CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS

Mestrado em Modelagem Matemática e Computacional

# Aula Prática V

# Aproximação de funções com uso de RNA

**Nome**: Samara Soares Leal

**Disciplina:** Inteligência Computacional - MMC

**Data:** 21 de Agosto de 2014.

**Exercício 01**

Duas entradas: 250 pontos no intervalo de [-10,10; -10 10]

Saída: Função do Sombreiro

Erro menor que 10^-4

Número de neurônios para treinamento: 26

a) Algoritmo BP Clássico: 'traingd()'
Norma do Erro: 32.0526

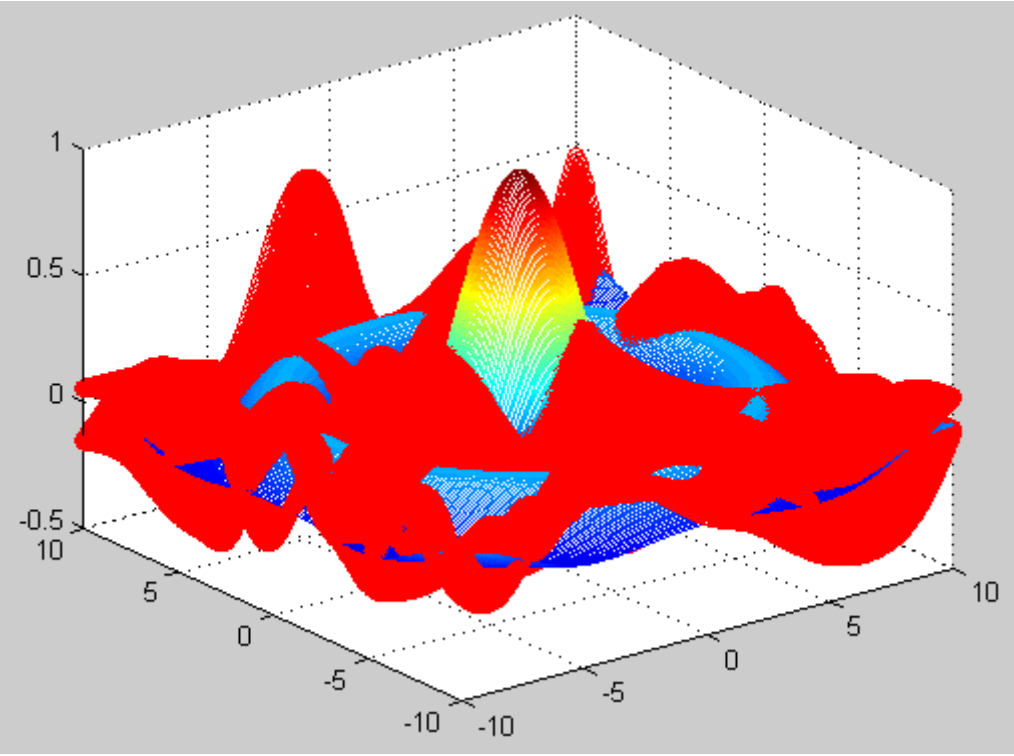


**Figura 01** – Gráfico vermelho: Saída gerada pela toolbox.

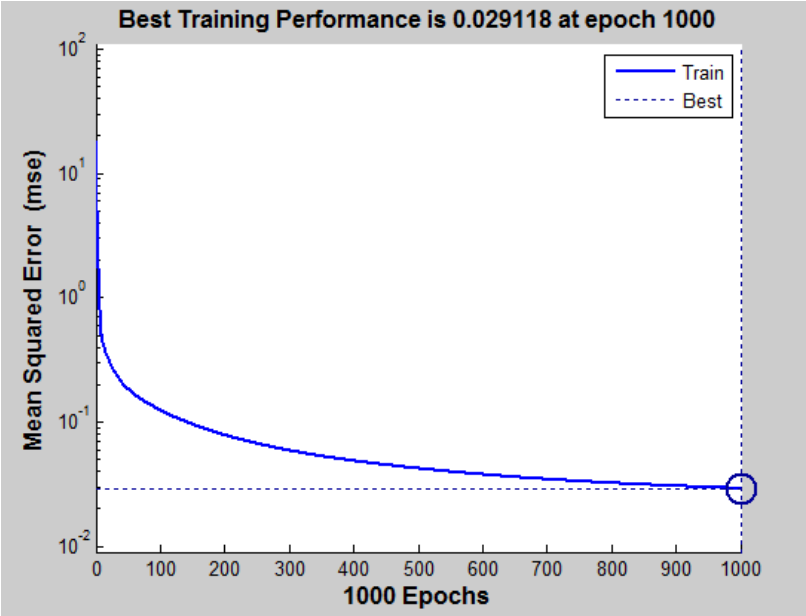


**Figura 02** – Valor do Erro Quadrático (mse).

b) Algoritmo BP com momento: 'traingdm()'
Norma do Erro: 28.8468

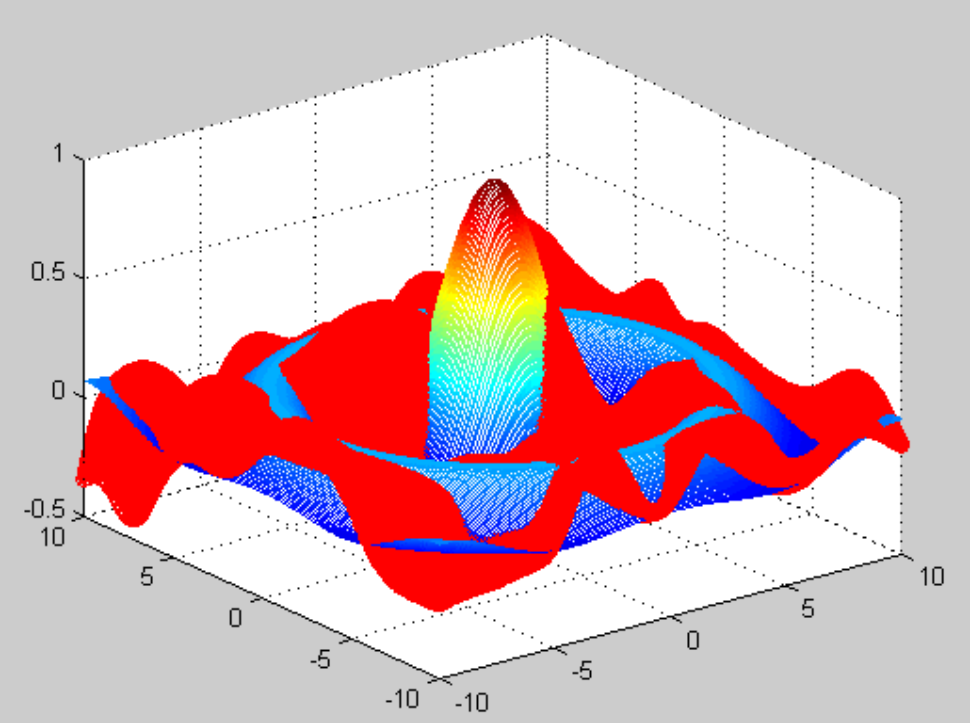

**Figura 03** – Gráfico vermelho: Saída gerada pela toolbox.

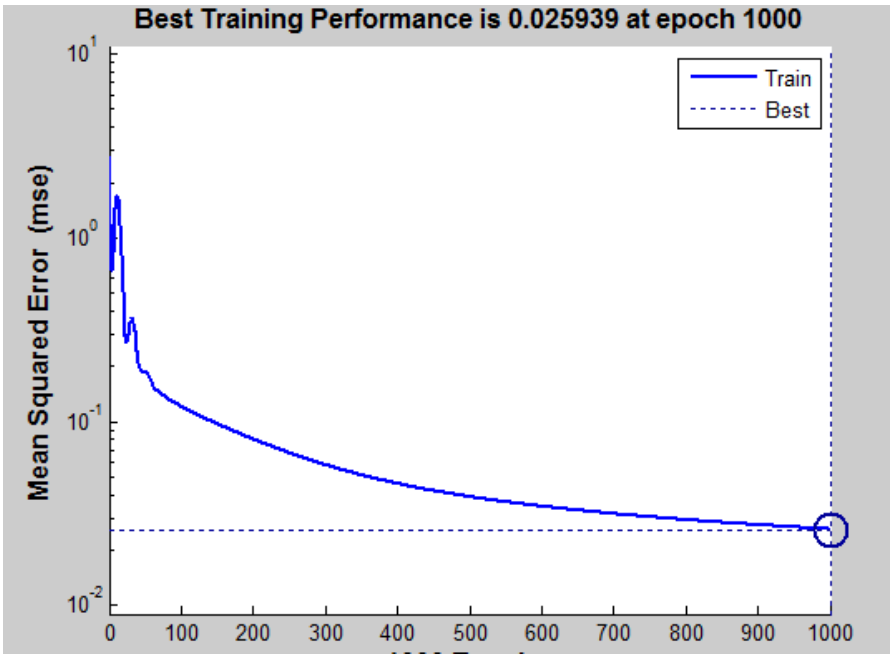


**Figura 04** – Valor do Erro Quadrático (mse).

c) Usando o algoritmo LM: 'trainlm()'
Norma do erro: 0.9988

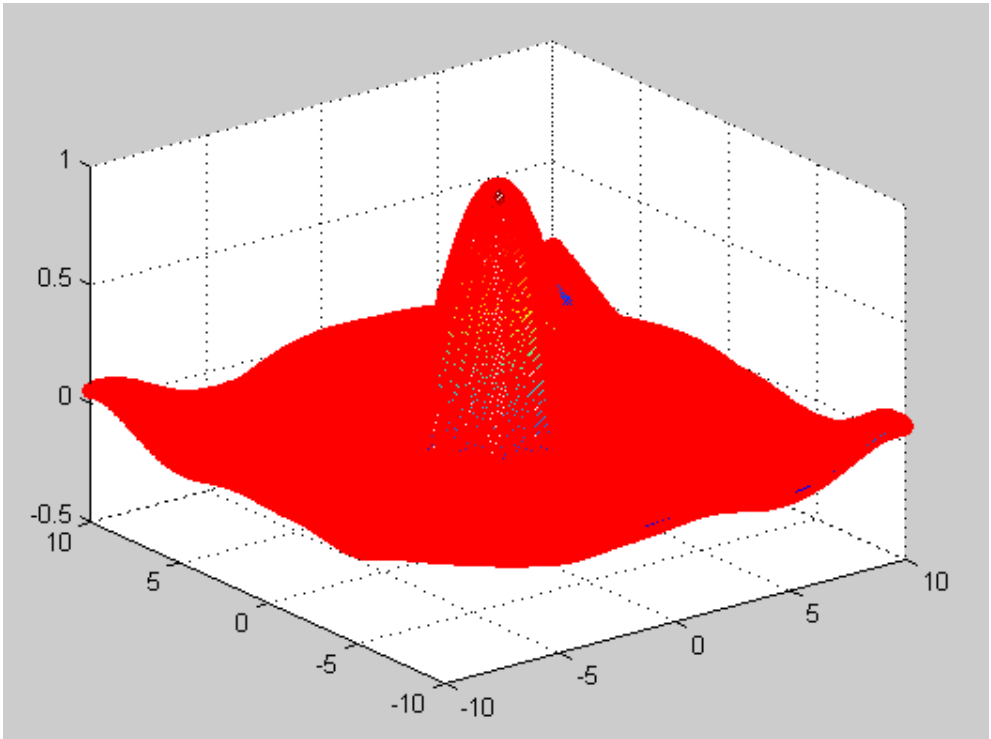


**Figura 05** – Gráfico vermelho: Saída gerada pela toolbox.

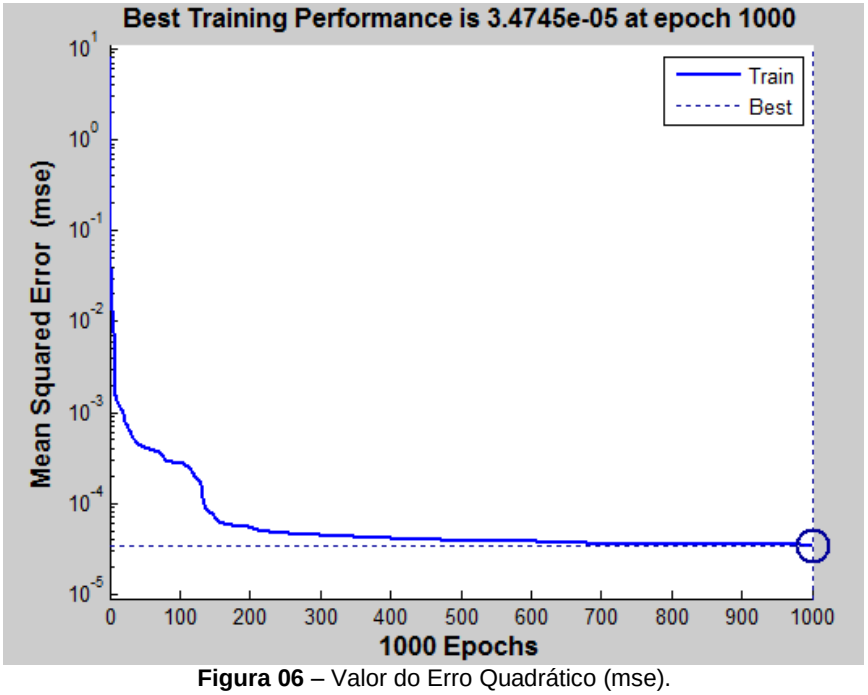


**Figura 06** – Valor do Erro Quadrático (mse).

Pode-se observar que com o aumento do número de neurônios a rede converge melhor, porém o tempo de execução aumenta muito. Assim foi escolhido para treinamento 26 neurônios e com este valor a rede que mostrou uma melhor convergência foi a rede treinada utilizando-se o algoritmo LM 'trainlm()', conforme já era esperado. Porém o tempo de execução do 'trainlm()' é maior do que os outros algoritmos.

**Exercício 02**

Testes de generalização: Simular a rede com uma quantidade menor de pontos (41 Pontos) dentro do intervalo. Estes pontos são diferentes dos pontos em que a rede foi treinada.

a) Algoritmo BP Clássico: 'traingd()'
Norma do Erro: 4.9367

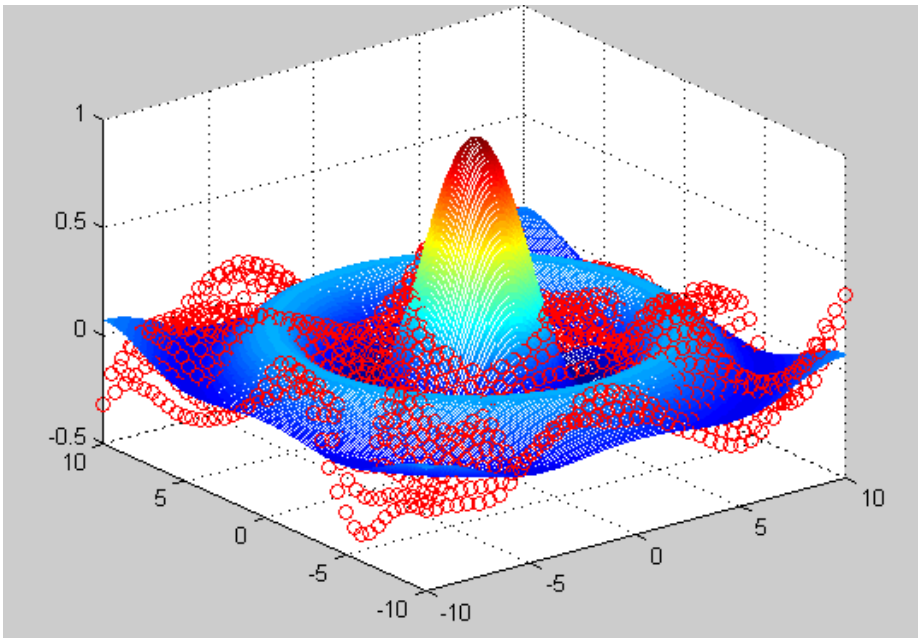

**Figura 07** – Gráfico vermelho: Saída gerada pela toolbox.

b) Algoritmo BP com momento: ‘traingdm()’
Norma do Erro: 5.6517

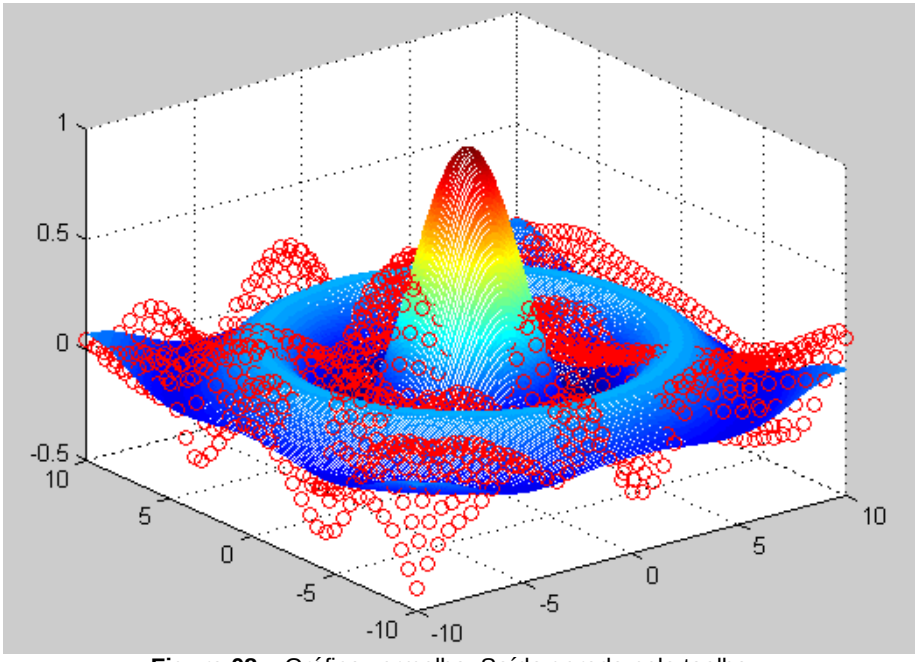


**Figura 08** – Gráfico vermelho: Saída gerada pela toolbox.

c) Usando o algoritmo LM: ‘trainlm()’
Norma do erro: 0.9994

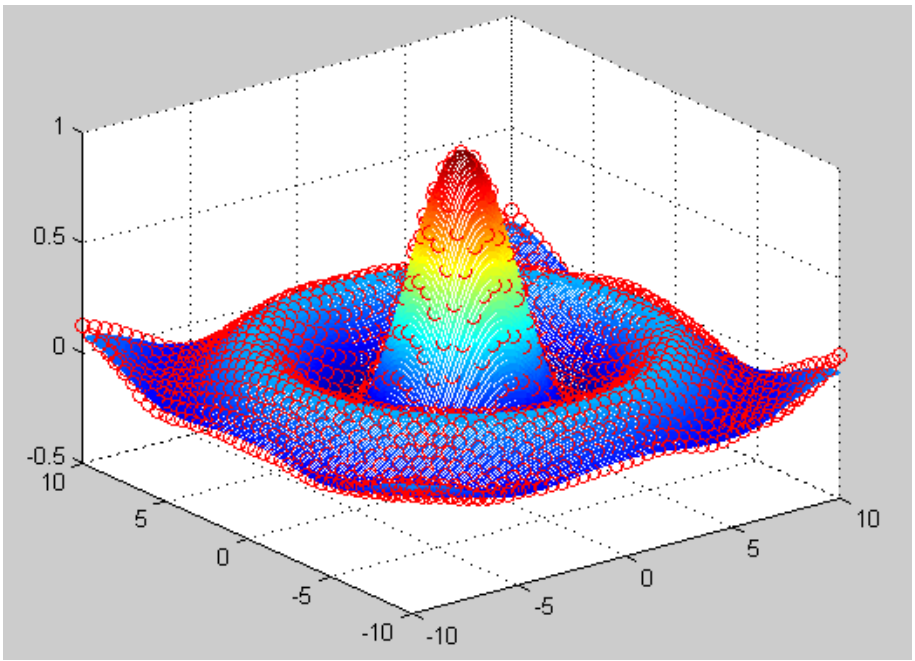


**Figura 09** – Gráfico vermelho: Saída gerada pela toolbox.

Pode-se observar que a rede aprendeu com treinamento e mostrou um bom desempenho com menos pontos. A norma do erro diminuiu para todos os casos e com o algoritmo LM teve o menor valor.

Algortimo:

```
function [resp] = redeNeuralBP( )
clear
clc
% Entrada
x = linspace(-10,10,250);
y = linspace(-10,10,250);

% Calcular função
[X,Y]=meshgrid(x,y);
r= sqrt(X.^2+Y.^2);
z=sin(r)./r;
z(round(length(z)/2),round(length(z)/2))=0;

% Gerar o gráfico
mesh(X,Y,z)
entrada=[x y];
Z=z;

% Mecanismo para fazer o reshape das matrizes
% Guarda linha e coluna de x
[lX,cX]=size(X);
nX=reshape(X,1,lX*cX);
% Guarda linha e coluna de y
[lY,cY]=size(Y);
nY=reshape(Y,1,lY*cY);
% Guarda linha e coluna de z
[lZ,cZ]=size(z);
nZ=reshape(z,1,lZ*cZ);
entrada=[nX;nY];
Z=nZ;

% Criar a rede
%net = newff ( minmax(entrada),[26 1],{'tansig', 'purelin'},'traingd');
net = newff ( minmax(entrada),[26 1],{'tansig', 'purelin'},'traingdm');
%net = newff ( minmax(entrada),[26 1],{'tansig', 'purelin'},'trainlm');

% Definir erro desejado
net.trainParam.goal<(10^-4);

% Treinar a rede
net = train(net,entrada,Z);

% Simular a rede
S=sim(net,entrada);

% Gerar o gráfico com a saída da rede neural
hold on
nS=reshape(S,lZ,cZ);
plot3(X,Y,nS,'ro')
norm(z-nS)
```

Para treinar com menos pontos, adicionar este trecho:

```
% Simular a rede com menos entradas (41 Pontos)
xx=-10:.5:10;
yy=-10:.5:10;

% Calcular função
[X,Y]=meshgrid(xx,yy);
r= sqrt(X.^2+Y.^2);
z=sin(r)./r;
z(round(length(z)/2),round(length(z)/2))=0;

% Mecanismo para fazer o reshape das matrizes
% Guarda linha e coluna de x
[lX,cX]=size(X);
```

```
nX=reshape(X,1,lX*cX);
% Guarda linha e coluna de y
[lY,cY]=size(Y);
nY=reshape(Y,1,lY*cY);
% Guarda linha e coluna de z
[lZ,cZ]=size(z);
nZ=reshape(z,1,lZ*cZ);
entrada2=[nX;nY];
Z=nZ;

% simular a rede com menos entrada
S= sim(net,entrada2);

% Gerar o gráfico com a saída da rede neural
hold on
nS=reshape(S,lZ,cZ);
plot3(X,Y,nS,'ro')
norm(z-nS)
```