Faculdade de Engenharia da Universidade do Porto

Redes Neuronais para Classificação de Expressões Faciais

Relatório Intercalar

Inteligência Artificial
3º ano do Mestrado Integrado em Engenharia Informática e Computação

GRUPO E5_1
Lázaro Costa – up201405342 – up201405342@fe.up.pt
Miguel Lira – up201405324 – up201405324@fe.up.pt
Miriam Gonçalves – up201403441 – up201403441@fe.up.pt

2 de abril de 2017

1. Objetivo

Este trabalho tem como objetivo a criação de uma Rede Neuronal Artificial para a classificação de expressões faciais. É pressuposta a utilização do algoritmo Retro Propagação do Gradiente(*BackPropagation)* para criar uma solução que proceda ao treino da rede, tendo por base um conjunto de dados disponibilizado. O conjunto de dados é constituído por 100 atributos que representam uma coordenada de um ponto da face humana e por um "alvo" /*target* que representa o sucesso ou insucesso na classificação dessa expressão. A solução encontrada deve permitir a introdução dos dados da expressão facial na rede neuronal, normalizar os valores dos dados introduzidos, por sua vez, esta retornará a classificação obtida.

2. Descrição

2.1. Especificação

O desenvolvimento deste trabalho vai-se dividir em 3 partes sequenciais: uma primeira em que é necessário o processamento dos dados fornecidos(*dataset)*, de seguida, procede-se ao treino da rede neuronal e, por fim, permitir a utilização da rede neuronal para a classificação de expressões faciais com base em entradas a serem fornecidas pelo utilizador. A implementação da rede neuronal vai ser feita recorrendo-se a uma *framework, Neuroph*, que permite a simplificação do desenvolvimento da arquitetura de uma rede neuronal comum através de uma biblioteca e uma ferramenta que ajuda à sua criação, treino e aplicação.

2.1.1. Descrição e análise do *dataset*

O *dataset* fornecido para o treino da rede neuronal é composto por 36 ficheiros, 9 ficheiros contêm os dados da expressão facial relativa ao indivíduo A em estudo e outros 9 relativas às expressões faciais do indivíduo B e os restantes 18 ficheiros com o resultado ou especificação para cada expressão facial de cada indivíduo, os dados da expressão facial são compostos por 100 coordenadas faciais tridimensionais (x,y,z). O x e o y são dados em pixéis e a terceira coordenada é dada em milímetros. Deste conjunto de atributos fazem parte coordenadas dos olhos, sobrancelhas, boca, contorno da face, nariz, a ponta do nariz, as linhas acima das sobrancelhas e as íris.

Os dados disponibilizados têm valores distintos e não contínuos, tornando-se necessário proceder à normalização destes nos limites de [0,1] ou de [-1,1], evitando-se a possibilidade de levar a rede a tomar decisões erradas. Esta normalização de dados é feita recorrendo-se às funções de normalização fornecidas pela *framework Neuroph.*

2.1.2. Estrutura da rede

Uma rede neuronal é uma rede de circuitos que simula a estrutura neuronal de organismos inteligentes e que adquire conhecimento através da experiência, tal como o cérebro humano. Este tipo de redes é constituído por 3 elementos: elemento de processamento, estrutura das ligações e a lei da aprendizagem que para este projeto será o

algoritmo *backpropagation.* A informação armazenada pela rede é partilhada por todas as suas unidades de processamento(conexionismo).

Depois de sabida o número de entradas da rede estudou-se a memória ocupada pela rede: 300*nrº de neurónios + 9*nrº de neurónios, em que 300 corresponde ao número de entradas e 9 ao número de saídas.

Sendo usado como algoritmo de aprendizagem o *backpropagation,* a rede será de múltiplas camadas, a camada de entrada, intermédia e de saída. É de notar que na camada intermédia é preciso ter 3 a 5 vezes mais ligações do que variáveis. A informação circula no sentido *input -> output*. Após ser calculado o output é necessário proceder ao cálculo do erro, e consequentemente à atualização dos valores da rede no caso em que valor do gradiente do erro em relação aos respetivos valores das arestas é elevado, esta atualização é feita no sentido *output->input*.

O que se pretende é encontrar o mínimo global do erro, que é uma função que tem como *input* os valores de referência da rede neuronal e o *target* da *dataset*.

Uma vez que o *dataset* contém 100 atributos que são coordenadas (x,y,z), então a rede de reconhecimento de expressões faciais terá 300 entradas e 9 saídas, que correspondem ao número de expressões faciais diferentes estudadas na base de dados utilizada. No final será avaliado o melhor output dado pelos 9 nós de saída da rede, ou seja, o valor mais alto será escolhido como sendo a expressão facial aos quais correspondem os atributos introduzidos.

2.1.3. Normalização de dados

No *dataset* são utilizadas as seguintes caraterísticas sobre a face humana:

1. *left eye* – olho esquerdo (numérico: de 0 a 7)
2. *right eye* – olho direito (numérico: de 8 a 15)
3. *left eyebrow* – sobrancelha esquerda (numérico: de 16 a 25)
4. *right eyebrow* – sobrancelha direita (numérico: de 26 a 35)
5. *nose* – nariz (numérico: de 36 a 47)
6. *mouth* – boca (numérico: de 48 a 67)
7. *face contour* – contorno do rosto (numérico: de 68 a 86)
8. *left iris* – íris esquerda (numérico: 87)
9. *right iris* – íris direita (numérico: 88)
10. *nose tip* – ponta do nariz (numérico: 89)
11. *line above left eyebrow* – linha acima da sobrancelha esquerda (numérico: de 90 a 94)
12. *line above right eyebrow* – linha acima da sobrancelha direita (numérico: de 95 a 99)

2.1.4. Lei de Aprendizagem

Dado que se está a desenvolver uma rede neuronal de múltiplas camadas usar-se-á um tipo de aprendizagem supervisionada recorrendo-se ao algoritmo de *backpropagation,* tal como se referiu anteriormente no ponto 2.1.2.. Este método tenta minimizar a função de custo quadrático.

2.1.5. Trabalho efetuado

O trabalho até agora desenvolvido concentrou-se no estudo e na análise de dados a utilizar no desenvolvimento da rede neuronal, assim como, na estrutura e criação da rede neuronal. Todas as ideias analisadas bem como resultados e decisões, poderão vir a sofrer alterações.

A implementação da rede neuronal não vai ser feita de raiz, utilizar-se-á uma biblioteca, Neuroph, que já foi referenciada anteriormente, secção 2.1, que vai permitir implementar a rede e fazer todos os treinos necessários para que a rede tenha resultados satisfatórios.

Procedeu-se à implementação de um parser para juntar o *target* com os respetivos dados das expressões faciais e também criou-se uma função que permite gerar os ficheiros de treino e de teste de forma aleatória através dos dados dos indivíduos A e B.

```
public class DataParser {
   String a_pathDataPoint;
   String a_pathTarget;
   String b_pathDataPoint;
   String b_pathTarget;
   File TrainingFile;
   File TestFile;
public DataParser(String a_pathDataPoint, String a_pathTarget, String b_pathDataPoint,
String b_pathTarget) throws IOException {
       this.a_pathDataPoint = a_pathDataPoint;
       this.a_pathTarget = a_pathTarget;
       this.b_pathDataPoint = b_pathDataPoint;
       this.b_pathTarget = b_pathTarget;
       this.TrainingFile = null;
       this.TestFile = null;
   }
public void generateTrainingTestFiles() throws IOException {
       File a_path = addTarget(this.a_pathDataPoint, this.a_pathTarget);
       File b_path = addTarget(this.b_pathDataPoint, this.b_pathTarget);
       this.concatenateFiles(a_path, b_path);
   }
public File addTarget(String pathDataPoint, String pathTarget) throws IOException {
       File datapoint = new File(pathDataPoint);
       FileInputStream DataPointstream = new FileInputStream(datapoint);
       File targets = new File(pathTarget);
       FileInputStream Targetstream = new FileInputStream(targets);

       String[] name = datapoint.getName().split("_");
       String folderPath = new
String(Arrays.copyOfRange(datapoint.getAbsolutePath().getBytes(), 0,
```

```
                datapoint.getAbsolutePath().lastIndexOf("\\")));
        Path pathToFile = Paths.get(folderPath + "\\" + name[0] + "_" + name[1] + ".txt");
        File outPutFile = new File(String.valueOf(pathToFile));
        FileOutputStream fos = new FileOutputStream(outPutFile);

        BufferedReader datapointStream = new BufferedReader(new
InputStreamReader(DataPointstream));
        BufferedReader tagetStream = new BufferedReader(new
InputStreamReader(Targetstream));
        BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(fos));

        String labelLine = datapointStream.readLine() + ";T;";
        bw.write(labelLine);
        bw.newLine();

        String targetLine, datapointLine;
        while (((datapointLine = datapointStream.readLine()) != null) &&
                ((targetLine = tagetStream.readLine()) != null)) {
            String str = (datapointLine + ";" + targetLine + ";");
            bw.write(str);
            bw.newLine();}

        datapointStream.close();
        tagetStream.close();
        bw.close();
        return outPutFile;
    }
public void concatenateFiles(File file1, File file2) throws IOException {
        FileInputStream Expression1 = new FileInputStream(file1);
        FileInputStream Expression2 = new FileInputStream(file2);
        String[] name = file1.getName().split("_");
        String folderPath = new
String(Arrays.copyOfRange(file1.getAbsolutePath().getBytes(), 0,
                file1.getAbsolutePath().lastIndexOf("\\")));
        Path pathToTrainFile = Paths.get(folderPath + "\\" + "TRAIN_" + name[1]);
        this.TrainingFile = new File(String.valueOf(pathToTrainFile));
        FileOutputStream training = new FileOutputStream(this.TrainingFile);
        Path pathToTestFile = Paths.get(folderPath + "\\" + "TEST_" + name[1]);
        this.TrainingFile = new File(String.valueOf(pathToTestFile));
        FileOutputStream test = new FileOutputStream(this.TrainingFile);
        BufferedReader Stream1 = new BufferedReader(new InputStreamReader(Expression1));
        BufferedReader Stream2 = new BufferedReader(new InputStreamReader(Expression2));
        BufferedWriter TrainingBuffer = new BufferedWriter(new
OutputStreamWriter(training));
        BufferedWriter TestingBuffer = new BufferedWriter(new OutputStreamWriter(test));
        String labelLine1 = Stream1.readLine();
        String labelLine2 = Stream2.readLine();
        Random rn2 = new Random();
        Random rn1 = new Random();
        TrainingBuffer.write(labelLine1);
        TestingBuffer.write(labelLine2);
        TestingBuffer.newLine();
        TrainingBuffer.newLine();
        String line1, line2;
        while (((line1 = Stream2.readLine()) != null) && ((line2 = Stream2.readLine()) !=
null)) {
            int n1 = rn1.nextInt(10) + 1;
```

```
            if (n1 <= 7) {
                TrainingBuffer.write(line1);
                TrainingBuffer.newLine();
            } else {
                TestingBuffer.write(line1);
                TestingBuffer.newLine();
            }
            int n2 = rn2.nextInt(10) + 1;
            if (n2 <= 7) {
                TrainingBuffer.write(line2);
                TrainingBuffer.newLine();
            } else {
                TestingBuffer.write(line1);
                TestingBuffer.newLine();
            }
        }
        Stream1.close();
        Stream2.close();
        TestingBuffer.close();
        TrainingBuffer.close();
        file1.delete();
        file2.delete();
    }
}
```

2.1.6. Resultados esperados e forma de avaliação

Após um breve estudo e análise ao *dataset* conclui-se que numa fase inicial utilizar-se-á 70% dos dados dos indivíduos misturados aleatoriamente para treino da rede neuronal e os restantes 30% serão utilizados para testes. Com estes dados de teste será possível verificar a exatidão da rede e analisar se alguns dos valores anteriormente decididos podem ser modificados para melhorar os resultados obtidos.

3. Conclusões

Em conclusão, as redes neuronais podem desempenhar um papel importante na civilização para estudos estatísticos e tomadas de decisões. No entanto, também é uma área que ainda se encontra em desenvolvimento, pelo que algumas técnicas de implementação de redes neuronais podem vir a ser melhoradas.

É de notar que o contexto realista do desenvolvimento desta rede neuronal desperta um maior interesse por esta área e também por parte dos estudantes para esta matéria. No entanto, a falta de tempo por parte das outras unidades curriculares também não permite um estudo mais intensivo e aprofundado sobre as diferentes estruturas de redes neuronais existentes e as vantagens e desvantagens destas.

4. Recursos

4.1. Bibliografia

- http://neuroph.sourceforge.net/
- http://paginas.fe.up.pt/~eol/IA/IA0708/APONTAMENTOS/IA-NN.pdf
- https://www.mql5.com/pt/articles/497
- FREITAS, F. A.; Peres, S.M.; Lima ; C. A. M. ; BARBOSA, F. V. . *Grammatical Facial Expressions Recognition with Machine Learning. In: 27th Florida Artificial Intelligence Research Society Conference (FLAIRS), 2014, Pensacola Beach. Proceedings of the 27th Florida Artificial Intelligence Research Society Conference (FLAIRS)*. Palo Alto: The AAAI Press, 2014. p. 180-185.

4.2. Software

- *Neuroph*
- Linguagem de programação utilizada: *Java*
- IDE: *Intellij*