ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ

1η ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ ΣΤΟ

ΜΑΘΗΜΑ:

ΥΠΟΛΟΓΙΣΤΙΚΗ ΝΟΥΜΟΣΥΝΗ

ΦΟΙΤΗΤΕΣ:

Σκούρα Κωνσταντίνα ΑΜ: 4168
Ζήγος Αυγουστίνος ΑΜ: 2973
Μπουραζάνης Νικόλαος ΑΜ: 2311

ΠΕΡΙΕΧΟΜΕΝΑ:

ΑΝΑΛΥΣΗ ΚΩΔΙΚΑ

Αρχικά στο αρχείο Main.java δημιουργούμε ένα αντικείμενο τύπου Load και καλούμε 4 φορές την συνάρτηση readFile προκειμένου να γεμίσουμε τον πίνακα points τύπου Point με τις τιμές από τα αρχεία εκπαίδευσης. Την ίδια διαδικασία ακολουθούμε για τον πίνακα elegxos τύπου Point ο οποίος περιέχει όλες τις τιμές από τα αρχεία ελέγχου.

```
        Load load = new Load(elegxos, pointIndex2, points,pointIndex,network, total_Neurons,layersize,H1,H2,H3,K,d);
        load.readFile( filename: "src/main/java/org/example/ekpaideusi1.txt");
        load.readFile( filename: "src/main/java/org/example/ekpaideusi2.txt");
        load.readFile( filename: "src/main/java/org/example/ekpaideusi3.txt");
        load.readFile( filename: "src/main/java/org/example/ekpaideusi4.txt");


        Load load2 = new Load(elegxos, pointIndex2, points, pointIndex,network, total_Neurons,layersize,H1,H2,H3,K,d);
        load2.readFile( filename: "src/main/java/org/example/elegxos1.txt");
        load2.readFile( filename: "src/main/java/org/example/elegxos2.txt");
        load2.readFile( filename: "src/main/java/org/example/elegxos3.txt");
        load2.readFile( filename: "src/main/java/org/example/elegxos4.txt");
```

Η συνάρτηση ReadFile ελέγχει από ποιο αρχείο προέρχονται οι συντεταγμένες και δημιουργεί ένα αντικείμενο Point με το αντίστοιχο flag και στη συνέχεια το προσθέτει στο σωστό πίνακα(points ή elegxos) ανάλογα το αρχείο από το οποίο προέρχεται.

```
public void readFile(String filename) {

    Scanner reader = null;
    try {
        reader = new Scanner(new FileInputStream(filename)); //Diavazw to arxeio filename sth metablhth reader
    } catch (FileNotFoundException e) {
        System.out.println("Could not find file, please verify the path location!!");
        System.exit( status: 0);
    }
    while (reader.hasNextLine()) {
        String line = reader.nextLine();
        Point point;
        Point point2;
        String currentPoint[] = line.split( regex: "\t"); //Kanw split ta x1, x2 me tab giati auths ths morfhs einai to arxeios txt mas kai ta
        if(filename == "src/main/java/org/example/ekpaideusi1.txt"){
            point = new Point(Double.parseDouble(currentPoint[0]), Double.parseDouble(currentPoint[1]), flag: 1);
            points[pointIndex++] = point;
        }else if(filename == "src/main/java/org/example/ekpaideusi2.txt"){
            point = new Point(Double.parseDouble(currentPoint[0]), Double.parseDouble(currentPoint[1]), flag: 2);
            points[pointIndex++] = point;
        }else if(filename == "src/main/java/org/example/ekpaideusi3.txt"){
            point = new Point(Double.parseDouble(currentPoint[0]), Double.parseDouble(currentPoint[1]), flag: 3);
            points[pointIndex++] = point;
        }else if(filename == "src/main/java/org/example/ekpaideusi4.txt"){
            point = new Point(Double.parseDouble(currentPoint[0]), Double.parseDouble(currentPoint[1]), flag: 4);
            points[pointIndex++] = point;
        }

        if(filename == "src/main/java/org/example/elegxos1.txt"){
            point2 = new Point(Double.parseDouble(currentPoint[0]), Double.parseDouble(currentPoint[1]), flag: 1);
            elegxos[pointIndex2++] = point2;
        }else if(filename == "src/main/java/org/example/elegxos2.txt"){
            point2 = new Point(Double.parseDouble(currentPoint[0]), Double.parseDouble(currentPoint[1]), flag: 2);
            elegxos[pointIndex2++] = point2;
        }else if(filename == "src/main/java/org/example/elegxos3.txt"){
            point2 = new Point(Double.parseDouble(currentPoint[0]), Double.parseDouble(currentPoint[1]), flag: 3);
            elegxos[pointIndex2++] = point2;
        }else if(filename == "src/main/java/org/example/elegxos4.txt"){
            point2 = new Point(Double.parseDouble(currentPoint[0]), Double.parseDouble(currentPoint[1]), flag: 4);
            elegxos[pointIndex2++] = point2;
        }
    }
    reader.close();
}
```

Στη κλάση Point αρχικοποιούμε 3 private πεδία τα x1,x2 που αντιστοιχούν στις συντεταγμένες των σημείων που λαμβάνουμε από τα αρχεία .txt και το πεδίο flag το οποίο μας βοηθάει να ξέρουμε από ποιο αρχείο προέρχεται το συγκεκριμένο point.

```java
package org.example;

public class Point {
    3 usages
    private double x1;
    3 usages
    private double x2;
    3 usages
    private int flag;// Gia na ksero se poio arxeio ekpaiushs brisketai kai ti kathgorias einai h eksodos
    8 usages
    Point(double x1, double x2, int flag) {
        this.x1 = x1;
        this.x2 = x2;
        this.flag = flag;
    }

    5 usages
    public double getX1() { return x1; }
    public void setX1(double x1) { this.x1 = x1; }
    5 usages
    public double getX2() { return x2; }
    public void setX2(double x2) { this.x2 = x2; }
    6 usages
    public int getFlag() { return flag; }
    public void setFlag(int flag) { this.flag = flag; }
}
```

Στη συνέχεια για τα 4 επίπεδα του δικτύου μας δημιουργούμε αντικείμενα τύπου Neuron με τα αντίστοιχα id και layer_id και τα προσθέτουμε στο ArrayList network.

```java
for(int i=0;i< total_Neurons;i++){// Arxikopoihsh twn nevronwn
    if(i < H1){
        Neuron n = new Neuron( layer_id: 1, id: i+1);
        network.add(n);
    }else if((i >= H1) && (i < H1 + H2)){
        Neuron n = new Neuron( layer_id: 2, id: i+1);
        network.add(n);
    }else if((i >= H1 + H2) && (i < H1 + H2 + H3)){
        Neuron n = new Neuron( layer_id: 3, id: i+1);
        network.add(n);
    }else{
        Neuron n = new Neuron( layer_id: 4, id: i+1);
        network.add(n);
    }
}
```

Στη κλάση Neuron έχουμε 6 private πεδία:

- Το bias του κάθε νευρώνα
- Ένα ArrayList με τα βάρη που συνδέουν τον κάθε νευρώνα με τους νευρώνες του προηγούμενου επιπέδου
- Ένα ArrayList με την παράγωγο σφάλματος κάθε βάρους του νευρώνα
- Το id του layer στο οποίο ανήκει
- Το προσωπικό του id(το οποίο μετράει από το 1 για τον πρώτο νευρώνα που βάλαμε στο δικτύο και +1 για κάθε καινούριο νευρώνα)
- Τέλος ένα ArrayList με το Τ για κάθε point για να ξέρουμε σε ποια κατηγορία εξόδου ανήκει

```java
package org.example;

import java.util.ArrayList;

18 usages
public class Neuron{
    2 usages
    private double bias;
    2 usages
    private ArrayList<Double> weight = new ArrayList<>();
    2 usages
    private ArrayList<Double> derivativeError = new ArrayList<>();
    3 usages
    private int layer_id;
    3 usages
    private int id;
    4 usages
    private ArrayList<Integer> T = new ArrayList<>();
    4 usages
    Neuron(int layer_id,int id){
        this.layer_id = layer_id;
        this.id = id;

        for(int i = 0;i < 4000;i++){
            T.add(-5);
        }
    }

    36 usages
    public ArrayList<Integer> getT() { return T; }
    public void setT(int t) { this.T = T; }
    18 usages
    public ArrayList<Double> getWeight() { return weight; }
    public void setWeight(ArrayList<Double> weight) { this.weight = weight; }
    4 usages
    public int getLayer_id() { return layer_id; }
    public void setLayer_id(int layer_id) { this.layer_id = layer_id; }
    9 usages
    public double getBias() {return bias;}
    3 usages
    public void setBias(double bias) {this.bias = bias;}
    16 usages
    public ArrayList<Double> getDerivativeError() {return derivativeError;}
    public void setDerivativeError(ArrayList<Double> derivativeError) {this.derivativeError = derivativeError;}
    public int getId() { return id; }
    public void setId(int id) { this.id = id; }
}
```

Αρχικά τα T λαμβάνουν όλα την τιμή -5 στον constructor της κλάσης όμως αργότερα οι νευρώνες που βρίσκονται στο τελευταίο layer(εξόδου) θα λάβουν τις σωστές τιμές χρησιμοποιώντας το flag της κλάσης Point όπως φαίνεται στην παρακάτω εικόνα:

```
71      for(int i = 0; i < points.length;i++){
72          if(points[i].getFlag() == 1){
73              network.get(network.size() - 1).getT().set(i,0);
74              network.get(network.size() - 2).getT().set(i,0);
75              network.get(network.size() - 3).getT().set(i,0);
76              network.get(network.size() - 4).getT().set(i,1);
77          }else if(points[i].getFlag() == 2){
78              network.get(network.size() - 1).getT().set(i,0);
79              network.get(network.size() - 2).getT().set(i,0);
80              network.get(network.size() - 3).getT().set(i,1);
81              network.get(network.size() - 4).getT().set(i,0);
82          }else if(points[i].getFlag() == 3){
83              network.get(network.size() - 1).getT().set(i,0);
84              network.get(network.size() - 2).getT().set(i,1);
85              network.get(network.size() - 3).getT().set(i,0);
86              network.get(network.size() - 4).getT().set(i,0);
87          }else{
88              network.get(network.size() - 1).getT().set(i,1);
89              network.get(network.size() - 2).getT().set(i,0);
90              network.get(network.size() - 3).getT().set(i,0);
91              network.get(network.size() - 4).getT().set(i,0);
92          }
93      }
```

Όμοια για τον πίνακα elegxos:

```
95          for(int i = 0; i < elegxos.length;i++){
96              if(elegxos[i].getFlag() == 1){
97                  network.get(network.size() - 1).getT().set(i,0);
98                  network.get(network.size() - 2).getT().set(i,0);
99                  network.get(network.size() - 3).getT().set(i,0);
100                 network.get(network.size() - 4).getT().set(i,1);
101             }else if(elegxos[i].getFlag() == 2){
102                 network.get(network.size() - 1).getT().set(i,0);
103                 network.get(network.size() - 2).getT().set(i,0);
104                 network.get(network.size() - 3).getT().set(i,1);
105                 network.get(network.size() - 4).getT().set(i,0);
106             }else if(elegxos[i].getFlag() == 3){
107                 network.get(network.size() - 1).getT().set(i,0);
108                 network.get(network.size() - 2).getT().set(i,1);
109                 network.get(network.size() - 3).getT().set(i,0);
110                 network.get(network.size() - 4).getT().set(i,0);
111             }else{
112                 network.get(network.size() - 1).getT().set(i,1);
113                 network.get(network.size() - 2).getT().set(i,0);
114                 network.get(network.size() - 3).getT().set(i,0);
115                 network.get(network.size() - 4).getT().set(i,0);
116             }
117         }
```

Επίσης στη γραμμή 69 της Main.java καλούμε τη συνάρτηση initializeRandomWeights() η οποία αρχικοποιεί με τυχαίες τιμές(χρήση της συνάρτησης Random()) τα weights και τα Bias κάθε νευρώνα όπως φαίνεται στην εικόνα:

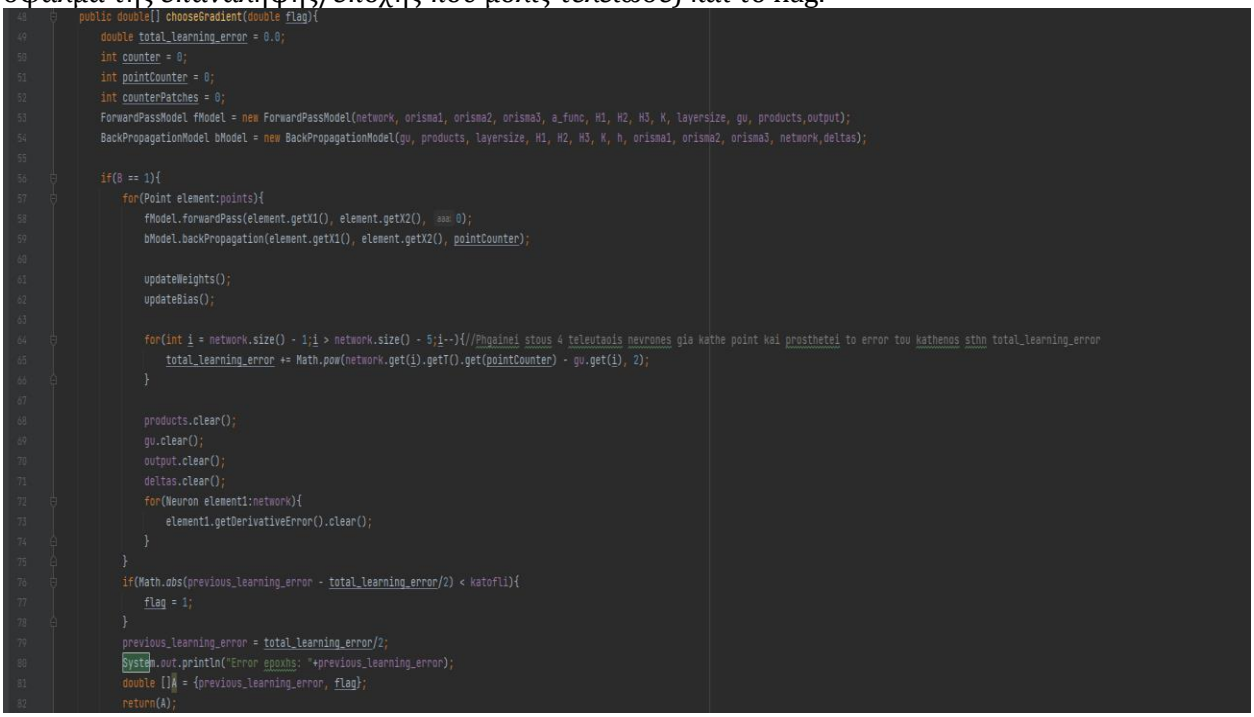
```
84      public void initializeRandomWeights() {
85          Random r = new Random();
86
87          for (int i = 0; i < H1; i++) {//gia kathe neyrwna toy prwtoy krymmenoy
88              for (int j = 0; j < d; j++) {//bazei 2 random weights se kathe neurwna
89                  network.get(i).getWeight().add(Math.round((-1 + 2 * r.nextDouble()) * 100.0) / 100.0);
90              }
91          }
92          for (int i = H1; i < H1 + H2; i++) {//gia kathe neyrwna toy deyterou krymmenoy
93              for (int j = 0; j < H1; j++) {//bazei H1 random weights se kathe neurwna
94                  network.get(i).getWeight().add(Math.round((-1 + 2 * r.nextDouble()) * 100.0) / 100.0);
95              }
96          }
97          for (int i = H1 + H2; i < H1 + H2 + H3; i++) {//gia kathe neyrwna toy tritoy krymmenoy
98              for (int j = 0; j < H2; j++) {//bazei H2 random weights se kathe neurwna
99                  network.get(i).getWeight().add(Math.round((-1 + 2 * r.nextDouble()) * 100.0) / 100.0);
100             }
101         }
102         for (int i = H1 + H2 + H3; i < H1 + H2 + H3 + K; i++) {//gia kathe neyrwna toy tetartoy epipedou
103             for (int j = 0; j < H3; j++) {//bazei H3 random weights se kathe neurwna
104                 network.get(i).getWeight().add(Math.round((-1 + 2 * r.nextDouble()) * 100.0) / 100.0);
105             }
106         }
107
108         for (int i = 0; i < total_Neurons; i++) {
109             network.get(i).setBias(Math.round((-1 + 2 * r.nextDouble()) * 100.0) / 100.0);
110         }
111     }
112 }
```

Στη συνέχεια δημιουργούμε ένα αντικείμενο τύπου GradientDescent και ξεκινάμε μία while loop η οποία τερματίζει αφού ολοκληρωθούν τουλάχιστον 700 επαναλήψεις/εποχές και η διαφορά του σφάλματος της τωρινής και της τελευταίας επανάληψης/εποχής είναι μικρότερο από το κατώφλι, το οποίο θα εξηγηθεί πιο αναλυτικά αργότερα. Μέσα στη while καλούμε τη συνάρτηση chooseGradient(flag) στην οποία δίνουμε ως όρισμα το flag με τιμή 0 εκείνη τη στιγμή η οποία τελικά θα μας επιστρέψει έναν πίνακα με 2 στοιχεία, το πρώτο στοιχείο είναι η τιμή του σφάλματος της εποχής και το δεύτερο η τιμή του flag από την οποία εξαρτάται ο τερματισμός της while όπως προαναφέραμε.

```
119         int g = 0;
120         GradientDescent gradientDescent = new GradientDescent(katofli, a_func, points, network, h, deltas, B, gu, products, layersize, H1, H2, H3, K, orisma1, orisma2, orisma3,output);
121         while((g <=700) || (flag == 0)){
122             flag = 0;
123             double []A = gradientDescent.chooseGradient(flag);
124             flag = A[1];
125             g ++;
126         }
```

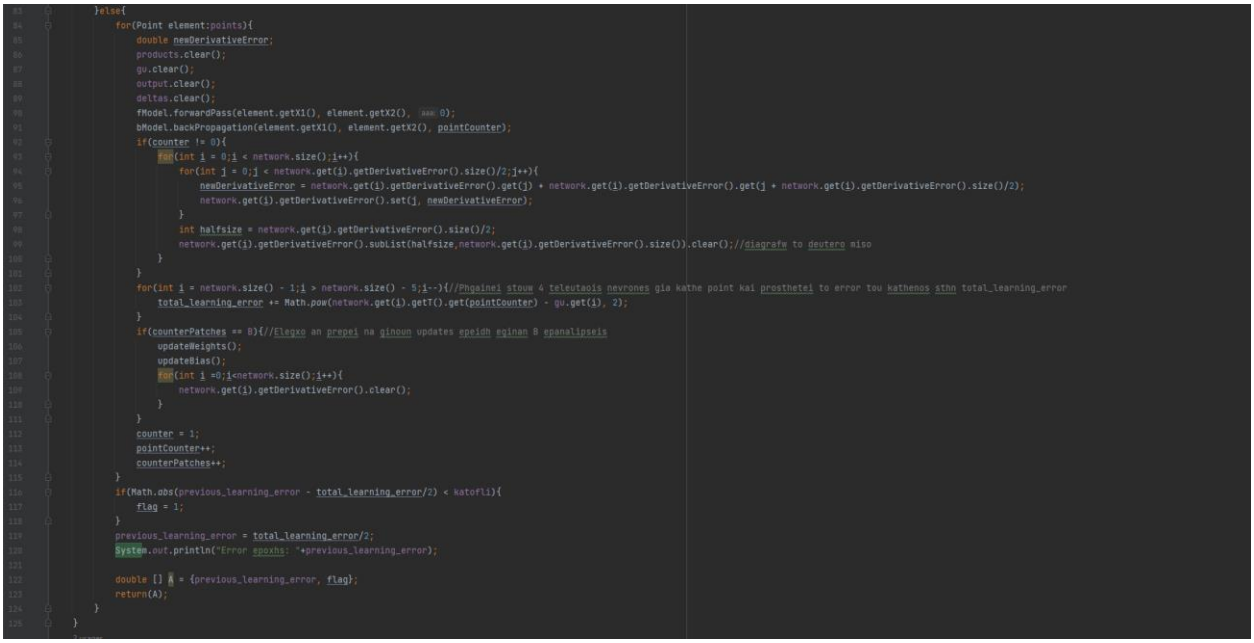
Η chooseGradient(double flag) ξεχωρίζει δύο περιπτώσεις με το if statement της, η πρώτη είναι αν το B==1 και η δεύτερη για B!=1(δηλαδή για την περίπτωση όπου B==N ή B==N/10 ή B==N/100 όπου N=4000). Στην πρώτη περίπτωση διατρέχουμε τον πίνακα points και για κάθε στοιχείο του καλούμε τις συναρτήσεις forwardPass() και backpropagation()(οι λειτουργίες τους εξηγούνται παρακάτω), στη συνέχεια κάνουμε update τα weights και Bias των νευρώνων μέσω των τιμών των deltas και DerivativeErrors που υπολογίζουμε στη συνάρτηση backpropagation(). Τέλος αδειάζουμε τα ArrayList product, gu, output, deltas και DerivativeError με τη χρήση της συνάρτησης clear()ώστε στην επόμενη επανάληψη να μην χρησιμοποιηθούν τα

προηγούμενα , αλλάζουμε την τιμή της flag σε ένα αν η διαφορά μεταξύ του τωρινού σφάλματος με το προηγούμενο είναι μικρότερο του κατωφλίου, εμφανίζουμε το σφάλμα αυτής της εποχής και κάνουμε return τον διθέσιο πίνακα A[] με τις τιμές των μεταβλητών previous_learning_error(το σφάλμα της επανάληψης/εποχής που μόλις τελείωσε) και το flag.

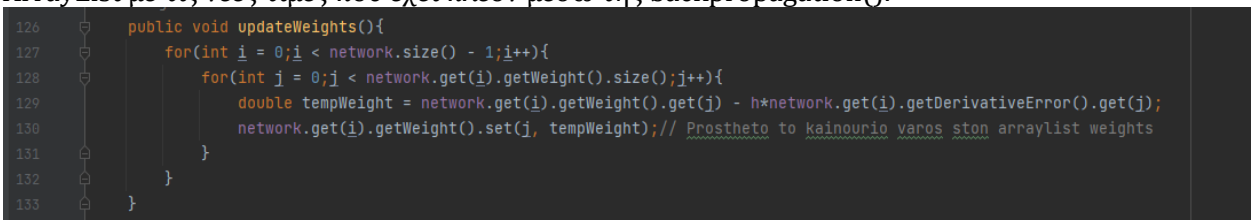


```
public double[] chooseGradient(double flag){
    double total_learning_error = 0.0;
    int counter = 0;
    int pointCounter = 0;
    int counterPatches = 0;
    ForwardPassModel fModel = new ForwardPassModel(network, orisma1, orisma2, orisma3, a_func, H1, H2, H3, K, layersize, gu, products,output);
    BackPropagationModel bModel = new BackPropagationModel(gu, products, layersize, H1, H2, H3, K, h, orisma1, orisma2, orisma3, network,deltas);

    if(0 == 1){
        for(Point element:points){
            fModel.forwardPass(element.getX1(), element.getX2(), aaa 0);
            bModel.backPropagation(element.getX1(), element.getX2(), pointCounter);

            updateWeights();
            updateBias();

            for(int i = network.size() - 1;i > network.size() - 5;i--){//Phgainei stous 4 teleutaois nevrones gia kathe point kai prosthetei to error tou kathenos sthn total_learning_error
                total_learning_error += Math.pow(network.get(i).getT().get(pointCounter) - gu.get(i), 2);
            }

            products.clear();
            gu.clear();
            output.clear();
            deltas.clear();
            for(Neuron element1:network){
                element1.getDerivativeError().clear();
            }
        }
        if(Math.abs(previous_learning_error - total_learning_error/2) < katofli){
            flag = 1;
        }
        previous_learning_error = total_learning_error/2;
        System.out.println("Error epoxhs: "+previous_learning_error);
        double []A = {previous_learning_error, flag};
        return(A);
```

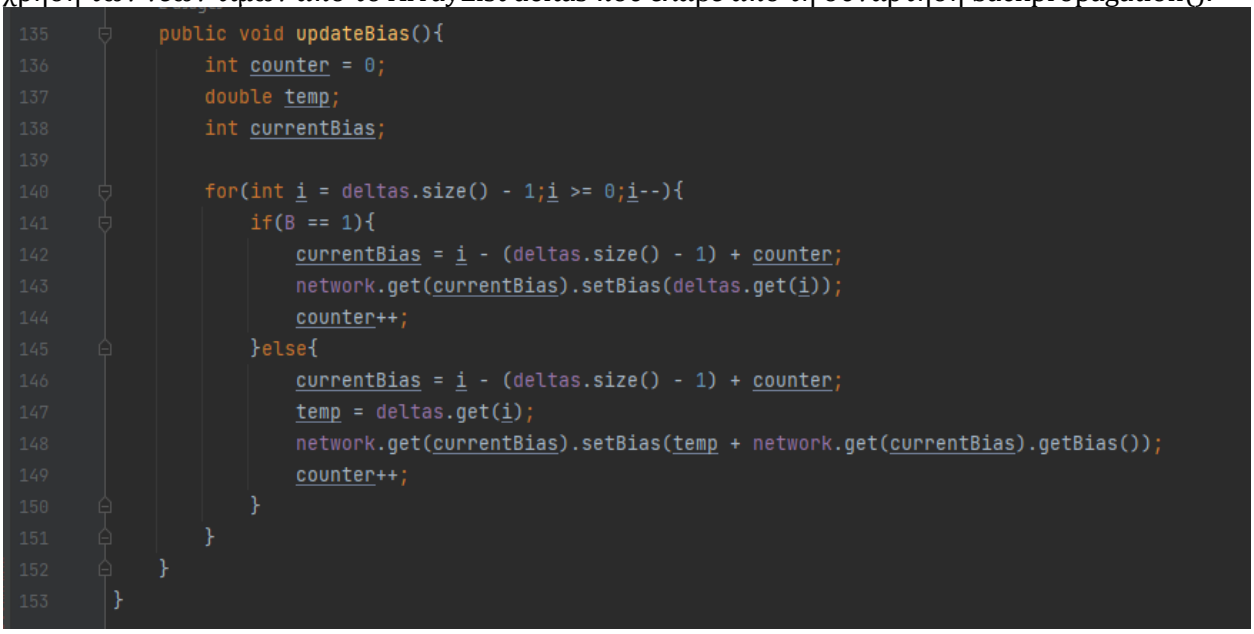
Στην δεύτερη περίπτωση όπου ακολουθούμε την else πρώτα κάνουμε clear τα ArrayList product, gu, output, deltas και στη συνέχεια καλούμε τις συναρτήσεις forwardPass() και backpropagation(). Στη συνέχεια αν δεν βρισκόμαστε στην πρώτη επανάληψη προσθέτουμε τα στοιχεία του πρώτου μισού του ArrayList DerivativeError με αυτά του δεύτερου μισού του(πιο συγκεκριμένα το πρώτο στοιχείο προστίθεται με το πρώτο στοιχείο μετά το μέσο και το αποτέλεσμα αποθηκεύεται στη πρώτη θεση, το δεύτερο στοιχείο προστίθεται με το δεύτερο στοιχείο μετά το μέσο και αποθηκεύεται στη δεύτερη θέση κλπ), αυτό συμβαίνει γιατί όπως θα δούμε και παρακάτω το DerivativeError στις πρώτες του θέσεις κρατάει το sum των παραγώγων των σφαλμάτων των βαρών. Πιο συγκεκριμένα στην backpropagation() στο Arraylist DerivativeError κάνουμε add τις τιμές των παράγωγων των σφαλμάτων των βαρών. (οπότε το πρώτο μισό του DerivativeError χρησιμοποιείται για να κρατάει το sum των παραγώγων των σφαλμάτων των βαρών του νευρώνα και στο δεύτερο μισό μπαίνουν οι παραγωγοί των σφαλμάτων της τελευταίας επανάληψης). Στη συνέχεια υπολογίζουμε το σφάλμα αυτής της επανάληψης/εποχής και ελέγχουμε αν ο αριθμός των points που έχουμε διατρέξει ως τώρα ισούται με το B, σε αυτή τη περίπτωση χρησιμοποιούμε τις συναρτήσεις updateWeights() και updateBias() και κάνουμε clear το DerivativeError. Τέλος ελέγχουμε και εδώ αν ικανοποιείται η συνθήκη κατωφλίου για να αλλάξουμε το flag σε 1, εκτυπώνουμε το σφάλμα αυτής της εποχής και επιστρέφουμε τον πίνακα A[].



Στη συναρτήση updateWeights() χρησιμοποιούμε μία διπλή for loop προκειμένου να διατρέξουμε κάθε βάρος κάθε νευρώνα και να του δώσουμε μια νέα τιμη χρησιμοποιώντας το DerivativeError ArrayList με τις νέες τιμές που έχει πλέον μέσω της backpropagation().



Στη συνάρτηση updateBias() χρησιμοποιούμε μια for loop που διατρέχει το ArrayList deltas με τα σφάλματα κάθε νευρώνα και μέσα της ένα if statement που ελέγχει σε ποια περίπτωση βρισκόμαστε ανάλογα με την τιμή του B προκειμένου να ανανεώσουμε τα bias κάθε νευρώνα με τη χρήση των νέων τιμών από το ArrayList deltas που έλαβε από τη συνάρτηση backpropagation().



```
135     public void updateBias(){
136         int counter = 0;
137         double temp;
138         int currentBias;
139 
140         for(int i = deltas.size() - 1;i >= 0;i--){
141             if(B == 1){
142                 currentBias = i - (deltas.size() - 1) + counter;
143                 network.get(currentBias).setBias(deltas.get(i));
144                 counter++;
145             }else{
146                 currentBias = i - (deltas.size() - 1) + counter;
147                 temp = deltas.get(i);
148                 network.get(currentBias).setBias(temp + network.get(currentBias).getBias());
149                 counter++;
150             }
151         }
152     }
```

Η συνάρτηση forwardPass() χρησιμοποιεί διπλά for loop για κάθε layer του δικτύου(εκτός για την περίπτωση των εισόδων όπου χρειαζόμαστε μονό for loop γιατί διατρέχει τους νευρώνες μόνο του πρώτου κρυμμένου επιπέδου) και υπολογίζει στο καθένα το product του κάθε νευρώνα και το προσθέτει στο ArrayList products.Στη συνέχεια, υπολογίζει με τη χρήση της συνάρτησης activation_function()(θα την εξηγήσουμε παρακάτω) τις εξόδους των νευρώνων για το product του συγκεκριμένου νευρώνα που προσπελάζουμε σε αυτή την επανάληψη, το κάνει add στο ArrayList gu και μηδενίζει το result ώστε να είναι έτοιμο για την επόμενη επανάληψη. Τέλος ,αξίζει να σημειωθεί ότι στην τελευταία περίπτωση για τους νευρώνες εξόδου κάνουμε add όχι μόνο στο ArrayList gu αλλά και στο ArrayList output το οποίο αποθηκεύει μόνο τις εξόδους των συναρτήσεων ενεργοποίησης για τους νευρώνες του layer εξόδου.

```
public void forwardPass(double x1,double x2){
    double result = 0.0;
    double gu1;
    for(int i =0;i<H1;i++){//diatrexw tous neyrwnes  apo to 1o krummeno
        result += x1 * network.get(i).getWeight().get(0) + x2* network.get(i).getWeight().get(1); //prwto baros prwtou nwurwna
        products.add(result + network.get(i).getBias());
        gu1 = activate_function( u: result + network.get(i).getBias(),network.get(i).getLayer_id());
        gu.add(gu1);
        result = 0.0;
    }
    for(int i = H1;i < H1 + H2;i ++){ //diatrexw neyrwnes toy  2ou krymmenou
        for(int j = 0;j <network.get(i).getWeight().size();j++){ //barh toy 2ou krymmenou kathe neurwna
            result += gu.get(j) * network.get(i).getWeight().get(j);//xrhsimopoiw j kai sta 2 gt yparxei antistoixia
        }
        products.add(result + network.get(i).getBias());//prosthetw to u sto arraylist
        gu1 = activate_function( u: result + network.get(i).getBias(),network.get(i).getLayer_id());
        gu.add(gu1);
        result = 0.0;
    }
    for(int i = H1+H2;i < H1+ H2+ H3;i ++){ //diatrexw neyrwnes toy 3ou krymmenou
        for(int j = 0;j <network.get(i).getWeight().size();j++){ //barh toy 3ou krymmenou kathe nwurwna
            result += gu.get(j+H1) * network.get(i).getWeight().get(j);
        }
        products.add(result +  network.get(i).getBias());
        gu1 = activate_function( u: result + network.get(i).getBias(),network.get(i).getLayer_id());
        gu.add(gu1);
        result = 0.0;
    }
    for(int i = H1+H2+H3;i < H1+ H2+ H3 + K;i ++){ //diatrexw neyrwnes toy 4ou epipedou eksodou
        for(int j = 0;j <network.get(i).getWeight().size();j++){ //barh toy 4ou epipedou kathe nwurwna
            result += gu.get(j+H1+H2) * network.get(i).getWeight().get(j);
        }
        products.add(result + network.get(i).getBias());
        gu1 = activate_function( u: result + network.get(i).getBias(),network.get(i).getLayer_id());
        gu.add(gu1);
        output.add(gu1);
        result = 0.0;
    }
}
```

Η συνάρτηση activate_function() δέχεται ως ορίσματα ένα product και το layer_id του νευρώνα στο οποίο ανήκει το product αυτό προκειμένου να αναγνωρίσει ποια συνάρτηση ενεργοποίησης να χρησιμοποιήσει. Αυτό επιτυγχάνεται με τη χρήση if statement όπως φαίνεται στην εικόνα παρακάτω. Αφού αναγνωριστεί ποια συνάρτηση ενεργοποίησης πρέπει να χρησιμοποιήσει ,παίρνει το product που έλαβε για να υπολογίσει το gu που αργότερα θα κάνει add στο ArrayList gu όπως προαναφέρθηκε. Τέλος ,αξίζει να επισημάνουμε ότι κάνουμε μία στρογγυλοποίηση στα 10 δεκαδικά ψηφία το product διότι η συνάρτηση ενεργοποίησης tahn παράγει “Nan” τιμές αν λάβει νούμερο με υπερβολικά μεγάλο αριθμό δεκαδικών ψηφίων.

```java
public double activate_function(double u, int s) {
    double scale = Math.pow(10, 10);  // 10^10 for 10 decimal places
    u = Math.round(u * scale) / scale;

    if(s == 1){
        if (orisma1.equals("relu")){
            a_func = max(0, u); //relu
        }else if(orisma1.equals("tahn")){
            a_func = (exp(u) - exp(-u)) / (exp(u) + exp(-u));//tahn

        }else if(orisma1.equals("sigmoid")){
            a_func = (double) 1 / (1 + exp(-u));//sigmoid
        }else{
            System.out.println("Wrong name");
            exit( status: 1);
        }
    }else if(s==2){
        if (orisma2.equals("relu")){
            a_func = max(0, u); //relu
        }else if(orisma2.equals("tahn")){
            a_func = (exp(u) - exp(-u)) / (exp(u) + exp(-u)); //tahn
        }else if(orisma2.equals("sigmoid")){
            a_func = (double) 1 / (1 + exp(-u));//sigmoid
        }else{
            System.out.println("Wrong name");
            exit( status: 1);
        }
    }else if(s==3){
        if (orisma3.equals("relu")){
            a_func = max(0, u); //relu
        }else if(orisma3.equals("tahn")){
            a_func = (exp(u) - exp(-u)) / (exp(u) + exp(-u)); //tahn
        }else if(orisma3.equals("sigmoid")){
            a_func = (double) 1 / (1 + exp(-u));//sigmoid
        }else{
            System.out.println("Wrong name");
            exit( status: 1);
        }
    }else{
        a_func = (double) 1 / (1 + exp(-u));//sigmoid;
    }
    return a_func;
}
}
```

Επιπρόσθετα, η συνάρτηση backpropagation() αρχικά υπολογίζει τα σφάλματα των νευρώνων του επιπέδου εξόδου(deltas) χρησιμοποιώντας μια for loop και καλώντας την συνάρτηση create_Derivative(εξηγούμε παρακάτω τη λειτουργία) και τα κάνει add στο ArrayList deltas. Στη συνέχεια, χρησιμοποιεί ένα ακόμα for loop για να διατρέξει τα υπόλοιπα κρυμμένα επίπεδα και 2 εμφωλευμένα for ,το ένα ελέγχει σε ποιο επίπεδο βρίσκεται και καλεί την create_Derivative(), το άλλο επίσης ελέγχει σε ποιο επίπεδο βρίσκεται και προσθέτει στη μεταβλητή result το αποτέλεσμα της πράξης βάρους επί σφάλμα των νευρώνων του επόμενου επιπέδου. Στη συνέχεια, υπολογίζει το νέο σφάλμα του νευρώνα πολλαπλασιάζοντας το result και το derivative που μόλις υπολόγισε και το κάνει add στο deltas ArrayList.

```
37      public void backPropagation(double x1, double x2, int pointCounter){
38          double derivative;
39          double delta;
40          double result = 0.0;
41          int delta_count = 0;
42
43          for(int i = network.size() - 1;i > network.size() - (K + 1);i--){
44              derivative = create_Derivative(products.get(i),  neuron_id: i + 1);
45              delta = derivative*(gu.get(i) - network.get(i).getT().get(pointCounter));
46              deltas.add(delta);
47          }
48          for(int i = 2; i>=0; i--){
49              for(int j = layersize[i] - 1;j >=0; j--){
50                  if(i == 2){
51                      derivative = create_Derivative(products.get(j + H1 + H2 ), neuron_id: j + H1 + H2 + 1 );
52
53                  }else if(i ==1){
54                      derivative = create_Derivative(products.get(j + H1 + H2 - layersize[i]), neuron_id: j + H1 + H2 + 1 - layersize[i]);
55
56                  }else{
57                      derivative = create_Derivative(products.get(j + H1  - layersize[i]), neuron_id: j + H1 + 1 - layersize[i]);
58                  }
59                  for(int k =layersize[i + 1] -1; k >=0; k--){
60                      if (i == 2){
61                          result += network.get(H1 + H2 + H3 + k).getWeight().get(j)  * deltas.get(delta_count);
62                          delta_count ++;
63
64                      }else if(i ==1){
65                          result += network.get(H1 + H2 + H3 + k - layersize[i + 1 ]).getWeight().get(j) * deltas.get(delta_count);
66                          delta_count ++;
67                      }else{
68                          result += network.get(H1 + H2  + k - layersize[i + 1 ]).getWeight().get(j) * deltas.get(delta_count);
69                          delta_count ++;
70                      }
71                  }
72                  delta_count =0;
73                  delta = derivative * result;
74                  deltas.add(delta);
75                  result = 0.0;
76              }
77              delta_count += layersize[i+ 1];
78          }
79          compute_Derivative_Error_Weights(x1, x2);
80      }
```

Τέλος, καλεί την συνάρτηση compute_Derivative_Error_Weights(),η οποία χρησιμοποιεί διπλές for προκειμένου να διατρέχει δυο γειτονικά επίπεδα ταυτόχρονα την εξωτερική για να παίρνει τα σφάλματα των νευρώνων του συγκεκριμένου επιπέδου και την εσωτερική για να παίρνει τα gu που αντιστοιχούν στον νευρώνα που καταλήγει το βάρος. Η μόνη εξαίρεση είναι η πρώτη for η οποία είναι μονή επειδή αναφέρεται στο 1ο κρυμμένο, οπότε δε χρειάζεται εμφωλευμένη for για να αναφερόμαστε στις εισόδους.

```
81      public void compute_Derivative_Error_Weights(double x1, double x2){
82          int deltaSize = deltas.size();
83          double metablhth;
84          for(int i = 0;i < H1;i++){// 1o krymmeno kai eisodoi
85              network.get(i).getDerivativeError().add(x1*deltas.get(deltaSize - 1 - i));
86              network.get(i).getDerivativeError().add(x2*deltas.get(deltaSize - 1 - i));
87          }
88          for(int i = H1;i < H2 + H1;i++){// 2o Krummeno
89              for(int j = 0;j < H1;j++){//1o Krummeno
90                  metablhth = gu.get(j)*deltas.get(deltaSize - 1 - i);
91                  network.get(i).getDerivativeError().add(metablhth);
92              }
93          }
94          for(int i = H2 + H1;i < H3 + H2 + H1;i++){// 3o krymmeno
95              for(int j = H1;j < H1 + H2;j++){// 2o krymmeno
96                  network.get(i).getDerivativeError().add(gu.get(j)*deltas.get(deltaSize - 1 - i));
97              }
98          }
99          for(int i = H3 + H2 + H1;i < K + H3 + H2 + H1;i++){// Epipedo eksodou
100             for(int j = H1 + H2;j < H1 + H2 + H3;j++){// 3o krymmeno
101                 network.get(i).getDerivativeError().add(gu.get(j)*deltas.get(deltaSize - 1 - i));
102             }
103         }
104     }
```

Η συνάρτηση create_Derivative() δέχεται 2 ορίσματα, ένα για το product και ένα για το id του νευρώνα με αυτό το product. Αρχικά με if statement ελέγχει σε ποιο επίπεδο βρίσκεται ο νευρώνας αυτός προκειμένου να δώσει στην συνάρτηση helpMeDerivative() ένα String με το όνομα της συνάρτησης ενεργοποίησης και αποθηκεύει στη μεταβλητή derivative την τιμή που επιστρέφει.

```
106     public double create_Derivative(double product, int neuron_id){
107         double derivative;
108
109         if(neuron_id > H3 + H2 + H1){
110             derivative = helpMeDerivative( orisma: "sigmoid", product);
111         }else if(neuron_id > H2 + H1){
112             derivative = helpMeDerivative(orisma3, product);
113         }else if(neuron_id > H1){
114             derivative = helpMeDerivative(orisma2, product);
115         }else{
116             derivative = helpMeDerivative(orisma1, product);
117         }
118         return derivative;
119     }
```

Η συνάρτηση helpMeDerivative() δέχεται δύο ορίσματα, ένα τη συνάρτηση ενεργοποίησης που πρέπει να χρησιμοποιήσει και το product. Ελέγχει με ένα If statement ποια συνάρτηση ενεργοποίησης έλαβε και υπολογίζει με το product την έξοδο της συνάρτησης ενεργοποίησης και την επιστρέφει στην συνάρτηση create_Derivative().

```
121 @    public double helpMeDerivative(String orisma, double product){
122          double derivative;
123
124          if (orisma.equals("relu")){
125              if(product > 0){
126                  derivative = 1.0;
127              }else{
128                  derivative = 0.0;
129              }
130          }else if(orisma.equals("tahn")){
131              derivative = 1 - Math.tanh(product) * Math.tanh(product);
132          }else{
133              double sigmoidX = 1 / (1 + exp(-product)); //ebgala to math
134              derivative = sigmoidX * (1 - sigmoidX);
135          }
136          return derivative;
137      }
138  }
```

Τελικά επιστρέφουμε στο αρχείο Main.java αφού έχει ικανοποιηθεί η συνθήκη της while loop ,στη συνέχεια έχουμε μία for loop η οποία διατρέχει τον πίνακα elegxos για όλα του τα στοιχεία. Πρώτα κάνει clear τα ArrayList output, product, gu και καλεί την συνάρτηση forwardPass() με παραμέτρους τα x1,x2 από κάθε point του πίνακα elegxos. Έτσι υπολογίζονται καινούριες τιμές για το ArrayList output από τις οποίες βρίσκουμε σε ποια θέση του υπάρχει αυτή με τη μεγαλύτερη τιμή και αποθηκεύουμε αυτή τη θέση στη μεταβλητή output_pos. Στη συνέχεια διατρέχουμε τους νευρώνες του επιπέδου εξόδου με μία ακόμα for loop και βρίσκουμε με μία if ποιος από τους 4 αυτούς νευρώνες έχει τιμή 1 για να καταλάβουμε σε ποια κατηγορία εξόδου ανήκει και αποθηκεύουμε τη τιμή της θέσης του νευρώνα στο επίπεδο εξόδου στη μεταβλητή T_pos. Τέλος συγκρίνουμε τις τιμές αυτών των δύο μεταβλητών και αν βρούμε ότι είναι ίσες αυξάνουμε το success_counter κατά 1 γιατί αυτό σημαίνει ότι είχαμε σωστή ταξινόμηση αυτού του point και μόλις βγούμε από την for θα έχουμε υπολογίσει το συνολικό αριθμό επιτυχημένων ταξινομήσεων, τον οποίο εκτυπώνουμε.

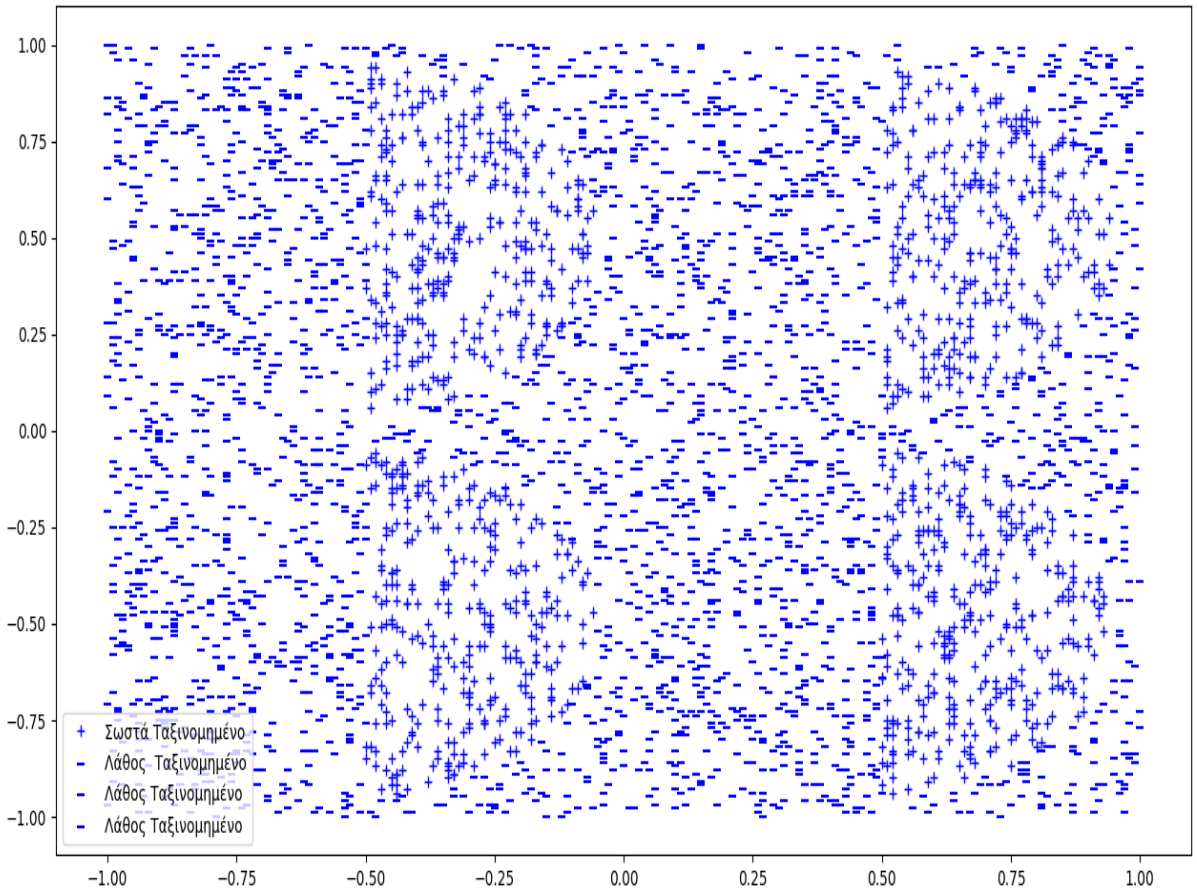
ΠΕΙΡΑΜΑΤΑ

H1	H2	H3	1ο κρυμμένο επίπεδο	2ο κρυμμένο επίπεδο	3ο κρυμμένο επίπεδο	B	Γενικευτική Ικανότητα δικτύου
1	2	1	relu	tahn	relu	N/10	32,52%
1	2	1	relu	sigmoid	sigmoid	N/100	32,52%
5	7	2	relu	sigmoid	sigmoid	N/100	17,57%
5	7	2	tahn	tahn	tahn	N/100	32,12%
5	7	2	tahn	tahn	tahn	N/10	32,52%
5	7	2	relu	relu	relu	N/10	17,77%
5	7	2	sigmoid	sigmoid	sigmoid	N/10	32,12%
5	7	2	sigmoid	relu	tahn	N/10	17,57%
10	13	20	sigmoid	relu	tahn	N/100	17,57%
10	13	20	sigmoid	sigmoid	sigmoid	N/100	32,12%
10	13	20	relu	tahn	relu	N/10	17,77%

Παρατηρούμε ότι τα ποσοστά της γενικευτικής ικανότητας δικτύου επαναλαμβάνονται. Εναλλάσσονται από 17,57%,17,77%,32,12% και 32,52%.Αυτο συμβαίνει διότι ο counter που μετράει τις επιτυχίες γίνεται 703,711,1285 και 1301 αντίστοιχα. Κάθε αρχείο ελέγχου έχει τον ίδιο αριθμό εγγραφών με τις τιμές του counter που προαναφέρθηκαν, δηλαδή το elegxos1.txt έχει 1301,το elegxos2.txt έχει 1285,το elegxos3.txt έχει 711 και το elegxos4.txt έχει 703.Το πρόβλημά μας είναι ότι αναγνωρίζει κάθε φορά τα σημεία μιας κατηγορίας και βγάζει λανθασμένα τα σημεία των άλλων κατηγοριών. Υποθέτουμε ότι

το πρόβλημα μας είναι κάπου στον έλεγχο διότι το ευθύ και το αντίστροφο πέρασμα αλλά και η ενημέρωση των βαρών γίνονται σωστά ,γι’ αυτό καταφέρνει και να αναγνωρίζει όλα τα σημεία μιας κατηγορίας.

<u>Το διάγραμμα του δικτύου με τη καλύτερη γενικευτική ικανότητα</u>



Στον υποφάκελο «createExamples» έχουμε το αρχείο createExamp.java ,με το οποίο δημιουργούμε τα αρχεία ekpaideusi1.txt , ekpaideusi2.txt , ekpaideusi3.txt , ekpaideusi4.txt ,που περιέχουν τα 4000 παραδείγματα εκπαίδευσης. Αντίστοιχα δημιουργούνται τα αρχεία elegxos1.txt, elegxos2.txt , elegxos3.txt , elegxos4.txt που περιέχουν τα παραδείγματα ελέγχου. Έχουμε προσθέσει και ενδεικτικά αρχεία με παραδείγματα εκπαίδευσης και ελέγχου στον υποφάκελο. Πιο συγκεκριμένα, για να δημιουργηθούν 4000 τυχαία σημεία στο τετράγωνο [-1,1]x[-1,1] έχουμε ορίσει ένα άνω όριο upperbound= 100 και ένα κάτω όριο lowerbound = -100 για το x1 και x2 αντίστοιχα και κάθε φορά υπολογίζεται ένα τυχαίο ζεύγος ανάμεσα στο -100 και στο 100 ,το οποίο διαιρούμε με 100 ώστε να είναι στο επιθυμητό εύρος τιμών. Έπειτα ελέγχουμε τα x1,x2 σε ποια από τις 4 κατηγορίες ανήκουν και τα βάζουμε στο αντίστοιχο αρχείο εκπαίδευσης. Για να επιλέξουμε σε ποια κατηγορία είναι τα x1,x2 ελέγχουμε αν ικανοποιούν κάθε μια από τις ανισότητες που δίνει η εκφώνηση. Το ekpaideusi1.txt έχει τα Points για την κατηγόρια 1, το ekpaideusi2.txt για την κατηγορία 2 , το ekpaideusi3.txt για την κατηγορία 3 και το ekpaideusi4.txt για την κατηγορία 4. Ίδια διαδικασία γίνεται και για τα σημεία στα αρχεία ελέγχου.

```java
public class createExamp {
    public static void main(String[] args) {
        float x1= 0.0F;
        float x2= 0.0F;
        int upperbound1 = 0;
        int lowerbound1 = 0;
        int upperbound2 = 0;
        int lowerbound2 = 0;
        try {
            FileWriter myObj1 = new FileWriter("ekpaideusi1.txt");
            FileWriter myObj2 = new FileWriter("ekpaideusi2.txt");
            FileWriter myObj3 = new FileWriter("ekpaideusi3.txt");
            FileWriter myObj4 = new FileWriter("ekpaideusi4.txt");
            FileWriter myObj5 = new FileWriter("elegxos1.txt");
            FileWriter myObj6 = new FileWriter("elegxos2.txt");
            FileWriter myObj7 = new FileWriter("elegxos3.txt");
            FileWriter myObj8 = new FileWriter("elegxos4.txt");

            Random rand = new Random();
            upperbound1 = 100;
            lowerbound1 = -100;
            upperbound2 = 100;
            lowerbound2 = -100;

            for (int i = 0; i < 4000; i++) {
                x1 = (rand.nextFloat() * (upperbound1 - lowerbound1) + lowerbound1) / 100;
                x2 = (rand.nextFloat() * (upperbound2 - lowerbound2) + lowerbound2) / 100;
                String formattedFloat1 = String.format("%.2f", x1);
                String formattedFloat2 = String.format("%.2f", x2);
                if(formattedFloat1.equals("-0.00")){
                if(formattedFloat1.equals("-0.00")){
                    formattedFloat1 = "0.00";
                }
                if(formattedFloat2.equals("-0.00")){
                    formattedFloat2 = "0.00";
                }
                if((Math.pow((x1-0.5),2) + Math.pow((x2-0.5),2) < 0.2) && (x1 > 0.5)){
                    myObj1.write(formattedFloat1);
                    myObj1.write("\t");
                    myObj1.write(formattedFloat2);
                    myObj1.write("\n");
                }else if(( Math.pow((x1-0.5),2) +  Math.pow((x2-0.5),2) < 0.2) && (x1 < 0.5)){
                    myObj2.write(formattedFloat1);
                    myObj2.write("\t");
                    myObj2.write(formattedFloat2);
                    myObj2.write("\n");
                }else if((Math.pow((x1+0.5),2) + Math.pow((x2+0.5),2) < 0.2) && (x1 > -0.5)){
                    myObj1.write(formattedFloat1);
                    myObj1.write("\t");
                    myObj1.write(formattedFloat2);
                    myObj1.write("\n");
                }else if(( Math.pow((x1+0.5),2) +  Math.pow((x2+0.5),2) < 0.2) && (x1 < -0.5)){
                    myObj2.write(formattedFloat1);
                    myObj2.write("\t");
                    myObj2.write(formattedFloat2);
                    myObj2.write("\n");
```

```
            }else if(( Math.pow((x1-0.5),2) +  Math.pow((x2+0.5),2) < 0.2) && (x1 > 0.5)){
                myObj1.write(formattedFloat1);
                myObj1.write("\t");
                myObj1.write(formattedFloat2);
                myObj1.write("\n");
            }else if(( Math.pow((x1-0.5),2) +  Math.pow((x2+0.5),2) < 0.2) && (x1 < 0.5)){
                myObj2.write(formattedFloat1);
                myObj2.write("\t");
                myObj2.write(formattedFloat2);
                myObj2.write("\n");
            }else if(( Math.pow((x1+0.5),2) +  Math.pow((x2-0.5),2) < 0.2) && (x1 > -0.5)){
                myObj1.write(formattedFloat1);
                myObj1.write("\t");
                myObj1.write(formattedFloat2);
                myObj1.write("\n");
            }else if(( Math.pow((x1+0.5),2) +  Math.pow((x2-0.5),2) < 0.2) && (x1 < -0.5)){
                myObj2.write(formattedFloat1);
                myObj2.write("\t");
                myObj2.write(formattedFloat2);
                myObj2.write("\n");
            }else{
                if(x1 > 0){
                    myObj3.write(formattedFloat1);
                    myObj3.write("\t");
                    myObj3.write(formattedFloat2);
                    myObj3.write("\n");

                }else{
                    myObj4.write(formattedFloat1);
                    myObj4.write("\t");
                    myObj4.write(formattedFloat2);
                    myObj4.write("\n");

                }
            }
        }
```

ΟΔΗΓΙΕΣ ΕΚΤΕΛΕΣΗΣ ΠΡΟΓΡΑΜΜΑΤΟΣ

Στον φάκελο ergasia1 έχουμε όλο τον πηγαίο κώδικα και όλα τα .txt αρχεία που πρέπει να είναι στον ίδιο φάκελο για να εκτελεστεί ο κώδικας μας. Πρώτα πρέπει να γίνει η μεταγλώττιση των αρχείων με την εντολή **javac *.java** και στη συνέχεια με την εντολή **java Main orisma1 orisma2 orisma3** όπου orisma1, orisma2,orisma3 πρέπει να πάρει τις τιμές των συναρτήσεων ενεργοποίησης των κρυμμένων επίπεδων. Δηλαδή τα ονόματα είναι :sigmoid, tahn ,relu .Στο τέταρτο επίπεδο εξόδου παίρνει πάντα τη λογιστική συνάρτηση ενεργοποίησης και δεν την βάζει ο χρήστης.
Στον υποφάκελο createExamples έχουμε το αρχείο createExamp που δημιουργεί τα 8 αρχεία .txt.