχή» ή «παρέκκλιση» από αυτή την κανονική λειτουργία μπορούμε να την εκλάβουμε ως παραβίαση της ασφάλειας. Αυτός είναι ένας από

τους τρόπους να διαπιστωθεί κάποια παραβίαση της ασφαλείας ενός συστήματος: η ανίχνευση παρέκκλισης από την κανονική λειτουργία. Για παράδειγμα, δικτυακή κίνηση με ασυνήθιστο ρυθμό ή περιεχόμενο, μια κάπως περίεργη εγγραφή σε κάποιο αρχείο καταγραφής του λειτουργικού συστήματος ή ένας ασυνήθιστος συνδυασμός κλήσεων συστήματος αποτελούν αδιάψευστα πειστήρια ότι υπήρξε ή υπάρχει σε εξέλιξη κάποια παραβίαση ασφάλειας του πληροφοριακού συστήματος.

Στην εποχή που τα πληροφοριακά συστήματα και οι αυτοματισμοί εξελίσσονται με ραγδαίους ρυθμούς, αυτό που έχουμε ανάγκη είναι η προστασία όχι μόνο από γνωστές από το παρελθόν απειλές, αλλά μηχανισμούς πού να είναι σε θέση να εντοπίζουν και να αντιδρούν σε νέους κινδύνους και απειλές.

1.1 Σκοπός

Ο σκοπός αυτής της εργασίας είναι η μελέτη του προβλήματος της αναγνώρισης δικτυακών παρεισφρήσεων ή επιθέσεων για τα πληροφοριακά συστήματα. Για την επίτευξη του στόχου γίνεται αρχικά μια μελέτη των συστημάτων τα οποία υπάρχουν για τον σκοπό αυτό, τα γνωστά ως Συστήματα Ανίχνευσης Παρεισφρήσεων (Intrusion Detection Systems IDS). Μέσα από βιβλιογραφική ανασκόπηση περιγράφεται το ιστορικό της δημιουργίας τους και η εξέλιξή τους, οι διάφορες αρχιτεκτονικές προσεγγίσεις που έχουν προταθεί, τα είδη των απειλών που έχουν να αντιμετωπίσουν καθώς και τα προβλήματα που έχουν παρουσιαστεί από τη λειτουργία τους.

Στη συνέχεια και για την καλύτερη μελέτη των συστημάτων αυτών υλοποιούμε ένα μοντέλο μηχανής ανάλυσης. Η μηχανή ανάλυσης αποτελεί τμήμα των IDS στο οποίο βρίσκεται ο αλγόριθμος που αναλύει και ανιχνεύει τα δεδομένα με σκοπό να εντοπίσει επιθέσεις. Τα δεδομένα αυτά μπορεί να προέρχονται από την κίνηση στο δίκτυο ή από κάποιο αρχείο καταγραφής. Το μοντέλο αυτό υλοποιείται με τη βοήθεια ενός Νευρωνικού Δικτύου το οποίο εκπαιδεύεται από έναν εξελικτικό αλγόριθμο. Τα δεδομένα που χρησιμοποιούμε για την εκπαίδευσή του προέρχονται από τον διαγωνισμό KDD 99 cup, όπου παρόμοια συστήματα εκπαιδεύτηκαν και αξιολογήθηκαν με σκοπό την ανάδειξη της καλύτερης τεχνικής για τον εντοπισμό παρεισφρήσεων. Τα ίδια τα αποτελέσματα αυτά χρησιμοποιήθηκαν για την αξιολόγηση της πρότασης αυτής εδώ της εργασίας και παρουσιάζονται σε ξεχωριστό κεφάλαιο στο τέλος της εργασίας.

1.2 Δομή της εργασίας

Στο πρώτο κεφάλαιο γίνεται μια γενική εισαγωγή πάνω στο θέμα της εργασίας, αναλύεται ο σκοπός για τον οποίο έχει γίνει και παρουσιάζεται η δομή της.

Στο δεύτερο κεφάλαιο γίνεται μια βιβλιογραφική επισκόπηση των Συστημάτων Ανίχνευσης Παρεισφρήσεων. Παρουσιάζονται οι διαφορετικές αρχιτεκτονικές προσεγγίσεις, τα χαρακτηριστικά τους και γίνεται μια κατηγοριοποίηση βάσει αυτών των χαρακτηριστικών. Στη συνέχεια γίνεται μια παρουσίαση όλων των γνωστών τύπων επιθέσεων, όπως αυτές έχουν εντοπιστεί και κατηγοριοποιηθεί στη διεθνή

βιβλιογραφία – μιας και ο μηχανισμός που αναπτύσσεται αποτελεί μέρος ενός τέτοιου συστήματος – καθώς και των Εξελικτικών Νευρωνικών Δικτύων, αλλά και του αλγορίθμου Βελτιστοποίησης με Σμήνη Σωματιδίων ο οποίος εκπαιδεύει το Νευρωνικό Δίκτυο.

Στο τρίτο κεφάλαιο περιγράφεται ο αρχιτεκτονικός σχεδιασμός του Νευρωνικού Δικτύου και η μεθοδολογία υλοποίησής του. Επίσης γίνεται μια περιγραφή των βασικών λειτουργιών του καθώς και περιγραφή του user interface. Η παρουσίαση του κώδικα καθώς και η τεκμηρίωσή του, περιλαμβάνονται στα Παραρτήματα Α και Β. Στο ίδιο κεφάλαιο γίνεται περιγραφή του διαγωνισμού KDD 99, μιας και τα δεδομένα για την εκπαίδευση του Νευρωνικού Δικτύου καθώς και τα αποτελέσματά του χρησιμοποιούνται στην εργασία αυτή ως κριτήρια αξιολόγησης.

Στο τέταρτο κεφάλαιο εκτελούνται αρχικά δοκιμές με σκοπό την εύρεση των βέλτιστων παραμέτρων του Εξελικτικού Νευρωνικού Δικτύου. Αφού εντοπιστούν οι βέλτιστες τιμές, δημιουργούνται τα κατάλληλα σύνολα δεδομένων για την εκπαίδευση και αξιολόγηση του αλγορίθμου και τέλος εκτελούνται τα πειράματα και καταγράφονται τα αποτελέσματα.

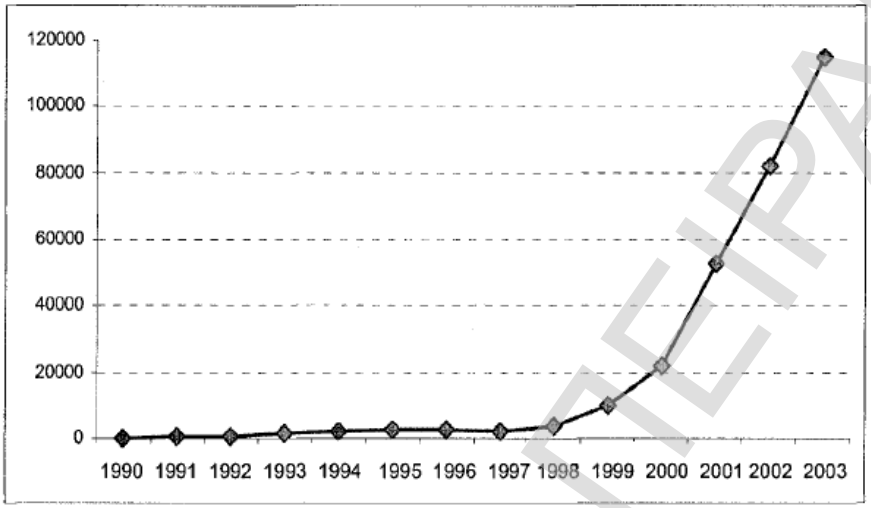
Στο πέμπτο και τελευταίο κεφάλαιο γίνεται μια ανάλυση και αξιολόγηση των πειραματικών αποτελεσμάτων καθώς και σύγκριση με άλλες παρόμοιες προσπάθειες με σκοπό την εξαγωγή χρήσιμων συμπερασμάτων. Τέλος παρουσιάζονται θέματα για περαιτέρω μελέτη.

2. ΒΙΒΛΙΟΓΡΑΦΙΚΗ ΕΠΙΣΚΟΠΗΣΗ

2.1 Εισαγωγή

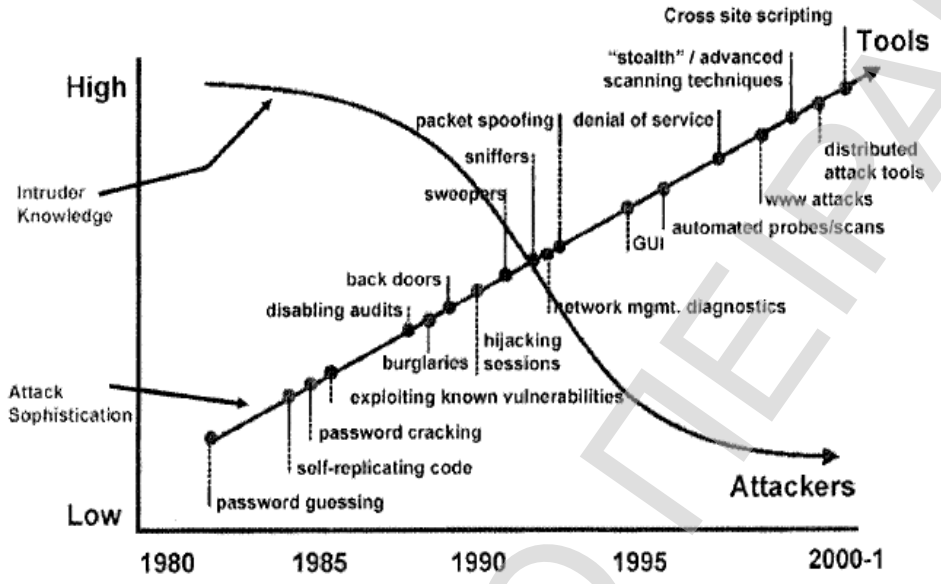
Το 2004 η αντιμετώπιση της απειλής των διάφορων ιών κόστισε περίπου 15 εκατ. Ευρώ μόνο στη Μεγάλη Βρετανία. Η διαρκώς διευρυμένη συνδεσιμότητα μεταξύ των υπολογιστών μεγαλώνει συνεχώς το κόστος αυτό. Είναι συνεπώς επιτακτική η ανάγκη της αντιμετώπισης των διάφορων ιών αλλά και γενικότερα των διάφορων απειλών ασφάλειας. Ωστόσο «η αγορά διαπιστώνει ότι τα μέτρα για την ασφάλεια των υπολογιστικών συστημάτων είναι ακόμα ανεπαρκή» (BBC news, 2004).

Είναι γεγονός ότι νέες απειλές για την ασφάλεια των πληροφοριακών συστημάτων δημιουργούνται καθημερινά. Όπως αναφέρεται από την CERT/CC (Computer Emergency Response Team/ Coordination Center), ο αριθμός των επιθέσεων σε συστήματα αυξάνεται εκθετικά (Εικόνα 1).



Εικόνα 1. **Ρυθμός ανάπτυξης επιθέσεων καταγεγραμμένος από τον οργανισμό** CERT/CC (CERT/CC, 2004)

Αυτό που έχει μεγαλύτερη σημασία είναι το γεγονός ότι η σοβαρότητα αλλά και η πολυπλοκότητα των επιθέσεων έχουν αυξηθεί σημαντικά (Εικόνα 2). Για παράδειγμα η επίθεση με το «σκουλήκι» Slammer/Sapphire υπήρξε η γρηγορότερα εξελισσόμενη στην ιστορία: από την στιγμή που διαδόθηκε στο internet διπλασιαζόταν σε μέγεθος κάθε 8,5 sec με αποτέλεσμα να μολύνει περισσότερους από 75.000 υπολογιστές προκαλώντας ανυπολόγιστες ζημίες στις μεταφορές (ακυρώσεις πτήσεων), και στην οικονομία (δημοπρασίες, ATM βλάβες). Στο παρελθόν υπήρχε η ανάγκη για πολύ καλή γνώση των υπολογιστών και των δικτύων για να πραγματοποιηθεί κάποια επίθεση. Σήμερα σχεδόν ο καθένας, οποιαδήποτε στιγμή, μπορεί να κάνει χρήση των εγγενών αδυναμιών των συστημάτων, εξ αιτίας κυρίως της ύπαρξης πληθώρας εργαλείων που μπορεί να βρει εύκολα στο διαδίκτυο.



Εικόνα 2. **Πολυπλοκότητα επιθέσεων ως προς τις τεχνικές γνώσεις των εισβολέων (Incidents Reported to Computer Emergency Response Team/Coordination Center(CERT/CC, 2004))**

Η συνηθισμένη πρακτική για την ασφάλεια υπολογιστικών συστημάτων είναι η χρήση μηχανισμών όπως τα firewalls, οι μηχανισμοί πιστοποίησης χρήστη (authentication mechanisms) και τα VPN's. Οι μηχανισμοί αυτοί δημιουργούν μια ασπίδα προστασίας γύρω από τα συστήματα. Όμως είναι αρκετά συνηθισμένο να υπάρχουν κενά και αδυναμίες που αφορούν την ασφάλεια αυτών των ίδιων των μηχανισμών.

Ένα από τα πλέον συνηθισμένα προληπτικά μέτρα που μπορεί να λάβει ένας οργανισμός ή μια επιχείρηση για τη διαφύλαξη της ασφάλειάς της είναι τα Συστήματα Ανίχνευσης Παρεισφρήσεων (Intrusion Detection Systems - IDS). Ένα τέτοιο σύστημα μπορεί να υλοποιηθεί ως λογισμικό ή να αποτελεί μια αυτόνομη

συσκευή. Ο σκοπός της λειτουργίας του είναι να ανιχνεύει κάθε μορφή παραβίασης της ασφάλειας ενός πληροφοριακού συστήματος με τρόπο αποτελεσματικό. Αποτελεσματικός τρόπος θεωρείται η έγκαιρη αναγνώριση εισβολών ή επιθέσεων ακόμα και σε περιπτώσεις που αυτές δεν έχουν αναγνωριστεί ποτέ στο παρελθόν. Επίσης θα πρέπει να είναι όσο το δυνατό λιγότερες ή ακόμα και μηδενικές οι λάθος εκτιμήσεις (false positive, false negative).

2.2 Συστήματα Ανίχνευσης Παρεισφρήσεων (Intrusion Detection Systems - IDS)

Το National Institute of Standards and Technology (Base and Mell) ορίζει την ανίχνευση παρεισφρήσεων ως «τη διαδικασία παρακολούθησης και ανάλυσης συμβάντων σε υπολογιστικό σύστημα ή δίκτυο με σκοπό την αναγνώριση κακόβουλων προσπαθειών που ως στόχο έχουν την παραβίαση των μηχανισμών ασφαλείας».

Ο σκοπός των IDS είναι θεωρητικά απλός: να ανιχνεύουν παρεισφρήσεις. Στην πράξη όμως αυτό είναι δύσκολο μιας και στην πραγματικότητα αυτό που κάνουν είναι η συλλογή «τεκμηρίων» ή «ενδείξεων» είτε όταν αυτές βρίσκονται σε εξέλιξη είτε εκ των υστέρων. Όταν οι «ενδείξεις» αυτές πληθαίνουν είναι πιθανό να πάρουν τέτοια μορφή που να μπορεί να ειπωθεί ότι εκδηλώνεται κάποια επίθεση. Όμως, για να αποφασίσει το IDS αν πρόκειται για επίθεση, απαραίτητη προϋπόθεση είναι να υπάρχει επαρκής πληροφορία. Για παράδειγμα, αν η κάμερα ασφαλείας σε ένα

κατάστημα είναι σκονισμένη, ενδεχομένως δεν παρέχει επαρκή πληροφορία για το αν το πρόσωπο που βρίσκεται απέναντι είναι ο ιδιοκτήτης ή κάποιος άλλος.

Για αποτελεσματική ανίχνευση παρεισφρήσεων θα πρέπει να υπάρχουν δεδομένα πλήρη και έγκυρα από τις δραστηριότητες του υπό παρακολούθηση συστήματος. Αυτό είναι ένα αρκετά περίπλοκο θέμα, είτε πρόκειται για ένα απλό υπολογιστή είτε για ολόκληρο δίκτυο. Τα περισσότερα λειτουργικά συστήματα προσφέρουν τη δυνατότητα δημιουργίας αρχείων καταγραφής (log files) για κάθε είδους δραστηριότητα. Αυτά τα αρχεία μπορεί να περιορίζονται μόνο σε θέματα ασφαλείας ή να καταγράφουν το σύνολο των δραστηριοτήτων του χρήστη ή και του ίδιου του συστήματος. Όμοια, δρομολογητές ή τα firewalls έχουν τη δυνατότητα καταγραφής διάφορων δικτυακών γεγονότων, είτε σε απλή μορφή όπως το άνοιγμα ή το κλείσιμο κάποιας σύνδεσης είτε σε λεπτομερή μορφή με την καταγραφή του κάθε διερχόμενου δικτυακού πακέτου. Το πρόβλημα σε αυτή την περίπτωση είναι ποια θεωρείται ως επαρκής πληροφορία, αφού δημιουργείται επιβάρυνση στα συστήματα με τη συσσώρευση όλων αυτών των δεδομένων. Για παράδειγμα, ένα πλήρες αρχείο καταγραφής δικτυακής κίνησης στη διάρκεια μιας μέρας μπορεί να χρειαστεί εκατοντάδες Gigabytes αποθηκευτικό χώρο.

Οι διάφοροι μηχανισμοί ασφαλείας σχεδιάζονται έτσι ώστε να εμποδίζουν τη μη εξουσιοδοτημένη πρόσβαση στα διάφορα συστήματα. δίκτυα κλπ. Προς το παρόν όμως η απόλυτη παρεμπόδιση τέτοιων προσπαθειών είναι μάλλον ανέφικτη. Αυτό που μπορούμε να κάνουμε είναι να γνωρίσουμε πότε εξελίσσεται μια τέτοια προσπάθεια έτσι ώστε να την αντιμετωπίσουμε με όσο το δυνατό μικρότερο κόστος.

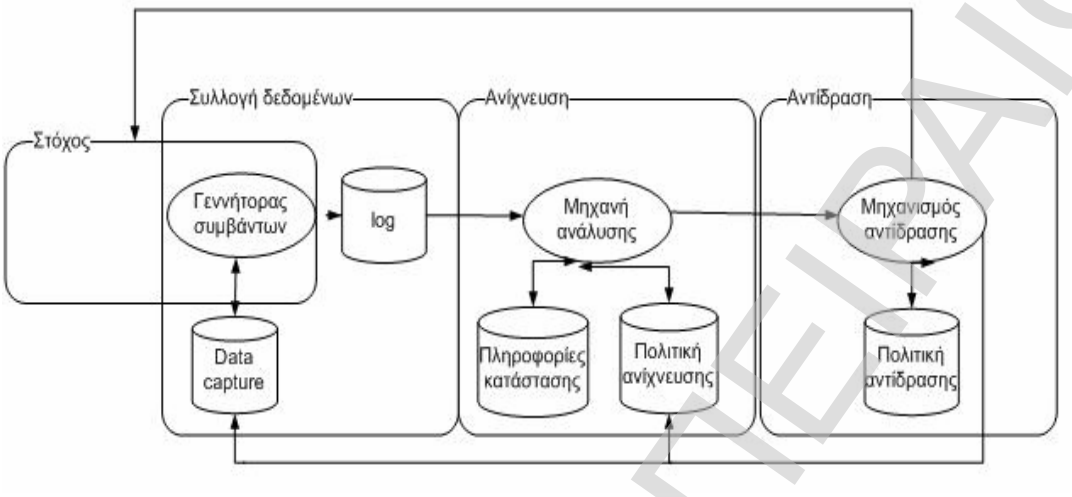
Το πεδίο έρευνας που ασχολείται με τις μεθόδους έγκαιρης αναγνώρισης και αντιμετώπισης της παράκαμψης της ασφάλειας ενός συστήματος λέγεται «Ανίχνευση Παρεισφρήσεων» (Intrusion Detection). Θα μπορούσε να αναρωτηθεί κανείς για την αναγκαιότητα ύπαρξης ενός τέτοιου αντικειμένου δεδομένου ότι έχουν αναπτυχθεί πιο άμεσοι τρόποι διασφάλισης της ακεραιότητας και της πρόσβασης σε δεδομένα όπως τα αντιικά, τα διάφορα firewalls, τα αναχώματα ασφαλείας κλπ. Σχετικά με αυτήν την παρατήρηση αξίζει να καταγράψουμε μερικές παρατηρήσεις. Αρχικά θα πρέπει να πούμε ότι στην πράξη είναι αδύνατο να φτιαχτεί ένα απολύτως ασφαλές σύστημα. Η μελέτη του Katsikas (Katsikas et al 1996) καταδεικνύει την όλο και μεγαλύτερη σχέση που υπάρχει ανάμεσα στον κίνδυνο και τη συνεχώς αυξανόμενη αυτοματοποίηση, τόσο σε επίπεδο διαδικασιών όσο και σε επίπεδο συστημάτων. Ο Miller (Miller et al, 2000) κάνει μια καταγραφή όλων των αδυναμιών που έχουν τα σύγχρονα συστήματα ασφαλείας αλλά και τα μεγάλα πακέτα λογισμικού. Μια παρόμοια λίστα, αλλά πιο σύγχρονη και ενημερωμένη, μπορεί κανείς να βρει στα διάφορα response teams (CERTs), τα οποία είναι ανεξάρτητοι φορείς υπό κρατική εποπτεία που έχουν ως σκοπό τη παροχή γνώσης σε χρήστες και διαχειριστές συστημάτων σε σχέση με αδυναμίες και bugs που υπάρχουν καθώς και τρόπους αντιμετώπισης. Φυσικά αυτό που κάνει εντύπωση είναι το πλήθος των καταγεγραμμένων αδυναμιών. Άρα λοιπόν το να εγκατασταθεί ένα αντιικό σε ένα λειτουργικό σύστημα που έχει προβλήματα ασφαλείας δε σημαίνει και πολλά πράγματα. Η κρυπτογραφία, ένα σημαντικό όπλο για την ασφάλεια των συστημάτων, έχει το σημαντικό μειονέκτημα της πολυπλοκότητας στην εφαρμογή. Σε συστήματα με μεγάλο βαθμό ασφαλείας προς τα έξω, υπάρχει πάντα ο κίνδυνος εκ των έσω,

δηλαδή από κάποιο εξουσιοδοτημένο χρήστη που προσπαθεί να κάνει κατάχρηση των δικαιωμάτων του (privileges).

Σύμφωνα με τα παραπάνω μπορούμε να συμπεράνουμε ότι δεν υπάρχει αυτή της στιγμή μια συνολική και απόλυτη λύση για την ασφάλεια των πληροφοριακών συστημάτων αλλά η διαφύλαξή της προϋποθέτει μια συνεχή προσπάθεια φύλαξης και παρατήρησης των συστημάτων. Ένα όπλο σε αυτή την προσπάθεια είναι τα Συστήματα Ανίχνευσης Παρεισφρήσεων (Intrusion Detection Systems - IDS).

2.2.1 Η αρχιτεκτονική των ID συστημάτων

Το πρώτο μοντέλο για ανίχνευση παρεισφρήσεων υλοποιήθηκε από την Dorothy Denning (D. Denning 1987) στο εργαστήριο του Stanford (SRI International). Από τότε έχουν προταθεί ένα πλήθος από υλοποιήσεις IDS, τόσο σε ερευνητικό όσο και σε εμπορικό επίπεδο (CERIAS Intrusion Detection Resources 2004, Google Directory, Meier. and Sobirey, SecurityTechNet.com 2005).



Εικόνα 3. **Γενική αρχιτεκτονική ανάλυση ενός Intrusion Detection System (Lundin , Jonsson ,2004)**

Παρότι τα συστήματα αυτά διαφέρουν αρκετά μεταξύ τους, έχουν κοινό αρχιτεκτονικό πλαίσιο (Εικόνα 3). Στη συνέχεια γίνεται αναλυτική παρουσίαση των υποσυστημάτων ενός τυπικού IDS (Lundin, Johnsson,2004).

– *Υποσύστημα Συλλογής Δεδομένων (Data collection Unit).* Αρχικά υπάρχει το υποσύστημα συλλογής δεδομένων, όπου το σύστημα στόχος (target system) έχει εκείνους τους μηχανισμούς που απαιτούνται για τη συλλογή των δεδομένων, όπως είναι τα δεδομένα που κινούνται στο δίκτυο (data capture), τα αρχεία καταγραφής (log files) του λειτουργικού συστήματος και τα logs αρχεία των εφαρμογών. Ο γεννήτορας συμβάντων είναι ο κατ' εξοχήν μηχανισμός που λαμβάνει τα δεδομένα από τα αρχεία καταγραφής ή τα δεδομένα από το δίκτυο. Στον γεννήτορα συμβάντων μπορεί να γίνεται και κάποιου είδους προεπεξεργασία με σκοπό τη

μετατροπή ή το φιλτράρισμα των δεδομένων και κατόπιν η αποθήκευσή τους και η αποστολή στο υποσύστημα ανάλυσης.

- *Υποσύστημα Ανίχνευσης* ***(Analysis Engine).*** Στο υποσύστημα αυτό υλοποιείται ο αλγόριθμος ανίχνευσης. Διακρίνονται δύο διαφορετικές φιλοσοφίες αναφορικά με τον τρόπο που γίνεται η ανίχνευση: Η πρώτη χρησιμοποιεί τεχνικές «ταιριάσματος προτύπων» (pattern matching) ενώ στη βιβλιογραφία συχνά αναφέρεται με τον όρο misuse detection. Σύμφωνα με τη φιλοσοφία αυτή, για τον εντοπισμό επιθέσεων αναζητείται κάποια γνωστή υπογραφή (signature) στα δεδομένα, είτε αυτά προέρχονται από κίνηση στο δίκτυο είτε από κάποια Log αρχεία. Αυτή η υπογραφή συγκρίνεται με υπογραφές γνωστών τύπων επιθέσεων έτσι ώστε να ταυτοποιηθεί η επίθεση. Μια συνηθισμένη υλοποίηση είναι με έμπειρα συστήματα, τα οποία μπορούν να ανταποκριθούν και να εντοπίσουν ιδιαίτερα πολύπλοκες υπογραφές. Κοινό χαρακτηριστικό αυτής της φιλοσοφίας είναι το γεγονός ότι για να είναι αποτελεσματική η ανίχνευση θα πρέπει το IDS να είναι ενημερωμένο με υπογραφές ήδη γνωστών επιθέσεων. Η δεύτερη φιλοσοφία κάνει την παραδοχή ότι κάθε σύστημα που παρακολουθείται έχει δύο ειδών συμπεριφορές: μία που χαρακτηρίζεται ως κανονική (normal) και αντιστοιχεί στη συνηθισμένη χρήση του συστήματος (υπολογιστή, δικτύου κλπ) και μια ως ανώμαλη (anomalous) που αντιστοιχεί σε μη συνηθισμένη χρήση ή δραστηριότητα. Το IDS δημιουργεί κάποιο προφίλ χρήσης που μπορεί να αφορά το δίκτυο, ή τον ίδιο τον υπολογιστή, τα προγράμματα του υπολογιστή ή χρήστες. Κάθε απόκλιση από αυτό το προφίλ χρήσης κρίνεται από το IDS ως επίθεση. Εργαλεία υλοποίησης μπορεί να είναι

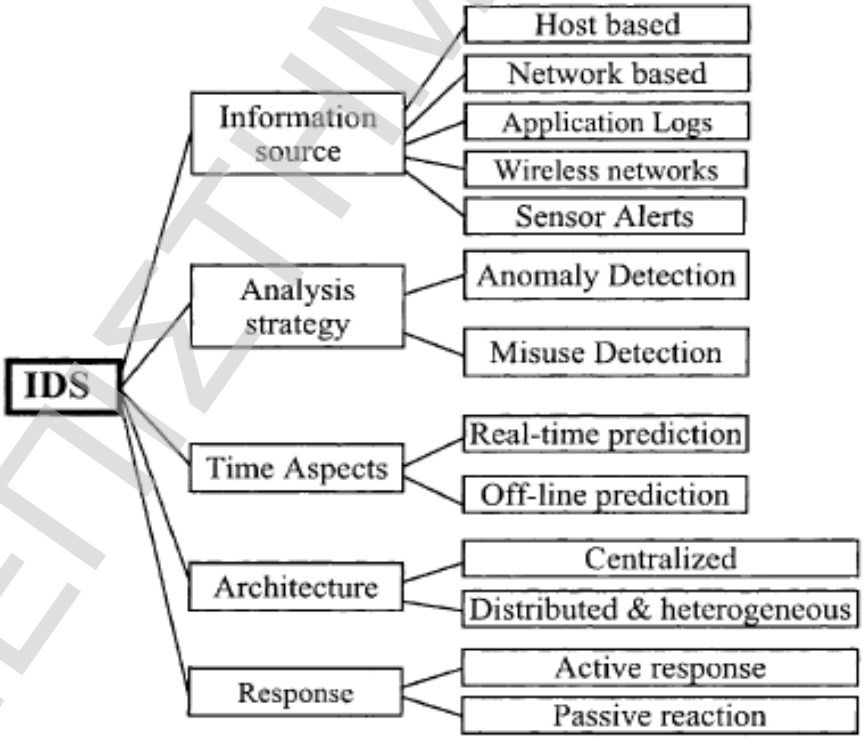
απλές στατιστικές μέθοδοι, «έξυπνα εργαλεία» όπως τα Νευρωνικά Δίκτυα, τεχνικές εξαγωγής δεδομένων (Data mining), γενετικός προγραμματισμός και η οπτικοποίηση. Πολύ συχνά αυτές οι τεχνικές ενημερώνουν αυτόματα το προφίλ χρήσης και κάποιες άλλες εκπαιδεύονται έτσι ώστε να «μάθουν» ποιο προφίλ είναι το κανονικό έτσι ώστε να αναγνωρίζουν την όποια παρέκκλιση. Η μηχανή ανάλυσης μπορεί να συνδυάζει περισσότερες από μια τεχνικές για μεγαλύτερη αποτελεσματικότητα. Η πολιτική ανίχνευσης είναι ένα σύνολο κανόνων που έχει στη διάθεσή του το Intrusion Detection System έτσι ώστε να ανιχνεύει επιθέσεις. Εδώ βρίσκονται οι υπογραφές των επιθέσεων, τα διάφορα κατώφλια που συμβουλεύεται το σύστημα για να πραγματοποιήσει την ανίχνευση, πληροφορίες για το ποια θεωρείται μη συνηθισμένη δραστηριότητα, όπως επίσης και οδηγίες για το ποια θα είναι η αντίδραση. Στο υποσύστημα αυτό το IDS κρατά «πληροφορίες κατάστασης» όπως είναι πληροφορίες για επίθεση που δεν έχει ολοκληρωθεί ή η τρέχουσα κατάσταση του συστήματος.

– *Υποσύστημα Αντίδρασης* ***(Response Unit).*** Η μηχανή ανάλυσης αποστέλλει στην μονάδα απάντησης πληροφορίες για γεγονότα που χαρακτηρίζονται ως επιθέσεις. Αφού η μονάδα βεβαιωθεί ότι πρόκειται για επίθεση, ή αν πρόκειται για κατανεμημένο IDS συλλέξει το σύνολο της πληροφορίας, απαντά βάσει της προκαθορισμένης πολιτικής αντίδρασης. Η πολιτική αντίδρασης μπορεί να περιλαμβάνει την ενημέρωση του administrator, το κλείσιμο συνδέσεων ή ports, ή την εντολή για τη συλλογή πιο λεπτομερούς πληροφορίας με στόχο την επιβεβαίωση της επίθεσης.

2.2.2 Κατηγορίες και λειτουργικά χαρακτηριστικά

Διάφορες κατηγοριοποιήσεις συστημάτων ανίχνευσης παρεισφρήσεων που έχουν κατά καιρούς προταθεί βασίζονται στα διάφορα χαρακτηριστικά των συστημάτων αυτών. Μια τέτοια πρόταση ταξινόμησης, η οποία έχει προταθεί από τους (Lazarevic et al, 2004), στηρίζεται στα ακόλουθα πέντε βασικά χαρακτηριστικά (Εικόνα 4):

- Την πηγή της πληροφορίας (Information Source)
- Τη στρατηγική ανάλυσης (Analysis strategy)
- Τη χρονική διάσταση (Time aspects)
- Την αρχιτεκτονική (Architecture)
- Την αντίδραση (Response)



Εικόνα 4. **Συγκεντρωτικός πίνακας ταξινόμησης** IDS (Lazarevic, 2004) **σύμφωνα με τα χαρακτηριστικά τους**

Το πρώτο κριτήριο, η πηγή της πληροφορίας, διαχωρίζει τα IDS ανάλογα με το από πού προέρχεται η πηγή της πληροφορίας. Αυτή μπορεί να προέρχεται από αρχεία καταγραφής του συστήματος, από τα δικτυακά πακέτα, από αρχεία καταγραφής εφαρμογών, από δικτυακή κίνηση σε ασύρματο δίκτυο ή από αισθητήρες κατανεμημένους σε κάποιο δίκτυο.

Η στρατηγική ανάλυσης διαχωρίζει τα IDS ανάλογα με την πολιτική ανίχνευσης που έχει επιλεγεί. Εδώ υπάρχουν δύο βασικές κατηγορίες πολιτικών ανίχνευσης: α) η ανίχνευση κακής χρήσης (misuse detection) σύμφωνα με την οποία το IDS παρακολουθεί και ταυτοποιεί γεγονότα στο σύστημα-στόχο τα οποία έχουν συγκεκριμένη μορφή και ταιριάζουν σε ήδη γνωστές επιθέσεις και β) η ανίχνευση ανωμαλίας (anomaly detection) σύμφωνα με την οποία το IDS παρακολουθεί για να εντοπίσει ασυνήθιστη συμπεριφορά ή κάποια συμπεριφορά η οποία διαφέρει από τη συνηθισμένη συμπεριφορά του συστήματος που παρακολουθείται.

Το κριτήριο της χρονικής διάστασης κατηγοριοποιεί τα IDS με βάση το χρόνο δραστηριοποίησης τους. Τα on-line IDS λειτουργούν και αντιδρούν σε επιθέσεις σε πραγματικό χρόνο. Αντίθετα τα off-line συστήματα πρώτα αποθηκεύουν τα δεδομένα, τα οποία στη συνέχεια αναλύουν και αξιολογούν.

Τα IDS επίσης διακρίνονται ανάλογα με τη αρχιτεκτονική τους διάταξη σε συγκεντρωτικά (Centralized), τα οποία συλλέγουν δεδομένα από ένα σημείο και τα κατανεμημένα, τα οποία συλλέγουν δεδομένα από διάφορες πηγές. Τέλος, ανάλογα

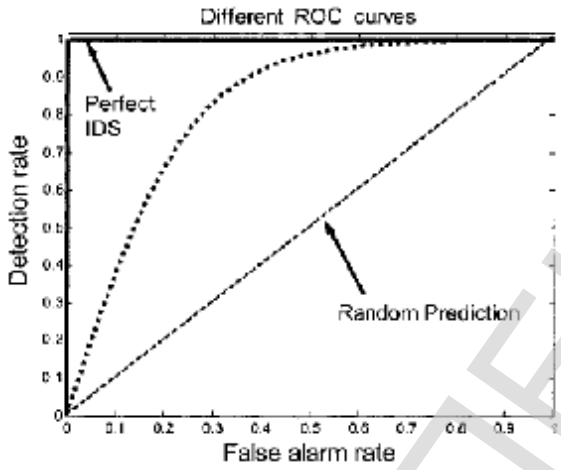
με τον τρόπο που αντιδρούν, τα IDS διαχωρίζονται σε δραστικά (active) τα οποία και αντιδρούν όταν εντοπιστεί επίθεση, κλείνοντας πχ κάποια ports, και τα παθητικά (passive) τα οποία απλά ενημερώνουν το διαχειριστή.

Εμπορικά IDS μπορεί να συνδυάζουν κάποια από τα προαναφερθέντα χαρακτηριστικά. Ιδιαίτερη απήχηση φαίνεται ότι έχουν τα Network based IDS, δεδομένου ότι συγκεντρώνουν μια σειρά από πλεονεκτήματα τα οποία και συνοψίζονται ως εξής (Sans Institute, 2002):

1. Χαμηλότερο συνολικό κόστος κτήσης. Σε κάθε δίκτυο αντιστοιχεί ένα IDS το οποίο είναι σε θέση να ελέγχει όλη την κίνηση, σε αντίθεση με τα host based στα οποία αντιστοιχεί ένα IDS ανά υπολογιστή.
2. Ευκολότερα στην εγκατάσταση. Είναι ευκολότερα δεδομένου ότι εγκαθίστανται σε υπάρχουσα δικτυακή υποδομή χωρίς να απαιτούνται αλλαγές ή άλλες προσθήκες. Επίσης είναι ανεξάρτητα από διαφοροποιήσεις λειτουργικών συστημάτων, εφαρμογών κλπ
3. Μπορούν να ανιχνεύσουν δικτυακού τύπου επιθέσεις όπως οι tcp syn attack, δεδομένου ότι μπορούν να διαβάζουν πληροφορίες από την επικεφαλίδα του πακέτου.
4. Ανίχνευση κοντά σε πραγματικούς χρόνους και δυνατότητα άμεσης απάντησης.
5. Ανίχνευση και αποτυχημένων επιθέσεων. Αυτή η πληροφορία είναι ιδιαίτερα χρήσιμη για περαιτέρω νομική διερεύνηση, αν χρειαστεί.

Αναφορικά με τα κριτήρια αξιολόγησης τα IDS κατηγοριοποιούνται με βάση την αποδοτικότητα, την ταχύτητα και τη σταθερότητά τους.

– *Απόδοση ανίχνευσης*. Για να είναι αποτελεσματικό ένα IDS αναφορικά με την απόδοση πρόβλεψης θα πρέπει να ικανοποιεί κάποια κριτήρια: θα πρέπει να μπορεί να αναγνωρίζει σωστά τις διάφορες επιθέσεις και δεν θα πρέπει να αναγνωρίζει κανονική δραστηριότητα ως επίθεση. Τυπικές μονάδες μέτρησης για την αξιολόγηση απόδοσης είναι το ποσοστό ανίχνευσης (Detection rate) και το ποσοστό λανθασμένων συναγερμών (False alarm rate-False positives). Το ποσοστό ανίχνευσης ορίζεται ως ο λόγος του αριθμού των σωστά ανιχνεύσιμων επιθέσεων προς τον συνολικό αριθμό των επιθέσεων. Ως ποσοστό λανθασμένων συναγερμών ορίζεται ο λόγος του αριθμού των κανονικών συνδέσεων οι οποίες λανθασμένα έχουν αξιολογηθεί ως επιθέσεις προς τον αριθμό των κανονικών συνδέσεων. Αντικειμενικά είναι δύσκολο να αξιολογήσει κανείς αυτές τις μετρήσεις δεδομένου ότι δεν υπάρχει γνώση του συνόλου των επιθέσεων. Η αξιολόγηση IDS γίνεται χρησιμοποιώντας το μοντέλο ανάλυσης ROC (Receiver Operating Characteristics) (Provost, Fawcett 2001). Σύμφωνα με το μοντέλο αυτό υπάρχει ένα αντιστάθμισμα ανάμεσα στο ποσοστό ανίχνευσης και στο ποσοστό λανθασμένων συναγερμών όπως διαφαίνεται και στην Εικόνα 5.



Εικόνα 5. ROC **διάγραμμα για αξιολόγηση** IDS (Lazarevic, 2004)

Όπως μπορεί να παρατηρήσει κανείς, η ιδανική περίπτωση είναι η καμπύλη αξιολόγησης να περνά όσο το δυνατό πιο κοντά στην πάνω αριστερή γωνία του διαγράμματος, το οποίο αντιστοιχεί σε 0% false alarms και 100% detection rate. Η γραμμή random prediction αναδεικνύει το ότι οι κλάσεις είναι δύο: ένα τυχαίο δείγμα κίνησης είτε θα είναι κανονική κίνηση είτε επίθεση.

– *Χρονική απόδοση.* Η χρονική απόδοση ενός IDS είναι ο χρόνος που χρειάζεται για να αναγνωρίσει μια επίθεση. Στον χρόνο αυτό περιλαμβάνεται ο χρόνος επεξεργασίας και ο χρόνος διάδοσης. Ο χρόνος επεξεργασίας είναι ο χρόνος που το IDS επεξεργάζεται τα ίχνη συμβάντων (Audit Events). Αν ο χρόνος αυτός είναι μεγάλος τότε πιθανότατα το IDS να είναι αναποτελεσματικό, δεδομένου ότι η επίθεση θα πρέπει να αναγνωρίζεται σε χρόνους όσο το δυνατό πιο κοντά στον πραγματικό χρόνο που εκτυλίσσεται κάποια επίθεση. Ο χρόνος διάδοσης είναι ο χρόνος που χρειάζεται για να φτάσει η πληροφορία από την επεξεργασία στο

υποσύστημα ανάλυσης. Σε κάθε περίπτωση οι χρόνοι αυτοί θα πρέπει να είναι εξαιρετικά μικροί.

- *Ανοχή σε σφάλματα (fault tolerance).* Το ίδιο το IDS θα πρέπει να ανεξάρτητο, σταθερό και να μπορεί να ανακάμπτει γρήγορα για να μπορεί να συνεχίζει να προσφέρει τις υπηρεσίες ασφάλειας. Τα ίδια τα IDS είναι συχνά στόχος επιθέσεων αφού παρακάμπτοντάς τα γίνεται ευκολότερο το έργο της διείσδυσης για τον επιτιθέμενο. Εξίσου συχνές είναι οι κατανεμημένες DoS επιθέσεις σε IDS, ειδικά σε περιπτώσεις που έχουμε software implemented IDS, μιας και τα λειτουργικά συστήματα που τα φιλοξενούν έχουν γνωστές αδυναμίες. Τέλος τα IDS θα πρέπει να διαθέτουν ιδιαίτερους τρόπους αντιμετώπισης σεναρίων όπου οι επιτιθέμενοι προκαλούν τη δημιουργία πολλών ψευδών ή παραπλανητικών alarms.

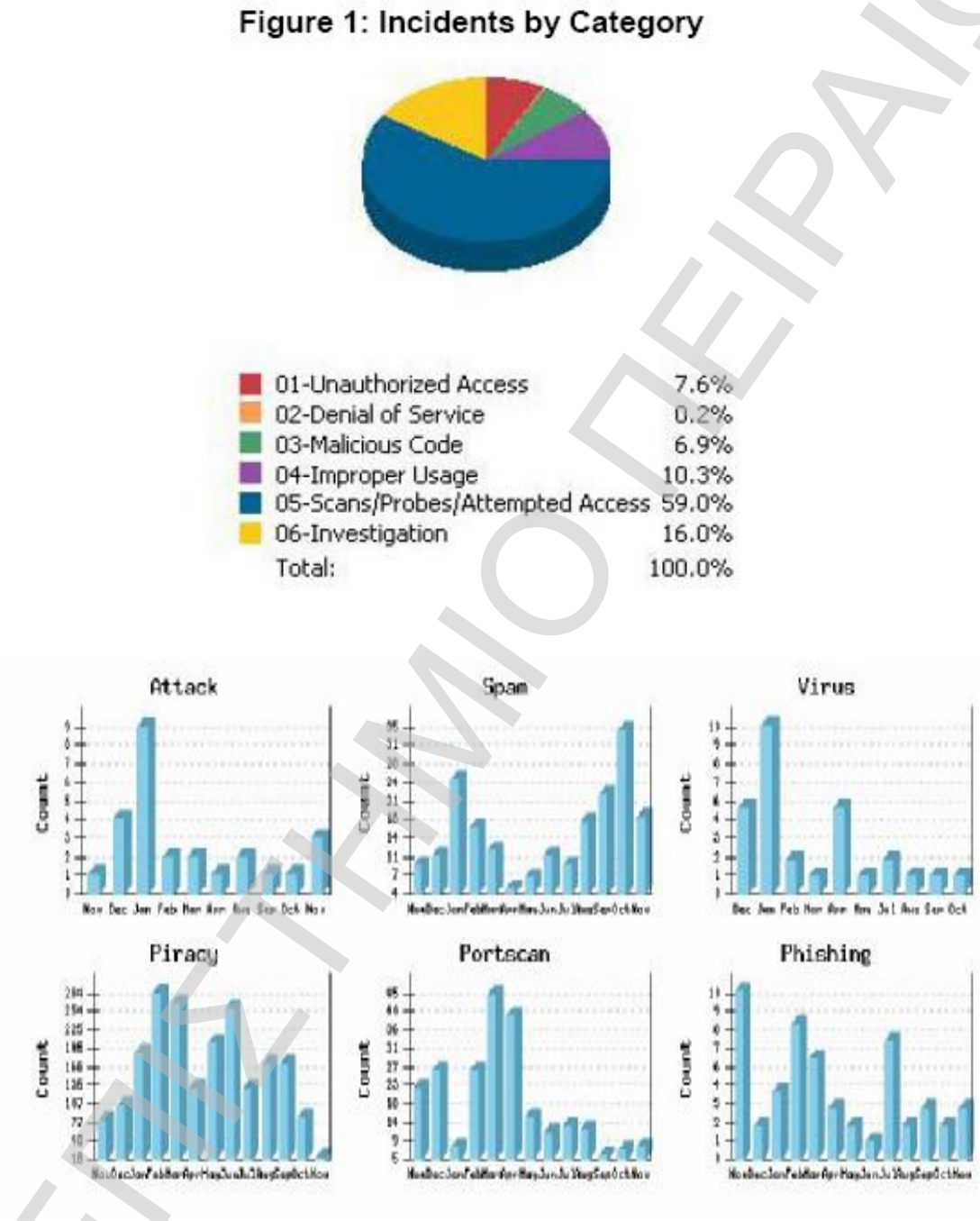
Παρά τη μεγάλη ανάπτυξη των IDS υπάρχουν αρκετά σημεία στα οποία θα πρέπει να βελτιωθούν έτσι ώστε να αποτελούν αξιόπιστα εργαλεία για την ασφάλεια των πληροφοριακών συστημάτων. Τα σημεία αυτά αφορούν την αποτελεσματικότητά τους, και την απόδοσή τους. Συγκεκριμένα, αναφορικά με την αποτελεσματικότητα θα μπορούσε να ειπωθεί ότι ο ιδανικός στόχος για ένα IDS θα ήταν 100% επιτυχία στην ανίχνευση επιθέσεων με σχεδόν 0% false alarms. Είμαστε ακόμα πολύ μακριά από τον στόχο αυτό. Δύο από τα πιο διαδεδομένα IDS, το ελεύθερα διαθέσιμο Snort και το εμπορικό RealSecure, χρησιμοποιούν υπογραφές για την ανίχνευση επιθέσεων. Αυτό σημαίνει ανάγκη για συνεχή ενημέρωση και άρα εξάρτηση από τα

προϊόντα αυτά. Αυτό που χρειάζονται τα πληροφοριακά συστήματα είναι η μόνιμη δυνατότητα ανίχνευσης νέων επιθέσεων με όσο το δυνατό λιγότερα false positives.

Σε σχέση με την απόδοση θα μπορούσε να ειπωθεί ότι η ανίχνευση μόνο ενός πλήθους επιθέσεων δεν αρκετή. Τα σύγχρονα IDS θα πρέπει να είναι σε θέση να ανταποκριθούν στις αυξημένες απαιτήσεις των σύγχρονων πληροφοριακών συστημάτων. Το Gigabit Ethernet, σε συνδυασμό με τα σύγχρονα λειτουργικά συστήματα που δημιουργούν χιλιάδες αρχεία καταγραφής, δημιουργούν μεγάλες ποσότητες δεδομένων για επεξεργασία από τα IDS. Επιπλέον υπάρχει πάντα η ανάγκη για ανίχνευση παρεισφρήσεων σε πραγματικό χρόνο.

2.3 Κατηγορίες επιθέσεων

Η ερευνητική κοινότητα έχει ασχοληθεί αρκετά με την ταξινόμηση των διαφόρων ειδών επιθέσεων. Θα πρέπει να αναφερθεί ο σαφής διαχωρισμός ανάμεσα σε επιθέσεις (attacks) και παρεισφρήσεις (intrusions): οι επιθέσεις δεν καταλήγουν πάντοτε σε παρεισφρήσεις αλλά ενίοτε αντιμετωπίζονται από την υπάρχουσα υποδομή ασφαλείας ενός δικτύου. Επίσης μπορεί να μην είναι η παρείσφρηση ο τελικός σκοπός κάποιας επίθεσης αλλά πχ η άρνηση κάποιας δικτυακής υπηρεσίας (Denial of Service). Γενικά μπορούμε να πούμε ότι μια επίθεση είναι μια προσπάθεια παρείσφρησης ενώ η παρείσφρηση είναι μια επιτυχημένη -εν μέρει ίσως- επίθεση (Powell, Stroud, 2003).



Εικόνα 6. Καταγραφή περιστατικών το τελευταίο δωδεκάμηνο για την Ελληνική επικράτεια.

Στοιχεία από το CERT του Δικτύου έρευνας και τεχνολογίας (http://cert.grnet.gr/, 2005)

Σχετικά με την κατηγοριοποίηση, η προσπάθεια επικεντρώθηκε στο διαχωρισμό βάσει των τυπικών χαρακτηριστικών των διαφόρων επιθέσεων, όπως οι τεχνικές λεπτομέρειες της επίθεσης (Bishop, 1990), η λογική των επιθέσεων (Howard, 2000), ο σκοπός (Krsul, 2003), η επαναληπτικότητα κλπ. Εκτενέστερη αναφορά υπάρχει στην έρευνα του Lough (Lough, 2001). Στη συνέχεια, η προσπάθεια κατηγοριοποίησης επικεντρώθηκε στην καταγραφή και ταξινόμηση των πληροφοριακών συστημάτων με βάση τις εγγενείς αδυναμίες ασφάλειάς τους, όπως ο κακός σχεδιασμός (Attanasio, 1976) και οι λειτουργικές αδυναμίες (Parker, 1975) για να καταλήξουν τελικά σε δύο κατηγορίες: την κατηγορία κακής χρήσης (misuse) υπολογιστών και την κατηγορία μη εξουσιοδοτημένης πρόσβασης (unauthorized access).

Στη συνέχεια παρουσιάζεται μια πιο σύγχρονη κατηγοριοποίηση (Abraham, 2001), η οποία είναι ένας συνδυασμός των προηγούμενων, αν και συμπεριλαμβάνει και κάποια νέα είδη απειλών. Τα χαρακτηριστικά στα οποία στηρίζεται αυτή η κατηγοριοποίηση είναι:

- ο τύπος της επίθεσης
- ο αριθμός των δικτυακών συνδέσεων που συμμετέχουν στην επίθεση
- η πηγή της επίθεσης
- το περιβάλλον που εξελίσσεται η επίθεση
- το επίπεδο αυτοματισμού της επίθεσης

Στη συνέχεια αναπτύσσουμε ένα-ένα αυτά τα χαρακτηριστικά των επιθέσεων.

- *Ο τύπος επίθεσης*. Σύμφωνα με τη βιβλιογραφία είναι το πιο συνηθισμένο κριτήριο ταξινόμησης επιθέσεων (Howard 1997, Kendall, 1998). Σύμφωνα με αυτό το κριτήριο υπάρχουν οι εξής κλάσεις επιθέσεων:
  - *Επιθέσεις άρνησης παροχής υπηρεσιών (Denial of Service - DoS-attacks).* Αυτού του είδους οι επιθέσεις έχουν ως στόχο το κλείσιμο ενός υπολογιστή ή μιας διεργασίας ή ακόμα την άρνηση χρήσης των πόρων ενός υπολογιστικού συστήματος σε εξουσιοδοτημένους χρήστες (Marchette, 2001). Υπάρχουν δύο τύποι DoS επιθέσεων i) επιθέσεις σε λειτουργικά συστήματα τα οποία έχουν συγκεκριμένες αδυναμίες τις οποίες ακριβώς εκμεταλλεύονται οι DoS επιθέσεις, ενώ τα προβλήματα συνήθως επιλύονται με κάποιο «μπάλωμα» (patch). ii) δικτυακές επιθέσεις οι οποίες εκμεταλλεύονται αδυναμίες υποδομών και δικτυακών πρωτοκόλλων. Παράδειγμα DoS επίθεσης σε λειτουργικό σύστημα είναι η επίθεση "teardrop" με την αποστολή fragmented πακέτων τα οποία τα σύστημα δεν μπορεί να χειριστεί, με αποτέλεσμα να καταρρεύσει. Τυπικό παράδειγμα δικτυακής DoS επίθεσης είναι η "SYN flood" με την οποία ο επιτιθέμενος εκμεταλλεύεται την three way handshake διαδικασία σύνδεσης στέλνοντας πολλαπλές αιτήσεις χωρίς την ack διαδικασία, με αποτέλεσμα το σύστημα να κρατά ανοικτές συνδέσεις περιμένοντας να ολοκληρωθούν και τελικά να καταρρέει. Μια πιο εξελιγμένη μορφή DoS επίθεσης είναι η DDoS (Distributed Denial of Service) σύμφωνα με την οποία γίνεται συνδυασμένη επίθεση από πολλούς υπολογιστές ταυτόχρονα

σε κάποιο υπολογιστή-στόχο. Η αντιμετώπιση των DoS επιθέσεων αποτελεί ένα ανοιχτό ερευνητικό πεδίο (Mircovich et al 2001,2004) αφού θα πρέπει να αντιμετωπιστούν δύο βασικά προβλήματα : η ανάπτυξη μηχανισμών για την έγκαιρη αντίδραση σε τέτοιου είδους επιθέσεις (Cheng 2002, Gil 2001) και η ανάπτυξη μηχανισμών αντίδρασης σε DoS επιθέσεις, έτσι ώστε να μειωθεί ο αντίκτυπός τους (Burch,2000, Snoeren 2001).

– *Επιθέσεις διερεύνησης (Probing, surveillance, scanning).* Αυτού του είδους οι επιθέσεις έχουν ως στόχο τη «χαρτογράφηση» κάποιου δικτύου-θύματος. Συνήθως αποτελούν το πρώτο στάδιο για μια άλλου είδους επίθεση, μιας και πιθανότατα αποκαλύπτουν στον επιτιθέμενο αδυναμίες στην ασφάλεια του συστήματος-στόχου. Παραδείγματα αυτού του τύπου επιθέσεων είναι η IPsweep επίθεση με την οποία το δίκτυο-στόχος σαρώνεται από τον επιτιθέμενο σε συγκεκριμένες πόρτες, η επίθεση portsweep με την οποία επιλέγεται συγκεκριμένος υπολογιστής και σαρώνεται για να διαπιστωθεί ποια από τα ports -και άρα δικτυακές υπηρεσίες- είναι διαθέσιμες από τον υπολογιστή, η επίθεση με το nmap εργαλείο με το οποίο γίνεται mapping πόρων ενός δικτύου. Οι επιθέσεις αυτού του είδους είναι σχετικά εύκολο να εντοπιστούν από εργαλεία παρακολούθησης δικτύων, εξ αιτίας του γεγονότος ότι παρατηρείται μεγάλη εναλλαγή αναζήτησης υπηρεσιών σε πολύ μικρό χρονικό διάστημα. Εκεί όμως που υπάρχει δυσκολία στον εντοπισμό είναι σε

κάποιες παραλλαγές των επιθέσεων αυτών. Τα εργαλεία που χρησιμοποιούνται δεν σαρώνουν γρήγορα τους υπολογιστές-στόχους αλλά καθυστερούν σκόπιμα τη σάρωση, έτσι ώστε να φαίνεται ως κανονική η όλη λειτουργία και να μην ενεργοποιείται κάποια υπηρεσία στα συστήματα παρακολούθησης των δικτύων. Αυτού του είδους οι επιθέσεις είναι γνωστές με το όνομα Stealth. Για την αντιμετώπισή τους χρησιμοποιούνται τεχνικές οι οποίες βασίζονται σε συλλογή και επεξεργασία στατιστικών στοιχείων (Jung et all 2004, Mazu Profiler 2003, Robertson et al 2003).

- *Επιθέσεις τύπου έκθεσης (Compromise).* Οι επιθέσεις αυτού του τύπου χρησιμοποιούν γνωστές αδυναμίες συστημάτων, όπως είναι τα buffer overflows (CERT Advisory, 2003), με σκοπό να διασπάσουν την ασφάλειά τους και να αποκτήσουν δικαιώματα πρόσβασης. Με κριτήριο την προέλευση των επιθέσεων, δηλαδή αν οι επιθέσεις προέρχονται από το εξωτερικό ή το εξωτερικό ενός δικτύου, οι επιθέσεις χωρίζονται σε δύο υποκατηγορίες:
  - R2L (Remote to Local) επιθέσεις, με τις οποίες ο επιτιθέμενος έχει τη δυνατότητα επικοινωνίας με κάποιο υπολογιστή και καταφέρνει να αποκτήσει δικαιώματα χρήστη ή διαχειριστή στον υπολογιστή αυτό. Στις περισσότερες περιπτώσεις η επίθεση εκδηλώνεται πάνω από το Internet. Τυπικά παραδείγματα τέτοιων επιθέσεων είναι το password

guessing (guest και dictionary attacks) και η προσπάθεια πρόσβασης με εκμετάλλευση κενών ασφαλείας (πχ phf attack).

- U2R (User to Root) επιθέσεις, κατά τις οποίες ο επιτιθέμενος έχει ήδη ένα ενεργό λογαριασμό σε κάποιο σύστημα και ο στόχος είναι να αποκτήσει δικαιώματα διαχειριστή. Αντίθετα με την Remote to Local επίθεση, η οποία εκδηλώνεται από το εξωτερικό του δικτύου, η User to Root εκδηλώνεται από το εσωτερικό του. Οι πιο γνωστές επιθέσεις αυτού του τύπου είναι τα buffer overflows όπου ο επιτιθέμενος εκμεταλλεύεται λάθη του κώδικα στο λειτουργικό σύστημα και τροφοδοτεί κάποιους καταχωρητές με δεδομένα περισσότερα από όσα μπορούν να χωρέσουν, με αποτέλεσμα να καταστρέφονται "γειτονικά δεδομένα" που συχνά έχουν ως αποτέλεσμα την απόδοση δικαιωμάτων διαχειριστή σε κάποιον απλό χρήστη. Γι' αυτού του είδους τις επιθέσεις έχουν προταθεί κάποιες λύσεις (C. Cowan et al, H. Feng et al 2003), ενώ θα πρέπει να σημειωθεί ότι buffer overflows μπορεί να πραγματοποιηθούν και με R2L επιθέσεις με πιο πρόσφατο παράδειγμα τις επιθέσεις σε υπολογιστές που χρησιμοποιούσαν MS Outlook ως email clients (CERT Advisory 2000).

– *Ιοί/Σκουλήκια /Δούρειοι Ίπποι (Viruses, Worms, Trojan Horses)* είναι προγράμματα τα οποία αναπαράγονται σε υπολογιστές-στόχους και διαδίδονται μέσω δικτύων.

- οι Ιοί (Viruses) είναι προγράμματα που διαδίδονται με την ενσωμάτωση σε άλλα προγράμματα που πλέον θεωρούνται μολυσμένα. Μπορεί να είναι από λίγο έως τελείως καταστροφικοί για υπολογιστικά συστήματα, όπως για παράδειγμα ο ιός Michaelangelo, ο οποίος ενεργοποιήθηκε σε συγκεκριμένη ημερομηνία. Υπάρχουν πολλές κατηγορίες ιών οι οποίοι κατηγοριοποιούνται ανάλογα με τον αλγόριθμο ή τις καταστροφικές δυνατότητες που έχουν και ανάλογα με το τι και πώς μολύνουν (cknow.com 2003, Internet guide 2002).
- τα Σκουλήκια (worms) είναι προγράμματα που αυτοαναπαράγονται και διαδίδονται ταχύτατα στα δίκτυα. Υπάρχουν αρκετές κατηγορίες (Kienzle and Elder 2003) εκ των οποίων οι πιο σημαντικές είναι:
  - τα παραδοσιακά σκουλήκια (πχ Slammer (CERT Advisory 2003)) που διαδίδονται πολύ γρήγορα στο δίκτυο χωρίς καμία παρέμβαση από το χρήστη
  - τα e-mail worms (πχ ο Melissa (CERT Advisory 1999)) τα οποία μεταδίδονται με email ή με ICQ
  - τα windows sharing worms (πχ explorerzip) τα οποία εκμεταλλεύονται την υπηρεσία peer to peer των Windows
  - τα υβριδικά worms, όπως είναι ο Nimda (CERT Advisory 2001), τα οποία εκμεταλλεύονται διαφορετικές αδυναμίες των συστημάτων για να διαδοθούν. Πρόσφατα έχει εμφανιστεί ένας

νέος τύπος worm που χρησιμοποιείται για να εξαπολύσει DoS επιθέσεις (Houle 2001). Παραδείγματα τέτοιων worms είναι ο erkms και ο Code Red οι οποίοι επιτίθενται σε συγκεκριμένη διεύθυνση, ενώ πιο επικίνδυνο worm θεωρείται ο SoBig.F ο οποίος μπορεί να επιτεθεί σε ένα εύρος διευθύνσεων. Υπάρχει η πρόβλεψη ότι σχεδόν όλες οι DoS επιθέσεις στο μέλλον θα εξελίσσονται δια μέσου worm payload.

- Δούρειοι Ίπποι (Trojan Horses). Είναι κακόβουλο λογισμικό που ως στόχο έχει τη διάσπαση της ασφάλειας ενός συστήματος (Lo 2004). Συνήθως εγκαθίσταται σε κάποιο σύστημα εν αγνοία ενός χρήστη, είτε ως μέρος ενός πακέτου λογισμικού είτε δια μέσου ανταλλαγής αρχείων peer to peer. Παραδείγματα Trojan horses είναι τα Silk Rope και Saran Wrap.

- *ο αριθμός των δικτυακών συνδέσεων που συμμετέχουν στην επίθεση*. Οι επιθέσεις μπορούν να ταξινομηθούν ανάλογα με τον αριθμό των συνδέσεων που συμμετέχουν στην επίθεση. Υπάρχουν δύο κατηγορίες:
  - επιθέσεις όπου συμμετέχουν πολλές συνδέσεις πχ DoS επιθέσεις
  - επιθέσεις με μια ή πολύ περιορισμένο αριθμό δικτυακών συνδέσεων πχ Buffer Overflow.
- *η πηγή της επίθεσης*. Οι επιθέσεις μπορούν να ξεκινήσουν από μια απλή θέση (single source attacks) ή από πολλά διαφορετικά σημεία

(distributed/coordinated attacks). Επιπλέον οι distributed/coordinated επιθέσεις μπορούν να στοχεύουν είτε σε ένα είτε σε πολλούς υπολογιστές.

- *το περιβάλλον που εξελίσσεται η επίθεση*. Αυτές οι επιθέσεις κατηγοριοποιούνται ανάλογα με το περιβάλλον που εκτυλίσσονται:
    - επιθέσεις σε μεμονωμένους υπολογιστές. Οι επιθέσεις αυτές γίνονται σε συγκεκριμένους υπολογιστές και συνήθως καταγράφονται στο ίδιο το σύστημα (log files). Για την ανίχνευση τέτοιων επιθέσεων χρησιμοποιούνται είτε ανάλυση των log αρχείων είτε ανάλυση των εντολών σε επίπεδο συστήματος.
    - Επιθέσεις σε δίκτυα οι οποίες συνήθως ξεκινούν από το εξωτερικό τους. Η ανίχνευση τέτοιων επιθέσεων προϋποθέτει την ανάλυση της δικτυακής κίνησης
    - Επιθέσεις σε ομότιμα (peer to peer) δίκτυα. Αποτελούν ιδιαίτερη κατηγορία δεδομένου ότι στα δίκτυα αυτά δεν ισχύει το κλασικό client server μοντέλο και επιπλέον δεν υπάρχουν σταθερές IP διευθύνσεις. Αυτό κάνει ιδιαίτερα δύσκολη την ανίχνευση επιθέσεων σε τέτοια περιβάλλοντα.
    - Επιθέσεις σε ασύρματα δίκτυα εμφανίζονται ανάμεσα σε υπολογιστές που είναι συνδεδεμένοι ασύρματα. Ο εντοπισμός των επιθέσεων γίνεται δια μέσου των access points μιας και αυτοί είναι οι κόμβοι από

όπου διέρχεται η δικτυακή κίνηση. Οι δικτυακές απειλές σε ασύρματα δίκτυα χωρίζονται σε:

a) *Κρυφάκουσμα (eaves dropping).* Ο επιτιθέμενος μονάχα παρακολουθεί μια σύνδεση

b) *Παρείσφρηση.* Ο επιτιθέμενος προσπελαύνει ή τροποποιεί δεδομένα

c) *Πειρατεία (hijacking).* Ο επιτιθέμενος καταλαμβάνει ένα κανάλι επικοινωνίας και προσποιούμενος ότι αποτελεί κάποιο σημείο πρόσβασης συλλέγει δεδομένα.

d) *Άρνηση υπηρεσίας* όταν ο επιτιθέμενος παρεμποδίζει ή διακόπτει την επικοινωνία ανάμεσα σε συσκευές που επικοινωνούν ασύρματα.

- *το επίπεδο αυτοματισμού της επίθεσης.* Αυτός ο διαχωρισμός προκύπτει από το επίπεδο αυτοματισμού που έχει μια επίθεση. Οι επιθέσεις διακρίνονται σε:
    - *Αυτοματοποιημένες επιθέσεις.* Πραγματοποιούνται συνήθως με τη βοήθεια πολύπλοκων και ιδιαίτερα ισχυρών εργαλείων. Αυτά τα εργαλεία καθίστανται αποτελεσματικά ακόμα και στα χέρια αρχάριων χρηστών και αποτελούν τον πιο συνηθισμένο τρόπο επιθέσεων σήμερα.
    - *Ημιαυτοματοποιημένες επιθέσεις.* Πρόκειται συνήθως για κώδικα ο οποίος ανιχνεύει σε πρώτη φάση αδυναμίες του στόχου και στη

συνέχεια ενημερώνει τον επιτιθέμενο για να αποφασιστεί ο τρόπος της επίθεσης.

- *Μη αυτοματοποιημένες* ***(Manual)*** *επιθέσεις*. Αυτό το είδος των επιθέσεων προϋποθέτει πολύ καλή γνώση δικτύων από μέρους του επιτιθέμενου. Ως εκ τούτου είναι και οι πιο επικίνδυνες, αφού στο μεγαλύτερο μέρος τους υπάρχει ανθρώπινη συμμετοχή.

2.4 Ανίχνευση Παρεισφρήσεων με χρήση τεχνολογιών Τεχνητής Νοημοσύνης

Οι τεχνολογίες Τεχνητής Νοημοσύνης έχουν χρησιμοποιηθεί ευρύτατα στο χώρο της Ανίχνευσης Παρεισφρήσεων. Οι ερευνητές έχουν επικεντρώσει την έρευνα και προς τις δύο κατηγορίες ανάλυσης (Misuse και anomaly) ενώ σε πολλές περιπτώσεις γίνεται συνδυασμός περισσοτέρων τεχνολογιών.

Το IDES (Lunt et al, 1992) είναι ένα πραγματικού χρόνου έμπειρο σύστημα ανίχνευσης παρεισφρήσεων. Η λειτουργία του στηρίζεται στη δημιουργία προφίλ χρήσης για κάθε χρήστη. Έτσι όταν παρατηρείται απόκλιση από τη συνηθισμένη «συμπεριφορά» κάποιου χρήστη, το σύστημα μπορεί να εντοπίσει παρείσφρηση. Για την αναγνώριση γνωστών τύπων επιθέσεων αξιοποιεί διάφορους κανόνες με τη μορφή έμπειρου συστήματος.

Το MIDAS (Sebring et al, 1988) είναι ένα έμπειρο σύστημα με δυνατότητα επεξεργασίας ενός μεγάλου αριθμού συμπερασματολογιών και βασίζεται στις αρχές

του μοντέλου Denning, δηλαδή στην αναπαράσταση της συμπεριφοράς των χρηστών σε ένα σύστημα μέσω στατιστικών μετρήσεων και κανόνων.

Το SECURENET (Spyrakis. et al, 1994) έχει σχεδιαστεί για την προστασία δικτυακού περιβάλλοντος, και το βασικό χαρακτηριστικό του είναι ότι συνδυάζει διάφορες μεθοδολογίες, όπως έμπειρα συστήματα και νευρωνικά δίκτυα. Το SECURENET θέτει ως βασικό σκοπό την ολοκληρωμένη προστασία μέσω τριών διαδικασιών: την ανίχνευση μιας επίθεσης (detection of attack), την κατηγοριοποίηση της επίθεσης σε πραγματικό χρόνο (classification of attack) και την επιλογή και εφαρμογή των κατάλληλων μέτρων κατά της επίθεσης που εκδηλώνεται (countermeasurs). Έτσι στο σύστημα περιλαμβάνονται όλα τα τμήματα, από την παρακολούθηση του δικτύου (monitoring), μέχρι την τελική συμβουλή στον διαχειριστή ασφάλειας του δικτύου για την αντιμετώπιση πιθανού κινδύνου.

Τα νευρωνικά δίκτυα, τα fuzzy συστήματα και οι εξελικτικοί αλγόριθμοι έχουν χρησιμοποιηθεί αρκετά για υλοποιήσεις Δικτυακής Ανίχνευσης Παρεισφρήσεων (Network Intrusion Detection-NID). Τα νευρωνικά δίκτυα ως γνωστό αποτελούν ισχυρούς ταξινομητές και επιλογείς χαρακτηριστικών (feature selectors/dimensionality reducers). Νευρωνικό δίκτυο με back propagation αλγόριθμο εκπαίδευσης έχει χρησιμοποιηθεί στο MAID σύστημα (Li and Manikopoulos 2003), για την αντιμετώπιση DoS επιθέσεων, ενώ η είσοδος για το νευρωνικό δίκτυο ήταν στατιστικά επεξεργασμένα δεδομένα από MIB πάνω από SNMP πρωτόκολλο. Οι Ng et al (Ng et al, 2003) έχουν χρησιμοποιήσει στοχαστικό Radial-Basis Function νευρωνικό δίκτυο, για εξαγωγή και αξιολόγηση χαρακτηριστικών DoS επίθεσης με

πρωτογενή δεδομένα από τη βάση δεδομένων DARPA. Μια από τις πλέον συνηθισμένες υλοποιήσεις NIDS είναι με τη χρήση Support Vector Machines μιας και από ότι φαίνεται είναι τουλάχιστο το ίδιο αποτελεσματικά με τα νευρωνικά δίκτυα. Οι Liu et al (Liu et al. 2003) περιγράφουν το σχεδιασμό και την υλοποίηση ενός συστήματος για ανίχνευση ανωμαλιών σε πραγματικό χρόνο χρησιμοποιώντας ART (Adaptive Resonance Theory) νευρωνικό δίκτυο. Ο Valdes (Valdes, 2003) χρησιμοποιεί νευρωνικά δίκτυα με εκπαίδευση χωρίς επίβλεψη (Unsupervised learning) με σκοπό να διαχωρίσει δεδομένα σε συστοιχίες (clusters). Τα νευρωνικά δίκτυα αυτού του τύπου έχουν το χαρακτηριστικό ότι δεν χρειάζονται ετικέτες (labels) για κάθε είσοδο ώστε να κάνουν το διαχωρισμό (επιβλεπόμενη μάθηση), αλλά έχουν την ικανότητα της αυτο-οργάνωσης. Έτσι δεν υπάρχει ανάγκη του διαχωρισμού των δεδομένων κατά την εκπαίδευση σε κανονικά και δεδομένα επίθεσης. Οι Sarasamma et al. (Sarasamma et al, 2005) προτείνουν ένα ιεραρχικό μοντέλο νευρωνικού δικτύου Kohonen. Τεχνολογίες Data Mining και ειδικότερα ο κανόνας συσχέτισης έχουν χρησιμοποιηθεί (Barbara et al, 2001) με σκοπό την εξεύρεση «κανόνων» στα δεδομένα που συλλέγονται. Επίσης έχουν χρησιμοποιηθεί τεχνικές Fuzzy Logic (Dickerson et al, 2000) καθώς και συνδυασμός τους με γενετικούς αλγορίθμους (Gomez et al, 2001).

Είναι γεγονός ότι η διάσταση των δεδομένων εισόδου είναι αρκετά μεγάλη, μιας και συμπεριλαμβάνει όλη σχεδόν την πληροφορία ενός πακέτου από κάποιο δικτυακό πρωτόκολλο με αποτέλεσμα ο χρόνος επεξεργασίας των δεδομένων να είναι μεγάλος. Έτσι σε πολλές περιπτώσεις οι τεχνολογίες τεχνητής νοημοσύνης όπως είναι τα

νευρωνικά δίκτυα σε συνδυασμό με στατιστικές μεθόδους στοχεύουν ακριβώς στη μείωση της διάστασης αυτής (Zanero et al 2004, Kayacic 2003).

2.5 Νευρωνικά Δίκτυα

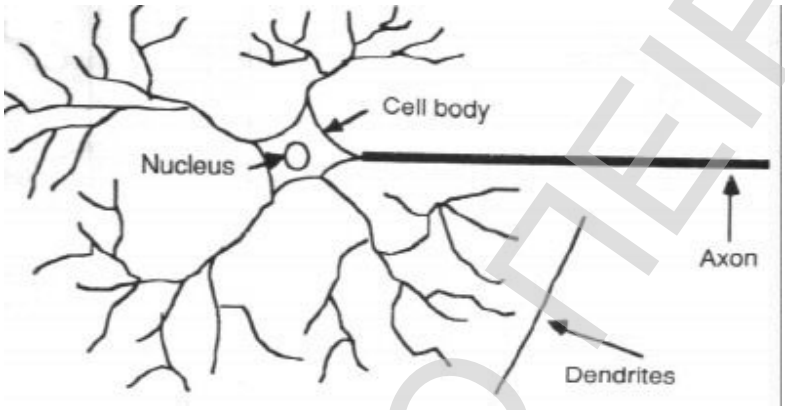
Τα Τεχνητά Νευρωνικά Δίκτυα είναι μια ιδέα στο χώρο των υπολογιστών η οποία βασίστηκε στη μοντελοποίηση της δομής και της λειτουργίας του ανθρώπινου εγκεφάλου. Η ιδέα προϋπήρχε από τη δεκαετία του ΄40 αλλά οι σχετικά πρόσφατες εξελίξεις της δεκαετίας του ΄80 στο χώρο της πληροφορικής έδωσαν νέα ώθηση στην τεχνολογία αυτή.

Η διαφορά που έχουν τα Νευρωνικά Δίκτυα σε σχέση με τα κλασικά υπολογιστικά συστήματα προέρχεται από το γεγονός ότι δεν χρησιμοποιούν την προσέγγιση της διαδοχικής εκτέλεσης εντολών για την επίλυση των προβλημάτων αλλά, μέσα από μια διαδικασία εκπαίδευσης, «μαθαίνουν» και προσεγγίζουν τη λύση ακόμα και αν τα δεδομένα δεν είναι πλήρη ή περιέχουν «θόρυβο».

Η εκπαίδευση περιλαμβάνει την τροφοδοσία του Νευρωνικού δικτύου με δεδομένα τα οποία έχουν συγκεκριμένη δομή. Αφού τα δεδομένα κριθούν επαρκή, το δίκτυο θεωρείται εκπαιδευμένο και μπορεί να επιλύσει προβλήματα παρόμοια με τα αρχικά.

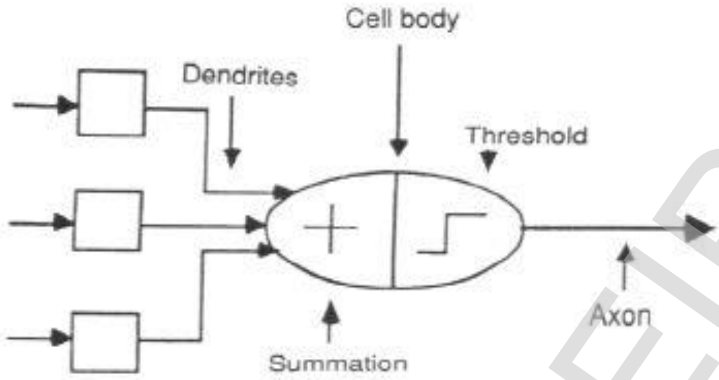
Τα μοντέλα των Νευρωνικών Δικτύων είναι αλγόριθμοι γνωστικών διαδικασιών όπως η μάθηση και η βελτιστοποίηση, τα οποία βασίζονται σε έννοιες που αντλούνται από τη μελέτη της φύσης του εγκεφάλου.

Τεχνητό Νευρωνικό Δίκτυο είναι ένα μοντέλο επεξεργασίας πληροφοριών εμπνευσμένο από τον τρόπο που τα βιολογικά νευρικά συστήματα επεξεργάζονται πληροφορίες.



Εικόνα 7. **Το μοντέλο ενός φυσικού νευρώνα** (SURPRISE 96 Journal, NEURAL NETWORKS by Christos Stergiou and Dimitrios Siganos, 1996)

Το βασικό στοιχείο αυτών των συστημάτων είναι η δομή επεξεργασίας πληροφοριών. Η δομή αυτή αποτελείται από μεγάλο αριθμό στοιχείων επεξεργασίας, τους νευρώνες (neurons), οι οποίοι συνεργάζονται μεταξύ τους με σκοπό την επίλυση κάποιου προβλήματος.



Εικόνα 8. **Το μοντέλο ενός τεχνητού νευρώνα**(SURPRISE 96 Journal, NEURAL NETWORKS by Christos Stergiou and Dimitrios Siganos, 1996)

Τα Νευρωνικά Δίκτυα, όπως και οι άνθρωποι, μαθαίνουν με παραδείγματα. Για την αναγνώριση κάποιου προτύπου θα πρέπει πρώτα το δίκτυο να τροφοδοτηθεί με παραδείγματα, έτσι ώστε μετά από κάποια παραδείγματα να θεωρείται το δίκτυο εκπαιδευμένο.

Τα Νευρωνικά Δίκτυα είναι παραλλαγές της ιδέας της παράλληλης κατανεμημένης επεξεργασίας (Parallel Distributed Processing), η οποία παρουσιάστηκε το 1986 από τους Mc Clelland .and Rumelhart (Mc Clelland .and Rumelhart ,1986). Σύμφωνα με αυτή την ιδέα, παρουσιάστηκε ένα ολοκληρωμένο μοντέλο νευρωνικού δικτύου του οποίου τα βασικά στοιχεία είναι:

- ένα σύνολο μονάδων επεξεργασίας (νευρώνες)

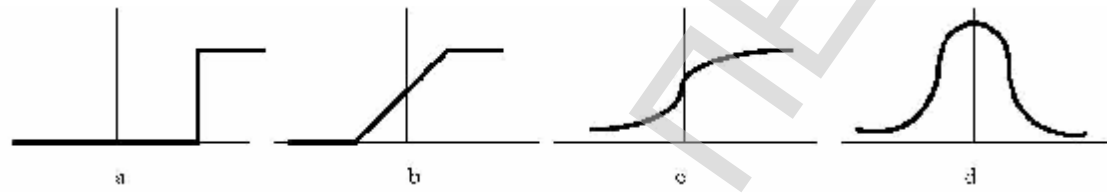
- μια κατάσταση ενεργοποίησης για κάθε μονάδα επεξεργασίας, η οποία ισοδυναμεί με την έξοδο της μονάδας
- συνδέσεις μεταξύ των μονάδων. Για κάθε σύνδεση ορίζεται από ένα βάρος το οποίο καθορίζει το σήμα της μονάδας k προς τη μονάδα i
- ένα κανόνα διάδοσης (propagation rule)
- μια συνάρτηση ενεργοποίησης, η οποία καθορίζει το νέο επίπεδο ενεργοποίησης, στηριζόμενη στην ενεργή είσοδο και στην τρέχουσα κατάσταση ενεργοποίησης
- μια είσοδο πόλωσης (bias/offset) για κάθε μονάδα
- μια μέθοδο εκπαίδευσης
- ένα περιβάλλον στο οποίο θα πρέπει να λειτουργεί και το οποίο παρέχει τα σήματα εισόδου.

Κάθε νευρώνας εκτελεί την εξής εργασία: δέχεται τις εισόδους των γειτονικών νευρώνων ή τις εξωτερικές και τις χρησιμοποιεί για να υπολογίσει το σήμα εξόδου, το οποίο και μεταδίδει στους άλλους νευρώνες.

Ο υπολογισμός της εξόδου οποιουδήποτε νευρώνα του δικτύου δίνεται από τον τύπο:

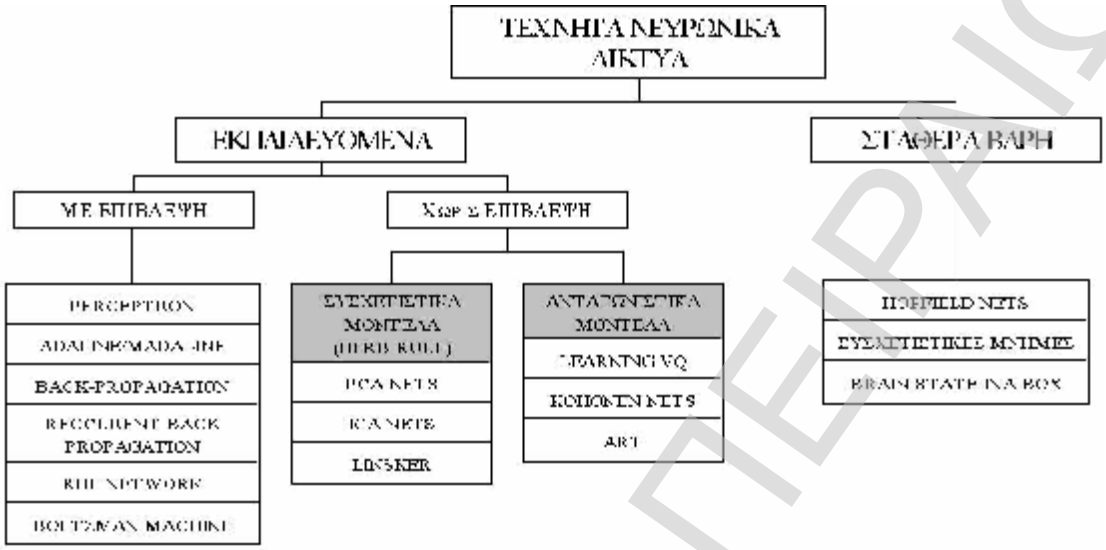
$$y_i = f\left(\sum_{i=1}^{n} w_i x_i - \theta_i\right).$$

Όπου w είναι το συνδετικό βάρος, x ο κόμβος και θ το bias. Η συνάρτηση f είναι γνωστή ως συνάρτηση ενεργοποίησης. Μπορεί να είναι μια απλή συνάρτηση κατωφλίου, σιγμοειδής ή Gaussian. Υπάρχει η δυνατότητα η συνάρτηση αυτή να είναι οποιαδήποτε επιθυμητή μαθηματική συνάρτηση.



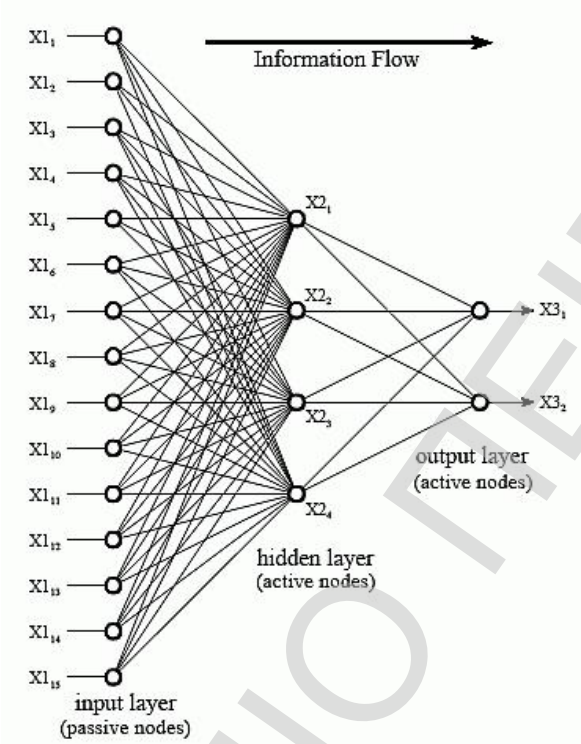
Εικόνα 9. **Τέσσερα από τα βασικότερα είδη συναρτήσεων ενεργοποίησης**:
(a)Threshold, (b)Linear, (c)Sigmoid, (d) Gaussian

Ο κάθε νευρώνας συνδέεται με τους υπόλοιπους και ο τρόπος με τον οποίο η σύνδεση λαμβάνει χώρα καθορίζεται από την αρχιτεκτονική του Νευρωνικού Δικτύου. Επίσης σημείο αναφοράς αποτελεί ο τρόπος με τον οποίο εκπαιδεύονται τα Νευρωνικά Δίκτυα. Οι κυριότερες κατηγορίες Τεχνητών Νευρωνικών Δικτύων ανάλογα με τον τρόπο εκπαίδευσής τους φαίνονται στην Εικόνα 10.



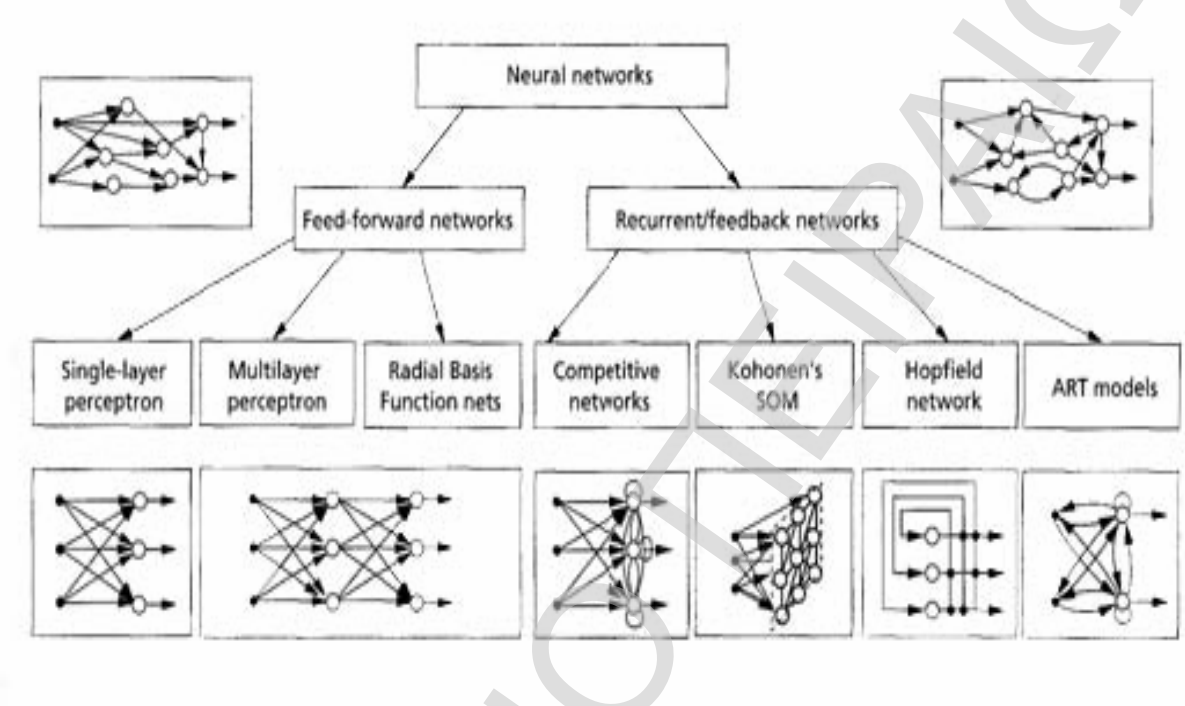
Εικόνα 10. **Διάφορες κατηγορίες Νευρωνικών Δικτύων**

Ένα Νευρωνικό Δίκτυο μπορεί να θεωρηθεί ως κατευθυνόμενος γράφος του οποίου οι νευρώνες είναι οι κόμβοι και οι συνδέσεις μεταξύ των νευρώνων είναι οι ακμές του (Εικόνα 11). Οι νευρώνες ομαδοποιούνται σε επίπεδα ανάλογα με τη λειτουργία τους στο δίκτυο. Έτσι υπάρχει το επίπεδο εισόδου (input layer), το κρυφό επίπεδο (hidden layer), το οποίο μπορεί να παραλειφθεί - που αποτελείται από ένα ή περισσότερα επίπεδα, - και το επίπεδο εξόδου (output layer) στο οποίο έχουμε την έξοδο του δικτύου.



Εικόνα 11. **Το γενικό μοντέλο ενός Νευρωνικού Δικτύου εμπρόσθιας τροφοδότησης με ένα κρυφό επίπεδο** (The Scientist and Engineer's Guide to Digital Signal Processing By Steven W. Smith)

Υπάρχουν δύο βασικές κατηγορίες στις οποίες χωρίζονται τα Νευρωνικά Δίκτυα ανάλογα με την αρχιτεκτονική τους δομή: τα δίκτυα εμπρόσθιας τροφοδότησης (feed-forward) και τα δίκτυα ανατροφοδότησης (feedback ή recurrent). Η βασική τους διαφορά έγκειται στον τρόπο με τον οποίο το σήμα μεταδίδεται μέσα στο δίκτυο. Στα εμπρόσθιας τροφοδότησης δίκτυα το σήμα μεταδίδεται από την είσοδο μόνο προς εμπρός, ενώ στα δίκτυα ανατροφοδότησης μπορεί να μεταδοθεί προς κάθε κατεύθυνση (Εικόνα 12).



Εικόνα 12. **Μοντέλα νευρωνικών δικτύων εμπρόσθιας τροφοδότησης και αναδρομικά (A. K. Jain, J Mao, K.M. Mohiuddin, 1996)**

Πρέπει να σημειωθεί ότι διαφορετικές αρχιτεκτονικές και διασυνδέσεις οδηγούν σε διαφορετική συμπεριφορά του Νευρωνικού Δικτύου. Τα δίκτυα εμπρόσθιας τροφοδότησης είναι πιο στατικά, υπό την έννοια ότι η έξοδος σε κάθε νευρώνα δεν εξαρτάται από την παρούσα κατάστασή του, οπότε θα μπορούσαμε να πούμε ότι δεν έχουν μνήμη και η έξοδός τους είναι ανεξάρτητη από την είσοδο. Αντίθετα, τα δίκτυα ανατροφοδότησης είναι δυναμικά και η έξοδός τους εξαρτάται από την είσοδό τους. Σε κάθε περίπτωση, για διαφορετική αρχιτεκτονική του νευρωνικού δικτύου ακολουθείται διαφορετικός αλγόριθμος εκπαίδευσης.

2.5.1 Η εκπαίδευση των νευρωνικών δικτύων

Η ικανότητα εκμάθησης είναι ένα από τα χαρακτηριστικά της νοημοσύνης. Σε περιβάλλον Νευρωνικών Δικτύων ως εκμάθηση θεωρείται η μεταβολή των συνδετικών βαρών έτσι ώστε το δίκτυο να μπορεί να εκτελέσει συγκεκριμένη λειτουργία, είτε πρόκειται για αναγνώριση προτύπων είτε για ταξινόμηση. Την πηγή της εκπαίδευσης αποτελούν τα πρότυπα εκπαίδευσης (training patterns).

Τρεις είναι οι βασικές φιλοσοφίες εκπαίδευσης ενός Νευρωνικού Δικτύου (Jain, Mao, Mohiuddin, 1996):

- *Επιβλεπόμενη ή Συσχετιζόμενη μάθηση (supervised or associative learning).* Σύμφωνα με αυτό τον τρόπο εκπαίδευσης το δίκτυο τροφοδοτείται με συγκεκριμένες εισόδους, οι οποίες συγκρίνονται με τις αναμενόμενες εξόδους. Το ζεύγος αυτό εισόδων και εξόδων είναι που καθορίζει την εκπαίδευση του δικτύου. Μια σημαντική υποκατηγορία αυτού του είδους της εκπαίδευσης είναι η Ενισχυμένη εκπαίδευση (Reinforcement learning). Το δίκτυο λαμβάνει ένα συνολικό σήμα επιβράβευσης ή τιμωρίας. Τα βάρη μεταβάλλονται έτσι ώστε να αναπτύξουν μια συμπεριφορά εισόδου-εξόδου έτσι ώστε να μεγιστοποιείται η πιθανότητα επιβράβευσης.
- *Μη επιβλεπομένη μάθηση ή αυτο-οργάνωση (Unsupervised learning).* Σύμφωνα με τη μεθοδολογία αυτή μια μονάδα εξόδου εκπαιδεύεται να ανταποκρίνεται σε ομάδες προτύπων που δίνονται στην είσοδο. Με αυτό τον τρόπο το σύστημα ανακαλύπτει χαρακτηριστικά των προτύπων των εισόδων.

- *Υβριδική εκπαίδευση* ***(Hybrid learning)***. Αποτελεί ένα συνδυασμό των παραπάνω προσεγγίσεων. Ένα τμήμα των βαρών εκπαιδεύεται με τη μια μέθοδο και τα υπόλοιπα με την άλλη.

Τρία είναι τα βασικά ζητήματα που πρέπει να αντιμετωπίσει η θεωρία της μάθησης. Ικανότητα (capacity), πολυπλοκότητα δειγμάτων (sample complexity) και υπολογιστική πολυπλοκότητα (computational complexity). Η ικανότητα αφορά στο πόσα πρότυπα μπορεί να αποθηκεύσει το δίκτυο και πόσα όρια απόφασης μπορεί να υλοποιήσει. Η πολυπλοκότητα δειγμάτων αφορά τον αριθμό των προτύπων εκπαίδευσης που χρειάζεται το δίκτυο για να αποκτήσει τη μέγιστη ικανότητα γενίκευσης. Λίγα πρότυπα οδηγούν σε υπερεκπαίδευση (over-fitting), την περίπτωση δηλαδή που το δίκτυο συμπεριφέρεται μεν καλά στα δεδομένα εκπαίδευσης, όμως παρουσιάζει φτωχά αποτελέσματα κατά τη διάρκεια της αξιολόγησης. Η υπολογιστική πολυπλοκότητα αφορά το χρόνο που χρειάζεται το δίκτυο να προσεγγίσει μια λύση κατά τη διάρκεια της εκπαίδευσης.

Όταν αναφερόμαστε σε κανόνες μάθησης, στην ουσία εννοούμε τον τρόπο με τον οποίο μεταβάλλονται τα συναπτικά βάρη κατά τη διάρκεια της εκπαίδευσης. Υπάρχουν τέσσερις βασικές κατηγορίες κανόνων μάθησης:

- κανόνας μείωσης λάθους (error-correction rules). Χρησιμοποιείται στην εποπτευομένη εκπαίδευση. Βασική αρχή του κανόνα αυτού είναι η χρήση του σήματος διαφοράς εισόδου-επιθυμητής εξόδου για τη μεταβολή των βαρών.

- εκπαίδευση Boltzmann. Η εκπαίδευση Boltzmann είναι ένας στοχαστικός κανόνας εκπαίδευσης ο οποίος προέρχεται από τις αρχές της θεωρίας της πληροφορίας και της θερμοδυναμικής.
- κανόνας Hebbian. Είναι ο πρώτος κανόνας που εφαρμόστηκε ποτέ και βασίζεται σε νευροφυσιολογικές παρατηρήσεις.
- ανταγωνιστικοί κανόνες μάθησης (competitive learning rules). Οι κανόνες αυτοί στηρίζονται στην αρχή της ανταγωνιστικότητας μεταξύ των εξόδων για δραστηριοποίηση.

Στην εικόνα 13 καταγράφεται το σύνολο των κανόνων εκπαίδευσης σε συνδυασμό με αρχιτεκτονικές και αλγόριθμους εκπαίδευσης και παρουσιάζεται μια συνολική εικόνα της τρέχουσας κατάστασης στο χώρο των νευρωνικών δικτύων.

Είδος εκπαίδευσης	Κανόνας εκπαίδευσης	Αρχιτεκτονική	Αλγόριθμος εκπαίδευσης	Χώρος που χρησιμοποιείται
Επιβλεπόμενη	Διόρθωση λάθους	Απλό ή πολυεπίπεδο Perceptron	Αλγόριθμος εκπαίδευσης Perceptron, αλγόριθμος οπισθοδιάδοσης του λάθους (BackPropagation), Adaline, Madaline	Αναγνώριση προτύπων, προσέγγιση συναρτήσεων, πρόβλεψη, έλεγχος
	Boltzmann	Αναδρομικά δίκτυα	Αλγόριθμος εκπαίδευσης Boltzmann	Αναγνώριση προτύπων
	Hebbian	Πολυεπίπεδο δίκτυο εμπρόσθιας τροφοδότησης	Ανάλυση γραμμικής διακρίνουσας	Ανάλυση δεδομένων, αναγνώριση προτύπων
	Ανταγωνιστικά	Ανταγωνιστικά δίκτυα	Κανονικοποίηση διανύσματος εκπαίδευσης	Συμπίεση δεδομένων
Μη επιβλεπόμενη	Διόρθωση λάθους	Πολυεπίπεδο Perceptron	Sammon's projection	Ανάλυση δεδομένων
	Hebbian	Εμπρόσθιας τροφοδότησης ή ανταγωνιστικά	Ανάλυση πρωτεύοντος στοιχείου	Ανάλυση δεδομένων, συμπίεση δεδομένων
		Δίκτυα Hopfield	Συσχετιστικες μνήμες	Συσχετιστική μνημη
	Ανταγωνιστικά	Ανταγωνιστικά δίκτυα	Κανονικοποίηση διανύσματος	Κατηγοριοποίηση, συμπίεση δεδομένων
		Δίκτυα Kohonen	Kohonen αλγόριθμος	Κατηγοριοποίηση, ανάλυση δεδομένων
Υβριδικά	Διόρθωση λάθους και ανταγωνιστικά	RBF δίκτυα	RBF αλγόριθμος εκπαίδευσης	Αναγνώριση προτύπων, προσέγγιση συναρτήσεων, πρόβλεψη, έλεγχος

Εικόνα 13. **Κατηγορίες Νευρωνικών Δικτύων με βάση τα χαρακτηριστικά τους**

2.5.2 Εξελικτικά νευρωνικά δίκτυα

Τα Εξελικτικά Νευρωνικά Δίκτυα (Evolutionary Neural Networks, ENNs) είναι το προϊόν της σύγκλισης δύο τεχνολογιών που κατά βάση προσομοιώνουν βιολογικές λειτουργίες, των Νευρωνικών Δικτύων και των Εξελικτικών Αλγορίθμων (Evolution Algorithms, EAs). Τα Εξελικτικά Νευρωνικά Δίκτυα αποτελούν το τεχνολογικό αντίστοιχο της εξέλιξης και της αυτόματης προσαρμογής. Έχουν την ικανότητα να μαθαίνουν, να βελτιστοποιούνται και να προσαρμόζονται αυτόματα, ανάλογα με το πρόβλημα που έχουν να επιλύσουν.

Τα σημεία στα οποία αλληλεπιδρούν τα Εξελικτικά Νευρωνικά Δίκτυα και οι Εξελικτικοί Αλγόριθμοι είναι αρκετά. Στην ουσία σχεδόν κάθε παράμετρος ενός Νευρωνικού Δικτυού μπορεί να βελτιστοποιηθεί. Όμως κάποια σημεία έχουν ιδιαίτερο ενδιαφέρον, μιας και έχουν δώσει πολύ καλά αποτελέσματα σε πειράματα και αποδεικνύουν την υπεροχή των Εξελικτικών Νευρωνικών Δικτύων έναντι των κλασικών Νευρωνικών Δικτύων. Τα σημεία αυτά είναι:

- Τα συναπτικά βάρη (synaptic weights)
- Η αρχιτεκτονική του Νευρωνικού Δικτύου
- Οι κανόνες εκπαίδευσης

Η αλληλεπίδραση ανάμεσα στις συνιστώσες τεχνολογίες ερμηνεύεται από τα κυρίαρχα χαρακτηριστικά τους. Τα Εξελικτικά Νευρωνικά Δίκτυα έχουν την ικανότητα να εκπαιδεύονται και να μαθαίνουν ενώ αντίθετα οι Εξελικτικοί Αλγόριθμοι είναι ισχυρά ευρετικά εργαλεία σε μεγάλους χώρους αναζήτησης. Έτσι τα ENNs χρησιμοποιούν κατά κάποιο τρόπο τους Eas, με σκοπό αυτοί να αναζητήσουν τις βέλτιστες παραμέτρους που κατά κανόνα οδηγούν και σε βέλτιστα Νευρωνικά Δίκτυα. Ένα βέλτιστο Νευρωνικό Δίκτυο είναι κατά κανόνα ταχύτερο, πιο συνεκτικό, με μικρότερο κόστος υλοποίησης και καλύτερη ικανότητα γενίκευσης από ένα μη βέλτιστο.

2.6 Βελτιστοποίηση με Σμήνη Σωματιδίων (Particle Swarm Optimization)

Η βελτιστοποίηση με σμήνη σωματιδίων (Particle Swarm Optimization PSO) είναι μια στοχαστική τεχνική βελτιστοποίησης που βασίζεται στην αναζήτηση σε πληθυσμούς. Ανήκει στην ευρύτερη οικογένεια των εξελικτικών αλγορίθμων και παρουσιάστηκε για πρώτη φορά από τους Eberhart και Kennedy το 1995 (Eberhart και Kennedy, 1995) οι οποίοι τον εμπνεύστηκαν από την συμπεριφορά των πουλιών όταν αναζητούν τροφή.

Η τεχνική PSO έχει αρκετές ομοιότητες με εξελικτικές τεχνικές όπως οι γενετικοί αλγόριθμοι. Το σύστημα αρχικοποιείται κατά τυχαίο τρόπο και αναζητά βέλτιστα σημεία στο χώρο, ενημερώνοντας τις νέες γενεές που δημιουργούνται. Αντίθετα όμως με τους γενετικούς αλγόριθμους, η τεχνική PSO δεν χρησιμοποιεί καθόλου γενετικούς τελεστές όπως η διασταύρωση και η μετάλλαξη αλλά οι εν δυνάμει λύσεις (particles) κινούνται στο χώρο των λύσεων ακολουθώντας τις καλύτερες μέχρι εκείνη τη στιγμή λύσεις. Κάθε σωματίδιο αποθηκεύει την δική του θέση στο χώρο λύσεων, η οποία σχετίζεται με την καλύτερη λύση (best fitness) που έχει επιτύχει το σωματίδιο μέχρι εκείνη τη στιγμή (particle best). Μια άλλη τιμή με την οποία σχετίζεται η θέση του σωματιδίου είναι η καλύτερη λύση που έχει επιτευχθεί από την ομάδα των γειτόνων στην οποία ανήκει το σωματίδιο (local best). Το πλήθος των μελών της ομάδας αυτής εξαρτάται από την αρχική επιλογή που έχει γίνει για την αρχιτεκτονική όλης της ομάδας και μπορεί να είναι από 2 έως το σύνολο του

πληθυσμού. Ο σκοπός της τεχνικής αυτής είναι να οδηγηθεί κάποιο σωματίδιο στη βέλτιστη λύση μεταβάλλοντας την αρχική του θέση κατά μέγεθος που εξαρτάται από την ατομική αλλά και τη συλλογική βέλτιστη λύση.

Ο αλγόριθμος PSO έχει αποδειχθεί αποτελεσματικός για δύο κυρίως λόγους. Ο ένας έχει να κάνει με το γεγονός ότι είναι αρκετά πιο γρήγορος σε ένα ευρύ φάσμα εφαρμογών σε σύγκριση με άλλος εξελικτικούς αλγορίθμους. Ο δεύτερος έχει να κάνει με το γεγονός ότι είναι πολύ απλός, ενώ έχει πολύ λίγες παραμέτρους λειτουργίας.

2.6.1 Ο αλγόριθμος

Ο αλγόριθμος PSO προσομοιώνει τη συμπεριφορά ενός σμήνους πουλιών που αναζητούν τροφή: όταν κάποιο από τα πουλιά εντοπίσει τροφή πετά προς το σημείο εκείνο, και τα υπόλοιπα, ενώ δεν ξέρουν ακριβώς τη θέση της τροφής, ακολουθούν εκείνο ή εκείνα που βρίσκονται πιο κοντά τους. Τελικά κάποιο από όλα θα βρει την τροφή (λύση). Για τον αλγόριθμο, σε κάθε μέλος του σμήνους αντιστοιχεί μια πιθανή λύση. Στην κάθε λύση αντιστοιχεί και ένας βαθμός καταλληλότητας (fitness) ο οποίος πληροφορεί τον αλγόριθμο για το πόσο καλή είναι η λύση αυτή. Ο βαθμός αυτός προκύπτει από μια συνάρτηση καταλληλότητας (fitness function). Η κάθε πιθανή λύση μεταβάλλεται κατά ένα μέγεθος που ονομάζεται ταχύτητα (velocity), η οποία ορίζεται ως η μετακίνηση της πιθανής λύσης προς την κατεύθυνση της

βέλτιστης συνολικής λύσης κατά τρόπο παρόμοιο με αυτόν που μετακινούνται τα μέλη του σμήνους προς το μέλος εκείνο που βρίσκεται πλησιέστερα στη λύση.

Η αρχικοποίηση του αλγορίθμου περιλαμβάνει τη συγκρότηση μιας αρχικής ομάδας από τυχαία σωματίδια τα οποία αναζητούν επαναληπτικά κάποιο βέλτιστο στο χώρο των λύσεων. Σε κάθε επανάληψη, κάθε σωματίδιο ενημερώνει τη θέση του με βάση την προηγούμενη δική του καλύτερη θέση και τη συνολικά καλύτερη θέση. Η σχέση βάση της οποίας ενημερώνεται η ταχύτητα κάθε σωματιδίου είναι:

$$v = v + c1 * rand() * (pbest - present) + c2 * rand() * (gbest - present) \quad (1)$$

ενώ η νέα του θέση στο χώρο των λύσεων δίνεται από τη σχέση:

$$present = present + v \quad (2)$$

όπου

- n είναι η ταχύτητα του σωματιδίου,
- present είναι η τρέχουσα θέση του σωματιδίου
- pbest gbest είναι η μεχρι τώρα θέση που αντιστοιχεί στο καλύτερο fitness του σωματιδίου
- gbest είναι η θέση που αντιστοιχεί στο καλύτερο fitness όλου του συνόλου ή της ομάδας όπου ανήκει το σωματίδιο
- rand () είναι ένας τυχαίος αριθμός ανάμεσα στο 0 και στο 1

– c1, c2 είναι παράγοντες μάθησης που τυπικά παίρνουν τιμές ίσον με 2

Ο ψευδοκώδικας του αλγορίθμου είναι ο ακόλουθος:

```
FOR EACH particle
   initialize particle
END
DO
   FOR EACH particle
   calculate fitness value
   IF fitness best in history current value=pBest
END
SET gBest=best(pBest) for each particle
FOR EACH particle
   calculate newVelocity
   calculate newPosition
   END
WHILE termination creteria is met
```

Εικόνα 14. **Ο αλγόριθμος** Particle Swarm Optimization

Σε κάποιες περιπτώσεις η ταχύτητα παίρνει μεγάλες τιμές, με αποτέλεσμα ο αλγόριθμος να μη συγκλίνει σε βέλτιστη λύση. Για τον λόγο αυτό συχνά χρησιμοποιείται ένα όριο η μέγιστη ταχύτητα (MaxVelocity), με το οποίο περιορίζεται η μέγιστη μετακίνηση των σωματιδίων σε καθορισμένα μεγέθη.

Πριν τη χρησιμοποίηση του αλγορίθμου PSO προηγούνται δύο βήματα. Το ένα είναι η επιλογή της αναπαράστασης της λύσης και το άλλο η επιλογή της συνάρτησης καταλληλότητας (fitness function). Σε σχέση με την αναπαράσταση της λύσης, ο αλγόριθμος PSO μπορεί να δουλέψει εξίσου καλά και με πραγματικούς αριθμούς

αλλά και με δυαδική αναπαράσταση. Ως συνάρτηση καταλληλότητας μπορεί να επιλεγεί οποιαδήποτε συνάρτηση, συνεχής ή όχι, μη διαφοροποιήσιμη, πολυπαραμετρική (multiobjective). Αυτό άλλωστε είναι και από τα μεγαλύτερα πλεονεκτήματα που έχει ο αλγόριθμος στην εφαρμογή εκπαίδευσης νευρωνικών δικτύων.

Όπως είδαμε και στην περιγραφή του αλγορίθμου, υπάρχει ένας αριθμός πλήθος παραμέτρων που πρέπει να καθοριστεί.

Ακολουθεί μια λίστα με τυπικές παραμέτρους του αλγόριθμου, οι οποίες εξαρτώνται κυρίως από το είδος του προς επίλυση προβλήματος (http://www.swarmintelligence.org).

- *Ο αριθμός των σωματιδίων*. Το τυπικό εύρος είναι 20 ως 40. Για τα περισσότερα προβλήματα ένα πλήθος από 10 σωματίδια είναι συχνά αρκετό. Για πιο δύσκολα προβλήματα μπορεί το πλήθος να αυξηθεί στα 100 ή 200 σωματίδια. Είναι προφανές ότι αύξηση των σωματιδίων συνεπάγεται μείωση της ταχύτητας του αλγορίθμου.
- *Η Διάσταση των σωματιδίων*. Καθορίζεται από το είδος του προβλήματος όπως και το εύρος τιμών.
- *Η μέγιστη ταχύτητα* ***vMax***. Ορίζει το εύρος της μέγιστης μετακίνησης που μπορεί να κάνει ένα σωματίδιο σε κάθε επανάληψη. Το εύρος εξαρτάται και από τη συνάρτηση καταλληλότητας και από το πρόβλημα και αυτό που επιτυγχάνεται είναι ο περιορισμός της αναζήτησης σε μικρότερες περιοχές.

- *Οι παράγοντες εκμάθησης c1 c2*. Παίρνουν συνήθως την τιμή c1=c2=2. Σε κάθε περίπτωση, το εύρος τους κυμαίνεται στο διάστημα (0,4).

- *Η συνθήκη ολοκλήρωσης του αλγορίθμου*. Υπάρχουν δύο τρόποι για τον τερματισμό του αλγορίθμου. Ο ένας είναι να ολοκληρωθεί ο μέγιστος αριθμός επαναλήψεων, ο οποίος συνήθως για μέτριας δυσκολίας προβλήματα είναι οι 1000 επαναλήψεις. Ο άλλος είναι ο βαθμός καταλληλότητας ή η μεταβολή του να φτάσει σε κάποιο προκαθορισμένο επίπεδο. Η δεύτερη επιλογή χρησιμοποιείται συνήθως στην εκπαίδευση νευρωνικών δικτύων.

- *Αρχιτεκτονική global ή local*. Οι δύο αυτές εκδοχές εξαρτώνται από το αν την πληροφορία για ολικό βέλτιστο το κάθε σωματίδιο θα την παίρνει από το σύνολο των σωματιδίων (global) ή από ένα μέρος αυτών (local ή neighborhood). Η global επιλογή προσφέρει ταχύτερη σύγκλιση του αλγορίθμου, όμως υπάρχει ο κίνδυνος για παγίδευση του αλγορίθμου σε τοπικό ακρότατο.

2.7 Εξέλιξη Νευρωνικών Δικτύων με Σμήνη Σωματιδίων

Τα οφέλη από την εξέλιξη χαρακτηριστικών των Νευρωνικών Δικτύων είναι αρκετά. Τα πολυεπίπεδα perceptrons έχουν αποδειχτεί ικανοί προσεγγιστές (universal approximators), δεδομένου ότι μπορούν να προσεγγίσουν αποτελεσματικά οποιαδήποτε συνάρτηση, όσο πολύπλοκη και αν είναι αυτή. Μια από τις πρώτες εφαρμογές PSO ήταν η εξέλιξη των συναπτικών βαρών των Νευρωνικών Δικτύων

αντικαθιστώντας έτσι κλασικούς αλγορίθμους εκπαίδευσης όπως ο back propagation (Eberhart, Simpson, and Dobbins 1996). Επίσης ο PSO χρησιμοποιήθηκε και για την εξεύρεση της βέλτιστης αρχιτεκτονικής των Νευρωνικών Δικτύων, ένα πρόβλημα για το οποίο ο μόνος τρόπος επίλυσης που υπήρχε ήταν η δοκιμή-και-λάθος (trial and error), καθώς επίσης και την κανονικοποίηση και την κλιμάκωση των δεδομένων εισόδου.

Η εκπαίδευση Νευρωνικών Δικτύων με τον PSO συνίσταται στην εύρεση των κατάλληλων συνδετικών βαρών του Νευρωνικού Δικτύου έτσι ώστε το Μέσο Τετραγωνικό Σφάλμα Εξόδου (Mean Square Error – MSE) για το σύνολο των δεδομένων εκπαίδευσης να πάρει μια ελάχιστη τιμή.

$$mse = \frac{\sum_{k=0}^{patterns} \sum_{i=0}^{output} \left(output_i - target_i\right)_k^2}{training\ patterns}$$

Συνάρτηση Μέσου Τετραγωνικού Σφάλματος για το σύνολο των προτύπων εκπαίδευσης

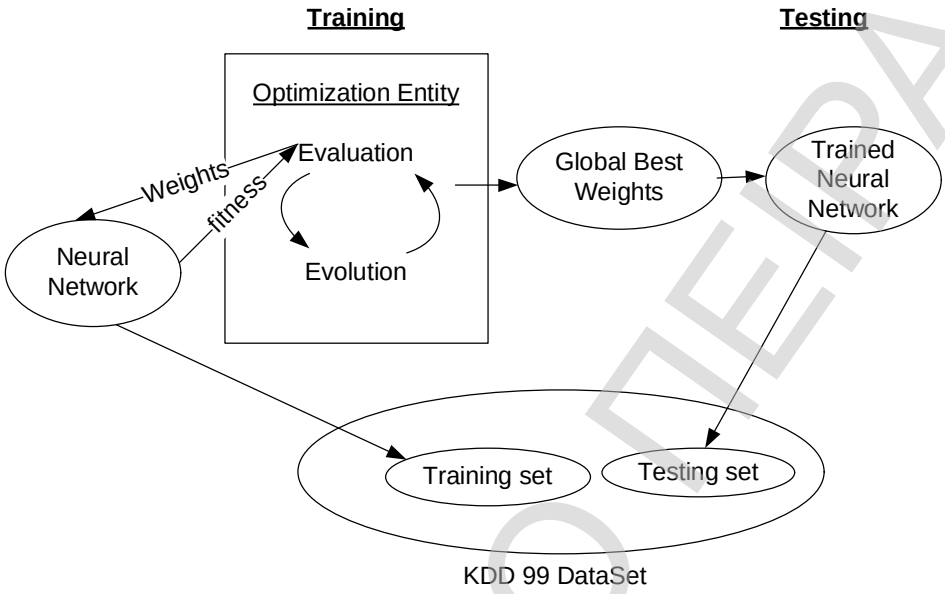
Όταν το MSE πάρει μια αρκετά μικρή τιμή, το νευρωνικό δίκτυο θεωρείται εκπαιδευμένο και μπορεί να αξιολογηθεί με βάση τα συνδετικά βάρη που αντιστοιχούν στο ελάχιστο Μέσο Τετραγωνικό Σφάλμα που έχει ανευρεθεί.

3. ΑΝΑΠΤΥΞΗ ΕΞΕΛΙΚΤΙΚΟΥ ΝΕΥΡΩΝΙΚΟΥ ΔΙΚΤΥΟΥ ΓΙΑ ΤΗΝ ΑΝΤΙΜΕΤΩΠΙΣΗ ΔΙΚΤΥΑΚΩΝ ΕΠΙΘΕΣΕΩΝ

Όπως προαναφέρθηκε, ο σκοπός της εργασίας αυτής είναι η μελέτη και η αξιολόγηση ενός μηχανισμού ανάλυσης επιθέσεων. Στο παρόν κεφάλαιο παρουσιάζονται τα βήματα του σχεδιασμού και της υλοποίησης αυτού του μηχανισμού καθώς και τα δεδομένα που χρησιμοποιήθηκαν για την αξιολόγηση. Ο σχεδιασμός έγινε με την αντικειμενοστραφή προσέγγιση και η ανάπτυξη σε περιβάλλον Java.

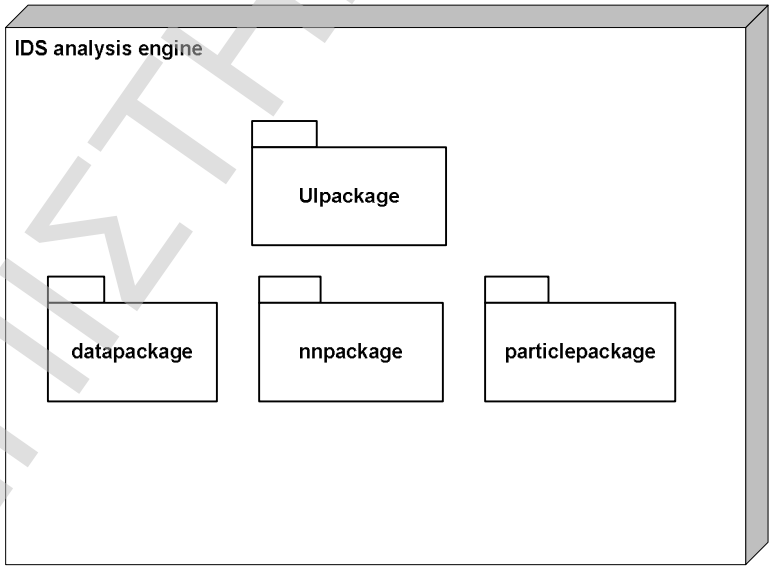
3.1 Αρχιτεκτονικός σχεδιασμός

Για την κατασκευή της μηχανής ανάλυσης επιλέχθηκε η από πάνω προς τα κάτω προσέγγιση, με τη δημιουργία αρχικά ενός γενικού μοντέλου του μηχανισμού, που απεικονίζεται στην εικόνα 15. Περιλαμβάνει δυο αυτόνομες οντότητες: την οντότητα της εκπαίδευσης και την οντότητα της αξιολόγησης. Η οντότητα της εκπαίδευσης (Training Module) περιλαμβάνει τον αλγόριθμο βελτιστοποίησης PSO και το νευρωνικό δίκτυο εμπρόσθιας τροφοδότησης ενώ η οντότητα αξιολόγησης (Testing module) περιλαμβάνει το εκπαιδευμένο νευρωνικό δίκτυο εμπρόσθιας τροφοδότησης. Οι δύο μονάδες συνδέονται με κοινό υποσύστημα τροφοδοσίας δεδομένων.



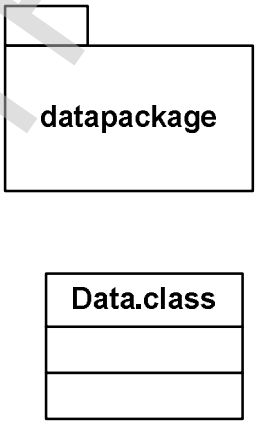
Εικόνα 15. **Γενικό αρχιτεκτονικό μοντέλο της μηχανής ανάλυσης**

Στη συνέχεια έχουμε την ανάλυση της μηχανής στο επίπεδο πακέτων και κλάσεων.



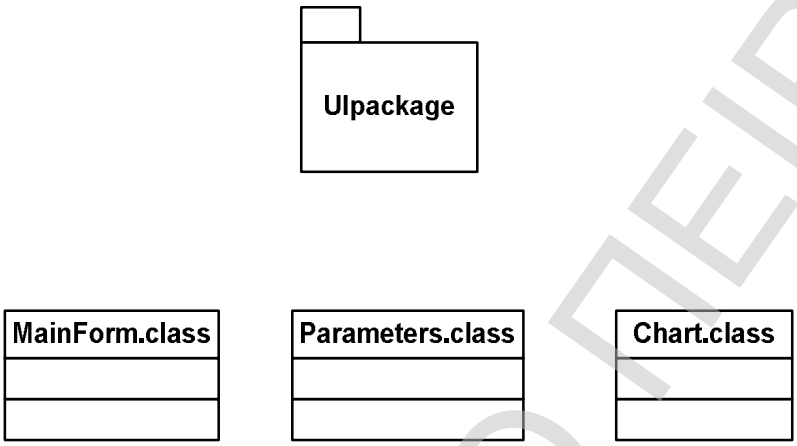
Εικόνα 16. **Τα** packets **που απαρτίζουν την εφαρμογή**

- *Datapackage*. Το πακέτο αυτό περιλαμβάνει μια κλάση, τη data.class η οποία είναι η κλάση διαχείρισης των δεδομένων. Η διαχείριση περιλαμβάνει α) την προεπεξεργασία. Τα διάφορα δεδομένα για την εκπαίδευση και την αξιολόγηση του νευρωνικών δικτύων δεν βρίσκονται πάντα στην επιθυμητή μορφή. Για το λόγο αυτό υπόκεινται σε προεπεξεργασία με σκοπό να διευκολυνθεί το νευρωνικό δίκτυο στη διαδικασία της εκπαίδευσης. Παράδειγμα προεπεξεργασίας δεδομένων είναι η μετατροπή μη αριθμητικών χαρακτηριστικών (πχ κατηγοριών), σε αριθμητικά. Επίσης άλλο παράδειγμα προεπεξεργασίας είναι η κανονικοποίηση αριθμητικών δεδομένων στο διάστημα (0,1). Αυτό γίνεται στην περίπτωση που κάποια από τα χαρακτηριστικά έχουν μεγάλες αποκλίσεις ως προς το εύρος τιμών που μπορούν να πάρουν στο training set. Τα νευρωνικά δίκτυα αξιολογούν καλύτερα τις μεγάλες τιμές, αγνοώντας τις μικρότερες.



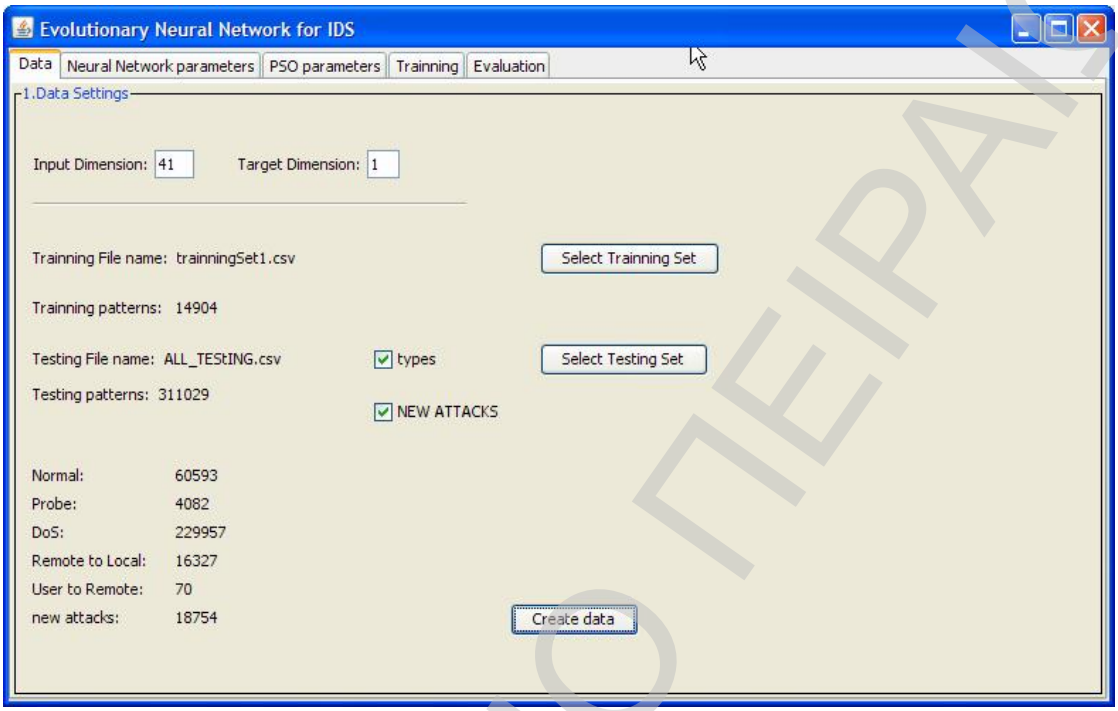
Εικόνα 17. **Το πακέτο** datapackage **περιέχει την κλάση** data.class

Ø *UIpackage*. Το πακέτο αυτό περιλαμβάνει τρεις κλάσεις, οι οποίες σχετίζονται με το user interface της μηχανής ανάλυσης (Εικόνα 18).



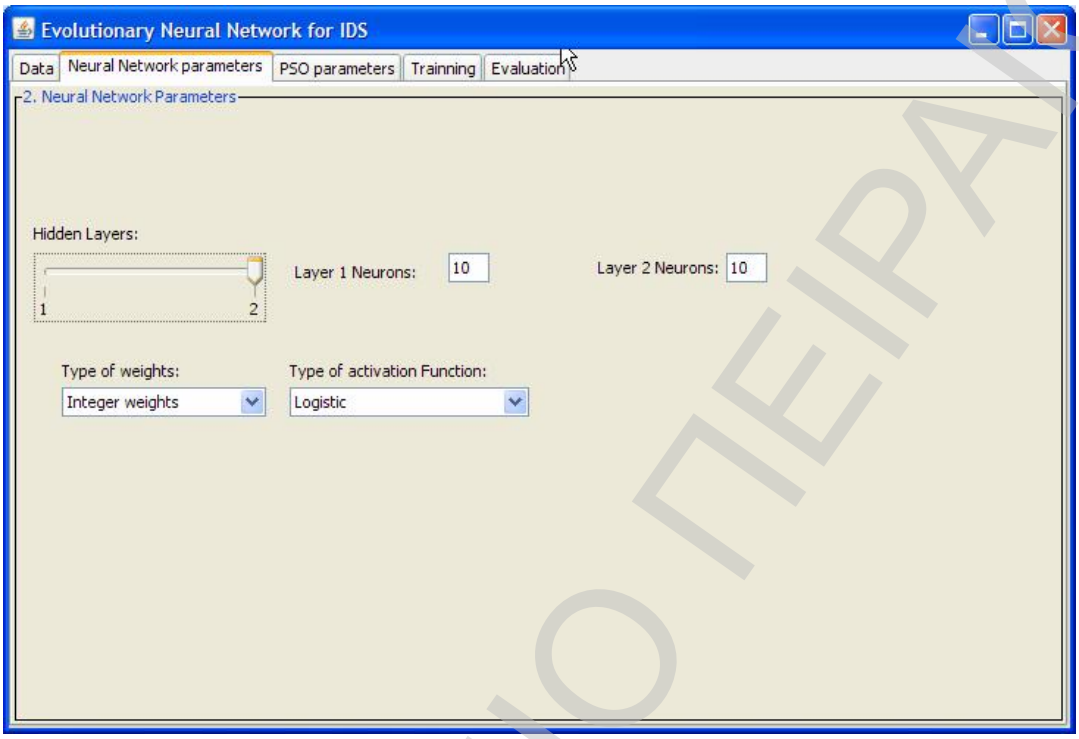
Εικόνα 18. **Οι κλάσεις του** UIpackage

– *MainForm.class*. Υλοποιεί τη φόρμα μέσα από την οποία επιλέγονται οι βασικές παράμετροι της μηχανής.



Εικόνα 19. **Φόρμα εισαγωγής των δεδομένων για την εκπαίδευση και αξιολόγηση του αλγορίθμου**

Μέσα από το data tab (Εικόνα 19) επιλέγονται τα αρχεία δεδομένων. Από τα αρχεία αυτά δημιουργείται το Training Set με τα δεδομένα εκπαίδευσης, αφού πρώτα γίνει κανονικοποίηση των δεδομένων και το Testing set με τα δεδομένα για την αξιολόγηση του νευρωνικού δικτύου.



Εικόνα 20. **Φόρμα εισαγωγής παραμέτρων Νευρωνικού Δικτύου**

Μέσα από το NeuralNetwork parameters tab (Εικόνα 20) επιλέγονται παράμετροι που αφορούν το Νευρωνικό Δίκτυο. Αυτές είναι:

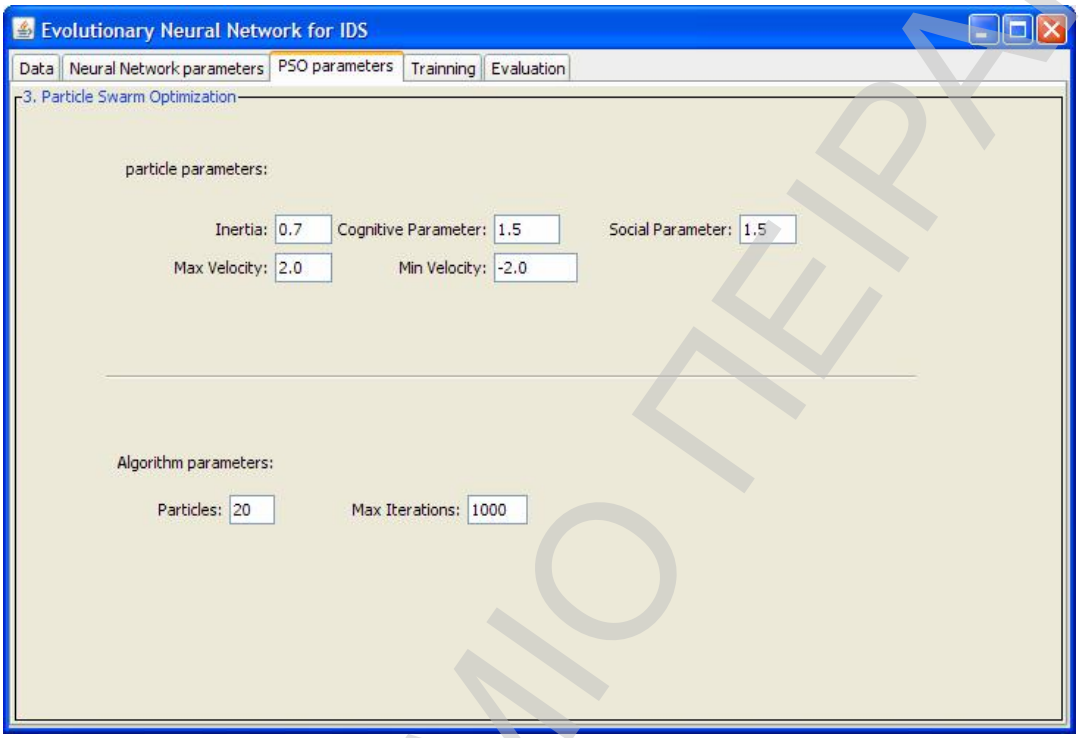
a. Ο αριθμός των κρυφών επιπέδων. Συνήθως για απλά προβλήματα η χρήση ενός μόνο κρυφού επιπέδου είναι αρκετή. Το δεύτερο επίπεδο είναι απαραίτητο στην περίπτωση που τα δεδομένα αντιστοιχούν σε ασυνεχείς χώρους.

b. ο αριθμός των νευρώνων. Δεν υπάρχει κανόνας που να ορίζει τον ακριβή αριθμό των νευρώνων για κάθε πρόβλημα. Εδώ υπάρχουν οι εξής επιλογές: ή με την διαδικασία trial and error δοκιμάζονται διάφορες τιμές με σκοπό να βρεθεί αυτή που δίνει το μικρότερο

σφάλμα εξόδου ή στηριζόμαστε σε διάφορους εμπειρικούς κανόνες και τη βιβλιογραφία για παρόμοια προβλήματα ή αναθέτουμε την εύρεση του ιδανικού αριθμού νευρώνων σε κάποιο εξελικτικό αλγόριθμο. Η τελευταία, αν και είναι αποτελεσματική επιλογή, είναι εξαιρετικά χρονοβόρα. Αρχικά υπήρξε η σκέψη μαζί με την εκπαίδευση του νευρωνικού δικτύου ο αλγόριθμος PSO να αναζητεί και τον βέλτιστο αριθμό νευρώνων. Η ιδέα όμως εγκαταλείφτηκε γιατί θα απαιτούσε πολύ μεγάλο χρόνο εκπαίδευσης εξ αιτίας και του μεγάλου training set. Γενικά ο πολύ μικρός αριθμός νευρώνων επιφέρει μικρή απόδοση, ενώ ο πολύ μεγάλος επιφέρει το πρόβλημα του over fitting, δηλαδή την προσαρμογή του νευρωνικού δικτύου μόνο στα δεδομένα εκπαίδευσης, με αποτέλεσμα να μην μπορεί να γενικεύσει καλά.

c. ο τύπος δεδομένων των συνδετικών βαρών. Μπορεί να επιλεγεί ο τύπος δεδομένων double ή ο τύπος Integer. Η επιλογή αυτή έχει ιδιαίτερη σημασία όταν κάποιος θέλει να πειραματιστεί με Νευρωνικά Δίκτυα με ακέραια συνδετικά βάρη. Τα νευρωνικά δίκτυα αυτά έχουν μερικές ιδιαιτερότητες με σημαντικότερη αυτή του ότι είναι κατάλληλα για υλοποιήσεις σε hardware.

d. το είδος της συνάρτησης ενεργοποίησης. Μπορεί να επιλεγεί η γραμμική, η βηματική ή η λογιστική συνάρτηση ενεργοποίησης.



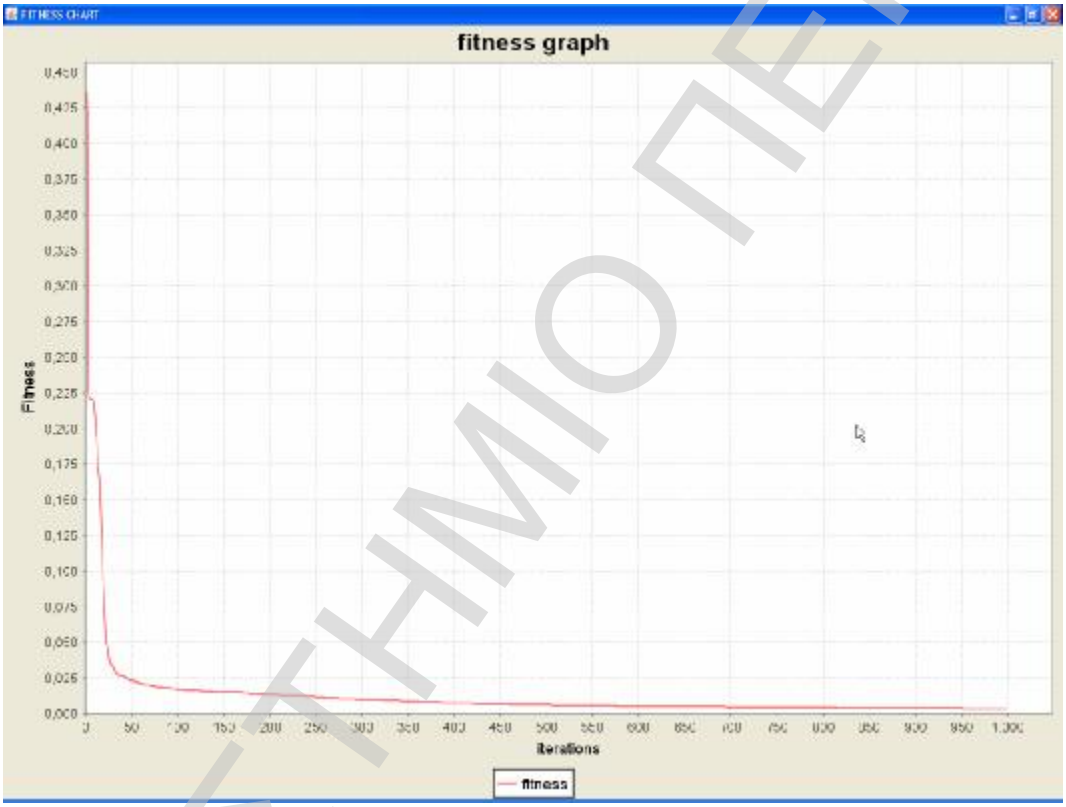
Εικόνα 21. **Φόρμα εισαγωγής παραμέτρων Εξελικτικού Αλγορίθμου**

Στο PSO parameters tab (Εικόνα 21) εισάγονται οι παράμετροι που αφορούν τον PSO αλγόριθμο. Η παράμετρος Inertia, ο μέγιστος αριθμός επαναλήψεων, το πλήθος των σωματιδίων κλπ.. Η παράμετρος inertia προστέθηκε σε νεότερη έκδοση του αλγορίθμου (Shi and Eberhard 1998a, 1998b) και ο ρόλος της είναι να ισοροπήσει τον αλγόριθμο ανάνεσα σε ολική (global) και τοπική (local) αναζήτηση.

– *Parameters.class* Η κλάση αυτή είναι βοηθητική για το πέρασμα των παραμέτρων από την MainForm φόρμα στις διάφορες κλάσεις. Η

δημιουργία κρίθηκε απαραίτητη λόγου του πλήθους των παραμέτρων.

Περιέχει μόνο δεδομένα μεταβλητών κλάσης.

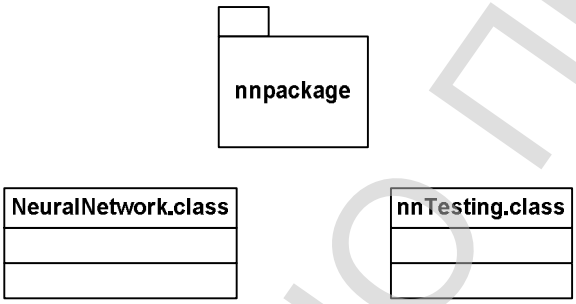


Εικόνα 22. Η φόρμα απεικόνισης του γραφήματος fitness κατά την διάρκεια της εκπαίδευσης του Νευρωνικού Δικτύου

Graph.class. Υλοποιεί τη φόρμα μέσα από την οποία απεικονίζεται η μεταβολή του βαθμού καταλληλότητας σε μια απλή γραφική παράσταση.

Ø Nnpackage. Το πακέτο αυτό περιέχει τις κλάσεις που υλοποιούν τα νευρωνικά δίκτυα (Εικόνα 23).

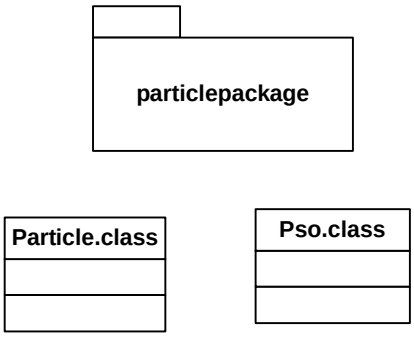
– *NeuralNetwork.class.* Η κλάση αυτή υλοποιεί το πολυεπίπεδο perceptron που χρησιμοποιείται για την εκπαίδευση.



Εικόνα 23. **Οι κλάσεις του πακέτου** nnpackage

– *nnTesting.class.* η κλάση αυτή υλοποιεί το πολυεπίπεδο perceptron για την αξιολόγηση. Στην ουσία είναι το ίδιο με το προηγούμενο με την προσθήκη μερικών μεθόδων για στατιστικούς λόγους.

Ø *Particlepackage*. Το πακέτο αυτό (Εικόνα 24) περιέχει δύο κλάσεις



Εικόνα 24. **Οι κλάσεις του πακέτου** particlepackage

– *Particle.class.* Η κλάση αυτή υλοποιεί ένα σωματίδιο που πρόκειται μέσα από την διαδικασία εκτέλεσης του αλγορίθμου να αποδώσει τη βέλτιστη λύση. Στην περίπτωση της εκπαίδευσης του νευρωνικού δικτύου αντιστοιχεί στα συναπτικά του βάρη. Σε κάθε σωματίδιο αξιολογείται με βάση τον βαθμό καταλληλότητας, ο οποίος όσο μικρότερος είναι τόσο πιο κοντά βρίσκεται το σωματίδιο στη βέλτιστη λύση. Ο βαθμός καταλληλότητας για το νευρωνικό δίκτυο αντιστοιχεί στο μέσο τετραγωνικό σφάλμα για το σύνολο των δεδομένων εκπαίδευσης.

– *Pso.class.* Η κλάση αυτή υλοποιεί τον αλγόριθμο Particle Swarm Optimization. Μόλις ολοκληρωθεί ο αριθμός επαναλήψεων αποθηκεύει τη βέλτιστη λύση που έχει βρει (συνδετικά βάρη που αντιστοιχούν στον μικρότερο βαθμό καταλληλότητας). Τα συνδετικά βάρη χρησιμοποιούνται από το instance της κλάσης nnTesting.class για την αξιολόγηση του αλγορίθμου.

3.2 Ο αλγόριθμος εκπαίδευσης

Ο αλγόριθμος εκπαίδευσης είναι μια παραλλαγή του βασικού αλγόριθμου Βελτιστοποίησης με Σμήνη Σωματιδίων (Εικόνα 14). Στη συνέχεια περιγράφονται τα βήματα εκτέλεσης:

```
INITIALIZE particles
WHILE(NOT maxIteration)
        FOREACH particle
                EVALUATE(particle)
                SAVE localBest
SAVE gloabalBest
COMPUTE newPosition
END WHILE
```

Εικόνα 25. Ο αλγόριθμος εκπαίδευσης του Νευρωνικού Δικτύου

Στη συνέχεια καταγράφονται τα βήματα του αλγόριθμου (Εικόνα 25) αναλυτικά:

INITIALIZE particle. Στο βήμα αυτό αρχικοποιούνται τα particles. Έχουμε ήδη δει και πριν ότι τα σωματίδια για τον αλγόριθμο εκπαίδευσης του νευρωνικού δικτύου αντιστοιχούν στα συνδετικά βάρη. Αρχικοποίηση λοιπόν των συνδετικών βαρών κατά απολύτως τυχαίο τρόπο και παράλληλα αρχικοποίηση της αρχικής ταχύτητας

(velocity) των σωματιδίων. Επίσης παίρνουν τις αρχικές μέγιστες τιμές ο βαθμός καταλληλότητας του κάθε σωματιδίου και ο συνολικά καλύτερος βαθμός καταλληλότητας.

WHILE…END WHILE. Αποτελεί το κύριο μέρος του αλγορίθμου. Ο βρόχος αυτός εκτελείται επαναληπτικά για όσες φορές του ορίσουμε στις παραμέτρους του αλγορίθμου. Εναλλακτικά θα μπορούσε να εκτελεστεί έως ότου το globalFitness φτάσει μέχρι μία προκαθορισμένη τιμή.

FOREACH particle..SAVE globalBest. Κάθε σωματίδιο, δηλαδή κάθε πιθανή λύση αξιολογείται. Η αξιολόγηση περιλαμβάνει τα δικά της βήματα:

- Την αρχικοποίηση του νευρωνικού δικτύου βάση των παραμέτρων που έχουν ήδη δοθεί.
- Το πέρασμα του σωματιδίου (συνδετικά βάρη) στο νευρωνικό δίκτυο.
- Άνοιγμα του training set. Κάθε pattern διατρέχει το νευρωνικό και στην έξοδο υπολογίζεται το MSE.
- Υπολογίζεται το μέσο τετραγωνικό σφάλμα για όλο το Training Set. Αυτό αντιστοιχεί στον βαθμό καταλληλότητας του σωματιδίου.

Με αυτό τον τρόπο αξιολογείται το κάθε σωματίδιο, δηλαδή τα συνδετικά βάρη του Νευρωνικού Δικτύου.

SAVE local..SAVE global. Υπολογίζονται τα localBest για κάθε σωματίδιο και το globalBest για όλο το σμήνος.

COMPUTE newPosition. Ο αλγόριθμος PSO υπολογίζει τη νέα ταχύτητα και τη νέα θέση σύμφωνα με τους τύπους που περιγράφηκαν σε προηγούμενο κεφάλαιο για κάθε σωματίδιο, για να ακολουθήσει αξιολόγηση εκ νέου.

Το αποτέλεσμα της εκτέλεσης του αλγορίθμου για το νευρωνικό δίκτυο είναι ένα βέλτιστο σύνολο συνδετικών βαρών που ελαχιστοποιεί το σφάλμα εξόδου για όλο το training set. Συνεπώς, με αυτά τα συνδετικά βάρη το νευρωνικό δίκτυο θεωρείται εκπαιδευμένο και μπορεί να δοκιμαστεί ως προς την αποτελεσματικότητά του δηλαδή να ταξινομεί σωστά τα δεδομένα του testing set.

3.3 Δεδομένα και αξιολόγηση

Η λογική του εγχειρήματος αυτής της εργασίας συνοψίζεται στο να «εκπαιδεύσουμε» μια μηχανή ώστε να αναγνωρίζει τύπους επιθέσεων, τροφοδοτώντας τη με δεδομένα, σύμφωνα με το τι είναι επίθεση και τι είναι κανονικά δεδομένα. Μετά την ολοκλήρωση της εκπαίδευσης αναμένουμε από τη μηχανή αυτή να αναγνωρίζει και να ταξινομεί σωστά δεδομένα με παρόμοια χαρακτηριστικά. Αυτή η ιδιότητα ονομάζεται γενίκευση. Όπως έχει προαναφερθεί, η μηχανή ανάλυσης που προτείνεται με τη συγκεκριμένη εργασία χρησιμοποιεί το μοντέλο anomaly detection για την αναγνώριση παρεισφρήσεων σε δικτυακό περιβάλλον. Αυτό σημαίνει ότι θα πρέπει να τροφοδοτηθεί με δεδομένα εκπαίδευσης έτσι ώστε να μπορέσει να αναγνωρίσει τις διάφορες επιθέσεις. Η εξεύρεση δεδομένων είναι αρκετά δύσκολη διαδικασία δεδομένου ότι:

a. Αν χρησιμοποιηθεί κάποιος μηχανισμός λήψης δεδομένων (data capturing) απ' ευθείας από το δίκτυο ο όγκος που θα προκύψει θα είναι εξαιρετικά μεγάλος και

b. Είναι πολύ δύσκολο να γίνει ανάλυση σε πρωτογενή δεδομένα δικτύου γιατί θα πρέπει να αναλυθεί το κάθε πακέτο χωριστά, να καθοριστεί σε ποιο δικτυακό session ανήκει, και να καθοριστεί εάν πρόκειται για επίθεση ή κανονική κίνηση έτσι ώστε να του αποδοθεί ή ανάλογη «ετικέτα» για την εκπαίδευση με επιβλεπόμενη μάθηση που απαιτείται για τον τύπο νευρωνικού δικτύου που έχει επιλεγεί.

Το πρόβλημα για όλους του ερευνητές IDS σε σχέση με τα δεδομένα που θα πρέπει να χρησιμοποιηθούν έχει λυθεί με την υιοθέτηση του συνόλου δεδομένων KDD '99 cup (International Knowledge Discovery and Data Mining Tools Competition, http://www.sigkdd.org) ως σημείου αναφοράς. Τα δεδομένα αυτά καθώς και τα αποτελέσματα του διαγωνισμού έχουν δεχθεί αρκετή κριτική, παρόλα αυτά είναι τα μοναδικά διαθέσιμα δεδομένα στην ερευνητικά κοινότητα για αξιολόγηση νέων ID συστημάτων ή αλγορίθμων.

Ο αλγόριθμος ταξινόμησης για το Εξελικτικό Νευρωνικό δίκτυο είναι αρκετά απλός και εφαρμόζεται στην έξοδο του νευρωνικού δικτύου κατά τη διάρκεια εκτέλεσης του testing set. Στη συνέχεια δίνεται ο ψευδοκώδικας για τον αλγόριθμο που ταξινομεί τα αποτελέσματα του testing set ανάλογα με την τιμή της εξόδου του Νευρωνικού Δικτύου και της ετικέτας του δείγματος που ταξινομεί.

Πίνακας 1. **Αλγόριθμος με τον οποίο διαχωρίζονται τα δεδομένα εξόδου στο Εξελικτικό Νευρωνικό Δίκτυο**

```
CASE OF(output=target=1)

        rightNormal++

CASE OF (output=target=0)

        rightAttack++

                IF(newTypeOfAttack) newAttacks++;

                IF (attack type= Probe) rightProbe++;

                IF (attack type= DoS) rightdDoS++;

                IF (attack type= U2R) rightU2R++;

                IF (attack type= R2L) rightR2L++;

CASE OF (output=0,target=1)

        False alarm++

CASE OF (output=1,target=0)

        Wrong classification

END
```

3.4 Διαγωνισμός KDD '99 cup

Δεδομένης της σημασίας και του γενικότερου ενδιαφέροντος για την ανίχνευση παρεισφρήσεων, το εργαστήριο Lincoln του αμερικανικού πανεπιστημίου MIT ασχολήθηκε με τη δημιουργία ενός συνόλου δεδομένων (DARPA 1998 dataset) το οποίο θα χρησιμοποιούνταν κυρίως για την αξιολόγηση ID συστημάτων που είχαν χρηματοδοτηθεί από το ίδιο το ίδρυμα. Κατ' επέκταση, θα μπορούσε να χρησιμοποιηθεί ως σημείο αναφοράς για τους ερευνητές που ανέπτυσσαν συστήματα ανίχνευσης παρεισφρήσεων και αλγόριθμους με διαφορετικές μεθοδολογίες. Για το λόγο αυτό, το 1998 σχεδιάστηκε ένα δίκτυο υπολογιστών στο οποίο θα υπήρχαν εξυπηρετητές-στόχοι διαφορετικών λειτουργικών συστημάτων στους οποίους θα έτρεχαν διάφορες δικτυακές υπηρεσίες. Στο δίκτυο θα υπήρχαν επιπλέον υπολογιστές για τη δημιουργία εσωτερικής αλλά και εικονικής εξωτερικής κίνησης και τέλος υπολογιστής sniffer για την καταγραφή της κίνησης του δικτύου σε TCP dump μορφή. Το δίκτυο λειτούργησε για επτά εβδομάδες με δικτυακή κίνηση η οποία αντιστοιχούσε σε αναμενόμενη κίνηση του δικτύου και επιθέσεις οι οποίες αντιστοιχούν σε τέσσερεις βασικούς τύπους :

- Αρνηση παροχής υπηρεσιών (Denial of Service). Ο επιτιθέμενος προσπαθεί να αποτρέψει τους χρήστες από την χρήση υπηρεσιών του δικτύου.
- Πρόσβαση χωρίς εξουσιοδότηση (Remote to Local). Ο επιτιθέμενος προσπαθεί να συνδεθεί σε κάποιο υπολογιστή-στόχο ενώ δεν έχει δικαίωμα.

- Προσπάθεια πρόσβασης με διαχειριστικά δικαιώματα (User to Root). Ο επιτιθέμενος προσπαθεί να αποκτήσει περισσότερα δικαιώματα από αυτά που του έχουν εκχωρηθεί.
- Επιθέσεις διερεύνησης (Probe). Ο επιτιθέμενος προσπαθεί να εντοπίσει αδυναμίες των υπολογιστών στόχων.

Στο χρονικό αυτό διάστημα, τα ID συστήματα που επρόκειτο να αξιολογηθούν «εκπαιδεύονταν» από τα δεδομένα του δικτύου σε κανονική κίνηση και επιθέσεις. Στη συνέχεια, για τις επόμενες δυο εβδομάδες, στο ίδιο εικονικό δίκτυο που είχε στηθεί, έγινε η αξιολόγηση των συστημάτων με την ίδια διαδικασία. Τα δεδομένα όπως και τα αποτελέσματα από όλη αυτή τη διαδικασία έγιναν διαθέσιμα σε άλλα ιδρύματα και ερευνητές από όπου όμως προέκυψαν αρκετές ενστάσεις σχετικά με την αξιοπιστία των δεδομένων και των αποτελεσμάτων. Έτσι η διαδικασία επαναλήφθηκε το 1999 με την προσθήκη και νέων υπολογιστών στόχων (Windows NT), νέων επιθέσεων τύπου Stealth και την υλοποίηση κάποιας πολιτικής ασφάλειας για το δίκτυο, έτσι ώστε το όλο σενάριο να γίνει πιο ρεαλιστικό. Παρόλα αυτά, οι ενστάσεις και η κριτική παρέμειναν (McHugh., 2001) και συνοψίζονται ως εξής:

- Η κίνηση που δημιουργήθηκε στο δίκτυο δεν είναι ρεαλιστική. Στην πραγματικότητα υπάρχουν στο διαδίκτυο περισσότερα είδη πακέτων, πολλά από τα οποία είναι αδιευκρίνιστα (κακοσχεδιασμένα πρωτόκολλα, νέες υπηρεσίες κλπ).
- Μια προσεκτική ματιά στα headers των πακέτων κατέδειξε ότι υπήρχε διαφοροποίηση στο πεδίο TTL (126 για κανονική κίνηση, 127 για επιθέσεις),

πράγμα που θα έκανε εύκολη τη διάκριση για τις διάφορες ID υλοποιήσεις (Malhony and Chan, 2003).

- Από τη διαδικασία της αξιολόγησης έλειψαν τελείως signature based συστήματα, τα οποία θα μπορούσαν να εμφανίσουν καλύτερα τελικά αποτελέσματα (Brugger, Chow, 2005).

Όμως, παρά την κριτική για τα DARPA δεδομένα και αποτελέσματα, αυτά παραμένουν τα μόνα αξιόπιστα που έχει αυτή τη στιγμή στη διάθεσή της η ερευνητική κοινότητα για την αξιολόγηση ID συστημάτων.

Περαιτέρω επεξεργασία των DARPA δεδομένων, όπως συνένωση πακέτων με ίδιες IP διευθύνσεις αποστολέα και λήπτη, που κινήθηκαν σε προκαθορισμένα χρονικά διαστήματα με το ίδιο πρωτόκολλο καθώς και δεδομένα από τα Log αρχεία των υπολογιστών στόχων αποτέλεσαν το dataset για τον διαγωνισμό KDD ’99 cup (Knowledge Discovery and Data Mining Tools Competition). Αριθμητικά, στο σύνολο των δεδομένων του KDD 99 cup το 94% (περίπου 5.000.000 στιγμιότυπα) ανήκει στο DARPA 1998 dataset και το 6% (περίπου 310.000 στιγμιότυπα) στο DARPA 1999. Η κάθε μια από αυτές τις «συνδέσεις» περιλαμβάνει 41 χαρακτηριστικά (features) που έχουν ομαδοποιηθεί ως εξής:

- *Βασικά χαρακτηριστικά*. Τα βασικά χαρακτηριστικά αποτελούνται από πληροφορίες που βρίσκονται μόνο στις επικεφαλίδες των πακέτων (Πίνακας 2).

Πίνακας 2. **Βασικά χαρακτηριστικά** TCP **συνδέσεων**

Όνομα χαρακτηριστικού	*Περιγραφή*	*Τύπος*
duration	διάρκεια της σύνδεσης σε δευτερόλεπτα	Συνεχής
protocol_type	τύπος πρωτοκόλλου πχ tcp, udp	Διακριτός
service	υπηρεσία, e.g., http, telnet, etc.	Διακριτός
src_bytes	αριθμός bytes από την πηγή προς τον στόχο	Συνεχής
dst_bytes	αριθμός bytes από τον στόχο προς την πηγή	Συνεχής
flag	κατάσταση σύνδεσης (κανονική ή με λάθη)	Διακριτός
land	1 αν η σύνδεση αφορά τον ίδο στόχο; 0 διαφορετικά	Διακριτός
wrong_fragment	αριθμός λανθασμένων τμημάτων (fragments)	Συνεχής
urgent	αριθμός πακέτων με ένδειξη urgent	Συνεχής

- *Χαρακτηριστικά περιεχομένου*. Πληροφορίες που βρίσκονται στο payload του πακέτου (Πίνακας 3).

Πίνακας 3. Χαρακτηριστικά περιεχομένου

Όνομα χαρακτηριστικού	*Περιγραφή*	*Τύπος*
hot	αριθμός δεικτών με την ένδειξη ‘hot”	Συνεχής
num_failed_logins	αριθμός αποτυχημένων προσπαθειών σύνδεσης	Συνεχής
logged_in	1 αν η σύνδεση είναι επιτυχημένη; 0 διαφορετικά	Διακριτός
num_compromised	αριθμός διαπιστωμένων αδυναμιών	Συνεχής
root_shell	1 αν έχει επιτευχθεί σύνδεση σε κέλυφος root; 0 διαφορετικά	Διακριτός
su_attempted	1 αν έχει γίνει προσπάθεια να εκτελεστεί η εντολή ``su root'; 0 διαφορετικά	Διακριτός
num_root	αριθμός συνδέσεων ως root	Συνεχής
num_file_creations	αριθμός εντολών δημιουργίας αρχείων	Συνεχής
num_shells	αριθμός συνδέσεων σε κέλυφος	Συνεχής
num_access_files	αριθμός προσπελάσεων σε αρχεία ελέγχου	Συνεχής
num_outbound_cmds	αριθμός εντολών σε σύνδεση ftp	Συνεχής
is_hot_login	1 αν η σύνδεση περιλαμβάνεται στή ``hot" λίστα; 0 διαφορετικά	Διακριτός
is_guest_login	1 αν η σύνδεση έχει γίνει με τον λογαριασμό ”guest” 0 διαφορετικά	Διακριτός

- *Χαρακτηριστικά βασισμένα στο χρόνο*. Περιλαμβάνονται χαρακτηριστικά που έχουν ολοκληρωθεί σε χρονικό παράθυρο 2 δευτερολέπτων. Παράδειγμα ο αριθμός των συνδέσεων με τον ίδιο υπολογιστή σε χρονικό διάστημα 2 sec (Πίνακας 4).

Πίνακας 4.Χαρακτηριστικά βασισμένα στο χρόνο

Όνομα χαρακτηριστικού	*Περιγραφή*	*Τύπος*
count	αριθμός των συνδέσεων τα τελευταία δύο δευτερόλεπτα	Συνεχής
serror_rate	% των συνδέσεων που έχουν λάθη τύπου ``SYN''	Συνεχής
rerror_rate	% των συνδέσεων που έχουν λάθη τύπου ``REJ''	Συνεχής
same_srv_rate	% των συνδέσεων στην ίδια υπηρεσία	Συνεχής
diff_srv_rate	% των συνδέσεων σε διαφορετική υπηρεσία	Συνεχής
srv_count	Αριθμός των συνδέσεων στην ίδια υπηρεσία τα τελευταία δύο δευτερόλεπτα	Συνεχής
srv_serror_rate	% των συνδέσεων που έχουν λάθη τύπου ``SYN'' και προέρχονται από τον στόχο	Συνεχής
srv_rerror_rate	% των συνδέσεων που έχουν λάθη τύπου ``REJ'' και προέρχονται από τον στόχο	Συνεχής
srv_diff_host_rate	% των συνδέσεων προς διαφορετικούς στόχους	Συνεχής

- Χαρακτηριστικά βασισμένα σε Hosts. Όμοια με το προηγούμενο, μόνο που αφορά παράθυρο 100 συνδέσεων αντί για χρόνο, για τον ίδιο υπολογιστή (Πίνακας 5).

Πίνακας 5. Χαρακτηριστικά βασισμένα σε host

Όνομα χαρακτηριστικού	*Περιγραφή*	*Τύπος*
dst_host_count		Συνεχής
dst_host_srv_count		Συνεχής
dst_host_same_srv_rate		Συνεχής
dst_host_diff_srv_rate		Συνεχής
dst_host_same_src_port_rate		Συνεχής
dst_host_srv_diff_host_rate		Συνεχής
dst_host_serror_rate		Συνεχής
dst_host_srv_serror_rate		Συνεχής
dst_host_rerror_rate		Συνεχής
dst_host_srv_rerror_rate		Συνεχής

Το KDD 99 dataset έχει χωριστεί σε τρία υποσύνολα δεδομένων (Πίνακας 6).

Πίνακας 6. Περιεχόμενα του KDD 99 dataset

DataSet	DoS	Probe	U2R	R2L	Normal
10% KDD	391458	4107	52	1126	97227
Corrected KDD	229853	4166	70	16347	60593
Whole KDD	3883370	41102	52	1126	979780

Το "10% KDD" δημιουργήθηκε για να αποτελέσει το Training dataset. Περιέχει 22 τύπους επιθέσεων (Πίνακας 7) και είναι μια πιο συνεκτική μορφή του "Whole KDD". Όπως φαίνεται, περιέχει περισσότερες εγγραφές επιθέσεων από ότι κανονικής κίνησης και επιπλέον – λόγω και της φύσης της επίθεσης – περισσότερες DoS επιθέσεις. Το "Corrected KDD" (Πίνακας 8) περιέχει διαφορετικές στατιστικές κατανομές για τις επιθέσεις αλλά η κυριότερη διαφορά έχει να κάνει με το γεγονός ότι περιέχει 14 νέες επιθέσεις οι οποίες δεν υπάρχουν στο " 10% KDD". Έτσι, αυτό το σύνολο δεδομένων καθίσταται ιδανική περίπτωση ως σύνολο δεδομένων αξιολόγησης (Testing set) μιας και η προτεινόμενη μεθοδολογία μπορεί να αξιολογηθεί ως προς την ικανότητά της να ανιχνεύει άγνωστες επιθέσεις.

Πίνακας 7. **Ανάλυση δειγμάτων επίθεσης και κανονικής κίνησης για το** "10% KDD" dataset

Επίθεση	*Εγγραφές*	*Κατηγορία επίθεσης*
smurf	280790	dos
neptune	107201	dos
back	2203	Dos
teardrop	979	dos
pod	264	Dos
land	21	dos
normal	97277	Normal
satan	1589	probe
ipsweep	1247	probe
portsweep	1040	Probe
nmap	231	probe
warezclient	1020	r2l
guess_passwd	53	r2l
warezmaster	20	r2l
imap	12	r2l
ftp_write	8	r2l
multihop	7	r2l
phf	4	r2l
spy	2	r2l
buffer_overflow	30	u2r
rootkit.	10	u2r
loadmodule	9	u2r
perl	3	u2r

Τα αποτελέσματα από τον διαγωνισμό KDD 99 χρησιμοποιούνται ως σημείο αναφοράς για κάθε νέα προσέγγιση που χρησιμοποιεί τα δεδομένα από αυτό το διαγωνισμό.

Πίνακας 8. **Η κατανομή των δεδομένων στο** "Corrected" dataset

	Εγγραφές	%	Υπάρχει στο Training Set	Τύπος
apache2.	794	,3	OXI	DoS
back.	1098	,4		DoS
buffer_overflow.	22	,0	OXI	U2R
ftp_write.	3	,0	OXI	R2L
guess_passwd.	4367	1,4		R2L
httptunnel.	158	,1	OXI	R2L
imap.	1	,0		R2L
ipsweep.	306	,1		Probe
land.	9	,0		DoS
loadmodule.	2	,0		U2R
mailbomb.	5000	1,6	OXI	DoS
mscan.	1053	,3	OXI	Probe
multihop.	18	,0		DoS
named.	17	,0	OXI	R2L
neptune.	58001	18,6		DoS
nmap.	84	,0		DoS
normal.	60593	19,5		Normal
perl.	2	,0		U2R
phf.	2	,0		R2L
pod.	87	,0		DoS
portsweep.	354	,1		Probe
processtable.	759	,2	OXI	DoS
ps.	16	,0	OXI	U2R
rootkit.	13	,0		U2R
saint.	736	,2	OXI	Probe
satan.	1633	,5		Probe
sendmail.	17	,0	OXI	R2L
smurf.	164091	52,8		DoS
snmpgetattack.	7741	2,5	OXI	R2L

snmpguess.	2406	,8	OXI	R2L
sqlattack.	2	,0	OXI	U2R
teardrop.	12	,0		DoS
udpstorm.	2	,0	OXI	DoS
warezmaster.	1602	,5		R2L
worm.	2	,0	OXI	DoS
xlock.	9	,0	OXI	R2L
xsnoop.	4	,0	OXI	R2L
xterm.	13	,0	OXI	U2R
Total	311029	100,0		

Κατά τον ίδιο τρόπο θα αξιολογηθεί και η προσέγγιση αυτής της εργασίας. Στον διαγωνισμό αυτό συμμετείχαν 24 ερευνητικές προτάσεις και τα αποτελέσματα του νικητή φαίνονται στον Πίνακα 9.

Πίνακας 9. **Αποτελέσματα του νικητή στον** KDD 99 **διαγωνισμό**

	Ποσοστό αναγνώρισης
Normal	99.5%
Probe	83.3%
Denial of Service	97.1%
User to Root	13.2%
Remote to Local	8.4%

Από τα αποτελέσματα παρατηρούμε αρκετά μεγάλα ποσοστά αναγνώρισης για την κανονική κίνηση και τις επιθέσεις τύπου Probe και DoS. Επιπλέον παρατηρούμε ότι στα είδη επιθέσεων Remote to Local και User to Root υπάρχουν χαμηλά ποσοστά επιτυχούς αναγνώρισης. Μια ερμηνεία που μπορεί να δοθεί είναι ότι η παρουσία αυτού του είδους επιθέσεων στο Training set ήταν κατά πολύ μικρότερη σε σχέση με την παρουσία των δειγμάτων Normal, Probe και DoS.

4. ΠΕΙΡΑΜΑΤΙΚΗ ΕΦΑΡΜΟΓΗ ΤΟΥ ΕΞΕΛΙΚΤΙΚΟΥ ΝΕΥΡΩΝΙΚΟΥ ΔΙΚΤΥΟΥ

4.1 Προεπεξεργασία δεδομένων

Τα δεδομένα που θα χρησιμοποιηθούν για την εκπαίδευση και την αξιολόγηση του Εξελικτικού Νευρωνικού Δικτύου είναι αυτά χρησιμοποιήθηκαν στον διαγωνισμό KDD 99. Παρόλη τη μεγάλη ακαδημαϊκή αποδοχή και ερευνητική χρήση έχουν εντοπιστεί κάποια προβλήματα σε σχέση με την κατανομή των δεδομένων αυτών, τα οποία είναι:

- Η κατανομή στο σύνολο των δεδομένων δεν είναι ρεαλιστική. Το 80% των δεδομένων είναι δεδομένα επίθεσης, κάτι που δεν συμβαίνει στην πραγματικότητα. Το πρόβλημα προκύπτει εξ αιτίας του γεγονότος ότι κάποιες επιθέσεις δεν δημιουργούν πλήρεις συνδέσεις αλλά είναι για παράδειγμα επιθέσεις εκπομπής (broadcasted). Τέτοιο παράδειγμα είναι η επίθεση smurf, που ενώ είναι icmp broadcast και δεν απαιτεί πλήρη σύνδεση, κάθε πακέτο καταλαμβάνει μια εγγραφή στο dataset.
- Η κατανομή των επιθέσεων μέσα στο dataset δεν είναι ρεαλιστική μιας και κυριαρχούν κυρίως δεδομένα από DoS επιθέσεις, ενώ δεν αντιπροσωπεύονται επαρκώς επιθέσεις τύπου remote to local ή user to remote.

Αυτή η άνιση κατανομή θα οδηγούσε το νευρωνικό δίκτυο να αναγνωρίσει καλύτερα τις επιθέσεις για τις οποίες είχε καλύτερη πληροφορία, εις βάρος αυτών που έχουν μικρότερη παρουσία στο dataset. Ένα άλλο πρόβλημα είναι το μεγάλο μέγεθος του

dataset. Θα ήταν πρακτικά αδύνατο να χρησιμοποιηθεί όλη αυτή η ποσότητα δεδομένων για την εκπαίδευση του Νευρωνικού Δικτύου.

Μια γνωστή πρακτική είναι η μείωση -μέσα από στατιστικές διαδικασίες- κάποιων χαρακτηριστικών (feature extraction). Υπάρχει μια σχετική ερευνητική δουλειά η οποία στηρίζεται στη θεωρία πληροφορίας και σχετίζει κάθε χαρακτηριστικό (feature) με κάθε μορφή επίθεσης, ανάλογα με την πληροφορία που περιέχει το χαρακτηριστικό. Άλλη πρακτική που χρησιμοποιείται είναι η δημιουργία με απόλυτα τυχαίο τρόπο ενός μικρότερου δείγματος για δημιουργηθεί το training set. Απαραίτητη προϋπόθεση είναι στο δείγμα αυτό να περιέχεται πληροφορία για όλα τα είδη επιθέσεων.

Με στόχο την παράκαμψη αυτών των προβλημάτων που αναφέρθηκαν, στην παρούσα εργασία ακολουθείται μια διαδικασία παρόμοια με αυτή η οποία προτείνεται και από τους Lascov et al (Lascov et al, 2004). Η διαδικασία αυτή μπορεί να περιγραφεί ως εξής:

1. Επιλογή από το σύνολο δεδομένων “10% KDD” όλων των εγγραφών και διαχωρισμός σε δυο υποσύνολα: το ένα με δεδομένα μόνο από κανονική κίνηση και το άλλο μόνο με δεδομένα επιθέσεων.
2. Το υποσύνολο με δεδομένα μόνο κανονικής κίνησης περιέχει 97.277 εγγραφές οι οποίες χωρίζονται σε δύο υποσύνολα. Το ένα υποσύνολο κανονικής κίνησης που περιέχει μόνο την υπηρεσία http με 61.000 εγγραφές και το άλλο υποσύνολο με περίπου 36.000 εγγραφές που περιέχει όλες τις άλλες υπηρεσίες. Γίνεται διαχωρισμός δηλαδή με βάση την υπηρεσία για

κάθε δείγμα των δεδομένων της κανονικής κίνησης. Από το πρώτο υποσύνολο, με απόλυτα τυχαίο τρόπο και επιλογή όλων των τύπων υπηρεσιών, επιλέγεται το 10% και από το δεύτερο υποσύνολο επιλέγεται το 1%. Προκύπτει έτσι ένα σύνολο δειγμάτων κανονικής κίνησης που περιέχει 4.152 εγγραφές.

3. Το υποσύνολο με τα δεδομένα των επιθέσεων περιέχει 396.744 εγγραφές με δεδομένα επιθέσεων. Αυτό το υποσύνολο χωρίζεται σε ένα υποσύνολο με δεδομένα επίθεσης smurf με 280.790 εγγραφές, ένα σύνολο με δεδομένα επίθεσης Neptune με 107.201 εγγραφές και όλα τα υπόλοιπα είδη με 8.753 εγγραφές. Το τελευταίο υποσύνολο επιλέγεται ολόκληρο, ενώ από τα άλλα δύο επιλέγονται από 1.000 εγγραφές με απόλυτα τυχαίο τρόπο. Προκύπτει έτσι ένα σύνολο με 10.753 εγγραφές, που περιλαμβάνει εγγραφές από όλα τα είδη των επιθέσεων.
4. Τα δύο σύνολα κανονικής κίνησης και επιθέσεων ενώνονται σε ένα, σχηματίζοντας το training set με 14.905 εγγραφές.
5. Το testing set αποτελείται από όλο το “Corrected KDD” δηλαδή 311.029 εγγραφές (πίνακας 8). Στο testing set περιλαμβάνονται όλες οι επιθέσεις που υπάρχουν στο training set καθώς και νέες άγνωστες επιθέσεις (πίνακας 11) και επιθέσεις τύπου stealth.

Στον πίνακα 10 παρουσιάζεται ο ακριβής αριθμός από κάθε είδος επίθεσης που υπάρχει στο training set.

Πίνακας 10. **Κατανομή δεδομένων για το** trainingSet

	Εγγραφές	%	% αθροιστικά
back.	2203	14,8	14,8
buffer_ove	30	,2	15,0
ftp_write.	8	,1	15,0
guess_pass	53	,4	15,4
imap.	12	,1	15,5
ipsweep.	1247	8,4	23,8
land.	21	,1	24,0
loadmodule	9	,1	24,0
multihop.	7	,0	24,1
neptune.	1000	6,7	30,8
nmap.	231	1,5	32,4
normal	4152	27,9	60,2
perl.	3	,0	60,2
phf.	4	,0	60,3
pod.	264	1,8	62,0
portsweep.	1040	7,0	69,0
rootkit.	10	,1	69,1
satan.	1589	10,7	79,7
smurf.	1000	6,7	86,4
spy.	2	,0	86,5
teardrop.	979	6,6	93,0
warezclien	1020	6,8	99,9
warezmaste	20	,1	100,0
Total	14905	100,0	

Στον πίνακα 11 παρουσιάζεται ο ακριβής αριθμός των εγγραφών που υπάρχουν στο testing set για κάθε τύπο επίθεσης καθώς και ο αριθμός δειγμάτων που δεν υπάρχουν στο training set.

Πίνακας 111. **Η κατανομή στο** correct KDD 99 **σύνολο δεδομένων παλαιών και νέων «άγνωστων» επιθέσεων**.

	Τύπος επίθεσης					
	normal	probe	dos	Remote to local	User to remote	Σύνολο
Εγγραφές από τύπους επιθέσεων που υπάρχουν στο training set	60.593	2.293	223.400	5.972	17	292.275
Εγγραφές από επιθέσεις που δεν υπάρχουν στο training set	0	1.789	6.557	10.355	53	18.754
σύνολο	60.593	4.082	229.957	16.327	70	311.029

Η διαδικασία της δημιουργίας του training set και του testing set ακολουθείται από τη διαδικασία της κωδικοποίησης και της κανονικοποίησης των δεδομένων. Κάποια από τα χαρακτηριστικά όπως τα έχουμε περιγράψει δεν είναι αριθμητικά, αλλά αποτελούν κατηγορίες, όπως το χαρακτηριστικό protocol με τις κατηγορίες πρωτοκόλλων TCP, UDP ICMP, και θα πρέπει να κωδικοποιηθούν. Αυτό γίνεται αντικαθιστώντας την κάθε τιμή με ακέραιο αριθμό ξεκινώντας από το 1 έως το πλήθος των διαφορετικών κατηγοριών. Έτσι, για παράδειγμα, το χαρακτηριστικό protocol γίνεται για icmp = 1 για tcp 2 κλπ. Επίσης θα πρέπει να κωδικοποιηθεί η

έξοδος. Για το λόγο αυτό δημιουργείται μια νέα στήλη, στην οποία για κάθε label=’normal’ δίνουμε την τιμή 1 ενώ σε κάθε άλλη περίπτωση την τιμή 0. Δηλαδή οι εγγραφές κανονικής κίνησης είναι 1 και οι εγγραφές επιθέσεων είναι 0. Η επιλογή αυτή επιβάλλεται και εξ αιτίας της λογιστικής συνάρτησης εξόδου, που δίνει τιμές στο διάστημα (0,1).

Μια άλλη τροποποίηση στο testing set είναι η κωδικοποίηση των επιθέσεων για στατιστικούς λόγους. Οι επιθέσεις κωδικοποιούνται ως ακέραιοι αριθμοί, ξεκινώντας από το 1 για DoS 2 για Probe κλπ. Αυτό γίνεται για να εντοπιστεί στο τέλος για ποια είδη επιθέσεων παρουσιάζει καλύτερες επιδόσεις το νευρωνικό δίκτυο. Επίσης κωδικοποιείται το αν το είδος της επίθεσης υπάρχει στο training set (παίρνει την τιμή 0) ή αν δεν βρίσκεται στο training set (παίρνει την τιμή 1)

Τέλος, για το σύνολο των δεδομένων εκπαίδευσης και αξιολόγησης γίνεται κανονικοποίηση στο διάστημα (0,1). Αυτό γίνεται για να μπορεί το νευρωνικό δίκτυο να συγκλίνει ευκολότερα, δεδομένου ότι κάποια από τα χαρακτηριστικά όπως τα src_bytes και dst_bytes έχουν μεγάλες αποκλίσεις στις ανώτερες και κατώτερες τιμές που λαμβάνουν.

4.2 Επιλογή παραμέτρων

Η επιλογή των διάφορων παραμέτρων του εξελικτικού αλγορίθμου είναι μια αρκετά σύνθετη διαδικασία, δεδομένου ότι δεν υπάρχει κάποιος γενικός κανόνας που να καθορίζει τις επιλογές των παραμέτρων ανεξάρτητα από το πρόβλημα. Υπάρχουν διάφοροι εμπειρικοί κανόνες σχετικά με αρχιτεκτονική του Νευρωνικού Δικτύου και

του αλγορίθμου PSO, αλλά στην εργασία αυτή εφαρμόστηκε η τεχνική trial and error, δηλαδή οι συνεχείς δοκιμές με αλλαγές, με σκοπό να βρεθεί το βέλτιστο σύνολο παραμέτρων. Επίσης, θα πρέπει να σημειωθεί ότι ούτε στην βιβλιογραφία εντοπίστηκε παρόμοια έρευνα για το συγκεκριμένο πρόβλημα των ID συστημάτων, έτσι ώστε να γίνει χρήση των παραμέτρων που χρησιμοποιήθηκαν εκεί.

Οι τελικές επιλογές που έγιναν παρουσιάζονται στον Πίνακα 12. α πρέπει όμως να δεχτούμε ότι το σύνολο αυτό μπορεί να μην είναι το βέλτιστο. Μια προσπάθεια εύρεσης του βέλτιστου συνδυασμού παραμέτρων για την εκπαίδευση ενός Εξελικτικού Νευρωνικού Δικτύου θα μπορούσε να αποτελέσει αντικείμενο μελλοντικής έρευνας.

Πίνακας 12. **Πίνακας παραμέτρων του Εξελικτικού Νευρωνικού Δικτύου για τη εκτέλεση των πειραμάτων**

Παράμετροι Νευρωνικού Δικτύου	
Architecture Δικτύου	Εμπρόσθια τροφοδότηση
Hidden layers	2
nodes	10-3
Activation function	Λογιστική
Fitness	Μέσο τετραγωνικό σφάλμα
Παράμετροι PSO αλγορίθμου	
Architecture	Global
Inertia	0,7
Cognitive parameter	1,5
Social Parameter	1,5
Max Veocity	0,5
Min Velocity	-0,5
Particles	20
Max Iterations	600

4.3 Πειραματικά αποτελέσματα

Με στόχο αυξημένη αξιοπιστία των πειραμάτων, και δεδομένου ότι ο αλγόριθμος PSO είναι στοχαστικός, το πείραμα εκτελέστηκε 30 φορές. Για κάθε πείραμα καταμετρήθηκε:

- ο τελικός βαθμός καταλληλότητας
- το ποσοστό των σωστά αναγνωρισμένων δειγμάτων (επιθέσεων και κανονικής κίνησης)
- το ποσοστό των σωστά ταξινομημένων επιθέσεων ανά είδος

- το ποσοστό των false alarms.

Στη συνέχεια, από τις 30 αυτές δοκιμές βγήκαν οι στατιστικοί μέσοι όροι, όπως παρουσιάζονται στον πίνακα 13.

Πίνακας 13. Συνοπτικός πίνακας αποτελεσμάτων

	average	median	Std deviation
fitness	0,018886	0,018887	0,001898
Συνολικό ποσοστό σωστά ταξινομημένων δειγμάτων	92,80%	92,91%	0,27%
Ποσοστό σωστά αναγνωρισμένων επιθέσεων	91,81%	91,87%	0,32%
Συνολικό Ποσοστό λάθος ταξινομημένων δειγμάτων	7,20%	7,09%	0,27%
Συνολικό ποσοστό σωστά αναγνωρισμένων νέων επιθέσεων	19,23%	19,04%	1,01%
Ποσοστό σωστά ταξινομημένων "normal" δειγμάτων	96,88%	96,86%	0,37%
Ποσοστό σωστά ταξινομημένων "Probe" δειγμάτων	92,20%	92,76%	1,71%
Ποσοστό σωστά ταξινομημένων "DoS" δειγμάτων	97,74%	97,74%	0,06%
Ποσοστό σωστά ταξινομημένων "Remote to Local" δειγμάτων	8,30%	9,35%	4,06%
Ποσοστό σωστά ταξινομημένων "User to remote" δειγμάτων	52,86%	53,57%	5,59%
Ποσοστό false alarms	0,61%	0,61%	0,07%

Η μέση χρονική διάρκεια εκπαίδευσης του Εξελικτικού Νευρωνικού Δικτύου ήταν περίπου 32 λεπτά. Η πρώτη γραμμή παρουσιάζει το μέσο όρο, η δεύτερη τον μέσο και η τρίτη την τυπική απόκλιση για τις τριάντα επαναλήψεις που έγιναν.

Τα αποτελέσματα αυτά συγκρίνονται με τα αποτελέσματα του διαγωνισμού KDD 99, από όπου προκύπτει ο πίνακας 14. Θα πρέπει να αναφερθεί ότι τα αποτελέσματα του διαγωνισμού αυτού αφορούν την καλύτερη επίδοση από όλες όσες πήραν μέρος. Τα αποτελέσματα αναλύονται περαιτέρω στο επόμενο κεφάλαιο.

Πίνακας 14. **Συγκριτικός πίνακας αποτελεσμάτων ανάμεσα στα αποτελέσματα του** KDD 99 **διαγωνισμού και στο Εξελικτικό Νευρωνικό Δίκτυο της εργασίας**

	KDD 99 Classifier Learning Contest	Evolutionary Neural Network
Normal	99,5%	96,88%
Probe	83,3%	92,20%
DoS	97,1%	97,74%
User to Root	13,2%	52,86%
Remote to Local	8,4%	8,30%

5. ΑΠΟΤΕΛΕΣΜΑΤΑ

Από την προσεκτική μελέτη των αποτελεσμάτων και ειδικότερα από τον Πίνακα 14 μπορούν να εξαχθούν χρήσιμα συμπεράσματα για την αποτελεσματικότητα του Εξελικτικού Νευρωνικού Δικτύου ως μηχανή ανάλυσης ενός IDS.

Καταρχήν, η συνολική εικόνα κρίνεται ενθαρρυντική. Τα ποσοστά επιτυχίας σε τρία είδη επιθέσεων είναι καλύτερα από αυτά του διαγωνισμού KDD 99,. ειδικότερα σε ότι αφορά επιθέσεις Probe, Dos και User to Root. Μάλιστα, στην πρώτη και στην τρίτη περίπτωση επιτυγχάνονται αρκετά καλύτερα ποσοστά αναγνώρισης από αυτά του διαγωνισμού.

Τα αποτελέσματα κρίνονται ως ενθαρρυντικά για δύο ακόμα λόγους. Ο πρώτος είναι ότι στην περίπτωση του DoS τύπου επίθεσης, λόγω του τρόπου δειγματοληψίας, έμεινε ένα μεγάλο μέρος εκτός training set, αφού οι επιθέσεις smurf και Neptune συμμετείχαν με λιγότερο από 1% στο τελικό σύνολο. Αν δηλαδή οι δύο αυτοί τύποι επιθέσεων είχαν περισσότερες εγγραφές στη διαδικασία της εκπαίδευσης, πιθανότατα το αποτέλεσμα να ήταν ακόμα καλύτερο για την αναγνώριση των DoS επιθέσεων.

Ένα ακόμα θετικό συμπέρασμα που προκύπτει είναι η αποδεδειγμένα καλή δυνατότητα γενίκευσης του Νευρωνικού Δικτύου. Αυτό αποδεικνύεται εύκολα από τη μεγάλη ποσοτική διαφορά training και testing set. Το training set περιείχε περίπου 14000 εγγραφές, ενώ το testing set περίπου 311000. Επίσης υπήρχαν εγγραφές στο testing set για τα οποία το Νευρωνικό δίκτυο δεν είχε εκπαιδευτεί καθόλου. Αυτό αποτελεί ένα πολύ ενθαρρυντικό αποτέλεσμα σε σχέση με τον αρχικό στόχο της εργασίας που ήταν ένας μηχανισμός ανάλυσης που να μπορεί να εντοπίζει ακόμα και

επιθέσεις για τις οποίες δεν έχει κάποια προηγούμενη γνώση. Το ποσοστό αναγνώρισης για αυτές τις επιθέσεις είναι μικρό σχετικά (19,23%) αλλά θα πρέπει να πούμε ότι ένα πολύ μεγάλο μέρος αυτών των επιθέσεων έχει stealth χαρακτηριστικά και γενικά είναι αρκετά δύσκολο να ανιχνευτούν ακόμα και από τα πιο προηγμένα συστήματα.

Επίσης, θα πρέπει να τονιστεί ότι το νευρωνικό δίκτυο, ενώ εκπαιδεύτηκε από τον αλγόριθμο PSO, δεν ήταν το βέλτιστο αρχιτεκτονικά. Άρα μπορούμε να θεωρήσουμε ότι υπάρχουν περιθώρια περαιτέρω βελτίωσης του αλγορίθμου.

Θετικό κρίνεται επίσης το γεγονός ότι τα false alarms γενικά ήταν σε χαμηλό επίπεδο: 0,61% στο σύνολο των δειγμάτων. Όπως έχει αναφερθεί, τα false alrms είναι ένα σημαντικό πρόβλημα που γενικά παρουσιάζεται στα συστήματα που έχουν την anomaly detection ως φιλοσοφία ανίχνευσης παρεισφρήσεων.

Στα αρνητικά συμπεράσματα μπορούμε να καταγράψουμε τον μεγάλο χρόνο εκπαίδευσης –περίπου 32 λεπτά για την εκπαίδευση- που όμως αποτελεί ένα γνωστό πρόβλημα στα Εξελικτικά Νευρωνικά Δίκτυα. Το πολύ μικρό ποσοστό ανίχνευσης για remote to local επιθέσεις είναι πρόβλημα, αλλά θα πρέπει να παρατηρήσουμε ότι δύο στις τρεις εγγραφές remote to local στο testing set αποτελούνταν από επιθέσεις για τις οποίες το νευρωνικό δίκτυο δεν είχε εκπαιδευτεί.

5.1 Προτάσεις για περαιτέρω μελέτη

Τα αποτελέσματα της εργασίας ανέδειξαν την αποτελεσματικότητα που μπορεί να έχουν τα Νευρωνικά Δίκτυα στο πεδίο της ασφάλειας πληροφοριακών συστημάτων

και ειδικότερα στην ανίχνευση παρεισφρήσεων. Παρόλο όμως που τα αποτελέσματα κρίνονται γενικά ως θετικά, υπάρχουν πολλά που μπορούν να γίνουν στην κατεύθυνση της επίτευξης του στόχου, που είναι να φτάσουμε στο 100% της αποτελεσματικότητας των IDS.

Ενδιαφέρον θα μπορούσε να έχει, πέρα από την εκπαίδευση, και η εύρεση του καταλληλότερου από αρχιτεκτονικής άποψης Νευρωνικού Δικτύου, μιας και όπως έχουμε πει ο αλγόριθμος PSO έχει αυτή τη δυνατότητα.. Η ταχύτητα εκπαίδευσης του Νευρωνικού Δικτύου θα μπορούσε να βελτιωθεί αρκετά με τη βοήθεια ενός παράλληλου αλγορίθμου PSO, όπου το κάθε Νευρωνικό Δίκτυο/Υποψήφια λύση εξελίσσεται παράλληλα. Επίσης ιδιαίτερο ενδιαφέρον θα είχε να εξεταστεί η εφαρμογή Εξελικτικών Νευρωνικών Δικτύων με ακέραια συνδετικά βάρη. Τα Νευρωνικά Δίκτυα με ακέραια βάρη έχουν το χαρακτηριστικό της καταλληλότητας για υλοποιήσεις Νευρωνικών Δικτύων σε hardware, υλοποιήσεις που είναι κατά πολύ γρηγορότερες και θα μπορούσαν να χρησιμοποιηθούν για ανιχνεύσεις σε πραγματικό χρόνο.

Τέλος, αποτελεσματικό θα μπορούσε να είναι ένα υβριδικό σύστημα ανίχνευσης με συνδυασμό Εξελικτικών Νευρωνικών δικτύων για network detection και κάποιας signature based τεχνικής για host detection.

ΒΙΒΛΙΟΓΡΑΦΙΚΕΣ ΑΝΑΦΟΡΕΣ

Attanasio C.R., Markstein and Phillips, Penetrating an Operating System: A Study ofVM/370 Integrity, IBM System Journal, vol. 15, l,pp. 102-116, 1976.

Bishop M., How Attackers Break Programs, and How To Write Programs More Securely, In Proceedings of the 8th USENIX Security Symposium, University of California, Davis, August 1999.

Brugger, S. T., Chow (January 2007). An assessment of the DARPA IDS Evaluation Dataset using Snort. Technical Report CSE-2007-1, University of California, Davis, Department of Computer Science, Davis, CA.

Burch H., Cheswick, Tracing Anonymous Packets to Their Approximate Source, In Proceedings of the USENIX Large Installation Systems Administration Conference, New Orleans, LA, 319-327, December 2000.

CERIAS Intrusion Detection Resources, http://www.cerias.purdue.edu/coast/ids/ids-body.html, 2004.

CERT® Advisory CA-1999-04 Melissa Worm and Macro Virus, http://www.cert.org/advisories/CA-1999-04.html, March 1999.

CERT® Advisory CA-2000-14 Microsoft Outlook and Outlook Express Cache

Bypass Vulnerability, http://www.cert.org/advisories/CA-2000-14.html, July 2000.

CERT® Advisory CA-2001-26 Nimda Worm, http://www.cert.org/advisories/CA-2001-26.html, September 2001.

CERT® Advisory CA-2003-04 MS-SQL Server Worm, http://www.cert.org/advisories/CA-2003-04.html, 2003.

CERT® Advisory CA-2003-25 Buffer Overflow in Sendmail, http://www.cert.org/advisories/CA-2003-25.html, September, 2003.

Cowan C., Pu, Maier, Hinton, Walpole, Bakke, Beattie, Grier and P. Zhang, StackGuard: Automatic Adaptive Detection and Prevention of Buffer-Overflow Attacks, In Proceedings of the 7th USENIX Security Symposium, San Antonio, TX, 63-77.

Debar H., Becker and Siboni, A Neural Network Component for an Intrusion-Detection System, In Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy, Oakland, CA, 240-250, May 1992.

Debar H., Dacier and Wespi, Towards a Taxonomy of Intrusion Detection Systems, Computer Networks, vol. 31,8, pp. 805-822, 1999.

Denning D., An Intrusion-Detection Model, IEEE Transactions on Software

Engineering, vol. 13, 2, pp. 222-232, 1987.

Gowadia G et al., 2005. PAID: A probabilistic agent-based intrusion detection system. Computers and Security. v24 i7. 529-545.

Howard J.D., An Analysis of Security Incidents on the Internet, Carnegie Mellon University, Pittsburgh, PA 15213 Ph.D. dissertation, April 1997.

http://www.snort.org

http://www.realsecure.com

Internet Guide, Computer Viruses / Virus Guide, http://www.internet-guide.co.uk/viruses.html, 2002.

Jain Anil K., Mao, Mohiuddin: Artificial Neural Networks: A Tutorial. IEEE Computer 29(3): 31-44 1996

Jung J., Paxson, Berger and Balakrishnan, Fast Portscan Detection Using Sequential Hypothesis Testing, In Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA, May, 2004.

Katsikas S.K., Spyrou, Gritzalis and Darzentas, "Model for Network Behaviour under Viral Attack", Computer Communications, Vol. 19, pp. 124-132, 1996.

Kendall K., A Database of Computer Attacks for the Evaluation of Intrusion

Detection Systems, Massachusetts Institute of Technology Master's Thesis, 1998.

Kennedy, J. and Eberhart, R. C. Particle swarm optimization. Proceedings of IEEE International Conference on Neural Networks, Piscataway, NJ. pp. 1942-1948, 1995

Kienzle D.,Elder, Recent Worms. A Survey and Trends, In Proceedings of the The Workshop on Rapid Malcode (WORM 2003), held in conjunction with the 10th ACM Conference on Computer and Communications Security, Washington, DC, October 27, 2003.

Krsul I.V., Software Vulnerability Analysis, Purdue University Ph.D. dissertation, May 1998.

Lazarevic, A., Ertoz, L., Ozgur, A, Srivastava, J., Kumar, V.: A Comparative Study of Anomaly Detection Schemes in Network Intrusion Detection, Proceedings of Third SIAM Conference on Data Mining, San Francisco, May. 2003.

Lough D., A Taxonomy of Computer Attacks with Applications to Wireless Networks, Virginia Polytechnic Institute PhD Thesis, April 2001.

Lundin B., Erland: Setting the scene for intrusion detection. Göteborg : Chalmers University of Technology. (Technical report - Chalmers University of Technology, Department of Computer Engineering, Göteborg;04-05)

Lunt T.F., Tamatu, Gilham, Jagannathan, Jalali, Javitz, Valdes and Neumann, A

Real-time Intrusion Detection Expert System, Technical Report SRI-CSL-90-05, Stanford Research Institute, 1990

Mahoney, M. V. and P. K. Chan. An analysis of the 1999 DARPA/Lincoln Laboratory Evaluation Data for network anomaly detection. In G. Vigna, E. Jonsson, and C. Krugel (Eds.), Proc. 6th Intl. Symp. on Recent Advances in Intrusion Detection (RAID 2003), Volume 2820 of Lecture Notes in Computer Science, Pittsburgh, PA, pp. 220-237. Springer.

Marchette D., Computer Intrusion Detection and Network Monitoring, A Statistical Viewpoint, New York, Springer, 2001.

McClelland, J.L, Rumelhart Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1,2: Psychological and Biological Models. Cambridge, Mass.: MIT Press 1986

McHugh J: Testing Intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory. ACM Trans. Inf. Syst. Secur. 3(4): 262-294 2000

Meier. M., Sobirey, Intrusion Detection Systems List and Bibliography, http://www-rnks.infoimatik.tu-cottbus.de/en/security/ids.html

Miller B.P., Koski, Lee, Maganty, Murthy, Natarajan, and Steidl, "Fuzz Revisited: A Re-examination of the Reliability of UNIX Utilities and Services", Computer

Sciences Technical Report #1268, University of Wisconsin-Madison, April 1995. Appears (in German translation) as "Empirische Studie zur Zuverlasskeit von UNIX-Utilities: Nichts dazu Gerlernt", iX, September 1995

Mirkovic J., Prier and Reiher, Attacking DDoS at the Source, 10th IEEE International Conference on Network Protocols, November 2002.

Mirkovic J., Reiher, A Taxonomy of DDoS Attacks and Defense Mechanisms, ACM Computer Communication Review, April 2004.

Parker D.B., Computer Abuse Perpetrators and Vulnerabilities of Computer Systems, Stanford Research Institute, Menlo Park, CA 94025 Technical Report, December 1975.

Powell D., Stroud, Conceptual Model and Architecture, Deliverable D2, Project MAFTIA 1ST-1999-11583, IBM Zurich Research Laboratory Research Report RZ 3377, Nov. 2001.

Provost F., Fawcett, Robust Classification for Imprecise Environments, Machine Learning, vol. 42, 3, pp. 203-231, 2001.

Race R., Mell, NIST Special Publication on Intrusion Detection Systems, 2001

Sebring M., Shellhouse, Hanna. and Whitehurst, Expert Systems in Intrusion

Detection: A Case Study, in Proc. 11th National Computer Security Conference, 1988

SecurityTechNet.com Intrusion Detection Links, http://cnscenter.future.co.kr/security/ids.html, 2004.

Spyrakis P., Katsikas, Gritzalis, Allegre, Darzentas, Gigante, Karagiannis, Putkonen and Spyrou, "SECURENET: A Network-Oriented Intelligent Intrusion, Prevention and Detection System, *10th IFIP International Information Security Conference*, 1994

ΠΑΡΑΡΤΗΜΑ Α –Ο κώδικας

```
package UIpackage;
import datapackage.Data;
import java.awt.BorderLayout;
import java.io.File;
import java.util.ArrayList;
import java.util.Iterator;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JList;
import javax.swing.LookAndFeel;
import javax.swing.SwingUtilities;
import javax.swing.UIManager;
import javax.swing.UnsupportedLookAndFeelException;
import nnpackage.NeuralNetwork;
import nnpackage.nnTesting;

import particlepackage.Pso;


public class MainForm extends javax.swing.JFrame {


        Data data;
        Pso pso;
        nnTesting testNeural;
        double ()() winningWeights;

        private Monitor mon;

        private Charts chart;

        public MainForm() {
                LookAndFeel lf = UIManager.getLookAndFeel();
```

```
            try {

UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndFeel");
            } catch (InstantiationException e) {
            } catch (ClassNotFoundException e) {
            } catch (UnsupportedLookAndFeelException e) {
            } catch (IllegalAccessException e) {
            }
            initComponents();
            this.setLocationRelativeTo(null);


        }


        // <editor-fold defaultstate="collapsed" desc=" Generated Code ">//GEN-BEGIN:initComponents
        private void initComponents() {
            jTabbedPane1 = new javax.swing.JTabbedPane();
            jPanel5 = new javax.swing.JPanel();
            jLabel2 = new javax.swing.JLabel();
            jTextField1 = new javax.swing.JTextField();
            jLabel3 = new javax.swing.JLabel();
            jTextField2 = new javax.swing.JTextField();
            jButton1 = new javax.swing.JButton();
            jSeparator1 = new javax.swing.JSeparator();
            jLabel4 = new javax.swing.JLabel();
            jLabel6 = new javax.swing.JLabel();
            jLabel7 = new javax.swing.JLabel();
            jButton2 = new javax.swing.JButton();
            jLabel8 = new javax.swing.JLabel();
            jLabel10 = new javax.swing.JLabel();
            jButton5 = new javax.swing.JButton();
            jLabel1 = new javax.swing.JLabel();
            jCheckBox1 = new javax.swing.JCheckBox();
            jLabel27 = new javax.swing.JLabel();
            jLabel28 = new javax.swing.JLabel();
```

```
jLabel29 = new javax.swing.JLabel();
jLabel30 = new javax.swing.JLabel();
jLabel31 = new javax.swing.JLabel();
jLabel32 = new javax.swing.JLabel();
jLabel33 = new javax.swing.JLabel();
jLabel34 = new javax.swing.JLabel();
jLabel35 = new javax.swing.JLabel();
jLabel36 = new javax.swing.JLabel();
jLabel37 = new javax.swing.JLabel();
jLabel38 = new javax.swing.JLabel();
jPanel6 = new javax.swing.JPanel();
jSlider1 = new javax.swing.JSlider();
jLabel11 = new javax.swing.JLabel();
jLabel12 = new javax.swing.JLabel();
jTextField3 = new javax.swing.JTextField();
jLabel13 = new javax.swing.JLabel();
jTextField4 = new javax.swing.JTextField();
jComboBox2 = new javax.swing.JComboBox();
jLabel14 = new javax.swing.JLabel();
jComboBox3 = new javax.swing.JComboBox();
jLabel15 = new javax.swing.JLabel();
jPanel7 = new javax.swing.JPanel();
jLabel16 = new javax.swing.JLabel();
jTextField5 = new javax.swing.JTextField();
jLabel17 = new javax.swing.JLabel();
jTextField6 = new javax.swing.JTextField();
jLabel18 = new javax.swing.JLabel();
jTextField7 = new javax.swing.JTextField();
jLabel19 = new javax.swing.JLabel();
jTextField8 = new javax.swing.JTextField();
jLabel20 = new javax.swing.JLabel();
jTextField9 = new javax.swing.JTextField();
jLabel21 = new javax.swing.JLabel();
jTextField10 = new javax.swing.JTextField();
jLabel22 = new javax.swing.JLabel();
```

```
jTextField11 = new javax.swing.JTextField();
jSeparator2 = new javax.swing.JSeparator();
jLabel24 = new javax.swing.JLabel();
jLabel25 = new javax.swing.JLabel();
jPanel3 = new javax.swing.JPanel();
jButton4 = new javax.swing.JButton();
jScrollPane1 = new javax.swing.JScrollPane();
jTextArea1 = new javax.swing.JTextArea();
jProgressBar1 = new javax.swing.JProgressBar();
jLabel5 = new javax.swing.JLabel();
jLabel9 = new javax.swing.JLabel();
jScrollPane2 = new javax.swing.JScrollPane();
jTextArea2 = new javax.swing.JTextArea();
jLabel23 = new javax.swing.JLabel();
jPanel4 = new javax.swing.JPanel();
jButton3 = new javax.swing.JButton();
jScrollPane3 = new javax.swing.JScrollPane();
jTextArea3 = new javax.swing.JTextArea();
jLabel26 = new javax.swing.JLabel();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
setTitle("Evolutionary Neural Network for IDS");

jPanel5.setBorder(javax.swing.BorderFactory.createTitledBorder(javax.swing.BorderFactory.createLi
neBorder(new java.awt.Color(0, 0, 0)), "1.Data Settings"));
jLabel2.setText("Input Dimension:");

jTextField1.setText("41");

jLabel3.setText("Target Dimension:");

jTextField2.setText("1");

jButton1.setText("Select Trainning Set");
jButton1.addMouseListener(new java.awt.event.MouseAdapter() {
        public void mouseClicked(java.awt.event.MouseEvent evt) {
```

```
                jButton1MouseClicked(evt);
            }
        });

        jLabel4.setText("Trainning patterns: ");

        jLabel6.setText("Testing File name:");

        jLabel7.setText("Trainning File name:");

        jButton2.setText("Create data");
        jButton2.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                jButton2MouseClicked(evt);
            }
        });

        jButton5.setText("Select Testing Set");
        jButton5.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                jButton5MouseClicked(evt);
            }
        });

        jCheckBox1.setSelected(true);
        jCheckBox1.setText("types");
        jCheckBox1.setBorder(javax.swing.BorderFactory.createEmptyBorder(0, 0, 0, 0));
        jCheckBox1.setMargin(new java.awt.Insets(0, 0, 0, 0));

        jLabel27.setText("Testing patterns:");

        jLabel29.setText("Normal:");

        jLabel30.setText("Probe:");
```

```
        jLabel31.setText("DoS:");

        jLabel32.setText("Remote to Local:");

        jLabel33.setText("User to Remote:");

        javax.swing.GroupLayout jPanel5Layout = new javax.swing.GroupLayout(jPanel5);
        jPanel5.setLayout(jPanel5Layout);
        jPanel5Layout.setHorizontalGroup(

jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(jPanel5Layout.createSequentialGroup()

.addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                        .addGroup(jPanel5Layout.createSequentialGroup()
                            .addGap(11, 11, 11)
                            .addComponent(jLabel2)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                            .addComponent(jTextField1,
javax.swing.GroupLayout.PREFERRED_SIZE, 28, javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addGap(31, 31, 31)
                            .addComponent(jLabel3)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                            .addComponent(jTextField2,
javax.swing.GroupLayout.PREFERRED_SIZE, 23, javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGroup(jPanel5Layout.createSequentialGroup()
                            .addContainerGap()
                            .addComponent(jSeparator1,
javax.swing.GroupLayout.PREFERRED_SIZE, 306, javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGroup(jPanel5Layout.createSequentialGroup()
                            .addContainerGap()
                            .addComponent(jLabel4)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                            .addComponent(jLabel10,
javax.swing.GroupLayout.PREFERRED_SIZE, 102, javax.swing.GroupLayout.PREFERRED_SIZE))
```

```
                .addGroup(jPanel5Layout.createSequentialGroup()
                    .addContainerGap()

.addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)

.addGroup(jPanel5Layout.createSequentialGroup()
                                .addComponent(jLabel7)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                .addComponent(jLabel8,
javax.swing.GroupLayout.PREFERRED_SIZE, 192, javax.swing.GroupLayout.PREFERRED_SIZE))

.addGroup(jPanel5Layout.createSequentialGroup()
                                .addComponent(jLabel6)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                                .addComponent(jLabel1,
javax.swing.GroupLayout.PREFERRED_SIZE, 132, javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addGap(16, 16, 16)
                                .addComponent(jCheckBox1)))
                    .addGap(28, 28, 28)

.addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                        .addComponent(jButton5)
                        .addComponent(jButton1)))
                .addGroup(jPanel5Layout.createSequentialGroup()
                    .addContainerGap()
                    .addComponent(jLabel27)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addComponent(jLabel28,
javax.swing.GroupLayout.PREFERRED_SIZE, 63, javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addContainerGap(303, Short.MAX_VALUE))
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel5Layout.createSequentialGroup()
                .addContainerGap(347, Short.MAX_VALUE)
                .addComponent(jButton2)
                .addGap(324, 324, 324))
```

```
                .addGroup(jPanel5Layout.createSequentialGroup()
                    .addContainerGap()

.addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                        .addGroup(javax.swing.GroupLayout.Alignment.LEADING,
jPanel5Layout.createSequentialGroup()
                            .addComponent(jLabel33)
                            .addGap(22, 22, 22)
                            .addComponent(jLabel38,
javax.swing.GroupLayout.DEFAULT_SIZE, 67, Short.MAX_VALUE))
                        .addGroup(javax.swing.GroupLayout.Alignment.LEADING,
jPanel5Layout.createSequentialGroup()

.addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                                .addComponent(jLabel29)
                                .addComponent(jLabel30)
                                .addComponent(jLabel31)
                                .addComponent(jLabel32))
                            .addGap(20, 20, 20)

.addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
                                .addComponent(jLabel37,
javax.swing.GroupLayout.DEFAULT_SIZE,       javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                                .addComponent(jLabel36,
javax.swing.GroupLayout.DEFAULT_SIZE,       javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                                .addComponent(jLabel35,
javax.swing.GroupLayout.DEFAULT_SIZE,       javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                                .addComponent(jLabel34,
javax.swing.GroupLayout.DEFAULT_SIZE, 67, Short.MAX_VALUE))))
                    .addContainerGap(584, javax.swing.GroupLayout.PREFERRED_SIZE))
            );
            jPanel5Layout.setVerticalGroup(

jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(jPanel5Layout.createSequentialGroup()
                    .addGap(28, 28, 28)
```

```
.addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jTextField1,
javax.swing.GroupLayout.PREFERRED_SIZE,        javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jLabel2)
                .addComponent(jLabel3)
                .addComponent(jTextField2,
javax.swing.GroupLayout.PREFERRED_SIZE,        javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(17, 17, 17)
            .addComponent(jSeparator1,
javax.swing.GroupLayout.PREFERRED_SIZE, 10, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(18, 18, 18)

.addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jLabel7)
                .addComponent(jLabel8,
javax.swing.GroupLayout.PREFERRED_SIZE, 14, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jButton1))
            .addGap(16, 16, 16)

.addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jLabel4)
                .addComponent(jLabel10,
javax.swing.GroupLayout.PREFERRED_SIZE, 14, javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(18, 18, 18)

.addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jLabel6)
                .addComponent(jLabel1)
                .addComponent(jCheckBox1)
                .addComponent(jButton5))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jLabel27)
                .addComponent(jLabel28,
javax.swing.GroupLayout.PREFERRED_SIZE, 14, javax.swing.GroupLayout.PREFERRED_SIZE))
```

```
                .addGap(43, 43, 43)

.addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jLabel29)
                .addComponent(jLabel34))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jLabel30)
                .addComponent(jLabel35))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jLabel31)
                .addComponent(jLabel36))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jLabel32)
                .addComponent(jLabel37))

.addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(jPanel5Layout.createSequentialGroup()
                        .addGap(23, 23, 23)
                        .addComponent(jButton2))
                .addGroup(jPanel5Layout.createSequentialGroup()

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                        .addComponent(jLabel33)
                        .addComponent(jLabel38))))
                .addContainerGap(38, Short.MAX_VALUE))
        );
        jTabbedPane1.addTab("Data", jPanel5);
```

```
jPanel6.setBorder(javax.swing.BorderFactory.createTitledBorder(javax.swing.BorderFactory.createLi
neBorder(new java.awt.Color(0, 0, 0)), "2. Neural Network Parameters"));
                jSlider1.setMajorTickSpacing(1);
                jSlider1.setMaximum(2);
                jSlider1.setMinimum(1);
                jSlider1.setMinorTickSpacing(1);
                jSlider1.setPaintLabels(true);
                jSlider1.setPaintTicks(true);
                jSlider1.setSnapToTicks(true);
                jSlider1.addMouseListener(new java.awt.event.MouseAdapter() {
                        public void mouseClicked(java.awt.event.MouseEvent evt) {
                                jSlider1MouseClicked(evt);
                        }
                });
                jSlider1.addChangeListener(new javax.swing.event.ChangeListener() {
                        public void stateChanged(javax.swing.event.ChangeEvent evt) {
                                jSlider1StateChanged(evt);
                        }
                });

                jLabel11.setText("Hidden Layers:");

                jLabel12.setText("Layer 1 Neurons:");

                jTextField3.setText("10");

                jLabel13.setText("Layer 2 Neurons:");

                jTextField4.setText("3");

                jComboBox2.setModel(new javax.swing.DefaultComboBoxModel(new String() {
"Integer weights", "Real weights" }));

                jLabel14.setText("Type of weights:");
```

```
jComboBox3.setModel(new javax.swing.DefaultComboBoxModel(new String() { "Logistic", "Linear", "Threshold" }));

jLabel15.setText("Type of activation Function:");

javax.swing.GroupLayout jPanel6Layout = new javax.swing.GroupLayout(jPanel6);
jPanel6.setLayout(jPanel6Layout);
jPanel6Layout.setHorizontalGroup(

jPanel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel6Layout.createSequentialGroup()
            .addContainerGap()

.addGroup(jPanel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jLabel11)
                .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel6Layout.createSequentialGroup()

.addGroup(jPanel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                        .addComponent(jSlider1, javax.swing.GroupLayout.PREFERRED_SIZE, 158, javax.swing.GroupLayout.PREFERRED_SIZE)

.addGroup(jPanel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(jLabel14)
                            .addComponent(jComboBox2, javax.swing.GroupLayout.PREFERRED_SIZE, 139, javax.swing.GroupLayout.PREFERRED_SIZE)))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 71, Short.MAX_VALUE)

.addGroup(jPanel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel6Layout.createSequentialGroup()
                            .addComponent(jLabel12)
                            .addGap(21, 21, 21)
                            .addComponent(jTextField3, javax.swing.GroupLayout.PREFERRED_SIZE, 28, javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addGap(74, 74, 74)
                            .addComponent(jLabel13)
```

```
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                                .addComponent(jTextField4,
javax.swing.GroupLayout.PREFERRED_SIZE, 28, javax.swing.GroupLayout.PREFERRED_SIZE)
                                                .addGap(198, 198, 198))

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel6Layout.createSequentialGroup()

.addGroup(jPanel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                                                        .addComponent(jLabel15)
                                                        .addComponent(jComboBox3,
javax.swing.GroupLayout.PREFERRED_SIZE, 163, javax.swing.GroupLayout.PREFERRED_SIZE))
                                                .addGap(360, 360, 360))))))

            );
            jPanel6Layout.setVerticalGroup(

jPanel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel6Layout.createSequentialGroup()
                        .addGap(74, 74, 74)
                        .addComponent(jLabel11)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(jPanel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)

.addGroup(jPanel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                                        .addComponent(jTextField3,
javax.swing.GroupLayout.PREFERRED_SIZE,     javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                        .addComponent(jLabel13)
                                        .addComponent(jTextField4,
javax.swing.GroupLayout.PREFERRED_SIZE,     javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                    .addComponent(jLabel12))
                                .addComponent(jSlider1,
javax.swing.GroupLayout.PREFERRED_SIZE,     javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGap(26, 26, 26)
```

```
.addGroup(jPanel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                                        .addComponent(jLabel14)
                                        .addComponent(jLabel15))
                                    .addGap(4, 4, 4)

.addGroup(jPanel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                                        .addComponent(jComboBox2,
javax.swing.GroupLayout.PREFERRED_SIZE, 20, javax.swing.GroupLayout.PREFERRED_SIZE)
                                        .addComponent(jComboBox3,
javax.swing.GroupLayout.PREFERRED_SIZE, 20, javax.swing.GroupLayout.PREFERRED_SIZE))
                                    .addContainerGap(203, Short.MAX_VALUE))
                );
                jTabbedPane1.addTab("Neural Network parameters", jPanel6);


jPanel7.setBorder(javax.swing.BorderFactory.createTitledBorder(javax.swing.BorderFactory.createLi
neBorder(new java.awt.Color(0, 0, 0)), "3. Particle Swarm Optimization"));
                jLabel16.setText("Particles:");

                jTextField5.setText("20");

                jLabel17.setText("Max Iterations:");

                jTextField6.setText("600");

                jLabel18.setText("Inertia:");

                jTextField7.setText("0.7");

                jLabel19.setText("Cognitive Parameter:");

                jTextField8.setText("1.5");

                jLabel20.setText("Social Parameter:");

                jTextField9.setText("1.5");
```

```
jLabel21.setText("Max Velocity:");

jTextField10.setText("0.5");

jLabel22.setText("Min Velocity:");

jTextField11.setText("-0.5");

jLabel24.setText("particle parameters:");

jLabel25.setText("Algorithm parameters:");

javax.swing.GroupLayout jPanel7Layout = new javax.swing.GroupLayout(jPanel7);
jPanel7.setLayout(jPanel7Layout);
jPanel7Layout.setHorizontalGroup(

jPanel7Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(jPanel7Layout.createSequentialGroup()

.addGroup(jPanel7Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                                .addGroup(jPanel7Layout.createSequentialGroup()
                                        .addGap(73, 73, 73)

.addGroup(jPanel7Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                                                .addComponent(jLabel18)
                                                .addComponent(jLabel24)
                                                .addComponent(jLabel21))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(jPanel7Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
                                                .addComponent(jTextField10,
javax.swing.GroupLayout.DEFAULT_SIZE, 39, Short.MAX_VALUE)
                                                .addComponent(jTextField7))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
```

```
.addGroup(jPanel7Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                                        .addComponent(jLabel19)
                                        .addComponent(jLabel22))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(jPanel7Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel7Layout.createSequentialGroup()
                                        .addComponent(jTextField8,
javax.swing.GroupLayout.PREFERRED_SIZE, 45, javax.swing.GroupLayout.PREFERRED_SIZE)
                                        .addGap(34, 34, 34)
                                        .addComponent(jLabel20)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                        .addComponent(jTextField9,
javax.swing.GroupLayout.PREFERRED_SIZE, 39, javax.swing.GroupLayout.PREFERRED_SIZE))
                                    .addComponent(jTextField11,
javax.swing.GroupLayout.PREFERRED_SIZE, 57, javax.swing.GroupLayout.PREFERRED_SIZE)))
                                .addGroup(jPanel7Layout.createSequentialGroup()
                                    .addGap(67, 67, 67)

.addGroup(jPanel7Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                                        .addComponent(jLabel25)

.addGroup(jPanel7Layout.createSequentialGroup()
                                        .addComponent(jLabel16)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                        .addComponent(jTextField5,
javax.swing.GroupLayout.PREFERRED_SIZE, 31, javax.swing.GroupLayout.PREFERRED_SIZE)))
                                    .addGap(53, 53, 53)
                                    .addComponent(jLabel17)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                    .addComponent(jTextField6,
javax.swing.GroupLayout.PREFERRED_SIZE, 41, javax.swing.GroupLayout.PREFERRED_SIZE))
                                .addGroup(jPanel7Layout.createSequentialGroup()
                                    .addGap(59, 59, 59)
```

```
                                        .addComponent(jSeparator2,
javax.swing.GroupLayout.PREFERRED_SIZE,                                 551,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                                .addContainerGap(152, Short.MAX_VALUE))
            );
            jPanel7Layout.setVerticalGroup(

jPanel7Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(jPanel7Layout.createSequentialGroup()
                    .addGap(30, 30, 30)
                    .addComponent(jLabel24)
                    .addGap(24, 24, 24)

.addGroup(jPanel7Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                        .addComponent(jTextField7,
javax.swing.GroupLayout.PREFERRED_SIZE,     javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(jLabel18)
                        .addComponent(jTextField8,
javax.swing.GroupLayout.PREFERRED_SIZE,     javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(jLabel19)
                        .addComponent(jLabel20)
                        .addComponent(jTextField9,
javax.swing.GroupLayout.PREFERRED_SIZE,     javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(jPanel7Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                        .addComponent(jTextField10,
javax.swing.GroupLayout.PREFERRED_SIZE,     javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(jLabel22)
                        .addComponent(jLabel21)
                        .addComponent(jTextField11,
javax.swing.GroupLayout.PREFERRED_SIZE,     javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGap(63, 63, 63)
                    .addComponent(jSeparator2,
javax.swing.GroupLayout.PREFERRED_SIZE, 10, javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
                .addGap(42, 42, 42)
                .addComponent(jLabel25)
                .addGap(15, 15, 15)

.addGroup(jPanel7Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jTextField5,
javax.swing.GroupLayout.PREFERRED_SIZE,        javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jLabel16)
                .addComponent(jLabel17)
                .addComponent(jTextField6,
javax.swing.GroupLayout.PREFERRED_SIZE,        javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addContainerGap(130, Short.MAX_VALUE))
        );
        jTabbedPane1.addTab("PSO parameters", jPanel7);

        jButton4.setText("Start trainning");
        jButton4.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                jButton4MouseClicked(evt);
            }
        });

        jTextArea1.setColumns(20);
        jTextArea1.setFont(new java.awt.Font("Arial", 0, 12));
        jTextArea1.setRows(5);
        jScrollPane1.setViewportView(jTextArea1);

        jProgressBar1.setMaximum(1000);

        jLabel5.setText("Fitness:");

        jLabel9.setText("Progress:");

        jTextArea2.setColumns(20);
        jTextArea2.setRows(5);
```

```
jScrollPane2.setViewportView(jTextArea2);

jLabel23.setText("winning weights:");

javax.swing.GroupLayout jPanel3Layout = new javax.swing.GroupLayout(jPanel3);
jPanel3.setLayout(jPanel3Layout);
jPanel3Layout.setHorizontalGroup(

jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(jPanel3Layout.createSequentialGroup()

.addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                        .addGroup(jPanel3Layout.createSequentialGroup()
                            .addGap(308, 308, 308)
                            .addComponent(jButton4))
                        .addGroup(jPanel3Layout.createSequentialGroup()
                            .addGap(27, 27, 27)

.addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(jLabel5)
                            .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 267, javax.swing.GroupLayout.PREFERRED_SIZE))

.addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel3Layout.createSequentialGroup()
                            .addGap(158, 158, 158)

.addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(jLabel9)
                            .addComponent(jProgressBar1,
javax.swing.GroupLayout.PREFERRED_SIZE,                                 218,
javax.swing.GroupLayout.PREFERRED_SIZE)))

.addGroup(jPanel3Layout.createSequentialGroup()
                            .addGap(40, 40, 40)

.addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(jLabel23)
```

```
                                                .addComponent(jScrollPane2,
javax.swing.GroupLayout.DEFAULT_SIZE, 336, Short.MAX_VALUE))))))
                                .addContainerGap(102, Short.MAX_VALUE))
        );
        jPanel3Layout.setVerticalGroup(

jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(jPanel3Layout.createSequentialGroup()

.addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
                                .addGroup(jPanel3Layout.createSequentialGroup()
                                        .addGap(53, 53, 53)
                                        .addComponent(jLabel5)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                        .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 251, javax.swing.GroupLayout.PREFERRED_SIZE))
                                .addGroup(jPanel3Layout.createSequentialGroup()
                                        .addGap(43, 43, 43)
                                        .addComponent(jLabel9)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                        .addComponent(jProgressBar1,
javax.swing.GroupLayout.PREFERRED_SIZE,        javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                        .addGap(16, 16, 16)
                                        .addComponent(jLabel23)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                        .addComponent(jScrollPane2)))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,                60,
Short.MAX_VALUE)
                        .addComponent(jButton4)
                        .addGap(25, 25, 25))
        );
        jTabbedPane1.addTab("Trainning", jPanel3);
```

```
        jButton3.setText("Evaluate");
        jButton3.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                jButton3MouseClicked(evt);
            }
        });

        jTextArea3.setColumns(20);
        jTextArea3.setFont(new java.awt.Font("Arial", 0, 12));
        jTextArea3.setRows(5);
        jScrollPane3.setViewportView(jTextArea3);

        jLabel26.setText("results:");

        javax.swing.GroupLayout jPanel4Layout = new javax.swing.GroupLayout(jPanel4);
        jPanel4.setLayout(jPanel4Layout);
        jPanel4Layout.setHorizontalGroup(

jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel4Layout.createSequentialGroup()

.addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel4Layout.createSequentialGroup()
                        .addGap(297, 297, 297)
                        .addComponent(jButton3))
                    .addGroup(jPanel4Layout.createSequentialGroup()
                        .addGap(28, 28, 28)

.addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(jLabel26)
                            .addComponent(jScrollPane3,
javax.swing.GroupLayout.PREFERRED_SIZE,                    469,
javax.swing.GroupLayout.PREFERRED_SIZE))))
            .addContainerGap(275, Short.MAX_VALUE))
        );
        jPanel4Layout.setVerticalGroup(
```

```
jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel4Layout.createSequentialGroup()
                        .addGap(26, 26, 26)
                        .addComponent(jLabel26)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(jScrollPane3,
javax.swing.GroupLayout.PREFERRED_SIZE, 244, javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 64,
Short.MAX_VALUE)
                        .addComponent(jButton3)
                        .addGap(55, 55, 55))
            );
            jTabbedPane1.addTab("Evaluation", jPanel4);

            javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
            getContentPane().setLayout(layout);
            layout.setHorizontalGroup(
                layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jTabbedPane1, javax.swing.GroupLayout.DEFAULT_SIZE,
777, Short.MAX_VALUE)
            );
            layout.setVerticalGroup(
                layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jTabbedPane1, javax.swing.GroupLayout.DEFAULT_SIZE,
460, Short.MAX_VALUE)
            );
            pack();
        }// </editor-fold>//GEN-END:initComponents

    private void jButton5MouseClicked(java.awt.event.MouseEvent evt) {//GEN-
FIRST:event_jButton5MouseClicked

            openTestingSet();
```

```
    }//GEN-LAST:event_jButton5MouseClicked

    private void jButton3MouseClicked(java.awt.event.MouseEvent evt) {//GEN-FIRST:event_jButton3MouseClicked

        startEvaluation();
        jTextArea3.append("Right classify: "+100*(double)(nnTesting.rightNormal+nnTesting.rightAttack)/nnTesting.totalData+" %\n");
        jTextArea3.append("Misclasified: "+100*(double)(nnTesting.missClassified)/nnTesting.totalData+" %\n");
        jTextArea3.append("=====================================\n");
        jTextArea3.append("Normal: "+100*(double)nnTesting.rightNormal/data.normal+" %\n");
        jTextArea3.append("Probe: "+100*(double)nnTesting.rightProbe/data.probe+" %\n");
        jTextArea3.append("DoS: "+100*(double)nnTesting.rightDos/data.dos+" %\n");
        jTextArea3.append("Remote to Local : "+100*(double)nnTesting.rightR2L/data.r2l+" %\n");
        jTextArea3.append("User to Remote: "+100*(double)nnTesting.rightU2R/data.u2r+" %\n");
        jTextArea3.append("=====================================\n");
        jTextArea3.append("False alarms: "+100*(double)nnTesting.falseAlarms/nnTesting.totalData+" %\n");

    }//GEN-LAST:event_jButton3MouseClicked

    private void jButton4MouseClicked(java.awt.event.MouseEvent evt) {//GEN-FIRST:event_jButton4MouseClicked

        Parameters.activationFunctionSelection=jComboBox3.getSelectedIndex();
        Parameters.inputDimension=Integer.parseInt((jTextField1.getText()));
        Parameters.targetDimension=Integer.parseInt(jTextField2.getText());
        Parameters.hiddenLayersCounter=jSlider1.getValue();
        Parameters.hiddenLayer1Nodes=Integer.parseInt(jTextField3.getText());
        // if(Parameters.hiddenLayersCounter==2){
        Parameters.hiddenLayer2Nodes=Integer.parseInt(jTextField4.getText());
        //    }
        //else Parameters.hiddenLayersCounter=0;
        Parameters.inertia=Double.parseDouble(jTextField7.getText());
```

```
            Parameters.cognitiveParameter=Double.parseDouble(jTextField8.getText());
            Parameters.socialParameter=Double.parseDouble(jTextField9.getText());
            Parameters.maxVelocity=Double.parseDouble(jTextField10.getText());
            Parameters.minVelocity=Double.parseDouble(jTextField11.getText());
            Parameters.particles=Integer.parseInt(jTextField5.getText());
            Parameters.maxIterations=Integer.parseInt(jTextField6.getText());



            winningWeights=new double(Parameters.hiddenLayersCounter+1)();


            winningWeights(0)=new
double(Parameters.inputDimension*Parameters.hiddenLayer1Nodes);
            winningWeights(1)=new
double(Parameters.hiddenLayer1Nodes*Parameters.hiddenLayer2Nodes);
            winningWeights(2)=new
double(Parameters.hiddenLayer2Nodes*Parameters.targetDimension);


            startTrainning();


    }//GEN-LAST:event_jButton4MouseClicked


    private void jSlider1StateChanged(javax.swing.event.ChangeEvent evt) {//GEN-
FIRST:event_jSlider1StateChanged


    }//GEN-LAST:event_jSlider1StateChanged


    private void jSlider1MouseClicked(java.awt.event.MouseEvent evt) {//GEN-
FIRST:event_jSlider1MouseClicked
// TODO add your handling code here:
            if(jSlider1.getValue()==1){
                jTextField4.setVisible(false);
            } else jTextField4.setVisible(true);
    }//GEN-LAST:event_jSlider1MouseClicked


    private void jButton2MouseClicked(java.awt.event.MouseEvent evt) {//GEN-
FIRST:event_jButton2MouseClicked
```

```
data=new Data();
data.setFilename("data/"+jLabel8.getText());

if(jCheckBox1.isSelected()){
        Parameters.types=1;
} else {
        Parameters.types=0;
}
data.setInputDimension(Integer.parseInt(jTextField1.getText()));
data.setTargetDimension(Integer.parseInt(jTextField2.getText()));
data.createTrainningSet();
data.setTestingSetFile("data/"+jLabel1.getText());
data.createTestingSet();
jLabel10.setText(String.valueOf(data.getPatterns()));
jLabel28.setText(""+data.getTestingPatterns());



jLabel34.setText(""+data.normal);
jLabel35.setText(""+data.probe);
jLabel36.setText(""+data.dos);
jLabel37.setText(""+data.r2l);
jLabel38.setText(""+data.u2r);




    }//GEN-LAST:event_jButton2MouseClicked

    private void jButton1MouseClicked(java.awt.event.MouseEvent evt) {//GEN-
FIRST:event_jButton1MouseClicked
        openDataset();
    }//GEN-LAST:event_jButton1MouseClicked


    public static void main(String args()) {
```

```
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new MainForm().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JButton jButton3;
    private javax.swing.JButton jButton4;
    private javax.swing.JButton jButton5;
    private javax.swing.JCheckBox jCheckBox1;
    private javax.swing.JComboBox jComboBox2;
    private javax.swing.JComboBox jComboBox3;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel10;
    private javax.swing.JLabel jLabel11;
    private javax.swing.JLabel jLabel12;
    private javax.swing.JLabel jLabel13;
    private javax.swing.JLabel jLabel14;
    private javax.swing.JLabel jLabel15;
    private javax.swing.JLabel jLabel16;
    private javax.swing.JLabel jLabel17;
    private javax.swing.JLabel jLabel18;
    private javax.swing.JLabel jLabel19;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel20;
    private javax.swing.JLabel jLabel21;
    private javax.swing.JLabel jLabel22;
    private javax.swing.JLabel jLabel23;
    private javax.swing.JLabel jLabel24;
    private javax.swing.JLabel jLabel25;
    private javax.swing.JLabel jLabel26;
```

```
private javax.swing.JLabel jLabel27;
private javax.swing.JLabel jLabel28;
private javax.swing.JLabel jLabel29;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel30;
private javax.swing.JLabel jLabel31;
private javax.swing.JLabel jLabel32;
private javax.swing.JLabel jLabel33;
private javax.swing.JLabel jLabel34;
private javax.swing.JLabel jLabel35;
private javax.swing.JLabel jLabel36;
private javax.swing.JLabel jLabel37;
private javax.swing.JLabel jLabel38;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
private javax.swing.JLabel jLabel6;
private javax.swing.JLabel jLabel7;
private javax.swing.JLabel jLabel8;
private javax.swing.JLabel jLabel9;
private javax.swing.JPanel jPanel3;
private javax.swing.JPanel jPanel4;
private javax.swing.JPanel jPanel5;
private javax.swing.JPanel jPanel6;
private javax.swing.JPanel jPanel7;
private javax.swing.JProgressBar jProgressBar1;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JScrollPane jScrollPane2;
private javax.swing.JScrollPane jScrollPane3;
private javax.swing.JSeparator jSeparator1;
private javax.swing.JSeparator jSeparator2;
private javax.swing.JSlider jSlider1;
private javax.swing.JTabbedPane jTabbedPane1;
private javax.swing.JTextArea jTextArea1;
private javax.swing.JTextArea jTextArea2;
private javax.swing.JTextArea jTextArea3;
```

```
 private javax.swing.JTextField jTextField1;
 private javax.swing.JTextField jTextField10;
 private javax.swing.JTextField jTextField11;
 private javax.swing.JTextField jTextField2;
 private javax.swing.JTextField jTextField3;
 private javax.swing.JTextField jTextField4;
 private javax.swing.JTextField jTextField5;
 private javax.swing.JTextField jTextField6;
 private javax.swing.JTextField jTextField7;
 private javax.swing.JTextField jTextField8;
 private javax.swing.JTextField jTextField9;
 // End of variables declaration//GEN-END:variables
private JFrame frame;

private JList list;
private JList list2;

public String openDataset() {
        String filename = null;

        String title = "DataSet File Chooser";
        frame = new JFrame(title);
        frame.setLocationRelativeTo(null);
        File dir=new File("data");
        String () result=dir.list();
        list=new JList();
        JButton ok=new JButton("select");

        ok.addMouseListener(new java.awt.event.MouseAdapter() {
                public void mouseClicked(java.awt.event.MouseEvent evt) {
                        okMouseClicked(evt);
                }
        });

        list.setListData(result);
```

```
        frame.getContentPane().add(list, BorderLayout.CENTER);
        frame.getContentPane().add(ok,BorderLayout.SOUTH);
        int width = 300;
        int height = 300;
        frame.setSize(width, height);
        frame.setVisible(true);
        return  filename;
}

public String openTestingSet() {
        String filename = null;

        String title = "TestingSet File Chooser";
        frame = new JFrame(title);
        frame.setLocationRelativeTo(null);
        File dir=new File("data");
        String () result=dir.list();
        list2=new JList();
        JButton ok1=new JButton("select");

        ok1.addMouseListener(new java.awt.event.MouseAdapter() {
                public void mouseClicked(java.awt.event.MouseEvent evt) {
                        ok1MouseClicked(evt);
                }
        });

        list2.setListData(result);
        frame.getContentPane().add(list2, BorderLayout.CENTER);
        frame.getContentPane().add(ok1,BorderLayout.SOUTH);
        int width = 300;
        int height = 300;
        frame.setSize(width, height);
        frame.setVisible(true);
        return  filename;
}
```

```
/**
 *το event για το κουμπί της φόρμας επιλογής αρχείου
 */
private void okMouseClicked(java.awt.event.MouseEvent evt){
        String selection=(String)list.getSelectedValue();
        jLabel8.setText(selection);
        frame.dispose();
}

private void ok1MouseClicked(java.awt.event.MouseEvent evt){
        String selection=(String)list2.getSelectedValue();
        jLabel1.setText(selection);
        frame.dispose();
}

private void startTrainning() {

        mon=new Monitor();
        pso=new Pso(data,mon);

        pso.runAlgorithm();

        winningWeights=pso.getGlobalBest();

        for (int i = 0; i < pso.getFitnessStore().length; i++) {
                jTextArea1.append(" "+pso.getFitnessStore()(i)+"\n");
        }

        for (int i = 0; i < winningWeights.length; i++) {
                for (int j = 0; j < winningWeights(i).length; j++) {
                        jTextArea2.append("   "+winningWeights(i)(j));
                }
                jTextArea2.append("\n");
        }
```

```
                Charts1 chart=new Charts1(pso.getFitnessStore(),"FITNESS CHART");
                chart.pack();
                chart.setPreferredSize(new java.awt.Dimension(600,800));
                chart.setVisible(true);

        }

        private void startEvaluation() {

                pso.getGlobalBest();
                testNeural=new nnTesting();
                testNeural.constructNN();
                testNeural.setTestingSet(data.getTestingSet());

                double temp=testNeural.runNN(winningWeights);
        }
}


package UIpackage;

import java.util.ArrayList;


public class Parameters {

        public static String filename;   //όνομα αρχείου δεδομένων
        public static int inputDimension; //οι στήλες των δεδομένων εισόδου
        public static int targetDimension;//οι στήλες των δεδομένων στόχου
        public static int patterns;//ο συνολικός αριθμός των προτύπων
        public static double()() trainningSet;  //ο πίνακας με τα δεδομένα για εκπαίδευση
        public static double()() testingSet; //ο πίνακας με τα δεδομένα για τεστ
        public static int wholeInput; // ο συνολικές στήλες του αρχείου δεδομένων
        public static int methodOfValidation; //επιλογή του τρόπου που θα γίνε η επιλογή των δεδομένων
```

```
        public static int hiddenLayersCounter;
        public static int hiddenLayer1Nodes;
        public static int hiddenLayer2Nodes;
        public static int activationFunctionSelection;

        public static double cognitiveParameter;
        public static double socialParameter;
        public static double inertia;
        public static double maxVelocity;
        public static double minVelocity;

        public static int particles;
        public static int maxIterations;
        public static int types;

        public Parameters() {
        }

}



package UIpackage;
import javax.swing.JFrame;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.xy.XYDataset;
import org.jfree.data.xy.XYSeries;
import org.jfree.data.xy.XYSeriesCollection;

/**
 * η κλάση αυτή υλοποιεί την δημιουργία γραφήματος για την αναπαράσταση
 * του fitness του αλγορίθμου
```

```
*/

public class Charts1 extends JFrame{


        public Charts1(double() d, String title) {
                super(title);
                XYDataset dataset=createDataset(d);
                JFreeChart chart=createChart(dataset);
                ChartPanel chartpanel=new ChartPanel(chart);
                chartpanel.setPreferredSize(new java.awt.Dimension(600,800));
                setContentPane(chartpanel);

        }

        private XYDataset createDataset(double() d) {
                XYSeries series=new XYSeries("fitness");
                for (int i = 0; i < d.length; i++) {
                        series.add(i,d(i));
                }
                XYSeriesCollection dataset=new XYSeriesCollection();
                dataset.addSeries(series);
                return dataset;
        }

        private JFreeChart createChart(XYDataset dataset) {
                JFreeChart chart=ChartFactory.createXYLineChart(
                                "fitness graph",
                                "iterations",
                                "Fitness",
                                dataset,
                                PlotOrientation.VERTICAL,
                                true,
                                true,
                                false
```

```
                                );
                return chart;
        }

}
```

```
package datapackage;

import UIpackage.Parameters;
import java.awt.BorderLayout;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.Random;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JList;
import javax.swing.JOptionPane;

/**
 * η κλάση αυτή υλοποιεί τον χειρισμό των δεδομένων. Διαβάζει το training set και το testing set στη μνήμη
 * σε πίνακες και εκει τα διαμορφώνει έτσι ώστε να γίνουν κατάλληλα για είδοδο στο Νευρωνικό δίκτυο.
 */
public class Data {

        private String trainningSetFile; //όνομα αρχείου δεδομένων
```

```
private String testingSetFile;

private int inputDimension; //οι στήλες των δεδομένων εισόδου
private int targetDimension;//οι στήλες των δεδομένων στόχου
private int patterns;//ο συνολικός αριθμός των προτύπων

private int testingPatterns;

private double()() trainningSet;  //ο πίνακας με τα δεδομένα για εκπαίδευση
private double()() testingSet; //ο πίνακας με τα δεδομένα για τεστ

private int input; // ο συνολικές στήλες του αρχείου δεδομένων

public int normal;
public int probe;
public int dos;
public int r2l;
public int u2r;

/**
 * O constructor
 */
public Data() {
}



/**
 * Η μέθοδος αυτή διαβάζει τα δεδομένα απο το αρχείο που έχει υποδείξει ο χρήστης στη MainForm και δημιουργεί ένα πίνακα δύο διαστάσεων
 * το training set. Τα δεδομένα κανονικοποιούνται στο διάστημα (0,1).
 */

public void createTrainningSet() {

        BufferedReader reader = null;
        String line;
```

```
int lineCounter=0;
int amountOfTestPatterns;
ArrayList <String> dataBuffer=new ArrayList<String>();

setInput(getInputDimension() + getTargetDimension());

try {
        reader = new BufferedReader(new FileReader(getTrainningSetFile()));
} catch (FileNotFoundException ex) {
        new JOptionPane().showMessageDialog(null,"Δεν βρέθηκε το αρχείο");
}
try {
        while((line=reader.readLine())!=null){

                dataBuffer.add(line);
                lineCounter++;
        }
} catch (IOException ex) {
        new JOptionPane().showMessageDialog(null,"Πρόβλημα ανάγνωσης αρχείου");
}

setPatterns(lineCounter);


setTrainningSet(new double(getPatterns())(getInput()));
String() temp=new String(getInput());

int index;
int length=0;

for (int i = 0; i < getTrainningSet().length; i++) {
        temp=dataBuffer.get(i).split(";");
  //      for (int j = 0; j < temp.length; j++) {
  //              if(temp(j).compareTo("")==0) System.out.println("line "+j);
```

```
//                                  System.out.println(temp(j));
//                              }
                        for (int j = 0; j <targetDimension+inputDimension; j++) {

                                getTrainningSet()(i)(j)=(Double.parseDouble(temp(j)));
                        }
                }

                double ()maxColumns=new double(inputDimension);
                double tempMax=0;
                for (int i = 0; i < inputDimension; i++) {
                        for (int j = 0; j < trainningSet.length; j++) {
                                if(trainningSet(j)(i)>tempMax) tempMax=trainningSet(j)(i);
                        }
                        maxColumns(i)=tempMax;
                        tempMax=0;
                }

                for (int i = 0; i < trainningSet.length; i++) {
                        for (int j = 0; j < inputDimension; j++) {
                                if(trainningSet(i)(j)==0) trainningSet(i)(j)=0;
                                else trainningSet(i)(j)=trainningSet(i)(j)/maxColumns(j);
                        }
                }

//                  for (int i = 0; i < trainningSet.length; i++) {
//                      for (int j = 0; j < trainningSet(i).length; j++) {
//                          System.out.print("  "+trainningSet(i)(j));
//                      }
//                      System.out.println("");
//                  }

        }

        /**
```

```
* η μέθοδος αυτή δημιουργεί το testing set. επίσης καταμετρά τα δείγματα έτσι ώστε να παρουσιάσει τις κλάσεις
* για να γίνουν στατιστικές συγκρίσεις.
*/
public void createTestingSet(){

        BufferedReader reader = null;
        String line;
        int types=Parameters.types;

        int lineCounter=0;

        ArrayList <String> dataBuffer=new ArrayList<String>();



        try {
                reader = new BufferedReader(new FileReader(this.getTestingSetFile()));
        } catch (FileNotFoundException ex) {
                new JOptionPane().showMessageDialog(null,"Δεν βρέθηκε το αρχείο");
        }
        try {
                while((line=reader.readLine())!=null){
                        dataBuffer.add(line);
                        lineCounter++;
                }
        } catch (IOException ex) {
                new JOptionPane().showMessageDialog(null,"Πρόβλημα ανάγνωσης αρχείου");
        }
        setTestingPatterns(lineCounter);

        setTestingSet(new double(lineCounter)(getInput()+types));

        String() temp=new String(getInput()+types);

    //  System.out.println("testingSet "+ getTestingSet()(1).length+"temp "+temp.length);
```

```
int index;
int length=0;

for (int i = 0; i < getTestingSet().length; i++) {
        temp=dataBuffer.get(i).split(";");
        for (int j = 0; j <temp.length; j++) {
                getTestingSet()(i)(j)=(Double.parseDouble(temp(j)));
        }
}

double ()maxColumns=new double(inputDimension);

double tempMax=0;

for (int i = 0; i < inputDimension; i++) {
        for (int j = 0; j < testingSet.length; j++) {
                if(testingSet(j)(i)>tempMax) tempMax=testingSet(j)(i);
        }
        maxColumns(i)=tempMax;
        tempMax=0;
}

for (int i = 0; i < testingSet.length; i++) {
        for (int j = 0; j < inputDimension; j++) {
                if(testingSet(i)(j)==0) testingSet(i)(j)=0;
                else testingSet(i)(j)=testingSet(i)(j)/maxColumns(j);
        }
}
int t;

for (int i = 0; i < testingSet.length; i++) {
        t= (int)testingSet(i)(testingSet(i).length-1);
        if(t==1) normal++;
        if(t==2) probe++;
        if(t==3) dos++;
```

```
                    if(t==4)r2l++;
                    if(t==5)u2r++;
                    }

//                  for (int i = 0; i < testingSet.length; i++) {
//                      for (int j = 0; j < testingSet(i).length; j++) {
//                          System.out.print("  "+testingSet(i)(j));
//                      }
//                      System.out.println("");
//                  }
        }


        public String getFilename() {
                return getTrainningSetFile();
        }

        public void setFilename(String filename) {
                this.setTrainningSetFile(filename);
        }

        public int getInputDimension() {
                return inputDimension;
        }

        public void setInputDimension(int inputDimension) {
                this.inputDimension = inputDimension;
        }

        public int getTargetDimension() {
                return targetDimension;
        }

        public void setTargetDimension(int targetDimension) {
                this.targetDimension = targetDimension;
```

```
}

public int getPatterns() {
	return patterns;
}

public void setPatterns(int patterns) {
	this.patterns = patterns;
}

public double()() getTrainningSet() {
	return trainningSet;
}

public void setTrainningSet(double()() trainningSet) {
	this.trainningSet = trainningSet;
}

public double()() getTestingSet() {
	return testingSet;
}

public void setTestingSet(double()() testingSet) {
	this.testingSet = testingSet;
}

public int getInput() {
	return input;
}

public void setInput(int input) {
	this.input = input;
}
```

```
	public String getTrainningSetFile() {
		return trainningSetFile;
	}

	public void setTrainningSetFile(String trainningSetFile) {
		this.trainningSetFile = trainningSetFile;
	}

	public String getTestingSetFile() {
		return testingSetFile;
	}

	public void setTestingSetFile(String testingSetFile) {
		this.testingSetFile = testingSetFile;
	}

	public int getTestingPatterns() {
		return testingPatterns;
	}

	public void setTestingPatterns(int testingPatterns) {
		this.testingPatterns = testingPatterns;
	}



}



package nnpackage;

import UIpackage.Parameters;
import java.util.ArrayList;
```

```
import java.util.Iterator;

/**
 * Η κλάση αυτή υλοποιεί ένα Νευρωνικό Δίκτυο
 */
public class NeuralNetwork {

	private int inputNodes;
	private int outputNodes;
	private int hiddenLayersCounter;
	private int hiddenLayer1Nodes;
	private int hiddenLayer2Nodes;
	private double() hiddenLayer1;
	private double() hiddenLayer2;
	private double() inputLayer;
	private double() outputLayer;
	private double()() trainningSet;
	private double()() testingSet;

	private int activationFunctionSelection;

	/**
	 * ο constructor της κλάσης αρχικοποιεί τα πεδία που περιλαμβάνουν τον αριθμό τω κόμβων εισόδου,
	 * τον αριθμό των κόμβων εξόδου, το πλήθος των κρυφών επιπέδων, και τους κόμβους του πρώτου
	 * κρυφού επιπέδου και του δεύτερου κρυφού επιπέδου. Τελος αρχικοποιεί και το πεδίο
	 * επιλογής της συνάρτησης ενεργοποίησης.
	 */

	public NeuralNetwork() {

		inputNodes=Parameters.inputDimension;
		outputNodes=Parameters.targetDimension;
		hiddenLayersCounter=Parameters.hiddenLayersCounter;
		hiddenLayer1Nodes=Parameters.hiddenLayer1Nodes;
```

```
        hiddenLayer2Nodes=Parameters.hiddenLayer2Nodes;
        activationFunctionSelection=Parameters.activationFunctionSelection;

    }

    /**
     *Η μέθοδος δημιουργεί τις δομές δεδομένων του Νευρωνικου δικτύου.
     */
    public void constructNN() {

        inputLayer=new double(inputNodes);
        outputLayer=new double(outputNodes);
        hiddenLayer1=new double(hiddenLayer1Nodes);

        if (hiddenLayersCounter==2){
            hiddenLayer2=new double(hiddenLayer2Nodes);
        }
    }

    /**
     * η μέθοδος αυτή υλοποιεί το περασμα απο το Νευρωνικο Δίκτυο όλου του trainnig set. Επιλέγει το αριθμό των κρυφών επιπέδων
     * ανάλογα με την επιλογή του χρήστη.
     * @param double()() Δέχεται ως παράμετρο ένα πίνακα με τα συνδετικά βάρη
     * @return double Επιστρέφει το mean square error για όλο το training set
     */
    public double runNN(double()() weights){
        double error = 0;

        switch(hiddenLayersCounter){
            case 1: error=runWithOneHiddenLayer(weights);
            break;
            case 2: error=runWithTwoHiddenLayers(weights);
            break;
        }
        return error;
```

```
}

public int getHiddenLayersCounter() {
	return hiddenLayersCounter;
}

public void setHiddenLayersCounter(int hiddenLayersCounter) {
	this.hiddenLayersCounter = hiddenLayersCounter;
}

public int getHiddenLayer1Nodes() {
	return hiddenLayer1Nodes;
}

public double()() getTrainningSet() {
	return trainningSet;
}

public void setTrainningSet(double()() trainningSet) {
	this.trainningSet = trainningSet;
}

public double()() getTestingSet() {
	return testingSet;
}

public void setTestingSet(double()() testingSet) {
	this.testingSet = testingSet;
}

/**
 * Υλοποιεί την επιλογή του Νευρωνικού Δικτύου με ένα κρυφό επίπεδο
 */
private double runWithOneHiddenLayer(double()() weights) {
	int counter=0;
```

```
int k=0;
double sum=0;
double error=0;
int i;

double () target=new double(outputNodes);
double mse=0;

for (i =0; i < trainningSet.length; i++) {
        for (int j = 0; j < inputLayer.length; j++) {
                inputLayer(j)=trainningSet(i)(j);
        }


        while(k<hiddenLayer1.length){
                for (int m = 0; m < weights(0).length; m++) {
                        for (int n = 0; n < inputLayer.length; n++) {
                                sum=(inputLayer(n)*weights(0)(m))+sum;
                        }
                }
                hiddenLayer1(k)=this.activationFunction(sum);
                System.out.print("  "+hiddenLayer1(k));
                k++;
        }
        sum=0;
        k=0;

        while(k<outputLayer.length){
                for (int m = 0; m < weights(1).length; m++) {
                        for (int n = 0; n < hiddenLayer1.length; n++) {
                                sum=(hiddenLayer1(n)*weights(1)(m))+sum;
                        }
                }
                outputLayer(k)=this.activationFunction(sum);
                k++;
```

```
            }

            for (int j = 0; j < target.length; j++) {
                target(j)=trainningSet(i)(inputNodes+j);
            }

            mse=mse(outputLayer,target);
            error=mse+error;
        }

        return error;
}

/**
 * Υλοποιεί την επιλογή του Νευρωνικού Δικτύου με δύο κρυφά επιπεδα
 *
 */
private double runWithTwoHiddenLayers(double()() weights) {

        int counter=0;

        int k=0;
        double sum=0;
        double error=0;
        int i;
        int m=0;

        double () target=new double(outputNodes);
        double mse=0;

        for (i =0; i < trainningSet.length; i++) {

            for (int j = 0; j < inputLayer.length; j++) {
                inputLayer(j)=trainningSet(i)(j);
```

```
}


while(k<hiddenLayer1.length){
        for (int n = 0; ((n < inputLayer.length)&&(m<weights(0).length)); n++) {
                sum=(inputLayer(n)*weights(0)(m))+sum;
                m++;
        }
        hiddenLayer1(k)=this.activationFunction(sum);
        k++;
        sum=0;
}
sum=0;
k=0;
m=0;


while(k<hiddenLayer2.length){
        for (int n = 0; ((n < hiddenLayer1.length)&&(m<weights(1).length)); n++) {
                sum=(hiddenLayer1(n)*weights(1)(m))+sum;
                m++;
        }
        hiddenLayer2(k)=this.activationFunction(sum);
        k++;
        sum=0;
}
sum=0;
k=0;
m=0;

while(k<outputLayer.length){
        for (int n = 0;((n < hiddenLayer2.length)&&(m<weights(2).length)); n++) {
                sum=(hiddenLayer2(n)*weights(2)(m))+sum;
                m++;
        }
```

```
                outputLayer(k)=this.activationFunction(sum);
                k++;
                sum=0;
            }

            for (int j = 0; j < target.length; j++) {
                target(j)=trainningSet(i)(inputNodes+j);
            }

            mse=mse(outputLayer,target);
            error+=mse;
        }

        return error/i;

    }
    /**
     * Υπολογίζει το mse για ένα πρότυπο
     * @param double() double() Δέχεται δύο πίνακες που αντιστοιχούν στην έξοδο και στο διάνυσμα στόχο
     * @return double το mse
     */
    public double mse(double() output, double() target) {

        double result=0;
        double temp=0;
        int i;

        for ( i=0 ; i < output.length; i++) {
            temp=Math.pow((target(i)-output(i)),2)+temp;
        }

        result=temp/i;
        return result;
    }
```

```
public int getActivationFunctionSelection() {
	return activationFunctionSelection;
}

public void setActivationFunctionSelection(int activationFunctionSelection) {
	this.activationFunctionSelection = activationFunctionSelection;
}

/**
 * Η μέθοδος αυτή επιλέγει την συνάρτηση ενεργοποίησης που έχει οριστεί στο
 *user interface
 */
public double activationFunction(double value) {
	double v;
	int selection=getActivationFunctionSelection();
	double result=0;
	v=value;
	switch(selection){
		case(0):result=logistic(v);
		break;
		case(1):result=value;
		break;
	}

	return result;
}

/**
 *Η μέθοδος αυτή υλοποεί την λογιστική συνάρτηση ενεργοποίησης
 *@param double
 *return double
 */
public double logistic(double v) {
	double result=0;
	result=1/(1+Math.pow((Math.E),(-v)));
```

```
            return result;
        }
}



package nnpackage;

import UIpackage.Parameters;


public class nnTesting {

        private int inputNodes;
        private int outputNodes;
        private int hiddenLayersCounter;
        private int hiddenLayer1Nodes;
        private int hiddenLayer2Nodes;
        private double() hiddenLayer1;
        private double() hiddenLayer2;
        private double() inputLayer;
        private double() outputLayer;
        private double()() testingSet;
        private double() target;
        private int activationFunctionSelection;

        public static int rightClassified;
        public static int rc;
        public static int c;
        public static int rightNormal=0;
        public static int missClassified=0;
        public static int rightAttack=0;
        public static int falseAlarms=0;
        public static int rightDos=0;
        public static int rightProbe=0;
```

```
public static int rightR2L=0;
public static int rightU2R=0;
public static int totalData=0;


public nnTesting() {
	inputNodes=Parameters.inputDimension;
	outputNodes=Parameters.targetDimension;
	hiddenLayersCounter=Parameters.hiddenLayersCounter;
	hiddenLayer1Nodes=Parameters.hiddenLayer1Nodes;
	hiddenLayer2Nodes=Parameters.hiddenLayer2Nodes;
	activationFunctionSelection=Parameters.activationFunctionSelection;
}


public void constructNN() {

	inputLayer=new double(inputNodes);
	outputLayer=new double(outputNodes);
	hiddenLayer1=new double(hiddenLayer1Nodes);
	target=new double(outputNodes+1);
	if (hiddenLayersCounter==2){
		hiddenLayer2=new double(hiddenLayer2Nodes);
	}
}

public double runNN(double()() weights){
	double error = 0;

	switch(hiddenLayersCounter){
		case 1:error=runWithOneHiddenLayer(weights);
		break;
		case 2: error=runWithTwoHiddenLayers(weights);
		break;
	}
```

```
        return 0.0;
}


private double runWithOneHiddenLayer(double()() weights) {
        int counter=0;
        int k=0;
        double sum=0;
        double error=0;
        int i;

        double () target=new double(outputNodes);
        double mse=0;

        for (i =0; i < testingSet.length; i++) {
                for (int j = 0; j < inputLayer.length; j++) {
                        inputLayer(j)=testingSet(i)(j);
                }


                while(k<hiddenLayer1.length){
                        for (int m = 0; m < weights(0).length; m++) {
                                for (int n = 0; n < inputLayer.length; n++) {
                                        sum=(inputLayer(n)*weights(0)(m))+sum;
                                }
                        }
                        hiddenLayer1(k)=this.activationFunction(sum);
                        System.out.print("  "+hiddenLayer1(k));
                        k++;
                }
                sum=0;
                k=0;

                while(k<outputLayer.length){
                        for (int m = 0; m < weights(1).length; m++) {
```

```
                        for (int n = 0; n < hiddenLayer1.length; n++) {
                            sum=(hiddenLayer1(n)*weights(1)(m))+sum;
                        }
                    }
                    outputLayer(k)=this.activationFunction(sum);
                    k++;
                }

                for (int j = 0; j < target.length; j++) {
                    target(j)=testingSet(i)(inputNodes+j);
                }

                for (int j = 0; j < target.length; j++) {
                    System.out.print("  "+outputLayer(j)+"  "+target(j)+" ");
                }
                System.out.println("");
            }

            return 0.0;
    }

    private double runWithTwoHiddenLayers(double()() weights) {

            int rightClassified=0;
            int counter=0;
            int k=0;
            double sum=0;
            double error=0;
            int i;
            int m=0;


            double mse=0;

            for (i =0; i < testingSet.length; i++) {
```

```
for (int j = 0; j < inputLayer.length; j++) {
        inputLayer(j)=testingSet(i)(j);
}


while(k<hiddenLayer1.length){
        for (int n = 0; ((n < inputLayer.length)&&(m<weights(0).length)); n++)
{
                sum=(inputLayer(n)*weights(0)(m))+sum;
                m++;
        }
        hiddenLayer1(k)=this.activationFunction(sum);
        k++;
        sum=0;
}
sum=0;
k=0;
m=0;

while(k<hiddenLayer2.length){
        for (int n = 0; ((n < hiddenLayer1.length)&&(m<weights(1).length));
n++) {
                sum=(hiddenLayer1(n)*weights(1)(m))+sum;
                m++;
        }
        hiddenLayer2(k)=this.activationFunction(sum);
        k++;
        sum=0;
}
sum=0;
k=0;
m=0;

while(k<outputLayer.length){
```

```
                    for (int n = 0;((n < hiddenLayer2.length)&&(m<weights(2).length));
n++) {
                        sum=(hiddenLayer2(n)*weights(2)(m))+sum;
                        m++;
                    }
                    outputLayer(k)=this.activationFunction(sum);
                    k++;
                    sum=0;
                }

                for (int j = 0; j < outputLayer.length; j++) {
                    if(outputLayer(j)>0.5) outputLayer(j)=1.0;
                    else outputLayer(j)=0.0;
                }
                for (int j = 0; j < target.length; j++) {
                    target(j)=testingSet(i)(inputNodes+j);
                }

                computeStatistics(target,outputLayer);

            }
           totalData=testingSet.length;
            c=testingSet.length;

            return 0.0;
        }

        public double()() getTestingSet() {
            return testingSet;
        }

        public void setTestingSet(double()() testingSet) {
            this.testingSet = testingSet;
        }

        private int checkResult(double() target, double() outputLayer) {
```

```
        int temp=0;
        for (int i = 0; i < outputLayer.length; i++) {
                if(outputLayer(i)==target(i)) temp++;
        }
        if(temp==outputLayer.length)
                return 1;
        else return 0;
}

private void computeStatistics(double() target, double() outputLayer) {
        double Target=target(0);
        double Output=outputLayer(0);
        int Type=(int)target(1);

        if((Target==1)&&(Output==1)) {
                rightNormal++;
        }
        if((Target==0)&&(Output==0)) {
                rightAttack++;
                switch(Type){
                        case 2: rightProbe++;
                        break;
                        case 3: rightDos++;
                        break;
                        case 4:rightR2L++;
                        break;
                        case 5:rightU2R++;
                        break;

                }
        }
        if((Target==0)&&(Output==1)) missClassified++;
        if((Target==1)&&(Output==0)) {
                missClassified++;
                falseAlarms++;
```

```
		}


	}

	public double activationFunction(double value) {
		double v;
		int selection=this.activationFunctionSelection;
		double result=0;
		v=value;
		switch(selection){
			case(0):result=logistic(v);
			break;
			case(1):result=value;
			break;
		}

		return result;
	}

	public double logistic(double v) {
		double result=0;
		result=1/(1+Math.pow((Math.E),(-v)));
		return result;
	}

}



package particlepackage;

import UIpackage.Monitor;
import UIpackage.Parameters;
import datapackage.Data;
```

```
import java.util.ArrayList;
import java.util.Iterator;
import nnpackage.NeuralNetwork;

/**
 * η κλάση αυτή υλοποιεί τον particle swarm optimization αλγόριθμο.
 */
public class Pso {

	private int particles;
	private ArrayList <Particle> swarm;
	private int iterations=0;
	private int maxIterations;
	private double ()() globalBest;
	private double()() trainningData;
	private Monitor m;
	private double meanFitness=0;
	private double globalFitness=10000;
	private double() fitnessStore;

	/**
	 * ο constructor της κλάσης αρχικοποιεί τις δομές δεδομένων που θα χρησιμοποιήσει
	 * ο αλγόριθμος.
	 */
	public Pso(Data data,Monitor monitor) {

		particles=Parameters.particles;
		swarm=new ArrayList<Particle>(particles);
		maxIterations=Parameters.maxIterations;
		trainningData=data.getTrainningSet();

		globalBest=new double(Parameters.hiddenLayersCounter+1)();

		globalBest(0)=new double(Parameters.inputDimension*Parameters.hiddenLayer1Nodes);
		globalBest(1)=new
double(Parameters.hiddenLayer1Nodes*Parameters.hiddenLayer2Nodes);
```

```
        globalBest(2)=new double(Parameters.hiddenLayer2Nodes*Parameters.targetDimension);
        m=monitor;
}

/**
 * η μέθοδος αυτή υλοποιεί την επαναληπτική διαδικασία για την εξέυρεση της λύσης
 */
public void runAlgorithm(){

        NeuralNetwork neural=new NeuralNetwork();
        neural.constructNN();
        neural.setTrainningSet(trainningData);
        setFitnessStore(new double(maxIterations));

        for (int i = 0; i < particles; i++) {
                swarm.add(i,new Particle());
        }

        for (Iterator it = swarm.iterator(); it.hasNext();) {
                Particle elem = (Particle) it.next();
                elem.initializeParticle();

        }

        while(iterations<maxIterations){
                int counter=0;
                for (Iterator it = swarm.iterator(); it.hasNext();) {
                        Particle elem = (Particle) it.next();
                        elem.setFitness(neural.runNN(elem.getPosition()));
                }

                findGlobalBest();

                for (Iterator it = swarm.iterator(); it.hasNext();) {
                        Particle elem = (Particle) it.next();
```

```
                    elem.setGlobalBestPosition(getGlobalBest());
                    elem.computeNewPosition();
                }
                getFitnessStore()(iterations)=getGlobalFitness();
                iterations++;

        }

}

public double()() getGlobalBest() {
        return globalBest;
}

public void setGlobalBest(double()() globalBest) {
        this.globalBest = globalBest;
}

/**
 * η μέθοδος αυτή βρίσκει και αποθηκευειτη βέλτιστη λύση απο το σύνολο
 * των particles που συμμετέχουν στον αλγόριθμο
 */
public void findGlobalBest() {
        for (Iterator it = swarm.iterator(); it.hasNext();) {
                Particle elem = (Particle) it.next();
                if(getGlobalFitness()>elem.getFitness()){
                        setGlobalFitness(elem.getFitness());
                        saveGlobalBest(elem.getPosition());
                }
        }
}

/**
 * η μέθοδος αυτή υλοποιεί την αποθήκευση της βέλτιστης λύσης
 *στο τέλος κάθε επανάληψης του αλγορίθμου
```

```
     */
    public void saveGlobalBest(double()() d) {
        for (int i = 0; i < globalBest.length; i++) {
            for (int j = 0; j < globalBest(i).length; j++) {
                globalBest(i)(j)=d(i)(j);
            }
        }
    }

     public double() getFitnessStore() {
         return fitnessStore;
     }

     public void setFitnessStore(double() fitnessStore) {
         this.fitnessStore = fitnessStore;
     }

    public synchronized double getGlobalFitness() {
        return globalFitness;
    }

    public synchronized void setGlobalFitness(double globalFitness) {
        this.globalFitness = globalFitness;
    }

}



package particlepackage;

import UIpackage.Parameters;
import java.util.Random;

/**
```

```
 * η κλάση αυτή υλοποιεί ένα particle δηλαδή μια υποψήφια λύση για το πρόβλημα.
 * στην περίπτωση εκπαιδευσης ενός νευρωνικού δικτύου μπορεί να αντιστοιχεί στα
 * συνδετικά του βάρη.
 */
public class Particle {


	private double()() position;
	private double()() velocity;

	private double fitness=10000.0;
	private double()() bestPositionEver;
	private double bestLocalFitness=10000.0;

	private double cognitiveParameter;
	private double socialParameter;
	private double inertia;
	private double maxVelocity;
	private double minVelocity;

	private double()() globalBestPosition;

	/**
	 * ο constructor αρχικοποιεί τις δομές δεδομένων που θα χρησιμοποιήσει η κλαση
	 */
	public Particle() {

		cognitiveParameter=Parameters.cognitiveParameter;
		socialParameter=Parameters.socialParameter;
		inertia=Parameters.inertia;
		maxVelocity=Parameters.maxVelocity;
		minVelocity=Parameters.minVelocity;

		setPosition(new double(Parameters.hiddenLayersCounter+1)());
```

```
		getPosition()(0)=new
double(Parameters.inputDimension*Parameters.hiddenLayer1Nodes);
		getPosition()(1)=new
double(Parameters.hiddenLayer1Nodes*Parameters.hiddenLayer2Nodes);
		getPosition()(2)=new
double(Parameters.hiddenLayer2Nodes*Parameters.targetDimension);


		setVelocity(new double(position.length)());
		for (int i = 0; i < position.length; i++) {
			getVelocity()(i)=new double(getPosition()(i).length);
		}


		bestPositionEver=new double(getPosition().length)();
		for (int i = 0; i < getPosition().length; i++) {
			bestPositionEver(i)=new double(getPosition()(i).length);
		}


		globalBestPosition=new double(getPosition().length)();
		for (int i = 0; i < getPosition().length; i++) {
			globalBestPosition(i)=new double(getPosition()(i).length);
		}
	}


	/**
	 * η μέθοδος αυτή υλοποεί την αρχικοποίηση των σωματιδίων(particles). Η αρχικοπιήσης
αντιστοιχει
	 * σε αρχική θέση (positon) και αρχική ταχύτητα (velocity) κατα απόλυτα τυχαίο τρόπο.
	 */
	public void initializeParticle(){
		Random rand=new Random();


		for (int i = 0; i < getPosition().length; i++) {
			for (int j = 0; j < getPosition()(i).length; j++) {
				getPosition()(i)(j)=rand.nextDouble();


			}
		}
```

```
        for (int i = 0; i < getVelocity().length; i++) {
                for (int j = 0; j < getVelocity()(i).length; j++) {
                        getVelocity()(i)(j)=rand.nextDouble();
                }
        }

}

public double getFitness() {
        return fitness;
}


public void setFitness(double fit) {
 //      System.out.println(""+fit);
        if(fit<bestLocalFitness){
                bestLocalFitness=fit;
                upadateBestPosition(getPosition());
        }
        this.fitness = fit;
}

/**
 * η μέθοδος αυτή υλοποιεί τον υπολογισμό της νέας θέση του σωματιδίου σύμφωνα με
 * τον αλγόριθμο.
 */
public void computeNewPosition() {
        double temp1;
        double temp2;
        double temp3;
        double temp4;
        Random rand=new Random();

        for (int i = 0; i < velocity.length; i++) {
```

```
                for (int j = 0; j < velocity(i).length; j++) {
                        temp1=(velocity(i)(j)*inertia);
                        temp2=((rand.nextDouble()*cognitiveParameter)*(bestPositionEver(i)(j)-
position(i)(j)));
                        temp3=rand.nextDouble()*socialParameter*(getGlobalBestPosition()(i)(j)-
position(i)(j));
                        temp4=temp1+temp2+temp3;
                        if(temp4>Parameters.maxVelocity) velocity(i)(j)=Parameters.maxVelocity;
                        else if (temp4<Parameters.minVelocity)
velocity(i)(j)=Parameters.minVelocity;
                        else velocity(i)(j)=temp4;
                }
        }

        for (int i = 0; i < position.length; i++) {
                for (int j = 0; j < position(i).length; j++) {
                        position(i)(j)=position(i)(j)+velocity(i)(j);
                }
        }

}



public double()() getGlobalBestPosition() {
        return globalBestPosition;
}

/**
 *η μέθοδος αυτή αποθηκευει στο κάθε particle το συνολικά καλύτερη λύση σε
 *κάθε επνάληψη.
 *@param double()() παράμετρος η global best λύση.
 */
public void setGlobalBestPosition(double()() g) {

        for (int i = 0; i < globalBestPosition.length; i++) {
```

```
                for (int j = 0; j < globalBestPosition(i).length; j++) {
                        this.globalBestPosition(i)(j)=g(i)(j);
                }
        }
}

public double()() getPosition() {
        return position;
}

public void setPosition(double()() position) {
        this.position = position;
}

public double()() getVelocity() {
        return velocity;
}

public void setVelocity(double()() velocity) {
        this.velocity = velocity;
}

/**
 * η μέθοδος αυτή αποθηκεύει την βέλτιστη λύση για το συγκεκριμένο σωματίδο αν αυτή
 * είναι καλύτερη απο τηνπροηγούμενη
 * @param double()() Παράμετρος της μεθόδου είναι η νέα λύση για το σωματίδιο
 */
public void upadateBestPosition(double()() d) {
        for (int i = 0; i < bestPositionEver.length; i++) {
                for (int j = 0; j < bestPositionEver(i).length; j++) {
                        bestPositionEver(i)(j)=d(i)(j);
                }
        }
}
}
```

ΠΑΡΑΡΤΗΜΑ Β –Η τεκμηρίωση του κώδικα

nnpackage

Class NeuralNetwork

`java.lang.Object`

`nnpackage.NeuralNetwork`

`public class` **`NeuralNetwork`** `extends java.lang.Object`

Η κλάση αυτή υλοποιεί ένα Νευρωνικό Δίκτυο

Constructor Summary

NeuralNetwork() ο constructor της κλάσης αρχικοποιεί τα πεδία που περαλαμβάνουν τον αριθμό τω κόμβων εισόδου, τον αριθμό των κόμβων εξόδου, το πλήθος των κρυφών επιπέδων, και τους κόμβους του πρώτου κρυφού επιπέδου και του δεύτερου κρυφού επιπέδου.

Method Summary

`double`	activationFunction(double value) Η μέθοδος αυτή επιλέγει την συνάρτηση ενεργοποίησης που έχει οριστεί στο user interface
`void`	constructNN() Η μέθοδος δημιουργεί τις δομές δεδομένων του Νευρωνικου δικτύου.
`int`	getActivationFunctionSelection()
`int`	getHiddenLayer1Nodes()

int	getHiddenLayersCounter()
double()()	getTestingSet()
double()()	getTrainningSet()
double	logistic(double v) Η μέθοδος αυτή υλοποεί την λογιστική συνάρτηση ενεργοποίησης
double	mse(double() output, double() target) Υπολογίζει το mse για ένα πρότυπο
double	runNN(double()() weights) η μέθοδος αυτή υλοποιεί το περασμα απο το Νευρωνικο Δίκτυο όλου του trainnig set.
void	setActivationFunctionSelection(int activationFunctionSelection)
void	setHiddenLayersCounter(int hiddenLayersCounter)
void	setTestingSet(double()() testingSet)
void	setTrainningSet(double()() trainningSet)

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

NeuralNetwork

public **NeuralNetwork**()

ο constructor της κλάσης αρχικοποιεί τα πεδία που περιλαμβάνουν τον αριθμό τω κόμβων εισόδου, τον αριθμό των κόμβων εξόδου, το πλήθος των κρυφών επιπέδων, και τους κόμβους του πρώτου κρυφού επιπέδου και του δεύτερου κρυφού

επιπέδου. Τελος αρχικοποιεί και το πεδίο επιλογής της συνάρτησης ενεργοποίησης.

Method Detail

activationFunction

```
public double activationFunction(double value)
```

Η μέθοδος αυτή επιλέγει την συνάρτηση ενεργοποίησης που έχει οριστεί στο user interface

---

constructNN

```
public void constructNN()
```

Η μέθοδος δημιουργεί τις δομές δεδομένων του Νευρωνικου δικτύου.

---

getActivationFunctionSelection

```
public int getActivationFunctionSelection()
```

---

getHiddenLayer1Nodes

```
public int getHiddenLayer1Nodes()
```

---

getHiddenLayersCounter

```
public int getHiddenLayersCounter()
```

---

getTestingSet

```
public double()() getTestingSet()
```

---

getTrainningSet

```
public double()() getTrainningSet()
```

---

logistic

```
public double logistic(double v)
```

Η μέθοδος αυτή υλοποεί την λογιστική συνάρτηση ενεργοποίησης

Parameters:

`double` - return double

---

mse

```
public double mse(double() output,
                  double() target)
```

Υπολογίζει το mse για ένα πρότυπο

Parameters:

`double()` - double() Δέχεται δύο πίνακες που αντιστοιχούν στην έξοδο και στο διάνυσμα στόχο

Returns:

double το mse

---

runNN

```
public double runNN(double()() weights)
```

η μέθοδος αυτή υλοποιεί το περασμα απο το Νευρωνικο Δίκτυο όλου του trainnig set. Επιλέγει το αριθμό των κρυφών επιπέδων ανάλογα με την επιλογή του χρήστη.

Parameters:

`double()()` - Δέχεται ως παράμετρο ένα πίνακα με τα συνδετικά βάρη

Returns:

double Επιστρέφει το mean square error για όλο το training set

---

setActivationFunctionSelection

```
public void
setActivationFunctionSelection(int activationFunctionSelection)
```

---

setHiddenLayersCounter

```
public void setHiddenLayersCounter(int hiddenLayersCounter)
```

---

setTestingSet

```
public void setTestingSet(double()() testingSet)
```

---

setTrainningSet

```
public void setTrainningSet(double()() trainningSet)
```

---

nnpackage

Class nnTesting

java.lang.Object

nnpackage.nnTesting

public class **nnTesting**extends java.lang.Object

Field Summary	
static int	c
static int	falseAlarms
static int	missClassified
static int	rc
static int	rightAttack
static int	rightClassified
static int	rightDos
static int	rightNormal
static int	rightProbe
static int	rightR2L
static int	rightU2R

static int	totalData

Constructor Summary

nnTesting()

Method Summary

double	activationFunction(double value)
void	constructNN()
double()()	getTestingSet()
double	logistic(double v)
double	runNN(double()() weights)
void	setTestingSet(double()() testingSet)

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

c

```
public static int c
```

---

falseAlarms

```
public static int falseAlarms
```

missClassified
public static int **missClassified**

rc
public static int **rc**

rightAttack
public static int **rightAttack**

rightClassified
public static int **rightClassified**

rightDos
public static int **rightDos**

rightNormal
public static int **rightNormal**

rightProbe
public static int **rightProbe**

rightR2L
public static int **rightR2L**

rightU2R
public static int **rightU2R**

totalData
public static int **totalData**

Constructor Detail

nnTesting

```
public nnTesting()
```

Method Detail

activationFunction

```
public double activationFunction(double value)
```

---

constructNN

```
public void constructNN()
```

---

getTestingSet

```
public double()() getTestingSet()
```

---

logistic

```
public double logistic(double v)
```

---

runNN

```
public double runNN(double()() weights)
```

---

setTestingSet

```
public void setTestingSet(double()() testingSet)
```

---

datapackage

Class Data

```
java.lang.Object
```

datapackage.Data

```
public class Dataextends java.lang.Object
```

η κλάση αυτή υλοποιεί τον χειρισμό των δεδομένων. Διαβάζει το training set και το testing set στη μνήμη σε πίνακες και εκει τα διαμορφώνει έτσι ώστε να γίνουν κατάλληλα για είδοδο στο Νευρωνικό δίκτυο.

Field Summary

int	dos
int	normal
int	probe
int	r2l
int	u2r

Constructor Summary

Data() ○ constructor

Method Summary

void	createTestingSet() η μέθοδος αυτή δημιουργεί το testing set. επίσης καταμετρά τα δείγματα έτσι ώστε να παρουσιάσει τις κλάσεις για να γίνουν στατιστικές συγκρίσεις.
void	createTrainningSet() Η μέθοδος αυτή διαβάζει τα δεδομένα απο το αρχείο που έχει υποδείξει ο χρήστης στη MainForm και δημιουργεί ένα πίνακα δύο διαστάσεων το training set.
java.lang.String	getFilename()
int	getInput()

int	getInputDimension()
int	getPatterns()
int	getTargetDimension()
int	getTestingPatterns()
double()()	getTestingSet()
java.lang.String	getTestingSetFile()
double()()	getTrainningSet()
java.lang.String	getTrainningSetFile()
void	setFilename(java.lang.String filename)
void	setInput(int input)
void	setInputDimension(int inputDimension)
void	setPatterns(int patterns)
void	setTargetDimension(int targetDimension)
void	setTestingPatterns(int testingPatterns)
void	setTestingSet(double()() testingSet)
void	setTestingSetFile(java.lang.String testingSetFile)
void	setTrainningSet(double()() trainningSet)

void	setTrainningSetFile(java.lang.String trainningSetFile)

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

dos
public int **dos**

---

normal
public int **normal**

---

probe
public int **probe**

---

r2l
public int **r2l**

---

u2r
public int **u2r**

Constructor Detail

Data
public **Data**()

- constructor

Method Detail

createTestingSet
public void **createTestingSet**()

η μέθοδος αυτή δημιουργεί το testing set. επίσης καταμετρά τα δείγματα έτσι ώστε να παρουσιάσει τις κλάσεις για να γίνουν στατιστικές συγκρίσεις.

---

createTrainningSet

```
public void createTrainningSet()
```

Η μέθοδος αυτή διαβάζει τα δεδομένα απο το αρχείο που έχει υποδείξει ο χρήστης στη MainForm και δημιουργεί ένα πίνακα δύο διαστάσεων το training set. Τα δεδομένα κανονικοποιούνται στο διάστημα (0,1).

---

getFilename

```
public java.lang.String getFilename()
```

---

getInput

```
public int getInput()
```

---

getInputDimension

```
public int getInputDimension()
```

---

getPatterns

```
public int getPatterns()
```

---

getTargetDimension

```
public int getTargetDimension()
```

---

getTestingPatterns

```
public int getTestingPatterns()
```

---

getTestingSet

```
public double()() getTestingSet()
```

---

getTestingSetFile

```
public java.lang.String getTestingSetFile()
```

---

getTrainningSet

```
public double()() getTrainningSet()
```

---

getTrainningSetFile

```
public java.lang.String getTrainningSetFile()
```

---

setFilename

```
public void setFilename(java.lang.String filename)
```

---

setInput

```
public void setInput(int input)
```

---

setInputDimension

```
public void setInputDimension(int inputDimension)
```

---

setPatterns

```
public void setPatterns(int patterns)
```

---

setTargetDimension

```
public void setTargetDimension(int targetDimension)
```

---

setTestingPatterns

```
public void setTestingPatterns(int testingPatterns)
```

---

setTestingSet

```
public void setTestingSet(double()() testingSet)
```

---

setTestingSetFile

```
public void setTestingSetFile(java.lang.String testingSetFile)
```

setTrainningSet

```
public void setTrainningSet(double()() trainningSet)
```

setTrainningSetFile

```
public void setTrainningSetFile(java.lang.String trainningSetFile)
```

particlepackage

Class Pso

```
java.lang.Object
```

particlepackage.Pso

```
public class Psoextends java.lang.Object
```

η κλάση αυτή υλοποιεί τον particle swarm optimization αλγόριθμο.

Constructor Summary

Pso(datapackage.Data data, UIpackage.Monitor monitor) ο constructor της κλάσης αρχικοποιεί τις δομές δεδομένων που θα χρησιμοποιήσει ο αλγόριθμος.	

Method Summary

void	findGlobalBest() η μέθοδος αυτή βρίσκει και αποθηκευειτη βέλτιστη λύση απο το σύνολο των particles που συμμετέχουν στον αλγόριθμο
double()	getFitnessStore()
double()()	getGlobalBest()

double	getGlobalFitness()
void	runAlgorithm() η μέθοδος αυτή υλοποιεί την επαναληπτική διαδικασία για την εξέυρεση της λύσης
void	saveGlobalBest(double()() d) η μέθοδος αυτή υλοποιεί την αποθήκευση της βέλτιστης λύσης στο τέλος κάθε επανάληψης του αλγορίθμου
void	setFitnessStore(double() fitnessStore)
void	setGlobalBest(double()() globalBest)
void	setGlobalFitness(double globalFitness)

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Pso

```
public Pso(datapackage.Data data,
           UIpackage.Monitor monitor)
```

ο constructor της κλάσης αρχικοποιεί τις δομές δεδομένων που θα χρησιμοποιήσει ο αλγόριθμος.

Method Detail

findGlobalBest

```
public void findGlobalBest()
```

η μέθοδος αυτή βρίσκει και αποθηκευειτη βέλτιστη λύση απο το σύνολο των particles που συμμετέχουν στον αλγόριθμο

---

getFitnessStore

```
public double() getFitnessStore()
```

getGlobalBest

```
public double()() getGlobalBest()
```

getGlobalFitness

```
public double getGlobalFitness()
```

runAlgorithm

```
public void runAlgorithm()
```

η μέθοδος αυτή υλοποιεί την επαναληπτική διαδικασία για την εξέυρεση της λύσης

saveGlobalBest

```
public void saveGlobalBest(double()() d)
```

η μέθοδος αυτή υλοποιεί την αποθήκευση της βέλτιστης λύσης στο τέλος κάθε επανάληψης του αλγορίθμου

setFitnessStore

```
public void setFitnessStore(double() fitnessStore)
```

setGlobalBest

```
public void setGlobalBest(double()() globalBest)
```

setGlobalFitness

```
public void setGlobalFitness(double globalFitness)
```

particlepackage

Class Particle

```
java.lang.Object
```

particlepackage.Particle

`public class` **`Particle`**`extends java.lang.Object`

η κλάση αυτή υλοποιεί ένα particle δηλαδή μια υποψήφια λύση για το πρόβλημα. στην περίπτωση εκπαιδευσης ενός νευρωνικού δικτύου μπορεί να αντιστοιχεί στα συνδετικά του βάρη.

Constructor Summary

Particle()   ο constructor αρχικοποιεί τις δομές δεδομένων που θα χρησιμοποιήσει η κλαση

Method Summary

void	computeNewPosition() η μέθοδος αυτή υλοποιεί τον υπολογισμό της νέας θέση του σωματιδίου σύμφωνα με τον αλγόριθμο.
double	getFitness()
double()()	getGlobalBestPosition()
double()()	getPosition()
double()()	getVelocity()
void	initializeParticle() η μέθοδος αυτή υλοποιεί την αρχικοποίηση των σωματιδίων(particles).
void	setFitness(double fit)
void	setGlobalBestPosition(double()() g) η μέθοδος αυτή αποθηκευει στο κάθε particle το συνολικά καλύτερη λύση σε κάθε επνάληψη.
void	setPosition(double()() position)

void	setVelocity(double()() velocity)
void	upadateBestPosition(double()() d) η μέθοδος αυτή αποθηκεύει την βέλτιστη λύση για το συγκεκριμένο σωματίδο αν αυτή είναι καλύτερη απο τηνπροηγούμενη

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Particle

```
public Particle()
```

ο constructor αρχικοποιεί τις δομές δεδομένων που θα χρησιμοποιήσει η κλαση

Method Detail

computeNewPosition

```
public void computeNewPosition()
```

η μέθοδος αυτή υλοποιεί τον υπολογισμό της νέας θέση του σωματιδίου σύμφωνα με τον αλγόριθμο.

---

getFitness

```
public double getFitness()
```

---

getGlobalBestPosition

```
public double()() getGlobalBestPosition()
```

---

getPosition

```
public double()() getPosition()
```

---

getVelocity

```
public double()() getVelocity()
```

---

initializeParticle

```
public void initializeParticle()
```

η μέθοδος αυτή υλοποεί την αρχικοποίηση των σωματιδίων(particles). Η αρχικοπιήσης αντιστοιχει σε αρχική θέση (positon) και αρχική ταχύτητα (velocity) κατα απόλυτα τυχαίο τρόπο.

---

setFitness

```
public void setFitness(double fit)
```

---

setGlobalBestPosition

```
public void setGlobalBestPosition(double()() g)
```

η μέθοδος αυτή αποθηκευει στο κάθε particle το συνολικά καλύτερη λύση σε κάθε επνάληψη.

Parameters:

`double()()` - παράμετρος η global best λύση.

---

setPosition

```
public void setPosition(double()() position)
```

---

setVelocity

```
public void setVelocity(double()() velocity)
```

---

upadateBestPosition

```
public void upadateBestPosition(double()() d)
```

η μέθοδος αυτή αποθηκεύει την βέλτιστη λύση για το συγκεκριμένο σωματίδο αν αυτή είναι καλύτερη απο τηνπροηγούμενη

Parameters:

`double()()` - Παράμετρος της μεθόδου είναι η νέα λύση για το σωματίδιο