# REDES NEURAIS ARTIFICIAIS

## Trabalho Prático 2

*Thayse Cristina Araújo Rodrigues mat: 09.2.8088*

### MOTIVAÇÃO DO TRABALHO

O professor Talles Medeiros, como trabalho prático 2 da disciplina de Redes Neurais Artificiais, nos pediu para treinar uma rede neural artificial usando a função Levenberg-Marquardt backpropagation (trainlm). Trainlm é uma função de treinamento da rede que atualiza os valores dos pesos de acordo com o a otimização Levenberg-Marquardt.

### EXPERIMENTOS

O treino ocorre de acordo com os parâmetros listados na Tabela 1.

| | | |
|---|---|---|
| **net.trainParam.epochs** | **1000** | **Maximum number of epochs to train** |
| **net.trainParam.goal** | 0 | Performance goal |
| **net.trainParam.max_fail** | 6 | Maximum validation failures |
| **net.trainParam.min_grad** | 1e-7 | Minimum performance gradient |
| **net.trainParam.mu** | 0.001 | Initial mu |
| **net.trainParam.mu_dec** | 0.1 | mu decrease factor |
| **net.trainParam.mu_inc** | 10 | mu increase fator |
| **net.trainParam.mu_max** | 1e10 | Maximum um |
| **net.trainParam.show** | 25 | Epochs between displays (NaN for no displays) |
| **net.trainParam.showCommandLine** | false | Generate command-line output |
| **net.trainParam.showWindow** | true | Show training GUI |
| **net.trainParam.time** | inf | Maximum time to train in seconds |

**Tabela 1 – Parâmetros de treinamento da rede neural.**

O código que usamos no matlab foi o seguinte:

```
load glassmat.txt

patterns = glassmat(:, 1:9);

desired = glassmat(:, 10:16);

patterns = [patterns ones(size(patterns,1),1) ]; %adiciono o bias

netlm = newff(patterns', desired', [5,7], {'tansig', 'purelin'}, 'trainlm');

netlm = train(netlm, patterns', desired');

tlm_obtida = sim(netlm, patterns');
```

O arquivo glassmat.txt: é o arquivo contendo a matriz de dados sendo esta uma base de dados contendo 9 atributos de composição de um vidro e 7 classes as quais o

vidro pode pertencer com esta composição. As classes originalmente são nomes mas nós os transformamos em saídas binárias sendo:

- '1,0,0,0,0,0,0' para a classe 'build wind float'
- '0,1,0,0,0,0,0' para a classe 'build wind non-float'
- '0,0,1,0,0,0,0' para a classe 'vehic wind float'
- '0,0,0,1,0,0,0' para a classe 'vehic wind non-float'
- '0,0,0,0,1,0,0' para a classe containers
- '0,0,0,0,0,1,0' para a classe tableware
- '0,0,0,0,0,0,1' para a classe headlamps

No comando *newff* 5 é a quantidade de neurônios da camada oculta e 7 é a quantidade de neurônios da saída do algoritmo. *Tansig* é a função de ativação sigmoide. Purelin indica que na camada de saída teremos 7 neurônios purelin. *Trainlm* é a função de treinamento conforme já descrito no início deste relatório.

Os resultados obtidos com este código estão nas figuras 1 e 2.

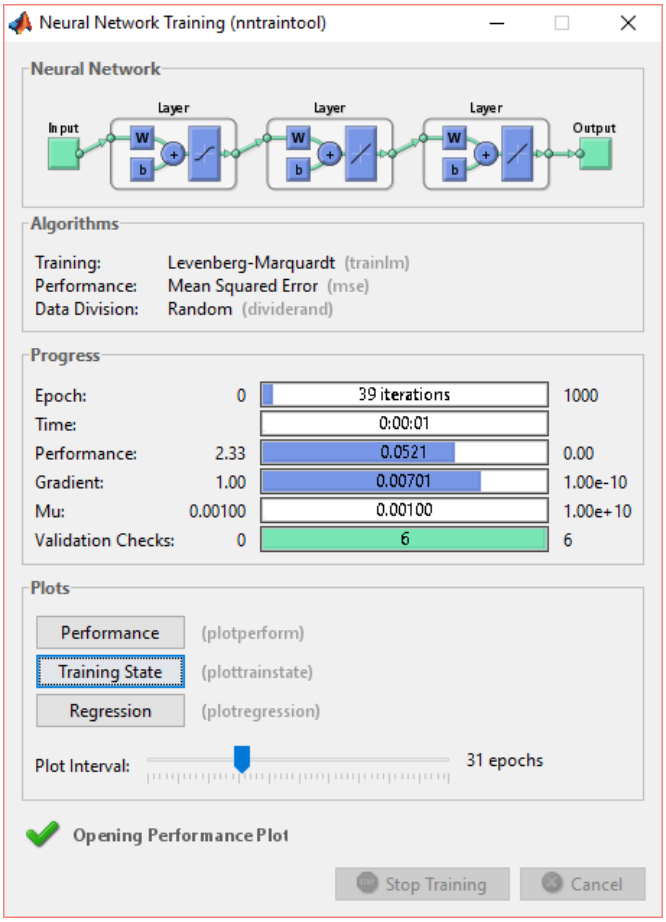


**Figura 1 – nnTrainTool do matlab mostrando informações sobre a execução do código.**

Pela figura 1 podemos ver que tivemos 39 interações o tempo gasto que foi de 1 segundo e informações de performance e gradiente.

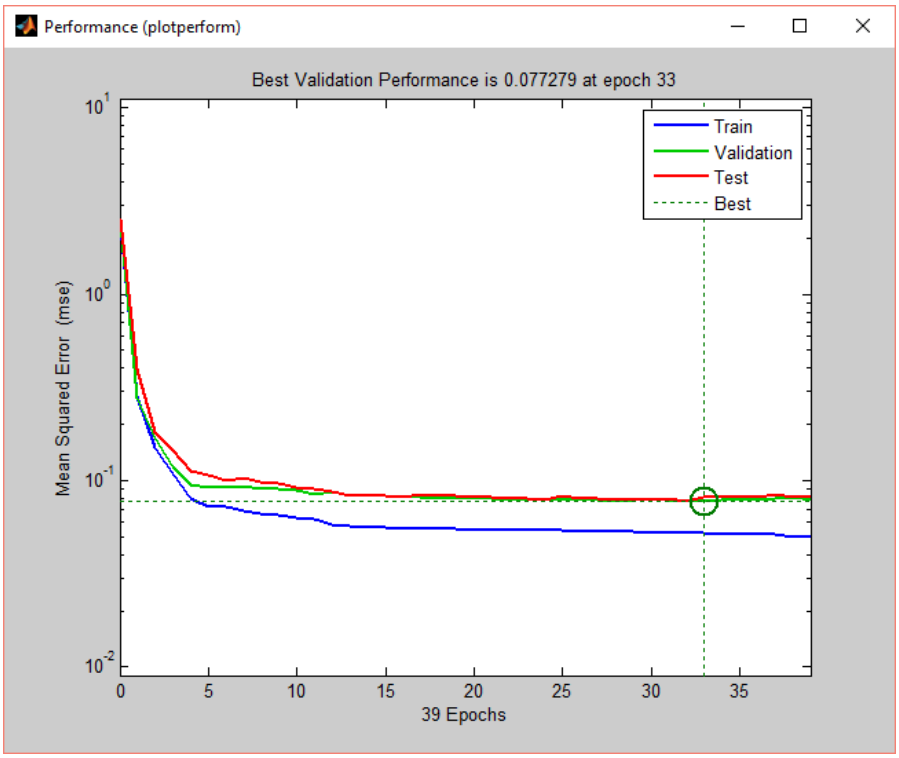


**Figura 2 – Erro médio quadrático obtido na execução da rede neural.**

É possível observar que tivemos um valor baixo de erro sendo o melhor obtido com 33 épocas sendo o valor do erro médio quadrático igual a 0,077.

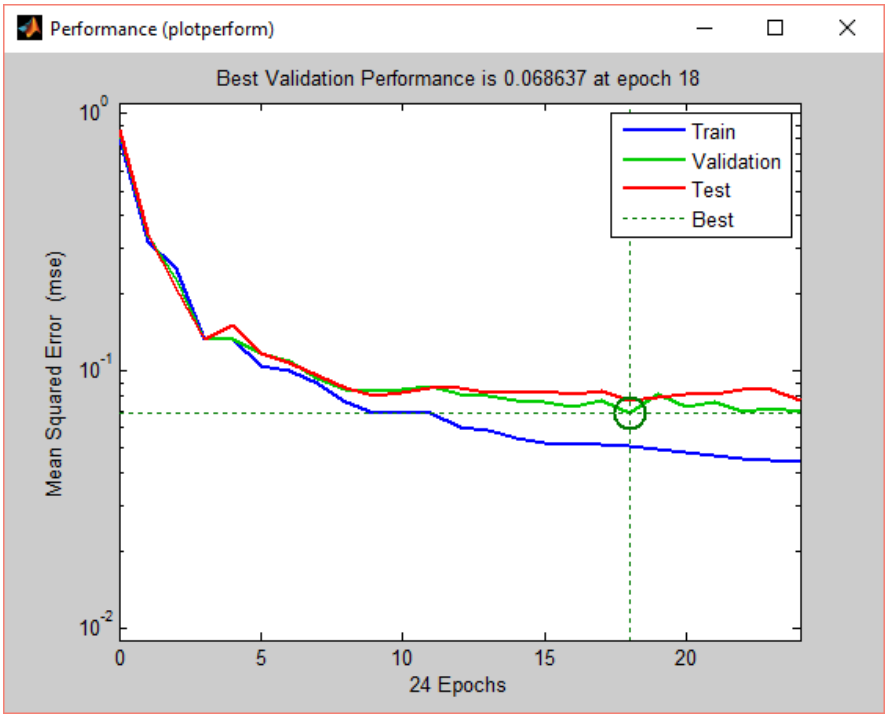


**Figura 3 – Melhor erro médio quadrático conseguido durante os testes.**

Alteramos a quantidade de neurônios na camada oculta para 6 obtivemos o melhor erro médio quadrático, sendo 0,0686 mostrado na Figura 3.

Alterando os valores de parâmetros citados na tabela 1 não conseguimos obter um resultado melhor do que o mostrado na figura 3. Então ficamos com os parâmetros default mostrados na Tabela 1 e com 6 neurônios na camada oculta para se chegar neste resultado.

## CONCLUSÃO

O Matlab auxilia muito o aprendizado e aplicação de algoritmos de Redes Neurais Artificiais visto durante a disciplina. Alterando os parâmetros do domando newff do Matlab o melhor resultado que conseguimos obter foi erro médio quadrático igual a 0,0686 mostrado na Figura 3. Este valor de erro mostra que nosso algoritmo conseguiu aprender o padrão da base de dados Glass e conseguiu prever, com um percentual baixo de erros, com os atributos dados a classe a qual o vidro pertencia.