Ficha de Trabalho nº 7
Redes Neuronais: 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 neuronais. Para exemplificar o uso dessas funções, nesta ficha serão implementadas as seguintes redes:

i) Perceptron 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: *trainlm*.

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(nome_rede,target,out)
  ```

- **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 perceptron com as funções da ToolBox

a) Edite o ficheiro **perceptron7a.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 perceptron. Para tentar resolver problema vai ser implementada uma rede neuronal ***feedforward*** com vários neurónio/camadas. Edite o ficheiro **rn7b.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: *traingdx*
     - 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:

a. Inicialize matriz de **entrada** com as várias possibilidades para 4 entradas.
b. Crie a variável **target** correspondente

c. Use as funções da **toolbox** para inicializar o *perceptron*, treinar e testar. Use diferentes funções de ativação. O *perceptron* conseguiu aprender?

d. 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

O ficheiro **heart_train.csv** consiste num dataset de classificação de doença cardíaca.
Possui 297 pacientes caracterizados por 13 atributos relativos a exames e analises médicas. Para cada paciente existe a indicação de se teve doença cardíaca ou não (coluna *target*)
O objetivo deste exercício é treinar uma RN para aprender a classificar os pacientes relativamente à doença cardíaca.
Pretende-se que usem o ficheiro ***heart_train.csv*** para treinar a rede usando as funções da toolbox exploradas nesta aula. Executem algumas alterações de parâmetros e topologias e guardem uma rede que tenha bom desempenho.
De seguida devem usar essa rede treinada com 6 pacientes que estão no ficheiro **heart_test.csv**, ver que diagnóstico a rede neuronal atribuiu, e verificar se corresponde à resposta correta.

Execute as seguintes tarefas:

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 >=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_train.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].