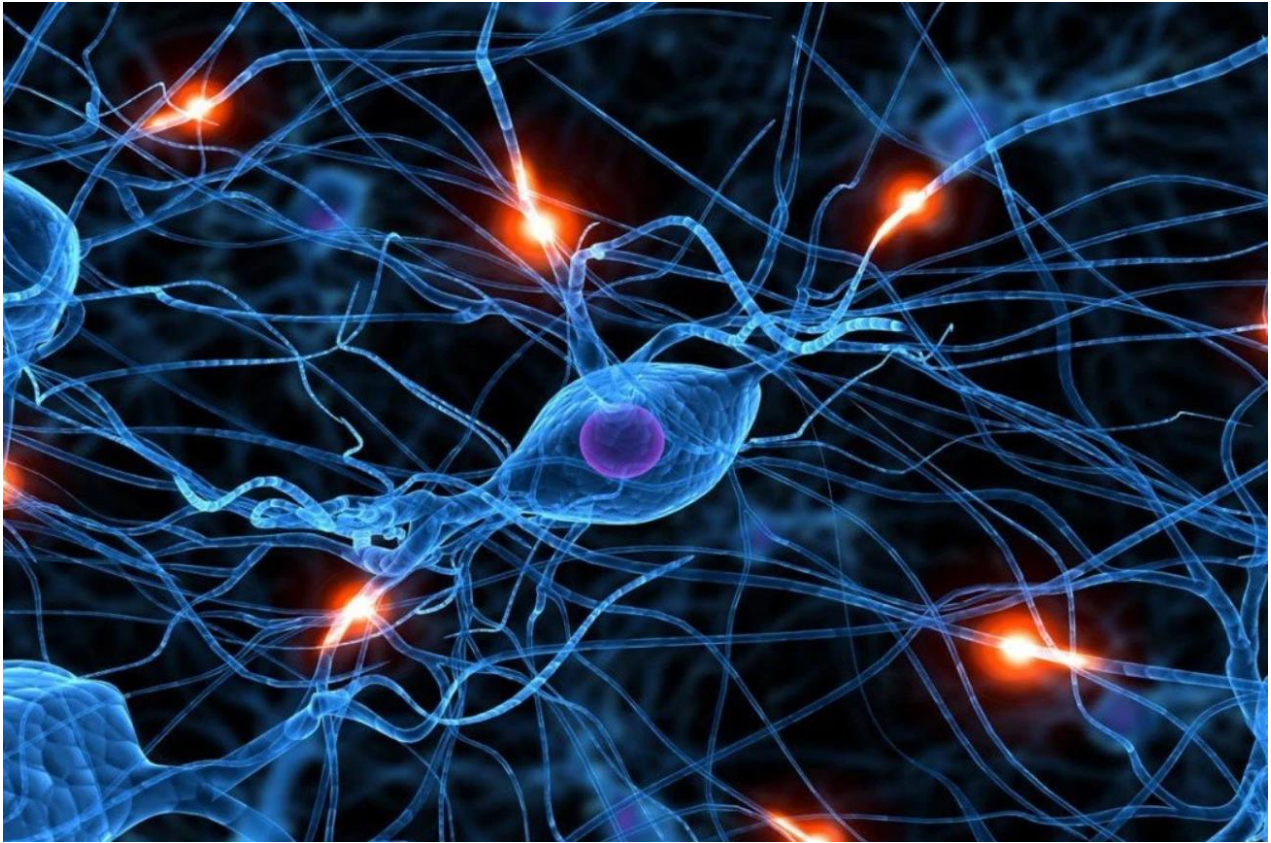


ΥΠΟΛΟΓΙΣΤΙΚΗ ΝΟΗΜΟΣΥΝΗ

ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ ΣΤΑ ΤΕΧΝΗΤΑ ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ

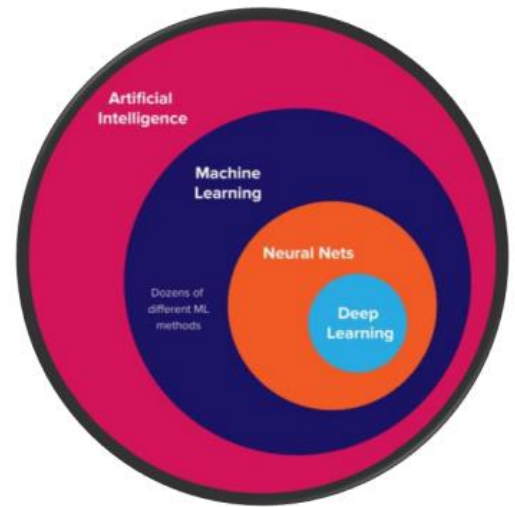


Θέμα: The six-bit parity problem

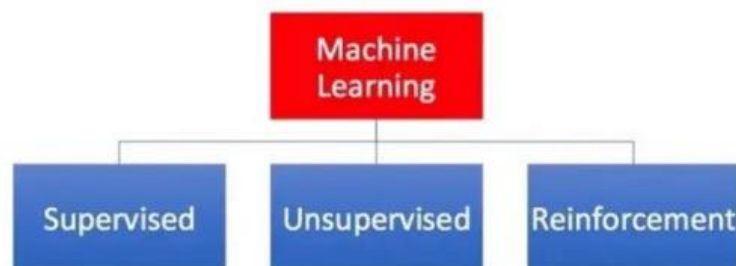
Τεχνητά Νευρωνικά Δίκτυα

Τα τεχνητά νευρωνικά δίκτυα (artificial neural networks) είναι αλγόριθμοι εμπνευσμένοι από τους βιολογικούς νευρώνες μέσα στο ανθρώπινο σώμα. Αποτελούν υποκατηγορία της μηχανικής μάθησης (machine learning) και εμπεριέχουν τα βαθιά νευρωνικά δίκτυα (deep neural networks) ή βαθιά μάθηση (deep learning). Όλα τα παραπάνω είναι υποκατηγορίες ενός εκτεταμένου πεδίου που ονομάζουμε τεχνητή νοημοσύνη (artificial intelligence).

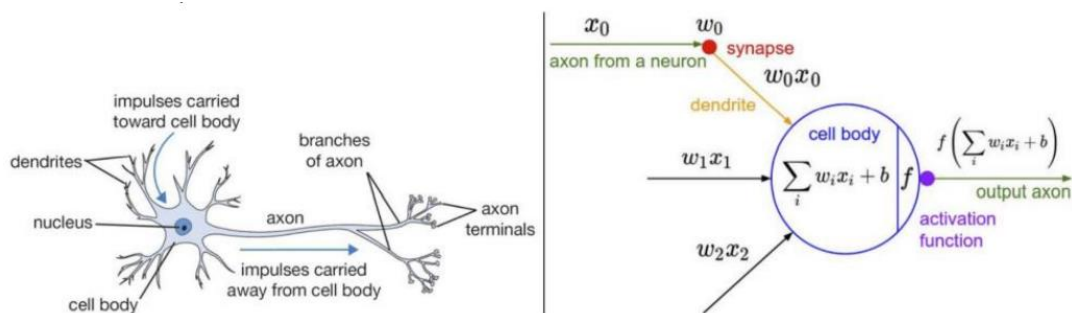
Το πρόβλημα της ταξινόμησης ανήκει στην κατηγορία επιβλεπόμενης μηχανικής μάθησης (supervised machine learning), επειδή οι κλάσεις των εικόνων είναι γνωστές, έχουμε δηλαδή τις ετικέτες (labels).



Types of Machine Learning



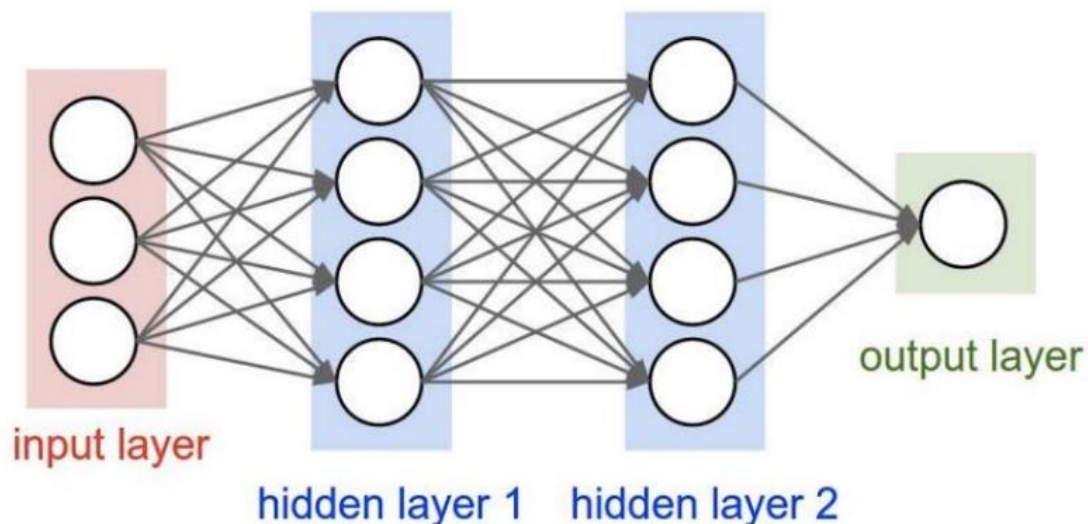
Ο τεχνητός νευρώνας είναι εμπνευσμένος από το βιολογικό νευρώνα του ανθρώπινου εγκεφάλου. Ο συσχετισμός αποτυπώνεται στην παρακάτω εικόνα.



Ένα τεχνητό νευρωνικό δίκτυο αποτελείται από ένα επίπεδο εισόδου (input layer), ένα επίπεδο εξόδου (output layer) και κάποια κρυμμένα επίπεδα (hidden layers).

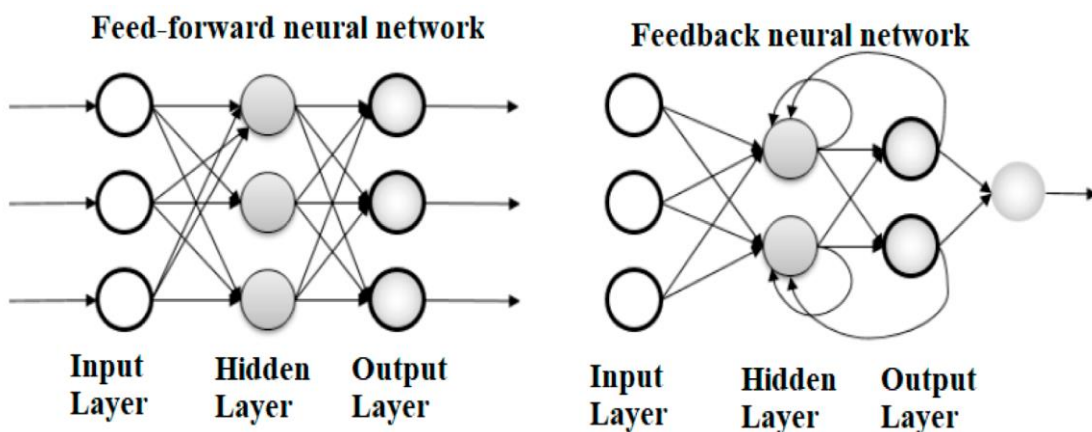
Στα προβλήματα ταξινόμησης, το επίπεδο εξόδου έχει τόσους νευρώνες, όσες και οι κλάσεις που θέλουμε να ταξινομήσουμε. Εξάιρεση αποτελεί το πρόβλημα ταξινόμησης δύο κλάσεων, στο οποίο το επίπεδο εξόδου μπορεί να αποτελείται από έναν μόνο νευρώνα.

Το παρακάτω δίκτυο αποτελεί ένα παράδειγμα νευρωνικού δικτύου που αποτελείται από δύο κρυμμένα επίπεδα.



Κάθε νευρώνας λαμβάνει εισόδους από άλλους νευρώνες ή από εξωτερικές πηγές και εκπέμπει ένα σήμα εξόδου. Το σήμα εξόδου ενός νευρώνα μεταδίδεται στους νευρώνες με τους οποίους συνδέεται μέσω των συνδέσεών του. Οι συνδέσεις μεταξύ των νευρώνων έχουν συνήθως βάρη, τα οποία ρυθμίζονται κατά τη διάρκεια της εκπαίδευσης του δικτύου. Η εκπαίδευση στοχεύει στην εύρεση των βέλτιστων βαρών για την επίλυση ενός συγκεκριμένου προβλήματος.

Τα feedforward και feedback neural networks είναι δύο κατηγορίες τεχνητών νευρωνικών δικτύων. Τα feedforward είναι δίκτυα στα οποία τα σήματα κυκλοφορούν μόνο προς μία κατεύθυνση, από την είσοδο προς την έξοδο. Τα feedback είναι δίκτυα στα οποία τα σήματα κυκλοφορούν και προς τα εμπρός και προς τα πίσω, χρησιμοποιώντας βρόχους ανάδρασης.



The six-bit parity problem

Σε όλες τις δυνατές ακολουθίες 6 δυαδικών ψηφίων αναγνωρίζεται το αν υπάρχει άρτιος ή περιττός αριθμός μη μηδενικών ψηφίων. Αφού δημιουργήσουμε τα δεδομένα, θα χρησιμοποιήσουμε κατάλληλο νευρωνικό δίκτυο που να ταξινομεί οποιαδήποτε ακολουθία 6 δυαδικών ψηφίων σε μια κατηγορία (έξοδος 1) αν ο αριθμός των μη μηδενικών bits είναι άρτιος και σε άλλη κατηγορία (έξοδος 0) αν ο αριθμός των μη μηδενικών bits είναι περιττός.

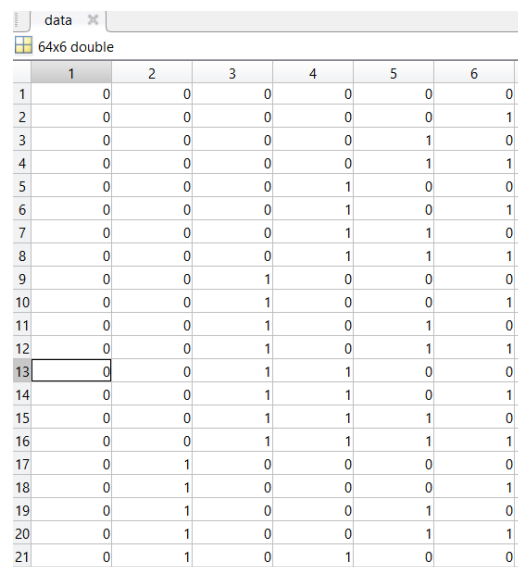
Επίλυση του προβλήματος

Το παραπάνω πρόβλημα προσπαθήσαμε να το επιλύσουμε με τη χρήση νευρωνικών δικτύων.

Αρχικά, πρέπει να δημιουργήσουμε τα δεδομένα εισόδου, τα οποία είναι όλοι οι πιθανοί συνδυασμοί 6 δυαδικών ψηφίων ($2^6=64$ συνδυασμοί). Κάθε ακολουθία είναι ένα διάνυσμα 6 θέσεων. Οι ακολουθίες αποθηκεύονται στη μεταβλητή data με την παρακάτω εντολή. Άρα η μεταβλητή data είναι ένας πίνακας που περιέχει 64 διανύσματα 1×6 , οπότε οι διαστάσεις του είναι τελικά 64×6 .

```
% input data  
data = dec2bin(0:63) - '0';
```

Στην παρακάτω εικόνα βλέπουμε κάποια δείγματα από τα δεδομένα μας.



	1	2	3	4	5	6
1	0	0	0	0	0	0
2	0	0	0	0	0	1
3	0	0	0	0	1	0
4	0	0	0	0	1	1
5	0	0	0	1	0	0
6	0	0	0	1	0	1
7	0	0	0	1	1	0
8	0	0	0	1	1	1
9	0	0	1	0	0	0
10	0	0	1	0	0	1
11	0	0	1	0	1	0
12	0	0	1	0	1	1
13	0	0	1	1	0	0
14	0	0	1	1	0	1
15	0	0	1	1	1	0
16	0	0	1	1	1	1
17	0	1	0	0	0	0
18	0	1	0	0	0	1
19	0	1	0	0	1	0
20	0	1	0	0	1	1
21	0	1	0	1	0	0

Στη συνέχεια, πρέπει να διαχωρίσουμε τα δεδομένα μας, σε δεδομένα εκπαίδευσης (training data) και δεδομένα δοκιμής (testing data).

Επιλέξαμε να τα χωρίζουμε με τυχαίο τρόπο κάθε φορά με τη συνάρτηση randperm, η οποία μας δίνει 50 τυχαίους αριθμούς στο εύρος 1 έως 64, χωρίς να επαναλαμβάνεται κάποιος αριθμός. Κάθε ένας από τους 50 αριθμούς δείχνει τη γραμμή στην οποία βρίσκεται η κάθε ακολουθία 6 δυαδικών ψηφίων στον πίνακα data. Αυτές τις 50 ακολουθίες που επιλέχθηκαν τυχαία τις αποθηκεύουμε στον πίνακα inputs_train. Έπειτα, χρησιμοποιώντας τη συνάρτηση setdiff μπορούμε να βρούμε ποιες ακολουθίες δεν έχουν επιλεγεί (τα id τους βρίσκουμε στην ουσία στον πίνακα data) για το training set και να τις αποθηκεύσουμε σε ένα πίνακα inputs_test. Αυτές οι 14 ακολουθίες αποτελούν το test set.

```
% randomly select 50 rows for training  
idx_train = randperm(size(data, 1), 50);  
inputs_train = data(idx_train, :);  
% use the remaining rows for testing  
idx_test = setdiff(1:size(data, 1), idx_train);  
inputs_test = data(idx_test, :);
```


Παρακάτω μπορούμε να δούμε κάποια δείγματα από το σύνολο εκπαίδευσης και από το σύνολο δοκιμής, καθώς και τα id τους, δηλαδή τις γραμμές στις οποίες αντιστοιχούν στον πίνακα data.

inputs_train							
50x6 double							
	1	2	3	4	5	6	
1	0	1	1	0	0	0	
2	1	0	1	1	0	0	
3	1	0	1	0	0	1	
4	1	0	1	1	0	1	
5	1	0	0	1	0	1	
6	1	0	0	0	1	0	
7	0	0	0	1	1	1	
8	0	1	1	1	0	1	
9	0	0	1	0	1	1	
10	0	0	1	1	0	0	
11	0	1	0	0	1	0	
12	1	1	0	1	1	1	
13	1	1	0	0	0	0	
14	1	1	1	1	0	0	
15	0	0	0	0	1	0	
16	1	1	0	0	1	1	

inputs_test						
14x6 double						
	1	2	3	4	5	6
1	0	0	1	0	0	0
2	0	0	1	1	1	1
3	0	1	0	0	0	1
4	0	1	0	1	1	0
5	0	1	1	1	1	0
6	0	1	1	0	1	1
7	0	1	1	1	0	0
8	1	0	0	0	0	1
9	1	0	0	1	0	0
10	1	0	1	0	1	1
11	1	0	1	1	1	0
12	1	1	0	1	0	1
13	1	1	1	0	0	1
14	1	1	1	0	1	1

idx_train															
1x50 double															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	25	45	42	46	38	35	8	30	12	13	19	56	49	6	3

idx_test													
1x14 double													
	1	2	3	4	5	6	7	8	9	10	11	12	13
1	9	16	18	23	27	28	29	34	37	44	47	54	58

Για παράδειγμα, το `idx_train(15)`, είναι ίσο με 3. Αυτό σημαίνει ότι η ακολουθία 15 του πίνακα `inputs_train` (15^η γραμμή), θα περιέχει την ακολουθία της 3^{ης} γραμμής του πίνακα `data`. Αυτό φαίνεται και στην παρακάτω εικόνα.

data						
64x6 double						
	1	2	3	4	5	6
1	0	0	0	0	0	0
2	0	0	0	0	0	1
3	0	0	0	0	1	0
4	0	0	0	0	1	1

Τέλος, αξίζει να σημειωθεί ότι κάθε φορά που τρέχουμε το πρόγραμμα το σύνολο δεδομένων χωρίζεται σε σύνολο εκπαίδευσης και δοκιμής με διαφορετικό τρόπο, οπότε τα επιμέρους σύνολα θα είναι διαφορετικά κάθε φορά. Επιπλέον, ο κανόνας που χρησιμοποιήθηκε για το διαχωρισμό των δεδομένων ήταν 80%-20%, δηλαδή 80% των συνολικών δεδομένων αποτελούν το training set (50 δείγματα στα 64 -> 78.125%~80%) και 20% το test set (14 δείγματα στα 64 -> 21.875%~20%).

Επίσης, πρέπει να βρούμε τις ετικέτες που αντιστοιχούν σε κάθε δείγμα του συνόλου δεδομένων μας. Με τη συνάρτηση `sum` αθροίζουμε τους άσσους που υπάρχουν σε κάθε ακολουθία και με το `mod` παίρνουμε το υπόλοιπο της διαίρεσης με το 2. Αν η ακολουθία περιέχει **άρτιο αριθμό άσσων**, τότε η πράξη αυτή θα μας επιστρέφει **0**, ενώ αν περιέχει **περιττό αριθμό άσσων** θα επιστρέφει **1**.

Όμως εμείς θέλουμε έξοδος **1** αν ο αριθμός των μη μηδενικών bits (των άσων δηλαδή) είναι **άρτιος** και έξοδος **0** αν ο αριθμός των μη μηδενικών bits είναι **περιττός**. Θέλουμε δηλαδή το αντίθετο, οπότε αρκεί να βάλουμε ένα not (αντιστροφή) μπροστά από την πράξη που κάνουμε. Το not είναι το σύμβολο ~.

Με τον τρόπο που περιγράψαμε βρίσκουμε τις επιθυμητές εξόδους για κάθε ακολουθία και τις αποθηκεύουμε σε δύο πίνακες, έναν για τα training data και ένα για τα testing data.

```
% labels
outputs_train = ~mod(sum(inputs_train, 2), 2); % 1x50 vector, 0
even or 1 odd
outputs_test = ~mod(sum(inputs_test, 2), 2); % 1x14 vector, 0
even or 1 odd
```

Παρακάτω φαίνονται κάποιες επιθυμητές εξοδοι.

outputs_train		outputs_test	
50x1 logical		14x1 logical	
1		1	
1	1	1	0
2	0	2	1
3	0	3	1
4	1	4	0
5	0	5	0
6	1	6	1
7	0	7	0
8	1	8	1
9	0	9	1
10	1	10	1
11	1	11	1
12	0	12	1
13	1	13	1
14	1	14	0
15	0	..	
16	1		
17	0		
18	0		
19	0		
20	0		

inputs_train						
1	2	3	4	5	6	
1	0	1	1	0	0	0
2	1	0	1	1	0	0
3	1	0	1	0	0	1
4	1	0	1	1	0	1
5	1	0	0	1	0	1
6	1	0	0	0	1	0
7	0	0	0	1	1	1
8	0	1	1	1	0	1
9	0	0	1	0	1	1
10	0	0	1	1	0	0
11	0	1	0	0	1	0
12	1	1	0	1	1	1
13	1	1	0	0	0	0
14	1	1	1	1	0	0
15	0	0	0	0	1	0
16	1	1	0	0	1	1

Για παράδειγμα, η 1^η ακολουθία στο σύνολο δοκιμής εδώ είναι η 011000 -> 2 άσσοι, άρτιος αριθμός -> επιθυμητή έξοδος 1, όπως φαίνεται και στην πρώτη θέση του outputs_train.

(~mod(2, 2)=~0=1)

Η 5η ακολουθία στο σύνολο δοκιμής εδώ είναι η 100101 -> 3 άσσοι, περιττός αριθμός -> επιθυμητή έξοδος 0, όπως φαίνεται και στην πέμπτη θέση του outputs_train.

(~mod(3, 2)=~1=0)

Παρακάτω φαίνεται η αρχιτεκτονική του νευρωνικού δικτύου που χρησιμοποιήσαμε. Η επιλογή έγινε μετά από πολλές δοκιμές. Δοκιμάστηκαν διαφορετικοί αριθμός επιπέδων και νευρώνων για το κάθε επίπεδο. Επίσης δοκιμάστηκαν διαφορετικές συναρτήσεις ενεργοποίησης για τα κρυμμένα επίπεδα και για τα επίπεδα εισόδου και εξόδου και διαφορετικοί αλγόριθμοι εκπαίδευσης.

Με τη βοήθεια της συνάρτησης `newff` δημιουργούμε ένα νευρωνικό δίκτυο, το οποίο έχει **6 εισόδους** (inputs) και **1 έξοδο** (output) και αποτελείται από **8 στρώματα (6 κρυμμένα)**, τα οποία αποτελούνται από **3 νευρώνες το καθένα**, εκτός από **το τελευταίο στρώμα που αποτελείται από 1 νευρώνα** και **το πρώτο στρώμα που αποτελείται από 6 νευρώνες**. Πιο συγκεκριμένα, το πρώτο στρώμα αποτελείται από 6 νευρώνες (όσες και οι εισοδοί) και χρησιμοποιείται για την είσοδο των δεδομένων στο δίκτυο. Οι ελάχιστες και οι μέγιστες τιμές των στοιχείων του διανύσματος εισόδου είναι αντίστοιχα [000000,111111]. Τα επόμενα 6 στρώματα αποτελούνται από 3 νευρώνες το καθένα και χρησιμοποιούν την συνάρτηση ενεργοποίησης `tansig`, η οποία είναι η υπερβολική εφάπτομένη. Τέλος, το τελευταίο στρώμα αποτελείται από 1 νευρώνα και χρησιμοποιείται η λογική συνάρτηση `logsig`, η οποία είναι η λογιστική συνάρτηση.

Επειδή το πρόβλημα μας είναι δυαδικής ταξινόμησης (binary classification), δηλαδή έχουμε μόνο δύο κλάσεις εξόδου, χρησιμοποιήσαμε τη λογιστική συνάρτηση ενεργοποίησης (logistic activation function) για το επίπεδο εξόδου, η οποία παράγει έναν αριθμό από το 0 έως το 1, ο οποίος ερμηνεύεται ως η πιθανότητα της εισόδου να ανήκει στη θετική κλάση (κλάση με έξοδο 1). Στη συνέχεια, μπορεί να γίνει ταξινόμηση με τη βοήθεια ενός κατωφλίου (threshold). Με τον τρόπο αυτό μπορεί το επίπεδο εξόδου να έχει έναν μόνο νευρώνα, ο οποίος θα βγάζει σαν έξοδο 0 ή 1.

Ο αλγόριθμος εκπαίδευσης που χρησιμοποιείται είναι ο Levenberg-Marquardt (`trainlm`).

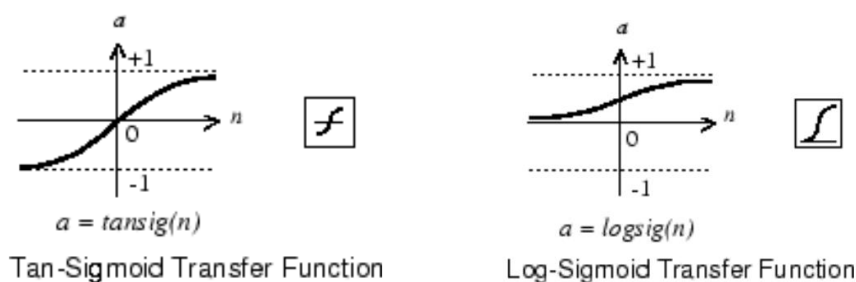
Παρακάτω βλέπουμε το δίκτυό μας, καθώς και ένα άλλο δίκτυο που δοκιμάσαμε και έχουμε βάλει σε σχόλια. Αυτό αποτελείται από 3 επίπεδα, το επίπεδο εισόδου και εξόδου είναι το ίδιο με πριν, όμως το κρυμμένο επίπεδο αποτελείται από 40 νευρώνες. Τα αποτελέσματα ήταν καλύτερα με το δίκτυο που αναλύσαμε παραπάνω, γι' αυτό κρατήσαμε εκείνο.

```
% neural network
%net = newff([zeros(6, 1) ones(6, 1)], [40 1], {'tansig',
'logsig'}, 'trainlm');
net = newff([zeros(6, 1) ones(6, 1)], [3 3 3 3 3 1], {'tansig',
'tansig', 'tansig', 'tansig', 'tansig', 'logsig'},
'trainlm');
```

Παρακάτω φαίνεται ένας πιο αυτοματοποιημένος τρόπος για να ορίσουμε τις ελάχιστες και τις μέγιστες τιμές των στοιχείων του διανύσματος εισόδου, με τη βοήθεια της συνάρτησης `minmax`.

```
net = newff(minmax(data'), [3 3 3 3 3 1], {'tansig', 'tansig',
'tansig', 'tansig', 'tansig', 'logsig'}, 'trainlm');
```

Επίσης, στην παρακάτω εικόνα φαίνονται οι γραφικές παραστάσεις των συναρτήσεων ενεργοποίησης που χρησιμοποιήσαμε στο δίκτυό μας.



Οι παράμετροι που χρησιμοποιήθηκαν για την εκπαίδευση του δικτύου στο σύνολο δεδομένων εκπαίδευσης είναι οι ακόλουθες:

- Ρυθμός μάθησης (learning rate): 0.01
- Αριθμός εποχών (epochs): 1000
- Ανοχή σε λάθος για πρώιμη διακοπή: 10^{-5}

Μια εποχή είναι ένα πλήρες πέρασμα όλων των δεδομένων εκπαίδευσης.

Το δίκτυο θα εκπαιδεύεται μέχρι να φτάσει τις 1000 εποχές ή μέχρι το σφάλμα να γίνει μικρότερο του 10^{-5} , ότι συμβεί πρώτο.

```
% train the network
net.trainParam.show = 50; % The result is shown at every 50th
iteration (epoch)
net.trainParam.lr = 0.01; % Learning rate used in some gradient
schemes
net.trainParam.epochs = 1000; % Max number of iterations
net.trainParam.goal = 1e-5; % Error tolerance; stopping criterion
net1 = train(net, inputs_train', outputs_train');
```

Αφού εκπαιδύσουμε το δίκτυο ήρθε η ώρα να το χρησιμοποιήσουμε για να κάνουμε προβλέψεις στα δεδομένα δοκιμής. Με βάση τις προβλέψεις θα αξιολογήσουμε το δίκτυο.

Παρακάτω βλέπουμε τις προβλέψεις που έγιναν σε κάποια δοκιμή που κάναμε. Παρατηρούμε ότι οι προβλέψεις μας δεν είναι 0 ή 1, αλλά όλες είναι δεκαδικές τιμές ανάμεσα στο 0 και το 1. Με τη χρήση ενός κατωφλίου που το ορίζουμε 0.5, λέμε ότι αν οι τιμές των προβλέψεων είναι ≥ 0.5 , θεωρήσε την έξοδο 1. Διαφορετικά θεωρήσε ότι η έξοδος είναι 0.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	0.9719	0.4180	8.5585e-06	8.5445e-06	0.9848	0.9670	8.5694e-06	1.0000	8.5439e-06	8.5585e-06	8.5438e-06	1.0000	8.5439e-06	8.5438e-06

Πλέον φαίνονται οι προβλέψεις μας μετά την εφαρμογή του κατωφλίου που περιέχουν μόνο τις τιμές 0 και 1.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	1	0	0	0	1	1	0	1	0	0	0	1	0	0

Ο παρακάτω κώδικας περιέχει επίσης τον υπολογισμό του μέσου τετραγωνικού σφάλματος και της ακρίβειας στο σύνολο δοκιμής.

Το μέσο τετραγωνικό σφάλμα καθορίζεται ως:

$$\sum_{i=1}^n (Y_i - \hat{Y}_i)^2$$

sum of the errors of all samples

Η ακρίβεια είναι το άθροισμα των προβλέψεων που ήταν σωστές διαιρεμένο με το συνολικό πλήθος των δειγμάτων του test set.

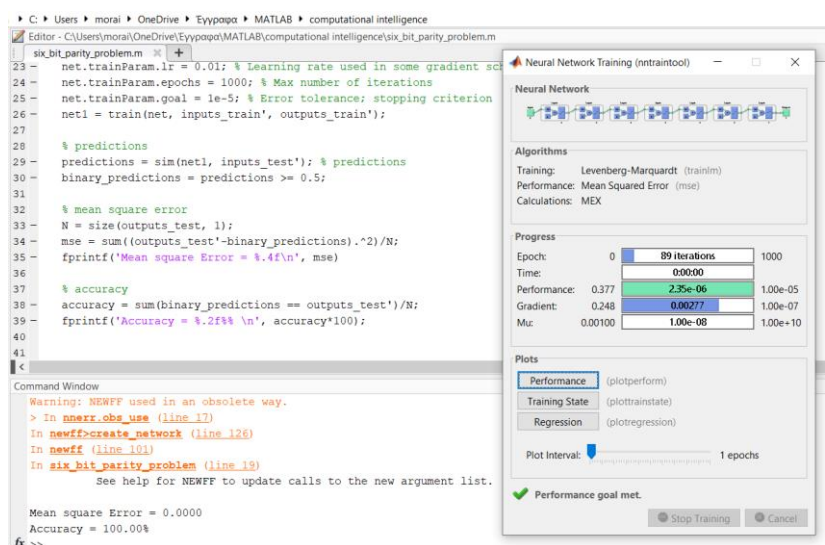
```
% predictions
predictions = sim(net1, inputs_test'); % predictions
binary_predictions = predictions >= 0.5;

% mean square error
N = size(outputs_test, 1);
mse = sum((outputs_test'-binary_predictions).^2)/N;
fprintf('Mean square Error = %.4f\n', mse)

% accuracy
accuracy = sum(binary_predictions == outputs_test')/N;
fprintf('Accuracy = %.2f%% \n', accuracy*100);
```

Αποτελέσματα και διαγράμματα

Τα αποτελέσματα κάθε φορά που τρέχαμε το πρόγραμμα ήταν διαφορετικά. Αυτό συμβαίνει λόγω της τυχαιότητας που υπάρχει στην αρχικοποίηση των βαρών του δικτύου στην αρχή της εκπαίδευσης. Αυτό που μπορούμε να κάνουμε είναι να τρέξουμε αρκετές φορές το δίκτυο και να αποθηκεύσουμε τις παραμέτρους που αντιστοιχούν στο δίκτυο με το μικρότερο σφάλμα. Εδώ κρατήσαμε το καλύτερο αποτέλεσμα που επιλύει και το πρόβλημά μας. Δυστυχώς αυτό το αποτέλεσμα το πετυχαίνουμε σπάνια, πρέπει να τρέξουμε το δίκτυο περίπου από 5 με 10 φορές για να το πετύχουμε, ίσως και παραπάνω.



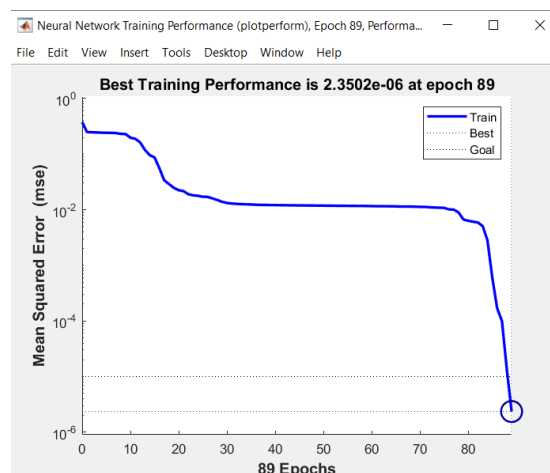
Το τελικό αποτέλεσμα μας ικανοποιεί, αφού έχουμε επιτύχει 0 σφάλματα.

```
Mean square Error = 0.0000  
Accuracy = 100.00%
```

Στο παρακάτω στιγμιότυπο φαίνεται ότι οι προβλέψεις είναι όλες σωστές, είναι δηλαδή ίδιες με τα επιθυμητά αποτελέσματα.

```
>> outputs_test'  
  
ans =  
  
1×14 logical array  
  
1 0 1 1 0 0 0 0 0 0 1 0 1 0  
  
>> binary_predictions  
  
binary_predictions =  
  
1×14 logical array  
  
1 0 1 1 0 0 0 0 0 0 1 0 1 0
```

Το κριτήριο τερματισμού ικανοποιείται στις 89 εποχές και η εκπαίδευση σταματάει, όπως φαίνεται και στο διάγραμμα. Το σφάλμα συνεχώς μειώνεται μέχρι της τιμή που ορίσαμε (10^{-5}).



Κάποιες άλλες τιμές που προέκυψαν όταν τρέχαμε το δίκτυο ήταν οι ακόλουθες:

Mean square Error = 0.0714, Accuracy = 92.86%

Mean square Error = 0.1429, Accuracy = 85.71%

Mean square Error = 0.2857, Accuracy = 71.43%

Mean square Error = 0.3571, Accuracy = 64.29%

Mean square Error = 0.4286, Accuracy = 57.14%

Mean square Error = 0.5714, Accuracy = 42.86%

Συμπεράσματα

Παρατηρήσαμε ότι στις περισσότερες περιπτώσεις το δίκτυό μας προβλέπει σωστά άνω του 50% των δεδομένων δοκιμής. Η ακρίβεια τις περισσότερες φορές που τρέχουμε το δίκτυο είναι άνω του 50%. Επίσης, κάποιες φορές το σφάλμα μηδενίζεται και η ακρίβεια είναι 100% (όλες οι προβλέψεις είναι σωστές), οπότε η αρχιτεκτονική με τα 6 κρυμμένα επίπεδα που έχουν 3 νευρώνες το καθένα προτιμήθηκε.

Η άλλη αρχιτεκτονική που αναφέραμε με ένα κρυμμένο επίπεδο των 40 νευρώνων εμφάνιζε πάρα πολύ μεγάλες διακυμάνσεις στα αποτελέσματα και πάρα πολλές φορές η ακρίβεια κυμαινόταν κάτω του 50%.

Επίσης, προχωρήσαμε σε αύξηση των νευρώνων ανά επίπεδο (10 με 30), παρατηρήσαμε ότι η ακρίβεια έπεφτε πάρα πολύ και η μέγιστη ακρίβεια δεν ξεπερνούσε το 70%. Αυτό μπορεί να συνέβη λόγω υπερπροσαρμογής (overfitting) στα δεδομένα εκπαίδευσης.

Απ' όλα τα παραπάνω καταλήγουμε στο συμπέρασμα ότι για το συγκεκριμένο πρόβλημα η χρήση νευρωνικών δικτύων μπορεί να προσφέρει μια ικανοποιητική και ορισμένες φορές βέλτιστη λύση, μετά από κατάλληλη προσαρμογή των υπερπαραμέτρων του δικτύου. Βέβαια, τα νευρωνικά δίκτυα χρησιμοποιούνται συνήθως για την επίλυση πιο περίπλοκων προβλημάτων που δεν μπορούν να λυθούν με τις κλασικές μεθόδους προγραμματισμού. Οπότε η χρήση νευρωνικών δικτύων σε ένα πρόβλημα που επιλύεται με περίπου 3 γραμμές κώδικα δεν είναι έξυπνη τεχνική και θα πρέπει να αποφεύγεται, εκτός αν πρόκειται για εκπαιδευτικούς λόγους.

Βιβλιογραφία

- Διαφάνειες μαθήματος και εργαστηρίου, "Υπολογιστική Νοημοσύνη" Δημοκρίτειο Πανεπιστήμιο-Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών.
- Διαφάνειες μαθήματος, "Όραση υπολογιστών" Δημοκρίτειο Πανεπιστήμιο-Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών.
- <https://suboptimal.wiki/explanation/mse/>
- <https://www.mathworks.com/help/deeplearning/ref/logsig.html>
- <https://www.mathworks.com/help/deeplearning/ref/tansig.html>
- <https://towardsdatascience.com/what-are-the-types-of-machine-learning-e2b9e5d1756f>
- <https://serokell.io/blog/ai-ml-dl-difference>