```java
package classes;

import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.io.UnsupportedEncodingException;
import java.util.ArrayList;
import java.util.List;
import java.util.Random;

/**
 * created by lgcaobianco on 2018-05-23
 */

public class RBF2 {
    private Double[][] centroide;
    private Double[] distanciasEuclidianasParaNeuronios;
    private static int quantidadeNeuronios = 2;
    private List<Double[]> conjuntoEntrada, conjuntoOperacao;
    private List<Double[]> vetorG;
    private List<Double[]> grupoOmega1;
    private List<Double[]> grupoOmega1Anterior;
    private List<Double[]> grupoOmega2;
    private int qtdEntradas = 2;
    private double varianciaCentroide1;
    private double varianciaCentroide2;
    private Double[] W2;
    public double I2, Y2, Y2Ajustado;
    private double deltaCamadaSaida;
    private double taxaAprendizagem = 0.01;

    public Double[][] getCentroide() {
        return centroide;
    }

    public void setCentroide(Double[][] centroide) {
        this.centroide = centroide;
    }

    public List<Double[]> getConjuntoEntrada() {
        return conjuntoEntrada;
    }

    public void setConjuntoEntrada(List<Double[]> conjuntoEntrada) {
        this.conjuntoEntrada = conjuntoEntrada;
    }

    public List<Double[]> getVetorG() {
        return vetorG;
```

```
49      }
50
51      public RBF2() {
52          LeitorPontosEntrada leitor = new
    LeitorPontosEntrada(
53                  "/home/lgcaobianco/repositorios/epc-rna/
    epc7/src/base/conjunto-treinamento", ".csv");
54          conjuntoEntrada = leitor.extrairPontos();
55
56          leitor = new LeitorPontosEntrada("/home/
    lgcaobianco/repositorios/epc-rna/epc7/src/base/conjunto-
    operacao",
57                  ".csv");
58          conjuntoOperacao = leitor.extrairPontos();
59
60          centroide = new Double[quantidadeNeuronios][2];
61
62          for (int i = 0; i < qtdEntradas; i++) {
63              centroide[0][i] = conjuntoEntrada.get(2)[i];
64          }
65
66          for (int i = 0; i < qtdEntradas; i++) {
67              centroide[1][i] = conjuntoEntrada.get(3)[i];
68          }
69
70          System.out.println("Centroide inicial");
71          for (int i = 0; i < 2; i++) {
72              for (int j = 0; j < 2; j++) {
73                  System.out.print(centroide[i][j] + " ");
74              }
75              System.out.println();
76          }
77          Random random = new Random();
78          W2 = new Double[3];
79          for (int i = 0; i < W2.length; i++) {
80              W2[i] = random.nextDouble();
81          }
82
83          grupoOmega1 = new ArrayList<Double[]>();
84          grupoOmega1Anterior = new ArrayList<Double[]>();
85          grupoOmega2 = new ArrayList<Double[]>();
86          vetorG = new ArrayList<Double[]>();
87          distanciasEuclidianasParaNeuronios = new Double[
    quantidadeNeuronios];
88
89      }
90
91      public Double[] getW2() {
92          return W2;
93      }
```

```
 94
 95      public List<Double[]> getGrupoOmega1() {
 96          return grupoOmega1;
 97      }
 98
 99      public void setGrupoOmega1(List<Double[]> grupoOmega1
    ) {
100          this.grupoOmega1 = grupoOmega1;
101      }
102
103      public List<Double[]> getGrupoOmega2() {
104          return grupoOmega2;
105      }
106
107      public void setGrupoOmega2(List<Double[]> grupoOmega2
    ) {
108          this.grupoOmega2 = grupoOmega2;
109      }
110
111      public double calcularDistanciaEuclidianaAAmostra(int
     linhaAmostra, int centroideReferencia) {
112          double catetoX, catetoY, hipotenusa;
113          catetoX = Math.pow((conjuntoEntrada.get(
    linhaAmostra)[0] - centroide[centroideReferencia][0]), 2)
    ;
114          catetoY = Math.pow((conjuntoEntrada.get(
    linhaAmostra)[1] - centroide[centroideReferencia][1]), 2)
    ;
115
116          hipotenusa = Math.sqrt(catetoY + catetoX);
117          return hipotenusa;
118      }
119
120      public void atribuirEntradasAGrupoOmega() {
121          limparGruposOmegas();
122          for (int i = 0; i < conjuntoEntrada.size(); i++)
    {
123              if (conjuntoEntrada.get(i)[2].doubleValue()
    == -1) {
124                  System.out.println("Essa amostra nao pode
     ser usada.");
125                  continue;
126              } else {
127                  distanciasEuclidianasParaNeuronios[0] =
    calcularDistanciaEuclidianaAAmostra(i, 0);
128                  distanciasEuclidianasParaNeuronios[1] =
    calcularDistanciaEuclidianaAAmostra(i, 1);
129                  if (distanciasEuclidianasParaNeuronios[0]
     < distanciasEuclidianasParaNeuronios[1]) {
130                      // System.out.println("A amostra
```

```
130 : " + (i + 1) + " foi colocada no grupo 1");
131                     grupoOmega1.add(conjuntoEntrada.get(i
    ));
132                 } else {
133                     grupoOmega2.add(conjuntoEntrada.get(i
    ));
134                     // System.out.println("A amostra
    : " + (i + 1) + " foi colocada no grupo 2");
135                 }
136             }
137         }
138     }
139 
140     public void zerarCentroide() {
141         for (int i = 0; i < centroide.length; i++) {
142             for (int j = 0; j < centroide[i].length; j++)
     {
143                 centroide[i][j] = 0.0;
144             }
145         }
146     }
147 
148     public void atualizarCentroide1() {
149         double somaCoordenadasX = 0.0;
150         double somaCoordenadasY = 0.0;
151         for (int i = 0; i < grupoOmega1.size(); i++) {
152             somaCoordenadasX += grupoOmega1.get(i)[0].
    doubleValue();
153             somaCoordenadasY += grupoOmega1.get(i)[1].
    doubleValue();
154         }
155         centroide[0][0] = somaCoordenadasX / grupoOmega1.
    size();
156         centroide[0][1] = somaCoordenadasY / grupoOmega1.
    size();
157 
158     }
159 
160     public void atualizarCentroide2() {
161         double somaCoordenadasX = 0.0;
162         double somaCoordenadasY = 0.0;
163         for (int i = 0; i < grupoOmega2.size(); i++) {
164             somaCoordenadasX += grupoOmega2.get(i)[0].
    doubleValue();
165             somaCoordenadasY += grupoOmega2.get(i)[1].
    doubleValue();
166         }
167         centroide[1][0] = somaCoordenadasX / grupoOmega2.
    size();
168         centroide[1][1] = somaCoordenadasY / grupoOmega2.
```

```
168 size();
169 
170     }
171 
172     public void limparGruposOmegas() {
173         grupoOmega1.removeAll(grupoOmega1);
174         grupoOmega2.removeAll(grupoOmega2);
175     }
176 
177     public void salvarGrupoOmega1Anterior() {
178         grupoOmega1Anterior.removeAll(grupoOmega1Anterior
    );
179         for (int i = 0; i < grupoOmega1.size(); i++) {
180             grupoOmega1Anterior.add(i, grupoOmega1.get(i)
    );
181         }
182     }
183 
184     public void encontrarCentroides() {
185         while (!grupoOmega1.equals(grupoOmega1Anterior))
    {
186             atualizarCentroide1();
187             atualizarCentroide2();
188             salvarGrupoOmega1Anterior();
189             atribuirEntradasAGrupoOmega();
190         }
191     }
192 
193     public void calcularVarianciaCentroide1() {
194         double variancia, somatorio = 0.0, catetoX,
    catetoY;
195         for (int i = 0; i < grupoOmega1.size(); i++) {
196             catetoX = Math.pow((centroide[0][0] -
    grupoOmega1.get(i)[0].doubleValue()), 2);
197             catetoY = Math.pow((centroide[0][1] -
    grupoOmega1.get(i)[1].doubleValue()), 2);
198             somatorio += (catetoX + catetoY);
199         }
200 
201         variancia = somatorio / grupoOmega1.size();
202         this.varianciaCentroide1 = variancia;
203         System.out.println("Variancia da centroide 1: " +
     variancia);
204     }
205 
206     public void calcularVarianciaCentroide2() {
207         double variancia, somatorio = 0.0, catetoX,
    catetoY;
208         for (int i = 0; i < grupoOmega2.size(); i++) {
209             catetoX = Math.pow((centroide[1][0] -
```

```
209 grupoOmega2.get(i)[0].doubleValue()), 2);
210             catetoY = Math.pow((centroide[1][1] -
    grupoOmega2.get(i)[1].doubleValue()), 2);
211             somatorio += (catetoX + catetoY);
212         }
213 
214         variancia = somatorio / grupoOmega2.size();
215         this.varianciaCentroide2 = variancia;
216         System.out.println("Variancia da centroide 2: " +
     variancia);
217     }
218 
219     public void primeiroEstagioTreinamento() {
220         atribuirEntradasAGrupoOmega();
221         encontrarCentroides();
222         calcularVarianciaCentroide1();
223         calcularVarianciaCentroide2();
224     }
225 
226     public void obterG(int linhaConjuntoEntrada) {
227         double distanciaCentroide1 = 0.0,
    distanciaCentroide2 = 0.0, g1, g2;
228         for (int i = 0; i < qtdEntradas; i++) {
229             distanciaCentroide1 += Math.pow((
    conjuntoEntrada.get(linhaConjuntoEntrada)[i] - centroide[
    0][i]), 2);
230         }
231 
232         double expoente = distanciaCentroide1 / (2 *
    varianciaCentroide1);
233         g1 = Math.exp(-expoente);
234         for (int i = 0; i < qtdEntradas; i++) {
235             distanciaCentroide2 += Math.pow((
    conjuntoEntrada.get(linhaConjuntoEntrada)[i] - centroide[
    1][i]), 2);
236         }
237         g2 = Math.exp(-(distanciaCentroide1) / (2 *
    varianciaCentroide2));
238 
239         System.out.println("g1: " + g1);
240         System.out.println("g2: " + g2);
241         Double[] auxiliar = new Double[2];
242         auxiliar[0] = g1;
243         auxiliar[1] = g2;
244 
245         vetorG.add(linhaConjuntoEntrada, auxiliar);
246     }
247 
248     public void obterI2(int linhaMatriz) {
249         I2 = 0.0;
```

```
250         I2 = W2[0] * -1.0;
251         for (int i = 0; i < vetorG.get(linhaMatriz).
    length; i++) {
252             I2 += vetorG.get(linhaMatriz)[i] * W2[i + 1];
253         }
254     }
255 
256     public void obterY2() {
257         Y2 = I2;
258     }
259 
260     public double obterY2Ajustado(double I2) {
261         if (Y2 >= 0) {
262             Y2Ajustado = 1;
263         } else {
264             Y2Ajustado = -1;
265         }
266         return Y2Ajustado;
267     }
268 
269     public void imprimirQualquer(List<Double[]> objeto) {
270         for (int i = 0; i < objeto.size(); i++) {
271             for (int j = 0; j < objeto.get(0).length; j++
    ) {
272                 System.out.print(objeto.get(i)[j] + ", ")
    ;
273             }
274             System.out.println();
275         }
276         System.out.println("\n\n");
277     }
278 
279     public void obterDeltaCamadaSaida(int
    linhaEntradaMatriz) {
280         deltaCamadaSaida = conjuntoEntrada.get(
    linhaEntradaMatriz)[2] - Y2;
281     }
282 
283     public void ajustarPesosCamadaSaida() {
284         for (int i = 0; i < W2.length; i++) {
285             W2[i] = W2[i] + taxaAprendizagem *
    deltaCamadaSaida * Y2;
286         }
287     }
288 
289     public double obterEk(int linhaMatrizEntrada) {
290         double ek = 0.0;
291         obterI2(linhaMatrizEntrada);
292         obterY2();
293         ek = Math.pow((conjuntoEntrada.get(
```

```
293 linhaMatrizEntrada)[2] - Y2), 2);
294         return ek / 2;
295     }
296 
297     public double obterEm() {
298         double somatorio = 0.0;
299         for (int i = 0; i < conjuntoEntrada.size(); i++)
    {
300             somatorio += obterEk(i);
301         }
302         return somatorio / conjuntoEntrada.size();
303     }
304 
305     public void propagation(int i) {
306         obterI2(i);
307         obterY2();
308         obterDeltaCamadaSaida(i);
309         ajustarPesosCamadaSaida();
310 
311     }
312 
313     public void resetarG() {
314         vetorG.clear();
315     }
316 
317     public void faseOperacao() {
318         resetarG();
319         System.out.println("Vetor G: ");
320         imprimirQualquer(vetorG);
321 
322         setConjuntoEntrada(conjuntoOperacao);
323         for (int i = 0; i < conjuntoEntrada.size(); i++)
    {
324             obterG(i);
325             obterI2(i);
326             obterY2();
327             obterY2Ajustado(I2);
328             System.out.println("Y2:  " + Y2 + ", Y2
    Ajustado: " + Y2Ajustado);
329         }
330     }
331 
332     public void imprimirQualquer(Double[] objeto) {
333         for (int i = 0; i < objeto.length; i++) {
334             System.out.print(objeto[i] + ", ");
335         }
336         System.out.println("\n\n");
337     }
338 
339     public void imprimirQualquer(Double[][] objeto) {
```

```
340         for (int i = 0; i < objeto.length; i++) {
341             for (int j = 0; j < objeto[i].length; j++) {
342                 System.out.print(objeto[i][j] + ", ");
343             }
344             System.out.println();
345         }
346         System.out.println("\n\n");
347     }
348 
349     public void imprimirGrupoCentroide1() throws
    FileNotFoundException, UnsupportedEncodingException {
350         PrintWriter writer = new PrintWriter("/home/
    lgcaobianco/repositorios/epc-rna/epc7/src/base/
    grupoCentroide1.csv",
351                 "UTF-8");
352         for (int i = 0; i < grupoOmega1.size(); i++) {
353             writer.println(i + "," + grupoOmega1.get(i)[0
    ] + "," + grupoOmega1.get(i)[1]);
354         }
355         writer.close();
356     }
357 
358     public void imprimirGrupoCentroide2() throws
    FileNotFoundException, UnsupportedEncodingException {
359         PrintWriter writer = new PrintWriter("/home/
    lgcaobianco/repositorios/epc-rna/epc7/src/base/
    grupoCentroide2.csv",
360                 "UTF-8");
361         for (int i = 0; i < grupoOmega2.size(); i++) {
362             writer.println(i + "," + grupoOmega2.get(i)[0
    ] + "," + grupoOmega2.get(i)[1]);
363         }
364         writer.close();
365     }
366 }
367 
```

```java
package classes;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

/**
 * created by lgcaobianco on 2018-05-23
 */

public class LeitorPontosEntrada {
    private String nomeArquivo;
    private String formato;
    private String separadorValor;

    private String getNomeArquivo() {
        return nomeArquivo;
    }

    private String getFormato() {
        return formato;
    }

    private String getSeparadorValor() {
        return separadorValor;
    }

    public LeitorPontosEntrada(String nomeArquivo, String formato) {
        this.nomeArquivo = nomeArquivo;
        this.formato = formato;
        switch (formato) {
        case ".csv":
            this.separadorValor = ",";
            break;
        case ".txt":
            this.separadorValor = " ";
            break;
        default:
            System.out.println("Formato ainda não suportado");
            System.exit(1);
            break;
        }
    }

    public List<Double[]> extrairPontos() {
```

```
49          List<Double[]> matrizPontos = new ArrayList<Double
    []>();
50          String linhaLida = "";
51          BufferedReader stream = null;
52          try {
53              stream = new BufferedReader(new FileReader(
    getNomeArquivo() + getFormato()));
54              while ((linhaLida = stream.readLine()) != null
    ) {
55                  String[] temporario = linhaLida.split(
    getSeparadorValor());
56                  Double[] numerosSeparados = new Double[
    temporario.length];
57                  for (int i = 0; i < temporario.length; i++
    ) {
58                      numerosSeparados[i] = Double.
    parseDouble(temporario[i]);
59                  }
60                  matrizPontos.add(numerosSeparados);
61              }
62
63          } catch (IOException e) {
64              e.printStackTrace();
65              System.exit(1);
66          } finally {
67              if (stream != null) {
68                  try {
69                      stream.close();
70                  } catch (IOException e) {
71                      e.printStackTrace();
72                      System.exit(1);
73                  }
74              }
75          }
76          return matrizPontos;
77      }
78
79      public int lerInputTerminal() {
80          Scanner reader = new Scanner(System.in); //
    Reading from System.in
81          return reader.nextInt(); // Scans the next token
    of the input as an int.
82      }
83  }
84
```