UNIVERSIDADE FEDERAL DE MINAS GERAIS
PROGRAMA DE PÓS - GRADUAÇÃO EM ENGENHARIA ELÉTRICA

EXERCÍCIO V - CLASSIFICADORES NÃO LINEARES ELM

Matheus Barros Oliveira

15 de novembro de 2024

Relatório - Classificadores Não Lineares ELM

Relatório sobre a Classificadores Não Lineares ELM

15 de novembro de 2024

1 Introdução

Extreme Learning Machines (ELMs) são redes neurais de camada única projetadas para lidar com problemas de classificação e regressão de forma eficiente. A estrutura das redes ELM são, na verade, a mesma estrutura das Redes Perceptron de Múltipas Camadas [MLP], sendo que a pricipal diferença entre elas é a forma em que elas são treinadas [de Pádua Braga, 2024].Diferentemente das redes tradicionais, as ELMs se destacam por seu treinamento extremamente rápido, pois os pesos da camada oculta são definidos aleatoriamente e permanecem fixos, enquanto os pesos da camada de saída são ajustados de forma analítica. Essa abordagem permite que as ELMs capturem relações não lineares complexas, tornando-as ideais para tarefas que exigem alta capacidade de generalização em tempo reduzido.

2 Objetivo

Este estudo tem como objetivo analisar a capacidade de generalização das ELMs e examinar as superfícies de separação geradas por diferentes configurações da rede. A visualização dessas superfícies busca avaliar como as formas de separação refletem os efeitos de *overfitting* e *underfitting*.

3 Materiais e Métodos

Para a execução deste exercício foram utilizados as seguintes ferramentas e base de dados descritas abaixo.

3.1 Materiais

Utilizou-se o RStudio (versão 4.4.1 (2024-06-14 ucrt) instalada em um Windows 11 (versão 23H2) com um processador 12th Gen Intel(R) Core(TM) i5-12500H 2.50 GHz e 32GB de memória RAM.

3.2 Descrição da Base de Dados

Foram utilizadas as seguintes bases de dados no estudo:

- **mlbench.2dnormals(200)**: Conjunto de dados com duas classes normais bidimensionais, gerado com 200 pontos.

- **mlbench.xor(100)**: Problema clássico de separação não linear no formato ”XOR”, contendo 100 pontos.
- **mlbench.circle(100)**: Conjunto de dados com dois círculos concêntricos, gerado com 100 pontos.
- **mlbench.spirals(100, sd = 0.05)**: Duas espirais entrelaçadas, geradas com 100 pontos e desvio padrão de 0.05.

3.3 Metodologia

Para cada conjunto de dados as ELM's foram treinadas variando-se o número de neurônicos p em: $p = 5$, $p = 20$, $p = 50$, $p = 80$, $p = 100$ e $p = 500$, ou seje, para cada conjunto de dados tem-se seis modelos para serem avaliados.

3.3.1 Passo 1: Carregamento da Base de Dados

Iniciamento realizou-se o carregamento do conjunto de dados para o ambiente R.

3.3.2 Passo 2: Separação do Conjunto de Testes e Treino

O conjunto de dados foi separado em 70% (conjunto de treino) e 30% (conjutno de validação) conforme pode-se observar no código em ”R”abaixo:

```
data_spirals <- mlbench.spirals(100, sd = 0.05)
xin <- data_spirals$x
yin <- as.numeric(data_spirals$classes)
n_total <- nrow(xin)
indices_treino <- sample(1:n_total, size = floor(0.7 * n_
   total))
indices_validacao <- setdiff(1:n_total, indices_treino)
xin_treino <- xin[indices_treino, ]
yin_treino <- yin[indices_treino]
xin_validacao <- xin[indices_validacao, ]
yin_validacao <- yin[indices_validacao]
```

Listing 1: Divisão do conjunto de dados Spirals em treino e validação

3.3.3 Passo 3: Conversão das Classes

As classes dos conjuntos de dados estavam classificadas como ”1”e ”2”, desta forma foi-se necessário convertê-las para -1”e ”1”. Pode-se observar abaixo o código que realiza a conversão:

```
yin_treino <- ifelse(yin_treino == 1, -1, 1)
yin_validacao <- ifelse(yin_validacao == 1, -1, 1)
```

Listing 2: Covnersão de Classes

3.3.4 Passo 4: Visualização dos Dados

Com os dados carregas e tratados, foi necessário realizar a visualização dos conjuntos de treino e validação para se observar a distribuição dos mesmos.

3.3.5 Passo 5: Treinamento da ELM

Após a avaliação dos conjuntos de treino e teste foi realizado o treinamento da ELM conforme a função abaixo:

```
treinaELM <- function(xin, yin, p, par) {
Carregar o pacote corpcor, necessario para a funcao de
   pseudoinversa
  library(corpcor)
Obter o numero de colunas (dimensao) da matriz de entrada
   xin
  n <- dim(xin)[2]
Verificar o valor do parametro par para ajustar a matriz
   xin e definir Z
  if (par == 1) {
Adiciona uma coluna de 1s a matriz de entrada xin (bias)
    xin <- cbind(1, xin)
Gera uma matriz Z com valores aleatorios uniformemente
   distribuidos
(n+1) linhas e p colunas, com valores no intervalo [-0.5,
   0.5]
    Z <- matrix(runif((n + 1) * p, -0.5, 0.5), nrow = (n +
       1), ncol = p)
  } else {
```

```
Gera a matriz Z sem adicionar a coluna de bias (n linhas e
   p colunas)
    Z <- matrix(runif(n * p, -0.5, 0.5), nrow = n, ncol = p
       )
  }
Calcula a funcao de ativacao (tangente hiperbolica) na
   camada oculta
  H <- tanh(xin %*% Z)
Adiciona uma coluna de 1s a matriz H para incluir o bias
  Haug <- cbind(1, H)
Calcula os pesos da camada de saida usando a pseudoinversa
  W <- pseudoinverse(Haug) %*% yin
Retorna os pesos da camada de saida, a matriz H e a matriz
   Z
  return(list(W, H, Z))
}
```

Listing 3: Função de Treinamento da ELM

Conforme dito anteriormente, para cada conjunto de testes o parâmetro ”p”foi variado em: $p = 5$, $p = 20$, $p = 50$, $p = 80$, $p = 100$ e $p = 500$ e os parâmetros de treinamento gerados foram utilizados na função de validação que será descrita abaixo.

3.3.6 Passo 6: Teste da ELM

Após o treinamento da ELM o cojunto de testes foi validado através da função abaixo:

```
YELM <- function(xin, Z, W, par) {
Obter o numero de colunas (dimensao) da matriz de entrada
   xin
  n <- dim(xin)[2]
Verificar o valor do parametro par para ajustar a matriz
   xin
  if (par == 1) {
Adiciona uma coluna de 1s a matriz de entrada xin (bias)
    xin <- cbind(1, xin)
  }
Calcula a funcao de ativacao (tangente hiperbolica) na
   camada oculta
```

```
  H <- tanh(xin %*% Z)
Adiciona uma coluna de 1s a matriz H para incluir o bias
  Haug <- cbind(1, H)
Calcula as previsoes (classificacao) usando o sinal do
   produto interno
  Yhat <- sign(Haug %*% W)
Retorna as previsoes
  return(Yhat)
}
```

Listing 4: Função de Validação da ELM

3.3.7 Passo 6: Visualização de Superfície

Após a realização dos testes de validação da ELM cada conjunto de validação obtido pela variação do parâmetro ”p”os resultados foram analisados através do gráfico de visualização de superfície. Pode-se observar abaixo o código utilizado para a plotagem da superfície de separação:

```
Plotar a superficie de separacao
seqx1x2 <- seq(min(xin) - 1, max(xin) + 1, 0.1)
lseq <- length(seqx1x2)
MZ <- matrix(nrow = lseq, ncol = lseq)
for (i in 1:lseq) {
  for (j in 1:lseq) {
    x1 <- seqx1x2[i]
    x2 <- seqx1x2[j]
    x1x2 <- as.matrix(cbind(1, x1, x2))  # Adiciona o bias
    h1 <- cbind(1, tanh(x1x2 %*% Z))      # Calcula a
       ativacao na camada oculta
    MZ[i, j] <- sign(h1 %*% W)            # Calcula a
       previsao
  }
}
main_title <- paste("Superficie de Separacao (XOR) p =", p)
contour(seqx1x2, seqx1x2, MZ, nlevels = 1,
        xlim = range(seqx1x2), ylim = range(seqx1x2),
        xlab = "X1", ylab = "X2",
        main = main_title, drawlabels = FALSE, col = "black
           ")
```

```
Adicionar os pontos de validacao ao grafico
points(xin_validacao[yin_validacao == -1, 1], xin_validacao
   [yin_validacao == -1, 2], col = "red", pch = 19)
points(xin_validacao[yin_validacao == 1, 1], xin_validacao[
   yin_validacao == 1, 2], col = "blue", pch = 19)
```

Listing 5: Superfície de Separação

3.3.8 Passo 6: Visualização de Superfície 3D

Para a verificação da influência da variação do parâmetro ”p”na superfície de separação do conjunto de validação, foi plotado a superfície de seperação em gráfico 3D para o parâmetro $p = 5$ e $p = 500$. Pode-se observar abaixo o código utilizado:

```
# Plotar a superficie de separacao em 3D
persp3d(seqx1x2, seqx1x2, MZ, col = "red",
        xlim = range(seqx1x2), ylim = range(seqx1x2),
        zlim = c(min(MZ), max(MZ)),  # Definindo os limites
            do eixo Z
        xlab = "X1", ylab = "X2", zlab = "Classe")
```

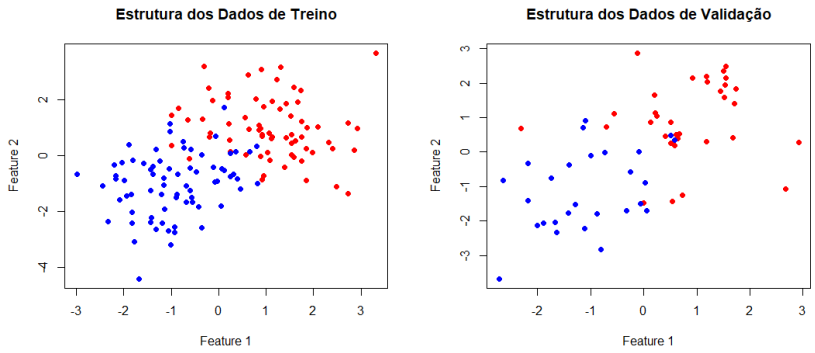
Listing 6: Superfície de Separação

4 Resultados

Pode-se observar abaixo os resultados obtidos para os experimento:

4.1 Base de Dados: mlbench.2dnormals

Pode-se obersar na Figura 1 o cojunto de treinamento e o conjunto de validação da base de dados mlbench.2dnormals.

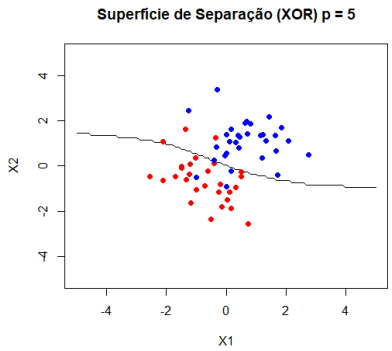


(a) Conjunto de Treinamento (b) Conjunto de Validação

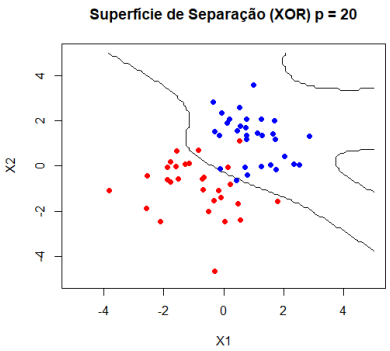
Figura 1: Conjunto de Treinamento e Validação do Conjunto de Dados: mlbench.2dnormals

Ao se observar tanto o conjunto de treinamento e de validação acima percebe-se que o conjunto de dados é linearmente separável já que eles foram gerados a partir de distruições normais com médias específicas. Isso significa que os pontos são distribuídos ao redes de suas respectivas médias (centroides) com uma certa dispersão definica pela variância.

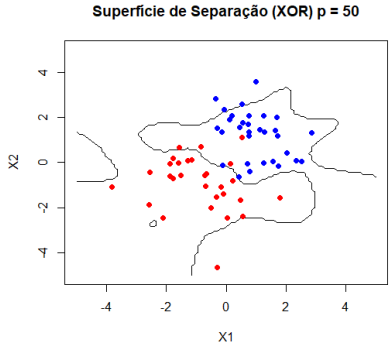
Pode-se obsevar na Figura 2 a distruição do conjunto de validação para a variação de p em: $p = 5$, $p = 20$, $p = 50$, $p = 80$, $p = 100$ e $p = 500$. Como o conjunto de análise é linearmente separável, o treinamento da ELM com $p = 5$ praticamente conseguiu realizar a separação dos conjuntos ”x1”e ”x2”. Conforme o valor de ”p”era aumentado percebe-se que o modelo se torna cada vez mais complexo gerando um *overfitting* do conjunto de treinamento, onde pode-se perceber a formação de certas ”ilhas”ao redor dos dados de validação. Em função da característica do conjunto de dados utilziado pode-se concluir que o modelo apresentou uma acurácia melhor para $p = 20$, pois ela consegue separar melhor o conjunto sem gerar muito *overfiting*.



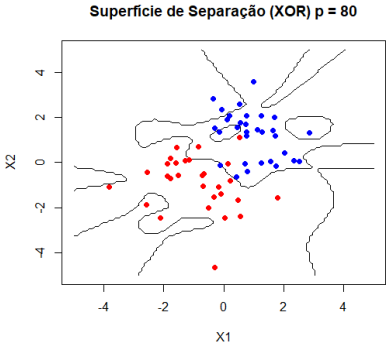
(a) Conjunto Validação p = 5



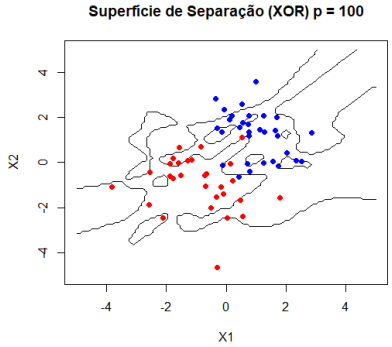
(b) Conjunto Validação p = 20



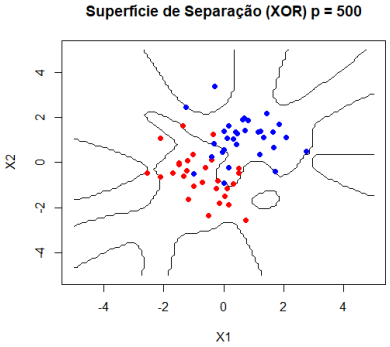
(c) Conjunto Validação p = 50



(d) Conjunto Validação p = 80



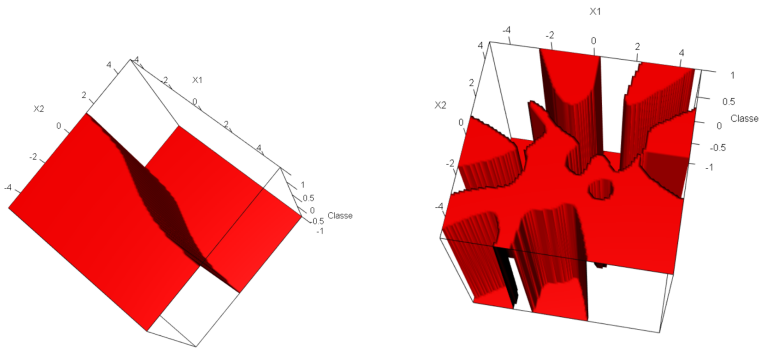
(e) Conjunto Validação p = 100



(f) Conjunto Validação p = 500

Figura 2: Gráficos das Superfícies de Separação do Conjunto de Dados: mlbench.2dnormals

Por fim, ao se observar a Figura 3 a superfície de separação em 3D para p = 5 e p = 500 percebe-se que quanto maior o valor de ”p”maior a complexidade do modelo.

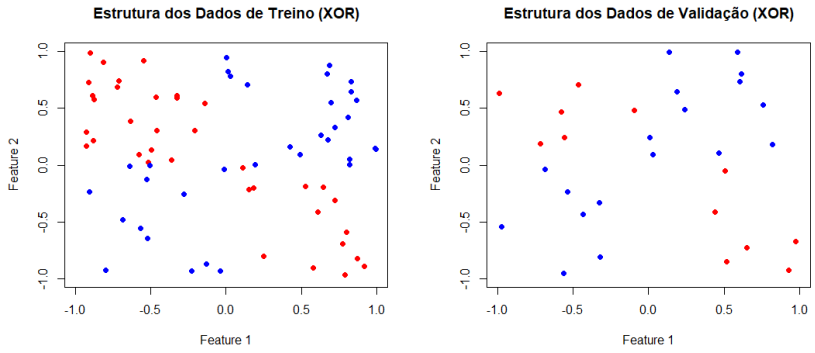


(a) Conjunto Validação p = 5 (b) Conjunto Validação p = 500

Figura 3: Gráficos das Superfícies 3D de Separação do Conjunto de Dados: mlbench.2dnormals

4.2 Base de Dados: mlbench.xor

Pode-se obersar na Figura 4 o cojunto de treinamento e o conjunto de validação da base de dados mlbench.2dnormals.

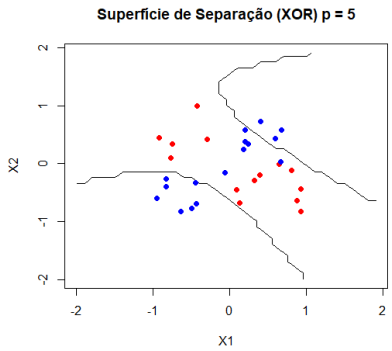


(a) Conjunto de Treinamento (b) Conjunto de Validação

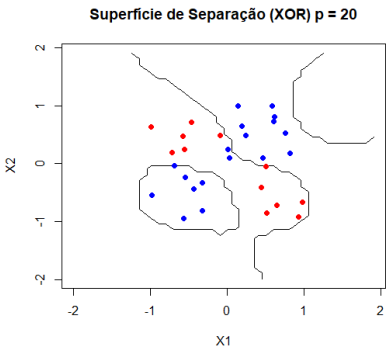
Figura 4: Conjunto de Treinamento e Validação do Conjunto de Dados: mlbench.xor

Ao se observar tanto o conjunto de treinamento e de validação acima percebe-se que o conjunto de dados não é linearmente seprável porque os pontos das duas classes estão organizados de forma que não é possível separar as classes com uma única linha reta. No padrão ”XOR”as classes estão dispostas nos cantos opostos de um espaço bidimensional.

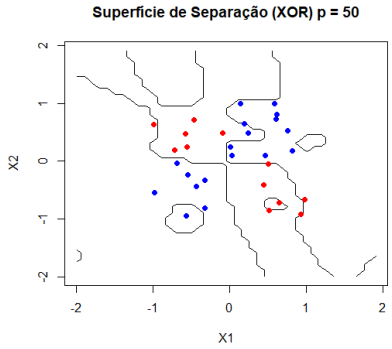
Pode-se obsevar na Figura 5 a distruição do conjunto de validação para a variação de p em: $p = 5$, $p = 20$, $p = 50$, $p = 80$, $p = 100$ e $p = 500$. Diferentemente do conjunto de testes mlbench.2dnormals o conjunto de dados mlbench.xor não é linearmente separável, mas tem um comportamente semelhante à variação do ”p”, ou seja, quanto maior o valor de ”p”mais o modelo se torna mais complexo. Desta forma, pode-se concluir que os valores de $p = 20$ e $p = 500$ aparentemente obtiveram a melhor acurácia já que conseguiram separar os conjuntos de uma forma mais eficiente. Por exemplo, o valor de $p = 80$ gerou muitas muitas ”ilhas”de separação o que consequentemente impactaram na acurácia do modelo.



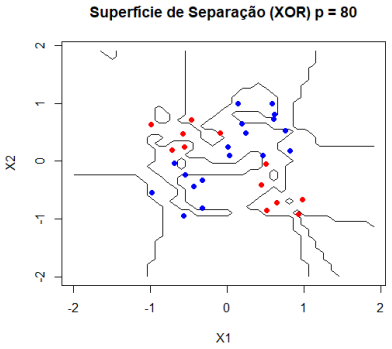
(a) Conjunto Validação p = 5



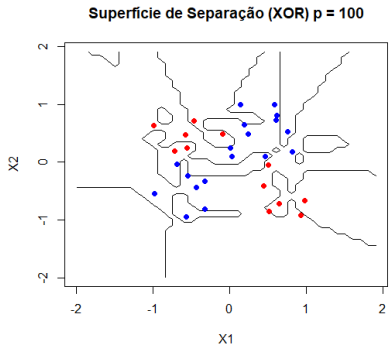
(b) Conjunto Validação p = 20



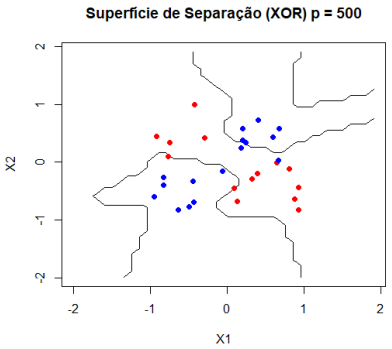
(c) Conjunto Validação p = 50



(d) Conjunto Validação p = 80



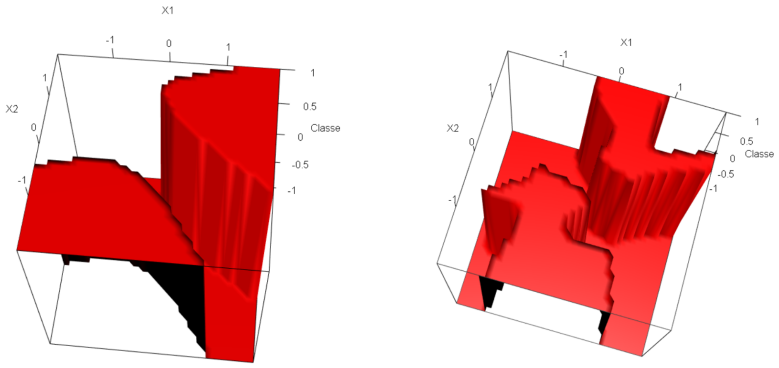
(e) Conjunto Validação p = 100



(f) Conjunto Validação p = 500

Figura 5: Gráficos das Superfícies de Separação do Conjunto de Dados: mlbench.xor

Por fim, ao se observar a Figura 6 a superfície de separação em 3D para p = 5 e p = 500 percebe-se que quanto maior o valor de ”p”maior a complexidade do modelo.

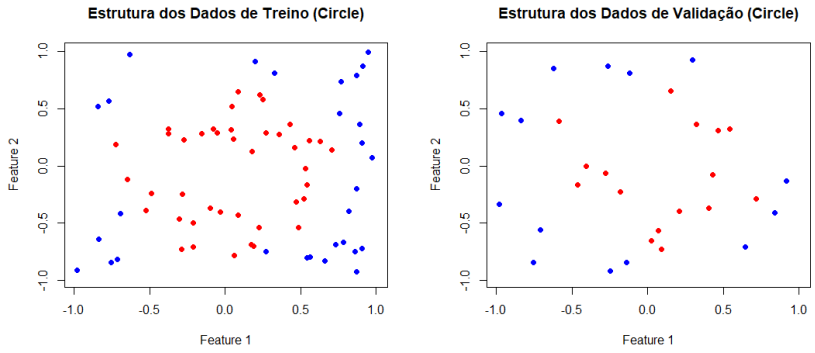


(a) Conjunto Validação p = 5 (b) Conjunto Validação p = 500

Figura 6: Gráficos das Superfícies 3D de Separação do Conjunto de Dados: mlbench.xor

4.3 Base de Dados: mlbench.circle

Pode-se obersar na Figura 7 o cojunto de treinamento e o conjunto de validação da base de dados mlbench.circle.

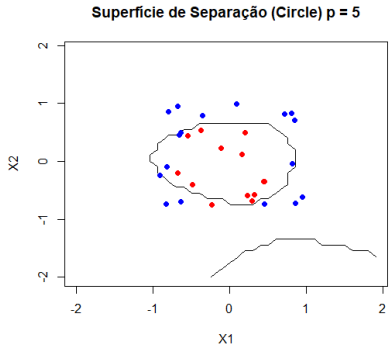


(a) Conjunto de Treinamento (b) Conjunto de Validação

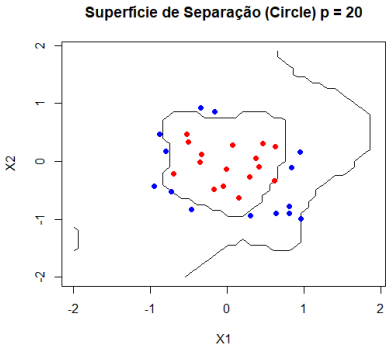
Figura 7: Conjunto de Treinamento e Validação do Conjunto de Dados: mlbench.circle

Ao se observar tanto o conjunto de treinamento e de validação acima percebe-se que o conjunto de dados não é linearmente seprável porque os pontos das duas classes estão organizados de forma que não é possível separar as classes com uma única linha reta, já que o conjunto de dados é distribuído em dois circulos concênctricos: um círculo interno representando uma classe e um círculo externo representando outra classe.

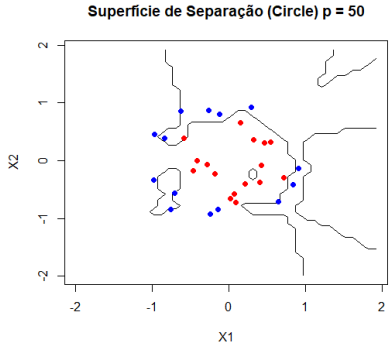
Pode-se obsevar na Figura 8 a distruição do conjunto de validação para a variação de p em: $p = 5$, $p = 20$, $p = 50$, $p = 80$, $p = 100$ e $p = 500$. Diferentemente do conjunto de testes mlbench.2dnormals o conjunto de dados mlbench.circle não é linearmente separável, mas tem um comportamente semelhante à variação do ”p”, ou seja, quanto maior o valor de ”p”mais o modelo se torna mais complexo. Desta forma, pode-se concluir que os valores de $p = 5$ e $p = 20$ aparentemente obtiveram a melhor acurácia já que conseguiram separar os conjuntos de uma forma mais eficiente. Por exemplo, o valor de $p = 100$ gerou muitas ”ilhas”de separação o que consequentemente impactaram na acurácia do modelo.



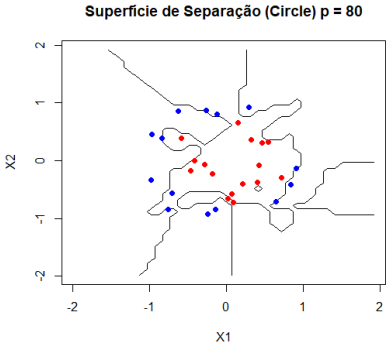
(a) Conjunto Validação p = 5



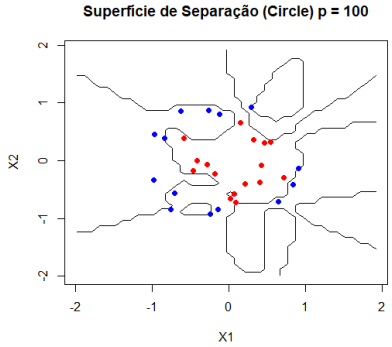
(b) Conjunto Validação p = 20



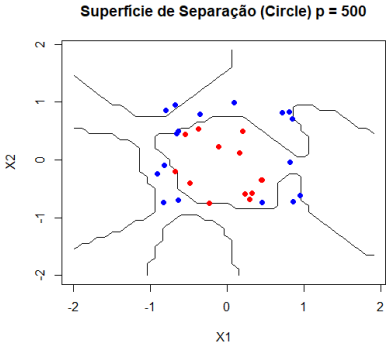
(c) Conjunto Validação p = 50



(d) Conjunto Validação p = 80



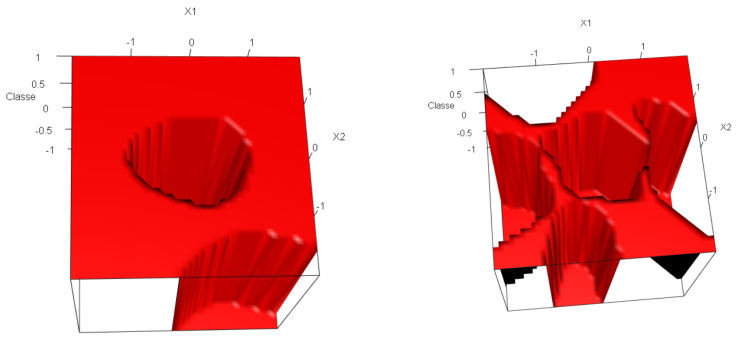
(e) Conjunto Validação p = 100



(f) Conjunto Validação p = 500

Figura 8: Gráficos das Superfícies de Separação do Conjunto de Dados: mlbench.circle

Por fim, ao se observar a Figura 9 a superfície de separação em 3D para p = 5 e p = 500 percebe-se que quanto maior o valor de ”p”maior a complexidade do modelo.

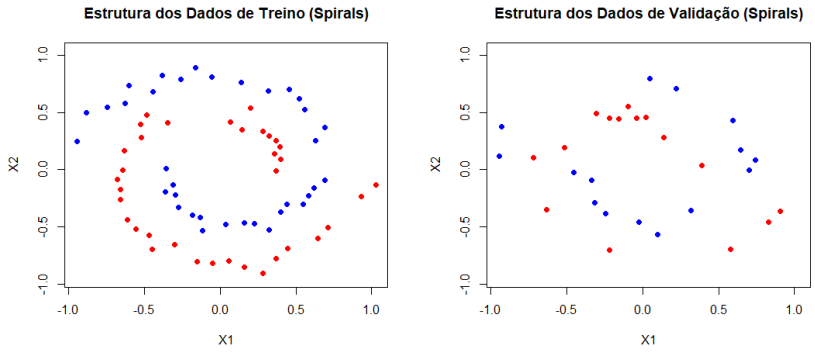


(a) Conjunto Validação p = 5 (b) Conjunto Validação p = 500

Figura 9: Gráficos das Superfícies 3D de Separação do Conjunto de Dados: mlbench.circle

4.4 Base de Dados: mlbench.spirals

Pode-se obersar na Figura 10 o cojunto de treinamento e o conjunto de validação da base de dados mlbench.spirals.

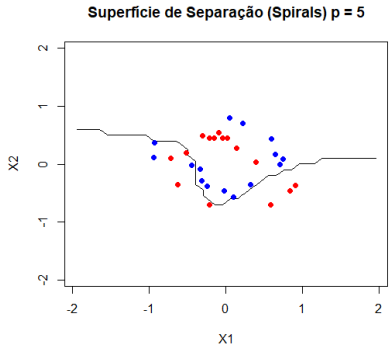


(a) Conjunto de Treinamento (b) Conjunto de Validação

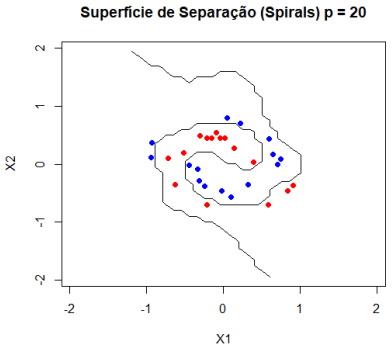
Figura 10: Conjunto de Treinamento e Validação do Conjunto de Dados: mlbench.spirals

Ao se observar tanto o conjunto de treinamento e de validação acima percebe-se que o conjunto de dados não é linearmente seprável porque os pontos das duas classes estão organizados em duas espirais que estão entrelaçadas uma com a outra. Ese padrão altamente não linear não é possível traçar uma linha reta para separar as duas classes.

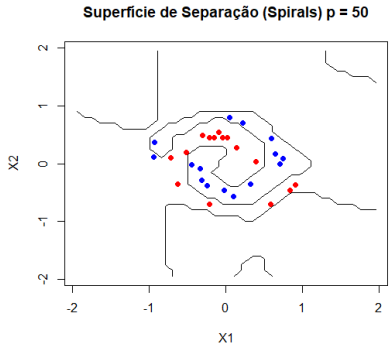
Pode-se obsevar na Figura 11 a distruição do conjunto de validação para a variação de p em: $p = 5$, $p = 20$, $p = 50$, $p = 80$, $p = 100$ e $p = 500$. Diferentemente do conjunto de testes mlbench.2dnormals o conjunto de dados mlbench.spirals não é linearmente separável, mas tem um comportamente semelhante à variação do ”p”, ou seja, quanto maior o valor de ”p”mais o modelo se torna mais complexo. Desta forma, pode-se concluir que os valores de $p = 20$ e $p = 500$ aparentemente obtiveram a melhor acurácia já que conseguiram separar os conjuntos de uma forma mais eficiente. Por exemplo, o valor de $p = 5$, provocou um *undefitting* já que nao conseguiu separar as duas classes de forma eficiente.



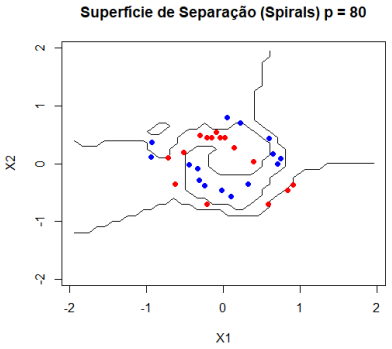
(a) Conjunto Validação p = 5



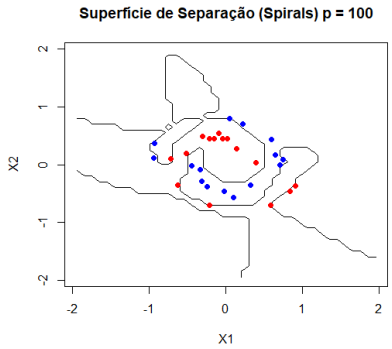
(b) Conjunto Validação p = 20



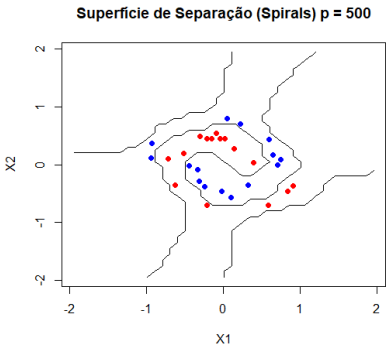
(c) Conjunto Validação p = 50



(d) Conjunto Validação p = 80



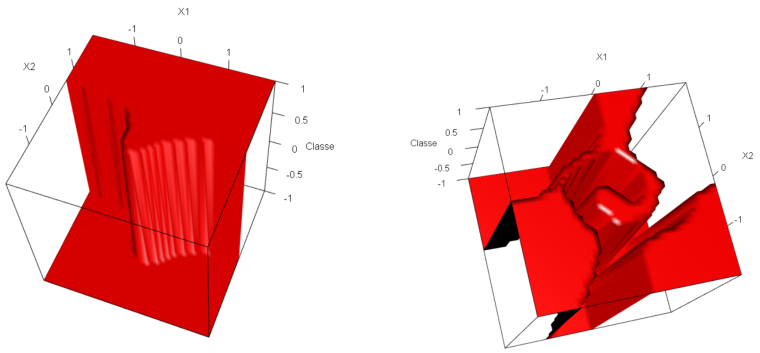
(e) Conjunto Validação p = 100



(f) Conjunto Validação p = 500

Figura 11: Gráficos das Superfícies de Separação do Conjunto de Dados: mlbench.circle

Por fim, ao se observar a Figura 12 a superfície de separação em 3D para p = 5 e p = 500 percebe-se que quanto maior o valor de ”p”maior a complexidade do modelo.



(a) Conjunto Validação p = 5 (b) Conjunto Validação p = 500

Figura 12: Gráficos das Superfícies 3D de Separação do Conjunto de Dados: mlbench.circle

4.4.1 Analise dos Resultados

Analisando-se os resultados obtidos pode-se perceber de uma maneira geral que quanto maiores os valores de ”p”mais complexo o modelo se torna o que aumenta a probabilidade de *overfitting* do modelo, ou seja, o modelo aprende excessivamente os detalhes e ruídos do conjunto de treinamento, resultando em baixo desempenho em dados novos. Mas obsversou-se que quanto maior o grau de ”não-linearidade”do conjunto de dados, para valores maiores de ”p”, o modelo aparentemente apresentou uma maior acurácia. Já para altos valores de ”p”em conjunto de dados ”mais linearizados”aparemente o modelo apresenta uma redução em sua acurácia. De forma geral os efeitos de *underfitting* e *overfitting* podem ser causados pela variação do fator ”p”, mas também dependem do cojunto de dados que esta sendo utilizado para treinamento, já que para conjunto de dados menos complexos, valores de ”p”aparentemente obtiveram uma maior acurácia e já para dados de maior complexidade valores altos de ”p”aumentaram a acurácia do modelo.

5 Conclusão

Pode-se concluir que o objetivo do trabalho foi alcançado e que foi possível observar nos conjuntos de dados utilizados na análise que a variação do ”p”impacta na acurácia do modelo e que este impacto provocará uma aumento ou redução da acurácia do modelo dependendo da complexidade do conjunto de dados.

Referências

[de Pádua Braga, 2024] de Pádua Braga, A. (2024). *Aprendendo com Exemplos: Princípios de Redes Neurais Artificiais e de Reconhecimento de Padrões*. Escola de Engenharia da UFMG, Departamento de Engenharia Eletrônica.