

```

1 package classes;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 import org.omg.PortableServer.LifespanPolicyOperations;
7
8 /**
9  * created by lgcaobianco on 2018-06-16
10 */
11
12 public class Kohonen {
13     private double[][] grid;
14     private List<Double[]> valoresEntrada;
15     private List<Double[]> conjuntoOperacao;
16     private double taxaAprendizagem;
17     private ArrayList<ArrayList<Integer[]>> listaVizinhos;
18     private double raioVizinhanca = 1;
19     private int quantidadeNeuronios = 16;
20     private double[][] W;
21
22     public void setValoresEntrada(List<Double[]>
valoresEntrada) {
23         this.valoresEntrada = valoresEntrada;
24     }
25
26     public List<Double[]> getConjuntoOperacao() {
27         return this.conjuntoOperacao;
28     }
29
30     public Kohonen() {
31         LeitorPontosEntrada leitor = new
LeitorPontosEntrada(
32             "/home/lgcaobianco/repositorios/epc-rna/
epc10/src/base/valoresEntrada", ".csv");
33         valoresEntrada = leitor.extrairPontos();
34         leitor = new LeitorPontosEntrada("/home/
lgcaobianco/repositorios/epc-rna/epc10/src/base/
conjuntoOperacao",
35             ".csv");
36         conjuntoOperacao = leitor.extrairPontos();
37         Printer.imprimirQualquer(valoresEntrada);
38         grid = new double[4][4];
39         W = new double[16][3];
40         listaVizinhos = new ArrayList<ArrayList<Integer[]>
>();
41         for (int i = 0; i < quantidadeNeuronios; i++) {
42             listaVizinhos.add(i, new ArrayList<Integer[]>()
));
43     }

```

```

44         int inicioX = 0, inicioY = 0;
45         double distanciaX, distanciaY, distancia;
46         while (inicioX < 4) {
47             while (inicioY < 4) {
48                 for (int i = 0; i < grid.length; i++) {
49                     for (int j = 0; j < grid[i].length; j
50 ++) {
51                         distanciaX = Math.pow(inicioX - i,
52 2);
53                         distanciaY = Math.pow(inicioY - j,
54 2);
55                         distancia = Math.sqrt(distanciaX +
56 distanciaY);
57                         if (distancia <= 1 && distancia !=
58 0) {
59                             Integer[] vetorAux = new
60 Integer[2];
61                             vetorAux[0] = i;
62                             vetorAux[1] = j;
63                             listaVizinhos.get((inicioX * 4
64 ) + inicioY).add(vetorAux);
65                         }
66                     }
67                 }
68                 inicioY++;
69             }
70             inicioY = 0;
71             inicioX++;
72         }
73         Printer.imprimirQualquer(listaVizinhos);
74         for (int i = 0; i < 16; i++) {
75             for (int j = 0; j < 3; j++) {
76                 W[i][j] = valoresEntrada.get(i)[j].
77 doubleValue();
78             }
79         }
80         System.out.println("W INICIAL");
81         Printer.imprimirQualquer(W);
82     }
83
84     public double
85     calcularDistanciaEuclidianaEntreEntradaENeuronio(int
86 linhaMatrizEntrada, int linhaMatrizNeuronios) {
87         double catetoX, catetoY, catetoZ;
88         catetoX = Math.pow(W[linhaMatrizNeuronios][0] -
89 valoresEntrada.get(linhaMatrizEntrada)[0], 2);
90         catetoY = Math.pow(W[linhaMatrizNeuronios][1] -
91 valoresEntrada.get(linhaMatrizEntrada)[1], 2);
92         catetoZ = Math.pow(W[linhaMatrizNeuronios][2] -
93 valoresEntrada.get(linhaMatrizEntrada)[2], 2);

```

```

81         return Math.sqrt(catetoX + catetoY + catetoZ);
82     }
83 }
84
85     public void faseTreinamento() {
86         double distanciaAmostraParaNeuronio,
87         distanciaAnterior = 1;
88         int posicaoMenorDistancia = 0;
89         for (int i = 0; i < valoresEntrada.size(); i++) {
90             distanciaAnterior = 1;
91             for (int j = 0; j < W.length; j++) {
92                 distanciaAmostraParaNeuronio =
93                 calcularDistanciaEuclidianaEntreEntradaENeuronio(i, j);
94                 if (distanciaAmostraParaNeuronio <
95                 distanciaAnterior) {
96                     distanciaAnterior =
97                     distanciaAmostraParaNeuronio;
98                     posicaoMenorDistancia = j;
99                 }
100             }
101             ajustarPesosNeuronio(posicaoMenorDistancia, i
102 );
103         }
104     }
105
106     private void ajustarPesosNeuronio(int
107 posicaoMenorDistancia, int linhaMatrizEntrada) {
108         for (int i = 0; i < W[posicaoMenorDistancia].
109         length; i++) {
110             W[posicaoMenorDistancia][i] = W[
111             posicaoMenorDistancia][i]
112             + taxaAprendizagem * (valoresEntrada.
113             get(linhaMatrizEntrada)[i] - W[posicaoMenorDistancia][i])
114             ;
115         }
116         for (int i = 0; i < listaVizinhos.get(
117         posicaoMenorDistancia).size(); i++) {
118             int linhaNeuronioVizinho = listaVizinhos.get
119             (posicaoMenorDistancia).get(i)[0] * 4
120             + listaVizinhos.get(
121             posicaoMenorDistancia).get(i)[1];
122             ajustePesoNeuronioVizinho(
123             linhaNeuronioVizinho, linhaMatrizEntrada);
124             normalizarVetorVizinho(linhaNeuronioVizinho)
125             ;
126         }
127     }
128
129     private void ajustePesoNeuronioVizinho(int

```

```

115 linhaNeuoronioVizinho, int linhaMatrizEntrada) {
116     for (int i = 0; i < W[linhaNeuoronioVizinho].
length; i++) {
117         W[linhaNeuoronioVizinho][i] = W[
linhaNeuoronioVizinho][i] + ((taxaAprendizagem / 2)
118             * (valoresEntrada.get(
linhaMatrizEntrada)[i] - W[linhaNeuoronioVizinho][i]));
119     }
120 }
121
122 private void normalizarVetorVizinho(int
linhaNeuoronioVizinho) {
123     double somaCoordenadas = 0;
124     for (int i = 0; i < W[linhaNeuoronioVizinho].
length; i++) {
125         somaCoordenadas += Math.pow(W[
linhaNeuoronioVizinho][i], 2);
126     }
127     for (int i = 0; i < W[linhaNeuoronioVizinho].
length; i++) {
128         W[linhaNeuoronioVizinho][i] = W[
linhaNeuoronioVizinho][i] / Math.sqrt(somaCoordenadas);
129     }
130 }
131 }
132
133 public void treinarKohonen() {
134     int contadorEpocas = 0;
135     while (contadorEpocas < 10000) {
136         faseTreinamento();
137         contadorEpocas++;
138     }
139     System.out.println("W FINAL");
140     Printer.imprimirQualquer(W);
141 }
142
143 public void faseOperacao() {
144     int posicaoFinal = 0;
145     for (int i = 0; i < valoresEntrada.size(); i++) {
146         double distanciaAnterior =
calcularDistanciaEuclidianaEntreEntradaENeuronio(i, 0);
147         for (int j = 1; j < W.length; j++) {
148             double distanciaAtual =
calcularDistanciaEuclidianaEntreEntradaENeuronio(i, j);
149             if (distanciaAtual < distanciaAnterior) {
150                 distanciaAtual = distanciaAnterior;
151                 posicaoFinal = j;
152             }
153         }
154         System.out.println("A amostra de entrada: " +

```

```
154 i + " foi atribuida ao neuronio: " + posicaoFinal);  
155     }  
156 }  
157  
158 }  
159
```

```

1 package classes;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 /**
7  * created by lgcaobianco on 2018-06-16
8  */
9
10 public class Printer {
11     public static void imprimirQualquer(double[][] object)
12     {
13         for (int i = 0; i < object.length; i++) {
14             for (int j = 0; j < object[i].length; j++) {
15                 System.out.print(object[i][j] + " ");
16             }
17             System.out.println();
18         }
19         System.out.println("\n\n");
20     }
21
22     public static void imprimirQualquer(double[] object) {
23         for (int i = 0; i < object.length; i++) {
24             System.out.print(object[i] + " ");
25         }
26         System.out.println("\n\n");
27     }
28
29     public static void imprimirQualquer(List<Double[]>
30 object) {
31         for (int i = 0; i < object.size(); i++) {
32             for (int j = 0; j < object.get(i).length; j++)
33             {
34                 System.out.print(object.get(i)[j] + " ");
35             }
36             System.out.println();
37         }
38         System.out.println("\n\n");
39     }
40
41     public static void imprimirQualquer(ArrayList<
42 ArrayList<Integer[]>> listaVizinhos) {
43         System.out.println("Impressao da lista de vizinhos
44 ");
45
46         for(int i=0; i<listaVizinhos.size(); i++) {
47             System.out.println("Os pontos vizinhos do neuronio
48 : " + (i+1) + " são: ");
49             for(int j=0; j<listaVizinhos.get(i).size(); j

```

```
44 ++) {
45         System.out.println("x: " + listaVizinhos.
        get(i).get(j)[0] + ", Y: " + listaVizinhos.get(i).get(j)[1
        ]);
46     }
47     System.out.println();
48 }
49 }
50
51 }
52
```

```

1 package classes;
2
3 import java.io.BufferedReader;
4 import java.io.FileReader;
5 import java.io.IOException;
6 import java.util.ArrayList;
7 import java.util.List;
8 import java.util.Scanner;
9
10 /**
11  * created by lgcaobianco on 2018-06-16
12  */
13 public class LeitorPontosEntrada {
14     private String nomeArquivo;
15     private String formato;
16     private String separadorValor;
17
18     private String getNomeArquivo() {
19         return nomeArquivo;
20     }
21
22     private String getFormato() {
23         return formato;
24     }
25
26     private String getSeparadorValor() {
27         return separadorValor;
28     }
29
30     public LeitorPontosEntrada(String nomeArquivo, String
31     formato) {
32         this.nomeArquivo = nomeArquivo;
33         this.formato = formato;
34         switch (formato) {
35             case ".csv":
36                 this.separadorValor = ",";
37                 break;
38             case ".txt":
39                 this.separadorValor = " ";
40                 break;
41             default:
42                 System.out.println("Formato ainda não
43                 suportado");
44                 System.exit(1);
45                 break;
46         }
47     }
48
49     public List<Double[]> extrairPontos() {
50         List<Double[]> matrizPontos = new ArrayList<Double

```



```

48 []>());
49     String linhaLida = "";
50     BufferedReader stream = null;
51     try {
52         stream = new BufferedReader(new FileReader(
53             getNomeArquivo() + getFormato()));
54         while ((linhaLida = stream.readLine()) != null
55             ) {
56             String[] temporario = linhaLida.split(
57                 getSeparadorValor());
58             Double[] numerosSeparados = new Double[
59                 temporario.length];
60             for (int i = 0; i < temporario.length; i++
61             ) {
62                 numerosSeparados[i] = Double.
63                 parseDouble(temporario[i]);
64             }
65             matrizPontos.add(numerosSeparados);
66         }
67     } catch (IOException e) {
68         e.printStackTrace();
69         System.exit(1);
70     } finally {
71         if (stream != null) {
72             try {
73                 stream.close();
74             } catch (IOException e) {
75                 e.printStackTrace();
76                 System.exit(1);
77             }
78         }
79     }
80     return matrizPontos;
81 }
82 }
83

```