



Instituto Tecnológico de Aeronáutica

Introdução às Redes Neurais Artificiais e aos Grandes Modelos de Linguagem

Prof. Mauri Aparecido de Oliveira

CURSO – Extensivo

PO-249



Modelos de Regressão e Redes Neurais

ai



Antes de iniciar a abordagem de Redes Neurais Artificiais para realizar tarefas de Regressão e Classificação, será apresentada uma revisão sobre modelos de regressão linear simples e múltipla.

A teoria estatística para construção de Modelos de Regressão, tem início em 1885 com Sir Francis Galton, o qual definiu inicialmente o termo “regressão” na análise estatística. Veja os trabalhos: (i) *Family Likeness in Stature*, dos autores Francis Galton e J. D. Hamilton Dickson no *Proceedings of the Royal Society of London Vol. 40 (1886), pp. 42-73 (32 pages)* Publicado pela: *Royal Society*; e (ii) *Regression towards Mediocrity in Hereditary Stature*, publicado no *Journal of the Anthropological Institute of Great Britain and Ireland, (Vol. 15, pp. 246-263)*.

Vale destacar que modelos estatísticos são construídos baseados em premissas e uma vez que o modelo foi determinado, são necessários diversos testes para verificação da adequabilidade dos parâmetros calculados. Modelos estatísticos são construídos com base em uma teoria matemática muito bem estabelecida e, conforme Salsburg (2009) “*só podem ser totalmente compreendidos em termos de fórmulas e símbolos matemáticos*”. Os resultados estatísticos dependem de verificações de hipóteses subjacentes nas diversas estimações que são realizadas, e esse processo de “auditoria” torna muito explícitos todos os passos dos procedimentos adotados, não possibilitando modelos do tipo black-box (caixa-preta), comuns em machine learning. Pelo contrário, teoria estatística tem sido desenvolvida e usada para entender o comportamento de algoritmos black-box de machine learning.



3.1 Modelo Estatístico de Regressão Linear

A análise de regressão é um método para estimar as relações entre variáveis (Montgomery *et al.*, 2012). Permite entender como uma variável dependente muda quando qualquer uma das variáveis independentes é alterada (Freedman, 2009). Este processo estima o valor médio da variável dependente quando as variáveis independentes são mantidas constantes. O objetivo é a estimação de uma função das variáveis independentes, conhecida como função de regressão (Hastie, Tibshirani e Friedman, 2009). De forma geral, a análise de regressão é uma técnica estatística para investigar e modelar a relação entre variáveis. Em certas situações, a análise de regressão pode inferir relações causais entre variáveis independentes e dependentes. Por exemplo, Angrist e Pischke (2008) discutem o uso da análise de regressão em econometria para inferir efeitos causais. No entanto, é importante observar que a correlação não implica necessariamente causalidade, portanto, é necessário cautela (Pearl, 2009).

3.1.1 Regressão Linear Simples

De acordo com Neter *et al.* (1983) a Análise de Regressão serve a três propósitos principais: (1) descrição, (2) controle e (3) predição. Dessa forma, vamos considerar a construção de um modelo de predição que consiste em explicar uma variável utilizando uma ou mais outras variáveis. Para iniciar a apresentação dessa modelagem que envolve o desenvolvimento de um modelo linear, utilizaremos valores observados de uma variável X (denominada de variável independente ou explicativa) que serão usados para prever uma variável Y (chamada de variável dependente, resposta ou explicada). A variável que será explicada, Y , constitui-se dos valores observados que correspondem aos respectivos valores de X . Valores conhecidos de X e Y são descritos de forma genérica como x e y , respectivamente. Na prática, nunca existirá uma relação matemática exata (por exemplo, linear) entre duas variáveis, de tal forma que temos um erro associado a cada valor gerado pelo modelo, ou seja

$$y = \beta_0 + \beta_1 x + \varepsilon, \quad (1)$$

descreve o **Modelo de Regressão Linear Simples (MRLS)**. Note que aplicando o operador esperança a (1), obtemos (2). A especificação do modelo precisa da estimativa de β_0 e β_1 , para isso necessitamos dos estimadores $\hat{\beta}_0$ e $\hat{\beta}_1$. Em termos de concepção do modelo, sua descrição geral é dada por

$$E[y|x] = \beta_0 + \beta_1 x, \quad (2)$$

que é denominada de **Função de Regressão Populacional (FRP)**. Além disso, consideramos que

$$\text{Var}[y|x] = \sigma^2. \quad (3)$$

De maneira geral, temos que (2) descreve que a média condicional da variável dependente Y é linear com relação a X , sendo β_0 o intercepto e β_1 a inclinação da reta que será ajustada aos dados observados. A função de regressão em (1) é linear no parâmetro β_1 , se β_1 aparece somente tendo expoente 1 e não estiver multiplicado ou dividido por nenhum outro parâmetro (por exemplo, $\beta_1\beta_2$, $\frac{\beta_1}{\beta_2}$, etc.). Como estamos iniciando a construção de

modelos de regressão, os resultados gerados serão obtidos a partir de um modelo que é tanto linear nos parâmetros, como na variável, tal como mostrado em (2). Em (3) é considerado que a variância de Y dado X é constante. De acordo com Gujarati (2011): “geometricamente, a curva de regressão populacional é simplesmente o lugar geométrico das médias condicionais da variável dependente para os valores fixos da(s) variável(is) explicativa(s). Mais simplesmente, é a curva que conecta as médias das subpopulações de Y correspondentes aos valores dados do regressor X ”. Graficamente, a Figura-1 mostra essa situação onde $\beta_0 > 0$ e $\beta_1 > 0$.

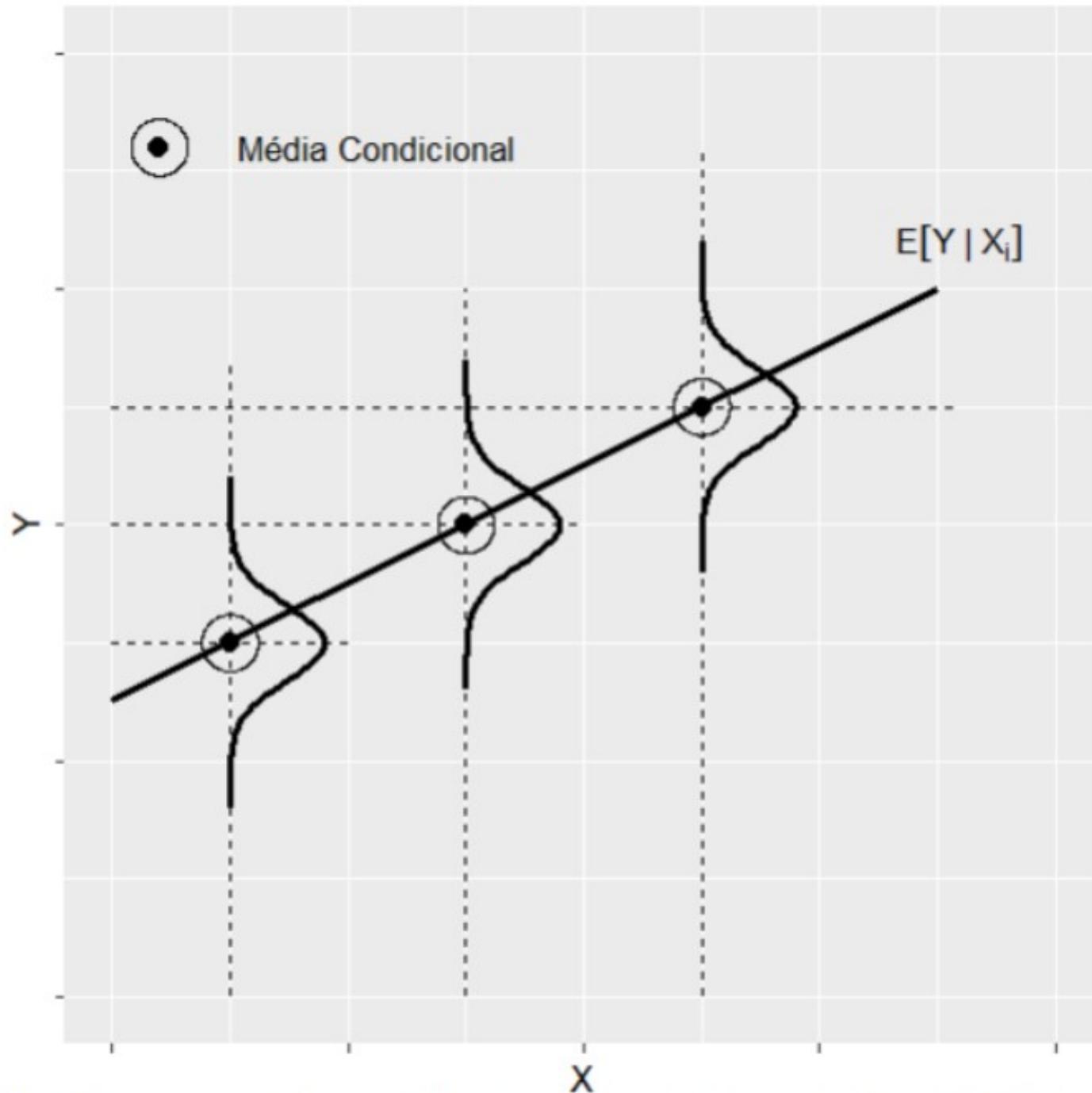


Figura-1 Modelo de Regressão Linear Simples. Fonte: Adaptado de Wooldridge (2000, pg. 26).



Como não temos acesso a todos os valores da população, para estimar os parâmetros β_0 e β_1 , precisamos de uma amostra obtida da população. Considerando que $\{(x_i, y_i); i = 1, 2, \dots, n\}$ denota uma amostra aleatória de tamanho n da população, temos

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i. \quad (4)$$

Das equações (1), (2) e (4) podemos também escrever que $\varepsilon_i = y_i - E[y_i]$. Ou seja, o que estamos fazendo é ajustando um modelo para explicar y , a partir dos valores coletados de x , tal que $y|x \sim N(\beta_0 + \beta_1 x; \sigma^2)$, atente como é apresentada a distribuição de cada valor de y na Figura-1. Isso implica que o modelo gerado está restrito a amplitude dos valores da variável independente (ou **preditora**) que foram coletados e inseridos no nosso conjunto de dados amostrais. Essa limitação, do modelo predizer valores de y considerando o intervalo disponível de x , em muitas áreas é uma justificativa para fazer-se uma distinção entre os conceitos de **predizer** e **prever**. Aqui, vamos utilizar o MRLS para a tarefa de **predição** e o termo **previsão** será reservado para casos em que um modelo retorna um valor a frente no tempo de uma variável de interesse (Veja o Quadro Conceitual-1). Dessa forma, na equação (4) temos que ε_i , o termo de **erro (resíduo ou desvio)** para a observação i , é uma variável aleatória não observável que assume valores positivos ou negativos.

Quadro Conceitual - 1

É comum, usarem de forma intercambiável as palavras previsão e predição. Porém, dependendo da área, temos conceitos distintos associados a cada uma delas. Por exemplo, em Silver (*The Signal and the Noise: Why So Many Predictions Fail—but Some Don't*, 2012) encontramos que:

“A ideia do homem como mestre de seu destino estava ganhando força. As palavras predizer e prever são amplamente usadas indistintamente hoje em dia, mas na época de Shakespeare, elas significavam coisas diferentes. Uma predição (prediction) era o que o adivinho dizia; uma previsão (forecast) era algo mais parecido com a ideia de Cassius. O termo previsão (forecast) veio das raízes germânicas do inglês, diferente de predizer (predict), que é do latim. A previsão refletia a nova mundanidade protestante em vez da sobrenaturalidade do Sacro Império Romano. Fazer uma previsão tipicamente implicava planejar sob condições de incerteza. Sugeria ter prudência, sabedoria e diligência, mais como a maneira como agora usamos a palavra previsão.”

Também escrevendo que:

“Uma **predição** é uma declaração definitiva e específica sobre quando e onde ocorrerá um terremoto” e “Enquanto uma **previsão** é uma afirmação probabilística, geralmente em uma escala de tempo mais longa: há 60 por cento de chance de um terremoto”.

Considere a Figura-2 em que temos as representações de uma variável y em termos de uma outra variável x e também sua evolução no tempo t . Para muitos autores, a predição consiste em ajustar um modelo para explicar y , porém somente no intervalo em que x está definida. Na previsão, queremos saber os valores futuros de y , tomando como base uma série histórica de seus valores.

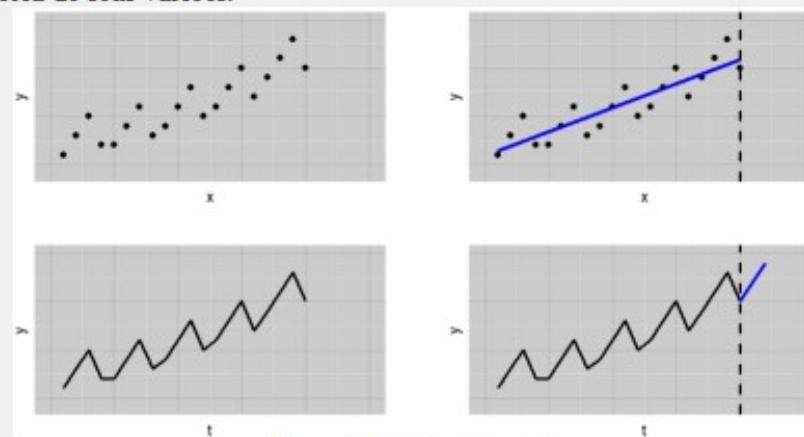


Figura-2 Predição e Previsão.

Esses dois tipos de modelos geram intervalos de predição e previsão. No caso das predições temos intervalos hiperbólicos, e no caso das previsões temos os intervalos no formato de “cones” de previsão. Portanto, a construção de modelos preditivos está associada com a precisão de estimativas considerando os intervalos dos dados observados, enquanto a



Sobre os erros, consideramos duas suposições importantes: (i) assumimos que $E[\varepsilon_i | x_i] = 0$ e (ii) que estes erros são independentes, ou seja, o valor de erro inerente a uma observação não fornece nenhuma informação sobre o erro associado a outra observação.

O MRLS estará plenamente especificado quando os coeficientes β_0 e β_1 estiverem determinados e as premissas do MRLS forem atendidas. As premissas do modelo de regressão são as seguintes (Gujarati, 2011):

1. O modelo de regressão é linear nos parâmetros.
2. Os valores dos regressores, os X , são fixados em amostras repetidas.
3. Para dados X , o valor médio do termo de erro ε_i é zero.
4. Para dados X , a variância de ε_i é constante e homocedástica.
5. Para dados X , não há autocorrelação entre os termos de erro.
6. Se os X forem estocásticos, o termo de erro e os X (estocásticos) são independentes, ou pelo menos não correlacionados.
7. O número de observações deve ser maior que o número de regressores.
8. Deve haver suficiente variabilidade nos valores assumidos pelos regressores.
9. O modelo de regressão está especificado da forma correta.
10. Não há relação linear exata (ou seja, multicolinearidade) entre os regressores.
11. O termo estocástico (de erro) ε_i se distribui normalmente.

PO-249



Exemplo: Modelo de Regressão Linear Simples

ai

EXEMPLO Modelo de Regressão Linear Simples

Todos os comandos do R utilizados na construção desse Exemplo estão disponíveis na seção de Apêndices. Nesses comandos, são implementadas as expressões matemáticas do MRLS e os resultados são comparados com os que são gerados pelos comandos/funções disponíveis nos pacotes no R.





De acordo com Bodie *et al.* (2015), o Modelo de Precificação de Ativos de Capital (em inglês, Capital Asset Pricing Model - CAPM), constitui-se em uma ferramenta de análise central na economia financeira moderna. Em 1952, Markowitz apresenta as bases da teoria de gestão moderna de carteiras. O CAPM é desenvolvido e divulgado em uma sequência de trabalhos de Sharpe (1964), Lintner (1965) e Mossin (1966), permitindo avaliar a relação entre o risco e o retorno de um ativo financeiro e seu retorno esperado. No CAPM, a medida usada para medir o desempenho de um ativo financeiro é o **beta**. Quando consultamos o trabalho do Sharpe (*Capital Asset Prices: A Theory of Market Equilibrium under Conditions of Risk. Journal of Finance*, 19, 425-442) extraímos o seguinte trecho: “*Mas parte da dispersão é devido a uma relação subjacente com o retorno da combinação g, mostrada por B_{ig} , a inclinação da linha de regressão*”. Como o Sharpe usou uma máquina de escrever para escrever (datilografar) o texto, o que ele realmente quis dizer com B era a letra grega β ou beta.

No glossário do Banco Central do Brasil (<https://www.bcb.gov.br/meubc/glossario>) temos o beta sendo descrito como a: “*estimativa do nível de oscilação que se deve esperar de um fundo (ou ativo qualquer) como resposta a variações do mercado de ações. Uma ação com beta igual a 2, por exemplo, indica que quando o Ibovespa sobe 1%, a ação tende a subir 2%. Quanto maior o beta, maior o risco do papel.*”

O beta é uma medida do risco de mercado ou risco sistemático e é utilizado por investidores para entender o risco de mercado inerente a uma ação. O beta também pode ajudar a criar um portfólio diversificado combinando algumas ações com betas superiores a 1 e algumas ações betas inferiores a 1.

Como o beta de um ativo (ação, título) mede sua volatilidade em relação ao mercado geral (ou uma carteira mais ampla), é natural considerar que envolva medidas de variabilidade. Dessa forma, o beta é escrito como

$$\beta_i = \frac{\text{Cov}[R_i, R_M]}{\text{Var}[R_M]}, \quad (1)$$

onde,

R_i = Retorno do ativo i .

R_M = Retorno da proxy do mercado.



Ou seja, o coeficiente beta de um ativo i é determinado pela razão entre a covariância dos retornos do ativo i e os retornos de uma proxy do mercado, e a variância dos retornos da carteira de mercado. Entenda por proxy a variável que representará o comportamento do mercado, que é formado por um conjunto de ativos. Esta razão apresentada na expressão (EMBR1) corresponde exatamente a mesma para o cálculo do coeficiente angular da reta de regressão. Sendo assim, vamos construir um MRLS em que um dos resultados de interesse será o beta do ativo escolhido.

O ativo escolhido será o EMBR3. A Figura-EX12.6 apresenta o histórico do beta desse ativo no período de 1998 a 2024. Nessa figura a linha tracejada corresponde ao valor médio dos betas no período analisado, sendo de aproximadamente 0,53. E a linha contínua corresponde ao beta igual a 1.

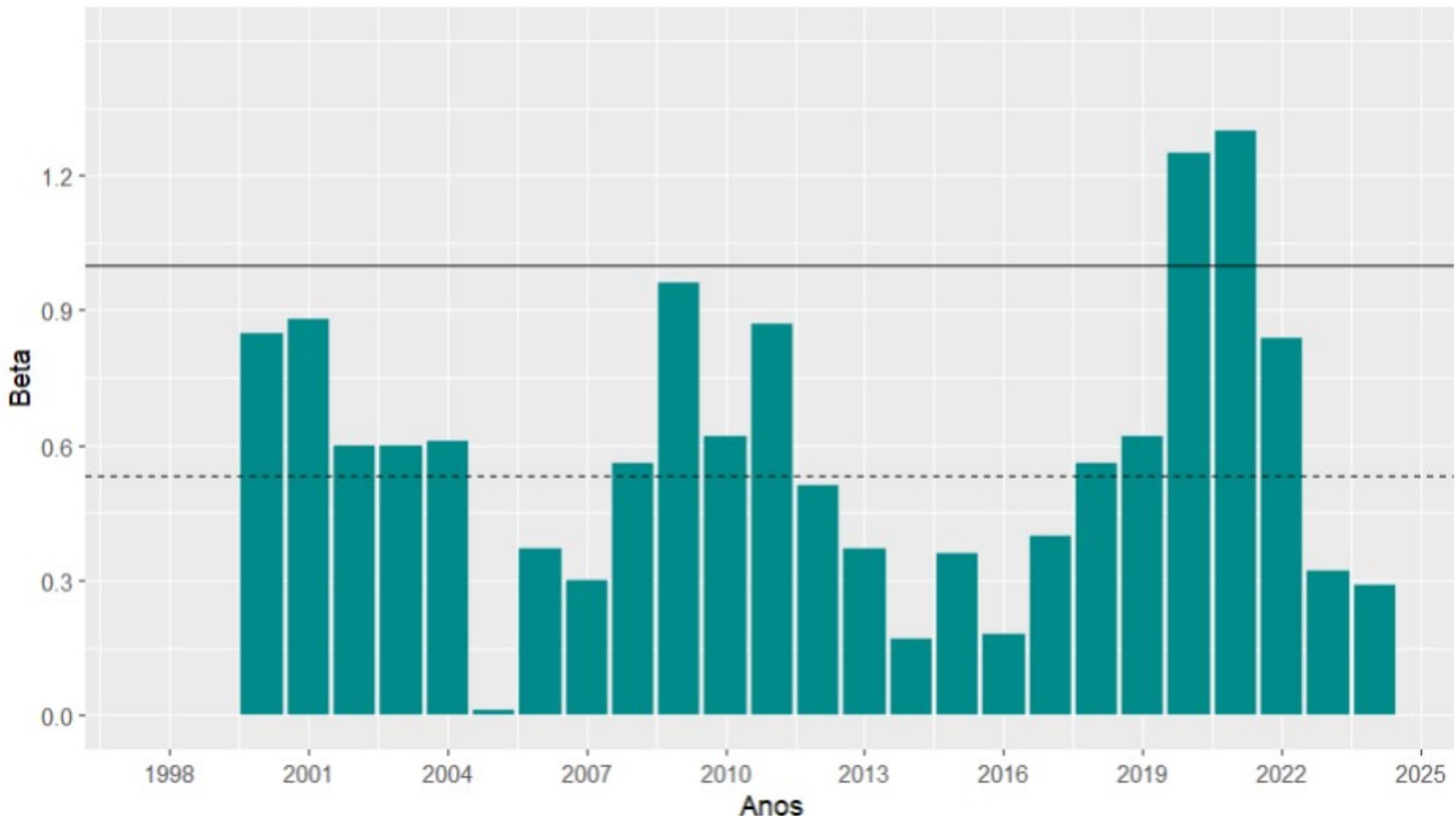


Figura-1 Evolução do beta do ativo EMBR3 no período 1998-2024.



Vamos construir o modelo de regressão para o ativo EMB3, em que a variável dependente é o retorno do ativo (R_A) e a variável independente é a taxa de retorno da carteira de mercado (R_M). Para representar o mercado, é comum utilizar um índice de bolsa como proxy para o índice de mercado, nesse caso utilizaremos o IBOVESPA.

$$R_A = \alpha + \beta R_M + \varepsilon. \quad (2)$$

O beta representa a influência do risco sistemático sobre a taxa de retorno da empresa. Vamos determinar o beta da empresa EMBR3 para o ano de 2021. As etapas da análise de regressão são apresentadas as seguir.

Obtenção dos dados

Os dados da empresa EMBR3 podem ser obtidos, por exemplo, do yahoo finance, utilizando o comando `getSymbols` da library `quantmod`.

```
EMBR <- getSymbols("EMBR3.SA", src = "yahoo", from = "2021-01-01", to =
"2021-12-31", auto.assign = FALSE)
```

Entre os dados obtidos temos os preços de fechamento para cada dia de 2021.

Da mesma forma carregamos os dados do IBOVESPA.

```
IBOV <- getSymbols("^BVSP", src = "yahoo", from = "2021-01-01", to =
"2021-12-31", auto.assign = FALSE)
```

Criação do vetores de retornos do ativo e do mercado

A partir dos preços de fechamento e dos valores do IBOVESPA para dia de negociação em 2021, calculas o retorno do ativo e o retorno de mercado (taxa de variação do IBOVESPA). Lembrando que o retorno em um instante t é calculado usando os valores de preço dos instantes t e $t - 1$, conforme mostrado a seguir.

$$R_t = \frac{P_t - P_{t-1}}{P_{t-1}}. \quad (3)$$



Scatter plot dos retornos

A seguir é apresentado o scatter plot dos retornos. Esse gráfico possibilita verificar o tipo de relacionamento linear (ou não) entre as variáveis.

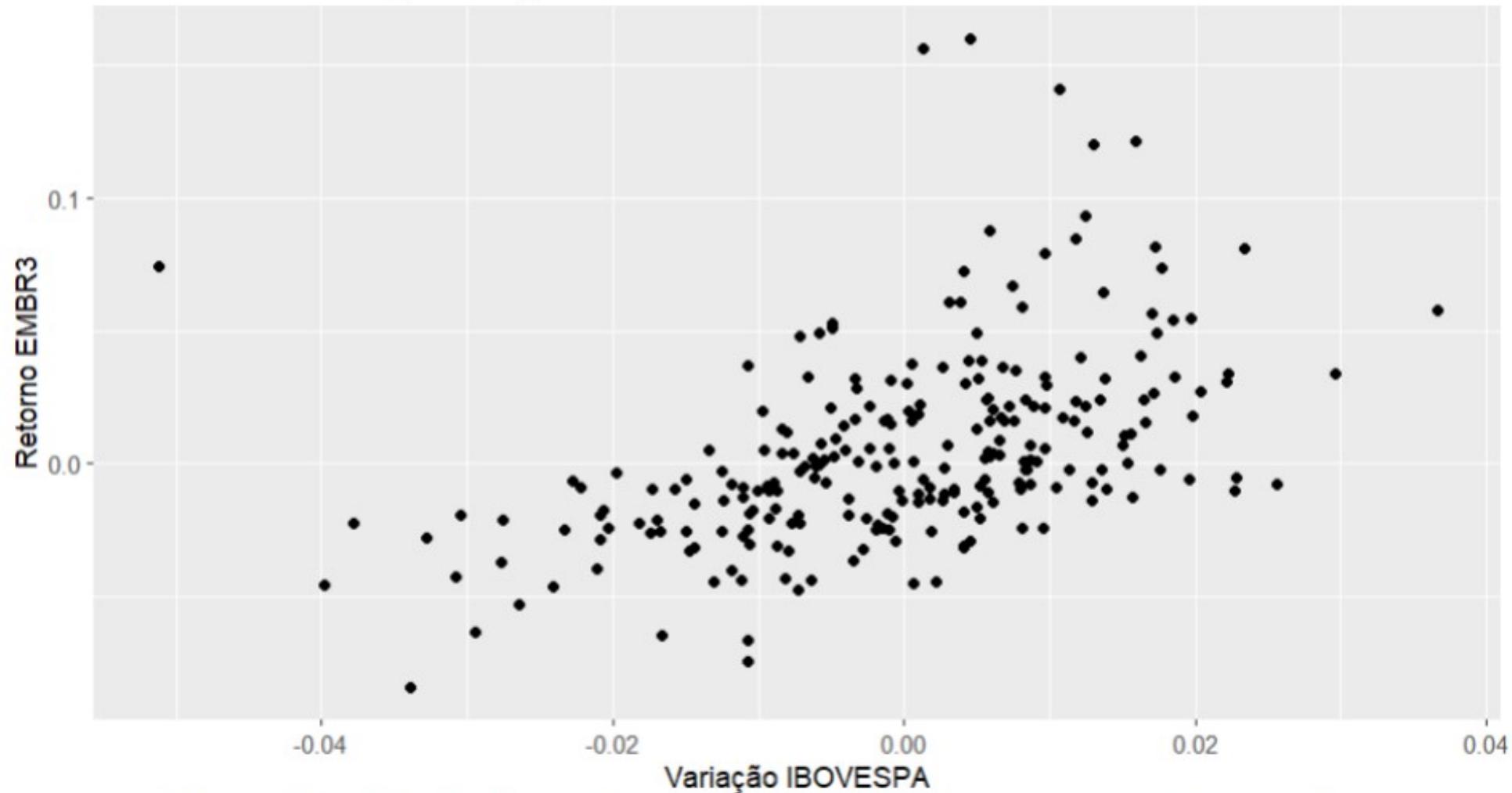


Figura-2 Gráfico de dispersão entre o retorno do ativo e o retorno de mercado.

Estimação do modelo de regressão

Na figura a seguir podemos observar a reta de regressão ajustada aos dados e o coeficiente angular sendo exatamente igual ao beta da empresa EMBR3 que aparece nos históricos financeiros. Essa reta de regressão foi obtida utilizando-se o comando `stat_smooth(method = "lm", formula = y ~ x, geom = "smooth")`. Esse comando já gera um intervalo de confiança para $E[y]$.

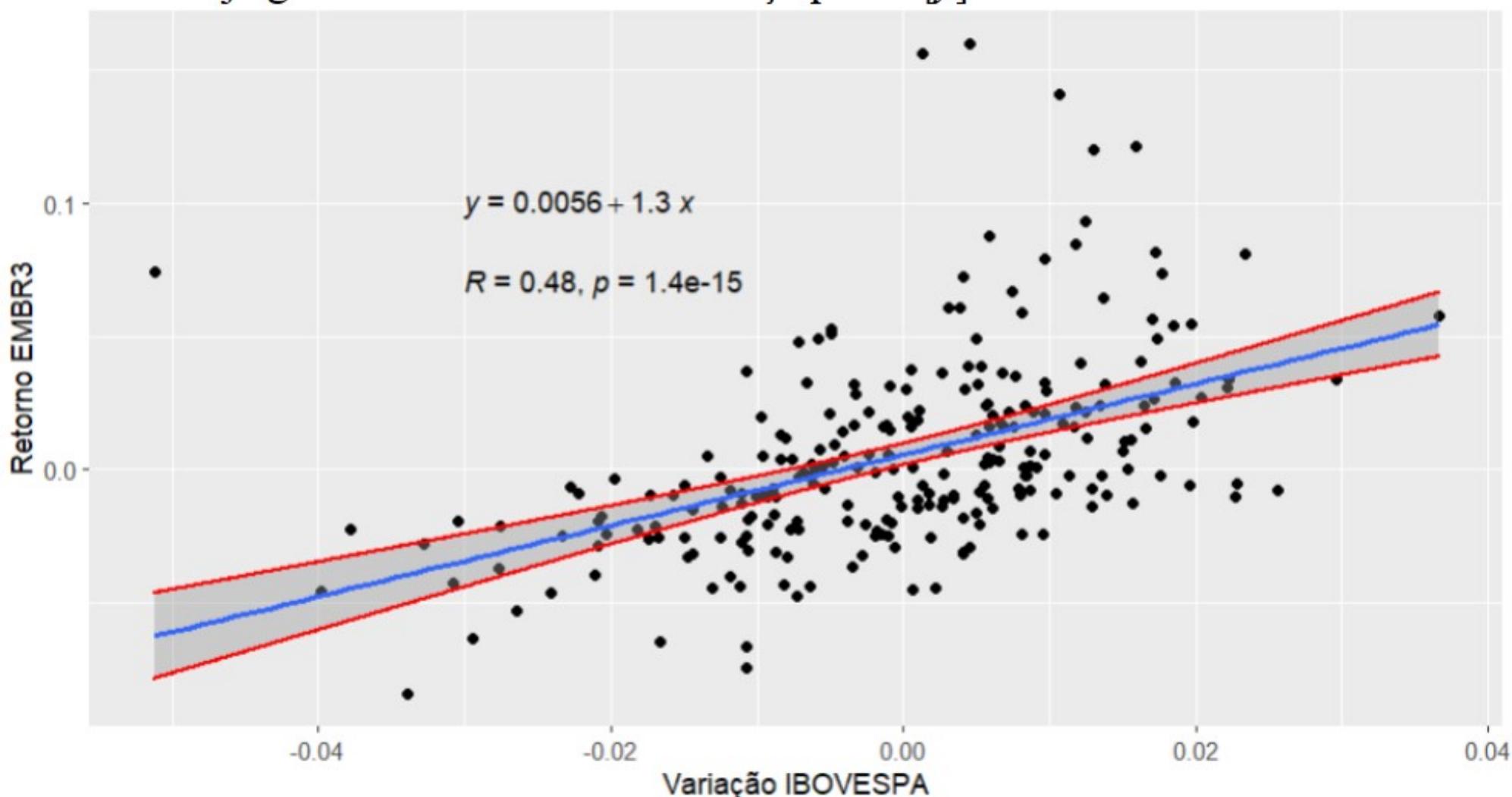


Figura-3 Reta de regressão ajustada para R_A e R_M com intervalo de confiança para $E[R_A]$.



O sumário da regressão é apresentado a seguir. Note que o p-valor associado ao coeficiente angular da reta de regressão é bastante pequeno, permitindo rejeitar a hipótese de que seja o beta estatisticamente igual a zero.

call:

```
lm(formula = EMBR3.SA.Close ~ BVSP.Close, data = IBOV_EMBR)
```

Residuals:

Min	1Q	Median	3Q	Max
-0.06581	-0.01959	-0.00315	0.01253	0.14871

Coefficients:

	Estimate	std. Error	t value	Pr(> t)
(Intercept)	0.005626	0.002053	2.741	0.00658 **
BVSP.Close	1.331904	0.155762	8.551	1.36e-15 ***

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.03218 on 244 degrees of freedom

Multiple R-squared: 0.2306, Adjusted R-squared: 0.2274

F-statistic: 73.12 on 1 and 244 DF, p-value: 1.359e-15

Intervalo de Predição para $Y_{p(novo)}$

Na próxima figura é inserido o intervalo de predição. Observe que o intervalo de predição é mais amplo que o intervalo para $E[y]$, como pode ser observado das expressões matemáticas para esses dois intervalos. Essa maior amplitude do IP é devido à incerteza adicional de novos dados observados que introduz um erro de predição e quanto mais longe o ponto de predição estiver dos dados utilizados para ajustar o modelo, maior será a incerteza e, consequentemente, mais amplo será o intervalo de predição.

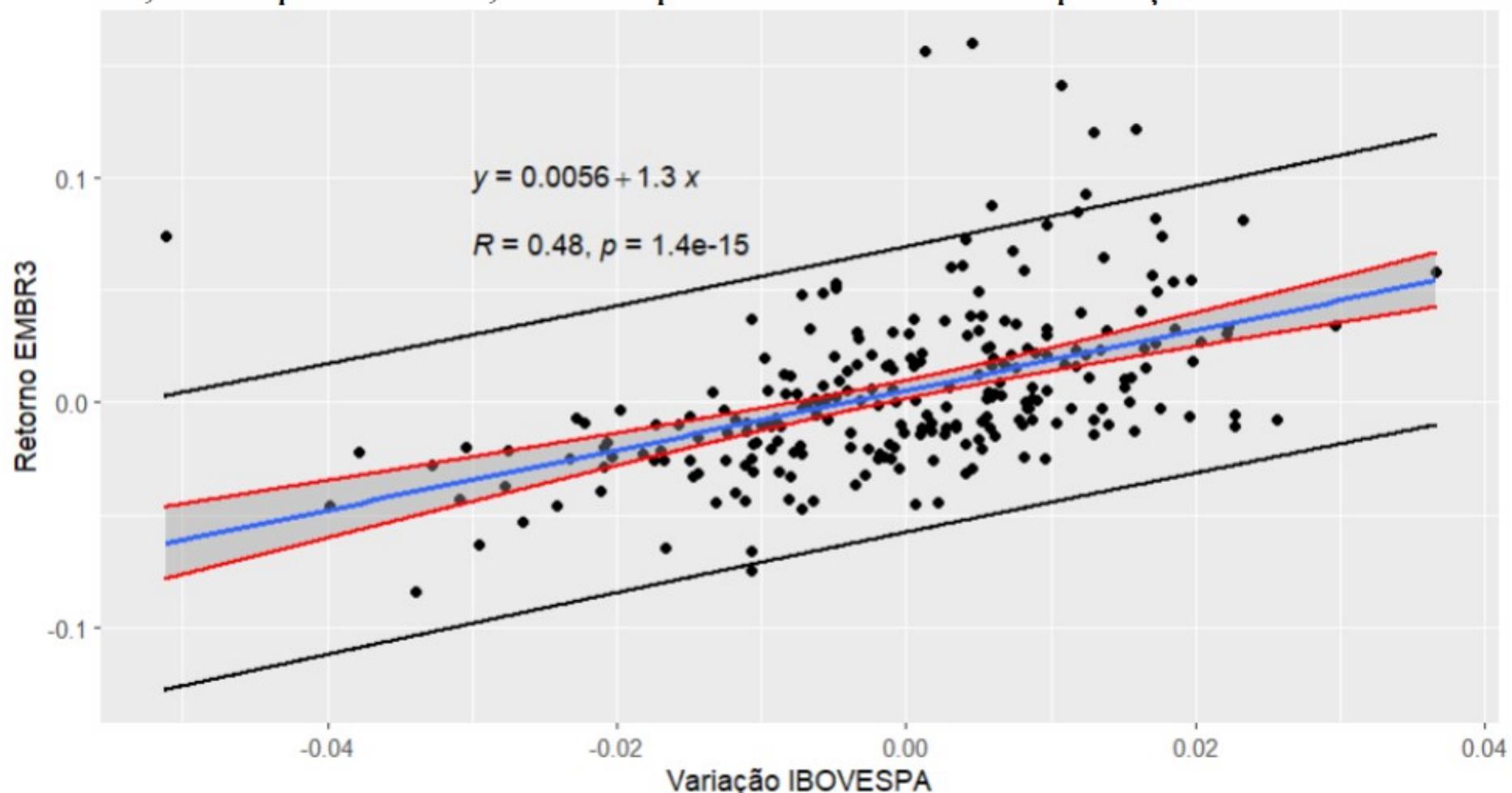


Figura-4 Reta de regressão ajustada para R_A e R_M com intervalo de predição.



Verificação da autorrelação entre os resíduos

Utilizamos o teste de Durbin-Watson para avaliar a autocorrelação dos resíduos. O resultado é apresentado a seguir.

Durbin-Watson test

```
data: regressao
DW = 2.1336, p-value = 0.8565
alternative hypothesis: true autocorrelation is greater than 0
```

Concluindo que a hipótese nula (H_0 : resíduos não são correlacionados) não é rejeitada.

Verificação da homocedasticidade dos resíduos

Fazendo o Teste de Breusch-Pagan em que a hipótese nula é dada por H_0 : Resíduos são Homocedásticos [Ou seja, os resíduos são distribuídos com igual varânci], temos o seguinte resultado:

```
bptest(regressao) # Está na library "lmtest"
studentized Breusch-Pagan test
```

```
data: regressao
BP = 0.011568, df = 1, p-value = 0.9143
```

Ou seja, não rejeitamos (com um nível de confiança de 5%, por exemplo) a hipótese de que os resíduos gerados pelo modelo de regressão são homocedásticos.

Teste de normalidade dos resíduos

Aplicando os Testes de Kolmogorov-Smirnov, Shapiro-Wilk, Anderson-Darling e de Lilliefors temos os resultados:

```
# KOLMOGOROV-SMIRNOV> ks.test(residuos, "pnorm")
    Asymptotic one-sample Kolmogorov-Smirnov test

data: residuos
D = 0.47376, p-value < 2.2e-16
alternative hypothesis: two-sided

# SHAPIRO-WILK> shapiro.test(residuos)
    Shapiro-Wilk normality test

data: residuos
W = 0.89143, p-value = 2.691e-12

# ANDERSON-DARLING> ad.test(residuos)
    Anderson-Darling normality test

data: residuos
A = 5.0146, p-value = 1.971e-12

# LILLIEFORST> lillie.test(residuos) # Também precisa estar na library "nortest"
    Lilliefors (Kolmogorov-Smirnov) normality test

data: residuos
D = 0.10293, p-value = 1.241e-06
```



Ou seja, em todos os testes de normalidade, rejeitamos a hipótese de que os resíduos se distribuem com distribuição Normal. Porém, como mencionado anteriormente, não normalidade dos erros não é um grande problema com amostras de tamanho grande. Nesse caso, o tamanho da amostra é de 246 observações. Para mais informações sobre não normalidade de resíduos, pode-se consultar as referências Wooldridge (2019) e Lumley *et al.* (2002).

Linearidade do modelo

A linha ajustada pela LOESS deve ser aproximadamente horizontal em zero. Na Figura-5 verificamos que a linha LOESS que ajusta aos dados não apresenta um comportamento funcional bem definido. Temos que a linha LOESS aproxima-se da linha horizontal próximo de zero, e dessa forma concluímos que a linearidade é satisfeita.

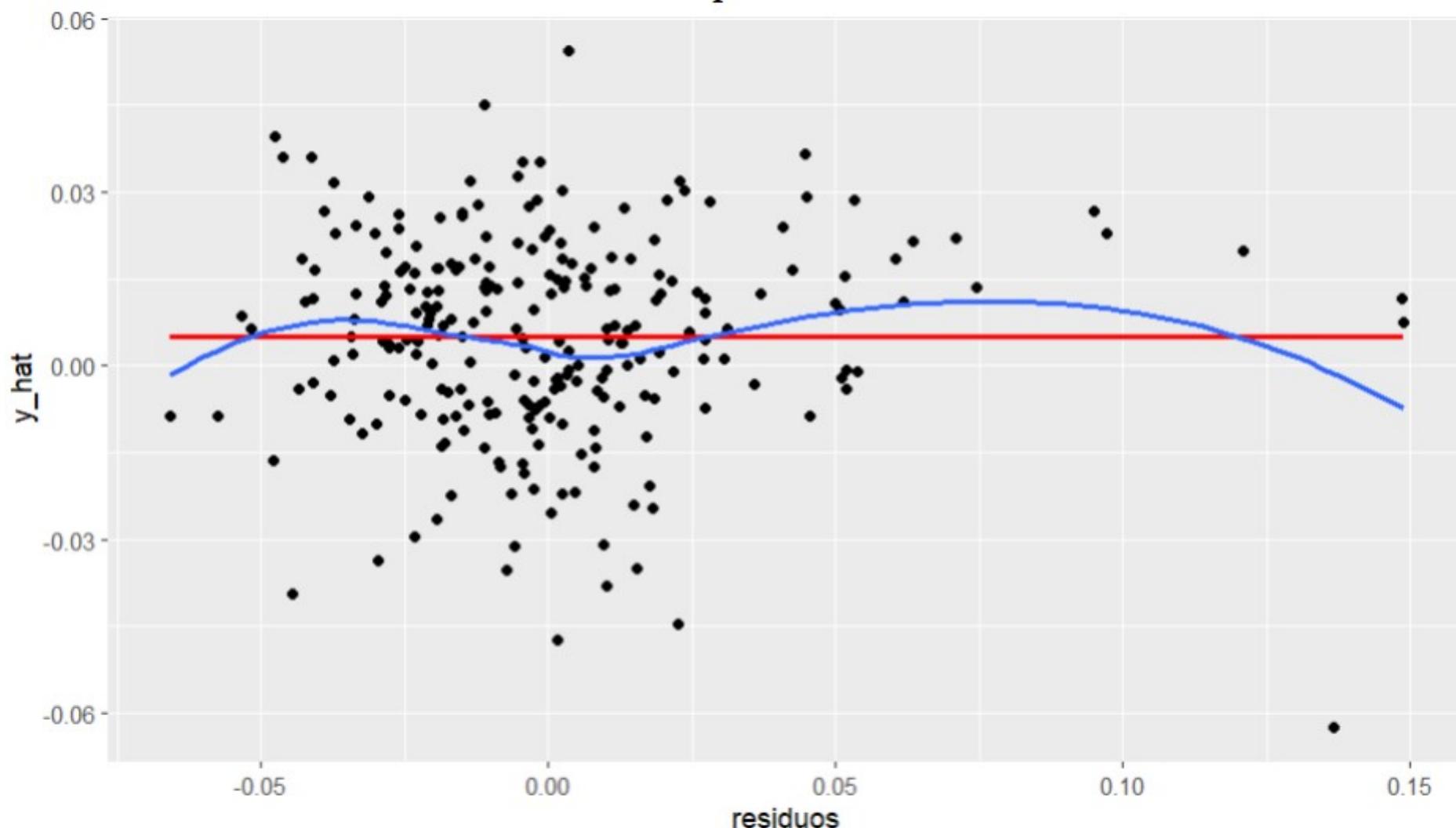


Figura-5 Linha ajustada LOESS.

Identificação de outliers

A Figura-6 e a Figura-7 mostram o gráfico outlier and Leverage Diagnostics e a distância de Cook para identificar outliers, verificamos a presença de vários outliers, sendo um desses valores excedendo bastante o limiar da distância de Cook.

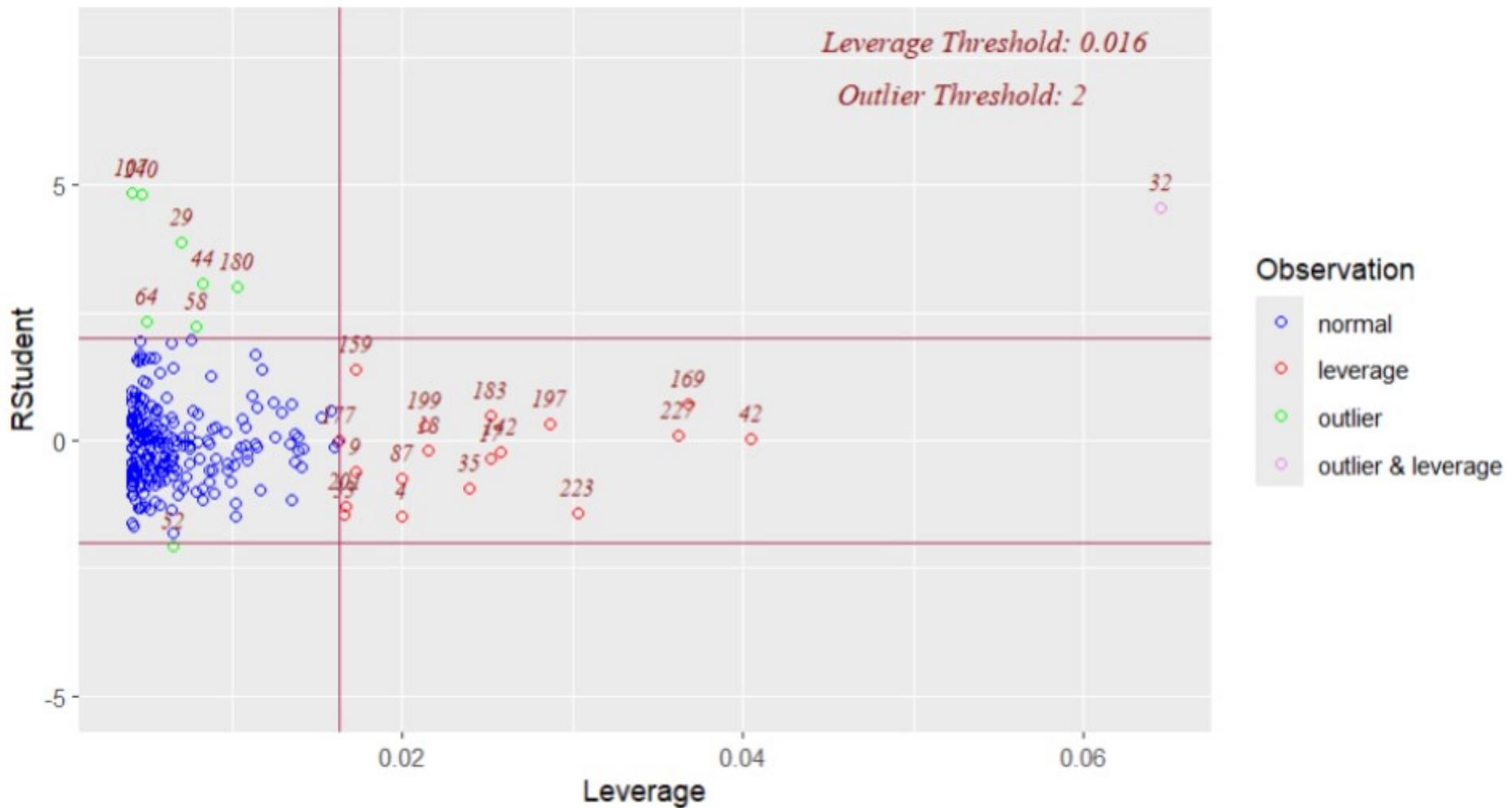


Figura-6 Gráfico outlier and Leverage Diagnostics.

Esse outlier, denotado como sendo a observação 32, corresponde exatamente ao valor da data 22 de Fevereiro de 2021. Esse ponto refere-se ao fato de que agência de classificação de risco Standard & Poor's (S&P) rebaixou nesta data (22/02/2021) o rating da empresa de BB+ para BB com perspectiva negativa, diante da expectativa de demanda mais fraca por causa da pandemia de coronavírus.

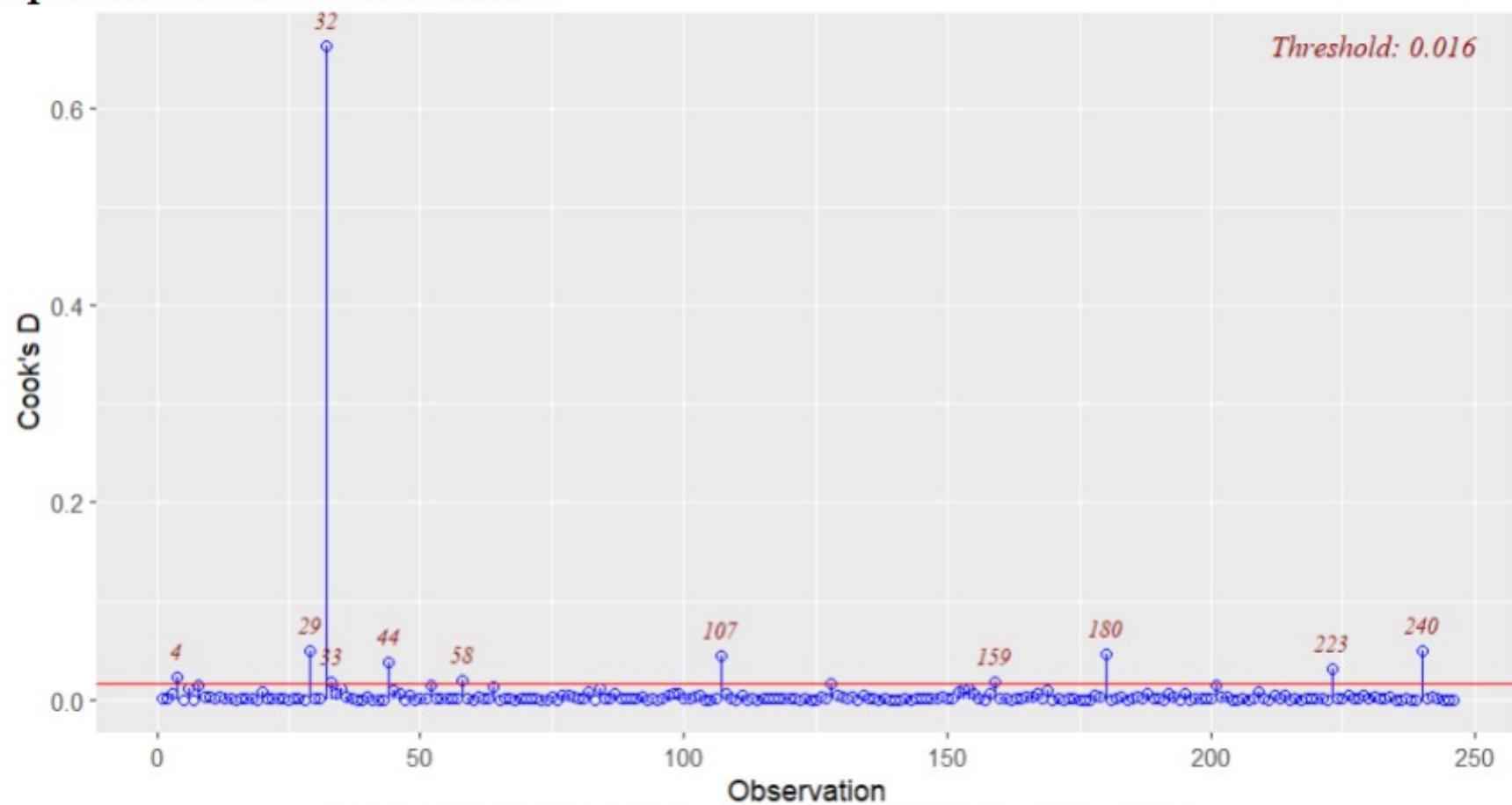


Figura-7 Distâncias de Cook para EMBR3 em 2021.

PO-249



Modelo de Regressão usando Neurônio Único

ai



3.3 Modelo de Regressão usando Neurônio Único|

O neurônio artificial constitui-se na unidade estrutural e funcional de um sistema de aprendizado de máquina (em inglês, Machine Learning - ML), apresenta uma complexidade inerente às tarefas que precisa executar, sendo sua formulação matemática sujeita às regras de implementação e organização de conjuntos interconectados de componentes. A viabilidade e as aplicações de uma rede neural com neurônio único mostraram-se uma tarefa desafiadora desde o início.

Sendo o início constituído por uma sequência de trabalhos seminais, o primeiro sendo a construção teórica do neurônio artificial, em 1943 com o trabalho de McCulloch e Pitts (*A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics, 5 (4), 115-133*), a seguir, em 1949 é apresentada a aprendizagem Hebbiana (*The organization of behavior: A neuropsychological theory. New York, Wiley*) e o Perceptron de Rosenblatt em 1958 sendo a evolução seguinte (*The perceptron: a probabilistic model for information storage and organization in the brain. Psychological review, 65 (6), 386*).

Porém, em 1969, Papert e Minsky (*Perceptrons: An Introduction to Computational Geometry*) destacaram algumas limitações computacionais do perceptron de camada única. Isso levou muitos na comunidade de Inteligência Artificial (IA) a acreditar que as estruturas das redes neurais eram demasiado limitadas para lidar com problemas mais complexos do mundo real. Isso levou ao que é considerado o “Primeiro Inverno da Inteligência Artificial”, iniciando no final da década de 1960 e adentrando à década de 1970 em que as soluções apresentadas nessa área caracterizavam-se por soluções a problemas específicos e pouco uso comercial das pesquisas acadêmicas. Apesar do impacto gerado pelas críticas teóricas aos primeiros modelos matemáticos de representação do neurônio, outros projetos seguiram adiante evidenciando outras abordagens e a necessidade de sistemas inteligentes.



Dessa forma, vamos considerar o processo de aprendizagem para um modelo de neurônio único utilizando como algoritmo de treinamento o gradiente descendente.

Modelo de neurônio único: consiste em único neurônio ou nó, conforme mostrado na Figura-1. Esse tipo mais simples de rede neural, é caracterizado pela soma das multiplicações ponderadas dos respectivos sinais de entrada pelos pesos sinápticos, adicionados a um termo de bias (cujo efeito é aumentar ou diminuir a entrada líquida para o bloco elementar seguinte), passando por uma função de ativação (que determina a faixa de valores de ativação do neurônio artificial).

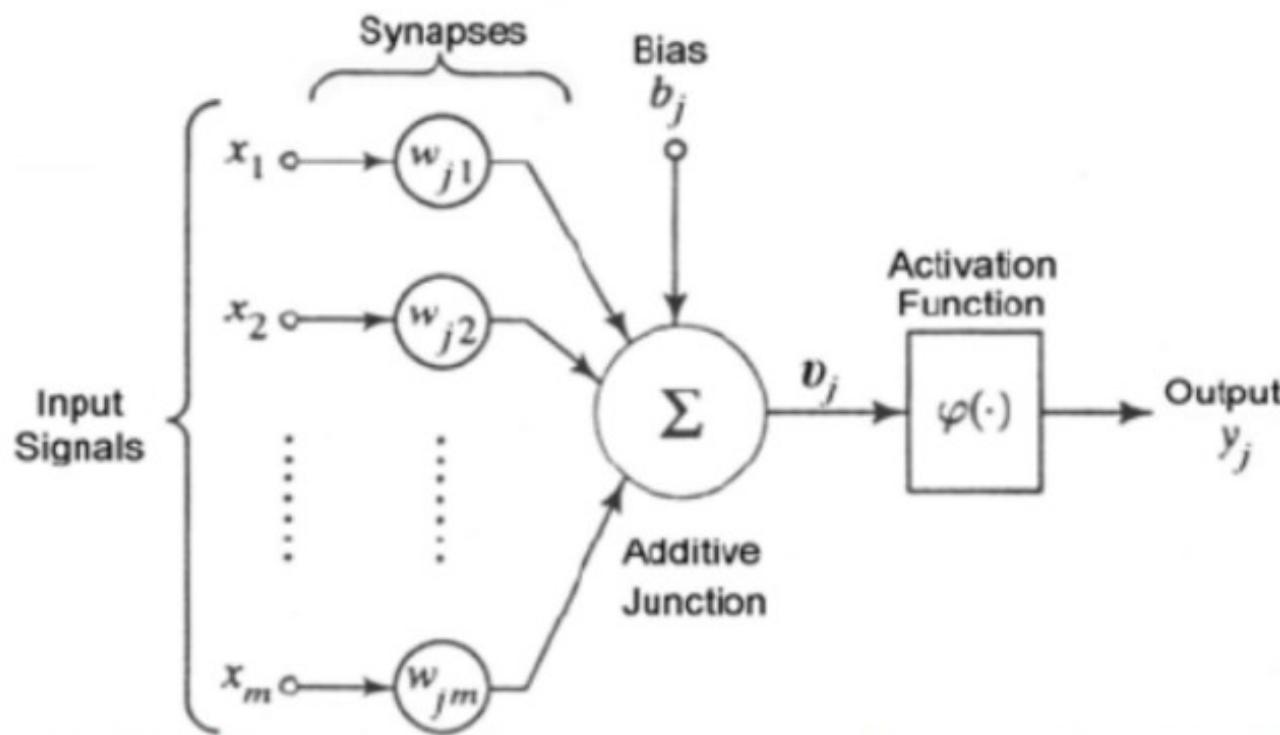


Figura-1 Modelo de Neurônio Único. Fonte: (<https://www.researchgate.net/figure/Artificial-Neuron-models->

Um modelo de rede neural artificial (RNA) constituída por um único neurônio pode ser aplicado basicamente para duas tarefas: (i) **Regressão**, onde fazemos a predição de uma variável de interesse. Nesse caso, utilizamos uma função de ativação linear, ou seja, a função identidade; (ii) **Classificação**, em que é implementado um classificador binário, sendo as saídas exibindo rótulos que podem pertencer a uma de apenas duas classes. Para a tarefa de classificação, usaremos a função sigmóide.

De maneira simplificada, um neurônio único executa as operações de processamento mostradas nas equações a seguir.

$$u = \mathbf{w}^T \mathbf{x}, \quad (1)$$

$$v = \mathbf{w}^T \mathbf{x} + w_0, \quad (2)$$

$$y = f(v). \quad (3)$$



Como vimos anteriormente, um Modelo de Regressão Linear Múltipla corresponde a um modelo funcional em que procuramos explicar uma variável y a partir de algumas variáveis x , podendo ser escrito como:

$$y = w_0 + w_1x_1 + w_2x_2 + \dots + w_mx_m + \varepsilon. \quad (4)$$

Onde,

- y é a variável resposta;
- w_0 , nos modelos de regressão é chamado de intercepto, é o valor de y quando a todas as outras variáveis x é atribuído o valor 0;
- w_i , corresponde ao i -ésimo coeficiente associado à variável explicativa x_i ;
- x_i , i -ésima variável explicativa;
- ε , é um termo de erro aleatório.

A equivalência do modelo de regressão com duas variáveis explicativas, em termos de um neurônio, pode ser conseguida utilizando a arquitetura mostrada na Figura-2.

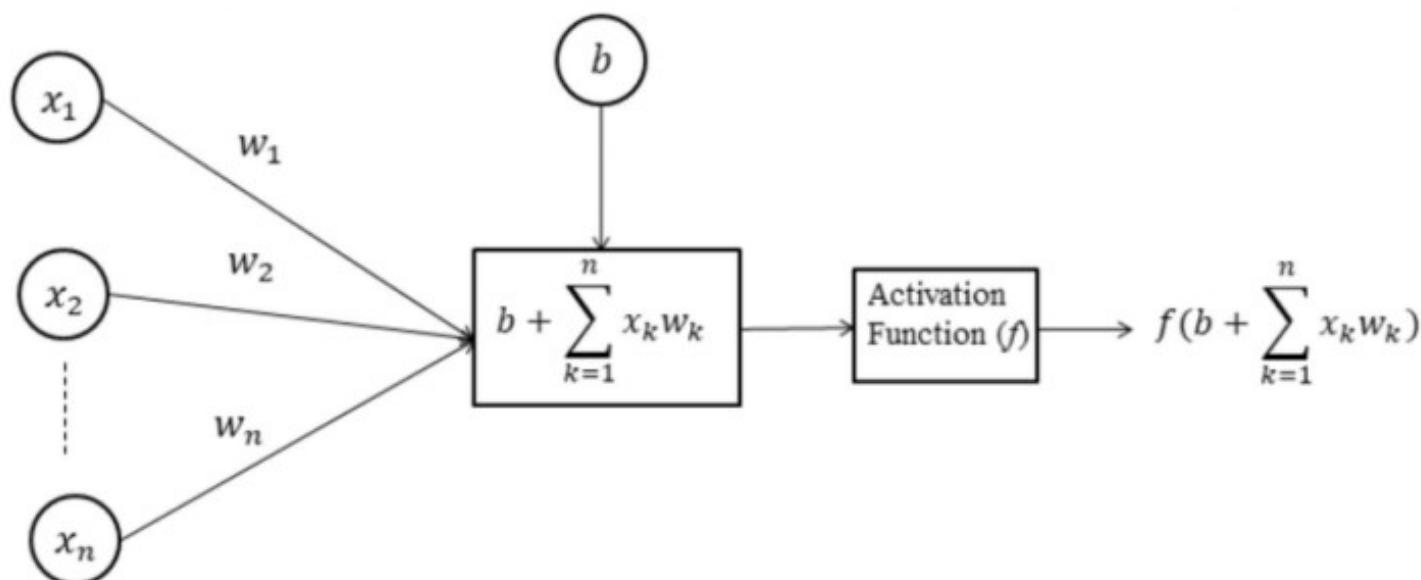


Figura-2 Neurônio único que gera um modelo de regressão. [ADAPTAR A FIGURA]



A seguir é apresentado um exemplo de código em Python que calcula a saída y a partir das entradas x_1 e x_2 na rede neural artificial apresentada na Figura-2.

EXEMPLO 12.1 Código Python para obter a saída do neurônio na Figura-2

Antes de escrever o código em Python, temos que:

$$u = w_1x_1 + w_2x_2. \quad (5)$$

Depois disso é adicionado o bias, produzindo:

$$v = w_0 + w_1x_1 + w_2x_2. \quad (6)$$

Aplicando a função de ativação identidade, resulta em:

$$y = f(v). \quad (7)$$

Considerando como sinais de entrada $x_1 = 3$, $x_2 = 5$, pesos das respectivas conexões sinápticas $w_1 = 1$ e $w_2 = 2$ e o bias $w_0 = -3$, o código Python para obter a saída y na Figura-2 é mostrado a seguir.

```
def neuronio_unico(w, w_0, x):
    u = 0
    v = 0
    for entrada_x, peso_w in zip(x, w):
        u += entrada_x * peso_w
    v = u + w_0
    y = linear(v)
    return y

# Considerando como entrada x1 = 3 e x2 = 5, com pesos iniciais para w iguais a
# w1 = 1 e w2 = 2
x = [3, 5]
w = [1, 2]
w_0 = -3

y = neuronio_unico(w, w_0, x)
print(y)
```

O que resulta na saída $y = 10$.



12.1.1 Modelo de Regressão e Gradiente Descendente

Utilizamos o método do gradiente descendente para determinar os pesos w_j e o bias w_0 que minimizam uma função de custo do modelo. Essa função de custo depende dos parâmetros que serão estimados, denotaremos por $C(\hat{y}^{(i)}, y^{(i)}) = C(w_0, w_1, \dots, w_m)$ e calculada como:

$$C(\hat{y}^{(i)}, y^{(i)}) = \frac{1}{2n} \sum_{i=1}^n (\hat{y}_{\text{predito}}^{(i)} - y_{\text{observado}}^{(i)})^2 = \frac{1}{2n} \sum_{i=1}^n (\hat{y}^{(i)} - y^{(i)})^2, \quad (8)$$

onde n corresponde ao número de observações consideradas, e está multiplicado por 2 por conveniência devido as derivadas que serão tomadas *a posteriori*. Em termos da linguagem de métodos de otimização podemos escrever:

função : $f_w(x) = w_0 + w_1x_1 + \dots + w_mx_m$

parâmetros : w_0, w_1, \dots, w_m

função custo : $C(w_0, w_1, \dots, w_m; \hat{y}^{(i)}, y^{(i)}) = \frac{1}{2n} \sum_{i=1}^n [f_w(x^{(i)}) - y^{(i)}]^2 \quad (9)$

objetivo : minimizar $\underset{w_0, w_1, \dots, w_m}{C(w_0, w_1, \dots, w_m)}$

Em termos de nomenclatura vamos utilizar a terminologia do software Python, onde variáveis de uma função são chamadas de parâmetros ou argumentos, sendo que: (i) **funções**: são blocos de código que executam operações ou tarefas específicas. Ao definir uma função (ou regra), os programadores podem evitar a repetição de código, chamando funções previamente definidas; (ii) **parâmetros ou params**: são variáveis que ficam disponíveis entre parênteses de uma função definida e (iii) **argumentos ou args**: são os valores atribuídos a uma função quando ela é chamada. O termo parâmetro usado no Python não deve ser confundido com pesos ou coeficientes de modelos de aprendizado de máquina. Ou seja, por exemplo, definimos uma função $g(x)$ com um parâmetro x . O corpo da função verifica qual condição é atendida para executar o bloco de código correspondente. Avaliando a função $g(x)$ no valor 9, implica que esse valor 9 é o argumento da função. A função $g(x)$ é chamada e retorna o resultado.

Para implementar o método do gradiente descendente podemos utilizar o pseudocódigo mostrado a seguir aplicado ao treinamento do neurônio único da Figura-3.

Pseudocódigo do algoritmo da descida do gradiente

Iniciar os pesos w_0, w_1, \dots, w_m

Setar o número de épocas num_epocas

Setar a taxa de aprendizagem α

for $k = 0$ **to** num_epocas **do**

Avalie $grad^{(k)} = \nabla L(w_0^{(k)}, w_1^{(k)}, \dots, w_m^{(k)})$

$w^{(k+1)} = w^{(k)} - \alpha \cdot grad^{(k)}$

end for

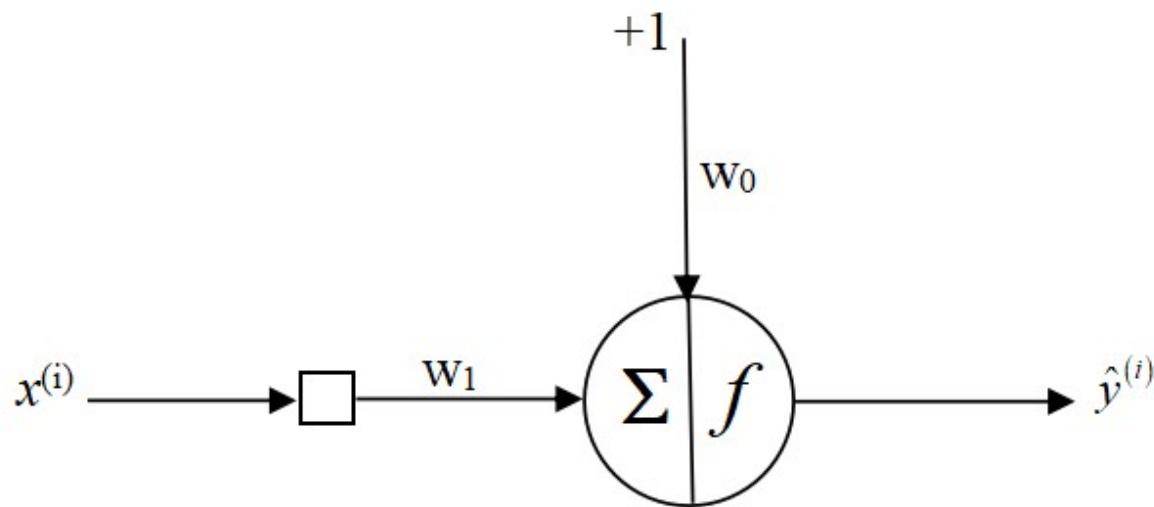


Figura-3 Neurônio único com uma entrada para modelo de regressão.

No caso do neurônio mostrado na Figura-3, executamos o treinamento utilizando o seguinte conjunto de equações:

$$v^{(i)} = w_0 + w_1 x^{(i)}. \quad (10)$$

$$\hat{y}^{(i)} = f(v^{(i)}), \quad (11)$$

sendo f , a função de ativação linear, temos que:

$$\hat{y}^{(i)} = w_0 + w_1 x^{(i)}. \quad (12)$$



O valor predito, que corresponde à saída do neurônio, tem sobrescrito (i) porque para cada valor do sinal de entrada x será gerado um valor de saída. A notação sobrescrita (i) indica qual das n observações de dados estamos considerando. Como a saída não será igual à entrada, essa diferença é um custo associado ao modelo que representa a variável y .

Quando consideramos somente um valor do sinal de entrada, digamos o valor (i) do sinal de entrada, o custo C associado a essa entrada será de

$$C(\hat{y}^{(i)}, y^{(i)}) = \frac{1}{2} [\hat{y}^{(i)} - y^{(i)}]^2. \quad (13)$$

Porém, é importante ressaltar que C é uma função que depende dos parâmetros que correspondem aos pesos sinápticos e ao bias da rede. Nesse caso, nossa função de custo é do tipo $C(w_0, w_1; \hat{y}^{(i)}, y^{(i)})$.

Como o sinal de entrada é constituído por n valores, a função custo que será calculada a cada época de treinamento é dada por

$$C(w_0, w_1; \hat{y}^{(i)}, y^{(i)}) = \frac{1}{2n} \sum_{i=1}^n [\hat{y}^{(i)} - y^{(i)}]^2. \quad (14)$$

Assim, temos uma função de custo calculada para toda a mostra de treinamento que é descrita em termos das diferenças (ou seja, os erros) de cada valor predito pelo modelo e o respectivo valor da variável de interesse y . Como queremos minimizar essa soma de erros, precisamos atualizar os parâmetros conforme alguma regra, que nesse caso será a descida do gradiente. Dessa forma, os pesos são atualizados através das seguintes regras:

$$\begin{aligned} w_0 &\leftarrow w_0 - \alpha \frac{\partial C(w_0, w_1; \hat{y}^{(i)}, y^{(i)})}{\partial w_0}, \\ w_1 &\leftarrow w_1 - \alpha \frac{\partial C(w_0, w_1; \hat{y}^{(i)}, y^{(i)})}{\partial w_1}. \end{aligned} \quad (15)$$

Sendo o parâmetro α usado para regular o tamanho do passo da descida do gradiente e denominado de taxa de aprendizagem.

A determinação das derivadas de C com relação a w_0 e w_1 é apresentada a seguir.

$$\frac{\partial C(w_0, w_1; \hat{y}^{(i)}, y^{(i)})}{\partial w_0} = \frac{\partial}{\partial w_0} \left\{ \frac{1}{2n} \sum_{i=1}^n [w_0 + w_1 x^{(i)} - y^{(i)}]^2 \right\} = \frac{1}{n} \sum_{i=1}^n [w_0 + w_1 x^{(i)} - y^{(i)}]. \quad (16)$$

$$\frac{\partial C(w_0, w_1; \hat{y}^{(i)}, y^{(i)})}{\partial w_1} = \frac{\partial}{\partial w_1} \left\{ \frac{1}{2n} \sum_{i=1}^n [w_0 + w_1 x^{(i)} - y^{(i)}]^2 \right\} = \frac{1}{n} \sum_{i=1}^n [w_0 + w_1 x^{(i)} - y^{(i)}] x^{(i)}. \quad (17)$$



Para continuar o processo de treinamento podemos basicamente utilizar duas estratégias, a primeira considerando que em cada época calculamos o gradiente total, ou seja, a soma de todos os valores de gradiente para cada valor de sinal de entrada, o que é denominado de aprendizado por lote (batch gradient descent). |

```
#####
#          REGRESSÃO USANDO NEURÔNIO ÚNICO [RNA]
#
#####

import numpy as np
import matplotlib.pyplot as plt

# Definição do Modelo Linear
def modelo_linear(w, w_0, x):
    v = np.dot(w, x) + w_0
    return v

# Treinamento do Modelo de Regressão Linear
def treino_modelo_regressao(X, Y, alfa, epocas, imprimir):
    # Inicializar os pesos
    w_0 = 0
    w = np.zeros(len(X[0]))
    rmse = [] # Armazenar a medida de ajuste para cada época
    for epoca in range(epocas):
        custo_total = 0
        for x, y in zip(X, Y):
            v = modelo_linear(w, w_0, x)
            erro = v - y
            custo_total += (erro ** 2) / 2
            # Atualizar os Pesos
            w_0 -= alfa * erro * 1
            w -= alfa * erro * x.flatten()
            if imprimir:
                print("x:", x, "y:", y, "Diferença ou Desvio:", erro)
                print("Pesos:", w, w_0)
        acuracia = np.sqrt(custo_total / len(X)) # Calcula: Root Mean Squared Error [RMSE]
        rmse.append(acuracia)
```



```
# Mostrar a Alteração da Função Custo a Medida que as Épocas Passam
mostrar_a_cada = max(1, epochas // 10)
if epocha % mostrar_a_cada == 0:
    print("Época", epocha, "Custo Total", custo_total)
return w, w_0, rmse

# Exemplo [Mendenhall e Sincich, 2016, Pág. 486]
x_entrada = np.array([[1], [2], [3], [4], [5]])
y_saida = np.array([1, 1, 2, 2, 4])
alfa = 0.01
epochas = 1000
imprimir = False

# Treinando o Modelo
w, w_0, rmse = treino_modelo_regressao(x_entrada, y_saida, alfa, epochas, imprimir)
print("\nPesos Finais:")
print(w, w_0)
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 5))

# Subplot: RMSE por Épocas
epochas = range(1, len(rmse) + 1)
ax1.plot(epochas, rmse, marker='o', linestyle='-', color='black', label='Root Mean Squared Error')
ax1.set_title('RMSE por Épocas')
ax1.set_xlabel('Épocas')
ax1.set_ylabel('RMSE')
ax1.legend()
ax1.grid(True)
```

```
# Plot do Ajuste Linear
def plot_ajuste_linear_rna(X, Y, w, w_0):
    ax2.scatter(X.flatten(), Y, color='black')
    ax2.set_xlabel('x')
    ax2.set_ylabel('y')
    ax2.set_title('Ajuste Linear - Neurônio Único')
    ax2.plot(X.flatten(), w * X.flatten() + w_0, color='black')
    ax2.grid(True)
plot_ajuste_linear_rna(x_entrada, y_saida, w, w_0)
```



```
#####
#          REGRESSÃO LINEAR SIMPLES [RLS]
#
#####

import numpy as np
import statsmodels.api as sm
import matplotlib.pyplot as plt

# Observações
x = np.array([1, 2, 3, 4, 5])
y = np.array([1, 1, 2, 2, 4])

# Adicionando o intercepto
x = sm.add_constant(x)

# Ajuste da Regressão usando o modelo OLS (Ordinary Least Squares - Mínimos Quadrados Ordinários)
modelo = sm.OLS(y, x).fit()

# Obtenção dos Valores Ajustados (Predições)
y_preditto = modelo.predict(x)

# Mostrando os Resultados da Regressão pelo OLS
print("Resultados da Regressão:")
print(modelo.summary())

# Inclinação e Intercepto
slope = modelo.params[1]
intercept = modelo.params[0]
fig, (ax3, ax4) = plt.subplots(1, 2, figsize=(15, 5))
```

```
# Plotando os Dados Originais e a Linha Ajustada
ax3.scatter(x[:, 1], y, label='Dados Originais - Observações')
ax3.plot(x[:, 1], y_predito, color='red', label=f'Modelo Ajustado: y = {intercept:.2f} + {slope:.2f}x')
ax3.set_xlabel('x')
ax3.set_ylabel('y')
ax3.set_title('Ajuste Linear - Regressão Simples')
ax3.legend()
ax3.grid(True)

# Plot dos Ajustes da RNA e da RLS
ax4.scatter(x[:, 1], y, label='Dados Originais - Observações')
ax4.plot(x[:, 1], y_predito, color='red', label='Modelo Ajustado RLS')
ax4.plot(x_entrada.flatten(), w * x_entrada.flatten() + w_0, color='black', label='Modelo Ajustado RNA')
ax4.set_xlabel('x')
ax4.set_ylabel('y')
ax4.set_title('Ajustes da RNA e da RLS')
ax4.legend()
ax4.grid(True)
```



Época 0 Custo Total 9.158778596611885
Época 100 Custo Total 0.6634318475621844
→ Época 200 Custo Total 0.656325677470472
Época 300 Custo Total 0.6555593631366254
Época 400 Custo Total 0.6556219112491557
Época 500 Custo Total 0.6557144536033169
Época 600 Custo Total 0.6557646048141427
Época 700 Custo Total 0.6557875296180113
Época 800 Custo Total 0.6557974395945291
Época 900 Custo Total 0.6558016331743399

Pesos Finais:

[0.73684377] -0.17576521601331777

Figura-4 Custo total a medida que são geradas as épocas.

OLS Regression Results

```
=====
Dep. Variable:                      y      R-squared:                 0.817
Model:                            OLS      Adj. R-squared:            0.756
Method:                           Least Squares      F-statistic:                13.36
Date:                            Tue, 09 Jan 2024      Prob (F-statistic):        0.0354
Time:                             12:08:02      Log-Likelihood:           -3.3094
No. Observations:                  5      AIC:                      10.62
Df Residuals:                     3      BIC:                      9.838
Df Model:                          1
Covariance Type:                nonrobust
=====
```

	coef	std err	t	P> t	[0.025	0.975]
const	-0.1000	0.635	-0.157	0.885	-2.121	1.921
x1	0.7000	0.191	3.656	0.035	0.091	1.309

=====

Omnibus:	nan	Durbin-Watson:	2.509
Prob(Omnibus):	nan	Jarque-Bera (JB):	0.396
Skew:	-0.174	Prob(JB):	0.821
Kurtosis:	1.667	Cond. No.	8.37

=====

Figura-5 Resultado do modelo de regressão linear simples.

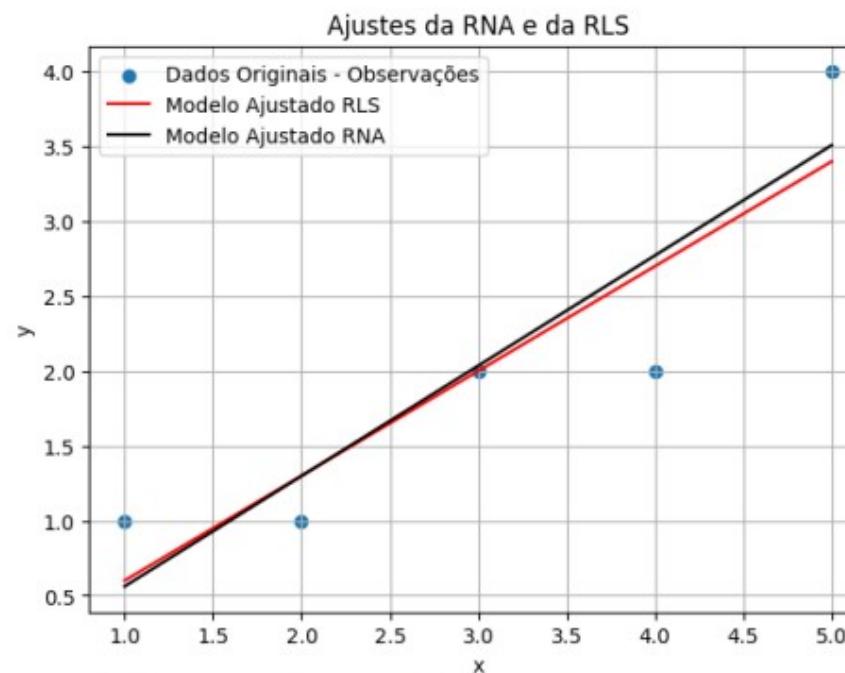
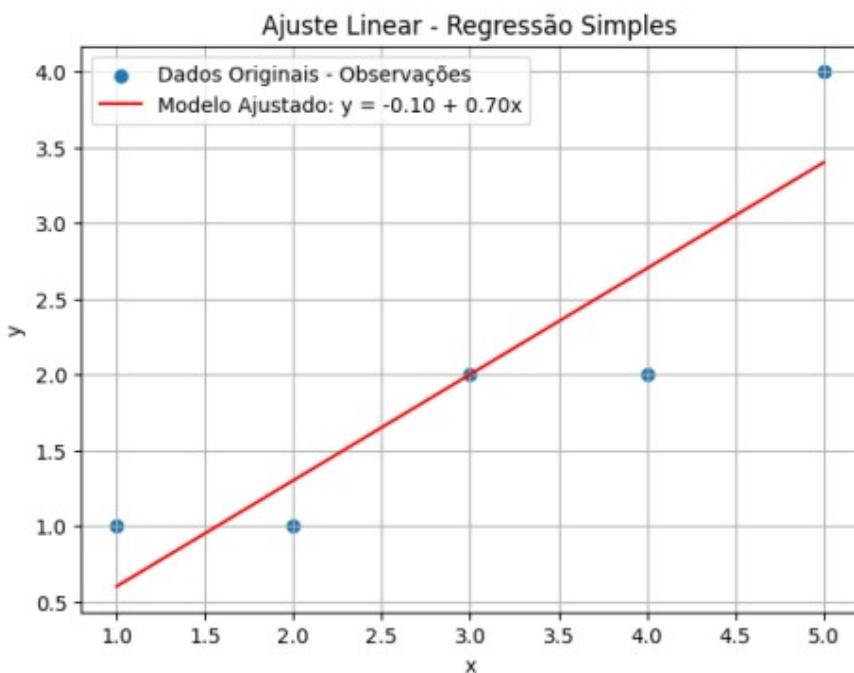
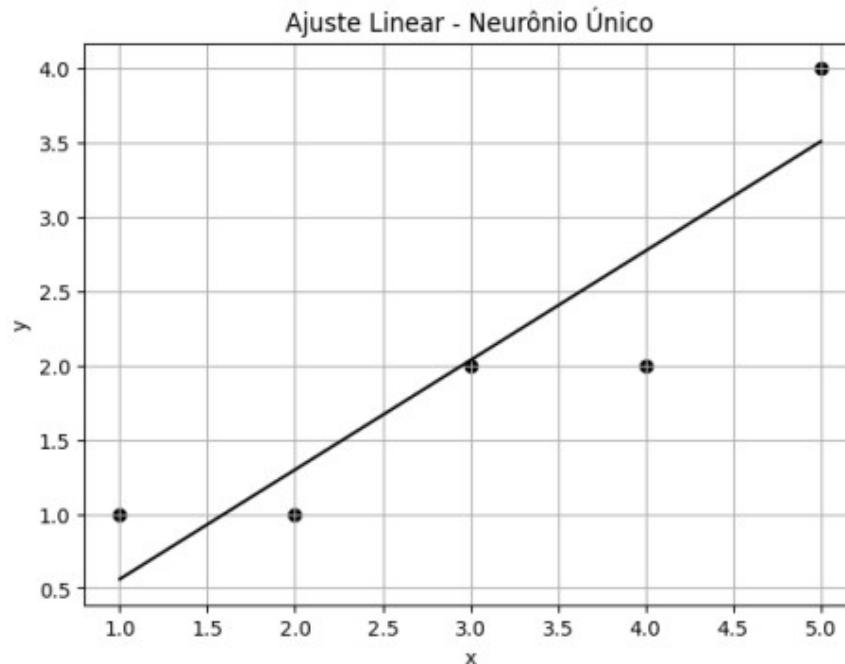
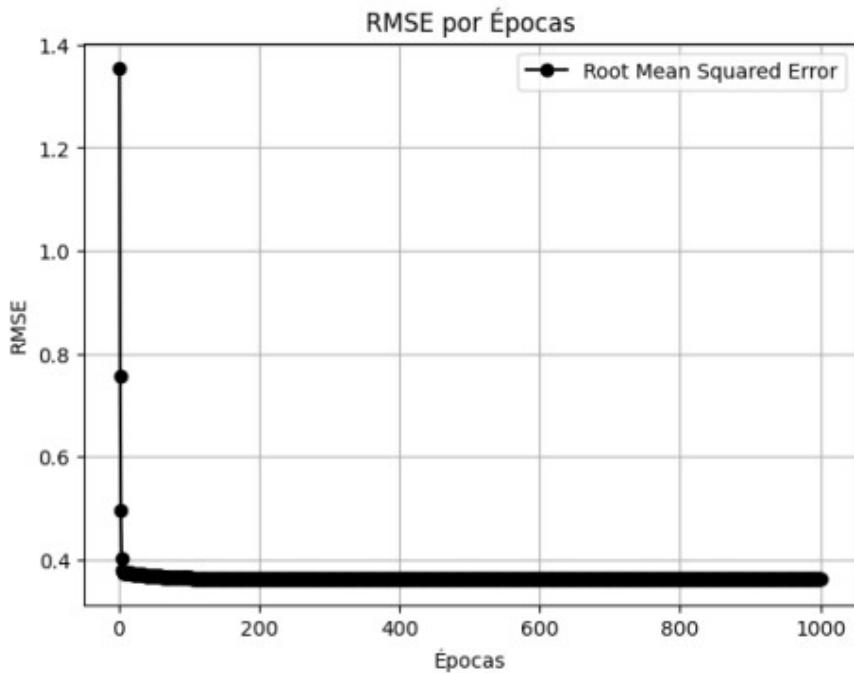


Figura-6 Curvas ajustadas pelo modelo de Regressão Linear Simples e pelo Neurônio único.

Ajustes da RNA e da RLS

