UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA - CIN
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
REDES NEURAIS - IN0997

CICERO SAMUEL RODRIGUES MENDES

RELATÓRIO DE EXERCÍCIO COMPUTACIONAL

RECIFE
2021

- As implementações para a solução deste exercício computacional podem ser acessadas através deste link.

5.5 Approximate the following functions by using MLP:

(a) $f(x_1, x_2) = \max\{e^{-x_1^2}, e^{-2x_2^2}, 2e^{-0.5(x_1^2+x_2^2)}\}$.

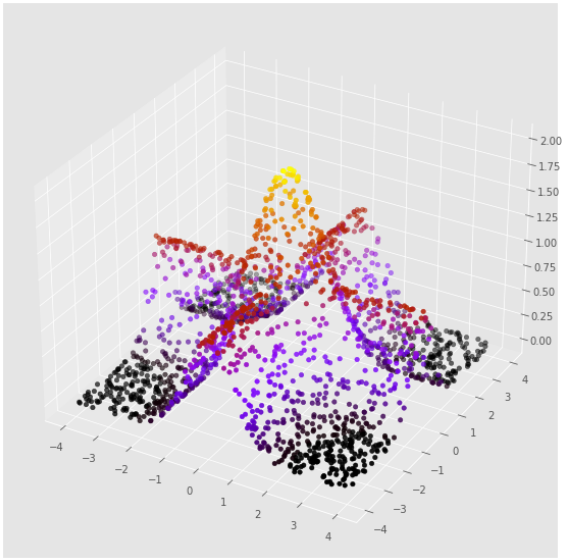
(b) $f(x, y) = 0.5 + 0.1x^2 \cos(y + 3) + 0.4xye^{1-y^2}$.

(c) $f(\boldsymbol{x}) = \sin\left(\sqrt{x_1^2 + x_2^2}\right) / \sqrt{x_1^2 + x_2^2}, \quad \boldsymbol{x} \in [-5, 5]^2$.

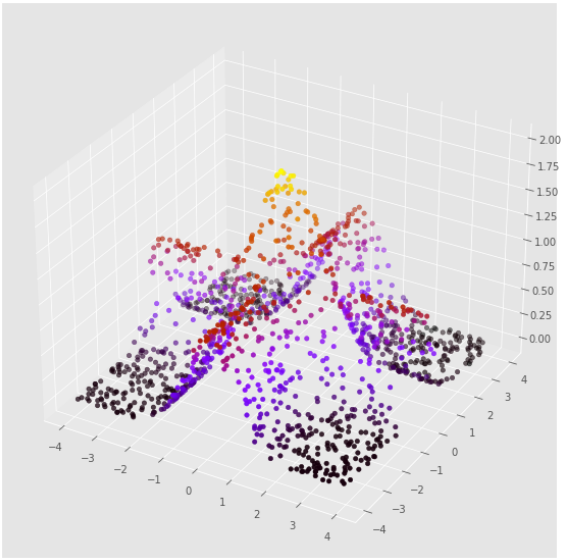
(d) $f(x) = \sqrt{2}\sin x + \sqrt{2}\cos x - \sqrt{2}\sin 3x + \sqrt{2}\cos 3x$.

(a) Para a aproximação da função do **Item 5.5 (a)** foram gerados aleatoriamente e seguindo uma distribuição uniforme, 2000 pontos para os parâmetros x1 e x2 no intervalo de [-4, 4]. Os valores de x1 e x2 foram aplicados à função e os dados obtidos foram usados como exemplos de treinamento (*target*) para a rede MLP. Os dados foram divididos em dois conjuntos, sendo 70% para treinamento e 30% para teste. A rede MLP foi construída com 5 camadas, sendo: 1 camada de entrada com dois neurônios e função de ativação ReLU, 3 camadas escondidas com 50 neurônios em cada uma e função de ativação ReLU, 1 camada de saída com 1 neurônio e função de ativação linear. A taxa de aprendizado utilizada foi de 0.05 e 800 épocas. Não foi adotado nenhum procedimento de pré-processamento dos valores apresentados à rede. A **Figura 1** apresenta os gráficos da função do **Item 5.5 (a)** versus o gráfico da função aproximada pela rede MLP de acordo com os parâmetros anteriormente descritos.

Figura 1 - Gráfico da Função 5.5(a) vs. Gráfico da Função Aproximada pela Rede MLP.



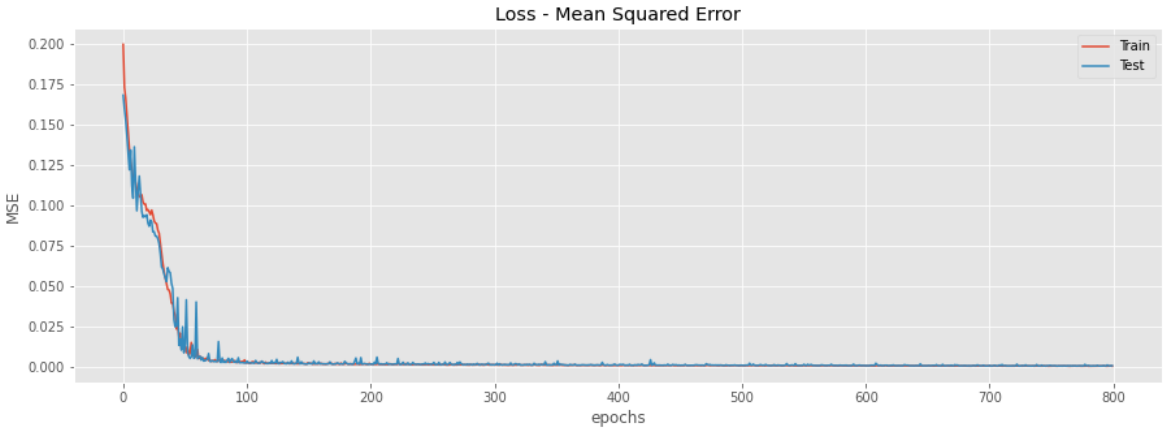
(a) Gráfico da Função 5.5(a)



(b) Gráfico da Função Aproximada pela Rede MLP para a Função 5.5(a)

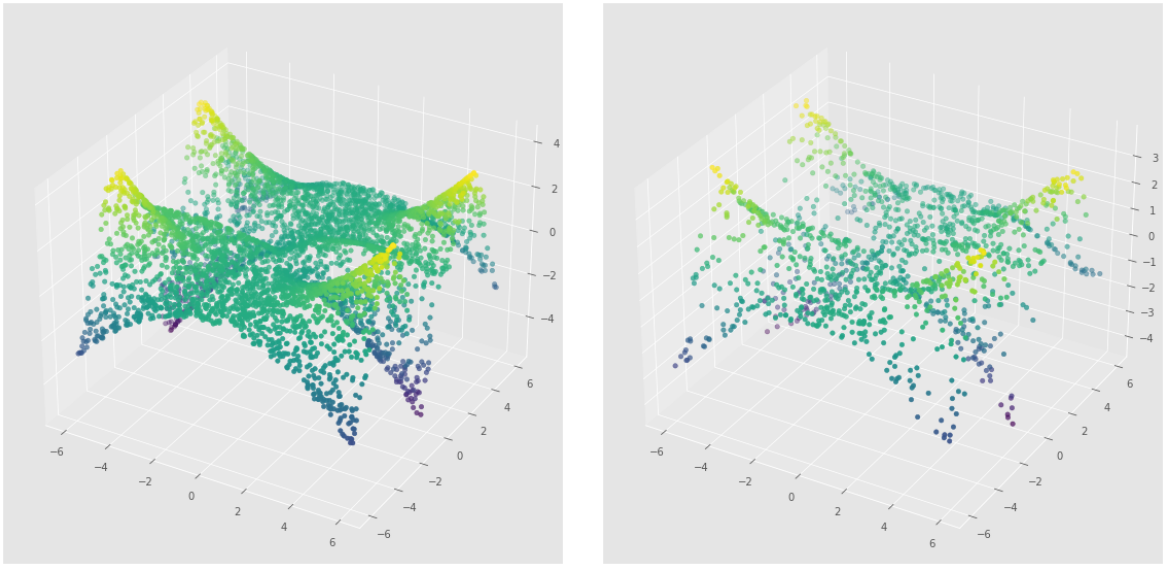
A **Figura 2** apresenta o gráfico do Mean Squared Error (MSE) para os conjuntos de treinamento e teste ao longo das épocas da rede MLP de acordo com os parâmetros anteriormente descritos.

Figura 2 - MSE de Treinamento e Teste da Rede MLP ao Longo das Épocas



(b) Para a aproximação da função do **Item 5.5 (b)** foram gerados aleatoriamente e seguindo uma distribuição uniforme, 2000 pontos para os parâmetros x e y no intervalo de [-6, 6]. Os valores de x e y foram aplicados à função e os dados obtidos foram usados como exemplos de treinamento (*target*) para a rede MLP. Os dados foram divididos em dois conjuntos, sendo 70% para treinamento e 30% para teste. A rede MLP foi construída com 5 camadas, sendo: 1 camada de entrada com dois neurônios e função de ativação ReLU, 3 camadas escondidas com 30 neurônios em cada uma e função de ativação ReLU, 1 camada de saída com 1 neurônio e função de ativação linear. A taxa de aprendizado utilizada foi de 0.01 e 1200 épocas. Não foi adotado nenhum procedimento de pré-processamento dos valores apresentados à rede. A **Figura 3** apresenta os gráficos da função do **Item 5.5 (b)** versus o gráfico da função aproximada pela rede MLP de acordo com os parâmetros anteriormente descritos.

Figura 3 - Gráfico da Função 5.5(b) vs. Gráfico da Função Aproximada pela Rede MLP.

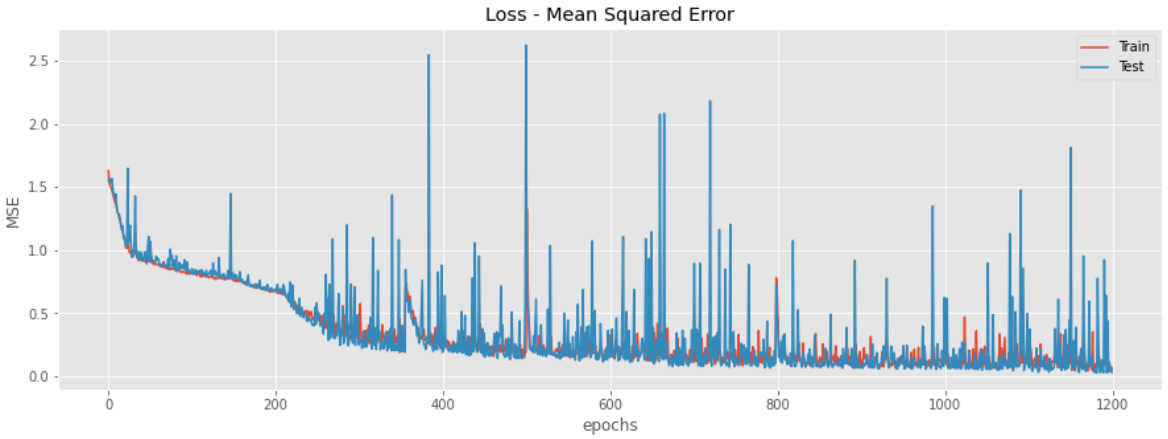


(a) Gráfico da Função 5.5(b)

(b) Gráfico da Função Aproximada pela Rede MLP para a Função 5.5(b)

A **Figura 4** apresenta o gráfico do Mean Squared Error (MSE) para os conjuntos de treinamento e teste ao longo das épocas da rede MLP de acordo com os parâmetros anteriormente descritos.

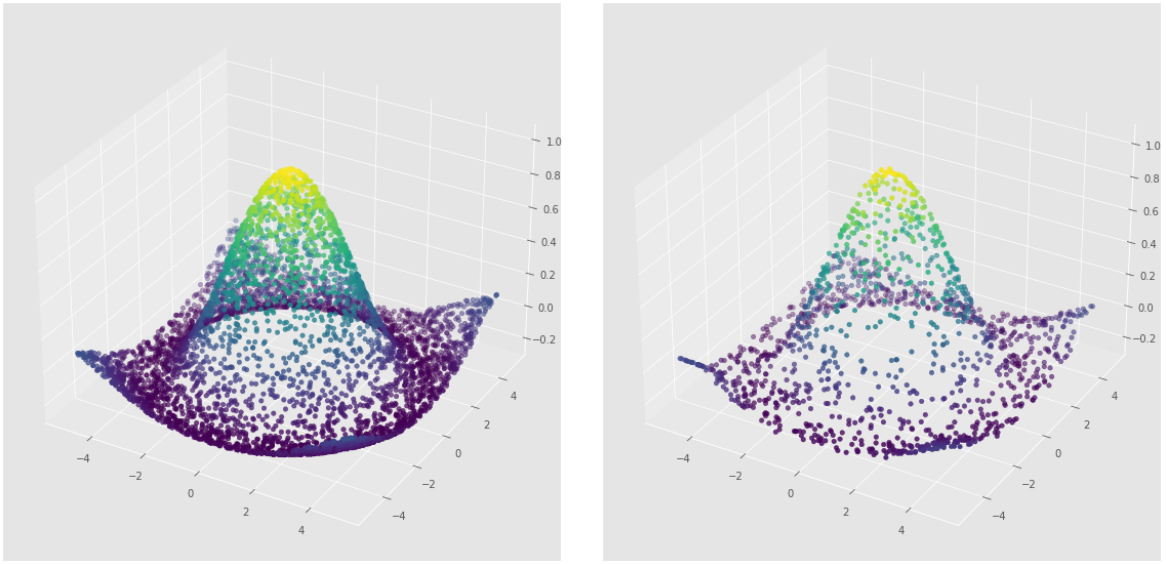
Figura 4 - MSE de Treinamento e Teste da Rede MLP ao Longo das Épocas.



(c) Para a aproximação da função do **Item 5.5 (c)** foram gerados aleatoriamente e seguindo uma distribuição uniforme, 2000 pontos para os parâmetros x1 e x2 no intervalo de [-5, 5]. Os valores de x1 e x2 foram aplicados à função e os dados obtidos foram usados como exemplos de treinamento (*target*) para a rede MLP. Os dados foram divididos em dois conjuntos, sendo 70% para treinamento e 30% para teste. A rede MLP foi construída com 5 camadas, sendo: 1 camada de entrada com dois neurônios e função de ativação ReLU, 3 camadas escondidas com 50 neurônios em cada uma e função de ativação ReLU, 1 camada de saída com 1 neurônio e função de

ativação linear. A taxa de aprendizado utilizada foi de 0.03 e 600 épocas. Não foi adotado nenhum procedimento de pré-processamento dos valores apresentados à rede. A **Figura 5** apresenta os gráficos da função do **Item 5.5 (c)** versus o gráfico da função aproximada pela rede MLP de acordo com os parâmetros anteriormente descritos.

Figura 5 - Gráfico da Função 5.5(c) vs. Gráfico da Função Aproximada pela Rede MLP.

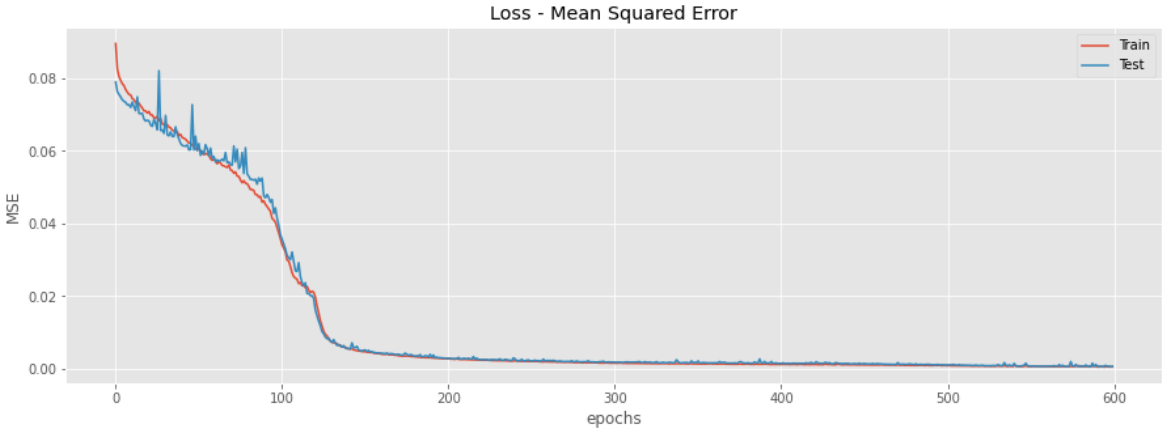


(a) Gráfico da Função 5.5(c)

(b) Gráfico da Função Aproximada pela Rede MLP para a Função 5.5(c)

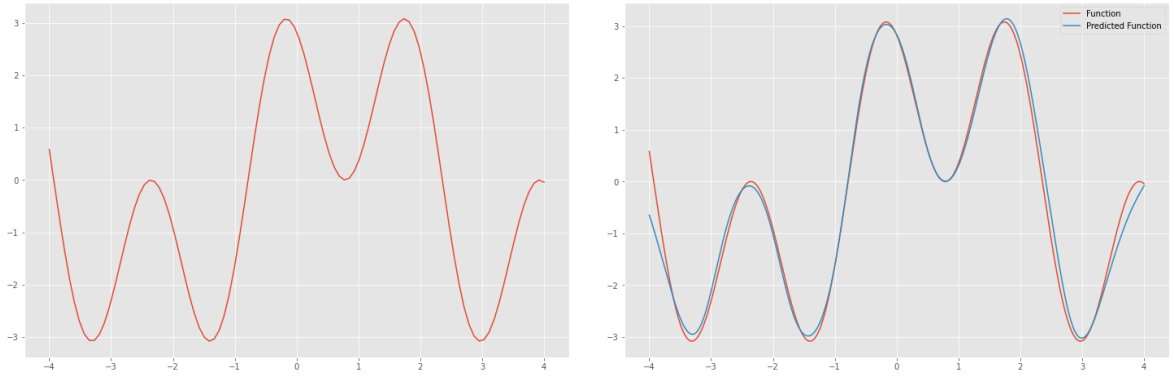
A **Figura 6** apresenta o gráfico do Mean Squared Error (MSE) para os conjuntos de treinamento e teste ao longo das épocas da rede MLP de acordo com os parâmetros anteriormente descritos.

Figura 6 - MSE de Treinamento e Teste da Rede MLP ao Longo das Épocas.



(d) Para a aproximação da função do **Item 5.5 (d)** foram gerados aleatoriamente e seguindo uma distribuição uniforme, 2000 pontos para o parâmetro x no intervalo de [-4, 4]. Os valores de x foram aplicados à função e os dados obtidos foram usados como exemplos de treinamento (*target*) para a rede MLP. Os dados foram divididos em dois conjuntos, sendo 70% para treinamento e 30% para teste. A rede MLP foi construída com 10 camadas, sendo: 1 camada de entrada com dois neurônios e função de ativação Tangente, 8 camadas escondidas com 25 neurônios em cada uma e função de ativação Tangente, 1 camada de saída com 1 neurônio e função de ativação linear. A taxa de aprendizado utilizada foi de 0.001 e 1000 épocas. Não foi adotado nenhum procedimento de pré-processamento dos valores apresentados à rede. A **Figura 7** apresenta os gráficos da função do **Item 5.5 (d)** versus o gráfico da função aproximada pela rede MLP de acordo com os parâmetros anteriormente descritos.

Figura 7 - Gráfico da Função 5.5(d) vs. Gráfico da Função Aproximada pela Rede MLP.

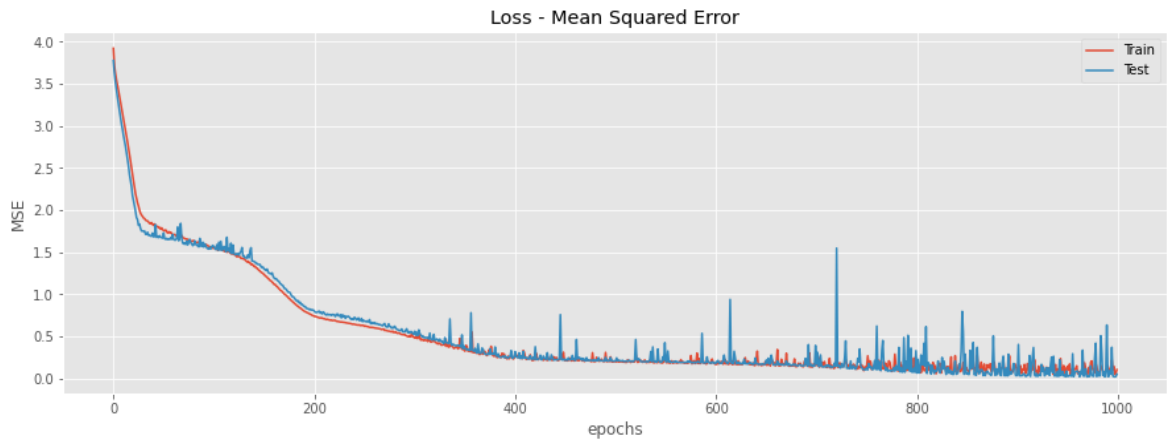


(a) Gráfico da Função 5.5(d)

(b) Gráfico da Função Aproximada pela Rede MLP para a Função 5.5(d)

A **Figura 8** apresenta o gráfico do Mean Squared Error (MSE) para os conjuntos de treinamento e teste ao longo das épocas da rede MLP de acordo com os parâmetros anteriormente descritos.

Figura 8 - MSE de Treinamento e Teste da Rede MLP ao Longo das Épocas.



5.6 Why is the MSE not a good measure of performance for classification problems?

De acordo com Du e Swamy (2019), "a função MSE pode ser obtida pelo princípio de ML, assumindo a independência e a gaussianidade dos dados alvo. No entanto, a suposição de Gaussianidade dos dados alvo na classificação não é válida, devido à sua natureza discreta dos rótulos de classe. Assim, a função MSE não é a mais adequada para problemas de classificação de dados". Além disso, os autores ainda afirmam que "o uso de funções de ativação não lineares causa mínimos locais nas funções objetivo com base no critério MSE".

5.9 Derive the forward and backward propagation equations for each of the loss functions:

(a) Kullback–Leibler divergence criterion (2.15), (2.16), (2.17).
(b) The cross-entropy cost function (2.18).
(c) The Minkowski-r metric (2.19).

(a) Solução para Kullback–Leibler divergence criterion (2.15), (2.16), (2.17).

a) $E_p = \frac{-1}{2}\sum_{i=1}^{J_M}[y_{p,i} ln \hat{y}_{p,i} - (1 - y_{p,i}) ln(1 - \hat{y}_{p,i})]$

Temos então a atualização de pesos:

$$\Delta_p W(2) = \frac{-1}{2}\sum_{i=1}^{J_M}[y_{p,i}\frac{\phi'_{(2)}(W^{(2)} net_p^{(2)}) net_p^{(2)}}{\hat{y}_{p,i}} + \frac{(1 - y_{p,i})\phi'_{(2)}(W^{(2)} net_p^{(2)}) net_p^{(2)}}{1 - \hat{y}_{p,i}}]$$

E para o conjunto $W_{(1)}$:

$$\Delta_p W(1) = \frac{-1}{2}\sum_{i=1}^{J_M}[y_{p,i}\frac{\phi'_{(2)}(W^{(2)} net_p^{(2)}) W^{(2)} (net_p^{(2)})'}{\hat{y}_{p,i}} + \frac{(1 - y_{p,i})\phi'_{(2)}(W^{(2)} net_p^{(2)}) W^{(2)} (net_p^{(2)})'}{1 - \hat{y}_{p,i}}]$$

Que implica:

$$\Delta_p W(1) = \frac{-1}{2}\sum_{i=1}^{J_M}[y_{p,i}\frac{\phi'_{(2)}(W^{(2)} net_p^{(2)}) W^{(2)} \phi'_{(1)}(net_p^{(1)})}{\hat{y}_{p,i}} + \frac{(1 - y_{p,i})\phi'_{(2)}(W^{(2)} net_p^{(2)}) W^{(2)} \phi'_{(1)}(net_p^{(1)})}{1 - \hat{y}_{p,i}}]$$

E finalmente:

$$\Delta_p W(1) = \frac{-1}{2}\sum_{i=1}^{J_M}[y_{p,i}\frac{\phi'_{(2)}(W^{(2)} net_p^{(2)}) W^{(2)} \phi'_{(1)}(net_p^{(1)}) x_p}{\hat{y}_{p,i}} + \frac{(1 - y_{p,i})\phi'_{(2)}(W^{(2)} net_p^{(2)}) W^{(2)} \phi'_{(1)}(net_p^{(1)}) x_p}{1 - \hat{y}_{p,i}}]$$

(b) Solução para cross-entropy cost function (2.18).

b) Sendo a medida de erro $E_i = -\sum_p \sum_i \hat{y}_{p,i} ln y_{p,i}$

Derivando em relação a cada conjunto de pesos $W^{(1)}$ e $W^{(2)}$ para obter cada fator de atualização dos pesos, temos:

$$\Delta_p W^{(2)} = -\sum_p \sum_i \frac{\partial}{\partial W^{(2)}} \hat{y}_{p,i} ln(y_{p,i})$$

$$\Delta_p W^{(2)} = -\sum_p \sum_i \phi'_{(2)}(net_p^{(2)}) net_p^{(2)} ln(y_{p,i})$$

E para o conjunto $W_{(1)}$:

$$\Delta_p W^{(1)} = -\sum_p \sum_i \phi'_{(2)}(net_p^{(2)}) W^{(2)} \phi'_1(net_p^{(1)}) ln(y_{p,i})$$

Que finalmente:

$$\Delta_p W^{(1)} = -\sum_p \sum_i \phi'_{(2)}(net_p^{(2)}) W^{(2)} \phi'_1(net_p^{(1)} x_p) ln(y_{p,i})$$

(c) Solução para Minkowski-r metric (2.19).

c) Seja $E_p = \frac{1}{r}\sum_{i=1}^{J_M} \mid \hat{y}_{p,i} - y_{p,i} \mid^r$ a métrica de Minkowski.
Para r par, temos a garantia da suavidade da função e podemos derivá-la em todo seu domínio. Assim:
Seja $\mathbf{W}^{(1)}$ o conjunto de pesos da camada de entra para *layer* oculto e $\mathbf{W}^{(2)}$ os pesos do *layer* oculto para a camada de saída, através do gradiente descendente, atualizamos cada peso como:

$\Delta_p \mathbf{W}^{(1)} - -\eta \frac{\partial E_p}{\partial \mathbf{W}^{(1)}}$

$\Delta_p \mathbf{W}^{(2)} = -\eta \frac{\partial E_p}{\partial \mathbf{W}^{(2)}}$

Derivando então a função de erro pela métrica de Minkowski:

$$\frac{\partial E_p}{\partial \mathbf{W}^{(2)}} = \frac{\partial \frac{1}{r}\sum_{i=1}^{J_M}(\hat{y}_{p,i} - y_{p,i})^r}{\partial \mathbf{W}^{(2)}} = r\frac{1}{r}\sum_{i=1}^{J_M}(\hat{y}_{p,i} - y_{p,i})^{r-1} * \phi'_{(2)}(W^{(2)} net_p^{(2)})$$

Logo:

$$\frac{\partial E_p}{\partial \mathbf{W}^{(2)}} = \frac{\partial \frac{1}{r}\sum_{i=1}^{J_M}(\hat{y}_{p,i} - y_{p,i})^r}{\partial \mathbf{W}^{(2)}} = \sum_{i=1}^{J_M}(\hat{y}_{p,i} - y_{p,i})^{r-1} * \phi'_{(2)}(W^{(2)} net_p^{(2)})$$

$$\frac{\partial E_p}{\partial \mathbf{W}^{(2)}} = \frac{\partial \frac{1}{r}\sum_{i=1}^{J_M}(\hat{y}_{p,i} - y_{p,i})^r}{\partial \mathbf{W}^{(2)}} = \sum_{i=1}^{J_M}(\hat{y}_{p,i} - y_{p,i})^{r-1} * \phi'_{(2)}(W^{(2)} net_p) * net_p^{(2)}$$

E para o conjunto de pesos $\mathbf{W}^{(1)}$:

$$\frac{\partial E_p}{\partial \mathbf{W}^{(1)}} - \frac{\partial \frac{1}{r}\sum_{i=1}^{J_M}(\hat{y}_{p,i} - y_{p,i})^r}{\partial \mathbf{W}^{(1)}} - r\frac{1}{r}\sum_{i-1}^{J_M}(\hat{y}_{p,i} - y_{p,i})^{r-1} * \phi'_{(2)}(W^{(2)} net_p)$$

$$\frac{\partial E_p}{\partial \mathbf{W}^{(1)}} = \frac{\partial \frac{1}{r}\sum_{i=1}^{J_M}(\hat{y}_{p,i} - y_{p,i})^r}{\partial \mathbf{W}^{(1)}} = \sum_{i=1}^{J_M}(\hat{y}_{p,i} - y_{p,i})^{r-1} * \phi'_{(2)}(W^{(2)} net_p^{(2)})$$

$$\frac{\partial E_p}{\partial \mathbf{W}^{(1)}} = \frac{\partial \frac{1}{r}\sum_{i=1}^{J_M}(\hat{y}_{p,i} - y_{p,i})^r}{\partial \mathbf{W}^{(1)}} = \sum_{i=1}^{J_M}(\hat{y}_{p,i} - y_{p,i})^{r-1} * \phi'_{(2)}(W^{(2)} net_p) * W^{(2)} * (net_p^{(2)})'$$

$$\frac{\partial E_p}{\partial \mathbf{W}^{(1)}} = \frac{\partial \frac{1}{r}\sum_{i=1}^{J_M}(\hat{y}_{p,i} - y_{p,i})^r}{\partial \mathbf{W}^{(1)}} = \sum_{i=1}^{J_M}(\hat{y}_{p,i} - y_{p,i})^{r-1} * \phi'_{(2)}(W^{(2)} net_p) * W^{(2)} * \phi'_{(1)}(net_p^{(1)})$$

E finalmente:

$$\frac{\partial E_p}{\partial \mathbf{W}^{(1)}} = \frac{\partial \frac{1}{r}\sum_{i=1}^{J_M}(\hat{y}_{p,i} - y_{p,i})^r}{\partial \mathbf{W}^{(1)}} = \sum_{i=1}^{J_M}(\hat{y}_{p,i} - y_{p,i})^{r-1} * \phi'_{(2)}(W^{(2)} net_p) * W^{(2)} * \phi'_{(1)}(net_p^{(1)}) x_p$$

5.11 Write a program implementing the three-layer MLP with the BP rule, incorporating the weight-decaying function.

(a) Generate 200 points from $y = \phi(x_1 + 2x_2) + 0.5(x_1 - x_2)^2 + 0.5N$, where $\phi(\cdot)$ is the logistic sigmoidal function, and N is a number drawn from the standard normal distribution. Apply the program on the samples.

(b) Test the program using 1000 randomly generated samples.

(c) Plot the training and testing errors versus the number of training epochs for differing weight-decay parameters.

(d) Describe the overfitting phenomenon observed.

Para a resolução do exercício 5.11 foram implementadas duas redes MLP. A primeira proposta de rede utiliza tensorflow[1] e, com vista a ter um entendimento mais completo sobre os conceitos e a implementação de redes neurais com arquiteturas MLP, implementou-se outra rede menos robusta.

Rede 1

O procedimento adotado consistiu na construção de uma rede MLP com 3 camadas, sendo: 1 camada de entrada, com 2 neurônios e função de ativação sigmóide; 1 camada escondida, com 8 neurônios e função de ativação sigmóide; e 1 camada de saída, com 1 neurônio e função de ativação linear. Utilizou-se uma taxa de aprendizado igual a 0.001 e 300 épocas. Em seguida, foram gerados 200 pontos assim como instruído no item 5.11(a), e destes foram selecionadas 1000 amostras para serem aplicadas à rede, assim como instruído no item 5.11(b). O gráfico da **Figura 9** apresenta o erro MSE para os conjuntos de treinamento e teste ao longo das épocas sem a aplicação do *weight decay*. De acordo com o observado, fica evidente que a rede não possui boa capacidade de generalização, tendo em vista a diferença nos erros apresentados para o conjunto de testes.

Tal como solicitado no item 5.11(c), os gráficos da **Figura 10**, **Figura 11**, **Figura 12** e **Figura 13** apresentam o erro MSE para os conjuntos de treinamento e teste ao longo das épocas com a aplicação de diferentes valores para o weight decay, sendo que cada gráfico apresenta o erro para este parâmetro configurado como 0.001, 0.005, 0.01, 0.05, respectivamente.

Tendo em vista que, de acordo com o gráfico da **Figura 9**, fica evidente o fenômeno de *overfitting*, a aplicação da técnica de *weight decay* apresenta-se como uma medida para tentar diminuir ou eliminar este problema durante o treinamento da rede neural. A aplicação de diferentes valores para o parâmetro *weight decay*, como esperado, também impactam de diferentes formas na atenuação do overfitting. Em geral, podemos observar uma importante redução deste fenômeno para os testes realizados com todos os valores adotados — 0.001, 0.005, 0.01 e 0.05 —. É importante observar que, de acordo com o gráfico da **Figura 11**, o

[1] "TensorFlow." https://www.tensorflow.org/?hl=pt-br.

valor que mais impactou na redução do overfitting durante o treinamento da rede neural utilizada neste exercício consistiu em 0.005.

Figura 9 - MSE de treinamento e teste versus o número de épocas sem aplicação do *Weight Decay*.

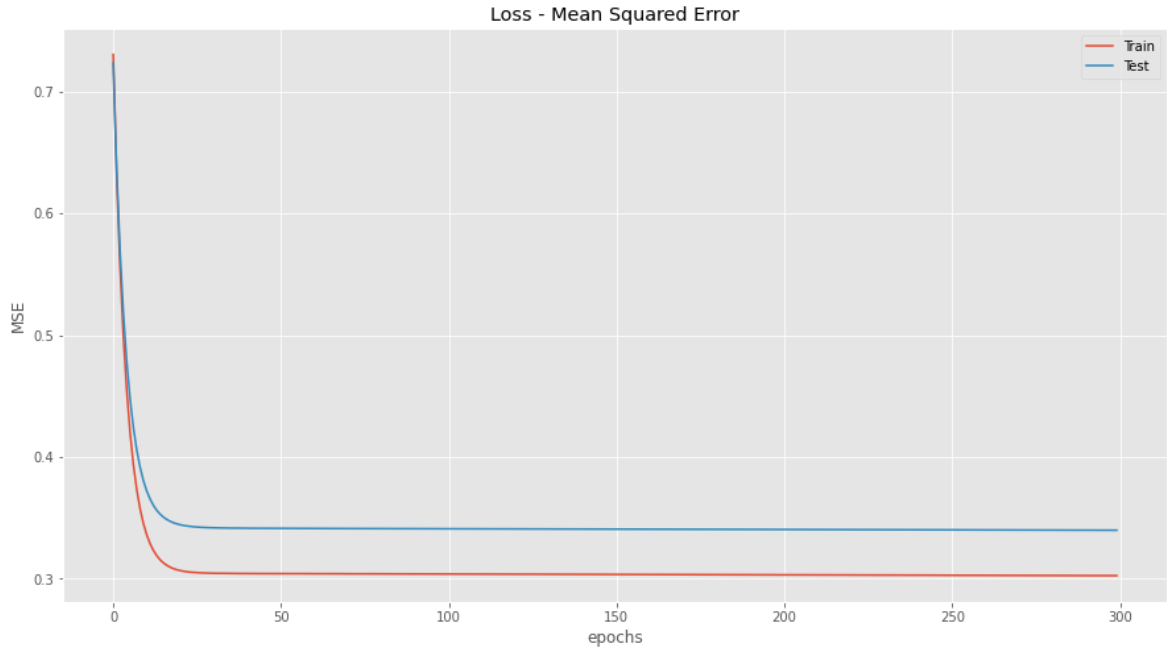


Figura 10 - MSE de treinamento e teste versus o número de épocas com *Weight Decay* (0.001).

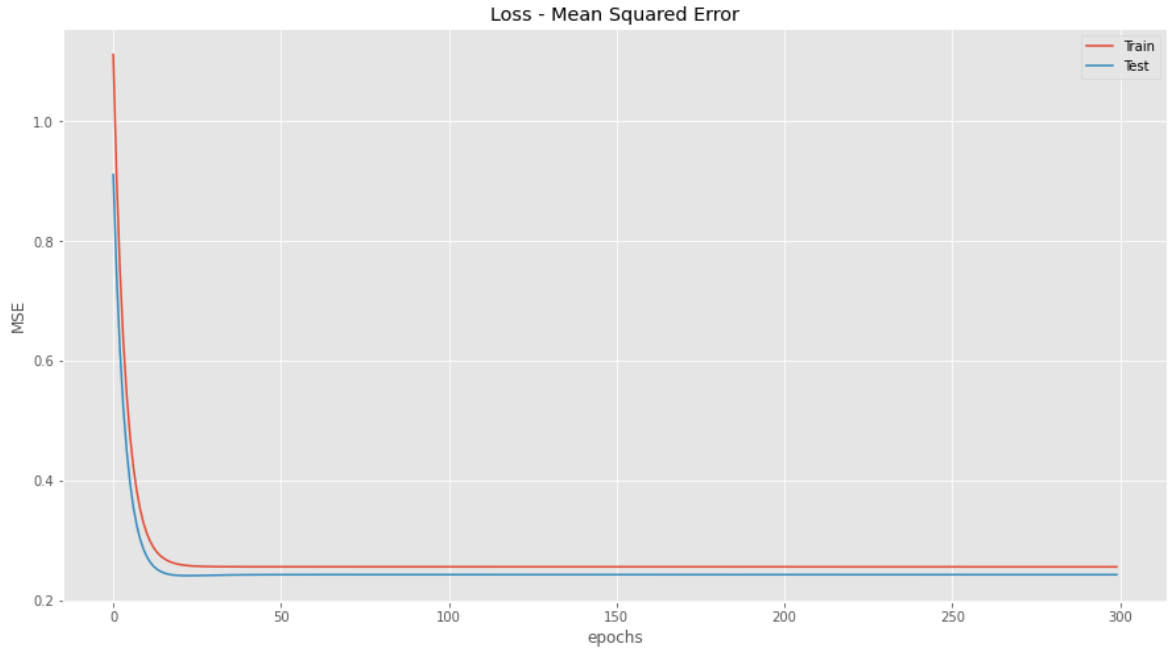


Figura 11 - MSE de treinamento e teste versus o número de épocas com *Weight Decay* (0.005).

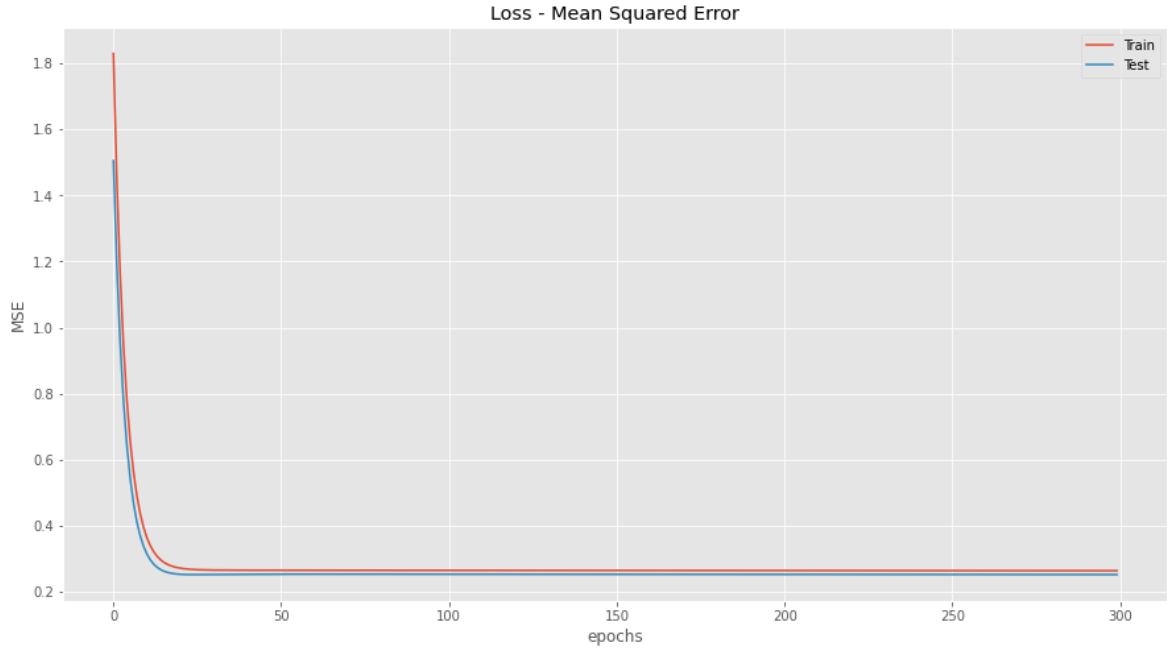


Figura 12 - MSE de treinamento e teste versus o número de épocas com *Weight Decay* (0.01).

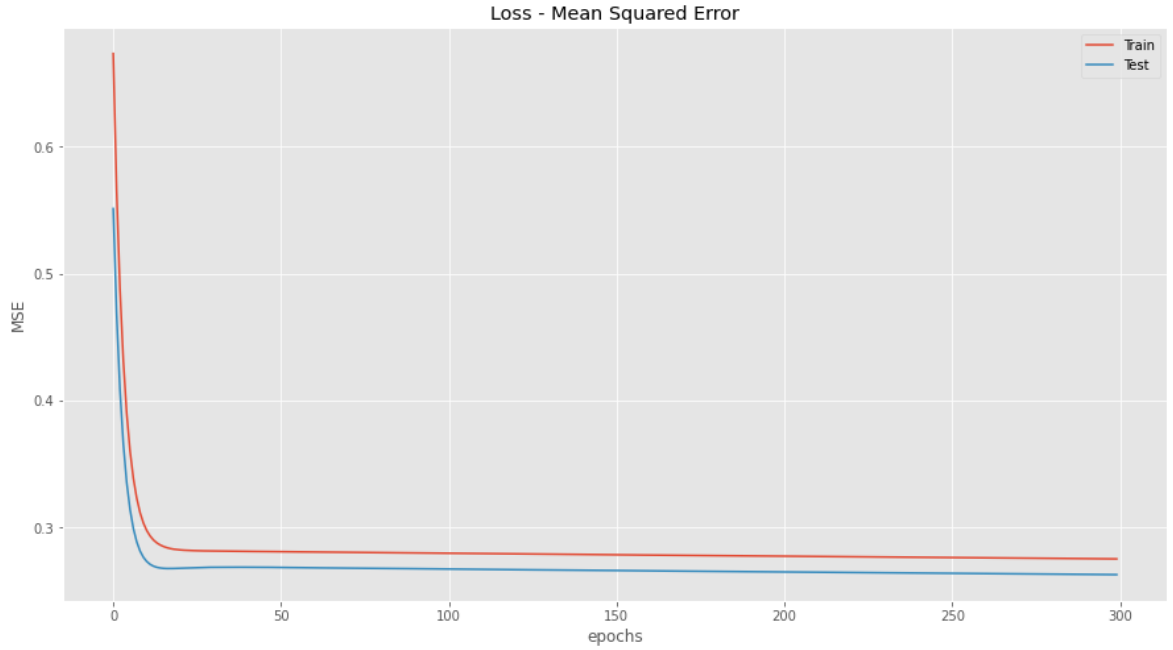
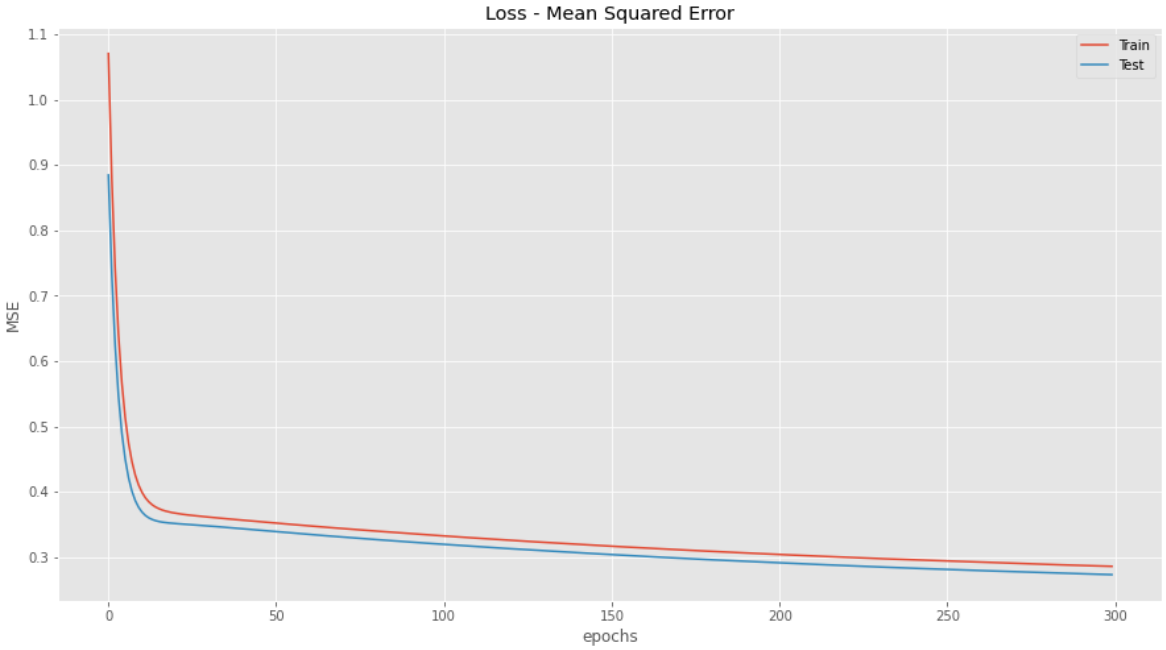


Figura 13 - MSE de treinamento e teste versus o número de épocas com *Weight Decay* (0.05).



Rede 2

O principal objetivo da implementação desta segunda rede consistiu, principalmente, numa tentativa de se apropriar dos conceitos básicos de redes neurais artificiais e ter noções práticas da implementação do perceptron, do gradiente descendente e do algoritmo de backpropagation. O procedimento adotado nesta etapa é semelhante ao adotado na **Rede 1**, e consistiu na construção de uma rede MLP com 3 camadas, sendo: 1 camada de entrada, com 2 neurônios e função de ativação sigmóide; 1 camada escondida, com 10 neurônios e função de ativação sigmóide; e 1 camada de saída, com 1 neurônio e função de ativação sigmóide. Utilizou-se uma taxa de aprendizado igual a 0.05 e 1000 épocas. Em seguida, foram gerados 200 pontos assim como instruído no item 5.11(a), e destes foram selecionadas 1000 amostras para serem aplicadas à rede, assim como instruído no item 5.11(b). O gráfico da **Figura 14** apresenta o erro MSE para os conjuntos de treinamento e teste ao longo das épocas sem a aplicação do *weight decay*. De acordo com o observado, fica evidente que a rede não possui boa capacidade de generalização, tendo em vista a diferença nos erros apresentados para o conjunto de testes. Tal como solicitado no item 5.11(c), os gráficos da **Figura 15**, **Figura 16**, **Figura 17** e **Figura 18** apresentam o erro MSE para os conjuntos de treinamento e teste ao longo das épocas com a aplicação de diferentes valores para o weight decay, sendo que cada gráfico apresenta o erro para este parâmetro configurado como 0.01, 0.05, 0.001, 0.005, respectivamente.

De acordo com o gráfico da **Figura 14**, fica evidente o fenômeno de *overfitting*. No entanto, mesmo com a aplicação da técnica de *weight decay* para tentar diminuir ou eliminar este problema durante o treinamento da rede neural, podemos perceber que, mesmo com a aplicação de diferentes valores para o parâmetro *weight decay*, não houveram mudanças significativas nos erros dos conjuntos de treinamento e testes. A principal hipótese para o mal desempenho da rede na solução deste exercício é que tenha havido erros durante o processo de codificação. Desta forma, tendo em vista que o processo de *debug* da rede neural é muito complexo, o erro não pôde ser identificado.

Figura 14 - MSE de treinamento e teste versus o número de épocas sem aplicação do *Weight Decay*.

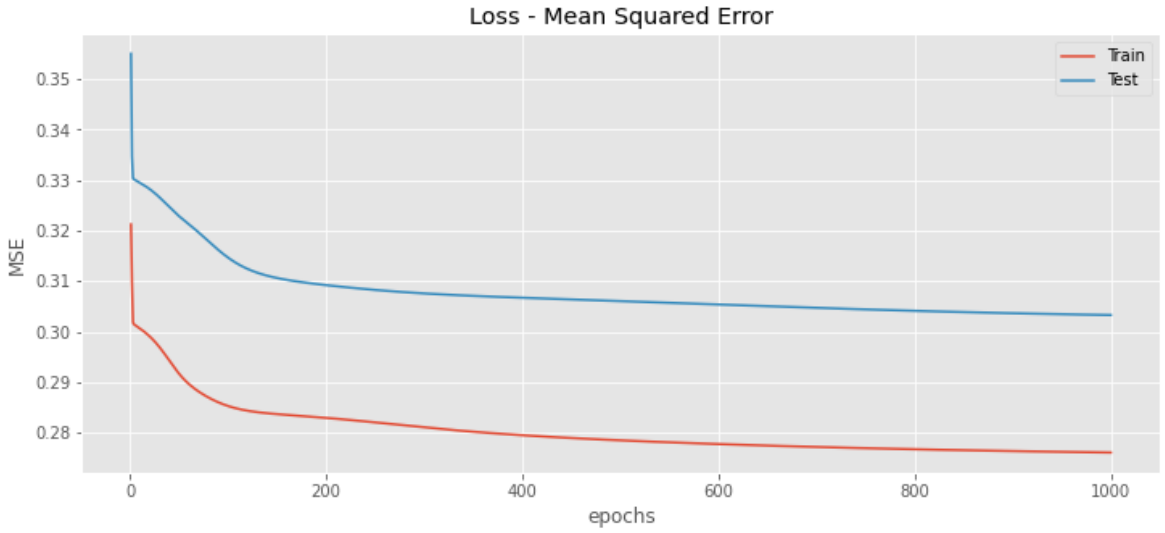


Figura 15 - MSE de treinamento e teste versus o número de épocas com *Weight Decay* (0.01).

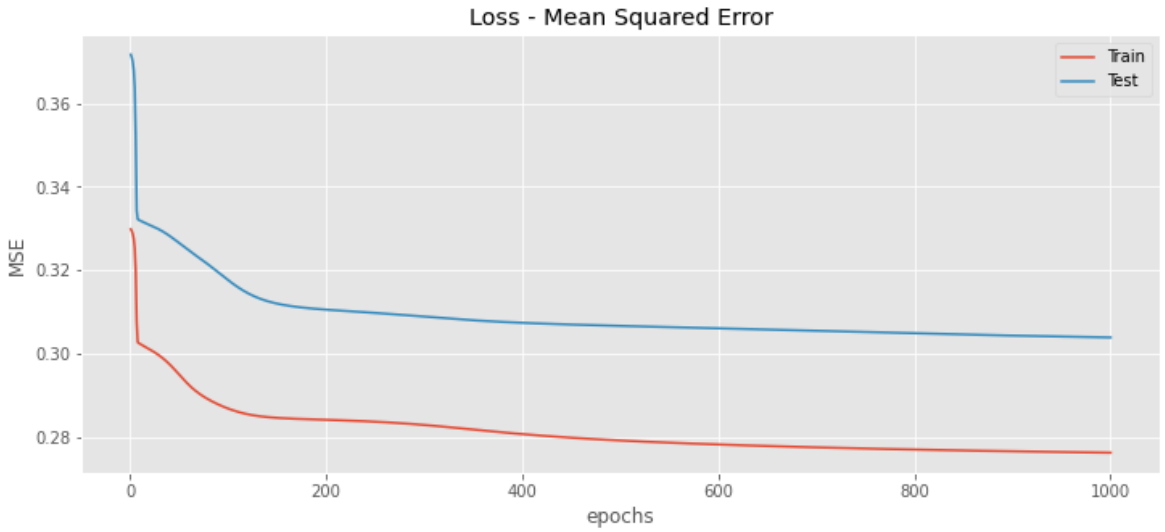


Figura 16 - MSE de treinamento e teste versus o número de épocas com *Weight Decay* (0.05).

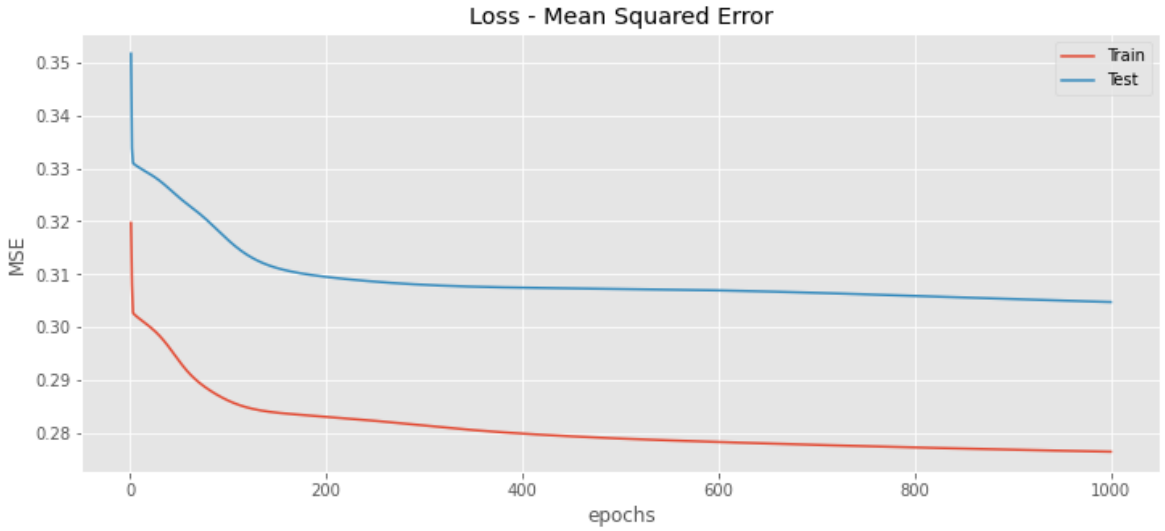
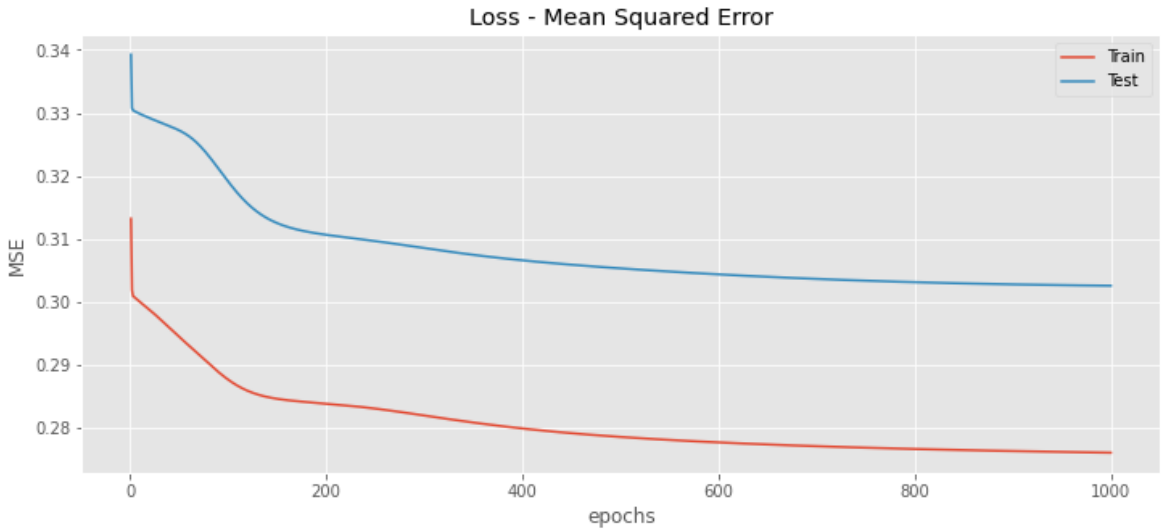


Figura 17 - MSE de treinamento e teste versus o número de épocas com *Weight Decay* (0.001).

Figura 18 - MSE de treinamento e teste versus o número de épocas com *Weight Decay* (0.005).



- As implementações para a solução deste exercício computacional podem ser acessadas através deste link.

REFERÊNCIAS

Du, K. L.; Swamy, M. N. Neural networks and statistical learning. 2a ed. Springer Science & Business Media. 2019.

NIELSEN, Michael A. Neural networks and deep learning. San Francisco, CA: Determination press, 2015.