```java
package classes;

import java.util.ArrayList;
import java.util.List;
import java.util.Random;

/**
 * created by lgcaobianco on 2018-05-26
 */

public class RBFDinamica {
    private Double[][] centroide;
    private Double[] distanciasEuclidianasParaNeuronios;
    private Double[] W2;
    private static final int quantidadeNeuronios = 15;
    private static final int quantidadeEntradas = 3;
    private List<Double[]> conjuntoEntrada, conjuntoOperacao;
    private List<Double[]> vetorG;
    private ArrayList<ArrayList<Double[]>> grupoOmega;
    private ArrayList<ArrayList<Double[]>> grupoOmegaAnterior;
    private List<Double> varianciasCentroides;
    private double I2, Y2, Y2Ajustado;
    private double deltaCamadaSaida;
    private double taxaAprendizagem = 0.01;

    public ArrayList<ArrayList<Double[]>> getGruposOmegas() {
        return grupoOmega;
    }

    public void setGruposOmegas(ArrayList<ArrayList<Double[]>> gruposOmegas) {
        this.grupoOmega = gruposOmegas;
    }

    public ArrayList<ArrayList<Double[]>> getGrupoOmegaAnterior() {
        return grupoOmegaAnterior;
    }

    public void setGrupoOmegaAnterior(ArrayList<ArrayList<Double[]>> grupoOmegaAnterior) {
        this.grupoOmegaAnterior = grupoOmegaAnterior;
    }

    public List<Double> getVarianciasCentroides() {
        return varianciasCentroides;
    }
```

```
45
46      public void setVarianciasCentroides(List<Double>
    varianciasCentroides) {
47          this.varianciasCentroides = varianciasCentroides;
48      }
49
50      public Double[][] getCentroide() {
51          return centroide;
52      }
53
54      public void setCentroide(Double[][] centroide) {
55          this.centroide = centroide;
56      }
57
58      public List<Double[]> getConjuntoEntrada() {
59          return conjuntoEntrada;
60      }
61
62      public void setConjuntoEntrada(List<Double[]>
    conjuntoEntrada) {
63          this.conjuntoEntrada = conjuntoEntrada;
64      }
65
66      public List<Double[]> getVetorG() {
67          return vetorG;
68      }
69
70      public Double[] getW2() {
71          return W2;
72      }
73
74      public RBFDinamica() {
75          LeitorPontosEntrada leitor = new
    LeitorPontosEntrada(
76                  "/home/lgcaobianco/repositorios/epc-rna/
    epc8/src/base/conjunto-treinamento", ".csv");
77          conjuntoEntrada = leitor.extrairPontos();
78
79          leitor = new LeitorPontosEntrada("/home/
    lgcaobianco/repositorios/epc-rna/epc8/src/base/conjunto-
    operacao",
80                  ".csv");
81          conjuntoOperacao = leitor.extrairPontos();
82
83          centroide = new Double[quantidadeNeuronios][
    quantidadeEntradas];
84
85          for (int i = 0; i < quantidadeNeuronios; i++) {
86              for (int j = 0; j < quantidadeEntradas; j++) {
87                  centroide[i][j] = conjuntoEntrada.get(i)[j
```

```
 87 ];
 88             }
 89         }
 90 
 91         System.out.println("Centroide inicial");
 92         for (int i = 0; i < quantidadeNeuronios; i++) {
 93             for (int j = 0; j < quantidadeEntradas; j++)
    {
 94                 System.out.print(centroide[i][j] + " ");
 95             }
 96             System.out.println();
 97         }
 98         Random random = new Random();
 99         W2 = new Double[quantidadeNeuronios+1];
100         for (int i = 0; i < W2.length; i++) {
101             W2[i] = random.nextDouble();
102         }
103 
104         grupoOmega = new ArrayList<ArrayList<Double[]>>()
    ;
105         for (int i = 0; i < quantidadeNeuronios; i++) {
106             grupoOmega.add(i, new ArrayList<Double[]>());
107         }
108         grupoOmegaAnterior = new ArrayList<ArrayList<
    Double[]>>();
109         for (int i = 0; i < quantidadeNeuronios; i++) {
110             grupoOmegaAnterior.add(i, new ArrayList<
    Double[]>());
111         }
112         vetorG = new ArrayList<Double[]>();
113         distanciasEuclidianasParaNeuronios = new Double[
    quantidadeNeuronios];
114         varianciasCentroides = new ArrayList<Double>();
115 
116     }
117 
118     public double calcularDistanciaEuclidianaAAmostra(int
     linhaAmostra, int centroideReferencia) {
119         double catetoX, catetoY, catetoZ, hipotenusa;
120         catetoX = Math.pow((conjuntoEntrada.get(
    linhaAmostra)[0] - centroide[centroideReferencia][0]), 2)
    ;
121         catetoY = Math.pow((conjuntoEntrada.get(
    linhaAmostra)[1] - centroide[centroideReferencia][1]), 2)
    ;
122         catetoZ = Math.pow((conjuntoEntrada.get(
    linhaAmostra)[2] - centroide[centroideReferencia][2]), 2)
    ;
123 
124         hipotenusa = Math.sqrt(catetoY + catetoX +
```

```
124 catetoZ);
125         return hipotenusa;
126     }
127 
128     public void atribuirEntradasAGrupoOmega() {
129         limparGruposOmegas();
130         for (int i = 0; i < conjuntoEntrada.size(); i++)
    {
131             for (int j = 0; j < quantidadeNeuronios; j++)
     {
132                 distanciasEuclidianasParaNeuronios[j] =
    calcularDistanciaEuclidianaAAmostra(i, j);
133             }
134             Double menorValor;
135             int posicaoCentroide = 0;
136             menorValor =
    distanciasEuclidianasParaNeuronios[0];
137             for (int j = 1; j <
    distanciasEuclidianasParaNeuronios.length; j++) {
138                 if (menorValor >
    distanciasEuclidianasParaNeuronios[j]) {
139                     menorValor =
    distanciasEuclidianasParaNeuronios[j];
140                     posicaoCentroide = j;
141                 }
142             }
143             ArrayList<Double[]> aux = new ArrayList<
    Double[]>();
144             aux = grupoOmega.get(posicaoCentroide);
145             aux.add(conjuntoEntrada.get(i));
146             grupoOmega.set(posicaoCentroide, aux);
147 
148         }
149     }
150 
151     public void zerarCentroide() {
152         for (int i = 0; i < centroide.length; i++) {
153             for (int j = 0; j < centroide[i].length; j++)
     {
154                 centroide[i][j] = 0.0;
155             }
156         }
157     }
158 
159     public void atualizarCentroides() {
160         double somaCoordenadaX = 0, somaCoordenadaY = 0,
    somaCoordenadaZ = 0;
161         for (int i = 0; i < grupoOmega.size(); i++) {
162             for (int j = 0; j < grupoOmega.get(i).size();
     j++) {
```

```
163                     somaCoordenadaX += grupoOmega.get(i).get(
    j)[0];
164                     somaCoordenadaY += grupoOmega.get(i).get(
    j)[1];
165                     somaCoordenadaZ += grupoOmega.get(i).get(
    j)[2];
166                 }
167                 centroide[i][0] = somaCoordenadaX /
    grupoOmega.get(i).size();
168                 centroide[i][1] = somaCoordenadaY /
    grupoOmega.get(i).size();
169                 centroide[i][2] = somaCoordenadaZ /
    grupoOmega.get(i).size();
170 
171                 somaCoordenadaX = 0.0;
172                 somaCoordenadaY = 0.0;
173                 somaCoordenadaZ = 0.0;
174             }
175         }
176 
177         public void limparGruposOmegas() {
178             for (int i = 0; i < grupoOmega.size(); i++) {
179                 grupoOmega.get(i).removeAll(grupoOmega.get(i)
    );
180             }
181         }
182 
183         public void salvarGrupoOmegaAnterior() {
184             for (int i = 0; i < grupoOmegaAnterior.size(); i
    ++) {
185                 grupoOmegaAnterior.get(i).removeAll(
    grupoOmegaAnterior.get(i));
186             }
187             for (int i = 0; i < grupoOmega.size(); i++) {
188                 for (int j = 0; j < grupoOmega.get(i).size();
     j++) {
189                     grupoOmegaAnterior.get(i).add(grupoOmega.
    get(i).get(j));
190                 }
191             }
192         }
193 
194         public boolean compararGruposOmegas() {
195             for (int i = 0; i < grupoOmega.size(); i++) {
196                 for (int j = 0; j < grupoOmega.get(i).size();
     j++) {
197                     for (int k = 0; k < grupoOmega.get(i).get
    (j).length; k++) {
198                         if (grupoOmega.get(i).get(j)[k].
    doubleValue() != grupoOmegaAnterior.get(i).get(j)[k]
```

```
199                                 .doubleValue()) {
200                             return false;
201                         }
202                     }
203                 }
204             }
205             return true;
206         }
207
208         public void encontrarCentroides() {
209             int loops = 0;
210             salvarGrupoOmegaAnterior();
211             atualizarCentroides();
212             limparGruposOmegas();
213             atribuirEntradasAGrupoOmega();
214             while (!compararGruposOmegas()) {
215                 salvarGrupoOmegaAnterior();
216                 atualizarCentroides();
217                 limparGruposOmegas();
218                 atribuirEntradasAGrupoOmega();
219
220             }
221             System.out.println((loops + 1) + " Loops
    executados");
222         }
223
224         public void primeiroEstagioTreinamento() {
225             atribuirEntradasAGrupoOmega();
226             encontrarCentroides();
227             calcularVarianciaCentroides();
228         }
229
230         public void calcularVarianciaCentroides() {
231             double variancia, somatorio = 0.0, catetoX,
    catetoY, catetoZ;
232             for (int i = 0; i < grupoOmega.size(); i++) {
233                 for (int j = 0; j < grupoOmega.get(i).size();
     j++) {
234                     catetoX = Math.pow((centroide[i][0] -
    grupoOmega.get(i).get(j)[0]), 2);
235                     catetoY = Math.pow((centroide[i][1] -
    grupoOmega.get(i).get(j)[1]), 2);
236                     catetoZ = Math.pow((centroide[i][2] -
    grupoOmega.get(i).get(j)[2]), 2);
237
238                     somatorio += catetoX + catetoY + catetoZ;
239                 }
240                 variancia = somatorio / grupoOmega.get(i).
    size();
241                 somatorio = 0.0;
```

```
242             varianciasCentroides.add(variancia);
243 
244             System.out.println("A variancia da centroide
    : " + i + "foi: " + variancia);
245         }
246     }
247 
248     public void obterG(int linhaConjuntoEntrada) {
249         double somatorio = 0.0;
250         Double[] vetorAuxiliar = new Double[1];
251         for (int i = 0; i < centroide.length; i++) {
252             for (int j = 0; j < centroide[i].length; j++)
     {
253                 somatorio += Math.pow((conjuntoEntrada.
    get(linhaConjuntoEntrada)[j] - centroide[i][j]), 2);
254             }
255             double expoente = somatorio / (2 *
    varianciasCentroides.get(i));
256             vetorAuxiliar[0] = Math.exp((-1 * expoente));
257             vetorG.add(vetorAuxiliar);
258             somatorio = 0.0;
259         }
260     }
261 
262     public void obterI2(int linhaPseudoElementos) {
263         I2 = 0.0;
264         I2 = W2[0] * -1;
265         for (int i = 1; i < W2.length; i++) {
266             I2 += vetorG.get(linhaPseudoElementos)[0] *
    W2[i];
267         }
268     }
269 
270     public void obterY2() {
271         Y2 = I2;
272     }
273 
274     public void obterDeltaCamadaSaida(int
    linhaMatrizEntrada) {
275         deltaCamadaSaida = conjuntoEntrada.get(
    linhaMatrizEntrada)[quantidadeEntradas] - Y2;
276     }
277 
278     public void ajustarPesosCamadaSaida() {
279         for (int i = 0; i < W2.length; i++) {
280             W2[i] = W2[i] + (taxaAprendizagem *
    deltaCamadaSaida * Y2);
281         }
282     }
283 
```

```
284     public double obterEk(int linhaMatrizEntrada) {
285         double ek = 0.0;
286         obterI2(linhaMatrizEntrada);
287         obterY2();
288         ek = Math.pow((conjuntoEntrada.get(
    linhaMatrizEntrada)[quantidadeEntradas] - Y2), 2);
289         return ek / 2;
290     }
291 
292     public double obterEm() {
293         double somatorio = 0.0;
294         for (int i = 0; i < conjuntoEntrada.size(); i++)
    {
295             somatorio += obterEk(i);
296         }
297         return (somatorio / conjuntoEntrada.size());
298     }
299 
300     public void propagation(int i) {
301         obterI2(i);
302         obterY2();
303         obterDeltaCamadaSaida(i);
304         ajustarPesosCamadaSaida();
305     }
306 
307     public void resetarG() {
308         vetorG.clear();
309     }
310 
311     public void faseOperacao() {
312         resetarG();
313 
314         setConjuntoEntrada(conjuntoOperacao);
315 
316         for (int i = 0; i < conjuntoEntrada.size(); i++)
    {
317             obterG(i);
318             obterI2(i);
319             obterY2();
320             System.out.println("Y: " + Y2);
321         }
322     }
323 
324     public void imprimirQualquer(Double[][] objeto) {
325         for (int i = 0; i < objeto.length; i++) {
326             for (int j = 0; j < objeto[i].length; j++) {
327                 System.out.print(objeto[i][j] + ", ");
328             }
329             System.out.println();
330         }
```

```
331         System.out.println("\n\n");
332     }
333 
334     public void imprimirGrupoOmega() {
335         System.out.println("Impressao de grupos omegas.")
    ;
336 
337         for (int i = 0; i < grupoOmega.size(); i++) {
338             System.out.println("No grupo omega:" + i + "
    temos: ");
339             for (int j = 0; j < grupoOmega.get(i).size();
     j++) {
340                 for (int k = 0; k < grupoOmega.get(i).get
    (j).length; k++) {
341                     System.out.print(grupoOmega.get(i).
    get(j)[k] + " ");
342                 }
343                 System.out.println();
344             }
345             System.out.println("Tamanho do grupo: " +
    grupoOmega.get(i).size());
346             System.out.println("\n");
347         }
348     }
349 
350     public void imprimirGrupoOmegaAnterior() {
351         System.out.println("Impressao dos grupos omegas
    anteriores.");
352 
353         for (int i = 0; i < grupoOmegaAnterior.size(); i
    ++) {
354             System.out.println("No grupo omega anterior:"
     + i + " temos: ");
355             for (int j = 0; j < grupoOmegaAnterior.get(i)
    .size(); j++) {
356                 for (int k = 0; k < grupoOmegaAnterior.
    get(i).get(j).length; k++) {
357                     System.out.print(grupoOmegaAnterior.
    get(i).get(j)[k] + " ");
358                 }
359                 System.out.println();
360             }
361             System.out.println("\n");
362         }
363     }
364 
365     public void imprimirQualquer(Double[] object) {
366         for (int i = 0; i < object.length; i++) {
367             System.out.println(object[i] + " ");
368         }
```

```
369             System.out.println("\n\n");
370         }
371 }
372
```