Trabalho 2: Reconhecimento de Caracteres

Pedro Victor Dos Santos Matias, 21601225

pvsm@icomp.ufam.edu.br

I. Introdução

O presente trabalho é sobre um problema reconhecimento de caracteres digitalizados. Na qual são dado algumas classes de 3 fontes para realizar o treinamento. A rede neural projetada é baseada no modelo Perceptron de Rosenblatt (1958) de uma camada, mas com múltiplos neurônios.

O problema apresenta 7 classes de caracteres A, B, C, D, E, J, K com 3 fontes diferentes para representá-lo. A duas bases uma de treinamento, a outra de teste, cada uma apresenta as letras como uma imagem binária de resolução 9x7. Assim será realizado o treinamento e posteriormente um teste de generalização para analisar se a rede consegue discernir padrões com ruídos e oclusões de bits (base de teste).

Este documento está organizado em 6 seções. Na segunda, será realizado uma revisão bibliográfica sobre as técnicas de reconhecimento de padrões . No capítulo 3 é descrita a arquitetura geral da rede e o algoritmo de treinamento para solução do problema. Os resultados, como gráficos e tabelas são apresentados na seção 4. Na seção de conclusão é realizado uma as considerações finais entre o problema e os resultados obtidos. E por último um apêndice com a listagem dos códigos em MATLAB.

II. Revisão Bibliográfica

As redes neurais artificiais são sistemas computacionais baseados em modelos matemáticos e possem diversos aplicação, sendo uma delas reconhecimento de padrões. Para esse trabalho realizou uma busca exploratório em torno desta função e as aplicações práticas mais similar são as averiguação em placas de veículos e em identificação de caracteres manuscritos.

Dieguez et al. (2010) desenvolveu um projeto de reconhecimento de caracteres de placa de automóveis usando uma rede neural do tipo Perceptron Multicamadas (MLP). Nesse trabalho ele apresenta duas técnicas de extração de descritores, a primeira usa um mapa de bits puro junto com as projeções (horizontal e vertical) para serem as entradas da rede. A segunda sintetiza momentos, em duas dimensões, sobre o mapa de bits sobre modelos estatísticos. Por fim ele realiza considerações sobre as duas soluções em relação a desempenho e facilidade de aprendizado da rede. Foi concluído que a solução de melhor desempenho é a da técnica de *Momentos*.

Outro trabalho neste mesmo segmento é o de Kocer and Cevik (2011) apresentado no contexto da necessidade de controle devido o aumento de veículos no tráfego. Nesta referência foi usado 259 imagens de placas, com a digitalização e segmentação para extração dos caracteres para, então, ser feito o reconhecimento, também, com um MLP feed-forward com o treinamento usando o backpropagation.

III. Metodologia

Neste capítulo descrevemos os procedimentos metodológicos que foram realizados na concepção do modelo, especificação da arquitetura e o algoritmo de treinamento utilizado.

Classifica-se este estudo como explicativo devido a correlação das aulas teóricas de Inteligência Artificial Aplicada, ministrada pelo professor Dr. Cícero Fernandes, com um problema prático de reconhecimento de padrões (caracteres).

Do ponto de vista dos procedimentos, o trabalho em questão faz o uso de métodos experimentais. Como a representação dos caracteres do problema como uma imagem digital binária, isto é uma matriz $f[MxN]$, onde M é o numero de linhas e N o de colunas, e cada valor de $f(x,y)$ é a intensidade do pixel. Como no trabalho anterior o neurônio é representado por uma estrutura matemática descrita e codificada em MATLAB.

I. Problema

1. O problema apresenta um conjunto de padrões de entrada para serem treinados usando um perceptron. Os caracteres estão apresentados na figura 1.

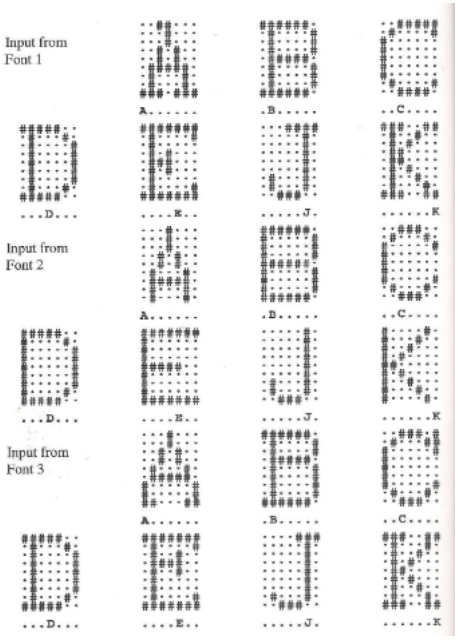


Figura 1: Padrões para treinamento

2. Pede-se para tabelar a saída de acordo com os padrões da figura 1.

3. Tabelar também a saída para os padrões de teste na figura 2. Nessa figura, o símbolo *o* representa um pixel que estava `on` no padrão original e que agora está `off`. Já o símbolo @ representa um pixel que estava `off` no padrão original e que agora está `on`.

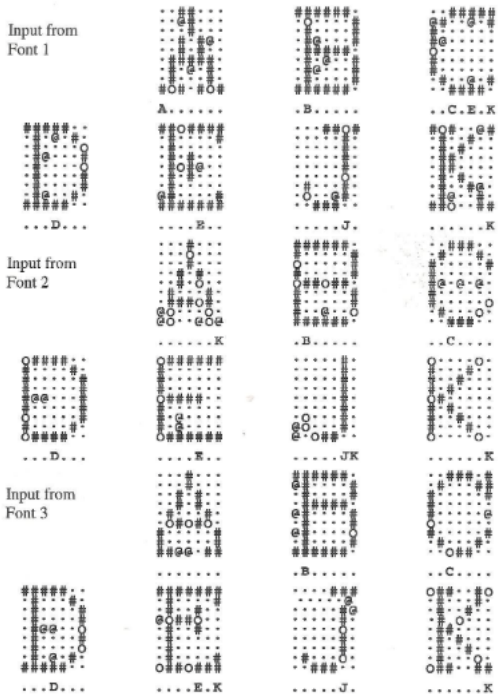


Figura 2: Padrões para testar

II. Arquitetura

A arquitetura projetada é de uma rede Perceptron com múltiplas saídas aplicado ao problema de reconhecimento de caracteres binários. São ao todo 7 padrões entrada que representam a classe de caracteres $\{A, B, C, D, E, J, K\}$ que foram definidas a partir de 3 fontes diferentes. Cada padrão possui uma resolução espacial de 9x7 pixels, e se tratando de uma imagem binária só ha duas intensidades discerníveis.

O Perceptron possui como regra de aprendizagem: a atualização dos pesos de um neurônio pelo erro gerado no treinamento. Assim a rede proposta possuirá apenas uma camada 7 neurônios de saída, cada um relacionado a uma elemento da classe de padrões. Para cada classe os resultados esperados da rede deve estar na forma:

Para cada classe:

Para cada classe é esperado as seguintes saídas:

$$A=\begin{bmatrix}1\\0\\0\\0\\0\\0\\0\end{bmatrix} \quad B=\begin{bmatrix}0\\1\\0\\0\\0\\0\\0\end{bmatrix} \quad C=\begin{bmatrix}0\\0\\1\\0\\0\\0\\0\end{bmatrix} \quad D=\begin{bmatrix}0\\0\\0\\1\\0\\0\\0\end{bmatrix} \quad E=\begin{bmatrix}0\\0\\0\\0\\1\\0\\0\end{bmatrix} \quad J=\begin{bmatrix}0\\0\\0\\0\\0\\1\\0\end{bmatrix} \quad K=\begin{bmatrix}0\\0\\0\\0\\0\\0\\1\end{bmatrix}$$

As características da arquitetura foram definidas seguindo o procedimento apresentado no livro de Hagan et al. (1997) que apresenta 3 passos para determinação da arquitetura

- O vetor **p** de entradas possui **R** elementos, cada um correspondendo a uma pixel dos padrões da classe. Como a resolução é 9×7.Logo:

$$\mathbf{p_i} \in \mathbb{R}^{R\times 1}; i = 1, \cdots, R; R = 63$$

- O número S de neurônios define o numero de elementos do vetor saída **a**. Podemos escrever como um vetor coluna de 7 elementos.

$$\mathbf{a} \in \mathbb{R}^{S\times 1}; S = 7$$

- Cada classe é representada por um elemento do vetor de saídas podendo ser 1, pertencente à classe, ou 0, não-pertencente. Assim para realizar esse processo usaremos a função de ativação usada é a hardlim do matlab.

$$hardlim(n) = \begin{cases} 1 & \text{se } n \geq 0 \\ 0 & \text{caso contrário} \end{cases}$$

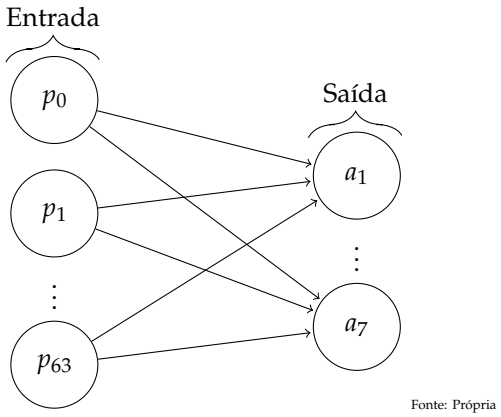
Haverá um limite de decisão para cada neurônio, para classificar as entradas nas 7 diferentes categorias. O valor de cada elemento do vetor **a** de saída seguirá o seguinte:

$$a_i = hardlim(\mathbf{W}p_i + b_i)$$

As matrizes de pesos **W** e vetor de polarização segue a definição:

$$W \in \mathbb{R}^{S\times R}; S = 7; R = 63$$

$$\mathbf{b} \in \mathbb{R}^{S\times 1}; S = 7$$



Fonte: Própria

Figura 3: Gráfico da Rede Perceptron com 63 entrada 7 unidades de saída. Segue $a_i = hardlim(\mathbf{W}p_i + b_i)$

III. Algoritmo

O algoritmo desenvolvido deve definir uma hiperplano (fronteira de decisão) em um espaço com 63 dimensões para realizar o reconhecimento das classes. O script realiza basicamente os seguintes passos:

1. Definição das dimensões da rede, informando o número de de entradas e de neurônios.
2. Inicialização da matriz de pesos e polarização com valores aleatórios usando uma distribuição normal em torno do valor zero.
3. Execução da função de aprendizagem que executa de forma iterativa a sub-função de teste da rede que retorna o erro, isto é, a diferença entre o valor desejado , `t`, e o desejado, `a`.

No treinamento da rede a função de aprendizagem executa um ciclo de que so finaliza depois que todas as amostras do treino tenham sido validadas sem nenhum erro. Dentro a outro laço que extrai os pares, entradas e saídas desejadas, da matriz de amostras aplicando a função de teste. Para cada amostra que for detectado um erro é realizado o ajuste dos pesos e da polarização, incrementando o contador de épocas da rede.

IV. Resultados

Como resultados tivemos o script implementado em matlab, `classificadorFinal.m` que está em apêndice com ele há os arquivos da base de treino e teste da rede. Esta seção foi subdividida apresentando os dados antes do treinamento e após, assim como os resultados de teste de generalização.

I. Inicialização da Rede

O processo de inicialização da rede busca gerar valores aleatórios para a matriz de pesos e de polarização , os valores de cada um estão respectivamente nas tabelas 3 e 2.

Para as amostras de entradas temos 21, cada classe em 3 fontes diferentes para representar os mesmos caracteres. Assim a saída desejada da rede seguindo a imagem 1 está representada na tabela abaixo.

	Fonte 1						
	t_1	t_2	t_3	t_4	t_5	t_6	t_7
S_1	1	0	0	0	0	0	0
S_2	0	1	0	0	0	0	0
S_3	0	0	1	0	0	0	0
S_4	0	0	0	1	0	0	0
S_5	0	0	0	0	1	0	0
S_6	0	0	0	0	0	1	0
S_7	0	0	0	0	0	0	1

	Fonte 2						
	t_1	t_2	t_3	t_4	t_5	t_6	t_7
S_1	1	0	0	0	0	0	0
S_2	0	1	0	0	0	0	0
S_3	0	0	1	0	0	0	0
S_4	0	0	0	1	0	0	0
S_5	0	0	0	0	1	0	0
S_6	0	0	0	0	0	1	0
S_7	0	0	0	0	0	0	1

	Fonte 3						
	t_1	t_2	t_3	t_4	t_5	t_6	t_7
S_1	1	0	0	0	0	0	0
S_2	0	1	0	0	0	0	0
S_3	0	0	1	0	0	0	0
S_4	0	0	0	1	0	0	0

S_5	0	0	0	0	1	0	0
S_6	0	0	0	0	0	1	0
S_7	0	0	0	0	0	0	1

Tabela 1: Tabela de saídas esperadas para as amostras de treinamento

1. Polarizações iniciais - b

	B_s
S_1	-0,8782
S_2	-0,4496
S_3	-1,0989
S_4	0,5149
S_5	-0,6176
S_6	0,5514
S_7	1,4132

Tabela 2: Valores obtidos para o vetor **b** após a inicialização da rede

2. Pesos iniciais - W

	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9
S_1	-1,1860	-0,4677	-0,8309	1,6182	-0,4468	-0,5960	-1,4864	-0,1401	-0,0224
S_2	-0,2907	-0,2418	1,0720	-0,0272	-0,6468	-0,6038	0,5755	-0,0254	-0,0341
S_3	0,2153	-1,3786	1,3249	0,1769	-0,4312	1,4077	0,1132	0,0844	-1,7322
S_4	1,7766	0,9367	0,7163	-1,3446	-0,7195	0,2390	1,3867	-2,0234	-0,8458
S_5	-0,6087	-0,1641	-0,0465	0,8490	-1,0142	2,8965	1,8903	0,4252	0,2838
S_6	-1,9930	0,1431	-0,5755	-0,6487	1,0147	-0,7010	-1,6046	-1,4251	0,4277
S_7	-0,3043	-0,7712	-0,2347	1,4526	0,5752	1,1951	-0,5236	1,0491	-0,1216

	W_{10}	W_{11}	W_{12}	W_{13}	W_{14}	W_{15}	W_{16}	W_{17}	W_{18}
S_1	0,3439	-0,5369	-1,2342	-0,4622	1,6759	-0,8393	-0,9373	-1,7643	0,8230
S_2	-0,8828	-1,3657	-1,2452	-0,2929	0,7395	-0,3874	-0,6586	0,7700	-1,3513
S_3	1,1010	0,7810	-1,0100	0,6133	-1,4036	0,1157	-0,1451	0,6344	0,3340
S_4	-0,6682	-0,1974	0,4638	-0,2057	-0,7005	-0,6098	0,9124	0,4687	-0,0839
S_5	0,3438	0,1932	0,3583	1,5858	1,2268	0,2351	-0,3171	-0,1258	-2,0570
S_6	-1,4649	-1,0659	1,2543	-2,0450	0,9741	0,0493	-1,0502	-1,5020	0,4851
S_7	-0,7624	1,0044	-0,8695	-1,9972	-0,2516	-1,1461	0,2496	0,1480	-0,5389

	W_{19}	W_{20}	W_{21}	W_{22}	W_{23}	W_{24}	W_{25}	W_{26}	W_{27}
S_1	1,6527	0,1472	-0,5199	-0,3209	-0,7280	1,6351	-0,3547	-0,9323	-0,1012
S_2	-0,9028	1,8932	-0,3733	0,6984	-0,7174	0,6658	-1,6226	-2,3960	1,3961
S_3	0,1194	-1,6106	0,8309	0,5537	-0,7830	-0,8634	-0,1311	-0,0535	-0,2028

S_4	2,5807	0,5019	1,0315	1,0278	-0,0715	0,8707	-0,5880	2,5178	-0,8390
S_5	-0,2174	-0,1374	-0,4228	1,2142	2,2374	-1,9322	-1,2610	-0,7060	-1,6364
S_6	-1,4057	-0,4213	0,1503	0,1487	1,9844	0,5639	-1,7217	-0,9710	-0,8178
S_7	-1,8272	-1,7652	1,2619	-1,6493	0,2348	1,9803	-0,0555	-0,6836	0,1758

	W_{28}	W_{29}	W_{30}	W_{31}	W_{32}	W_{33}	W_{34}	W_{35}	W_{36}
S_1	-0,2840	0,0061	-1,2357	0,1337	0,6539	-0,0651	0,4865	-2,2939	1,0899
S_2	1,1496	-2,0158	-0,1294	-0,8341	0,1933	0,0026	-0,1920	1,9078	0,0474
S_3	-0,2990	-0,8307	1,4636	-1,4944	-1,7220	0,1291	0,8914	-0,8851	-0,4963
S_4	0,0952	1,4468	-0,1117	-0,2579	-1,0335	-0,3106	0,4182	0,8616	-1,1718
S_5	1,0841	-0,4193	1,3826	-1,5399	1,4603	-1,6264	-0,4071	-1,1509	-0,9396
S_6	-0,0572	0,7691	1,6077	-0,2053	0,5943	0,0473	1,3571	1,1105	0,3001
S_7	-1,5972	0,4902	-0,5955	0,3225	-1,5135	0,3192	-0,2658	-0,7134	0,0076

	W_{37}	W_{38}	W_{39}	W_{40}	W_{41}	W_{42}	W_{43}	W_{44}	W_{45}
S_1	0,4147	-0,1977	0,8515	1,8156	-0,0959	-1,0626	0,6307	-1,0141	0,0022
S_2	1,3949	0,1557	0,5899	-0,6641	0,1940	-0,6532	0,3728	-0,5135	1,1365
S_3	-0,4108	-0,5941	1,0016	1,4821	0,7241	-0,0055	0,0607	-1,0459	0,0230
S_4	0,3784	-0,7842	0,0861	0,7093	1,1314	0,2608	-1,7451	1,7531	-0,6186
S_5	0,1577	-1,2336	-0,1030	-1,2727	-0,3731	0,6994	0,3553	-1,1447	0,1446
S_6	0,9086	-0,5738	-2,0804	-0,5339	-0,4887	-0,0821	0,0044	1,2011	-0,2227
S_7	0,5160	-0,6655	-0,1977	0,9148	0,4365	-0,8009	-0,6063	0,1476	0,3615

	W_{46}	W_{47}	W_{48}	W_{49}	W_{50}	W_{51}	W_{52}	W_{53}	W_{54}
S_1	-0,0534	0,6586	-0,7156	-0,0613	-0,5103	0,8856	-0,9972	-1,1411	-0,0777
S_2	-0,6058	0,7136	-2,4746	0,1842	0,0003	-0,7163	-1,4533	0,6856	-0,3561
S_3	-1,3650	0,4948	-0,1283	-0,5510	0,0577	-0,2238	-1,7357	-2,0778	-0,1724
S_4	-0,0716	0,6272	0,1981	0,5124	-0,7321	0,1117	-0,5810	2,4120	-1,1558
S_5	0,2484	-1,4452	0,8048	0,8606	0,2438	0,1357	-1,1434	-0,2843	0,3836
S_6	-0,6769	-0,4372	0,4230	-0,0491	-2,4074	1,1849	0,8470	0,8205	-0,2245
S_7	-0,9679	1,1359	-0,3613	-1,5123	-0,0891	1,1885	-1,6831	0,7382	0,2836

	W_{55}	W_{56}	W_{57}	W_{58}	W_{59}	W_{60}	W_{61}	W_{62}	W_{63}
S_1	1,1171	-0,5957	0,9070	-0,1067	-0,2598	-0,5746	2,7760	0,7797	1,7035
S_2	0,7562	-2,0895	-0,0844	0,3370	-0,1086	0,5901	0,1257	-1,5721	0,9524
S_3	-0,6130	0,8281	1,0191	-0,2644	0,2914	1,0234	1,7119	1,0841	-0,7230
S_4	0,1544	-0,3880	-0,3478	0,0964	-0,4000	-0,3553	-0,4938	0,8632	-2,1681
S_5	0,2147	-0,5686	0,3742	1,4429	2,2611	1,9845	-0,3765	1,1205	0,3498
S_6	1,6072	0,1561	0,5184	0,8860	0,6845	1,0301	0,9252	-0,0103	-1,2105
S_7	-0,2426	-0,0716	0,8229	-0,9698	-0,9681	-0,9955	-1,0504	-1,3232	-0,0562

Tabela 3: Valores obtidos para a matriz **W** com a inicialização da rede

II. Treinamento e Generalização

Após o treinamento com **36 épocas**, quantidade de atualizações nos parâmetros da rede, foi realizado testes com 21 amostras, referentes descritas no capítulo da metodologia, mais especificadamente na imagem 2. Esse teste tem como objetivo mostrar a capacidade da rede de generalizar o reconhecimento, permitindo detectar caracteres ruidosos e/ou com oclusão da dados. Os resultados dos parâmetros da rede, peso e polarização, obtidos após o treinamento e a saída após o testes são apresentados nas tabelas abaixo ().

1. Polarização Final - b_{new}

	B_s
S_1	-2,8782
S_2	-2,4496
S_3	-3,0989
S_4	-1,4851
S_5	-3,6176
S_6	-1,4486
S_7	0,4132

Tabela 4: Valores obtidos para o vetor **b** após o treinamento

2. Pesos Finais - - W_{new}

	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9
S_1	-3,1860	-0,4677	-0,8309	1,6182	-0,4468	-2,5960	0,5136	-0,1401	-0,0224
S_2	1,7093	1,7582	1,0720	-2,0272	-0,6468	5,3962	-3,4245	1,9746	-0,0341
S_3	-1,7847	-3,3786	1,3249	0,1769	1,5688	-0,5923	4,1132	-1,9156	4,2678
S_4	3,7766	2,9367	-1,2837	-3,3446	-0,7195	-3,7610	1,3867	-0,0234	-0,8458
S_5	-1,6087	0,8359	0,9535	3,8490	1,9858	-0,1035	4,8903	1,4252	1,2838
S_6	-1,9930	0,1431	-4,5755	-2,6487	1,0147	1,2990	2,3954	0,5749	-1,5723
S_7	4,6957	2,2288	-1,2347	-3,5474	-0,4248	4,1951	0,4764	4,0491	0,8784

	W_{10}	W_{11}	W_{12}	W_{13}	W_{14}	W_{15}	W_{16}	W_{17}	W_{18}
S_1	2,3439	1,4631	-1,2342	1,5378	1,6759	-0,8393	-0,9373	0,2357	0,8230
S_2	1,1172	-1,3657	0,7548	-4,2929	6,7395	-0,3874	1,3414	0,7700	0,6487
S_3	3,1010	0,7810	0,9900	2,6133	0,5964	2,1157	1,8549	2,6344	0,3340
S_4	1,3318	-0,1974	2,4638	5,7943	-4,7005	-0,6098	2,9124	2,4687	-0,0839
S_5	3,3438	3,1932	-0,6417	0,5858	4,2268	1,2351	0,6829	2,8742	-3,0570
S_6	0,5351	-1,0659	3,2543	-0,0450	0,9741	0,0493	-1,0502	0,4980	0,4851
S_7	0,2376	-1,9956	4,1305	-0,9972	-1,2516	-0,1461	3,2496	-0,8520	2,4611

	W_{19}	W_{20}	W_{21}	W_{22}	W_{23}	W_{24}	W_{25}	W_{26}	W_{27}
S_1	3,6527	2,1472	-0,5199	-0,3209	-0,7280	1,6351	1,6453	1,0677	1,8988

S_2	-2,9028	3,8932	5,6267	0,6984	1,2826	2,6658	0,3774	-0,3960	7,3961
S_3	2,1194	-1,6106	2,8309	2,5537	1,2170	-0,8634	1,8689	-0,0535	-0,2028
S_4	4,5807	2,5019	3,0315	1,0278	1,9285	-1,1293	-2,5880	0,5178	-0,8390
S_5	2,7826	0,8626	-1,4228	2,2142	3,2374	-0,9322	3,7390	4,2940	-0,6364
S_6	0,5943	3,5787	-1,8497	0,1487	1,9844	0,5639	0,2783	-0,9710	3,1822
S_7	-2,8272	-0,7652	2,2619	-0,6493	3,2348	2,9803	0,9445	-3,6836	1,1758

	W_{28}	W_{29}	W_{30}	W_{31}	W_{32}	W_{33}	W_{34}	W_{35}	W_{36}
S_1	-0,2840	0,0061	-3,2357	0,1337	0,6539	-0,0651	0,4865	-0,2939	1,0899
S_2	3,1496	-2,0158	1,8706	5,1659	6,1933	6,0026	5,8080	1,9078	0,0474
S_3	-2,2990	1,1693	1,4636	-3,4944	-1,7220	-1,8709	-1,1086	-0,8851	1,5037
S_4	6,0952	1,4468	-0,1117	-4,2579	-3,0335	-2,3106	0,4182	6,8616	-1,1718
S_5	0,0841	0,5807	0,3826	-2,5399	4,4603	-0,6264	-3,4071	1,8491	0,0604
S_6	-0,0572	0,7691	1,6077	-2,2053	0,5943	-1,9527	3,3571	3,1105	0,3001
S_7	-0,5972	1,4902	2,4045	1,3225	-0,5135	-0,6808	-1,2658	0,2866	1,0076

	W_{37}	W_{38}	W_{39}	W_{40}	W_{41}	W_{42}	W_{43}	W_{44}	W_{45}
S_1	0,4147	-0,1977	2,8515	3,8156	1,9041	-1,0626	0,6307	-1,0141	2,0022
S_2	1,3949	0,1557	-1,4101	-0,6641	0,1940	5,3468	-1,6272	1,4865	3,1365
S_3	-0,4108	-0,5941	1,0016	1,4821	-1,2759	-2,0055	2,0607	-3,0459	2,0230
S_4	0,3784	-0,7842	-1,9139	0,7093	1,1314	4,2608	-1,7451	1,7531	1,3814
S_5	1,1577	-0,2336	0,8970	1,7273	0,6269	-0,3006	1,3553	-2,1447	3,1446
S_6	-1,0914	-0,5738	-2,0804	-0,5339	1,5113	-0,0821	0,0044	1,2011	1,7773
S_7	-0,4840	-1,6655	-1,1977	-2,0852	-2,5635	0,1991	-1,6063	1,1476	1,3615

	W_{46}	W_{47}	W_{48}	W_{49}	W_{50}	W_{51}	W_{52}	W_{53}	W_{54}
S_1	-0,0534	2,6586	1,2844	-0,0613	-0,5103	0,8856	1,0028	0,8589	-0,0777
S_2	1,3942	2,7136	-0,4746	4,1842	0,0003	-0,7163	0,5467	2,6856	1,6439
S_3	0,6350	2,4948	-2,1283	1,4490	-1,9423	1,7762	0,2643	-0,0778	1,8276
S_4	1,9284	2,6272	0,1981	2,5124	1,2679	-1,8883	1,4190	4,4120	0,8442
S_5	1,2484	-2,4452	1,8048	-0,1394	1,2438	-0,8643	1,8566	2,7157	1,3836
S_6	1,3231	1,5628	2,4230	-2,0491	-0,4074	-0,8151	2,8470	2,8205	1,7755
S_7	2,0321	4,1359	-1,3613	-2,5123	0,9109	0,1885	-0,6831	1,7382	3,2836

	W_{55}	W_{56}	W_{57}	W_{58}	W_{59}	W_{60}	W_{61}	W_{62}	W_{63}
S_1	3,1171	-0,5957	-1,0930	-0,1067	-0,2598	-0,5746	2,7760	-1,2203	3,7035
S_2	-3,2438	1,9105	-0,0844	0,3370	-0,1086	0,5901	0,1257	2,4279	-3,0476
S_3	-0,6130	0,8281	-2,9809	-4,2644	-1,7086	1,0234	-0,2881	-0,9159	-0,7230
S_4	4,1544	-4,3880	-0,3478	0,0964	-2,4000	-0,3553	-2,4938	-5,1368	-4,1681
S_5	-2,7853	2,4314	-0,6258	2,4429	1,2611	4,9845	2,6235	0,1205	5,3498
S_6	1,6072	0,1561	-1,4816	-1,1140	-1,3155	1,0301	-1,0748	-2,0103	-1,2105

S_7	0,7574	-3,0716	1,8229	-1,9698	-1,9681	-1,9955	-4,0504	-2,3232	-1,0562

Tabela 5: Valores obtidos para a matriz **W** após o treinamento

3. Saída do teste

	Fonte 1						
	a_1	a_2	a_3	a_4	a_5	a_6	a_7
S_1	1	0	0	0	0	0	0
S_2	0	1	0	0	0	0	0
S_3	0	0	1	0	0	0	0
S_4	0	0	0	1	0	0	0
S_5	0	0	0	0	1	0	0
S_6	0	0	0	0	0	1	0
S_7	0	0	0	0	0	0	1

	Fonte 2						
	a_1	a_2	a_3	a_4	a_5	a_6	a_7
S_1	1	0	0	0	0	0	0
S_2	0	1	0	0	0	0	0
S_3	0	0	1	0	0	0	0
S_4	0	0	0	1	0	0	0
S_5	0	0	0	0	1	0	0
S_6	0	0	0	0	0	1	0
S_7	0	0	0	0	0	0	1

	Fonte 3						
	a_1	a_2	a_3	a_4	a_5	a_6	a_7
S_1	1	0	0	0	0	0	0
S_2	0	1	0	0	0	0	0
S_3	0	0	1	0	0	0	0
S_4	0	0	0	1	0	0	0
S_5	0	0	0	0	1	0	0
S_6	0	0	0	0	0	1	0
S_7	0	0	0	0	0	0	1

Tabela 6: Tabela de saídas obtidas para as amostras do teste de generalização da rede

V. Conclusão

Neste trabalho realizamos o estudo e implementação de uma rede neural de uma camada com 7 neurônios para classificar 7 classes de caracteres com 3 fontes diferentes. Os resultados do treinamento obtidos foram satisfatórios pois com aplicação de 21 amostras, 3 para cada classe, apresentando ruídos e oclusão de bits foi obtido 100% de acerto, significando que a rede treinada tem uma boa capacidade de generalização dentro dos limites apresentados. Contudo podemos afirmar que o espaço de amostras é muito pequeno para garantir que a rede sempre vai funcionar, assim como devemos levar em consideração a resolução dos caracteres que não foi feito nenhum processamento de imagem como segmentação e digitalização, até os defeitos já estavam prontos. Mas os resultados são suficientes para os problemas apresentados.

Referências

A. A. Dieguez, M. R. Petraglia, S. B. Villas-Boas, and F. L. de Mello. Reconhecimento de caracteres de placa veicular usando redes neurais. *Trabalho de conclusão de curso, Universidade Federal do Rio de Janeiro, Rio de Janeiro*, 2010.

M. T. Hagan, H. B. Demuth, M. Hudson, et al. Neural network design. 1997.

H. E. Kocer and K. K. Cevik. Artificial neural networks based vehicle license plate recognition. *Procedia Computer Science*, 3:1033–1037, 2011.

F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

VI. Apêndice

Neste Apêndice encontram-se os códigos fonte dos scripts e funções implementados em MATLAB utilizadas neste trabalho para realizar a especificação e treinamento da rede, desenvolvido para realizar a classificação de peças de acordo com duas características já mensuradas.

I. Script de Reconhecimento

```
close all,clear all,clc;

%% DEFINIÇÃO DAS AMOSTRAS DE TREINAMENTO

% Prototipo de entrada
S = 7; % Neuronios
Q = 21; % Sao 7 padroes de teste
% A entrada da rede é um vetor p_i de R linhas, onde i =1 ... Q, Q é o
% numero de amostras
% ------ Fonte 1 - Treinar  ----------
Trabalho2_fonte1;
p1 = p1'; p1 = p1(:); % Transformando mariz em vetor coluna ,linhas em sequencia
p2 = p2'; p2 = p2(:);
p3 = p3'; p3 = p3(:);
p4 = p4'; p4 = p4(:);
p5 = p5'; p5 = p5(:);
p6 = p6'; p6 = p6(:);
p7 = p7'; p7 = p7(:);
% P é uma matriz de  Q amostras de entradas
P1 = [p1 p2 p3 p4 p5 p6 p7];
% ------ Fonte 2 - Treinar  ----------
Trabalho2_fonte2;
p1 = p1'; p1 = p1(:); % Transformando mariz em vetor coluna ,linhas em sequencia
p2 = p2'; p2 = p2(:);
p3 = p3'; p3 = p3(:);
p4 = p4'; p4 = p4(:);
p5 = p5'; p5 = p5(:);
p6 = p6'; p6 = p6(:);
p7 = p7'; p7 = p7(:);
% P é uma matriz de  Q amostras de entradas
P2 = [p1 p2 p3 p4 p5 p6 p7];
% ------ Fonte 3 - Treinar  ----------
Trabalho2_fonte3;
p1 = p1'; p1 = p1(:); % Transformando mariz em vetor coluna ,linhas em sequencia
p2 = p2'; p2 = p2(:);
p3 = p3'; p3 = p3(:);
```

```
p4 = p4'; p4 = p4(:);
p5 = p5'; p5 = p5(:);
p6 = p6'; p6 = p6(:);
p7 = p7'; p7 = p7(:);
% P é uma matriz de  Q amostras de entradas
P3 = [p1 p2 p3 p4 p5 p6 p7];
%% TREINAMENTO
P = [P1 P2 P3]; % OS padrões de todas as fontes
% A saída desejada t_i é um vetor de S linhas , onde i= 1 ...Q, Q é o
% número de amostras. Cada t_i(S) é ativado uma unica vez
% Para esse CASO (S) T_j é exatamente a matriz identidade [SxQ]
T = [eye(7) eye(7) eye(7)]; % Para as três fontes

% Inicializando os valores de pesos e polarização
[W, b] = initNET(P,T); % W matriz SxR e B matriz SxQ,
%% Para tabelar os valores de saida inicial
A = zeros(size(T));
 for j = 1:Q
        p = P(:,j);
        t = T(:,j);
        [~, A(:,j)] = testNET(p,t,W,b);

 end

 save('treinamento.txt','A','-ascii'); % Para salvar os dados

%Aprendizagem
[Wn,bn,epoca] = learnNET(P,T,W,b);




%% GENERALIZAÇÃO
% ------ Fonte 1 - Teste ----------
Trabalho2_fonte1_teste;
p1 = p1'; p1 = p1(:); % Transformando mariz em vetor coluna ,linhas em sequencia
p2 = p2'; p2 = p2(:);
p3 = p3'; p3 = p3(:);
p4 = p4'; p4 = p4(:);
p5 = p5'; p5 = p5(:);
p6 = p6'; p6 = p6(:);
p7 = p7'; p7 = p7(:);
% P é uma matriz de  Q amostras de entradas
```

```
P1 = [p1 p2 p3 p4 p5 p6 p7];
% ------ Fonte 2 - Teste  ----------
Trabalho2_fonte2_teste;
p1 = p1'; p1 = p1(:); % Transformando mariz em vetor coluna ,linhas em sequencia
p2 = p2'; p2 = p2(:);
p3 = p3'; p3 = p3(:);
p4 = p4'; p4 = p4(:);
p5 = p5'; p5 = p5(:);
p6 = p6'; p6 = p6(:);
p7 = p7'; p7 = p7(:);
% P é uma matriz de  Q amostras de entradas
P2 = [p1 p2 p3 p4 p5 p6 p7];
% ------ Fonte 3 - Teste  ----------
Trabalho2_fonte3_teste;
p1 = p1'; p1 = p1(:); % Transformando mariz em vetor coluna ,linhas em sequencia
p2 = p2'; p2 = p2(:);
p3 = p3'; p3 = p3(:);
p4 = p4'; p4 = p4(:);
p5 = p5'; p5 = p5(:);
p6 = p6'; p6 = p6(:);
p7 = p7'; p7 = p7(:);
% P é uma matriz de  Q amostras de entradas
P3 = [p1 p2 p3 p4 p5 p6 p7];

%% Para tabelar os valores de saida com a rede treinada
B = zeros(size(T));
 for j = 1:Q
        p = P(:,j);
        t = T(:,j);
        [~,B(:,j)] = testNET(p,t,Wn1,bn1); % So é 0 ou 1
 end
 save('teste.txt','B','-ascii'); % Para salvar os dados
```

II. initNET

```
function [W,b] = initNET(P,T)
%INITNET Retorna a matriz de pesos e a polarização
%   Gera valores iniciais para os pesos e as polarização de um único
%   neurônio, sendo feito por um número aleatório simétrico próximo de
%   zero. Sendo estes números normalmente distribuídos.
%   P = RxQ matriz dos vetores de entrada (P1..PQ), pi = [p1..pR]. R é número de caracteristicas
%   T = SxQ vetor de saídas (t1..tQ), S = numero de neuronios e Q numero de amostras para uma cam

% R - entradas da rede neural
```

```
[R , Qp] = size(P);
[S , Qt] = size(T);
if(Qp == Qt)
    W = randn(S,R); % Matriz SxR
    b = randn(S,1); % Matriz Sx1,
else
    error('Deve haver correspondência de Q colunas de entradas e saídas(targets).');
end
```

III. testNET

```
function [e,a] = testNET(p,t,W,b)
%LEARNNET Regra unificada de aprendizagem do perceptron
%   Calcula as variações nos pesos e polarizações de um conjunto de
%   entradas P e o erro.
%    P[Rx1] vetor entrada de R caracteristcas #1
%    T[Sx1] vetor de saídas   #2
%    W[SxR]
%    A[Rx1] matriz dos vetores de saída com a aplicação da hardlim(W'A+b).
%    E [Sx1] matriz dos vetores de erro.
n = W*p + b; % N[Sx1] numero de saídas por amostras
a = hardlim(n); % aplicação da função de ativação para Q amostras
e = t - a; % e[Sx1], onde cada coluna e_i é o erro relacionado a uma amostra
end
```

IV. learnNET

```
function [Wn, bn,epoca] = learnNET(P,T,W_old,b_old)
%LEARNNET Regra unificada de aprendizagem do perceptron
%   Calcula as variações nos pesos e polarizações de um conjunto de
%   entradas P e o erro.
%    P[RxQ] matriz dos vetores de entrada (P1..PQ), Pi = [p1..pR].  #1
%    T[SxQ] matriz de saídas (t1..tQ), S = 1 para uma camada  #2
%    W[SxR] matriz de pesos
%    b[Sx1] vetor de polarizações associada aos neuronios por amostras
% RETORNA
%   epoca = numero de ajustes dos pesos sinápticos e limiares da rede
%   Wn matriz de pesos atualizada
%   bn vetor de polarização atualizada

epoca = 0;
Wn = W_old;
bn = b_old;
x = size(T);
```

```
18 a = zeros(x);
19 Q = x(2);
20 check = 0;
21
22 while check <= Q  % Se a verificação nao atingir a quantidade de amostras de sucesso
23     for j = 1:Q
24         p = P(:,j);
25         t = T(:,j);
26         [erro,a(:,j)] = testNET(p,t,Wn,bn); % erro da amostra p_j associado a t_j
27         if max(erro)~=0 || min(erro)~=0 % SE ha erros atualiza os pesos
28             Wn = Wn + erro*p';
29             bn = bn + erro;
30             epoca = epoca +1;
31             check = 0; % reinicia o ciclo de checagens
32         else
33             check = check +1; % se não avalia proxima amostra
34         end
35     end
36 end
37 end
```