Aprendizagem Computacional - Trabalho Prático 3

João Tiago Márcia do Nascimento Fernandes - 2011162899
Joaquim Pedro Bento Gonçalves Pratas Leitão - 2011150072

12 de Novembro de 2014

Índice

1 Introdução **3**

2 Aplicação Desenvolvida **4**
2.1 Graphical User Interface 4
2.2 Redes Neuronais Implementadas 5
2.3 Treino das Redes 6
2.4 Teste das Redes 7
2.5 Implementação em Matlab 8
2.6 Execução 8

3 Treino e Testes da Aplicação **9**

4 Conclusões **10**

5 Anexos **11**

1 Introdução

O presente trabalho foca-se na previsão e identificação de crises epiléticas, com base em informação de sinais cerebrais, recolhidos através da realização de um *EEG* (ElectroEncefaloGrama).

Este exame recolhe dados relativos à atividade cerebral do paciente que o realiza, sendo possível extrair um conjunto de características que permite a identificação de momentos de ocorrência de crises epiléticas (situações *ictais*) e de momentos nos quais o paciente não apresenta qualquer problema (situações *não-ictais*).

O trabalho proposto visa a criação de uma aplicação em *Matlab*, que analise os dados recolhidos após a realização de um *EEG* a um paciente, e que identifique eventuais situações em que a atividade cerebral registada corresponde a uma situação de crise epilética.

Para proceder à identificação das situações *ictais* e *não-ictais*, a aplicação desenvolvida faz uso, na sua arquitetura interna, de redes neuronais, disponíveis na *Neural Networks Toolbox* do próprio *Matlab*.

Para avaliar o desempenho e performance da aplicação desenvolvida, procederemos à análise da sensibilidade e especificidade de cada rede neuronal implementada. Estas duas métricas constituem requisitos necessários para a sua utilização em ambiente clínico, e podem ser definidas da seguinte forma:

$$Sensibilidade = \frac{PositivosVerdadeiros}{PositivosVerdadeiros + FalsosNegativos}$$

$$Especificidade = \frac{NegativosVerdadeiros}{NegativosVerdadeiros + FalsosPositivos}$$

Assim, no presente documento pretendemos apresentar de forma mais detalhada a aplicação desenvolvida, discutindo alguns detalhes da sua implementação e apresentando uma reflexão crítica sobre o seu desempenho e performance, nomeadamente da sua sensibilidade e especificidade.

2 Aplicação Desenvolvida

Tal como referido anteriormente, a aplicação desenvolvida visa analisar os dados referentes a um *EEG* de um paciente, identificando situações correspondentes a uma crise epilética.

Esta classificação pode ser realizada de duas formas distintas.

Numa primeira abordagem, a que chamamos *Classificação Individual*, é atribuído a cada elemento do conjunto de dados de entrada da aplicação uma de duas *classes*, representadas por dois valores binários:

- Classe *não-ictal*, correspondente a um estado normal do paciente (ausência de crises) e representada pelos valores *1 0*
- Classe *ictal*, correspondente a uma situação de crise, e representada pelos valores *0 1*

Na segunda abordagem, a que chamamos *Classificação em Grupo*, a classificação é realizada de forma semelhante, no entanto são considerados conjuntos de dados de entrada da aplicação, ao invés de cada elemento. Para este tipo de classificação podemos adotar duas métricas diferentes:

- Analisar o número de elementos consecutivos classificados individualmente como *ictais*, comparando-o com um dado limiar. Neste caso, se, por exemplo, existirem pelo menos 10 elementos consecutivos classificados como *ictais* então é detetada uma crise. Caso contrário nenhuma crise é detetada.
- Adotar um sistema de classificação em janela deslizante, analisando o número de elementos classificados individualmente como *ictias*, num dado universo restrito. Isto é, se pelo menos cinco dos últimos dez elementos foram classificados como *ictais* então todos os elementos nesse conjunto são classificados como *ictais*.

Relativamente a este último aspeto, optámos por adotar o segundo método de *Classificação em Grupo*, considerando uma abordagem por janelas.

De seguida apresentamos em maior detalhe a aplicação desenvolvida, salientando alguns dos seus aspetos mais importantes e relevantes.

2.1 Graphical User Interface

Para facilitar a interação do utilizador com a aplicação, foi-nos proposta a criação de uma interface gráfica onde são solicitadas ao utilizador todas as informações relevantes para a execução da aplicação, separando por completo a sua lógica interna com a especificação dos seus dados de entrada e outros parâmetros.

Assim, na interface gráfica desenvolvida são solicitadas ao utilizador várias informações que permitem a criação e treino das diferentes redes neuronais, nomeadamente:

- Tipo de rede neuronal a criar e treinar. Encontram-se disponíveis as redes *Radial Basis Function*, *Layer Recurrent Network*, *FeedForward*, *FeedForward Time Input Delay* e *Distributed Time Delay*.
- Função de Aprendizagem (ou Função de Treino) a utilizar na rede neuronal a criar (Se necessário). Encontram-se disponíveis as funções *trainscg*, *traingd* e *trainrp*.
- Função de Performance a utilizar no treino da rede neuronal (se necessário). Estão disponíveis as funções *mse* (mean squared error) e *sse* (sum squared error).
- Função de Activação dos neurónios da rede neuronal a implementar (se necessário). Estão disponíveis as funções *hardlim*, *purelin*, *logsig* e *tansig*.
- Tipo de Classificação a realizar (*Individual* ou *Em Grupo*)
- Ficheiro de dados a utilizar para treinar a rede criada
- Ficheiro de dados a utilizar para testar a rede criada
- Outros aspetos, como objetivo do treino (*Goal*), taxa de aprendizagem, etc

Para além disso, na interface desenvolvida, existe também uma secção onde são apresentados os resultados de cada teste realizado, nomeadamente a especificidade e sensibilidade da rede considera.

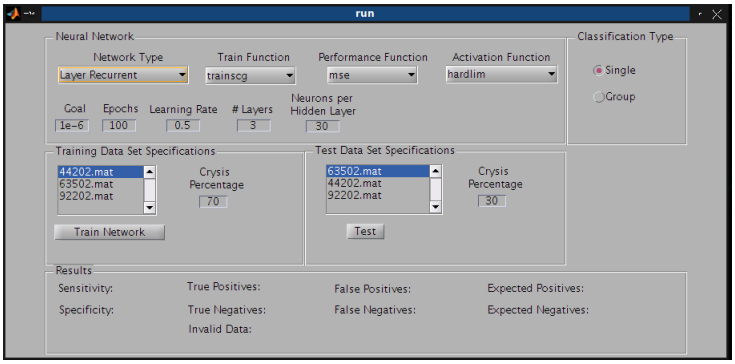

Figura 1: Interface Gráfica implementada

2.2 Redes Neuronais Implementadas

Como já referimos anteriormente, na nossa aplicação implementámos cinco redes neuronais distintas: *Radial Basis Function*, *Layer Recurrent Network*, *FeedForward*, *FeedForward Time Input Delay* e *Distributed Time Delay*.

Estas redes apresentam, naturalmente, características e propriedades distintas, sendo que umas se adequam mais ao trabalho que pretendemos realizar do que outras.

Por exemplo, considerando a rede *Layer Recurrent*, esta rede permite a introdução de atrasos em algumas características, o que lhe permite aprender a prever qualquer saída dinâmica, tendo por base entradas passadas. Este processo é possível se forem considerados neurónios e atrasos suficientes na rede.

De facto, esta é uma propriedade que vai, de certa forma, ao encontro do funcionamento de um cérebro humano, que para além de ser um sistema dinâmico, possui também memória.

Na mesma linha de raciocínio, redes que suportam a introdução de atrasos em algumas das características que constituem os dados de entrada surgem, a uma primeira vista, como boas opções para simular o comportamento de um cérebro humano, realizando uma melhor identificação das situações correspondentes a crises epiléticas. Exemplos destas redes são a rede *Distributed Time Delay* e a *FeedForward Input Time Delay*.

Por seu turno, a rede *FeedForward* também se apresenta como uma solução a considerar, dado o facto de permitir uma boa implementação de qualquer função de entradas e saídas arbitrárias, desde que considerados neurónios suficientes na(s) camada(s) escondida(s).

Por fim, é também necessário referir a rede *Radial Basis Function*, bastante utilizada para aproximar funções e cujo treino passa nomeadamente pela adição de neurónios à camada escondida até que a rede atinja a performance (*goal*) pretendida. Assim, embora possa ser necessário adicionar um elevado número de neurónios à camada escondida, acreditamos ser possível ter uma boa performance com esta rede.

2.3 Treino das Redes

Um dos principais aspetos do trabalho realizado, prende-se com o treino das redes neuronais, pois é ele que determina a boa (ou má) performance das redes implementadas.

Para o presente trabalho foram-nos fornecidos dados relativos a três pacientes, constituídos por um conjunto de características extraídas para cada elemento, e pela respetiva classe definida para cada elemento.

Uma vez que as situações em que os pacientes estão a sofrer de uma crise epilética são consideravelmente menos do que as situações em que o paciente não apresenta nenhum problema, a simples seleção de todos os elementos de um dos conjuntos fornecidos, ou de parte desses elementos, para realizar o treino da rede, sem qualquer cuidado na seleção dos elementos irá conduzir a dados de treino onde predominam situações *não-ictais*.

Nesses casos, iremos verificar uma especialização da rede na identificação de situações *não-ictais*, sem que faça uma classificação de casos *ictais* igualmente fiável.

De facto, tal situação não é desejável, uma vez que o nosso principal objetivo passa pela identificação de casos *ictais* com um grau de confiança mínimo, não

a identificação de situações *não-ictais*.

Assim, para evitar que as redes por nós treinadas se especializem em situações *não-ictais*, na constituição dos casos de treino das diferentes redes neuronais, consideramos um dos ficheiros fornecidos, e para esse ficheiro selecionamos uma percentagem dos casos *ictais* (essa percentagem é solicitada ao utilizador através da interface gráfica) que vamos incluir no nosso *data set* de treino.

Em seguida, selecionamos um número igual de situações *não-ictais*, preservando a ordem dos diferentes casos nos dados originais: Consideremos os elementos A e B, pertencentes ao *data set* original e de treino, em que A surge antes de B no *data set* original. Consideremos também que A corresponde a uma situação *ictal*, enquanto B corresponde a uma situação *não-ictal*. Então, no *data set* de treino, A surgirá também antes de B.

Para realizar o treino das diversas redes recorremos ainda a diferentes funções de treino, disponíveis e implementadas pela *Neural Network Toolbox* do *Matlab*. As funções de treino disponíveis são a função *traingd*, *trainscg* e *trainrp*.

Com exceção da rede *RBF* (*Radial Basis Function*) implementada, o treino das restantes redes neuronais é realizado com recurso à função *train* da *Neural Network Toolbox* do *Matlab*. Uma vez que o treino das redes é uma operação complexa e exigente em termos computacionais, tendo em conta o tipo de redes criadas e a dimensão dos dados para proceder ao treino das redes, estas foram treinadas com aceleração gráfica, disponível nas versões mais recentes do *Matlab*. Para tal, basta adicionar os parâmetros *'useGPU', 'yes'* aquando da chamada da função *train*: *train(network, P, T, 'useGPU', 'yes')*.

Para a rede *RBF*, o *Matlab* realiza o seu treino aquando da criação da rede, não sendo necessária a invocação da função *train*.

2.4 Teste das Redes

Uma vez completo o treino de uma rede neuronal, esta pode ser testada, de forma a verificar o seu bom, ou mau, funcionamento. Para isso, criámos um conjunto de dados de teste, baseados nos três *data sets* inicialmente fornecidos.

O processo de criação dos dados de teste é semelhante ao utilizado na constituição dos dados de treino das redes neuronais:

É solicitado ao utilizador que indique o ficheiro (de entre os três ficheiros fornecidos) de onde serão extraídos os dados de teste, e qual a percentagem de situações *ictais* a incluir. Em seguida, o ficheiro escolhido é analisado, e são considerados todos os dados nele presentes, a partir do final do ficheiro, até que o número de situações *ictais* incluídas seja igual à percentagem especificada.

Por outras palavras, se o utilizador especifica que pretende incluir 25% das situações *ictais* nos dados de treino, e se todas as situações *ictais* identificadas nesse *data set* se encontram nas posições $10-20$, $40-50$, $60-70$ e $80-100$, então o nosso *data set* de treino será constituído por todos os elementos do ficheiro, desde a posição 60 até ao final do ficheiro.

Uma vez que aquando da realização dos testes na rede esta já se encontra treinada, é irrelevante considerarmos nos *data sets* de teste situações *ictais* na mesma ordem de grandeza do que situações *não-ictais*, pois apenas estamos a

executar a rede para um conjunto de dados, sem que este afete de forma alguma o funcionamento da rede em situações futuras.

2.5 Implementação em Matlab

2.6 Execução

3 Treino e Testes da Aplicação

4 Conclusões

5 Anexos