Aluno: André G. C. Pacheco

Orientador: Renato A. Krohling

Redes Neural Artificiais utilizando MATLAB

Universidade Federal do Espírito Santo - UFES
Vitória - Outubro de 2014

Sumário

1 Introdução

O *MATLAB* é um ambiente de programação para desenvolvimento de algoritmos, análise de dados, visualização de gráficos, cálculo numérico, dentre outros. Uma das principais características do ambiente é possuir diversas aplicações incorporadas, tornando a resolução de problemas de computação técnica mais rápida do que com outras linguagens como C, C++ ou Java.

Dessa forma, neste tutorial será mostrado o básico para que se possa utilizar a caixa de ferramentas de redes neurais (*Neural Network Toolbox*) no *MATLAB*. Com a *toolbox* é possível desenhar, programar, visualizar, treinar e simular redes neurais artificiais de maneira fácil e rápida.

Uma observação importante é que esse tutorial mostrará apenas como utilizar a ferramenta por meio de linhas de comandos e m-files. Para trabalhar com a interface gráfica basta executar o comando *nntool* no *bash* do *MATLAB* e uma janela com a ferramenta se abrirá. Para mais sugere-se a vídeo aula fornecida pelo próprio *MATLAB* [1]. Além disso, não é escopo do tutorial se adentrar nos conceitos de Redes Neurais Artificiais (RNA). Sendo assim, é necessário um conhecimento prévio sobre o assunto.

2 Implementação

Para utilização de uma RNA são necessários os seguintes passos:

- Definir os padrões de entrada e saída
- Inicialização da rede; permanente na resposta
- Treinar a rede
- Executar a rede

2.1 Inicialização da rede

Para inicializar uma RNA *feedfoward* no*MATLAB* são disponibilizadas algumas funções. Todavia a principal delas é a mostrada a seguir:

```
net = feedforwardnet(hiddenSizes,trainFcn)
```

Essa função cria uma RNA *feedfoward* com o número de neurônio(s) na(s) camada(s) ocultadas descritos pelo parâmetro *hiddenSizes*. Esse parâmetro é um vetor linha em que cada elemento representa a quantidade de neurônios da camada oculta de seu índice. Por exemplo: *hiddenSizes* = [2 4 3], representa 3 camadas ocultas com 2, 4 e 3 neurônios respectivamente. (NOTA: no MATLAB um escalar nada mais é que um vetor 1x1, portanto é permitido passar um escalar como parâmetro, o que significa apenas uma camada oculta com o número de neurônios especificados). O número de neurônios das camadas de entrada e saída são definidos pelos padrões de entrada para o treinamento da rede. Por fim o parâmetro *trainFcn* é opcional e determina o modelo de treinamento da rede. Este último será melhor discutido no próximo tópico.

Além da função mostrada acima, o *MATLAB* possui mais três funções de inicialização. São elas:

```
net = cascadeforwardnet (hiddenSizes,trainFcn)
net = fitnet (hiddenSizes,trainFcn)
net = patternnet (hiddenSizes,trainFcn,performFcn)
```

Essas três funções visam facilitar a solução de problemas específicos como o de *fitting*, por exemplo. Com exceção da *cascadeforwardnet*, que muda a topologia da rede, as demais

são apenas redes específicas da função *feedforward*, que basicamente às generalizam. Para mais informações sobre essas funções sugerimos [2, 3 e 4].

Continuando, o *MATLAB* permite visualizar graficamente a topologia da rede por meio da função:

```
view(net)
```

Essa função recebe como parâmetro uma rede inicializada e exibe, graficamente, a arquitetura da mesma. Para exemplificar, vamos inicializar uma rede com 2 camadas ocultas com 2 e 3 neurônios em cada uma delas e vamos visualizá-la:

```
net = feedforwardnet([2 3]);
view (net);
```

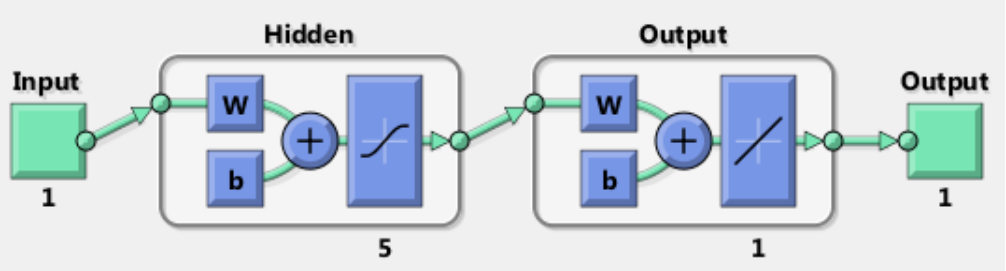

Figura 1: Exibindo a rede inicializada

A Figura 1 exibe gráficamente a rede. Observe que as entradas e saídas estão zeradas pois ainda não definimos os padrões de entrada para treinamento da rede. Para definir um padrão de entrada é utilizado a seguinte função:

```
net = configure(net,x,t);
```

Essa função recebe como parâmetro a rede inicializada e um padrão de entradas. Para exemplificar, é mostrado o código a seguir, no qual X e T são os vetores de treinamento da rede e irão definir a entrada e saída da mesma:

```
X = [270 287 272 261; 287 272 261 316; 272 261 316 294; 261 316 294 269];
T = [292 291 270 269; 287 286 272 271];
net = feedforwardnet(10);
net = configure(net,X,T);
view(net);
```

Foi criado uma rede com 10 neurônios na camada oculta e logo após configurada de acordo com o padrão de dados, que possui 4 entradas e 2 saídas. Na Figura 2 é possível observar o gráfico gerado pela funçao *view*.

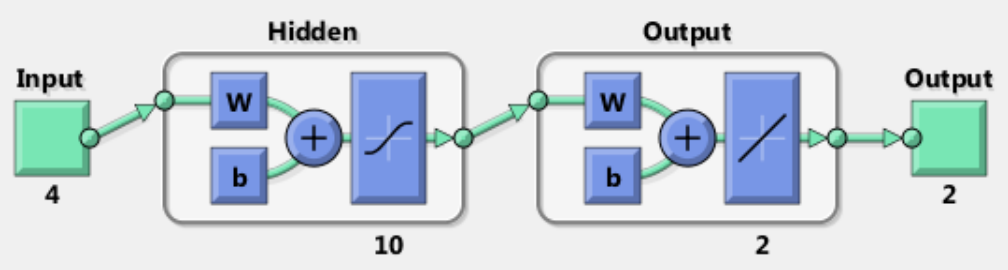

Figura 2: Exibindo a rede inicializada e configurada

2.2 Treinamento da rede

Para realizar o treinamento da rede o *MATLAB* disponibiliza a seguinte função:

```
[net,tr] = train(net,X,T)
```

A função *train* possui mais parâmetros, toda via, aqui, vamos discutir apenas os parâmetros básicos para o treinamento. Para mais sugerimos a leitura de [5]. Para começar, é necessário definir qual o algoritmo de treinamento. O MATLAB implementa o *backpropagation* com alguns algoritmos, como mostra a tabela a seguir:

Tag	Algoritmo
traingd	Gradient descent backpropagation
traingda	Gradient descent with adaptive learning rate backpropagation
traingdm	Gradient descent with momentum backpropagation
traingdx	Gradient descent with momentum and adaptive learning rate backpropagation
trainlm	Levenberg-Marquardt backpropagation

Tabela 1: Algoritmos de treinametos de uma RNA no *MATLAB*

Para escolher o algoritmo a ser executado no treinamento, existem duas formas. A primeira é colocando a tag no parâmetro *TrainFcn* da função *feedfowardnet*, apresentada na sessão 2.1. A segunda forma é fazendo:

```
net.trainFcn = 'traingd'
```

Sendo *'traingd'* a tag desejada e *net* a rede neural inicializada. Se isso não for realizado, a linguagem utiliza como default o algoritmo *'trainlm'*.

Os parâmetros X e T são os vetores de treinamento da rede, como mostrado no exemplo da sessão 2.1. Com isso, o treinamento retorna a rede treinada (*net*) e uma variável *tr* que possui o chamado *training record* com iterações e o desempenho da rede.

Além disso, vários parâmetros podem ser *setados* para o treinamento. Alguns deles são mostrados na tabela a seguir:

Sintaxe	**Default**	**Descrição**
net.trainParam.epochs	1000	Número máximo de iterações
net.trainParam.lr	0.01	Taxa de aprendizagem
net.trainParam.time	inf	Tempo máximo de treinamento
net.layers{i}.transferFcn	'tansig'*	Função de ativação ('purelin', 'tansig', 'logsig')

Tabela 2: Parâmetros para treinametos de uma RNA no *MATLAB*

* As camadas ocultas recebem como default *'tansig'* e as camadas de saídas *'purelin'* [7].

2.3 Execuçao da rede

Após o treinamento da rede o próximo passo é executa-la para uma entrada qualquer. Isso é realizado de maneira fácil, fazendo:

```
out = net (test);
```

Sendo o parâmetro *test* o seu vetor para testes da rede e *net* a sua rede treinada. Ao fim, ela retorna em *out* a saída de acordo com a entrada. Existe uma forma alternativa para executar a rede que é utilizando a função *perform*. Não vamos nos adentrar sobre ela, mas caso queira mais informaçoes, sugerimos a leitura de [6].

2.4 Funções relevantes

Existem algumas funções relevantes que podem ser úteis em algum algoritmo para redes neurais, dentre elas podemos destacar:

Função	Descrição
getwb(net)	Retorna os valores de peso e bias em um único vetor
net = setwb(net,wb)	Seta os valores dos pesos e bias em um único vetor
[b,IW,LW] = separatewb(net,wb)	Separa os valores de pesos e bias*
Wb = formwb(net,b,IW,LW)	Junta pesos e bias em um único vetor
perf = mse(net,t,y)	Retorna o desempenho da rede segundo o erro quadrático médio

Tabela 3: Funções relevantes para manipulaçao de uma RNA no *MATLAB*

* IW = pesos de entrada, LW = pesos das camadas e b = bias

3 Exemplo de treinamento

Como exemplo vamos treinar uma rede neural para aprender a função seno. Essa rede terá duas camadas ocultas com três e dois neurônios, respectivamente. Para isso, primeiro definimos os vetores de treinamento, que serão nossos padrões de entrada da rede.

```
X = [-2 -1.75 -1.5 -1.25 -1 -0.75 -0.5 -0.25 0 0.25 0.5 0.75 1 1.25 1.5 1.75 2];

T = [sin(-2) sin(-1.75) sin(-1.5) sin(-1.25) sin(-1) sin(-0.75) sin(-0.5) sin(-0.25)
sin(0) sin(0.25) sin(0.5) sin(0.75) sin(1) sin(1.25) sin(1.5) sin(1.75) sin(2)];
```

Esse padrão será utilizado no treinamento da rede. Agora vamos montar a rede e configura-la para o padrão acima.

```
net = feedforwardnet([3 2]);
net = configure(net,X,T);
view (net);
```

Após o comando *view* deve ser exibido a rede mostrada na Figura 3.

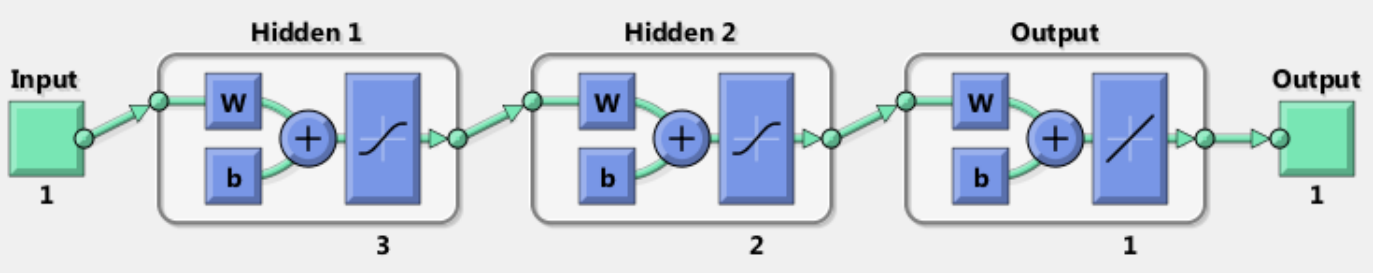

Figura 3: Rede do exemplo inicializada e configurada

Agora, vamos definir o algoritmo de treinamento e a função de ativaçao da camada oculta. Escolheremos o algoritmo *Gradient descent backpropagation* e como funçao de ativação a sigmóide.

```
net.trainFcn = 'traingd';
net.layers{1}.transferFcn = 'logsig';
```

Feito os passos acima, agora vamos treinar a rede mantendo os outros parâmetros como default.

```
[net,tr] = train(net,X,T);
```

Com a rede treinada, executamos a mesma para o mesmo padrão de entradas.

```
resultado = net (X);
```

A Figura 4 mostra o plot da funçao seno e da rede treinada. Observe que o resultado é praticamente o mesmo.

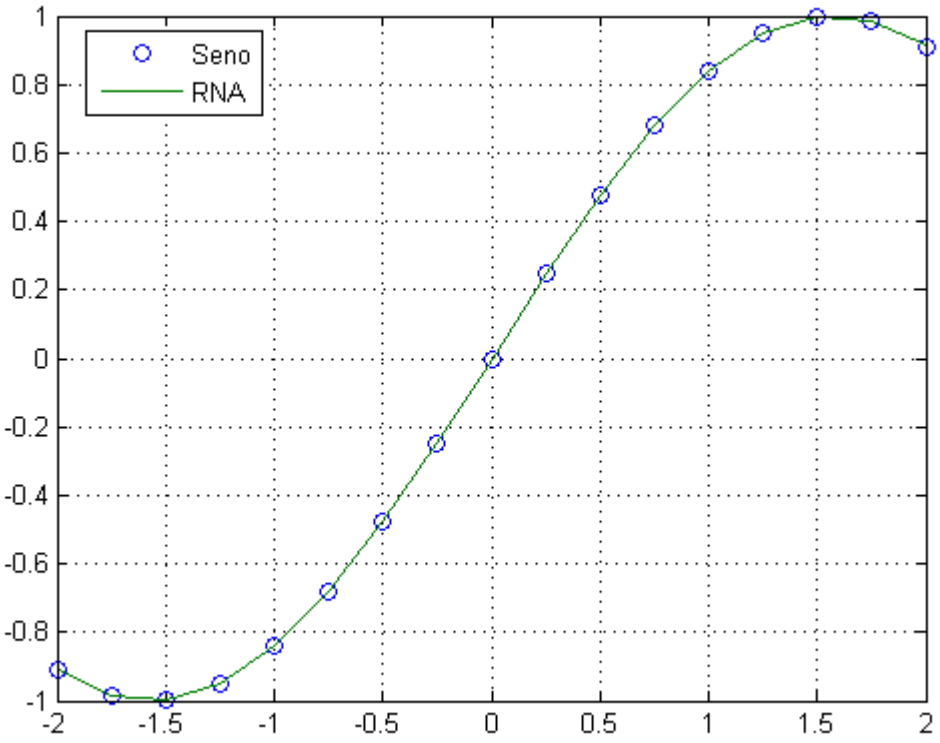

Figura 4: Resultado da rede treinada

4 Considerações finais

Neste tutorial procuramos mostrar e exemplificar os passos básicos para se trabalhar com redes neurais artificiais no *MATLAB*. O tutorial mostrou que é fácil e rápido trabalhar com a ferramenta. Todavia, existem mais funções na *toolbox* que não foram mostradas aqui justamente por fugir a ideia principal do tutorial: ser enxuto e eficaz. As funções citadas, mas não explicadas, possuem links de como funcionam na sesão de referências bibliográficas.

5 Referências Bibliográficas

[1] http://youtu.be/2Z4959acjKs
[2] http://www.mathworks.com/help/nnet/ref/cascadeforwardnet.html
[3] http://www.mathworks.com/help/nnet/ref/fitnet.html
[4] http://www.mathworks.com/help/nnet/ref/patternnet.html
[5] http://www.mathworks.com/help/nnet/ref/train.html
[6] http://www.mathworks.com/help/nnet/ref/perform.html
[7] http://www.mathworks.com/help/nnet/ug/multilayer-neural-network-architecture.html