Luiz Gustavo Caobianco

Implementação da Rede Perceptron na Linguagem Java

Exercício Para Casa - EPC01, da disciplina Redes Neurais Artificiais. Docente: Prof. Dr. Ivan Nunes

São Carlos - SP
Data de entrega: 03 de abril de 2018

Discussão do Conjunto de Entrada

O conjunto de dados fornecido na proposta impedia a convergência do software implementado em linguagem Java. Estes dados foram testados com a toolbox do MATLAB e este por sua vez também não apresentou convergência para a solução.

É possível que o problema dado seja não linearmente separável. Entretanto, para a demonstração do método desenvolvido, bem como para o preenchimento da tabela apresentada na proposta, o conjunto de dados de entrada foi alterado, de modo que fosse possível converger.

Na tabela 1, é possível ver os valores que foram necessários serem modificados para permitir a convergência do método. Os valores da coluna *d* em vermelho foram os valores que sofreram modificações. Se no conjunto de dados original um destes valores era 1, foi necessário a mudança para -1. Caso o valor original fosse -1, a mudança feita foi para 1.

Amostra	x_1	x_2	x_3	**d**
1	-0.6508	0.1097	4.0009	-1.0000
2	-1.4492	0.8896	4.4005	-1.0000
3	2.0850	0.6876	12.0710	-1.0000
4	0.2626	1.1476	7.7985	1.0000
5	0.6418	1.0234	7.0427	1.0000
6	0.2569	0.6730	8.3265	-1.0000
7	1.1155	0.6043	7.4446	1.0000
8	0.0914	0.3399	7.0677	-1.0000
9	0.0121	0.5256	4.6316	1.0000
10	-0.0429	0.4660	5.4323	**1.0000**
11	0.4340	0.6870	8.2287	-1.0000
12	0.2735	1.0287	7.1934	1.0000
13	0.4839	0.4851	7.4850	-1.0000
14	0.4089	-0.1267	5.5019	-1.0000
15	1.4391	0.1614	8.5843	-1.0000
16	-0.9115	-0.1973	2.1962	-1.0000
17	0.3654	1.0475	7.4858	1.0000
18	0.2144	0.7515	7.1699	1.0000
19	0.2013	1.0014	6.5489	1.0000
20	0.6483	0.2183	5.8991	**1.0000**
21	-0.1147	0.2242	7.2435	-1.0000
22	-0.7970	0.8795	3.8762	1.0000
23	-1.0625	0.6366	2.4707	1.0000
24	0.5307	0.1285	5.6883	**1.0000**
25	-1.2200	0.7777	1.7252	1.0000
26	0.3957	0.1076	5.6623	-1.0000
27	-0.1013	0.5989	7.1812	-1.0000
28	2.4482	0.9455	11.2095	1.0000
29	2.0149	0.6192	10.9263	-1.0000
30	0.2012	0.2611	5.4631	**1.0000**

Tabela 1: Conjunto de dados de entrada modificado.

Fase de Treinamento

Com o processo discutido na seção 1, o software foi capaz de convergir e encontrar os resultados mostrados na tabela 2. Os números iniciais foram gerados novamente para cada treinameto.

Treinamento	Vetor de Pesos Inicial				Vetor de Pesos Final				Núm. de Épocas
	w_0	w_1	w_2	w_3	w_0	w_1	w_2	w_3	
1	-1	0,0229	0,8063	0,8226	1,4835	4,944	-0,5562	54,0226	2660
2	-1	0,3129	0,3188	0,5744	1,4865	4,9359	-0,5564	52,8744	2615
3	-1	0,4178	0,2955	0,4094	1,4864	4,9298	-0,5558	52,5294	2606
4	-1	0,6045	0,588	0,5941	1,5058	4,9525	-0,561	52,0541	2573
5	-1	0,9607	0,775	0,5382	1,5022	4,9437	-0,56	50,8182	2514

Tabela 2: Tabela em LaTeX gerada automaticamente pelo software em Java

	Conjunto de entrada			Classificação				
Amostra	x_1	x_2	x_3	T1	T2	T3	T4	T5
1	-0.3565	0.0620	5.9891	-1	-1	-1	-1	-1
2	-0.7842	1.1267	5.5912	1	1	1	1	1
3	0.3012	0.5611	5.8234	-1	-1	-1	-1	-1
4	0.7757	1.0648	8.0677	1	1	1	1	1
5	0.1570	0.8028	6.3040	1	1	1	1	1
6	-0.7014	1.0316	3.6005	1	1	1	1	1
7	0.3748	0.1536	6.1537	-1	-1	-1	-1	-1
8	-0.6920	0.9404	4.4058	1	1	1	1	1
9	-1.3970	0.7141	4.9263	-1	-1	-1	-1	-1
10	-1.8842	-0.2805	1.2548	-1	-1	-1	-1	-1

Tabela 3: Resposta do Software

Classificação de Novas Amostras

Através dos treinamentos mostrados na seção 2, novas amostras foram expostas ao software, que apresentou a classificação conforme mostrado na tabela 3.

Variação do Número de Épocas

Os vetores de pesos iniciais são gerados aleatoriamente, e conforme a rede é treinada, estes valores convergem na direção da solução correta, desde que o conjunto de entrada seja linearmente separável. Isto significa que, uma vez que sorteados, estes valores são alterados até a solução do problema.

A variação do número de épocas deve-se exatamente ao sorteio inicial. Se, por coincidência, os números gerados aleatoriamente forem próximos da solução inicial, será necessário uma quantidade menor de épocas para encontrar a solução do sistema.

Analogamente, se os números sorteados forem distantes dos números que representam uma solução para o problema, será necessário uma quantidade maior de épocas para que seja encontrado o resultado.

A limitação da Perceptron

A principal limitação deve-se ao fato que só é possível resolver problemas linearmente separáveis. Isso significa que, conforme aumentamos a quantidade de entradas, a solução apresentada pela Perceptron perde a precisão.

Por exemeplo, com apenas duas entradas, um problema linearmente separável pode ser isolado por uma simples reta, e neste caso a Perceptron apresenta ótimo desempenho. Entretanto, caso seja necessário utilizar 3, 4, ou mais entradas, a solução apresentada pela Perceptron não será tão precisa quanto uma rede Perceptron de duas entradas.

Anexos

Nesta seção, os códigos em Java da implementação feita são demonstrados. Os códigos foram impressos através da funcionalidade embutida de impressão de código da IDE IntelliJ Ultimate, registrado no nome do autor deste trabalho.

```java
package classes;

import java.io.*;
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.List;
import java.util.Random;

/**
 * * created by lgcaobianco on 21/03/18 **
 */
public class Perceptron {
    List<double[]> matrizPontos = new ArrayList<double[]>();
    List<double[]> conjuntoTeste = new ArrayList<double[]>();
    private double[][] matrizPesosInicial = new double[4][1];
    private double[][] matrizPesosFinal = new double[4][1];
    private int contadorEpocas;


    public void imprimirmatrizPesosInicial() {
        for (int i = 0; i < this.matrizPesosInicial.length; i++) {
            for (int j = 0; j < this.matrizPesosInicial[i].length; j++) {
                System.out.print(this.matrizPesosInicial[i][j] + " ");
            }
            System.out.println();
        }
    }

    public void imprimirMatrizPontos() {
        for (int i = 0; i < this.matrizPontos.size(); i++) {
            for (int j = 0; j < this.matrizPontos.get(i).length; j++) {
                System.out.print(this.matrizPontos.get(i)[j] + " ");
            }
            System.out.println();
        }
    }

    public void construirMatrizPontos(String nomeArquivo, String extensaoArquivo) {
        LeInformacoes informacoes = new LeInformacoes(
```

```
39 nomeArquivo, extensaoArquivo);
40         this.matrizPontos = informacoes.extrairPontos();
41     }
42 
43     public void construirConjuntoTeste(String nomeArquivo,
    String extensaoArquivo) {
44         LeInformacoes informacoes = new LeInformacoes(
   nomeArquivo, extensaoArquivo);
45         this.conjuntoTeste = informacoes.extrairPontos();
46     }
47 
48     public void construirmatrizPesosInicial() {
49         matrizPesosInicial[0][0] = -1;
50         for (int i = 1; i < this.matrizPesosInicial.length
   ; i++) {
51             for (int j = 0; j < this.matrizPesosInicial[i]
   .length; j++) {
52                 Random r = new Random();
53                 this.matrizPesosInicial[i][j] = r.
   nextDouble();
54                 this.matrizPesosFinal[i][j] = this.
   matrizPesosInicial[i][j];
55             }
56         }
57 
58     }
59 
60     public int ativacao(double somatorio) {
61         int classificacao;//passar o somatório pelo g(u)
62         if (somatorio >= 0) {
63             classificacao = 1;
64         } else {
65             classificacao = -1;
66         }
67         return classificacao;
68     }
69 
70 
71     public void treinarPerceptron() {
72         double somatorio, taxaAprendizagem = 0.01,
   classificacao;
73         int erro, iMaximo = 0;
74 
75         percorreLinhasConjuntoTreinamento:
76         for (int i = 0; i < this.matrizPontos.size(); ) {
77             somatorio = 0;
78             for (int j = 0; j < this.matrizPontos.get(i).
   length - 1; j++) {
79                 somatorio += (this.matrizPontos.get(i)[j]
   * this.matrizPesosFinal[j][0]);
```

```
 80              }
 81
 82              classificacao = ativacao(somatorio);
 83
 84              if (classificacao == this.matrizPontos.get(i)
    [3]) {
 85                  i++;
 86
 87              } else { //se classificacao nao coincide com
    d_i, ajustar coeficientes
 88                  erro = (int) (this.matrizPontos.get(i)[3]
     - classificacao);
 89                  for (int j = 0; j < this.
    matrizPesosInicial.length; j++) {
 90                      this.matrizPesosFinal[j][0] += (
    taxaAprendizagem * erro * this.matrizPontos.get(i)[j]);
 91                  }
 92                  this.contadorEpocas++;
 93                  i = 0;
 94              }
 95
 96
 97          }
 98
 99
100      }
101
102
103      public void classificarVetores() {
104          double somatorio;
105
106          for (int i = 0; i < this.conjuntoTeste.size(); i
    ++) {
107              somatorio = 0;
108              for (int j = 0; j < this.conjuntoTeste.get(i)
    .length; j++) {
109                  somatorio += this.conjuntoTeste.get(i)[j]
     * this.matrizPesosFinal[j][0];
110              }
111              int classificacao = ativacao(somatorio);
112              System.out.println("A linha: " + (i + 1) + "
    foi classificada como: " + classificacao);
113
114          }
115      }
116
117      public static void iniciarTabelaLatex() {
118          try (Writer writer = new BufferedWriter(new
    OutputStreamWriter(
119                  new FileOutputStream("matrizSaida.tex"),
```

```
119 "utf-8"))) {
120             writer.write("\\documentclass{article}" +
    System.getProperty("line.separator"));
121             writer.write("\\usepackage{multirow}" +
    System.getProperty("line.separator"));
122             writer.write("\\usepackage[utf8]{inputenc}" +
     System.getProperty("line.separator"));
123             writer.write("\\usepackage[bottom]{footmisc}"
     + System.getProperty("line.separator"));
124             writer.write("\\usepackage{lscape}" + System.
    getProperty("line.separator"));
125             writer.write("\\renewcommand\\tablename{
    Tabela}" + System.getProperty("line.separator"));
126             writer.write("\\begin{document}" + System.
    getProperty("line.separator"));
127             writer.write("\\begin{landscape}" + System.
    getProperty("line.separator"));
128             writer.write("\\begin{table}" + System.
    getProperty("line.separator"));
129             writer.write("\\centering" + System.
    getProperty("line.separator"));
130             writer.write("\\begin{tabular}{lllllllll}" +
     System.getProperty("line.separator"));
131             writer.write("\\hline" + System.getProperty("
    line.separator"));
132             writer.write("\\multicolumn{1}{|c|}{\\
    multirow{2}{*}{Treinamento}} & \\multicolumn{4}{l|}{Vetor
     de Pesos Inicial} " +
133                     "& \\multicolumn{4}{l|}{Vetor de
    Pesos Final} & \\multicolumn{1}{l|}{\\multirow{2}{*}{Núm
    . de Épocas}} \\\\ " +
134                     "\\cline{2-9}\n \\multicolumn{1}{|c
    |}{} & \\multicolumn{1}{l|}{w_0} & \\multicolumn{1}{l|}{
    w_1} & " +
135                     "\\multicolumn{1}{l|}{w_2} & \\
    multicolumn{1}{l|}{w_3} & \\multicolumn{1}{l|}{w_0} & " +
136                     "\\multicolumn{1}{l|}{w_1} & \\
    multicolumn{1}{l|}{w_2} & \\multicolumn{1}{l|}{w_3} & " +
137                     "\\multicolumn{1}{l|}{} \\\\ \\hline"
    );
138
139         } catch (IOException e) {
140             e.printStackTrace();
141         }
142     }
143
144     public void preencherTabelaLatex(Perceptron p, int
    contaQuantidadeTreinamentos) {
145         try (Writer writer = new FileWriter("matrizSaida.
    tex", true)) {
```

```
146             writer.append("\\multicolumn{1}{|l|}{" + (
    contaQuantidadeTreinamentos+1) + "} &");
147 
148             for (int i = 0; i < 4; i++) {
149                 writer.append("\\multicolumn{1}{|l|}{" +
    new DecimalFormat("#.####").format(p.matrizPesosInicial[i
    ][0]) + "} & ");
150             }
151 
152             for (int i = 0; i < 4; i++) {
153                 writer.append("\\multicolumn{1}{|l|}{" +
    new DecimalFormat("#.####").format(p.matrizPesosFinal[i][
    0]) + "} & ");
154             }
155 
156             writer.append("\\multicolumn{1}{l|} {" + p.
    contadorEpocas + "}");
157             writer.append("\\\\" + "\\hline" + System.
    getProperty("line.separator"));
158         } catch (IOException e) {
159             e.printStackTrace();
160         }
161     }
162 
163     public static void finalizarTabelaLatex() {
164         try (Writer writer = new FileWriter("matrizSaida.
    tex", true)) {
165                 writer.append("\\end{tabular}" + System.
    getProperty("line.separator"));
166             writer.append("\\caption{Tabela em \\LaTeX{}
    gerada automaticamente pelo software em Java}" + System.
    getProperty("line.separator"));
167             writer.append("\\end{table}" + System.
    getProperty("line.separator"));
168             writer.append("\\end{landscape}" + System.
    getProperty("line.separator"));
169             writer.append("\\end{document}" + System.
    getProperty("line.separator"));
170 
171         } catch (IOException e) {
172             e.printStackTrace();
173         }
174     }
175 
176 }
177 
178 
```

```java
package classes;
/**
 * * created by lgcaobianco on 21/03/18 **
 */

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

public class LeInformacoes {

    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  LeInformacoes(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;

        }
    }


```

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

```java
package teste;

import classes.Perceptron;

import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

/**
 * * created by lgcaobianco on 21/03/18 **
 */
public class TestarPerceptron {
    public static void main(String[] args) {
        Perceptron.iniciarTabelaLatex();
        Scanner input = new Scanner(System.in);
        System.out.println("Insira a quantidade de testes");
        int quantidadeRedes = input.nextInt();

        List<Perceptron> listaPerceptron = new ArrayList<Perceptron>();
        for(int i=0; i < quantidadeRedes; i++){
            listaPerceptron.add(new Perceptron());
            listaPerceptron.get(i)
                    .construirMatrizPontos("/home/lgcaobianco/repositorios/epc/epc1/base/valores",
                    ".csv");
            listaPerceptron.get(i).construirmatrizPesosInicial();
            listaPerceptron.get(i).treinarPerceptron();
            listaPerceptron.get(i).construirConjuntoTeste("/home/lgcaobianco/repositorios/epc/epc1/base/conjuntoTeste",
                    ".csv");
            listaPerceptron.get(i).classificarVetores();
            listaPerceptron.get(i).preencherTabelaLatex(listaPerceptron.get(i), i);
        }
        Perceptron.finalizarTabelaLatex();

    }
}
```