Redes Neurais Artificiais

Lista 2

Vítor Gabriel Reis Caitité - 2016111849

16/12/2020

Questão 1 - Problema Não-Linearmente Separável

Considerando-se o conjunto de dados representado na Figura 1, pede-se que seja implementada, em R ou Python, uma projeção não linear arbitrária que torne o problema linearmente separável. O objetivo é aplicar uma ou mais funções sobre os dados de tal forma que estes dados se tornem separáveis no novo espaço caracterizado por estas funções. Uma função Gaussiana centrada na origem, por exemplo, pode realizar algum nível de linearização destes dados. O objetivo do exercício é entender melhor como a linearização é realizada. Apresentar os mapeamentos, gráficos, etc e discutir a sua solução.

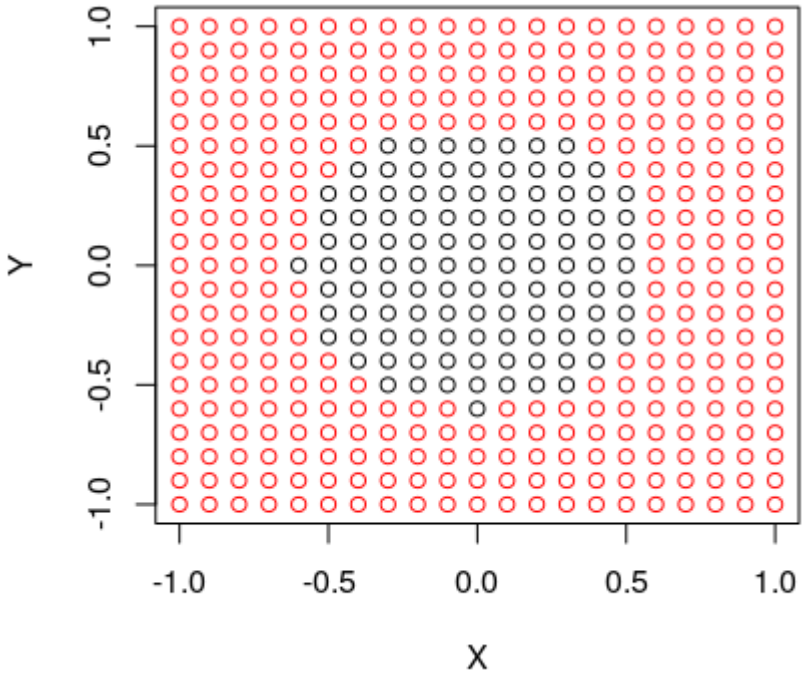
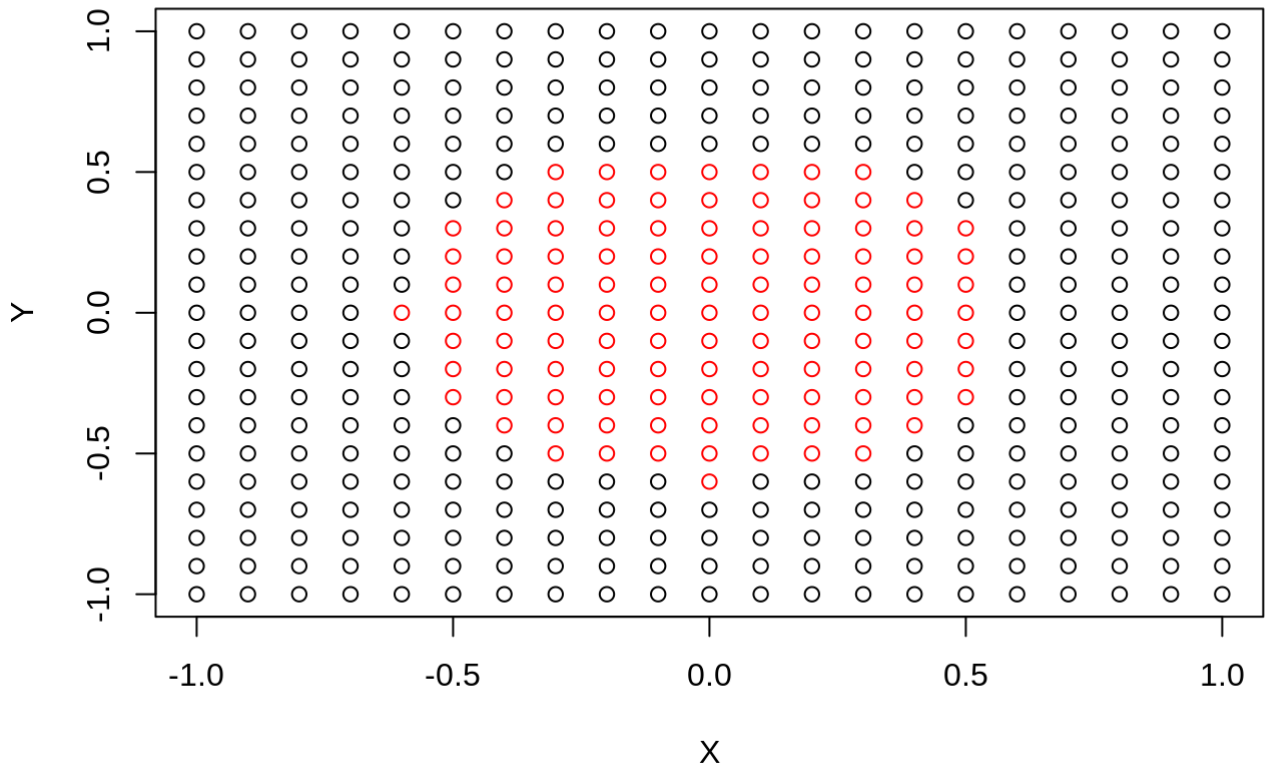


Figura 1: Pontos em vermelho pertencem à classe positiva e pontos em preto pertencem à classe negativa.

Resolução:

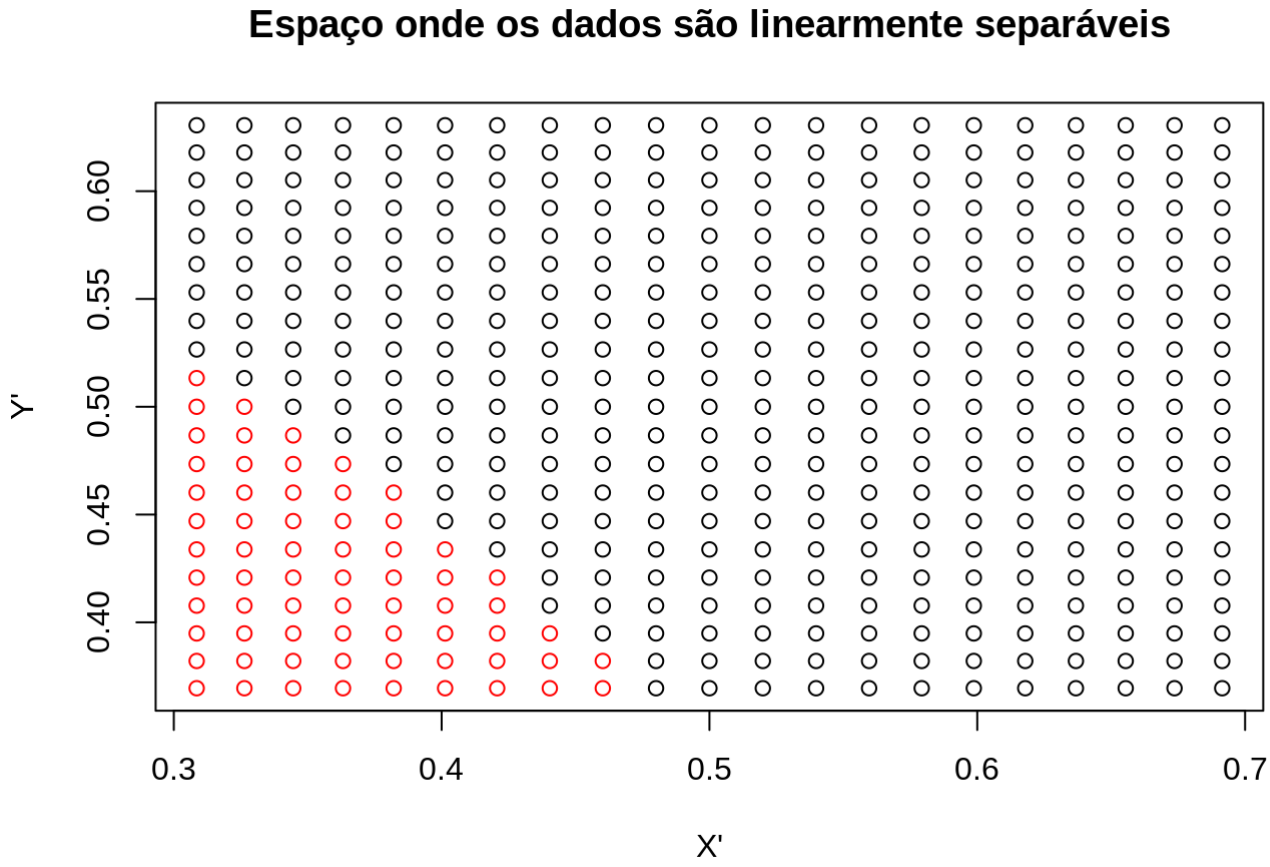
Primeiramente buscou-se plotar a Figura 1 da maneira indicada pela atividade. Para isso foi feito o código abaixo:

```
rm(list = ls())
# Dados originais
x = seq(-1, 1, by = 0.1)
y = seq(-1, 1, by = 0.1)
create_grid <- expand.grid(x,y)

# Função de círculo
circle <- function(x,y) {
  return(sqrt(x^2+y^2))
}

# Plotando eixos
plot(create_grid, type="n", xlab = 'X', ylab = 'Y', main="Dados Originais")
par(new=T)

# Classificando quais dados estão dentro e quais estão fora do círculo de raio 0,6.

raio = 0.6
classe = 1*(circle(create_grid$Var1,create_grid$Var2)>raio)
index=1

# Plotando dados
for (v1 in y) {
  for (v2 in x) {
    if(classe[index] == 1){
      points(v2,v1)
    }else{
      points(v2,v1,col="red")
    }
    index=index+1
  }
}
```



Gerada essa figura, buscou-se tornar o problema linearmente separável. Aplicou-se uma função gaussiana centrada na origem sobre os dados de tal forma que estes dados se tornem separáveis no novo espaço caracterizado por estas funções. Mais detalhadamente o que foi feito foi: com o eixo X foi criado um novo eixo X' = f(X) e o mesmo foi feito para o eixo Y. E então, nesse novo espaço, de X' e Y', o problema se tornou linearmente separável. Isso pode ser visto abaixo:

```
rm(list = ls())

# Dados originais
x = seq(-1, 1, by = 0.1)
y = seq(-1, 1, by = 0.1)

# Aplicação de função gaussiana nos dados
x1<-pnorm(x,mean = 0, sd=2)
y1<-pnorm(y,mean = 0, sd=3)
create_grid <- expand.grid(x1,y1)

# Função de círculo
circle <- function(x,y) {
  return(sqrt(x^2+y^2))
}

# Plotando eixos
plot(create_grid, col="red",type="n", xlab = "X'", ylab = "Y'", main="Espaço onde os
 dados são linearmente separáveis")
par(new=T)

# Classificando quais dados estão dentro e quais estão fora do círculo de raio 0,6.
raio = 0.6
classe = 1*(circle(create_grid$Var1,create_grid$Var2)>raio)
index=1

# Plotando dados
for (v1 in y1) {
  for (v2 in x1) {
    if(classe[index] == 1){
      points(v2,v1,col="black")
    }
    else{
      points(v2,v1,col="red")
    }
    index=index+1
  }
}
```



Questão 2 - Overfitting e Underfitting

Considere a Figura 2, que apresenta os dados de treinamento e funções aproximadoras para o problema de regressão da função seno(x). Observe que os dados foram amostrados desta função com alguma incerteza (ruído). Pede-se :

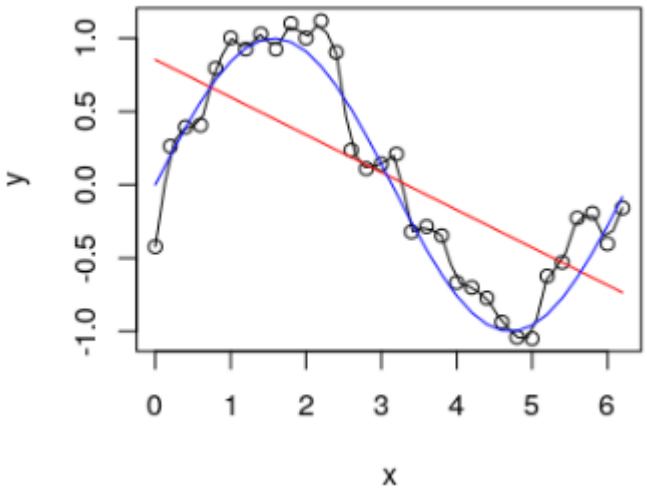


Figura 2: Ajuste de três modelos para um problema de regressão.

[a] Qual dos 3 modelos (preto, vermelho e azul) parece ser o mais adequado como função aproximadora? Discutir.

Resposta: O modelo azul parece ser o mais adequado, sendo o que mais se aproxima da função geradora. O modelo preto representa um caso de *overfitting* e o modelo vermelho reflete um caso de *underfitting*.

[b] Qual dos modelos apresenta menor erro de treinamento? Este modelo é adequado?

Resposta: O modelo preto apresenta menor erro de treinamento, pois ele apresenta menor desvio entre a função aproximada e os dados amostrados. Pode-se ver claramente que este modelo é um caso de *overfitting*, onde o modelo se ajusta muito bem ao conjunto de dados de treinamento, absorvendo também dados de ruído.

[c] Estes modelos possuem complexidades diferentes; se fossem polinômios teriam graus diferentes. Relacione as complexidades destes modelos com os resultados gráficos apresentados.

Resposta: Com base nos resultados gráficos apresentados pode-se inferir que modelo vermelho é o de menor complexidade, seguido pelo azul e por fim o preto, que é o que apresenta maior complexidade. Caso fossem polinômios poderíamos dizer que o vermelho seria o de menor grau e o preto o de maior grau.

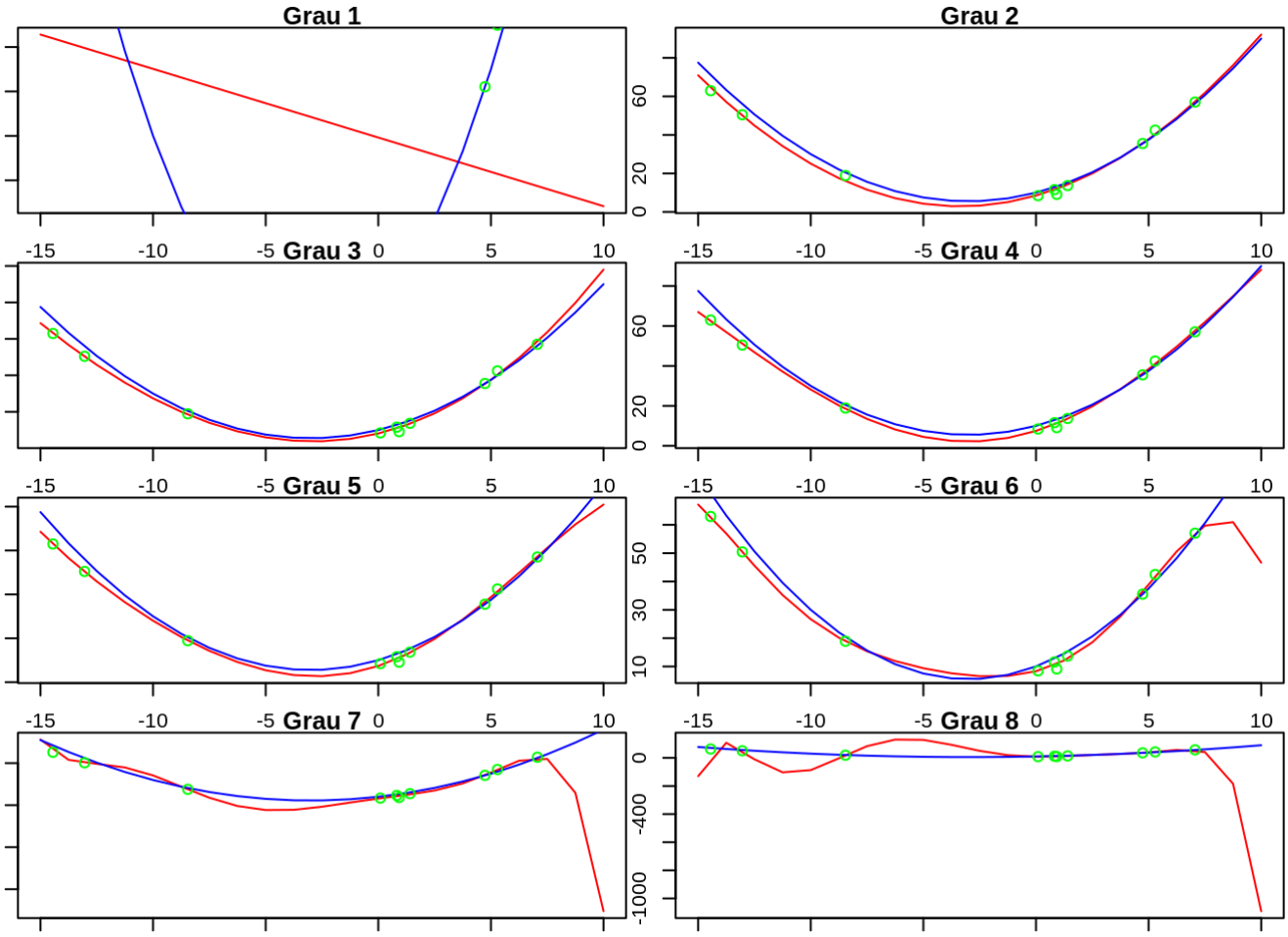
Questão 3 - Aproximação Polinomial

[a] Obtenha aproximações polinomiais a partir de 10 amostras da função geradora $f\_g(x) = (1/2)x^2 + 3x + 10$ somadas com um ruído gaussiano N(mean = 0, sd = 4) amostradas entre $x = -15$ e $x = 10$, com um número de amostras N = 20 e grau do polinônimo variando entre $p = 1$ a $p = 8$. Para cada aproximação, mostre um gráfico com a função geradora, as amostras e o polinômio obtido.

Resolução:

```
rm(list=ls())
library("corpcor") # Pacote para cálculo da pseudoinversa

fgx<-function(x) 0.5*x^2+3*x+10 # Função fg(x)
X<-runif(n = 10, min = -15, max = 10) # Amostra x
Y<-fgx(X)+rnorm(length(X), mean = 0, sd = 4) # 10 amostras da função geradora somadas
 com um ruído gaussiano.

# Gerando as matrizes H para polnômios de grau 1 a 8:
h1<-cbind(X^1, 1)
h2<-cbind(X^2, X^1, 1)
h3<-cbind(X^3, X^2, X^1, 1)
h4<-cbind(X^4, X^3, X^2, X^1, 1)
h5<-cbind(X^5, X^4, X^3, X^2, X^1, 1)
h6<-cbind(X^6, X^5, X^4, X^3, X^2, X^1, 1)
h7<-cbind(X^7, X^6, X^5, X^4, X^3, X^2, X^1, 1)
h8<-cbind(X^8, X^7, X^6, X^5, X^4, X^3, X^2, X^1, 1)

# Calculando a pseudo-inversa:
w1<-pseudoinverse(h1) %*% Y # w = H"*Y.
w2<-pseudoinverse(h2) %*% Y
w3<-pseudoinverse(h3) %*% Y
w4<-pseudoinverse(h4) %*% Y
w5<-pseudoinverse(h5) %*% Y
w6<-pseudoinverse(h6) %*% Y
w7<-pseudoinverse(h7) %*% Y
w8<-pseudoinverse(h8) %*% Y

# Gerando 20 amostras de dados:
x_grid<-seq(-15, 10, ((10-(-15))/20))
y_grid<-(0.5*x_grid^2+3*x_grid+10)

# Gerando as matrizes H para polinômios de grau 1 a 8:
h1_grid<-cbind(x_grid^1, 1)
h2_grid<-cbind(x_grid^2, x_grid^1, 1)
h3_grid<-cbind(x_grid^3, x_grid^2, x_grid^1, 1)
h4_grid<-cbind(x_grid^4, x_grid^3, x_grid^2, x_grid^1, 1)
h5_grid<-cbind(x_grid^5, x_grid^4, x_grid^3, x_grid^2, x_grid^1, 1)
h6_grid<-cbind(x_grid^6, x_grid^5, x_grid^4, x_grid^3, x_grid^2, x_grid^1, 1)
h7_grid<-cbind(x_grid^7, x_grid^6, x_grid^5, x_grid^4, x_grid^3, x_grid^2, x_grid^1,
1)
h8_grid<-cbind(x_grid^8, x_grid^7, x_grid^6, x_grid^5, x_grid^4, x_grid^3, x_grid^2,
 x_grid^1, 1)

# Resultados do modelo:
y1_hat_grid<-h1_grid %*% w1
y2_hat_grid<-h2_grid %*% w2
y3_hat_grid<-h3_grid %*% w3
y4_hat_grid<-h4_grid %*% w4
y5_hat_grid<-h5_grid %*% w5
y6_hat_grid<-h6_grid %*% w6
y7_hat_grid<-h7_grid %*% w7
y8_hat_grid<-h8_grid %*% w8


# Plots
par(mar=c(1,1,1,1))
```

```
par(mfrow=c(4,2))

plot(x_grid, y1_hat_grid, type = "l", col = "red", main = "Grau 1")
lines(x_grid, y_grid, type = "l", col = "blue")
points(X, Y,type = "p", col = "green")

plot(x_grid, y2_hat_grid, type = "l", col = "red", main = "Grau 2")
lines(x_grid, y_grid, type = "l", col = "blue")
points(X, Y,type = "p", col = "green")

plot(x_grid, y3_hat_grid, type = "l", col = "red", main = "Grau 3")
lines(x_grid, y_grid, type = "l", col = "blue")
points(X, Y,type = "p", col = "green")

plot(x_grid, y4_hat_grid, type = "l", col = "red", main = "Grau 4")
lines(x_grid, y_grid, type = "l", col = "blue")
points(X, Y,type = "p", col = "green")

plot(x_grid, y5_hat_grid, type = "l", col = "red", main = "Grau 5")
lines(x_grid, y_grid, type = "l", col = "blue")
points(X, Y,type = "p", col = "green")

plot(x_grid, y6_hat_grid, type = "l", col = "red", main = "Grau 6")
lines(x_grid, y_grid, type = "l", col = "blue")
points(X, Y,type = "p", col = "green")

plot(x_grid, y7_hat_grid, type = "l", col = "red", main = "Grau 7")
lines(x_grid, y_grid, type = "l", col = "blue")
points(X, Y,type = "p", col = "green")

plot(x_grid, y8_hat_grid, type = "l", col = "red", main = "Grau 8")
lines(x_grid, y_grid, type = "l", col = "blue")
points(X, Y,type = "p", col = "green")
```



[b] Responda: Ocorreu Overfitting? Ocorreu Underfitting? Em quais situações?

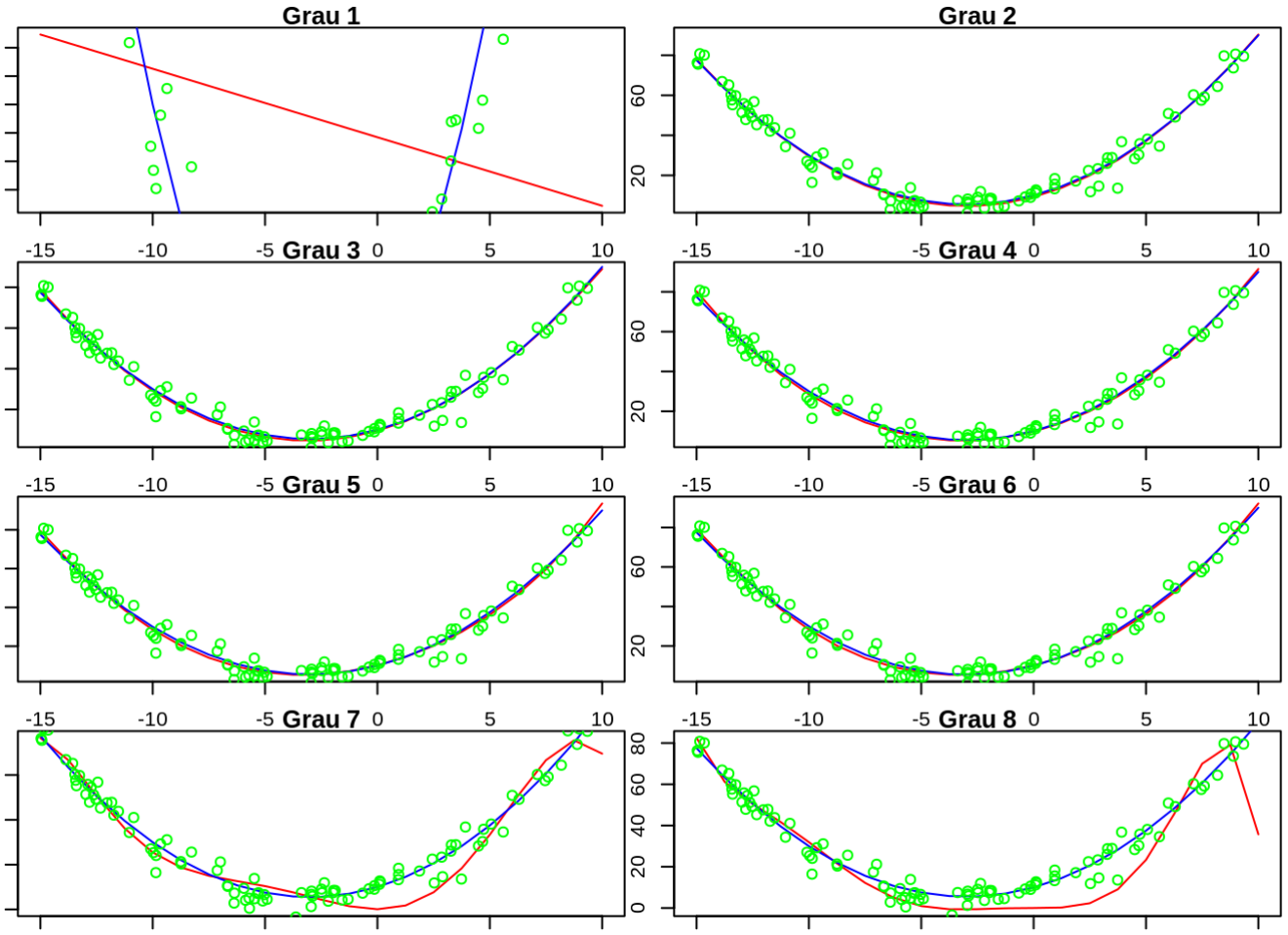
Resposta: No polinômino de grau 1 ocorreu *underfitting* e em polinômios de grau maior ou igual 5 ocorreram *overfitting*.

[c] Repita o procedimento para 100 amostras ao invés de 10. Qual o impacto do número de amostras na aproximação?

Resolução:

```
rm(list=ls())
library("corpcor") # Pacote para cálculo da pseudoinversa

fgx<-function(x) 0.5*x^2+3*x+10 # Função fg(x)
X<-runif(n = 100, min = -15, max = 10) # Amostra x
Y<-fgx(X)+rnorm(length(X), mean = 0, sd = 4) # 10 amostras da função geradora somadas
 com um ruído gaussiano.

# Gerando as matrizes H para polnômios de grau 1 a 8:
h1<-cbind(X^1, 1)
h2<-cbind(X^2, X^1, 1)
h3<-cbind(X^3, X^2, X^1, 1)
h4<-cbind(X^4, X^3, X^2, X^1, 1)
h5<-cbind(X^5, X^4, X^3, X^2, X^1, 1)
h6<-cbind(X^6, X^5, X^4, X^3, X^2, X^1, 1)
h7<-cbind(X^7, X^6, X^5, X^4, X^3, X^2, X^1, 1)
h8<-cbind(X^8, X^7, X^6, X^5, X^4, X^3, X^2, X^1, 1)

# Calculando a pseudo-inversa:
w1<-pseudoinverse(h1) %*% Y # w = H"*Y.
w2<-pseudoinverse(h2) %*% Y
w3<-pseudoinverse(h3) %*% Y
w4<-pseudoinverse(h4) %*% Y
w5<-pseudoinverse(h5) %*% Y
w6<-pseudoinverse(h6) %*% Y
w7<-pseudoinverse(h7) %*% Y
w8<-pseudoinverse(h8) %*% Y

# Gerando 20 amostras de dados:
x_grid<-seq(-15, 10, ((10-(-15))/20))
y_grid<-(0.5*x_grid^2+3*x_grid+10)

# Gerando as matrizes H para polinômios de grau 1 a 8:
h1_grid<-cbind(x_grid^1, 1)
h2_grid<-cbind(x_grid^2, x_grid^1, 1)
h3_grid<-cbind(x_grid^3, x_grid^2, x_grid^1, 1)
h4_grid<-cbind(x_grid^4, x_grid^3, x_grid^2, x_grid^1, 1)
h5_grid<-cbind(x_grid^5, x_grid^4, x_grid^3, x_grid^2, x_grid^1, 1)
h6_grid<-cbind(x_grid^6, x_grid^5, x_grid^4, x_grid^3, x_grid^2, x_grid^1, 1)
h7_grid<-cbind(x_grid^7, x_grid^6, x_grid^5, x_grid^4, x_grid^3, x_grid^2, x_grid^1,
1)
h8_grid<-cbind(x_grid^8, x_grid^7, x_grid^6, x_grid^5, x_grid^4, x_grid^3, x_grid^2,
 x_grid^1, 1)

# Resultados do modelo:
y1_hat_grid<-h1_grid %*% w1
y2_hat_grid<-h2_grid %*% w2
y3_hat_grid<-h3_grid %*% w3
y4_hat_grid<-h4_grid %*% w4
y5_hat_grid<-h5_grid %*% w5
y6_hat_grid<-h6_grid %*% w6
y7_hat_grid<-h7_grid %*% w7
y8_hat_grid<-h8_grid %*% w8


# Plots
par(mar=c(1,1,1,1))
```

```
par(mfrow=c(4,2))

plot(x_grid, y1_hat_grid, type = "l", col = "red", main = "Grau 1")
lines(x_grid, y_grid, type = "l", col = "blue")
points(X, Y,type = "p", col = "green")

plot(x_grid, y2_hat_grid, type = "l", col = "red", main = "Grau 2")
lines(x_grid, y_grid, type = "l", col = "blue")
points(X, Y,type = "p", col = "green")

plot(x_grid, y3_hat_grid, type = "l", col = "red", main = "Grau 3")
lines(x_grid, y_grid, type = "l", col = "blue")
points(X, Y,type = "p", col = "green")

plot(x_grid, y4_hat_grid, type = "l", col = "red", main = "Grau 4")
lines(x_grid, y_grid, type = "l", col = "blue")
points(X, Y,type = "p", col = "green")

plot(x_grid, y5_hat_grid, type = "l", col = "red", main = "Grau 5")
lines(x_grid, y_grid, type = "l", col = "blue")
points(X, Y,type = "p", col = "green")

plot(x_grid, y6_hat_grid, type = "l", col = "red", main = "Grau 6")
lines(x_grid, y_grid, type = "l", col = "blue")
points(X, Y,type = "p", col = "green")

plot(x_grid, y7_hat_grid, type = "l", col = "red", main = "Grau 7")
lines(x_grid, y_grid, type = "l", col = "blue")
points(X, Y,type = "p", col = "green")

plot(x_grid, y8_hat_grid, type = "l", col = "red", main = "Grau 8")
lines(x_grid, y_grid, type = "l", col = "blue")
points(X, Y,type = "p", col = "green")
```



Nesse caso ocorreu *underfitting* apenas no polinômio de grau 1, e *overfitting* para polinômios de grau maior ou igual a 7. Notou-se que com o número menor de amostras ocorreu as vezes de não se ter amostras em uma certa região e consequentemente o modelo não ficar bem especificado naquela região. Esse problema não ocorreu mais após o número de amostras ter aumentado 100. Notou-se também que com um número maior de amostras é mais difícil a função de aproximação se afastar muito da função geradora mesmo ao ocorrer *overfitting*.