Identificação de Caracteres com Rede Neuronal Artificial com Interface Gráfica

João Paulo Teixeira*, José Batista*, Anildio Toca**, João Gonçalves**, e Filipe Pereira**
* Departamento de Electrotecnia
Escola Superior de Tecnologia e de Gestão - Instituto Politécnico de Bragança
Campus Sta. Apolónia – Bragança
Telf: +351 273 303 3129; fax: +351 273 313 051; e-mail: joaopt@ipb.pt, jbatista@ipb.pt

**Escola Superior de Tecnologia e de Gestão - Instituto Politécnico de Bragança
Campus Sta. Apolónia – Bragança
Telf: +351 273 303 3129; fax: +351 273 313 051; e-mail: anildio.cb.toca@alunos.ipb.pt,
joao.nc.goncalves@alunos.ipb.pt, filipe.gp.pereira@alunos.ipb.pt

***Resumo* – O presente artigo aborda uma metodologia para efectuar o reconhecimento de caracteres utilizando redes neuronais artificiais, com recurso da Toolbox Neural Network do Matlab®. Cada caractere é desenhado numa matriz 5x5, em que os pontos escuros do caractere tomam o valor de ´1´ e os pontos claros equivalem a ´0´. Posteriormente, esta matriz é reorganizada numa única coluna, de modo que cada coluna da matriz de input para a função de treino da rede represente um único caractere. As colunas da matriz de output determinam os caracteres correspondentes às colunas da matriz de input, com uma representação de um '1' para o caractere desejado e '0' nos outros casos. Foi desenvolvida uma interface gráfica utilizando o GUIDE (Graphical User Interface Development Environment) do Matlab®. Deste modo, é possível escolher os parâmetros para criar a rede neuronal, treiná-la e proceder à sua simulação, de uma forma interactiva e agradável.**

Palavras-chave: Classificação de caracteres, Redes Neuronais Artificiais, Matlab.

1. Introdução

O presente artigo apresenta como se desenvolve e treina uma RNA com aprendizagem supervisionada através da toolbox do Matlab Neural Network para o reconhecimento de caracteres alfanuméricos, e a sua simulação com novas entradas.
O primeiro contributo sobre redes neuronais artificiais (RNA) foi dado por McCulloch e Pitts em 1943 com a introdução do modelo simplificado de um neurónio artificial baseado no neurónio biológico.
As RNA podem ser melhor caracterizadas como um modelo computacional abstracto do cérebro humano. Análogo ao cérebro, uma RNA é composta por neurónios artificiais, chamados de unidades de processamento e há uma grande interconexão entre estas unidades. Estas redes possuem algumas propriedades particulares tais como: a capacidade para adaptação e aprendizagem, para a generalização, organização de dados, ou reconhecimento de padrões. Cada operação pode ser realizada com recurso a processamento paralelo. As redes têm ainda a capacidade de produzir saídas razoáveis para entradas que não foram utilizadas no processo de treino da RNA [1] [2].
Muitas das propriedades acima mencionadas podem ser atribuídas também a modelos computacionais existentes que não são baseados em redes neuronais, a questão central a saber é em que casos as redes neuronais são mais eficazes do que os modelos existentes. As RNA são resultado de investigações académicas que envolvem metodologias baseadas em conceitos matemáticos. Possuem aspectos que as diferenciam dos modelos computacionais, tais como a aprendizagem, por exemplo, adaptação a novas situações, habilidade de generalização, a não linearidade das unidades de processamento, interconexões paralelas e tolerância a falhas. Uma RNA consiste num conjunto de unidades de processamento simples que se comunicam entre si, cada unidade de processamento efectua uma simples tarefa: recebe das unidades anteriores ou de unidades externas sinais e utiliza-os para calcular o sinal de saída que é propagado para outras unidades. Numa fase anterior decorre um processo de ajuste dos pesos das sinapses que unem os nós das sucessivas camadas, denominado por processo de treino. O sistema é inerentemente paralelo ou seja muitas unidades podem efectuar o cálculo da saída ao mesmo tempo [1] [3]. As RNA são usualmente caracterizadas em três tipos de unidades: unidades de entrada (inputs) que recebem os sinais exteriores à RNA, unidades de saída (output) que envia sinais para o exterior da RNA, e as unidades escondidas (hidden) em que os sinais de entrada e saída permanecem na RNA sem contacto com o exterior.
As RNA aprendem com exemplos, uma RNA é configurada para uma tarefa especifica como reconhecimento de padrões através de um processo de aprendizagem. Os neurónios biológicos aprendem ajustando as conexões sinápticas entre neurónios.
As redes neuronais mais habituais para a generalidade das situações têm uma arquitectura feed-forward.

A. Redes Feed-Forward

As redes feed-forward são caracterizadas, principalmente, pela representação interna do conhecimento que é capaz de organizar as unidades intermédias ou escondidas, para conseguir qualquer correspondência entre os sinais de entrada e os sinais de saída.
O funcionamento de uma rede feed-forward consiste numa aprendizagem através de um conjunto predefinido de pares entrada/saída dados como exemplo. Estas redes são treinadas por um algoritmo denominado de back-propagation por o erro se propagar da frente para trás num processo que se desenrola em duas fases [1]:

- Primeiro aplicam-se pesos aleatórios aos sinais de entrada como estimulo para a primeira camada (unidade) da rede e propaga-se através das outras camadas até gerar uma saída, compara-se a saída resultante com a saída desejada e calcula-se o valor de erro para cada neurónio da saída.
- A seguir transmitem-se os erros para trás de modo a que todos os neurónios das camadas intermédias que comunicam directamente com as camadas de saída, recebam uma parte do erro de acordo com a sua participação na rede para a saída original. Repete-se este processo camada por camada até a camada de entrada até que todos os neurónios da rede recebam a sua contribuição parta o erro total. Baseando-se no erro recebido, ajustam-se os valores dos pesos das conexões de cada neurónio, de maneira a que na próxima vez que se inserir a mesma entrada, a saída seja mais próxima da desejada, ou seja o erro seja menor.

2. Desenvolvimento e Treino da RNA no Matlab

O Matlab possui um conjunto de ferramentas (toolbox) para o desenvolvimento e manipulação de RNAs. Para criar uma RNA faz-se uso da função *newff*. Os parâmetros de entrada que necessita são: algoritmo de treino, função transferência da camada escondida e da camada de saída, número máximo de iterações, número de camadas escondidas e o valor alvo do erro. Para proceder ao treino da RNA utiliza-se a função *train* que recebe como parâmetros de entrada a estrutura da RNA e as matrizes de entrada e saída.

A. Matrizes de Entrada/Saída

Os caracteres alfanuméricos são representados numa matriz quadrada 5x5. Posteriormente a matriz de cada caractere é transformada numa coluna da matriz de entrada, como se representa na figura 1, criando-se assim uma correspondência biunívoca entre cada coluna da matriz de entrada (figura 2), com um caractere alfanumérico.

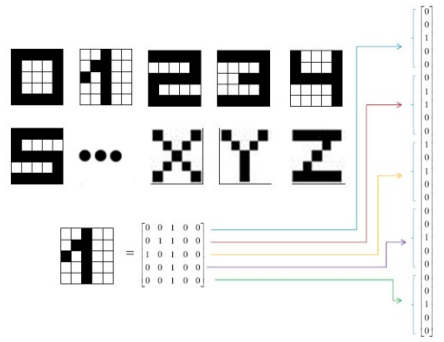

Fig. 1- Representação dos caracteres

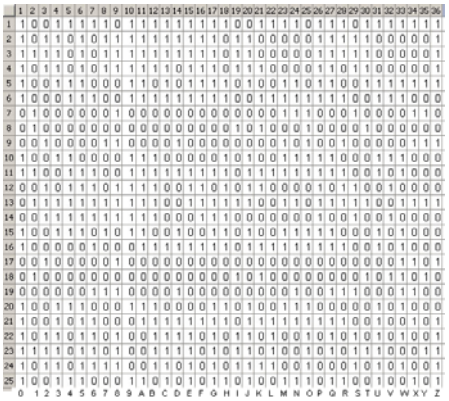

Fig. 2 - Matriz de entrada

Na matriz de saída cada coluna identifica um caractere codificado. Apenas um elemento da coluna está a 1 e os restantes a 0, identificando 1 dos 36 caracteres. Assim para a matriz de saída da figura 3 a primeira coluna tem a 1 apenas o primeiro termo e os restantes a 0, identificando o caractere 0.

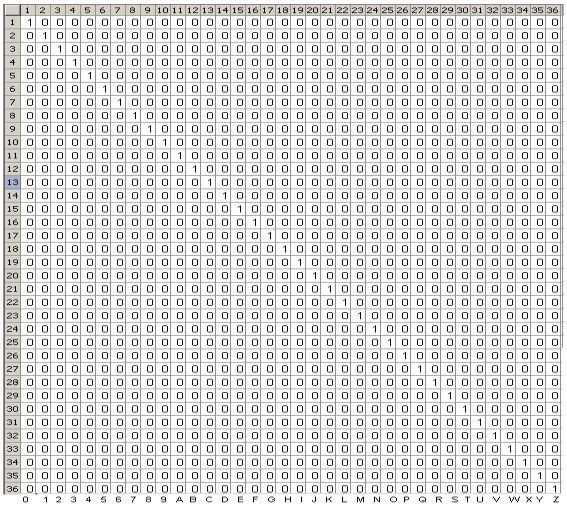

Fig. 3 - Matriz de saída

Cada caractere pode ser representado na entrada mais que uma vez, quanto mais representações houverem de um determinado caractere melhor será identificado o respectivo caractere.

B. Simulação da RNA

Para avaliar a performance da RNA é efectuada uma simulação para os diversos caracteres. Para isso utiliza-se a função *sim*, em que os parâmetros de entrada são a estrutura da RNA previamente treinada, e a matriz coluna do caractere que se pretende simular. A saída da RNA permite identificar o caractere correspondente à entrada como sendo o nó cujo valor mais se aproxima de 1. A RNA tem 25 nós na camada de entrada que corresponde às 25 linhas da matriz de entrada, por sua vez correspondentes aos 5x5 elementos da matriz de representação dos caracteres. A saída contém 36 nós que correspondem ao número de linhas da matriz de saída, que por sua vez correspondem aos 36 caracteres usados. A cada nó corresponde um caractere sendo identificado o caractere que no respectivo nó de saída tenha o valor mais próximo de 1.

C. Invocação de funções no Matlab

Os resultados a seguir apresentados foram desenvolvidos na versão 7.4.0 do Matlab, no entanto para versões posteriores é diferente a invocação das funções apresentadas anteriormente. Para criar a estrutura da RNA utiliza-se a função *newff* da seguinte forma:

net = newff(PR,[S1 S2],{TF1 TF2},BTF), em que:

PR – é uma matriz 25x2 em que na coluna 1 tem o valor mínimo e na coluna 2 tem valor máximo, são o mínimo e o máximo da matriz de entrada;
S1 – é o número de nós na camada escondidas;
S2 – é o número de nós na camada de saída;
TF1 – função transferência na camada escondida;
TF2 – função transferência na camada de saída;
BTF – função de treino da RNA.

Uma vez criada a RNA é treinada com a função *train* da seguinte forma:

Nnet=train(net,P,T), em que:

net – é a estrutura da RNA criada anteriormente;
P – é a matriz de entrada;
T – é a matriz de saída.

Construiu-se a RNA com os seguintes parâmetros:
PR – é uma matriz 25x2 em que na coluna 1 tem o valor zero e na coluna 2 tem valor um, são o mínimo e o máximo da matriz de entrada;
S1 – 70;
S2 – 36;
TF1 – purelin;
TF2 – purelin;
BTF – trainrp.

A RNA foi treinada com os pares de matrizes entrada/saída mencionadas anteriormente.

3. Desenvolvimento da Interface Gráfica – GUI (Graphical User Interface)

Para facilitar o uso da script desenvolvida em Matlab e tornar mais interactivo o processo de treino e simulação da RNA, foi desenvolvido uma interface gráfica com o GUI do Matlab. Este possui um ecrã inicial onde é possível escolher entre treinar ou simular a RNA (figura 4).
Para efectuar o treino da RNA, são apresentadas as várias funções para o treino assim como as funções de transferência da camada escondida e da camada de saída. Também é possível inserir o número de iterações máximo, o número de camadas escondidas e o valor alvo do erro, ou seja, os parâmetros necessários para criar e treinar uma RNA.
Para simular o funcionamento da RNA é apresentada uma matriz 5x5 onde é desenhado o caractere que se pretende para efectuar a simulação, e uma imagem do caractere que a RNA identificou como sendo o caractere inserido, como mostra a figura 4.

4. Resultados Experimentais

Os resultados a seguir apresentados foram treinados todos com o mesmo algoritmo de treino *trainrp***,** uma vez que é a melhor função de treino para problemas baseados em reconhecimento de padrões. Apenas a função transferência da camada escondida e da camada de saída é que foram alteradas. Nos resultados seguintes não são apresentados caracteres sem ruído uma vez que verificou-se que eram correctamente identificados nos casos dos 36 caracteres.

A. Função de transferência purelin na camada escondida e logsig na camada de saída

Nas figuras 5, 6, 7 e 8 estão representadas as simulações dos caracteres ‘3’ e ‘A’ com algum ruído (matriz de caracteres 5x5 com pixéis errados) e com o respectivo caractere identificado. Nestes casos utilizou na camada escondida a função de transferência purelin e na camada de saída a função de transferência logsig.

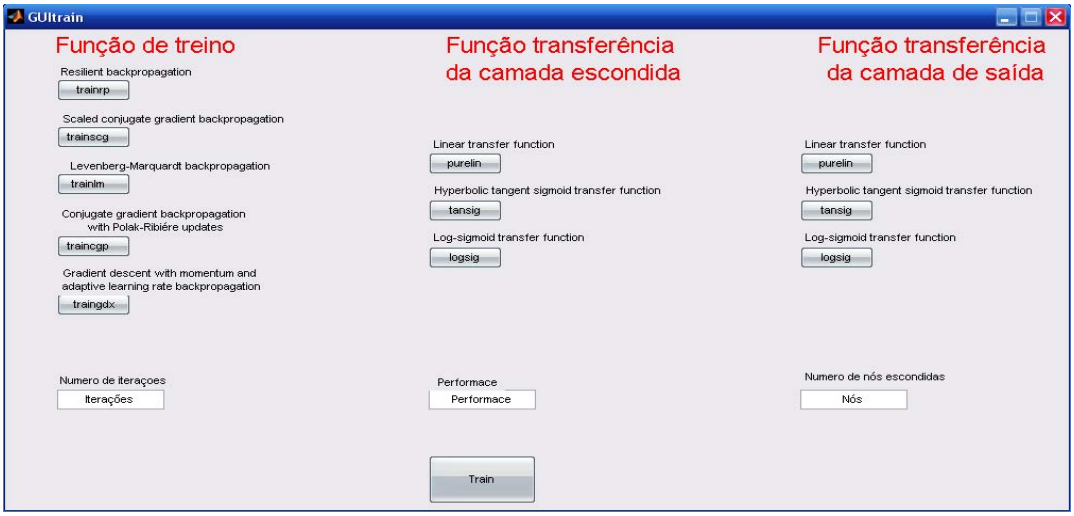

Fig. 4 – Janela principal da interface gráfica

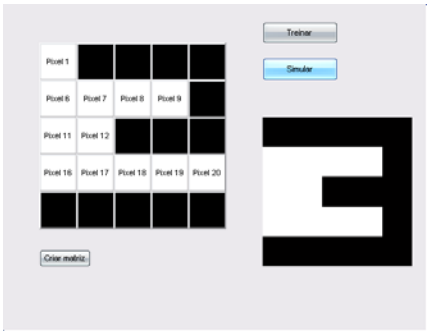

Fig. 5 - Caractere 3 com 2 pixéis errados

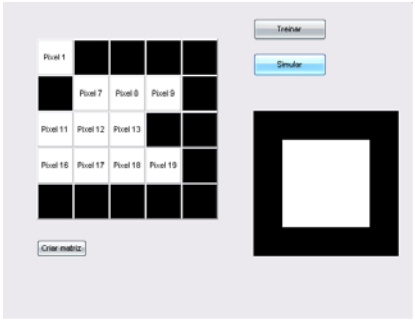

Fig. 6 - Caractere 3 com 3 pixéis errados

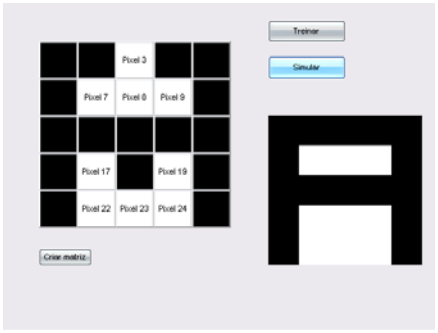

Fig. 7 - Caractere A com 2 pixéis errados

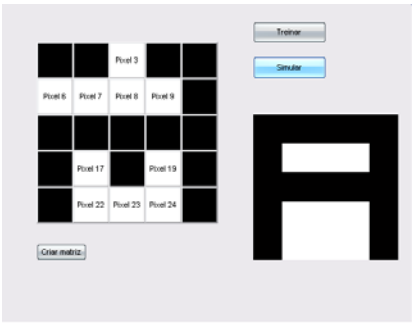

Fig. 8 - Caractere A com 3 pixéis errados

B. Função de transferência purelin na camada escondida e na camada de saída

Nas figuras 9, 10, 11 e 12 estão representadas as simulações dos caracteres '3' e 'A' e respectivo caractere identificado, usando algum ruído na codificação da entrada em que nas camadas escondidas e de saída se utilizou a função de transferência purelin.

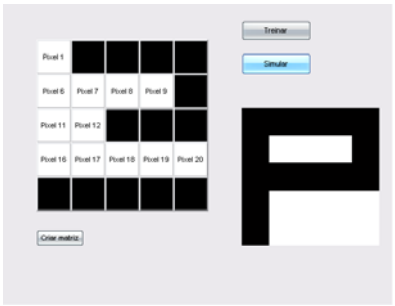

Fig. 9 - Caractere 3 com 2 pixeis errados

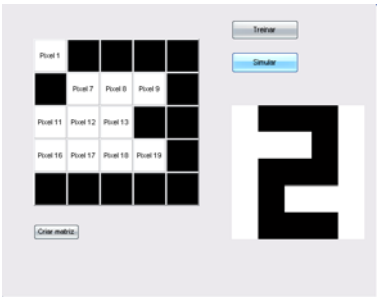

Fig. 10 - Caractere 3 com 3 pixéis errados

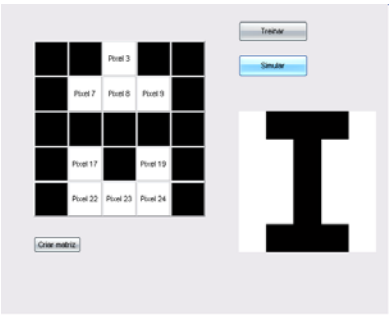

Fig. 11 - Caractere A com 2 pixéis errados

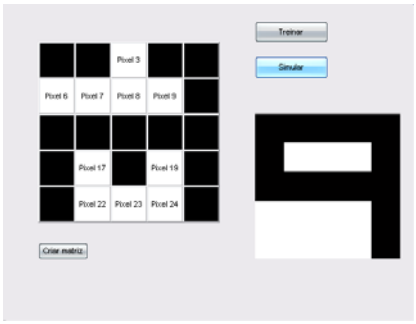

Fig. 12 - Caractere A com 3 pixéis errados

C. Função de transferência tansig na camada escondida e tansig na camada de saída

Nas figuras 5, 6, 7 e 8 estão representadas as simulações dos caracteres '3' e 'A' com algum ruído em que nas camadas escondidas e de saída utilizou-se a função de transferência tansig.

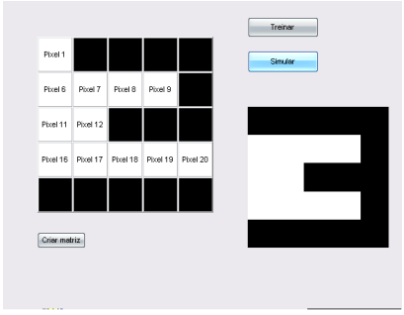

Fig. 13 - Caractere 3 com 2 pixéis errados

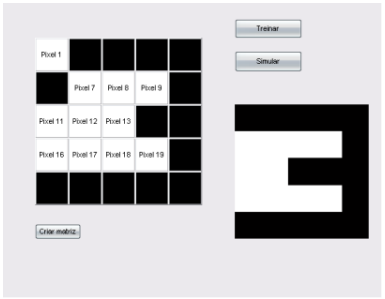

Fig. 14 - Caractere 3 com 3 pixéis errados

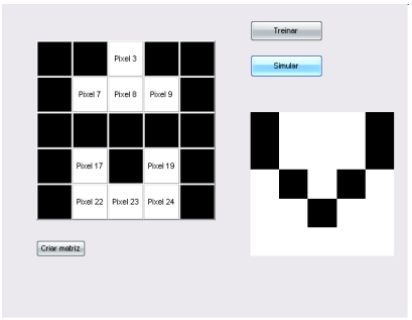

Fig. 15 - Caractere A com 2 pixéis errados

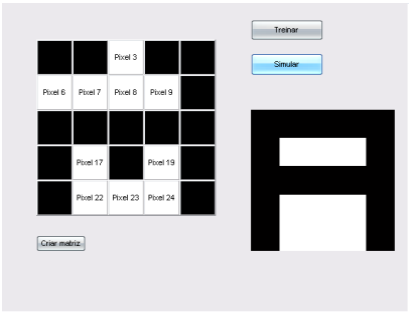

Fig. 16 - Caractere A com 3 pixéis errados

D. Evolução do erro no treino da RNA

Apresenta-se na figura 17 a evolução do erro durante o processo de treino da RNA. Como não foi usado um conjunto diferente para o teste e validação as linhas surgem sobrepostas. Neste caso o treino da RNA acaba quando atinge o número máximo de iterações impostas pelo utilizador, sendo a função de transferência purelin na camada escondida e na camada de saída.

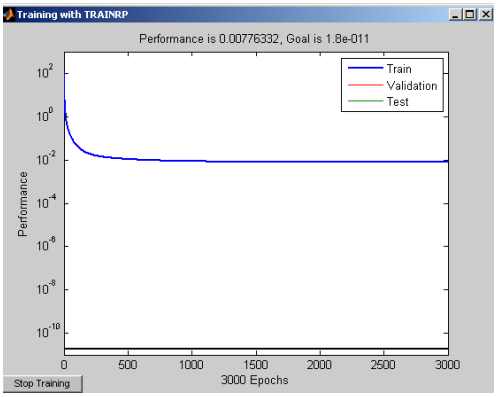

Fig. 17 - Performance da RNA (purelin)

Na figura 18 apresenta-se o gráfico em que o treino pára em apenas 79 iterações porque foi atingido o gradiente mínimo, imposto pela função transferência de saída. Neste caso a função de transferência é purelin na camada escondida e logsig na camada de saída.

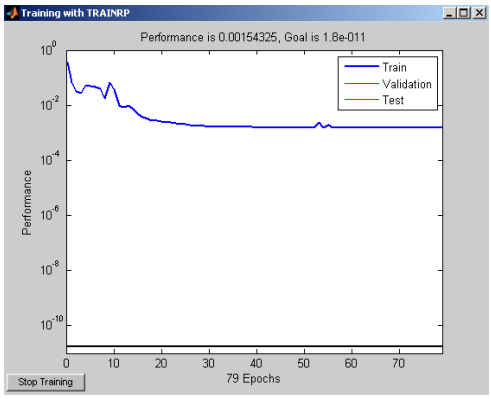

Fig. 18 - Performance da RNA (purelin/logsig)

4. Conclusões

Os resultados obtidos mostram que nos três exemplos apresentados a RNA consegue identificar correctamente o caractere se este não tiver qualquer tipo de ruído, no entanto para caracteres com ruído a RNA com a função logsig na camada de saída apresenta melhores resultados que a RNA com a função purelin. Assim conclui-se que esta RNA aprendeu muito bem os caracteres da matriz de entrada mas não aprendeu a generalizar para diferentes casos que possam aparecer.

Os resultados apresentados, ajudaram a perceber estatisticamente, ainda que com um universo de amostras pequeno, a maneira como para diferentes funções de transferência, a rede neuronal aprende e se comporta. Combinando estas funções de transferência da camada escondida e da camada de saída, concluiu-se que as combinações pureline\logsig, e tansig\tansig apresentam maior eficácia que o exemplo feito com as funções pureline\pureline (camada de entrada\camada de saída).

Em comparação com os resultados obtidos em [4], os resultados aqui apresentados ajudaram a entender que para diferentes tipos de treino e as suas possíveis variações, como o número de nós na camada escondida ou valor alvo para o erro, a rede consegue aprender até ao ponto de identificar caracteres mal definidos (com ruído). Os valores obtidos na camada de saída que servem para identificar o caractere a escolher são em alguns casos melhor que os obtidos em [4], e para outros caracteres piores, ou mesmo errados quando a rede utilizada em [4] acertou. Assim, chegou-se á conclusão que uma rede pode estar bem treinada e acertar num caractere com muito ruído, e a mesma rede com o mesmo treino falhar um outro caractere com pouco ruído. Sendo assim, como para uma rede a probabilidade do acerto é diferente para cada caractere ou diferentes ruídos no mesmo caractere, pode ser feito um treino já com erros introduzidos, para a rede aprender a reconhecer os caracteres com ruído e assim aumentar as probabilidades de acertar.

Este artigo descreve um trabalho académico que para além de mostrar as potencialidades das Redes Neuronais Artificiais, mostrou como uma ferramenta como o Matlab e a possibilidade de criar interfaces gráficas para utilizadores pode ser uma grande ajuda para resolver problemas de engenharia ou outras áreas de trabalho.

Referências

[1] Krose Ben, Smagt Patrick, “An Introduction to Neural Networks”, 1996 The University of Amsterdam.

[2] Teixeira, João Paulo, “Introdução as Redes Neuronais Artificiais - Redes feed-forward em Matlab”, 2009, Departamento de Electrotecnia, ESTiG - Instituto Politécnico de Bragança.

[3] MacKay David, “Information Theory, Pattern Recognition and Neural Networks”, 2002.

[4] João Paulo Teixeira, José Batista, Anildio Toca, João Gonçalves, Filipe Pereira, “Reconhecimento de Caracteres com Rede Neuronal Artificial para o Reconhecimento de Caracteres com Interface Gráfica”, Engenharia`2009 – Inovação e Desenvolvimento, 5.ª Conferência de Engenharia, UBI, Covilhã, 25 a 27 Novembro de 2009.