

Ficha de Trabalho nº 4

Redes Neurais para Resolução de Problemas

Bibliografia

Material de apoio disponível no Moodle.

Mathworks site: <http://www.mathworks.com/help/toolbox/nnet/>

1. Introdução

As redes neurais (RN) são bastante utilizadas para aproximar sistemas complexos, difíceis de modelar através das técnicas de otimização convencionais. As aplicações mais comuns das RN são a aproximação de funções, reconhecimento de padrões e classificação. Não existe uma fórmula exata que permita decidir sobre a melhor arquitetura de uma RN para um dado problema. Normalmente essa arquitetura é encontrada por tentativa e erro.

Com esta ficha pretende-se implementar e testar RNs multicamada para um problema de classificação. Além da implementação, serão testadas diferentes arquiteturas, funções de ativação e funções de treino de forma a obter conclusões sobre o desempenho das RNs para o problema em causa.

2. Classificação: Iris Data Set

Existem 3 variantes da flor Íris no Havai, com ligeiras diferenças no comprimento e largura das suas pétalas. O *Fisher Iris data set* (<http://archive.ics.uci.edu/ml/datasets.html>) contém um conjunto de exemplos de flores pertencentes às 3 classes, estando 4 atributos das respectivas pétalas associados a cada exemplo. A descrição geral do problema em estudo pode ser encontrada na tabela seguinte.

Número de atributos	4
Descrição dos atributos	1. Largura da pétala 2. Comprimento da pétala 3. Largura da sépala 4. Comprimento da sépala
Característica dos atributos	Real
Número de classes	3
Descrição das classes	1. <i>Iris Setosa</i> 2. <i>Iris Virginica</i> 3. <i>Iris Versicolor</i>
Número de instâncias	150 (50 por classe)
Tarefa associada	Classificação

O Matlab possui o *data set* que vai ser utilizado neste estudo. Para o carregar basta fazer:

```
» load iris_dataset
```

Esta instrução cria 2 variáveis:

- Matriz *irisInputs* com 4 linhas e 150 colunas contendo os atributos reais dos 150 exemplos
- Matriz binária *irisTargets* com 3 linhas e 150 colunas contendo a classe a que pertencem cada um dos 150 exemplos:
 - o Classe 1: saídas 1-0-0;
 - o Classe 2: saídas 0-1-0;
 - o Classe 3: Saídas 0-0-1.

3. Rede neuronal

Pretende-se implementar e testar redes neuronais multicamada para resolver o problema de classificação descrito no ponto anterior. Todas as redes neuronais testadas terão 4 entradas e 3 saídas, para poderem lidar com os dados do caso em estudo.

Divisão de Exemplos

Ao treinar redes neuronais para classificação, é habitual dividir os exemplos em 3 conjuntos: treino, validação e teste. O primeiro conjunto é utilizado durante a aplicação do algoritmo de treino supervisionado. O segundo conjunto é útil para detetar quando deve terminar o treino (i.e., ajuda a detetar quando a rede começa a perder capacidade de generalização). O terceiro não é utilizado durante o processo de treino e serve normalmente para fazer comparações entre diferentes estratégias de classificação.

O principal objectivo deste ponto é tentar perceber de que forma as variações na arquitetura e funções da rede influenciam a sua capacidade para classificar corretamente os exemplos que recebe. Em concreto pretende-se que estude a influência das seguintes componentes no desempenho da rede:

- a) Número de camadas escondidas e número de nós por camada escondida. Estes valores devem ser especificados na criação da rede (como argumentos da função `feedforwardnet`).
- b) Funções de ativação das camadas escondidas e da camada de saída. Para alterar a função de ativação da camada *y* na rede *net* deve utilizar-se a instrução:
`net.layers{y}.transferFcn = '.....'`
- c) Função de treino da rede neuronal. A função de treino da rede *net* pode ser indicada na criação da rede ou através da instrução: `net.trainFcn = '.....'`
- d) Divisão dos exemplos. A função de divisão por defeito (*dividerand*) cria os 3 conjuntos de treino, validação e teste, respetivamente, com 70%, 15% e 15% dos exemplos. Estes valores podem ser alterados através das variáveis pertencentes ao objeto `net.divideParam`.
 - o `net.divideFcn = 'dividerand'`
 - o `net.divideParam.trainRatio = 0.70`
 - o `net.divideParam.valRatio = 0.15`
 - o `net.divideParam.testRatio = 0.15`

4. Tarefas a executar

4.1 Rede Neuronal por defeito

O ficheiro *iris_ex.m* contém uma implementação completa de uma rede neuronal para resolver o problema de classificação proposto nesta ficha. A sua configuração é a seguinte:

- 1 camada escondida com 10 nós;
- Funções de ativação: $\{tansig, purelin\}$;
- Função de treino: $\{trainlm\}$;
- Divisão dos exemplos: $\{70\%, 15\%, 15\}$.

Analise o código e identifique as suas principais secções. Execute a função *iris_ex.m* e analise os resultados obtidos:

- Gráfico do desempenho do classificador nos diferentes conjuntos de exemplos. Encontre o ponto em que a rede neuronal começa a perder a capacidade de generalizar.
- Matriz de confusão das classificações. Quantos exemplos foram classificados de forma errada?
- Precisão obtida no total dos 150 exemplos e apenas no conjunto teste (valores surgem na janela de comando).

4.2 Estudo comparativo

Realize experiências e analise o impacto das variações nas componentes da rede neuronal:

- Varie as camadas escondidas entre 1 e 3 e o número de nós por camada entre 2 e 10;
- Varie as funções de ativação: $\{logsig, tansig, purelin\}$;
- Varie as funções de treino: $\{trainlm, trainbfg, traingd\}$;
- Teste diferentes distribuições dos 150 exemplos pelos 3 conjuntos.

Utilize o ficheiro excel *estudoNN.xlsx* disponível no moodle para recolha de informação. Analise os resultados e retire algumas conclusões.

5. Novo exercício

Implemente uma nova função *colesterol* que implemente uma rede neuronal multicamada para classificar um conjunto de pacientes de acordo com o tipo de colesterol que possuem

1. LDL
2. VLDL
3. HDL

O data set **cho_dataset** possui 264 exemplos de pacientes analisados com base em 21 parâmetros analisados nas amostras sanguíneas e a respectiva classificação se possui ou não colesterol.

Pode carregar este dataset fazendo na linha de comando **load cho_dataset**

Analise as duas variáveis criadas na memória:

- **choInputs** – matriz 21x264 que define os 264 pacientes analisados, baseados nos 21 parâmetros sanguíneos,
- **choTargets** - matriz 3x264 que para cada paciente da variável de entrada, indica o tipo de colesterol que possui 1. LDL 2. VLDL 3. HDL

Implemente a função colesterol fazendo as seguintes tarefas:

1. Carregue o dataset `cho_dataset`
2. Crie uma rede neuronal feedforward:
 - 1 camada escondida com 10 nós;
 - Funções de ativação: `{tansig, purelin}`;
 - Função de treino: `{trainlm}`;
 - Divisão dos exemplos: `{70%, 15%, 15}`.
3. Treine a rede neuronal
4. Teste a rede neuronal
5. Implemente as medidas de desempenho da rede nos exemplos de treino e de teste
6. Gere a matriz de confusão.

Execute a função e registe os valores obtidos no ficheiro EXCEL no separador colesterol

1. Altere a topologia da rede de acordo com o ficheiro e verifique se obtém melhorias nos resultados
2. Altere a função de treino de acordo com o ficheiro e verifique se obtém melhorias nos resultados
3. Altere as funções de activação de acordo com o ficheiro e verifique se obtém melhorias nos resultados
4. Altere a divisão dos exemplos para treino/teste de acordo com o ficheiro e verifique se obtém melhorias nos resultados