Ficha de Trabalho nº 2

Redes Neuronais: NNToolBox do Matlab

Bibliografia

Material de apoio disponível no Moodle.
Mathworks site: http://www.mathworks.com/help/toolbox/nnet/

1. Neural Network Data Manager

A janela de edição da NNtoolbox disponibiliza uma interface gráfica para criar e testar redes neuronais. Explore as funcionalidades desta ferramenta realizando os exemplos descritos nos slides **NNTool_DM_Ficha3.pdf**.

2. Funções da NNToolBox

Para além da interface gráfica, o 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) Perceptrão semelhante ao da aula anterior (rede monocamada).
ii) Rede neuronal multicamada do tipo *feedforward*.

A descrição das funções está feita de forma mais detalhada o ficheiro **resumoNNtool.pdf** que se encontra no Moodle.
As funções mais importantes e necessárias para a realização desta ficha de trabalho são:

- **perceptron[1]: 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[2]: 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);

[1] Versões do Matlab anteriores à release R2010b devem usar a função *newp* (ver help para detalhes de utilização).
[2] Versões do Matlab anteriores à release R2010b devem usar a função *newff* (ver help para detalhes de utilização).

- Os inputs e outputs não são indicados neste ponto. A sua dimensão será automaticamente configurada mais tarde durante o processo de treino (também podem ser configurados explicitamente através da função **configure**);
- Funções de ativação por defeito: camadas escondidas (*tansig*) e saída (*purelin*);
- Algoritmo de treino: *trainln*.

- **train: treina a rede neuronal**
```
nome_rede = train(nome_rede, input, target)
```

- **sim: testa/simula a rede neuronal**
```
out = sim(nome_rede, input)
```

- **view: visualizar a rede neuronal**
```
view(nome_rede)
```

Para mais detalhes sobre estas funções faça: >> help *nome_da_função*

3. Implementação de um perceptrão com as funções da NNToolBox

a) Edite o ficheiro **perceptrao3a.m** disponibilizado no Moodle. Usando as funções da NNToolBox 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 **log_op** 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

b) Execute a função e teste a sua funcionalidade para as funções AND, OR, NAND e XOR. Analise e comente os resultados obtidos.

4. Implementação de uma rede *feedforward* com 2 camadas usando a NNtool

a) O exercício anterior e o realizado na aula passada mostraram que a função XOR não pode ser aprendida com um perceptrão de uma camada. Para tentar resolver problema vai ser implementada uma rede neuronal com duas camadas. Edite o ficheiro **rn3b.m** disponibilizado no Moodle e use as funções da NNToolBox para completar o código:
- Defina os targets para as funções lógicas OR, NAND, XOR. Analise a resposta do utilizador na variável **log_op** 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

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.

5. 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.

a) 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 **nntool** para inicializar o perceptrão, treinar e testar. Use diferentes funções de activação. O perceptrão conseguiu aprender?

    d. Use agora uma rede neuronal com uma camada escondida para resolver este problema.

        - Experimente diferentes topologias e analise os resultados obtidos.