****

**UNIVERSIDADE FEDERAL DO MARANHÃO**

**BACHARELADO INTERDISCIPLINAR EM CIÊNCIA E TECNOLOGIA**

**INTELIGÊNCIA ARTIFICIAL**

**Regressão Linear**

Juan Pablo Furtado Mondego

**São Luís**

**2025**

**1. Introdução**

Neste projeto, o estudante deverá implementar uma solução de regressão linear simples do zero, utilizando Python e bibliotecas científicas básicas, para investigar e compreender como a taxa de aprendizado (α) e a inicialização dos parâmetros (θ inicial) influenciam o comportamento do algoritmo de descida do gradiente. O desenvolvimento inclui tanto a criação dos componentes fundamentais do modelo — como os exercícios de aquecimento (warm up), visualização dos dados, cálculo da função de custo e implementação do gradiente descendente — quanto a realização de experimentos comparativos e análise crítica dos resultados.

A abordagem envolve primeiro implementar todas as funções essenciais que compõem o ciclo de aprendizado da regressão linear, garantindo entendimento do fluxo matemático e computacional envolvido. Em seguida, será necessário gerar diferentes gráficos que ajudem a visualizar o processo de aprendizagem, como a curva de convergência da função de custo, o ajuste da reta de regressão sobre os dados, a superfície 3D da função de custo e o contorno dessa função ilustrando a trajetória do gradiente.

Os experimentos principais envolvem comparar, em um mesmo contexto, a evolução do custo ao longo das iterações para três taxas de aprendizado diferentes, e visualizar o efeito de diversas inicializações dos parâmetros no processo de otimização, fixando a taxa de aprendizado e variando os valores iniciais de θ, tanto fixos quanto aleatórios. As trajetórias resultantes devem ser analisadas e ilustradas nos gráficos de contorno, permitindo observar como o ponto de partida influencia o caminho de convergência.

Ao final, é esperado que o estudante elabore uma análise escrita, em formato dissertativo seguindo as normas da ABNT, discutindo os resultados obtidos, inserindo os gráficos gerados e explicando detalhadamente como a taxa de aprendizado pode afetar a estabilidade e a velocidade de convergência, assim como a relevância da inicialização dos pesos para o bom desempenho do algoritmo. É importante incluir legendas e comentários claros em todas as figuras, bem como relacionar os conceitos discutidos com práticas modernas em deep learning, como o fine-tuning de redes neurais.

**1.1 Redes Neurais Artificiais**

As Redes Neurais Artificiais (RNAs) surgiram a partir de uma analogia ao funcionamento do cérebro humano, no qual bilhões de neurônios se interconectam para processar informações de forma paralela e distribuída (McCulloch & Pitts, 1943). Cada neurônio artificial recebe entradas associadas a pesos, realiza uma combinação linear desses sinais e aplica uma função de ativação para gerar a saída (Haykin, 1999). Essa capacidade de aprendizado ocorre por meio do ajuste dos pesos sinápticos com base em algoritmos de retropropagação do erro (Rumelhart, Hinton & Williams, 1986), que minimizam uma função de custo específica do problema, como a entropia cruzada no caso de classificações (slides-12).

Ao contrário de métodos tradicionais de aprendizado de máquina, que frequentemente exigem engenharia de atributos manual, as RNAs são capazes de extrair representações dos dados de forma mais autônoma, sobretudo quando se empregam redes profundas (Deep Learning). Esse enfoque garante maior flexibilidade e um poder de generalização superior em diversos cenários, como reconhecimento de voz, processamento de texto e classificação de imagens (LeCun, Bengio & Hinton, 2015).

# **2 Metodologia**

A metodologia adotada neste projeto baseia-se na construção modular dos elementos que compõem o ciclo de aprendizagem da regressão linear, proporcionando ao estudante uma visão clara tanto da teoria quanto da implementação prática dos algoritmos. Cada função foi projetada para cumprir um papel específico dentro do fluxo do modelo, permitindo a separação de responsabilidades, maior organização do código e facilidade na identificação de possíveis erros ou melhorias.

O arquivo warm\_up\_exercises.py tem como objetivo inicial preparar o estudante para a lógica de manipulação de matrizes e vetores em Python, especialmente utilizando a biblioteca NumPy, que é fundamental para qualquer projeto de machine learning. As funções contidas neste módulo, como a geração de uma matriz identidade, a criação de vetores de 1’s para simular o termo de bias e a concatenação de colunas para formar a matriz de atributos, servem de base para o entendimento do tratamento de dados e preparação das entradas para o modelo. Funções como o cálculo do produto matricial, o erro quadrático entre predições e valores reais e o custo médio aproximam o estudante da rotina de avaliação de desempenho em modelos preditivos.

Já o arquivo plot\_data.py desempenha um papel fundamental na visualização dos dados. A capacidade de visualizar a relação entre as variáveis independentes e dependentes é crucial para qualquer análise exploratória. Por meio da representação gráfica dos pontos de dados, torna-se possível verificar tendências, identificar padrões e, eventualmente, antecipar desafios no ajuste do modelo.

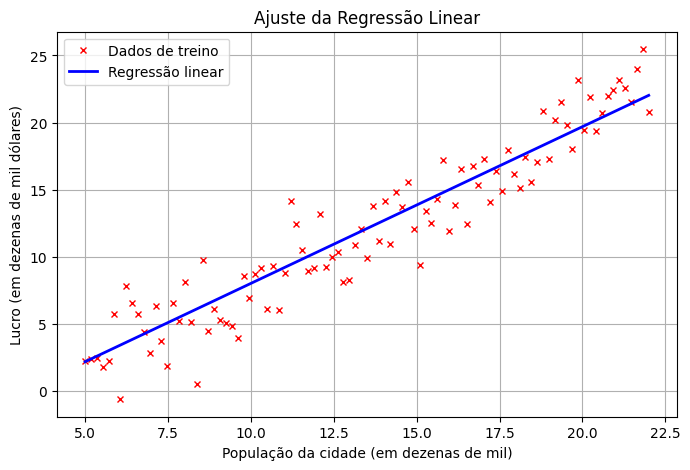
No que diz respeito ao cálculo da função de custo, o módulo compute\_cost.py encapsula toda a lógica relacionada à avaliação quantitativa do ajuste do modelo aos dados observados. A função principal deste arquivo recebe como parâmetros a matriz de atributos, o vetor de valores reais e o vetor de parâmetros do modelo, retornando o valor escalar do custo que serve como métrica para o processo de otimização. Essa abordagem modularizada favorece tanto o reuso da função em diferentes pontos do projeto quanto a clareza na separação entre o que é cálculo matemático e o que é lógica de controle.

O algoritmo de descida do gradiente, peça central do aprendizado do modelo, é implementado no arquivo gradient\_descent.py. Esta função é responsável por realizar as atualizações sucessivas dos parâmetros do modelo, sempre na direção de minimização da função de custo. O gradiente do custo em relação aos parâmetros é calculado a cada iteração e utilizado para ajustar os valores de θ, armazenando tanto o histórico dos custos quanto das trajetórias dos parâmetros ao longo do processo. Esse detalhamento permite, posteriormente, a análise visual do comportamento do algoritmo e a avaliação da influência das escolhas de taxa de aprendizado e inicialização dos pesos.

O script principal, regressao-linear-ex1.py, orquestra a execução de todas essas funções. Ele coordena desde a leitura e preparação dos dados, passando pela execução dos exercícios de aquecimento, a realização dos experimentos controlados, até a geração dos gráficos necessários para a análise comparativa. É neste arquivo que se define o fluxo geral da experiência: o estudante carrega os dados, inicializa os parâmetros, escolhe as taxas de aprendizado e pontos iniciais, executa o gradiente descendente e coleta os resultados para interpretação. O código principal ainda é responsável por salvar os gráficos gerados em arquivos para posterior inclusão no relatório, garantir a correta visualização dos experimentos e fornecer pontos de pausa para facilitar o acompanhamento do processo passo a passo.

Com essa estrutura, a metodologia garante que cada componente do projeto seja compreendido em sua essência e, ao mesmo tempo, integrado em um fluxo contínuo que conduz do pré-processamento dos dados à análise crítica dos resultados.

# **3 Resultados**

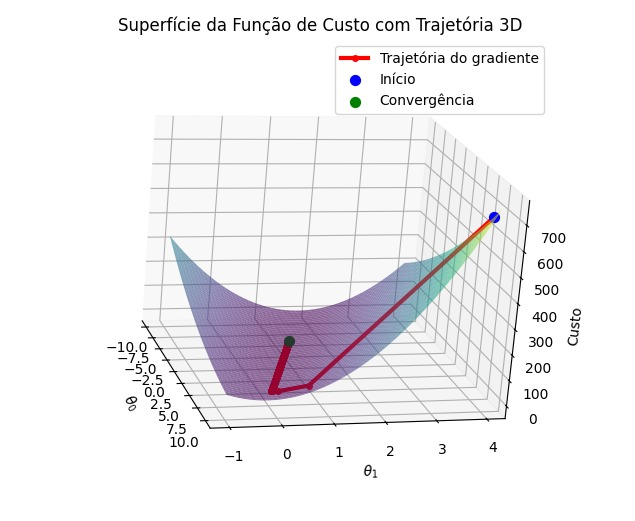


O gráfico exibido retrata o resultado do ajuste de um modelo de regressão linear simples aplicado a um conjunto de dados reais, em que se busca estimar o lucro em função da população de cidades. No eixo horizontal está representada a população das cidades, expressa em dezenas de mil habitantes, enquanto o eixo vertical traz o lucro, também escalonado em dezenas de mil dólares. Cada marcação em vermelho, com o símbolo de “x”, corresponde a uma cidade do conjunto de treinamento, permitindo visualizar a dispersão dos dados e identificar padrões ou tendências.

A presença da linha azul contínua representa o modelo de regressão linear ajustado sobre esses dados, cuja equação foi estimada pelo algoritmo de descida do gradiente. Essa linha é traçada de maneira a minimizar os erros quadráticos médios entre os valores previstos pelo modelo e os valores reais de lucro, buscando um equilíbrio que explique, da melhor forma possível, a relação existente entre as duas variáveis. Visualmente, pode-se notar que a linha passa pelo centro da nuvem de pontos, sugerindo que o modelo conseguiu capturar a tendência principal de crescimento do lucro conforme aumenta a população.

A disposição dos pontos em torno da linha evidencia, ainda, que há uma relação linear positiva, embora se observe algum grau de dispersão, indicando que fatores não representados no modelo podem também exercer influência sobre o lucro. Essa dispersão é natural em dados reais e sugere que, apesar do ajuste, o modelo explica parte, mas não toda, a variabilidade presente nos resultados observados.

Esse gráfico tem papel fundamental na análise porque permite, em uma única imagem, compreender a qualidade do ajuste realizado, verificar a pertinência do modelo linear para o problema em questão e identificar possíveis padrões, desvios ou outliers. A partir dessa visualização, é possível tanto comunicar resultados de maneira intuitiva quanto subsidiar decisões sobre a necessidade de aprimoramento do modelo ou de investigações adicionais sobre os dados.



Essa visualização tridimensional representa a superfície da função de custo *J(θ)J(\theta)*J(θ), que é utilizada para avaliar o desempenho de um modelo de regressão linear durante o processo de aprendizagem. Cada ponto dessa superfície é definido por um par de parâmetros *θ0\theta\_0*θ0 e *θ1\theta\_1*θ1 , que são os coeficientes do modelo linear. A altura da superfície em cada ponto, dada pelo eixo vertical, corresponde ao valor do custo associado àquele par de parâmetros. Quanto maior o valor, pior o desempenho do modelo com aquela configuração de parâmetros; quanto menor, mais próximo da solução ótima.

O gráfico tem como objetivo ilustrar como o algoritmo de descida do gradiente percorre essa superfície em busca do ponto de mínimo. O caminho percorrido é evidenciado pela linha vermelha com marcadores pretos, que representa a trajetória da atualização dos parâmetros ao longo das iterações. Essa linha começa no ponto azul, que marca a inicialização dos pesos, ou seja, o ponto de partida do modelo antes de qualquer ajuste. A trajetória avança passo a passo em direção à parte mais baixa da superfície, onde se encontra o ponto verde, que indica a convergência, ou seja, o ponto onde os valores de *θ\theta*θ geram o menor custo possível para o modelo.

A forma da superfície evidencia que a função de custo é convexa, o que significa que ela possui um único mínimo global. Essa característica garante que, com uma taxa de aprendizado adequada, o gradiente descendente será capaz de encontrar esse ponto ótimo sem ficar preso em mínimos locais. A curvatura da superfície e o formato da trajetória também revelam a sensibilidade da função de custo em relação a cada parâmetro: em algumas direções, o custo varia mais rapidamente do que em outras, o que influencia a velocidade com que cada parâmetro é ajustado.

Esse tipo de gráfico é extremamente útil para compreender, de forma visual, como ocorre o processo de otimização em aprendizado de máquina. Ele permite observar se o caminho seguido pelo gradiente descendente é eficiente, se a taxa de aprendizado está adequada (evitando saltos ou convergência lenta) e se a inicialização dos parâmetros influencia o trajeto até a convergência. A imagem também reforça o conceito de que o objetivo do treinamento é encontrar o ponto mais baixo da superfície, onde o custo do modelo é mínimo e, portanto, suas predições são mais precisas.

## **5 Referências**

## SEABORN. Seaborn: statistical data visualization. Disponível em: <https://seaborn.pydata.org/>. Acesso em: [18/02/25].

## CHOLET, F. Keras: The Python Deep Learning library. 2015. Disponível em: https://keras.io. Acesso em: [18/02/25].

PEDREGOSA, F. et al. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011. Disponível em:<https://jmlr.org/papers/v12/pedregosa11a.html>. Acesso em: [18/02/25].

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. Cambridge: MIT Press, 2016.

RASCHKA, S.; MIRJALILI, V. **Python Machine Learning**. 3. ed. Birmingham: Packt Publishing, 2019.

HAYKIN, S. **Redes neurais: princípios e prática**. 2. ed. Porto Alegre: Bookman, 2001.

**CHOLLET, F.** *Deep Learning with Python.* 1. ed. Shelter Island: Manning Publications Co., 2018.

**DUA, D.; GRAFF, C.** UCI Machine Learning Repository. 2019. Disponível em:<http://archive.ics.uci.edu/ml>. Acesso em: 20 fev. 2025.

**FISHER, R. A.** The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, v. 7, n. 2, p. 179-188, 1936.

**HAYKIN, S.** *Neural networks: a comprehensive foundation.* 2. ed. Upper Saddle River, NJ: Prentice-Hall, 1999.

**LECUN, Y.; BENGIO, Y.; HINTON, G.** Deep learning. *Nature*, v. 521, n. 7553, p. 436-444, 2015.

**MCCULLOCH, W. S.; PITTS, W.** A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, v. 5, n. 4, p. 115-133, 1943.

**NIELSEN, M.** *Neural Networks and Deep Learning.* Determination Press, 2015.

**RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J.** Learning representations by back-propagating errors. *Nature*, v. 323, n. 6088, p. 533-536, 1986.

**SLIDES-11**. *Redes neurais p1: Inspiração biológica e aplicações.* Material de aula, 2024.

**SLIDES-12**. *Redes neurais p2: Conceitos.* Material de aula, 2024.