

## Ficha de Trabalho nº 3

### Redes Neurais: Deep Learning ToolBox do Matlab

#### Bibliografia

Material de apoio disponível no Moodle.

Mathworks site: <https://www.mathworks.com/help/deeplearning/>

#### 1. Funções da Deep Learning ToolBox

Esta toolbox do Matlab possui funções próprias para criar, inicializar, treinar e simular redes neurais. Para exemplificar o uso dessas funções, nesta ficha serão implementadas as seguintes redes:

- i) Perceptrão semelhante ao da aula anterior.
- ii) Rede neuronal multicamada do tipo *feedforward*.

As funções mais importantes e necessárias para a realização desta ficha de trabalho são:

- **perceptron: cria uma rede neuronal tipo perceptrão**  
nome\_rede = perceptron;
  - Por defeito, a função de ativação é a *hardlim* e a função de treino é a *learnp* (podem ser indicadas alternativas utilizando os argumentos opcionais da função *perceptron*)
- **feedforwardnet: cria uma rede neuronal tipo feedforward**  
nome\_rede = feedforwardnet;
  - Por defeito, cria uma rede neuronal com uma camada escondida com 10 nós (a arquitetura por defeito pode ser alterada utilizando os argumentos opcionais da função);
  - Os inputs e outputs não são indicados neste ponto. A sua dimensão será automaticamente configurada mais tarde durante o processo de treino
  - Funções de ativação por defeito: camadas escondidas (*tansig*) e camada de saída (*purelin*);
  - Função de treino: *trainln*.

Algumas configurações que pode fazer em redes **feedforward**:

Nº de camadas / neurónios

Nº de iterações do treino (o *default* é 1000)

Funções de ativação

Função de treino

Usar todos os exemplos no treino

Fazer a divisão dos exemplos em treino/validação/teste (o *default* é 70%, 15%, 15%)

- **train: treina a rede neuronal**  
nome\_rede = train(nome\_rede, input, target)
- **testar/simula a rede neuronal**  
out = nome\_rede(input)  
out = sim(nome\_rede, input)
- **view: visualizar a rede neuronal**  
view(nome\_rede)
- **perform: desempenho da rede neuronal**  
erro = perform(net,t,y)
- **plotperform: desempenho da rede neuronal**  
se no treino da rede guardar a variável tr  
[nome\_rede, tr] = train(nome\_rede, input, target)  
  
pode visualizar o gráfico de performance:  
plotperform(tr)
- **save: guardar uma rede**  
save('ficheiro', 'nome\_rede')
- **load: carregar uma rede guardada**  
load('ficheiro')

Para mais detalhes sobre estas funções faça: >> *help nome\_da\_função*

## 2. Implementação de um perceptrão com as funções da Toolbox

- a) Edite o ficheiro **perceptrao3a.m** disponibilizado no Moodle. Usando as funções da toolbox descritas no início desta ficha complete o código:
- Defina os targets para as funções lógicas OR, NAND, XOR. Analise a resposta do utilizador na variável **tmp** e use a instrução **switch ... case** para proceder à inicialização dos diferentes **targets**.
  - Crie uma rede neuronal do tipo **perceptrão**
  - Defina o nº de épocas = 100 (**nome\_da\_rede.trainParam.epochs = 100**)
  - Treine a rede criada
  - Teste a rede, usando os mesmos dados de entrada
  - Visualize o erro e o gráfico de performance
- b) Execute a função e teste a sua funcionalidade para as funções AND, OR, NAND e XOR. Analise e comente os resultados obtidos.

## 3. Implementação de uma rede *feedforward* usando as funções da Toolbox

- a) O exercício anterior e o realizado na aula passada mostraram que a função XOR não pode ser aprendida com um perceptrão. Para tentar resolver problema vai ser implementada uma rede neuronal *feedforward* com vários neurónio/camadas. Edite o ficheiro **rn3b.m** disponibilizado no Moodle e use as funções da toolbox para completar o código:
- Defina os targets para as funções lógicas OR, NAND, XOR. Analise a resposta do utilizador na variável **tmp** e use a instrução **switch ... case** para proceder à inicialização dos diferentes targets.
  - Crie uma rede neuronal do tipo *feedforward* com uma camada escondida com 10 nós;
  - Ajuste os seguintes parâmetros da rede (nos restantes devem ser usados os valores por defeito):
    - Função de ativação da camada de saída: *tansig*
    - Função de treino: *traindx*
    - Número de épocas de treino: 100
    - **Todos os exemplos de input devem ser usados no treino**
  - Treine a rede criada
  - Teste a rede, usando os mesmos dados de entrada
  - Visualize o erro e o gráfico de performance
- b) Execute a função e teste a sua funcionalidade para as funções AND, OR, NAND e XOR. Analise e comente os resultados obtidos.
- c) Altere a função de treino para a *trainlm*: Repita os testes efectuados na alínea 4b) e analise eventuais diferenças em relação aos resultados obtidos anteriormente.

## 4. Rede Neuronal para verificação de paridade par

Implemente uma função **paridade\_par** para quatro entradas binárias.

Num problema de paridade par com N entradas, a rede deve devolver 1 se um número par de inputs tiver o valor 1. Caso contrário, devolve o valor 0.

Execute as seguintes tarefas:

- Inicialize matriz de **entrada** com as várias possibilidades para 4 entradas.
- Crie a variável **target** correspondente
- Use as funções da **toolbox** para inicializar o perceptrão, treinar e testar. Use diferentes funções de activação. O perceptrão conseguiu aprender?
- Use agora uma rede neuronal com uma camada escondida para resolver este problema.
  - Experimente diferentes topologias e analise os resultados obtidos.

## 5. Rede Neuronal para classificação de pacientes cardíacos

### 5.1 Treinar e gravar uma rede neuronal

Crie uma nova função de nome **trainHeart.m**

Leia o ficheiro **heart\_train.csv** para uma matriz

```
S = readmatrix('heart_train.csv', 'Delimiter', ',', 'DecimalSeparator', '.');
```

Usando as funções do Matlab prepare as matrizes de *entrada* e *target*:

- A matriz de entrada deve ter os exemplos de treino nas colunas e as entradas nas linhas.
- O *target* corresponde à última coluna do ficheiro.

Crie uma rede *feedforward* default com a instrução **net = feedforwardnet;**

Treine a rede com todos os exemplos de treino.

Simule a rede e veja o erro obtido

A saída *y* da rede deve ser convertida para binária: se  $y \geq 0.5$  então  $y = 1$  senão  $y = 0$

Teste algumas topologias diferentes, por exemplo, mais camadas, mais neurónios.

Guarde a melhor rede com a instrução

```
save('nn_heart1.mat', 'net');
```

### 5.2 Carregar uma rede gravada e testá-la com novas instâncias

Crie uma nova função de nome **testHeart.m**

Leia o ficheiro **heart\_test.csv** para uma matriz

```
S = readmatrix('heart_test.csv', 'Delimiter', ',', 'DecimalSeparator', '.');
```

Esta matriz de entrada tem 6 instâncias que nunca foram usadas no treino da rede. Coloque os exemplos nas colunas e as entradas nas linhas.

Faça o *load* da rede que guardou anteriormente com a instrução:

```
load('nn_heart1.mat')
```

Usando a função **sim**, execute a rede e analise a saída. Veja se a classificação feita foi correta.

A saída correta destas 6 instâncias deve ser [1 1 1 0 0 0].