Universidade Federal de Minas Gerais

Nome: Guilherme Vinícius Amorim

Matrícula: 2017089081

Data: 09/2020

Exercício 5

Utilizando o pacote em R *mlbench*, as bases de dados 2D *normals, xor, circle* e *spirals* foram carregadas, assim como suas respectivas entradas e saídas. Vale ressaltar que um condicionamento dos valores de Y foi necessário para que pudéssemos aplicar os treinamentos ao longo da atividade.

```
# Data for training
data1 <- mlbench.2dnormals(200)
data2 <- mlbench.xor(100)
data3 <- mlbench.circle(100)
data4 <- mlbench.spirals(100,sd = 0.05)

# Getting the X for training and it output
X1 <- data1$x
Y1 <- c(data1$classes)
X2 <- data2$x
Y2 <- c(data2$classes)
X3 <- data3$x
Y3 <- c(data3$classes)
X4 <- data4$x
Y4 <- c(data4$classes)
```

Figura 1: Rotina em R que carrega as bases de dados.

O objetivo da tarefa em questão é analisar os resultados obtidos a partir do algoritmo de ELM (*Extreme Learning Machine)* ao separarmos em duas classes os dados das 4 bases de dados em questão. No algoritmo do ELM, o primeiro passo é criarmos a matriz Z, matriz de parâmetros da camada escondida e gerada a partir de uma distribuição uniforme e tendo dimensão n+1 x p (n é o número de dimensões do problema em questão e p é o número de neurônios a ser usado na camada escondida).

```
# Calculating the hidden layer coef. matrix
Z <- replicate(p, runif(3, -0.5, 0.5))
```

Figura 2: Calculando a matriz de parâmetros Z da camada escondida.

O próximo passo é computar a matriz H, projeção da entrada na camada escondida:

```
# Projection of the input in the hidden layer
H1 <- tanh(cbind(1, X1) %*% Z)
H2 <- tanh(cbind(1, X2) %*% Z)
H3 <- tanh(cbind(1, X3) %*% Z)
H4 <- tanh(cbind(1, X4) %*% Z)
```

Figura 3: Calculando a projeção da entrada na camada escondida (matriz H).

A partir da projeção da entrada na camada escondida, devemos calcular o vetor de parâmetros W:

$$\mathbf{W} = \mathbf{H}^{+}\mathbf{Y}$$

```
# Calculating the weights vector
W1 <- pseudoinverse(H1) %*% Y1
W2 <- pseudoinverse(H2) %*% Y2
W3 <- pseudoinverse(H3) %*% Y3
W4 <- pseudoinverse(H4) %*% Y4
```

Figura 4: Calculando o vetor de pesos w.

Portanto, a parir do vetor de parâmetros w e da projeção da entrada na camada escondida H podemos encontrar o $\hat{y}$:

$$\hat{y} = HY$$

```
# Calculating Yhat
Yhat1 <- sign(H1 %*% W1)
Yhat2 <- sign(H2 %*% W2)
Yhat3 <- sign(H3 %*% W3)
Yhat4 <- sign(H4 %*% W4)
```

Figura 5: Calculando Yhat.

Uma vez explicado a implementação do algoritmo de ELM, podemos começar a analisar resultados.

Base de dados 2D *normals*

Analisando, primeiramente, a base de dados 2D *normals*, observamos que as classes 1 e 2 são praticamente lineares. Assim, com poucos neurônios o algoritmo consegue separar bem as classes. Segue abaixo os resultados obtidos para 5,10 e 30 neurônios:

```
> print(erro1)
[1] 16
```

Figura 6: Número de amostras que tiveram uma classificação equivocada para p = 5.

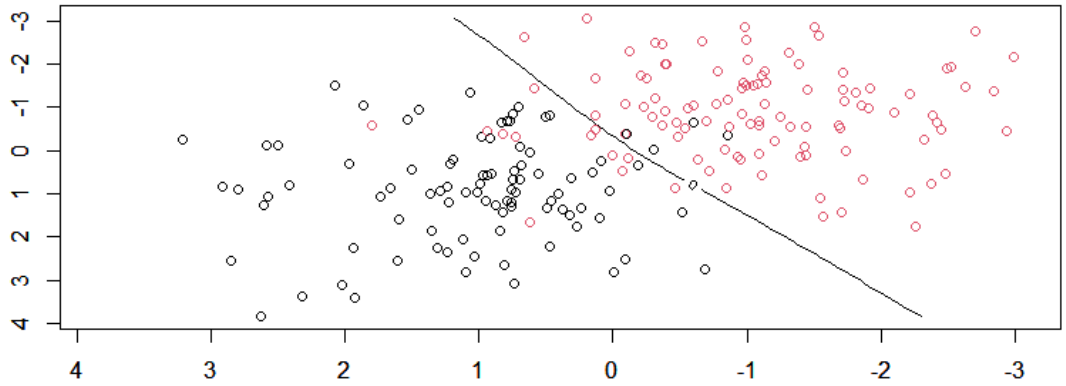

Figura 7: Amostras das classes 1 e 2 e superfície de classificação gerada pelo algoritmo do ELM para p = 5.

```
> print(erro1)
[1] 12
```

Figura 8: Número de amostras que tiveram uma classificação equivocada para p = 10.

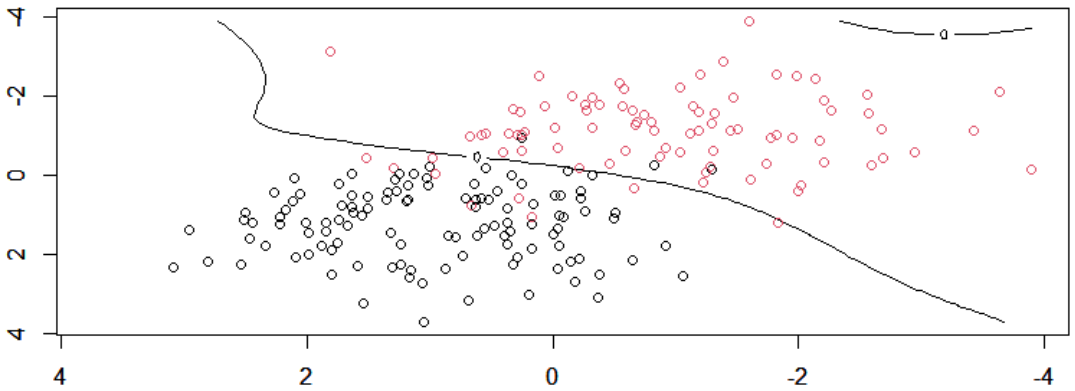

Figura 9: Amostras das classes 1 e 2 e superfície de classificação gerada pelo algoritmo do ELM para p = 10.

```
> print(erro1)
[1] 13
```

Figura 10: Número de amostras que tiveram uma classificação equivocada para p = 30.

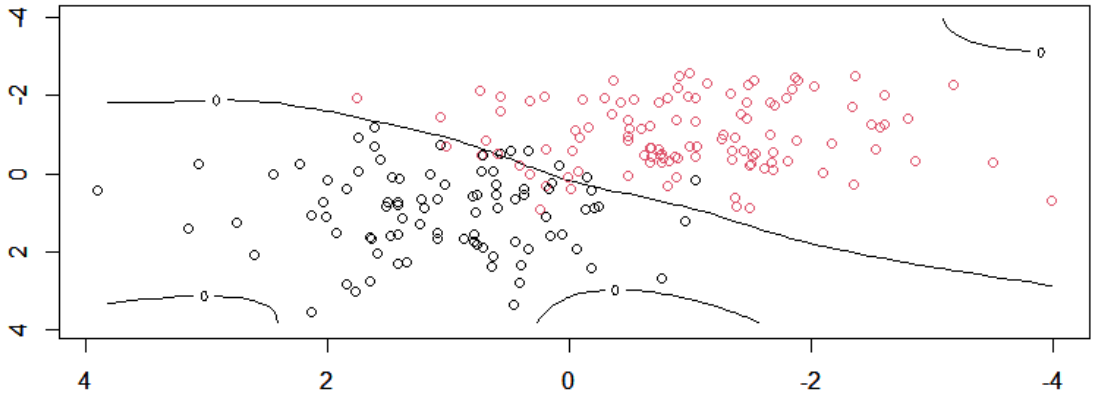

Figura 11: Amostras das classes 1 e 2 e superfície de classificação gerada pelo algoritmo do ELM para p = 30.

Observamos que para valores de p maiores que 10 acontece um *overfit*.

Base de dados XOR

O problema da XOR já não é um problema linear, tornando a análise mais interessante. Segue abaixo os resultados obtidos para p = 5, 10, 30:

```
> print(erro2)
[1] 1
```

Figura 12: Número de amostras que tiveram uma classificação equivocada para p = 5.

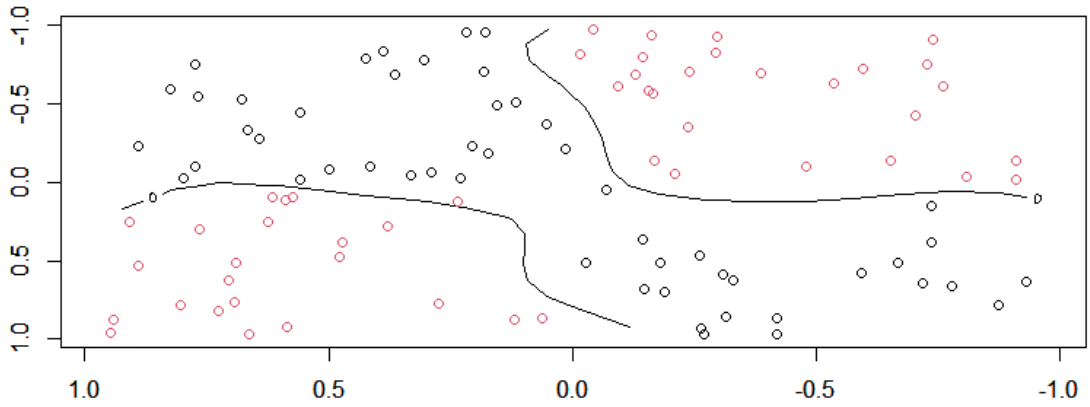

Figura 13: Amostras das classes 1 e 2 e superfície de classificação gerada pelo algoritmo do ELM para p = 5.

```
> print(erro2)
[1] 7
```

Figura 14: Número de amostras que tiveram uma classificação equivocada para p = 10.

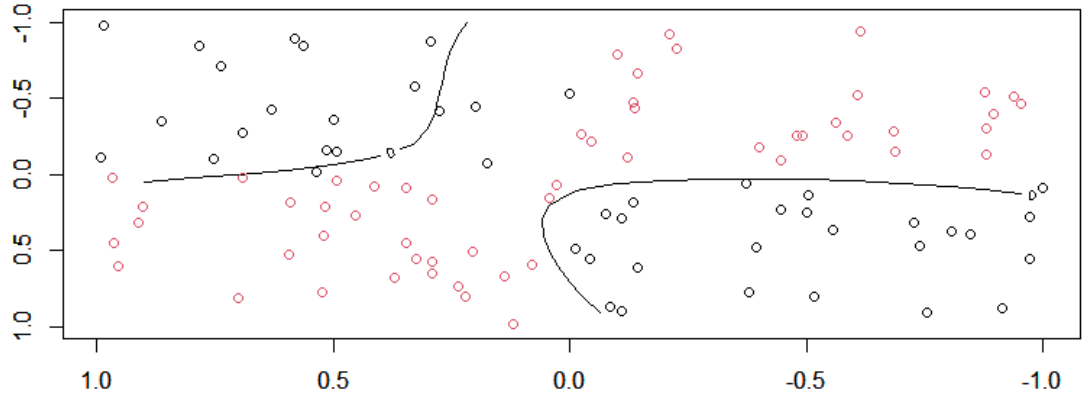

Figura 15: Amostras das classes 1 e 2 e superfície de classificação gerada pelo algoritmo do ELM para p = 10.

```
> print(erro2)
[1] 2
```

Figura 16: Número de amostras que tiveram uma classificação equivocada para p = 30.

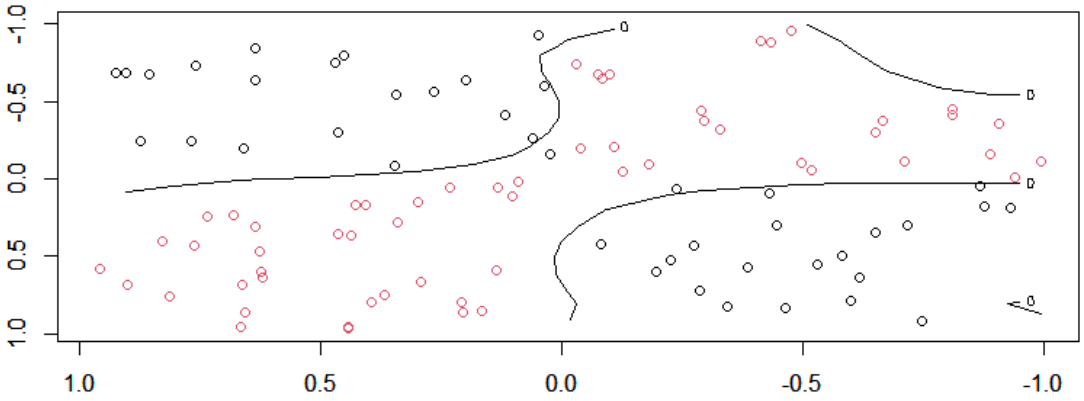

Figura 17: Amostras das classes 1 e 2 e superfície de classificação gerada pelo algoritmo do ELM para p = 30.

Nesse exemplo em questão, o *overfit* ficou mais claro para p = 30. Para p = 5 tivemos um erro menor que para p = 10, inclusive.

Base de dados *Circle*

O problema da base de dados circle já um problema mais desafiador, fazendo com que o número de neurônios para uma boa classificação cresça em relação aos exemplos anteriores. Segue abaixo os resultados obtidos para p = 10, 20, 40:

```
> print(erro3)
[1] 6
```

Figura 18: Número de amostras que tiveram uma classificação equivocada para p = 10.

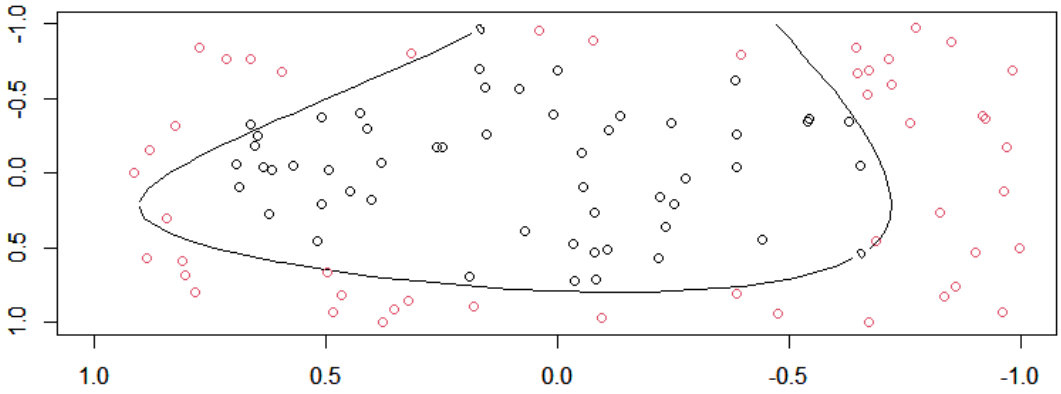

Figura 19: Amostras das classes 1 e 2 e superfície de classificação gerada pelo algoritmo do ELM para p = 10.

```
> print(erro3)
[1] 5
```

Figura 20: Número de amostras que tiveram uma classificação equivocada para p = 20.

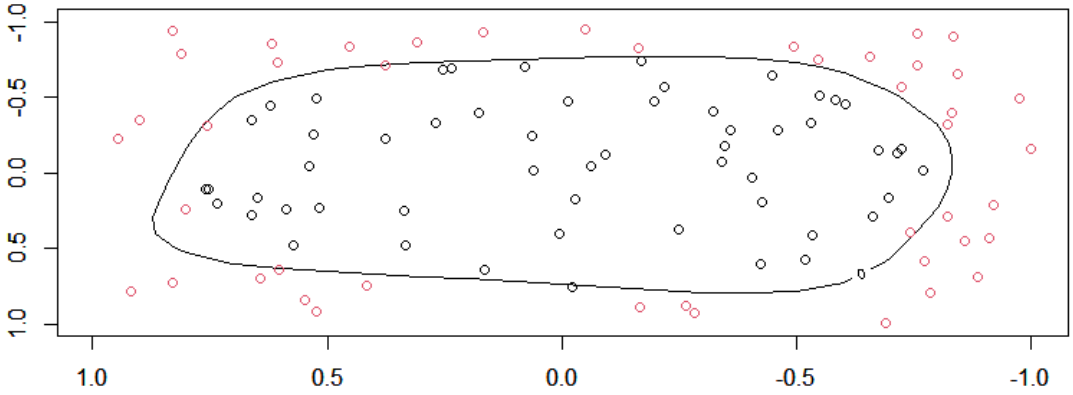

Figura 21: Amostras das classes 1 e 2 e superfície de classificação gerada pelo algoritmo do ELM para p = 20.

```
> print(erro3)
[1] 2
```

Figura 22: Número de amostras que tiveram uma classificação equivocada para p = 40.

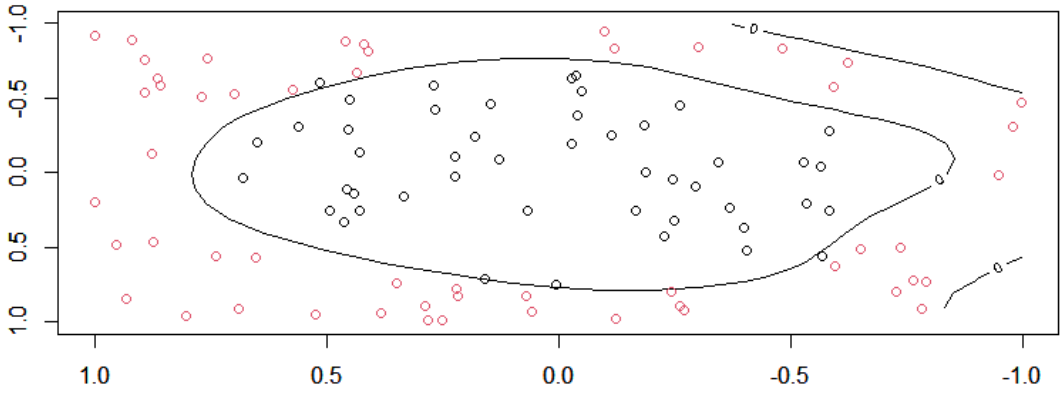

Figura 23: Amostras das classes 1 e 2 e superfície de classificação gerada pelo algoritmo do ELM para p = 40.

Nesse exemplo em questão, o *overfit* se mostrou presente para as amostras com p = 40 somente. Esse problema em questão necessita de um número p maior, o que justifica o *overfit* somente para um p elevado.

Base de dados *Spirals*

O problema da base de dados *Spirals* é o problema mais desafiador dos 4 em questão. Dessa forma, um número considerável de neurônios será agora necessário para a correta classificação das classes. Segue abaixo os resultados obtidos para p = 20, 40, 60:

```
> print(erro4)
[1] 1
```

Figura 24: Número de amostras que tiveram uma classificação equivocada para p = 20.

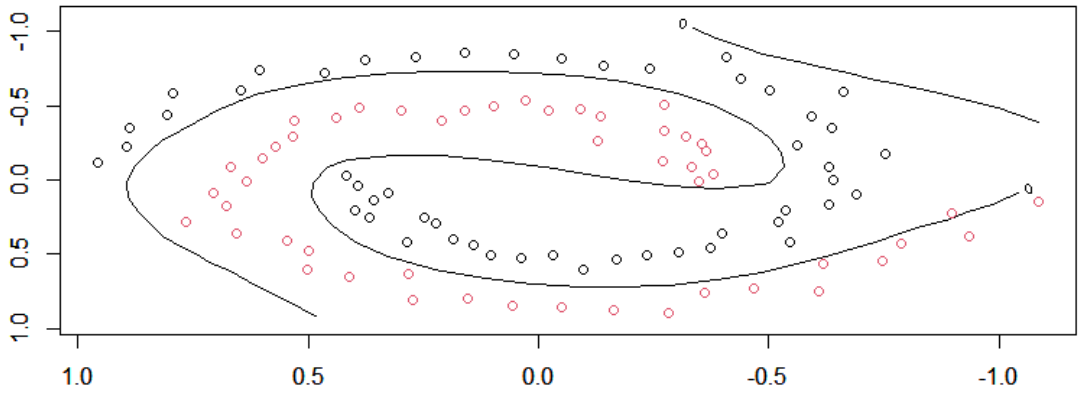

Figura 25: Amostras das classes 1 e 2 e superfície de classificação gerada pelo algoritmo do ELM para p = 20.

```
> print(erro4)
[1] 0
```

Figura 26: Número de amostras que tiveram uma classificação equivocada para p = 40.

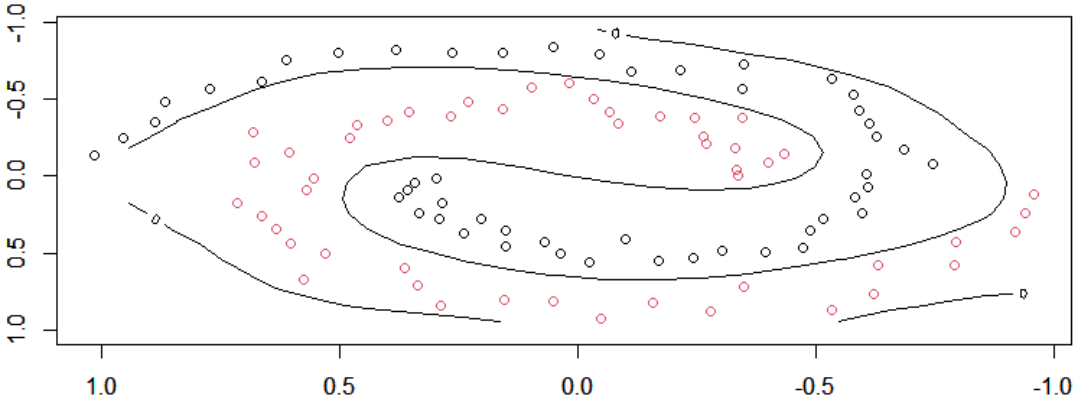

Figura 27: Amostras das classes 1 e 2 e superfície de classificação gerada pelo algoritmo do ELM para p = 40.

```
> print(erro4)
[1] 0
```

Figura 28: Número de amostras que tiveram uma classificação equivocada para p = 60.

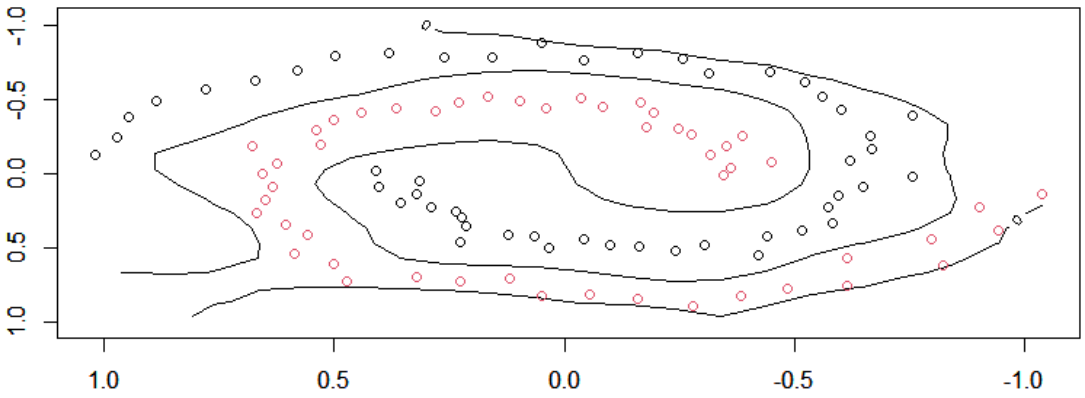

Figura 29: Amostras das classes 1 e 2 e superfície de classificação gerada pelo algoritmo do ELM para p = 60.

Nesse exemplo em questão, observamos que para p = 40 não há erro de classificações. Contudo, a partir da curva obtida observa-se o início de um *overfit. Overfit* tal bem mais marcante na curva observada para p = 60.

Assim, a partir da análise dos 4 exemplos em questão foi possível observar o Teorema de Cover e o funcionamento do algoritmo do ELM. Assim, ficou claro que mesmo para problemas não lineares a expansão do problema para um espaço de alta dimensão é capaz de tornar o problema linearmente solucionável. Contudo, foi observado também que para um aumento muito grande no número de dimensões do espaço da camada escondida consegue-se classificar melhor as classes, mas isso acaba sendo um grande gerador de *overfit.*