Redes Neuronais

Classificação de expressões faciais

Inteligência Artificial

Mestrado Integrado em Engenharia Informática e Computação

Maria João Mira Paulo - up201403820@fe.up.pt
Nuno Miguel Mendes Ramos - up201405498@fe.up.pt
Pedro Duarte da Costa - up201403291@fe.up.pt

2 de Abril de 2017

Conteúdo

1 Objetivo

Este trabalho tem por objetivo a aplicação de Redes Neuronais artificiais para classificação de expressões faciais.

A Linguagem Gestual é uma língua de sinais, uma língua que utiliza gestos, sinais e expressões faciais e corporais, ao invés de sons na comunicação. Na linguagem de sinais, a existência de expressões faciais é o que permite atingir uma expressão gramatical, caso contrário seria impossível detetar as diferenças entre uma frase afirmativa e uma frase interrogativa, por exemplo.

Existe um limite de expressões faciais que podem ser demonstradas pelos seres humanos. Por esta razão existem técnicas de reconhecimento de *Grammatical Facial Expressions*. Contudo, estas técnicas ultrapassam diversas dificuldades uma vez que uma expressão facial pode variar de pessoa para pessoa. Além disso, outra dificuldade que têm que ser superada é o reconhecimento de expressões em casos de co-ocorrência de *Grammatical Facial Expressions* (GFE), pois poderá causar conflitos de interpretação. Por esta razão de inconsistência é que se escolheu usar *Machine Learning*.

Machine Learning é o que permite a uma máquina reconhecer padrões, é o que dá aos computadores a habilidade de aprender sem serem explicitamente programados. Podendo esta depois de treinada, reconhecer e classificar expressões realizadas por outras pessoas.

Usaremos neste trabalho um método de aprendizagem supervisionada indutiva denominado *neural network Multilayer Perceptron* (MLP).

2 Descrição

2.1 Especificação

2.1.1 Descrição e Análise do Dataset

As expressões faciais são importantes no reconhecimento de expressões gramaticais. Existem nove tipos de identificadores gramaticais capazes de transformar uma simples frase numa expressão gramatical:

- Questões "WH", por exemplo *Quem*, *Onde*, *Como* ou *Porquê*
- Questões fechadas, de resposta *Sim* ou *Não*
- Questões de dúvida
- Tópico
- Negação
- Afirmação
- Cláusula Condicional
- Realce
- Cláusula Relativa

Treinar a rede implica que haja uma preparação dos dados. A base de dados é composta por um conjunto de 18 vídeos, conseguidos através do uso do *Microsoft Kinect sensor*, um sensor de movimentos. Um utilizador expressa uma determinada *Grammatical Facial Expression*, 5 vezes em cada vídeo, e são extraídos ficheiros de texto com informação relativa a cada um desses vídeos.

Estes são etiquetados por outra, um tradutor humano fluente em Libras, por forma obter uma maior imparcialidade.

Ao todo existem 27965 instâncias capturadas através desses vídeos.

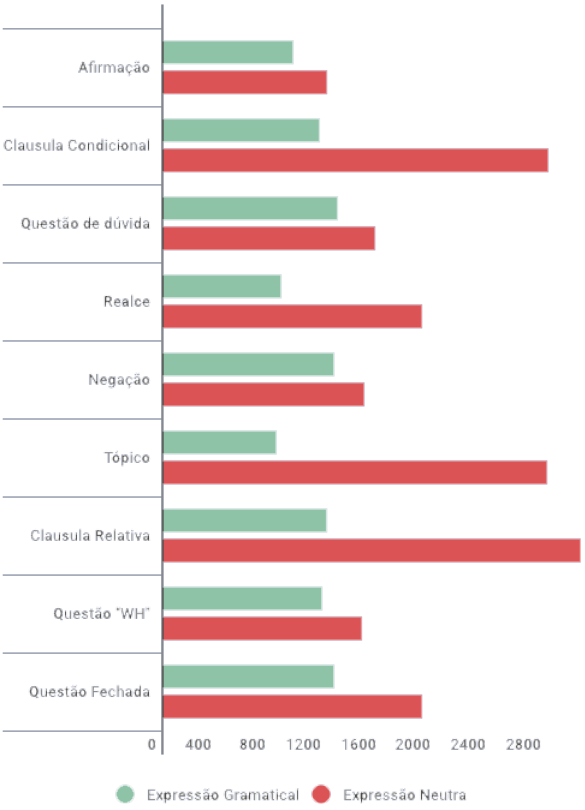
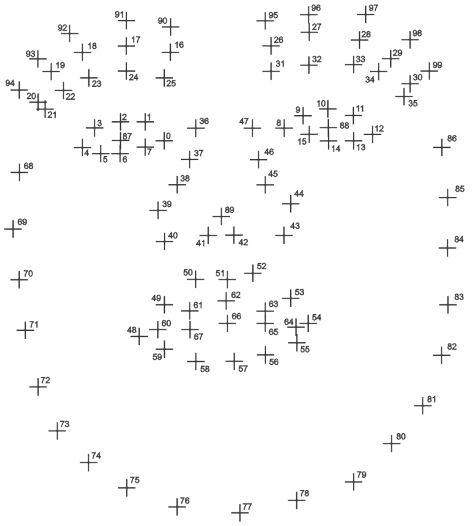


Figura 1: Instâncias por expressão facial

Esta base de dados baseia-se no pressuposto de que uma GFE é composta por uma GFE específica e por Expressões Neutras.

A base de dados usada é então constituída por 36 ficheiros, 18 *Datapoint Files* e 18 *Target Files*. A cada *Datapoint File* corresponde um *Target File*, sendo que o nome dos ficheiros contém a letra A ou a letra B, simbolizando um utilizador A e um utilizador B respetivamente.

Desta forma, um *Datapoint File* é constituído por um *FrameId* e por 100 pontos, cada um representado pela sua coordenada x e coordenada y, em pixeis e pela coordenada z, em milímetros.



Identificador da Instância
0 - 7 (x,y,z) - Olho esquerdo
8 - 15 (x,y,z) - Olho direito
16 - 25 (x,y,z) - Sobrancelha esquerda
26 - 35 (x,y,z) - Sobrancelha direita
36 - 47 (x,y,z) - Nariz
48 - 67 (x,y,z) - Boca
68 - 86 (x,y,z) - Contorno face
87 (x,y,z) - Iris olho esquerdo
88 (x,y,z) - Iris olho direito
89 (x,y,z) - Ponta Nariz
90 - 94 (x,y,z) - Linha acima da sobrancelha esquerda
95 - 99 (x,y,z) - Linha acima da sobrancelha direita

Figura 2: Imagem representativa dos pontos extraídos

Os *Target Files* atribuem a cada instância do ficheiro *Datapoint File* correspondente **1** caso seja uma GFE e **0** caso seja uma Expressão Neutra.

2.1.2 Pré Processamento de Dados

A partir dos pontos extraídos dos vídeos, pré-processamos um conjunto de medidas que representam os determinados identificadores (*features*) de uma expressão facial.

Semantic Functions	Eyebrows	Eyes	Mouth	Head
WH-question	↑			↑
Yes/no question	↑			↓
Doubt question	↓	*	*	⊖
Topic		⋄		↓
Negation	↓		∩	↔
Assertion				↕
Conditional clause	As in yes/no question			
Focus	As in topic			
Relative clause	↑			

↑ – upward head; ↓ – downward head
↕ – up and downward head; ↔ – left and rightward head
* – compressed mouth; ⋄ – open mouth; ∩ – downward mouth
⊕ – aproximation; ⊖ – detachment

Figura 3: Tabela com os identificadores associados a cada expressão

Obtivemos assim as seguintes distâncias e ângulos:

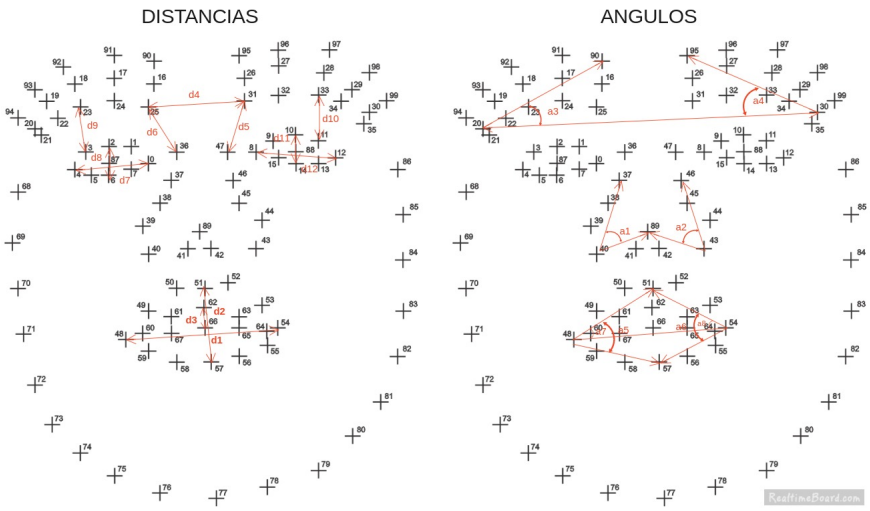


Figura 4: Representação das medidas extraídas dos pontos

Distancias:

- d1,d2 e d3 permitem verificar se a boca está aberta, fechada ou cerrada;
- d4, d5, d6, d9 e d1, permitem verificar se as sobrancelhas estão cerradas, descontraídas ou levantadas;
- d7,d8,d11,d12 correspondem a olhos abertos ou fechados.

Ângulos:

- a1 e a2 permitem verificar se a face está levantada ou baixa;
- a3 e a4 permitem verificar se as sobrancelhas estão cerradas, descontraídas ou levantadas;
- a5 e a6 verificam se a boca esta aberta ou fechada;
- a7 e a8 verificam se a boca está curvada negativamente.

Por forma a obter uma analise de movimentos de expressões, estes dados são depois organizados num vetor temporal. O vetor irá conter de forma sequencial os *frames* correspondentes a uma certa expressão. Contendo a seguinte representação: v = {medidas do frame 1, medidas do frame 2, ...}. As medidas de cada *frame* seriam as seguintes: distancias, ângulos e profundidade dos pontos. O vetor seria por exemplo, v = {d11, d12, a11, z11, z12, z13, z14, d21, d22, a21, z21, z22, z23, z24, ...}. O primeiro índice corresponde ao *frame* e o segundo índice à distancia/ângulo/profundidade.

2.1.3 Modelo de Aprendizagem a aplicar

Neste trabalho vamos utilizar uma rede neuronal, *Multilayer Perceptron* (MLP), este modelo utiliza um tipo de aprendizagem supervisionada indutiva denominado *backpropagation*.

O modelo MLP é uma rede neuronal artificial do tipo *feedforward*, ou seja, as conexões entre os neurónios não formam ciclos. A informação move num único sentido. Este modelo consiste em múltiplas camadas de neurónios, num grafo direcionado, todo ele conectado. Cada neurónio numa dada camada, tem

conexões diretas a neurónios da camada seguinte.

Aprendizagem supervisionada, é uma tarefa de *machine learning* que infere uma função a partir de um conjunto de dados de treino categorizados. Aplica-se neste projeto, pois recebe um conjunto de dados de entrada, as expressões faciais, e o correspondente valor de output desejado para cada entrada, tipo de expressão. Este tipo de aprendizagem indutiva permite, generalizar os dados de entrada e obter uma função que permite inferir , com alguma aproximação, o tipo de expressão facial através de novas expressões.

Backpropagation, é um método de treino a duas fases, propagação e depois atualização dos pesos das arestas por retrocesso. Numa primeira fase o vetor de entrada é propagado desde o inicio da rede, camada a camada, até ao chegar à ultima, à camada de saída. O resultado desta propagação é comparado com o valor desejado, usando uma função de perda, sendo calculado um valor de erro para cada um dos neurónios da camada de saída. O valor de erro é propagado por retrocesso, a partir da camada de saída, até todos os neurónios terem um valor de erro associado que representa de forma grosseira a sua contribuição em relação aos dados de entrada originais. Através destes valores de erro os pesos das arestas da rede são atualizados por forma a minimizar a função de perda.

Após receber dados de entrada suficientes a rede converge para um estado em que os erros são cada vez menores, dizendo-se que "aprendeu"um certo objetivo. Numa tentativa de ajustar corretamente os pesos das arestas, este método é usado em conjunto com o método de otimização *gradient descent*, sendo usada a derivada da função de perda, minimizando a mesma e apontando a direção da função de erro para "baixo"(*downhill*). Esta otimização recorre a uma variável *learning rate*, que corresponde ao quanto "descemos a colina"a cada iteração.

2.1.4 Arquitetura das redes neuronais

Os neurónios são organizados em redes de neurónios, conectados num grafo acíclico. Em redes *feedforward* o *output* de uma camada serve de *input* para a camada seguinte.
Desta forma, as redes neuronais são sempre constituídas por:

1. Uma **Input Layer**, primeira camada da rede neuronal, constituída por neurónios de input e responsável por receber os dados, apresentar os padrões de reconhecimento e declarar o tamanho da dataset;

2. Uma ou mais **Hidden Layers** responsáveis por grande parte do processamento, mais concretamente pela extração de características;

3. Uma **Output Layer** constituída por neurónios de output e responsável por apresentar resultados. O número de neurónios de output depende dos diferentes valores desejados como resultado. Além disso é uma camada que não contém nenhuma função de ativação;

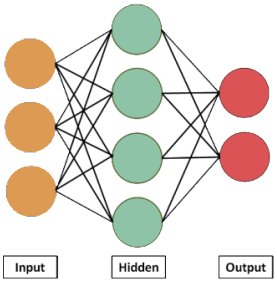


Figura 5: Representação geral da rede

2.1.5 Configuração prevista da rede

É complicado prever a configuração de uma rede, e é algo que requer testes com diferentes composições e que vamos fazer para encontrar a melhor. São encontradas algumas dificuldades na determinação do tamanho da rede (número de camadas), do número de ligações e da arquitetura. Depois de alguma análise nós esperamos que a melhor configuração seja a seguinte:

1. **Input Layer** : Nesta camada é uma boa regra criar um neurónio por cada *feature* que tivermos e mais um para o *bias*, ficando com dez neurónios no total. O nó *bias* é adicionado para aumentar a flexibilidade do modelo para ajustar os dados. Especificamente, ele permite que a rede ajuste os dados quando todas as características de entrada são iguais a zero.

2. **Hidden Layer** Optamos por apenas uma camada escondida por três razões:

   - Queremos que a rede tenha capacidade de generalizar;
   - Desejamos uma boa *perfomance*;
   - E por último, após alguma pesquisa, descobrimos que uma camada escondidada é suficiente para grande parte dos problemas;

   Para calcular o número de neurónio nesta camada, chegamos a uma boa *thumb rule*, o número de nós deve ficar entre o tamanho da camada de *input* e o tamanho da camada de *output* onde sugeriam a seguinte fórmula:

   $$(\text{Número de } \textit{inputs} + \textit{outputs}) \text{ x } 2/3$$

   Aplicando a fórmula obtemos o valor oito. Logo, teremos oito neurónios na camada escondida. Igualmente relevante, é a função de ativação utilizada nesta camada. Para o efeito, escolhemos a função *sigmoid*, que é uma função não-linear e tem a forma matemática seguinte:

   $$\sigma(x) = 1/(1 + e^{-x})$$

   Esta função pega num valor-real e coloca-o no intervalo [0,1], por exemplo, se um número for muito negativo, será zero e se um número for um positivo elevado, será 1. Iremos testar outras combinações, usando por exemplo, a função ReLU(*rectified linear unit*), que tem a seguinte forma matemática:

$$f(x) = \max(0, x)$$

onde x é o *input* do neurónio.

3. **Output Layer** Como queremos que a rede neuronal classifique os dados, restringimos os as funções de ativação que podemos usar. Por norma, para classificar os dados são utilizados dois métodos:

   - *Softmax*
   - SVM (*support vector machines*)

   Não existem grandes diferenças entre estas duas funções de ativação. Uma diferença é a forma como classificam os dados, o svm retorna uma pontuação relativamente a cada classe, enquanto que o *softmax* retorna probabilidades associadas a cada classe. Como achamos que é mais fácil analisar probabilidades iremos optar pela função de ativação "*softmax*. Logo, esta decisão, faz com que o número de neurónios na camada de "output"seja igual ao número de classes, que neste caso são dez, nove expressões faciais e uma expressão neutra.

Esta configuração foi obtida aplicando a teoria, após a elaboração do projeto iremos testar diferentes configurações de forma a perceber qual a melhor. Com forme já foi explicado, a nossa rede será *feed forward* logo, nós da mesma camada não estarão ligados mas estarão completamente conectados com os nós da camada seguinte.

2.2 Trabalho Efetuado

O grupo dividiu o trabalho em três pontos:

1. Transformar os ficheiros de texto em formato *JSON*;
2. Retirar dos conjuntos de pontos, as distâncias e ângulos mencionadas, em cima no relatório. Para isso desenvolveremos um pequeno programa para obter os pontos desejados e executar os cálculos;
3. Treinar a rede neuronal, até ser alcançado um erro que achamos suficiente baixo;

Destes três pontos o grupo já executou o primeiro e encontra-se atualmente a implementar o segundo e a testar as ferramentas que vai usar no terceiro

2.3 Resultados Esperados e forma de validação

Iremos treinar a rede de duas formas:

1. Introduzindo *frames* únicos e atribuindo uma certa expressão a estes;
2. Introduzindo conjuntos de *frames*, contendo uma informação temporal dos mesmos.

Esperamos obter um melhor reconhecimento de expressões nesta segunda fase. Isto porque algumas expressões, como negação e afirmação, requerem respetivamente, movimento horizontal e vertical da cabeça.

Iremos ainda analisar a influencia da informação de profundidade (variável z) no reconhecimento das expressões. Esperamos que a introdução desta informação traga algumas melhorias em expressões que sejam simples de reconhecer, mas pode ter o efeito inverso em expressões com uma complexidade elevada no seu reconhecimento como a negação.

De modo a validar e testar os nosso resultados, iremos separar o nosso *dataset* em 3 conjuntos:

1. **Conjunto de treino 50%**: exemplos utilizadores para treinar a rede e ajustar os pesos das arestas;
2. **Conjunto de validação 25%**: exemplos utilizados para aperfeiçoar os parâmetros da rede, por exemplo para definir o numero ótimo de neurónios na *hidden layer*;
3. **Conjunto de teste 25%**: exemplos utilizados exclusivamente para para analisar a performance da nossa rede e estimar a taxa de erro.

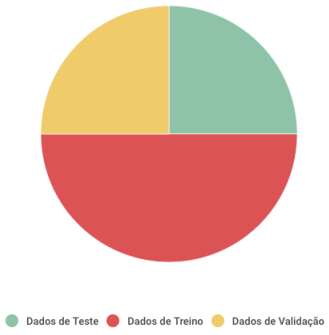


Figura 6: Gráfico representativo dos conjuntos

3 Conclusões

Em suma, acreditamos que os resultados obtidos a partir deste trabalho serão semelhantes e positivos tal como os resultados esperados, uma vez que se trata de um trabalho com um tema interessante e atual nos dias de hoje o que o torna bastante atrativo.

4 Recursos

Por forma a perceber melhor o nosso problema e o conteúdo do mesmo analisamos recorremos a estas fontes:

- FREITAS, F. A. ; Peres, S. M. ; Lima, C. A. M. ; BARBOSA, F. V. . Grammatical Facial Expressions Recognition with Machine Learning
- Notes of the Stanford CS class CS231n: Convolutional Neural Networks for Visual Recognition - http://cs231n.github.io/neural-networks-1/

Para realizar o "parse"dos ficheiros cedidos pelo "paper"para o formato "JSON"iremos utilizar a seguinte ferramenta:

- https://github.com/danwild/text2json

Por fim, para implementar a rede neuronal iremos utilizar a seguinte biblioteca:

- https://github.com/karpathy/convnetjs