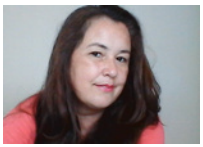
SEVEN
PUBLICAÇÕES ACADÊMICAS
2025

INTELIGÊNCIA ARTIFICIAL NO ENSINO MÉDIO

DESENVOLVENDO O PENSAMENTO COMPUTACIONAL ATRAVÉS DA REDE NEURAL PERCEPTRON

INÊS GUADALUPE, JUAN CARLOS Z. AGUILAR

AUTORES



Inês Guadalupe

Graduada em Ciências Econômicas pela Universidade Federal de São João del-Rei (UFSJ, 2001), licenciada em Matemática pela UNIUBE (2013) e em Pedagogia pela UNINTER (2018), Mestre em Matemática pelo PROFMAT/UFSJ. Atua como professora de Matemática desde 2010, dedicando-se à educação.
Seu interesse pela Inteligência Artificial surgiu durante sua dissertação de mestrado, levando-a a explorar seu potencial como ferramenta educacional. Com uma abordagem didática e acessível, busca aproximar a IA do ensino, ajudando educadores a integrá-la de forma prática e significativa.

Juan Carlos Z. Aguilar

Graduado em Ciências Matemáticas pela Universidade Nacional de Trujillo - Perú (2000), Mestre em Engenharia Elétrica pela Escola Politécnica da Universidade de São Paulo (2004), Doutor em Ciências na área de Matemática Aplicada pelo Instituto de Matemática e Estatística da Universidade de São Paulo (2009). Atualmente é professor efetivo da Universidade Federal de São João del-Rei (UFSJ). Tem experiência em Matemática Aplicada com ênfase em Métodos Numéricos para Equações Diferenciais e na área de Ensino tem interesse pela Inserção do Pensamento Computacional no Ensino Básico através da integração entre a Modelagem Matemática, Metodologias Ativas de Aprendizagem, softwares educacionais e linguagens de programação.

APRESENTAÇÃO

Querido(a) Professor(a),

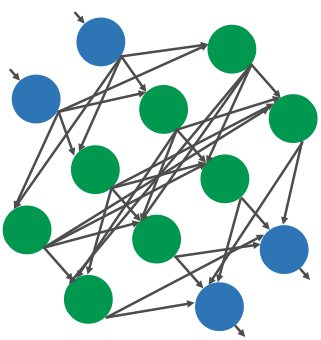
Este manual apresenta uma sequência didática para a construção de uma rede neural perceptron de camada única no Ensino Médio, integrando modelagem matemática e pensamento computacional. O material foi elaborado para auxiliar na organização das aulas nos Itinerários Formativos do Novo Ensino Médio, podendo ser adaptado a diferentes séries de ensino e contextos educacionais.

A proposta promove uma abordagem interdisciplinar, conectando conceitos de Matemática e Inteligência Artificial, favorecendo um aprendizado mais significativo.

Espera-se que este material contribua para uma aplicação dinâmica do conteúdo, ampliando a compreensão dos estudantes sobre a relação entre Matemática, Computação e Inteligência Artificial.

Aqui, você encontrará informações e orientações essenciais para a implementação do projeto, mas sinta-se à vontade para adaptá-lo conforme as necessidades e desafios da sua realidade.

Boas aulas!



SUMÁRIO

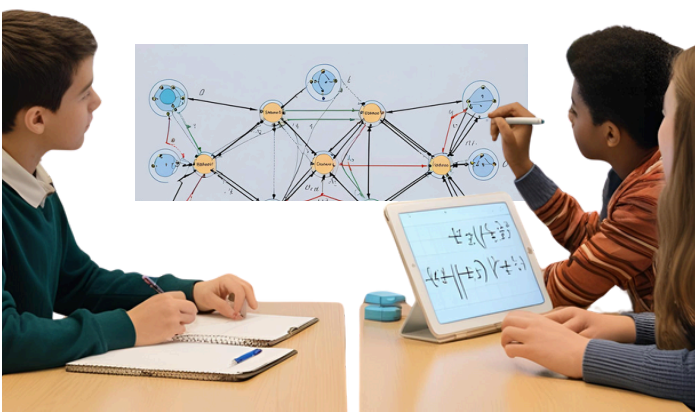
SUMÁRIO

INTRODUÇÃO

O ensino de Matemática deve desenvolver o raciocínio lógico, a resolução de problemas e a autonomia intelectual, conforme orienta a Base Nacional Comum Curricular (BNCC). Para isso, metodologias como a modelagem matemática (MM), o pensamento computacional (PCO) e a inteligência artificial (IA) favorecem a aprendizagem significativa e contextualizada.

Este e-Book apresenta uma sequência didática para o Itinerário Formativo de Tecnologia e Inovação no Novo Ensino Médio, utilizando redes neurais artificiais (RNA), com foco no modelo perceptron, que permite um ensino estruturado e acessível, favorecendo o entendimento dos princípios matemáticos da IA.

Destinado a professores, o material busca tornar o ensino mais dinâmico e alinhado às demandas contemporâneas, preparando os estudantes para desafios acadêmicos e profissionais.



METODOLOGIA

Este material foi desenvolvido e organizado para aplicação dentro da carga horária de uma disciplina dos Itinerários Formativos. No entanto, pode ser adaptado conforme a necessidade e a abordagem do professor.

O projeto segue uma sequência didática e foi pensado para, no mínimo, 10 aulas de 50 minutos. Vídeos e playlists do YouTube auxiliarão na compreensão de temas complexos; caso algum não esteja disponível, recomenda-se buscar conteúdos similares na plataforma.

Os materiais e recursos utilizados foram:

- Data show
- Notebook
- Quadro e caneta
- Caderno para anotações
- Laboratório de informática
- Internet (para pesquisas, simulações e execução de códigos)

- Google Colab: ambiente de programação online que permite a execução de código em Python sem necessidade de instalação.
- Linguagem Python: facilita a implementação das redes neurais artificiais de forma didática e acessível.
- Gráficos e diagramas: auxiliam na visualização dos conceitos de RNA, funções de ativação e aprendizado de máquina.
- Slides e materiais de apoio: facilitam a abstração de conceitos complexos.

1ª AULA

Motivação do Tema

✅ Introdução à IA

- Apresentar a sequência didática e seus objetivos, destacando o de facilitar a compreensão da IA a partir de uma perspectiva matemática.
- Expor um fluxograma com a proposta para cada uma das próximas aulas . Veja um exemplo no Apêndice A.

✅Preparativos

- Distribuir kits para anotações e incentivar a documentação reflexiva.
- Formar grupos colaborativos para as atividades propostas em algumas aulas seguintes.

✅Sondagem de Conhecimento

- Aplicar um questionário via *Google Forms* para avaliar o entendimento inicial dos estudantes sobre IA , como proposto no Apêndice B.

✅Atividade para a Próxima Aula

- Solicitar aos alunos que tragam notícias recentes sobre IA para discussão.

✅Fechamento

- Exibir um vídeo curto sobre IA para reflexão final.

📌Sugestão: **Link**

Tempo utilizado:
1 aula de 50 minutos

2ª AULA

O que sabemos sobre IA?

✅ Reflexão sobre a IA e sua Relevância na Sociedade

- 🔹 Analisar as notícias trazidas pelos alunos, orientando-os sobre a distinção entre informação valiosa, sensacionalismo e preconceito.
- 🔹 Enfatizar a importância de entender como a IA molda nosso presente e futuro.

✅ Análise de Conhecimento Prévio

- 🔹 Apresentar e discutir as respostas da pesquisa realizada via *Google Forms* (Apêndice B) por meio de gráficos.
- 🔹 Estimular reflexões sobre as percepções iniciais dos alunos sobre IA e RNA.

✅ Campos de Estudo da IA

- 🔹 Apresentar os principais campos de estudo da IA.

📌 Sugestão: Para se informar melhor sobre este assunto, assista: **Link**

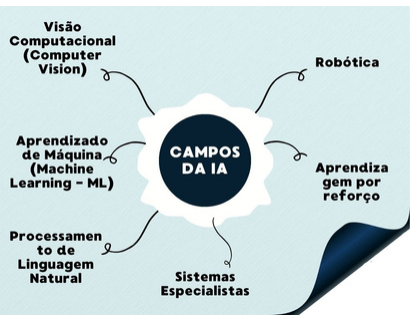
✅ Histórico e Definição de RNA

- 🔹 Explicar a definição de RNA como sistemas computacionais inspirados no cérebro humano.
- 🔹 Relacionar as origens da IA com marcos históricos significativos. Assista previamente: **Link**

✅ Fechamento

- 🔹 Exibir um vídeo de 15 minutos sobre a história da IA: **Link**

Tempo utilizado:
1 aula de 50 minutos

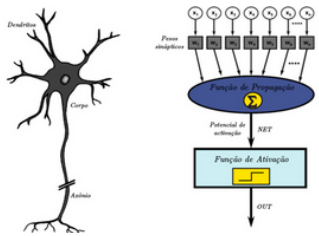


3ª AULA

Abordagem geral

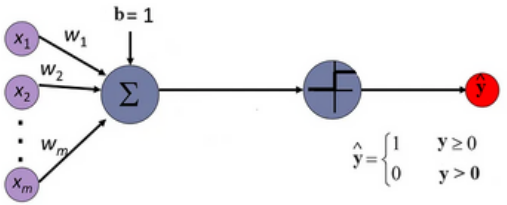
✅ Explicar a origem do neurônio artificial modelado a partir do neurônio humano. Utilize as Figuras 1 e 2.

Figura 1: Neurônio Humano x Neurônio Artificial



Fonte: Adaptado de Naranjo (2014), p.68

Figura 2: Representação de um Perceptron



Disponível em: https://www.deeplearningbook.com.br/funcao-de-ativacao/

✅ Apresentar o mapa mental disponível no Apêndice C, que resume os principais componentes das redes neurais, como entradas, pesos sinápticos, bias, funções de ativação e o processo de treinamento. Para aprofundar o entendimento sobre o tema, assistir à seguinte playlist:

Link para a playlist

✅ Explicar a importância das funções de ativação em RNA, com foco na continuidade e suavidade para um aprendizado eficiente: funções contínuas e suaves, promovem melhor generalização do modelo, tornando o treinamento mais eficiente e confiável.

✅ Utilizar recursos visuais, com apoio de retroprojetor, para ilustrar diferentes tipos de funções (Veja o Apêndice D).

- Função Definida e Suave: Exemplo da função y=sin(x),
- Função com Descontinuidade Brusca. Exemplo: y = 1/x-1
- Função com Pontos Singulares: Exemplo de x^(1/3).

✅Segundo Maria Inês Furtado, para que uma função matemática possa ser utilizada como função de ativação em uma rede neural artificial, é necessário que duas condições sejam satisfeitas: a função deve ser contínua e os limites da função no infinito devem ser finitos (FURTADO, 2006).

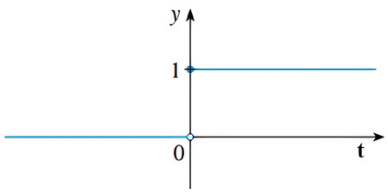
3ª AULA

(continuação)

✅ A continuidade da função assegura que pequenas variações nas entradas resultem em pequenas variações nas saídas, o que é essencial para a estabilidade e a convergência dos algoritmos de treinamento. Além disso, a exigência de limites finitos evita que os valores das ativações se tornem infinitos, o que poderia causar instabilidades numéricas e comprometer o treinamento da rede neural

✅ Introduzir a função degrau (Step Function) como uma função binária usada no perceptron, com um gráfico de sua descontinuidade (Figura 6).

Figura 6: Função Degrau.

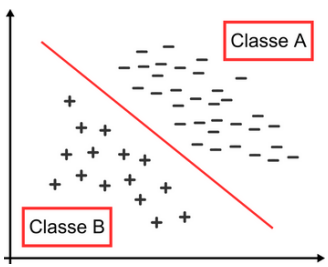


Fonte: Elaborada pela autora.

✅Destacar suas limitações em redes neurais complexas devido à descontinuidade.

A função degrau, apesar de sua descontinuidade, é apropriada para o perceptron que opera como um classificador linear, ativando ou não um neurônio com base em um valor pré-determinado que define a fronteira entre as classes (limiar fixo). Uma classificação linear de dados é uma separação de saídas através de uma reta, como pode ser visualizado na Figura 3.

Figura 3: Dados Linearmente Separáveis

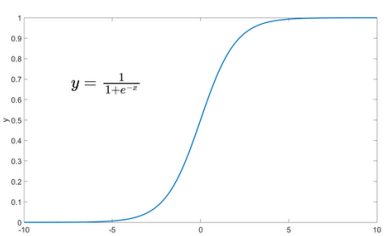


Fonte: Adaptado de FURTADO (2019), p. 48

✅ Apresentar a função sigmoide através da Figura 4, como exemplo de função suave e contínua.

✅ Discutir seu comportamento para valores do domínio (grandes e pequenos), e como ela transforma entradas em valores entre 0 e 1.

Figura 4: Função Sigmoide.



Fonte: Elaborada pela autora.

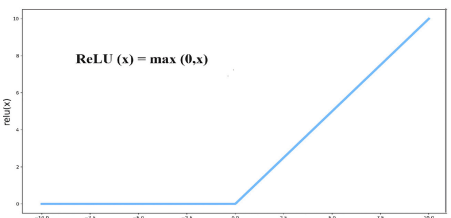
3ª AULA

(continuação)

✅ Explicar a importância da função sigmoide no contexto de redes neurais e o impacto de seu formato suave no aprendizado: Pode-se pensar na função sigmoide como uma rampa muito suave. Ela converte qualquer valor de entrada em um número entre 0 e 1; é suave e contínua. Ao tentar ensinar a rede neural a fazer algo complicado, essa rampa suave faz com que os ajustes que a rede precisa fazer fiquem cada vez menores à medida que ela aprende, como ao subir uma colina cada vez mais rasa. Isso pode tornar o aprendizado muito lento, especialmente quando a rede é muito profunda ou seja, quando possui muitas camadas ocultas entre a camada de entrada e a camada de saída.

✅ Apresentar a função ReLU (Rectified Linear Unit):
ReLU(x) = max (0,x); através do gráfico da Figura 5 para ilustrar seu comportamento.

Figura 5: Função ReLU.



Fonte: Elaborada pela autora.

✅ Explicar como a ReLU facilita o aprendizado rápido e eficiente, especialmente em redes profundas: a função ReLU, é como uma escada com degraus altos. Parece uma linha reta que é zero para valores negativos e igual ao valor de entrada para valores positivos. Cada passo é grande e direto, o que ajuda a rede a aprender mais rápido e de forma mais eficiente. Assim, a rede consegue fazer ajustes maiores e mais eficazes em menos tempo. É uma função simples e eficiente; muito usada em redes neurais modernas.

✅ A escolha da função adequada impacta diretamente no desempenho da rede e na qualidade do aprendizado.

✅ As funções de ativação são fundamentais para as RNA, pois introduzem não linearidade, permitindo que a rede aprenda padrões complexos.

✅ Explorar como as funções sigmoide, ReLU e degrau afetam o desempenho e o aprendizado das redes.

3ª AULA

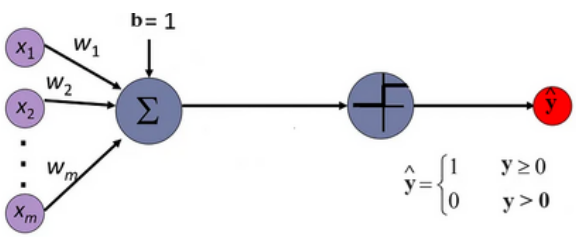
(continuação)

✅ As funções de ativação em uma rede perceptron determinam se um neurônio será ativado com base na soma ponderada das entradas. Aqui está um resumo de como cada uma atua:

- **Degrau:** Ativa o neurônio com 0 ou 1, dependendo de um limiar. Funciona bem para problemas simples, mas não permite ajustes graduais no aprendizado.
- **ReLU:** Retorna 0 para valores negativos e mantém valores positivos inalterados. A ReLU é eficiente para redes profundas, mas seu uso em um perceptron de duas camadas não traz benefícios significativos, pois não há ativação não linear.
- **Sigmoide:** Transforma os valores em um intervalo entre 0 e 1, facilitando a interpretação probabilística. Porém, pode tornar o treinamento mais lento devido à saturação dos valores extremos.

✅ Apresentar novamente a Figura 2, observando os principais componentes das redes neurais, como entradas, pesos sinápticos, bias e o processo de treinamento, enfatizando a função degrau como função de ativação.

Figura 2: Representação de um Perceptron



Fonte: Adaptado de Naranjo (2014), p.68

Concluir a aula com a proposta de uma tarefa de avaliação sobre funções de ativação, a ser realizada em grupos e com material impresso disponível (Apêndice E).

Tempo utilizado:
2 aulas de 50 minutos

4ª AULA

É Hora de Falar de Lógica Proposicional e Tomada de Decisão!!!

✅ Defina: A lógica proposicional é um ramo da matemática que estuda sentenças que podem ser classificadas como verdadeiras (*V*) ou falsas (*F*). Ela é amplamente utilizada na eletrônica, na computação e na IA para modelar decisões lógicas. Revise este assunto em: Link

✅ Utilizamos operadores lógicos para combinar proposições e determinar o resultado de uma expressão. Os principais são:

🔹 **Conjunção (AND - ∧-)**:
A expressão "$P \wedge Q$" só é verdadeira quando ambas as proposições são verdadeiras, como se vê na Tabela 1.

Tabela 1: Conectivo Lógico AND (∧)

P	Q	P ∧ Q
V	V	V
V	F	F
F	V	F
F	F	F

Fonte: Elaborada pela autora.

🔹 **Disjunção (OR - ∨ -):**
A expressão "$P \vee Q$" é verdadeira se pelo menos uma das proposições for verdadeira, como se vê na Tabela 2.

Tabela 2: Conectivo Lógico OR (∨)

P	Q	P ∨ Q
V	V	V
V	F	V
F	V	V
F	F	F

Fonte: Elaborada pela autora.

🔹 **Negação (NOT - ¬ -)**:
A negação "$\neg P$" inverte o valor de verdade de uma proposição.

Tabela 3: NÃO Lógico (¬)

R	¬ R
V	F
F	V

Fonte: Elaborada pela autora.

🔹 **Condicional (IMPLICAÇÃO - → -):** "$P \rightarrow Q$" significa que, se P for verdadeiro, então Q também deve ser verdadeiro.

🔹 **Bicondicional (BIIMPLICAÇÂO - ↔ -):** "$P \leftrightarrow Q$" é verdadeiro quando P e Q possuem o mesmo valor de verdade.

4ª AULA

(continuação)

✅Demonstre como a lógica proposicional é aplicada no cotidiano ou na computação.

Exemplo: "Se um estudante entrega a atividade e acerta mais de 60% das questões, então ele é aprovado."

Isso pode ser modelado como:

- P: "O estudante entrega a atividade"
- Q: "O estudante acerta mais de 60%"
- R: "O estudante é aprovado"
- Expressão lógica: $(P \land Q) \rightarrow R$

Prepare outros exemplos simples como esse e use a tabela - verdade para explicá-los.

É fundamental que o professor auxilie os alunos na compreensão das possíveis aplicações dos conectivos lógicos, pois essa etapa envolve a base do raciocínio lógico.

Tabela 4: Conectivos Lógicos-Símbolos, Significados e Representações Visuais

Conectivo	Símbolo	Significado	Conjunto	Diagrama
e (and) - conjunção	$\land$	Será verdadeira somente quando todas as proposições forem verdadeiras	interseção	
ou (or) - disjunção	$\lor$	Será verdadeira quando pelo menos uma das proposições forem verdadeiras	união	
se...então (if...then) - implicação	$\rightarrow$	Será falsa quando a proposição antecedente for verdadeira e a consequente for falsa.	inclusão	
se, e somente se (if and only if) - bicondicional	$\leftrightarrow$	Verdadeiro quando ambos são verdadeiros ou ambos são falsos.	igualdade	
ou...ou (xor) - disjunção exclusiva	$\veebar$	Conecta duas proposições onde apenas uma delas deve ser verdadeira, não ambas.	diferença	
não - negação	$\neg$	Terá valor falso quando a proposição for verdadeira e vice-versa.	-	

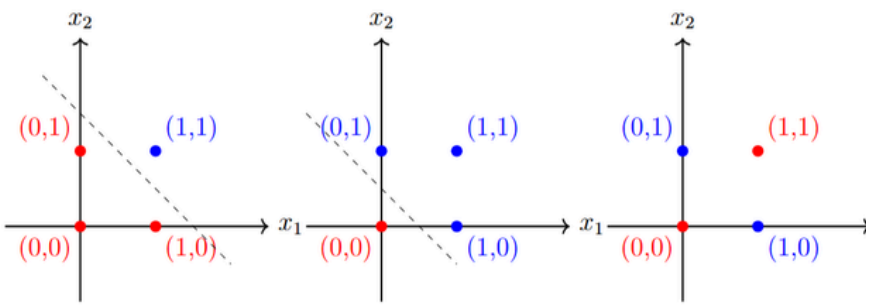
4ª AULA

(continuação)

✅ Utilize a Tabela 4 para mostrar as relações entre os conectivos lógicos, explicando suas propriedades com exemplos simples. Demonstre, por exemplo, como a conjunção (E) exige ambas as proposições verdadeiras, enquanto a disjunção (OU) é verdadeira se pelo menos uma for. Isso tornará a lógica proposicional mais clara e acessível.

✅ Enfatize, através da Figura 4, a separabilidade dos dados para os conectivos lógicos AND e OR e a não separabilidade para o conectivo XOR. Nestes casos, os valores numéricos 0 e 1 substituem atributos de verdade *F* e *V*. A coloração azul representa o valor verdade atribuído à tabela verdade dos conectivos AND, OR e XOR, respectivamente:

Figura 4: Separabilidade dos Conectivos AND, OR e XOR



Fonte: Elaborada pela autora.

✅Como os dados, para o caso AND, são linearmente separáveis, é possível a construção da RNA perceptron, o modelo mais simples de neurônio.

✅Relembre-os, contudo, que as RNA são geralmente usadas para tarefas mais complexas, como reconhecimento de padrões em imagens, processamento de linguagem natural, previsão, entre outros.

Tempo utilizado:
1 aula de 50 minutos

5ª AULA

CHEGOU A HORA DE CONSTRUIRMOS JUNTOS UM PERCEPTRON !!!

✅ Dedique-se então à elaboração manual (usando quadro e giz/caneta) dos algoritmos de uma rede perceptron.

Essa etapa é importante para o aluno pois precisam identificar um padrão de repetição no processo de construção da rede neural perceptron que vai lhes permitir perceber a lógica por trás do funcionamento da rede e programá-la, posteriormente.

✅ Nesta aula, o objetivo é entender como construir um algoritmo de uma RNA perceptron para simular o conector lógico AND. Assista: **Link**

✅ Crie um modelo matemático (MMO) para o conectivo lógico AND utilizando o perceptron AND.

Figura 5: Diagrama de fluxo de um modelo matemático.

Fonte: Elaborada pela autora.

✅ Coloque o problema de encontrar os valores verdade do conectivo lógico AND da seguinte maneira: dadas as proposições P e Q, busque criar um MMO para encontrar os valores verdade da sentença $P \wedge Q$, ou seja, o modelo tem que fazer a leitura dos valores de $P \wedge Q$ (valores de entrada) e encontrar como resposta os valores de saída $P \wedge Q$, tal como mostra a Tabela 1.

Tabela 1: Conectivo Lógico AND

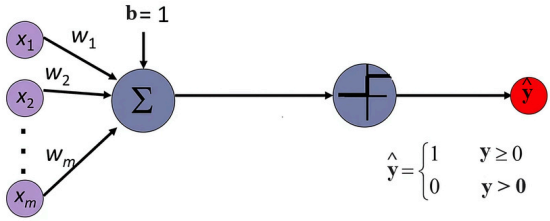
P	Q	P ∧ Q
V	V	V
V	F	F
F	V	F
F	F	F

Fonte: Elaborada pela autora.

5ª AULA

(continuação)

✅ Para elaborar o modelo matemático, deve-se observar novamente a figura do perceptron, Figura 1, e indicar o caminho de elaboração deste modelo.



✅ Primeiro faça a identificação das variáveis:

- x_1 = valor de verdade da proposição *P*.
- x_2 = valor de verdade da proposição *Q*.

✅ Como, no perceptron, as variáveis representam números, utilize a seguinte codificação: *V* = 1 e *F* = 0.

✅ Considerando todos os possíveis pares de valores de verdade para *P* e *Q*, tem-se: *(P, Q)* → *P* ∧ *Q*
(V, V) → *V,*
(V, F) → *F,*
(F, V) → *F,*
(F, F) → *F.*

✅ Aplicando a codificação definida, obtém-se os seguintes valores para as variáveis:
(1, 1) → 1, (1, 0) → 0, (0, 1) → 0, (0, 0) → 0

✅ O perceptron também nos oferece a possibilidade de que as entradas tenham pesos w_1, w_2 (sendo os subíndices associados às variáveis) além do bias (*b*).

✅ Os pesos iniciais podem ser zero e o bias diferente de zero, formando a seguinte soma: $y = x_1 w_1 + x_2 w_2 + b$

5ª AULA

(continuação)

✅Relembre aqui que os pesos ajustam a contribuição de cada entrada para o resultado final.

✅O bias (viés), por sua vez, é um valor constante adicionado ao resultado ponderado das entradas. Ele permite que a função de ativação seja deslocada ao longo do eixo y. Isso significa que ele ajusta a saída do modelo, mesmo quando todas as entradas são $x_1 = x_2 = ... = x_n = 0$.

✅ No perceptron, aplica-se a função de ativação *f* ao valor calculado, como, por exemplo, a função degrau, e que pode ser definida, considerando-se uma soma ponderada das entradas com um viés:

$$y = \begin{cases} 1, & x_1w_1 + x_2w_2 + b \geq 0 \\ 0, & x_1w_1 + x_2w_2 + b < 0 \end{cases}$$

✅Observe que as saídas de *f* são sempre 0 ou 1, ou seja, *F* ou *V*.

✅Caso o valor de saída seja o mesmo do valor de $P \wedge Q$, para a entrada correspondente, afirme que o modelo matemático, inspirado na rede perceptron, foi bem-sucedido e terá então um MMO que resolve o problema proposto.

✅Passe então aos primeiros cálculos:

Considere o par de entrada (0, 0), ou seja, (*F, F*). Considere também os pesos zero e o bias (viés) -1, tem-se então:

$x_1 = 0, x_2 = 0, w_1 = 0, w_2 = 0$ e $b = -1$

✅ O valor -1 para o viés indica que o perceptron tende a prever 0 quando todas as entradas são zero, o que é adequado para a função lógica AND, onde o resultado é zero apenas quando ambas as entradas são zero.

Forme a soma: $y = x_1w_1 + x_2w_2 + b = 0 \cdot 0 + 0 \cdot 0 - 1 = -1$

5ª AULA

(continuação)

✅ Agora aplique a função de ativação *f* (função degrau). Como y = −1 < 0, então y' = *f* (−1) = 0.

✅ Sabe-se que para a entrada (0, 0) é esperado o valor de saída 0 (pois (*F*, *F*) → *F*, logo o modelo foi bem-sucedido para essa entrada.

Agora, a questão é:
para as outras entradas, o modelo funciona?

✅Para comprovar isso, pegue uma segunda entrada, por exemplo, (1, 1) e o submeta ao modelo.
Nesse caso: $x_1 = 1, x_2 = 1, x_3 = 1, w_1 = 0, w_2 = 0$ e $b = -1$

✅ Forme a soma: $y = x_1 w_1 + x_2 w_2 + b = 1 \cdot 0 + 1 \cdot 0 - 1 = -1$
Agora aplique a função degrau $y' = f(-1) = 0 \rightarrow F$.

✅ Porém esperava-se que a saída fosse 1, pois (*V*), → *V*. Comprovando que, para essa entrada o modelo proposto não funciona.

O que se deve fazer?

Que tal atualizar os pesos e o bias? Mas, como fazer isso?

Em 1959, Bernard Widrow e Marcian Hoff desenvolveram os algoritmos ADALINE e MADALINE, além da Regra Delta. O ADALINE previa padrões binários, enquanto o MADALINE foi a primeira RNA aplicada a um problema real. A Regra Delta ajusta os pesos das conexões neurais para minimizar o erro entre a saída esperada e a real.

Enquanto o erro é diferente de 0, atualiza-se os pesos e do bias da seguinte forma:

$$\begin{cases} w_{ji} = w_{(j-1)i} + x_i \cdot \alpha \cdot E \\ b_j = b_{j-1} + \alpha \cdot E \end{cases}$$, onde:

w_{ji} é o peso atualizado,
w_{j-i} é o peso anterior,
x_i é valor da entrada,
α é a taxa de aprendizado,
E é o erro calculado ,
b_j é o novo bias e
b_{j-i} é o bias anterior.

✅**Refazer os cálculos com as entradas até que o erro seja zero para todas as amostras de treinamento. No caso descrito, foram necessárias apenas duas iterações até que a rede atingisse o equilíbrio, ou seja, os pesos e bias estabilizaram, permitindo que o perceptron classificasse corretamente todas as entradas. Para detalhes desse processo, consulte o Apêndice G.**

(continuação)

✅Encontrando o modelo final da soma:

$$y = x_1 \cdot 0,4 + x_2 \cdot 0,4 - 0,6$$

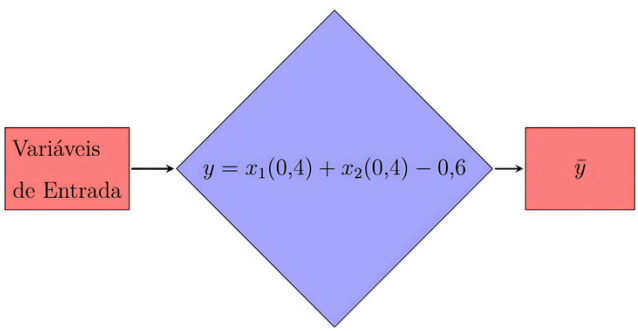
Como apresentado na Figura 6, observe que o algoritmo matemático possibilita:

🔹 obter um modelo matemático para resolver o problema proposto.

🔹 perceber a realização de uma concatenação de passos (fluxo) até atingir uma solução satisfatória.

🔹 Detectar uma padronização de procedimentos (repetição), fundamental para a organização do pensamento computacional.

Figura 6: Diagrama de fluxo MMO.



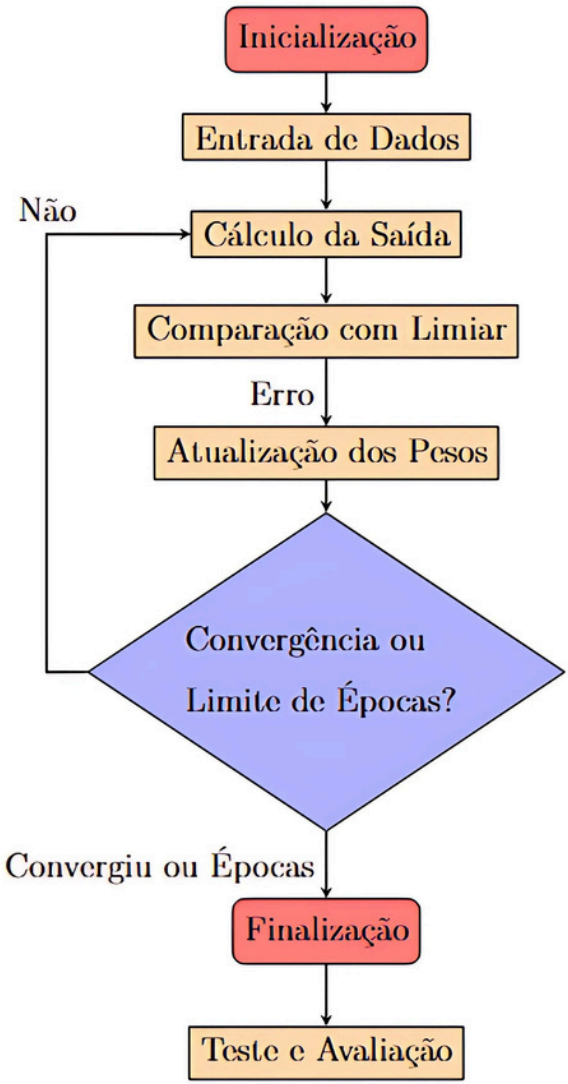
Fonte: Elaborada pela autora.

✅ A identificação de padrões nos cálculos facilita a sua automatização, tornando o processo mais eficiente e confiável quando realizado pelo computador. Dessa forma, é possível minimizar erros manuais e otimizar o tempo gasto na resolução do problema.

✅ Diante dessas observações, elabore com os estudantes um fluxograma para visualizar de forma clara e estruturada a concatenação de passos no computador. O fluxograma, a exemplo da Figura 7, será um guia visual para compreender a sequência lógica das operações, destacando os pontos-chave do processo e auxiliando na organização do raciocínio algorítmico.

(continuação)

Figura 7: Fluxograma de uma Rede Peceptron



Fonte: Elaborada pela autora.

5ª AULA

(continuação)

✅ Reforce com os estudantes que: o primeiro passo na construção da rede perceptron ocorre antes da entrada de dados: a inicialização dos pesos das conexões entre os neurônios e a definição dos hiperparâmetros, como a taxa de aprendizado (α) e o número de épocas. Os pesos das variáveis de entrada e o bias (b) para a construção da rede perceptron AND, wi, b e α, são definidos aleatoriamente.

✅ As entradas são apresentadas um par de cada vez: (0,0), (0,1), (1,0), (1,1), e o perceptron tenta aprender a mapear essas entradas para as saídas desejadas (0 ou 1), que representam o resultado da operação AND. A saída y é o valor da função AND aplicada a cada combinação de entrada.

✅ O objetivo é que o perceptron aprenda a produzir corretamente esses resultados.

✅ Utiliza-se a função linear para o cálculo da saída, passando-a em seguida pela função de ativação, que gera a saída binária. Como discutido na terceira aula, as funções de ativação são fundamentais para introduzir não linearidades nas saídas dos neurônios.

✅ Após a propagação para frente, o resultado da rede é comparado com o valor esperado, e o erro é calculado usando uma função de perda que mede a diferença entre o resultado da rede e o valor desejado. O processo de ajuste dos pesos e bias com base nos dados de treinamento é o que caracteriza o treinamento da rede, com o algoritmo de otimização minimizando a função de perda.

✅ A rede neural passa pelos dados várias vezes durante o treinamento, em ciclos chamados de épocas. Após várias épocas, os pesos são ajustados de forma a fornecer as respostas corretas para todas as entradas da operação AND. Esse processo torna a escolha inicial dos pesos e da taxa de aprendizado determinantes para o sucesso do treinamento.

5ª AULA

(continuação)

✅ Por último, testa-se o modelo treinado com dados independentes para avaliar seu desempenho e generalização.

✅ O objetivo ao apresentar esse fluxograma é estimular o desenvolvimento do PCO, que envolve a compreensão e aplicação de conceitos e processos computacionais para a resolução de problemas.

✅Durante a aula, revise os fundamentos de entradas, pesos e saídas, utilizando quadro e canetas coloridas para ilustrar a função soma e a função de ativação em diversas iterações. Para cada nova entrada, utilize uma cor diferente, facilitando a visualização dos padrões.

✅Após várias iterações, os pesos ideais que fornecem o melhor resultado foram determinados como w_i = 0,4, b =−0,6 e taxa de aprendizado α = 0,4.

✅O conjunto inicial permitiu a convergência da rede em apenas duas iterações, mas, com outros parâmetros, o processo poderia se prolongar até a estabilização dos pesos

Finalize a aula apresentando os próximos exercícios que compõem a atividade avaliativa desta sequência didática, proposto no Apêndice F

Tempo utilizado:
1 aula de 50 minutos

6ª AULA

VALE RESSALTAR:

A compreensão dos conceitos básicos de programação é fundamental para que os alunos possam, posteriormente, entender a implementação da rede neural perceptron.

A ferramenta computacional sugerida nessa etapa para ser utilizada pelos alunos, no laboratório de informática, é o Google Colab.

✅Nesta aula, aborde os fundamentos da programação em Python no Google Colab, plataforma que permite executar código diretamente no navegador.

✅Python é uma linguagem poderosa devido às suas bibliotecas, que simplificam tarefas complexas com poucas linhas de código.

✅Principais Conceitos:

- Comentários: Começam com # e são usados para documentar o código tornando-o mais compreensível sem interferir na execução dos comandos.
- Indentação: Define blocos de código e é obrigatória em Python.
- Strings: Representam textos exibidos como resultado após a execução do código e são delimitadas por aspas simples ('texto') ou duplas ("texto").
- Print: Exibe mensagens e resultados.
- Estruturas de Controle:

🔹Condicionais: if, elif, else para decisões no código.

🔹Laços: for (repetição controlada) e while (repetição condicional).

- Operações Matemáticas: Somar, dividir, multiplicar, comparar valores, etc.
- Variáveis: Armazenam dados e podem ser de vários tipos, como números inteiros, decimais e textos.

  Exemplo: idade = 25 altura = 1.75 nome = "João"
- Operadores:

🔹Aritméticos: +, -, *, /

🔹Comparação: ==, !=, >, <

🔹Lógicos: and, or, not

Para entender mais sobre programação em Phyton, assista Link

6ª AULA

Continuação

- Bibliotecas: Conjuntos de funções prontas para uso.
  Exemplo de instalação: !pip install numpy.

✅Esses conceitos são fundamentais para o desenvolvimento de uma rede perceptron, permitindo entender como a programação auxilia na construção de modelos de IA.

Códigos para programação das funções de ativação

🔹 Função Degrau

```
def step_function(x):
    return 1 if x >= 0 else 0
for valor in [-2, -1, 0, 1, 2]:
    resultado = step_function(valor)
    print(f'step_function({valor}) = {resultado}')
```

🔹 Função sigmoide

```
import numpy as np
def sigmoid(x):
    return 1 / (1 + np.exp(-x))
for valor in [-2, -1, 0, 1, 2]:
    resultado = sigmoid(valor)
    print(f'sigmoid({valor}) = {resultado:.2f}')
```

📌 Após executar este código, troque o comando resultado: .2f para .4f e o execute novamente, observando o resultado apresentado: agora, valores com quatro casas decimais.

🔹 Função ReLU

```
import numpy as np
def relu(x):
    return np.maximum(0, x)
test_values = [-2, -1, 0, 1, 2]
for valor in test_values:
    resultado = relu(valor)
    print(f'relu({valor}) = {resultado}')
```

Tempo utilizado:
1 aula de 50 minutos

7ª AULA

CHEGOU O MOMENTO DA PROGRAMAÇÃO DA NOSSA REDE NEURAL!!!

Objetivo da Aula

✅ Implementar uma rede perceptron simples utilizando Python.
✅ Compreender a aplicação prática do perceptron na solução de problemas lógicos.
✅ Explorar, usando programação em Python, diferentes funções de ativação e sua influência no aprendizado de máquina.

Passos para Implementação

1. Introdução ao perceptron (feitos nas aulas anteriores)

- Revisão do conceito de perceptron com duas entradas.
- Explicação sobre pesos, viés e função de ativação.
- Apresentação de um diagrama do perceptron com duas entradas.

2. Configuração do Ambiente

- Acessar o Google Colab.
- Importar a biblioteca NumPy e implementar o perceptron.

Projete o código (Apêndice H), linha por linha, indicando a função de cada comando para a programação da rede.

3. Definição da Função de Ativação (degrau)

- Explicar novamente sobre a função degrau como método para decidir a saída do perceptron: ativa o neurônio com um valor 1 se a soma ponderada das entradas for maior ou igual a um determinado limiar e 0 caso contrário, permitindo classificações lineares simples.

4. Treinamento do Perceptron

- Processo de ajuste dos pesos e viés utilizando a regra de aprendizado do perceptron: define o tamanho do ajuste dos pesos a cada iteração, influenciando a velocidade e estabilidade da convergência, enquanto o número de épocas determina quantas vezes o modelo revisita os dados para otimizar os pesos, impactando a precisão e generalização da rede.
- Importância da taxa de aprendizado e número de épocas na convergência do modelo.

7ª AULA

Continuação

Sugerimos que o professor use o retroprojetor na sala de informática para ir executando os códigos juntamente com os estudantes.

5. Definição dos Dados de Entrada e Saída (Função AND)

- Construção de uma tabela verdade para treinar o perceptron na função lógica AND.

6. Treinamento e Exibição dos Resultados

- Execução do treinamento com os parâmetros escolhidos.
- Impressão dos pesos e viés finais após o ajuste.

Resultados Esperados

- Exibição dos pesos e viés ajustados ao final das iterações.
- Compreensão sobre como a RNA ajusta os pesos para aprender a função AND.

Exploração de Outras Funções de Ativação

✅ Apresentar resultados utilizando as funções Sigmoide e ReLU (Anexos I e J).
✅ Comparar desempenho e convergência das funções.

Visualização da Separabilidade Linear

- Uso de gráficos para demonstrar como os dados podem ser separados linearmente.
- Importância da separabilidade linear para o funcionamento do perceptron.

Conclusão

✅ A rede Perceptron resolve problemas lógicos simples.
✅ O treinamento ajusta pesos e viés para minimizar o erro.
✅ Funções de ativação impactam o aprendizado.
✅ O perceptron não resolve problemas não linearmente separáveis.

Tempo utilizado:
1 aula de 50 minutos

Próxima Atividade:

📌 **Tarefa Complementar (Apêndice K)**

- **Explorar novas entradas.**
- **Implementar perceptron para outras funções lógicas.**
- **Criar um relatório com análise dos resultados.**

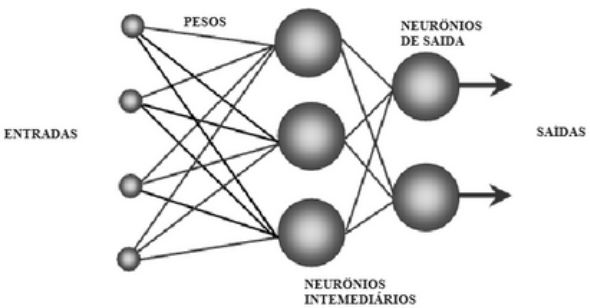
8ª AULA

Continuação

Construção de uma Rede Neural Multicamadas

Os estudantes precisam compreender que RNA multicamadas são importantes para resolver problemas mais complexos. As redes multicamadas possuem, pelo menos, uma camada intermediária de neurônios, como está ilustrado na Figura 8. Essa camada intermediária é chamada de camada oculta e é responsável por transformar as informações de entrada antes de enviá-las à camada de saída.

Figura 8: Rede Neural Multicamadas



Fonte: Furtado(2019), p.11

Objetivos da Aula

📌 Demonstrar a limitação do perceptron simples ao modelar funções não linearmente separáveis.

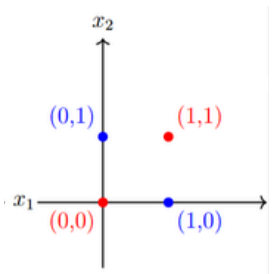
📌 Introduzir redes multicamadas como solução para esses desafios.

📌 Implementar e testar uma rede neural para a função XOR.

Exploração do Problema

🔹 Revisão das funções OR e XOR com uma explicação da tabela verdade e inseparabilidade linear do XOR: A função lógica XOR não é linearmente separável porque seus pontos formam um padrão que não pode ser dividido por uma única reta, como pode ser visualizado através da Figura 9.

Figura 9 : Inseparabilidade do conectivo xor



Fonte: Elaborada pela autora

8ª AULA

Continuação

🔹 Na tabela verdade do XOR, a saída é 1 apenas quando as entradas são diferentes, gerando um padrão em "X" no plano cartesiano. Como não há uma reta que separe corretamente os pontos das duas classes, é necessário o uso de modelos mais complexos, como redes neurais com camadas ocultas, para resolver esse problema.

🔹 Discutir sobre as limitações do perceptron simples e a necessidade de arquiteturas mais complexas.

Por que usar uma Rede Neural Multicamadas?

✅ Resolve problemas não linearmente separáveis;
✅ Extrai padrões complexos dos dados;
✅ Modela relações entre variáveis de forma mais precisa.

Implementação da Rede Neural

Programação da Rede XOR – Demonstração do código (Apêndice L).
Análise dos Resultados – Plotagem do gráfico para validar a solução.

Reflexão e Conexão com o Aprendizado

✅ Comparação com as redes já estudadas (AND e OR).
✅ Ênfase na evolução do perceptron para redes multicamadas.
✅ Demonstração da aplicação prática sem aprofundamento excessivo na matemática.

Essa abordagem permite que os alunos entendam o conceito de redes multicamadas sem se perderem em detalhes técnicos, focando na utilidade e nos resultados obtidos.

Tempo utilizado:
1 aula de 50 minutos

9ª AULA

Apresentação e Discussão dos Trabalhos

Após as atividades práticas, os estudantes apresentarão seus relatórios e compartilharão experiências, consolidando o aprendizado e reforçando os conceitos estudados.

Objetivos da Aula

📌 Verificar a compreensão do perceptron simples por meio da resolução de problemas e análise dos relatórios.

📌 Revisar conceitos-chave e esclarecer dúvidas remanescentes.

Dinâmica da Apresentação

1- Cada grupo deve entregar seu documento relatando o processo e os resultados obtidos.

- 🔹 Cálculo manual seguindo o algoritmo matemático
- 🔹 Criação e análise dos fluxogramas
- 🔹 Implementação e execução do código computacional

2- Os grupos apresentam, resumidamente, seus trabalhos, destacando os principais aspectos observados.

Uso de Recursos – Os alunos devem utilizar o projetor para demonstrar a programação desenvolvida.

Debate e Reflexão – Após as apresentações, abra um espaço para:

✅ Esclarecimento de dúvidas

✅ Complementação de informações

✅ Discussão sobre dificuldades e aprendizados.

Essa abordagem fortalece a aprendizagem colaborativa e permite que os alunos aprimorem suas habilidades analíticas e comunicativas, preparando-os para desafios mais avançados.

Tempo utilizado:
1 aula de 50 minutos

10ª AULA

É hora de Feedback !!!

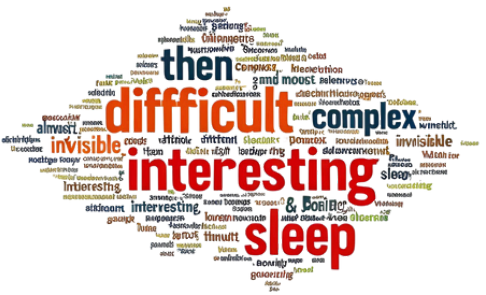
Para avaliar a percepção dos alunos sobre a sequência didática e o conteúdo abordado, realize atividades interativas:

1- **Nuvem de Palavras** – Os alunos expressarão suas opiniões de forma livre, destacando os termos que melhor representam o que aprenderam sobre IA e a sequência didática. As palavras mais mencionadas aparecerão com maior destaque, permitindo uma análise visual das tendências de resposta.

Você encontra orientações sobre como construir a nuvem de palavras em: Link

2- Elaboração de um Fluxograma no Canva – Para organizar os conceitos discutidos sobre RNA, os alunos podem criar fluxogramas que representem as principais ideias aprendidas, favorecendo a estruturação do conhecimento de forma visual.

3- Momento de discussão: conduzir uma conversa para que os alunos compartilhem suas impressões, analisando as próprias percepções e contribuindo para o aprimoramento das futuras aulas.



CONSIDERAÇÕES FINAIS

Diante da proposta apresentada, a inserção da modelagem matemática e do pensamento computacional no ensino proporciona uma abordagem dinâmica e significativa para a aprendizagem. Ao desenvolver projetos que envolvem a construção de redes neurais artificiais, os estudantes não apenas ampliam sua compreensão sobre inteligência artificial, mas também fortalecem habilidades essenciais, como resolução de problemas e raciocínio lógico.

Além disso, a aplicação prática dos conceitos permite que os alunos compreendam a importância das funções matemáticas e computacionais no mundo real, tornando o aprendizado mais envolvente e contextualizado. O uso de ferramentas tecnológicas e programação reforça a autonomia dos estudantes, estimulando o pensamento crítico e a capacidade de análise.

A proposta visa, portanto, não apenas transmitir conhecimentos teóricos, mas também incentivar uma visão mais ampla sobre a IA e seu impacto na sociedade. Através da experimentação e da resolução de desafios, os alunos são estimulados a construir seu próprio conhecimento, preparando-se para um futuro onde a inteligência artificial estará cada vez mais presente em diversas áreas do conhecimento e do mercado de trabalho.

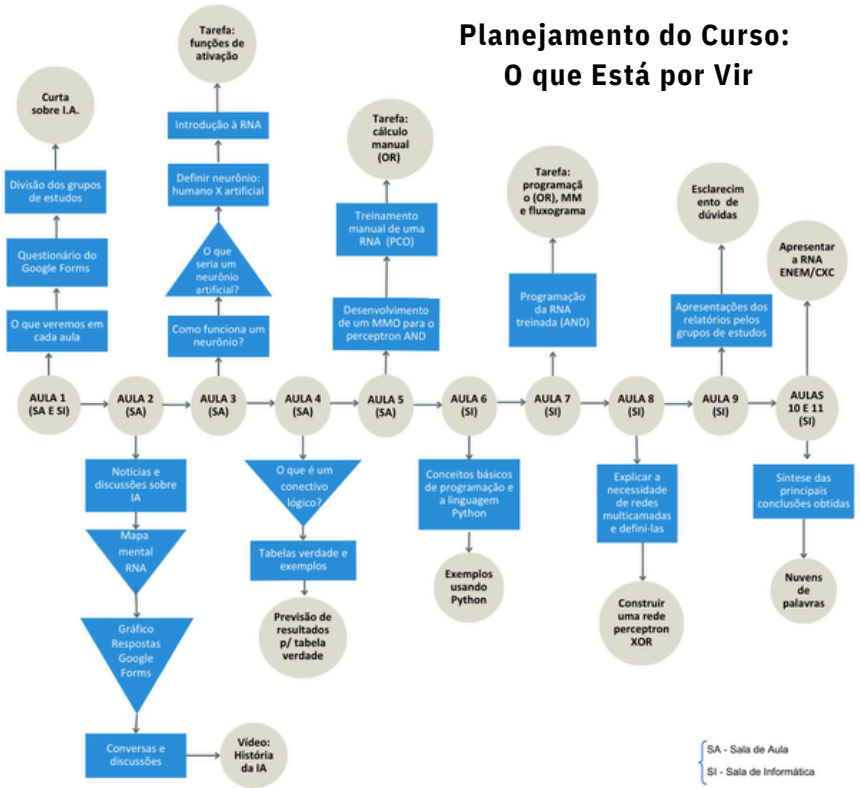
REFERÊNCIAS

- Ministério da Educação. Base Nacional Comum Curricular: Ensino Médio-Matemática. 2018. Disponível em: Link Acesso em: 22 dez. 2023
- FURTADO, M. I. V. Redes neurais artificiais: uma abordagem para sala de aula. Ponta Grossa, PR: Atena Editora, 2019.
- BURAK, D. Modelagem matemática sob um olhar de educação matemática e suas implicações para a construção do conhecimento matemático em sala de aula. Revista de Modelagem na Educação Matemática, v. 1, n. 1, p. 10–27, 2010. Disponível em: Link Acesso em: 17 jan. 2025.
- BASSANEZI, R. C. Ensino-aprendizagem com modelagem matemática: uma nova estratégia. São Paulo: Unicamp, 2002.
- HAYKIN, S. Redes neurais: princípios e prática. Porto Alegre: Editora Bookman, 2001.
- Deep Learning Book Brasil. Função de Ativação. 2025. Disponível em: Link Acesso em: 17 jan. 2025.
- NARANJO, J. F. L. Inteligência computacional aplicada na geração de respostas impulsivas bi-auriculares e em aurilização de salas. Dissertação (Mestrado) — Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2014.
- Sociedade Brasileira de Matemática. Redes neurais no ensino básico. 2022. Disponível em: Link Acesso em: 17 jan. 2025.
- PEREIRA, L. E. Contribuições da modelagem matemática para o desenvolvimento de habilidades de resolução de problemas no ensino médio. 2023. Disponível em: Link Acesso em: 17 jan. 2025.
- RUSSELL, S.; NORVIG, P. Inteligência Artificial: Uma Abordagem Moderna. 3. ed. Rio de Janeiro: Campus Elsevier, 2016. Tradução da obra original Artificial Intelligence: A Modern Approach.

REFERÊNCIAS

- ABOULNAGA, R. S. K. T. S. S. H.; EL-MASRI, M. Introdução à inteligência artificial e seu ensino nas escolas de ensino médio. São Paulo: Editora Exemplo, 2018.
- Data Science Academy. Uma breve história das redes neurais artificiais. Disponível em: Link Acesso em: 14 jan. 2025.
- KANDEL, E. R.; SCHWARTZ, J. H.; JESSELL, T. M. Princípios de neurociência. Porto Alegre, Brasil: AMGH Editora, 2000.
- HAYKIN, S. Redes neurais e máquinas de aprendizado. 3ª. ed. Upper Saddle River, NJ: Pearson Education, 2008. Título original: Neural Networks and Learning Machines.
- PEREIRA, F. G. S. G. Redes neurais artificiais: fundamentos e aplicações. Rio de Janeiro: Editora LTC, 2010.
- SILVA, D. H. S. e. R. A. F. Ivan Nunes da. Redes neurais artificiais para engenharia e ciências aplicadas. São Paulo: Editora Artliber, 2010.
- GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. Deep Learning. Cambridge: MIT Press, 2016.
- WING, J. Pensamento computacional: um conjunto de atitudes e habilidades que todos, não só cientistas da computação, ficaram ansiosos para aprender e usar. Revista Brasileira de Ensino de Ciência e Tecnologia, v. 9, n. 2, 2016.
- BATISTA, E. J. S. Pensamento Computacional: Teoria e Prática. Campo Grande, MS: Universidade Federal de Mato Grosso do Sul, 2023. Disponível em: Link Acesso em: 14 jan. 2025.
- COSTA, J.; SILVA, M.; OLIVEIRA, P. A aplicação do pensamento computacional em diferentes áreas do conhecimento. Revista Brasileira de Educação e Tecnologia, Editora ABC, v. 12, n. 3, p. 123–138, 2021. Disponível em: Link Acesso em: 21 jan. 2025.

APÊNDICE A



Este plano propõe uma sequência de 10 encontros, realizados ao longo de três meses, com aulas semanais alternando entre a sala de aula e o laboratório de informática. As atividades priorizam o uso de recursos tecnológicos, como retroprojetores, computadores e internet.

Na primeira aula, apresente os conceitos fundamentais de IA, avaliando o conhecimento prévio dos alunos. Nas aulas seguintes, introduza a teoria sobre Redes Neurais Artificiais (RNA) e os princípios básicos de programação, utilizando situações problema para a criação e análise de redes perceptron. Um dos pontos centrais será a construção manual de uma rede perceptron para resolver a função lógica AND, passo a passo, permitindo que os alunos compreendam intuitivamente o funcionamento dos pesos, do limiar e da ativação da rede.

Os alunos serão organizados em equipes para desenvolver soluções para os desafios propostos, apresentando suas abordagens e destacando as estratégias e conceitos matemáticos utilizados.

Ao final da sequência, os alunos deverão não apenas compreender os conceitos básicos da IA, mas também experimentar a criação prática de uma rede neural simples, motivando-os a explorar mais a fundo essa área. A proposta equilibra teoria e prática, com ênfase no desenvolvimento de habilidades aplicadas no laboratório de informática.

APÊNDICE B

Formulário sobre IA

Querido estudante, este é um questionário que tem o objetivo de entender o seu conhecimento prévio sobre inteligência artificial (IA). Suas respostas serão anônimas e me ajudarão a adaptar o ensino sobre esse tema.

Por favor, responda honestamente cada uma das perguntas a seguir, escolhendo opções que condizem com o seu conhecimento neste momento.
Obrigada por participar!
Marque a(s) opção(ões) correta(s):

1 - O que você entende por inteligência artificial (IA)?

IA é um termo usado para descrever sistemas computacionais que podem automatizar tarefas complexas, aprender com experiências passadas e realizar decisões autônomas sem intervenção humana constante.

A IA se relaciona com a criação de algoritmos e modelos de computador que possibilitam que as máquinas ajam de maneira inteligente.

IA refere-se à capacidade de máquinas imitarem a inteligência humana, realizando tarefas como aprendizado, raciocínio e resolução de problemas.

Não sei o que é IA.

2 - Você sabe como funciona a inteligência artificial?

◯ Sim, tenho um entendimento sólido de como a IA funciona, compreendendo os conceitos de aprendizado de máquina, algoritmos de processamento de dados e o uso de modelos preditivos para realizar tarefas diversas.
◯ Não, não tenho conhecimento sobre como a IA funciona. Tenho interesse em aprender mais sobre os princípios por trás dessa tecnologia.
◯ Tenho algum conhecimento superficial sobre a IA, mas gostaria de aprofundar meu entendimento sobre os mecanismos subjacentes e as aplicações práticas.
◯ Não tenho conhecimento algum sobre a IA.

3- Você já ouviu falar das redes neurais artificiais (RNA)?

◯ Não, nunca ouvi falar em redes neurais artificiais.
◯ Já ouvi falar do termo redes neurais, mas não tenho certeza sobre o que exatamente elas são ou como funcionam em contextos de IA.
◯ Já ouvi falar do termo redes neurais, mas não tenho certeza sobre o que exatamente elas são ou como funcionam.
◯ Já ouvi falar em redes neurais e sei exatamente como funcionam.

4- Você sabe qual é a relação entre a IA e as redes neurais artificiais?

◯ Não, não estou ciente da relação entre a IA e as RNA. Gostaria de saber mais sobre estes conceitos.
◯ Já ouvi falar da relação entre a IA e as RNA, mas não tenho um entendimento claro sobre como elas se conectam. Gostaria de aprender mais sobre essa associação específica.
◯ Sim, compreendo que as RNA são um subcampo da IA e constituem uma abordagem específica para modelar sistemas inteligentes, inspirada no funcionamento do cérebro humano.

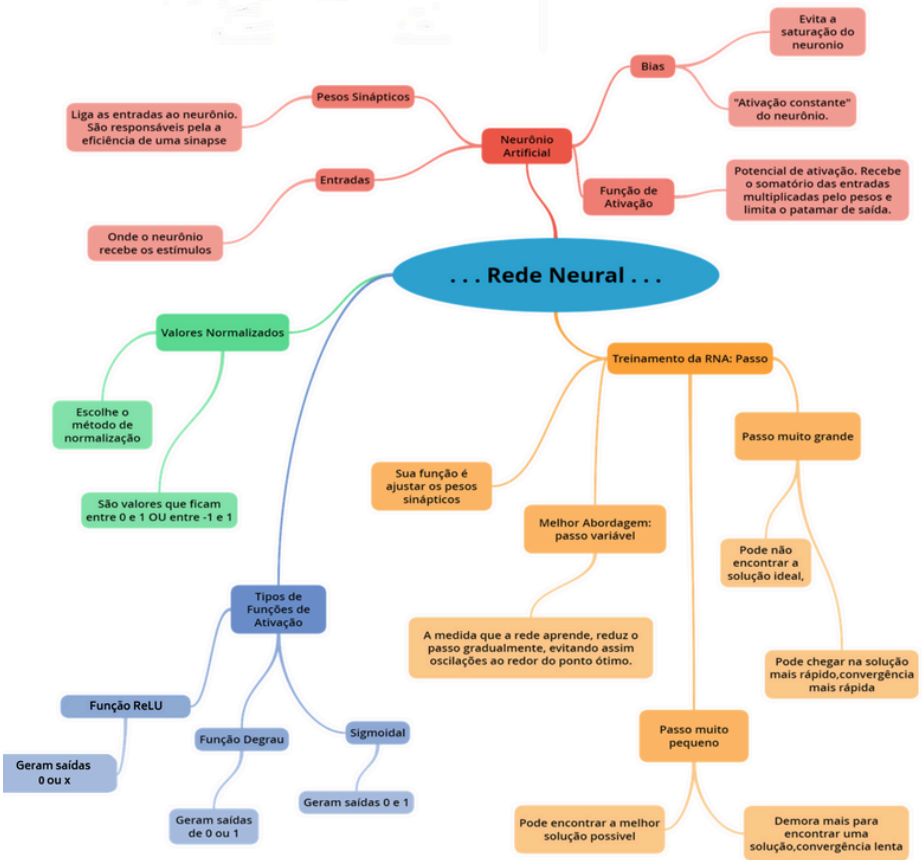
5- Você faz uso da inteligência artificial no seu dia a dia? Conte-me quando ou como (resposta aberta).

6- Você sabe se a matemática tem a ver com a IA?
◯ A relação entre matemática e a IA é mínima, pois a IA se baseia principalmente em lógica e programação, não dependendo significativamente de conceitos matemáticos.
◯ Sim, a matemática está intrinsecamente ligada à IA, pois fornece as bases teóricas e os algoritmos essenciais para o desenvolvimento de modelos inteligentes.
◯ Não, a matemática não tem relação com a IA, pois a IA é uma área independente que não requer conhecimentos matemáticos.

7- Você acha importante saber sobre a IA? Por quê?
◯ Sim, é importante saber sobre a IA, pois ela desempenha um papel crescente em diversas áreas da sociedade.
◯ Não, não vejo importância em aprender sobre a IA, pois acredito que ela não afeta diretamente minha vida cotidiana.
◯ Tenho alguma curiosidade sobre a IA, mas não tenho certeza de sua importância.

APÊNDICE C

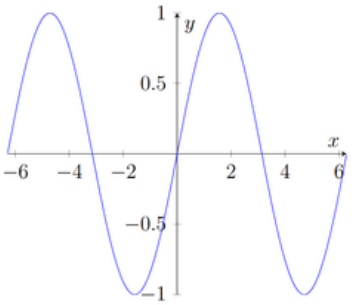
Fluxograma RNA



Fonte: Adaptado de FURTADO (2019), p. 34.

APÊNDICE D

Figura 1: Função Definida e Suave: y = sen(x)

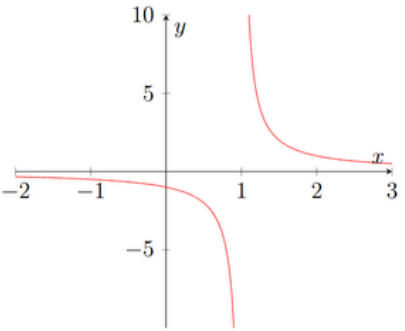


Fonte: Elaborada pela autora.

A função apresentada na Figura é contínua em todos os pontos, não apresenta quebras bruscas ou pontos onde não se define.

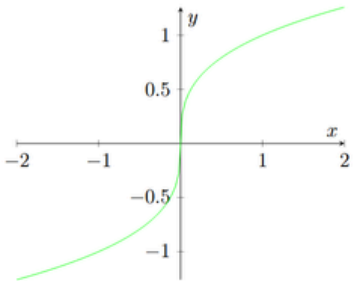
Figura 2 : Função com Descontinuidade Brusca: y = 1/(x−1).

A função apresentada na Figura 3.7 apresenta uma descontinuidade em x = 1, onde a função não é definida. Isso é representado pela separação das curvas de ambos os lados do ponto x = 1.



Fonte: Elaborada pela autora.

Figura 3 : Função com Pontos Singulares: y = x^1/3.



Fonte: Elaborada pela autora.

No gráfico da Figura 2, a função é contínua, mas com uma mudança abrupta na inclinação em x = 0.

Uma função é considerada não suave se tiver uma mudança abrupta na inclinação, como um ponto de inflexão, e continua se não há quebras, buracos ou saltos na curva da função.

A indefinição de uma função geralmente se refere a casos onde a função não está definida em determinados pontos do domínio. Por exemplo, um denominador igual a zero em uma função racional, como $y = 1 x$, se $x = 0$, ou funções com raízes quadradas de números negativos no caso de funções reais.

A função bem definida e suave, como a função seno apresentada na Figura 1, é contínua em todos os pontos e não exibe quebras ou saltos, representando um exemplo ideal de função de ativação que pode ser aplicada em uma rede neural. Por outro lado, a Figura 2 ilustra uma função que possui uma descontinuidade brusca, evidenciando o que acontece quando a função não é definida em um ponto específico, resultando em uma quebra no gráfico. Este tipo de função não é adequada para redes neurais, pois introduz incertezas e dificuldades no cálculo dos gradientes necessários para o aprendizado.

Finalmente, através da Figura 3 observe uma função com um ponto singular, onde ocorre uma mudança abrupta na inclinação. Embora essa função seja contínua, a presença de um ponto singular pode gerar instabilidades no processo de treinamento da rede neural.

É importante que uma função de ativação seja suave e contínua para que a rede neural possa aprender de forma eficiente. A continuidade garante que pequenas mudanças nos valores de entrada resultem em pequenas mudanças nos valores de saída, o que é essencial para o treinamento eficaz da RNA. Escolher a função de ativação correta é importante para que o modelo funcione bem. Funções como sigmoid, degrau e ReLU são as mais comuns e têm características diferentes que afetam como a rede aprende.

APÊNDICE E

Tarefa- Aula 3

Este exercício será avaliado em 1 ponto e deve ser entregue na data estipulada. Juntamente com os outros exercícios que serão solicitados nas aulas 5 e 7, o total de pontos avaliativos para o bimestre será 10.

Data de entrega:___/___/____

Dadas as funções de ativação:

1. Sigmoid: $\sigma(x) = \frac{1}{1+e^{-x}}$
2. ReLU (Rectified Linear Unit): $\text{ReLU}(x) = \max(0, x)$
3. Degrau: degrau(x) = $\begin{cases} 0 \text{ se } x < 0 \\ 1 \text{ se } x \geq 0 \end{cases}$

Calcule os resultados das funções de ativação sigmoide, ReLU e degrau para os valores fornecidos abaixo. Utilize uma calculadora, se necessário. Cada grupo tem valores de entrada diferentes, como nos 4 exemplos a seguir, e a quantidade de grupos, certamente, deverá ser adaptada à realidade da turma.

- Grupo 1: x = −3, −2.5, −2, −1.5, −1
- Grupo 2: x = −2.5, −2, −1.5, −1, −0.5
- Grupo 3: x = −2, −1.5, −1, −0.5, 0
- Grupo 4: x = −1.5, −1, −0.5, 0, 0.5

Cada grupo deve preencher o quadro a seguir com os resultados dos cálculos para cada valor de x.

x	$\sigma(x)$	$\text{ReLU}(x)$	$\text{Degrau}(x)$

Exemplos de Cálculos:

1. **Para x = 1:**
   - $\sigma(1) = \frac{1}{1+e^{-1}} \approx 0.731$
   - ReLU(1) = max(0, 1) = 1
   - degrau(1) = 1

2. **Para x = −2:**
   - ReLU(−2) = max(0, −2) = 0
   - degrau(−2) = 0
   - $\sigma(x) = \frac{1}{1+e^{-x}} \sigma(-2) = \frac{1}{1+e^{-(-2)}} = \frac{1}{1+e^{2}} \approx 0.119$

APÊNDICE F

Tarefa- Aula 5

Desafio sobre a Aplicação Manual e Computacional da RNA Perceptron

• **Questão 1 (2 pontos)**: Considerando as variáveis de entrada do algoritmo matemático da RNA Perceptron, utilize os valores das proposições ($P \vee Q$) da tabela verdade do conectivo lógico OR ($\vee$), indicados na Tabela 3.3 e reproduzidos a seguir:

Tabela 3.3- Tabela OR (**v**)

P	Q	P ∧ Q
V	V	V
V	F	F
F	V	F
F	F	F

Faça uso de uma calculadora para obter os cálculos da saída das primeiras épocas da RNA Perceptron, onde o objetivo é a obtenção da saída ($P \vee Q$). Assim como calculamos para ($P \wedge Q$), calcule agora para ($P \vee Q$), repetindo o algoritmo matemático desenvolvido em sala de aula para o conectivo ($P \wedge Q$).

Depois, proponha um modelo matemático final, usando as especificações de entradas dadas.

Sugere-se a seguinte distribuição de tarefas por grupo:

Grupo 1: Calcular a saída manualmente com pesos iniciais w_1= 0.3 e w_2 = 0.7, e bias inicial b = −0.4. Use a função degrau.

Grupo 2: Calcular a saída manualmente com pesos iniciais w_1= 0.6 e w_2= 0.9, e bias inicial b = −1.2. Use a função sigmoide.

Grupo 3: Calcular a saída manualmente com pesos iniciais w_1= 0.4 e w_2 = 0.6, e bias inicial b = −0.3. Use a função ReLU.

Grupo 4: Calcular a saída manualmente com pesos iniciais w_1= 1.2 e w_2= 0.5, e bias inicial b = −0.8. Use a função degrau.

Grupo 5: Calcular a saída manualmente com pesos iniciais w_1 = 0.5 e w_2= 0.5, e bias inicial b = −0.5.Use a função sigmoide.

- **Questão 2 (3 pontos):**

Durante o cálculo manual da RNA perceptron, identifique um padrão de processo de cálculo que possa ser automatizado por um programa computacional. Com base no fluxograma apresentado na aula, proponha um fluxograma para implementar este procedimento no computador.

Lembre-se que um fluxograma é uma representação gráfica que ilustra a sequência de passos ou etapas necessários para realizar uma tarefa ou resolver um problema. São ferramentas importantes para visualizar e entender a sequência de ações necessárias para completar uma tarefa.

Utilizando símbolos padronizados como caixas (ou blocos) e setas, o fluxograma descreve visualmente o fluxo do processo, tornando mais fácil a compreensão e a comunicação das etapas envolvidas.

Os componentes básicos incluem caixas de início/fim (retângulos com cantos arredondados), processos (retângulos), decisões (losangos) e entrada/saída (paralelogramos), enquanto as setas indicam a direção do fluxo entre as etapas.

Atente-se aos seguintes pontos:

• Explicação dos cálculos realizados para a Questão 1.

• Demonstração do padrão de processo de cálculo identificado na Questão 2 e apresentação do fluxograma proposto.

Sejam claros e objetivos na apresentação da solução. Ela fará parte de um relatório a ser entregue no nono encontro, conforme data de entrega prevista.

Bom trabalho!

Sugestão de Grupos e Tarefas

Grupo	Função de Ativação	Pesos (w1, w2)	Bias (b)
Grupo 1	Degrau	0.3, 0.7	-0.4
Grupo 2	Sigmoide	0.6, 0.9	-1.2
Grupo 3	ReLU	0.4, 0.6	-0.3
Grupo 4	Degrau	1.2, 0.5	-0.8
Grupo 5	Sigmoide	0.5, 0.5	-0.5

Tarefa:

Calcular a saída manualmente com os pesos e bias dados para $P \vee Q$ usando a função correspondente e propor um fluxograma para automatizar o cálculo da RNA perceptron.

APÊNDICE G

33 Passos para uma RNA em 2 iterações.

Passo 1:
Escolher uma das entradas (x_1, x_2), definir pesos iniciais ($w_{1,0}$, $w_{2,0}$); bias inicial (b_0) e fixar a taxa de (α).

Relembre aqui que α é um hiper parâmetro, um parâmetro cujo valor é definido antes do início do processo de treinamento de um modelo de aprendizado de máquina e que não é ajustado durante o treinamento; controla o tamanho dos ajustes feitos nos pesos e bias do modelo a cada iteração. Uma taxa de aprendizagem alta pode acelerar o treinamento, mas pode levar a uma convergência instável. Por outro lado, uma taxa de aprendizagem baixa proporciona uma convergência mais estável, mas pode tornar o treinamento muito lento e exigir muitas iterações para alcançar um resultado satisfatório.

Encontrar um equilíbrio adequado para a taxa de aprendizagem é essencial para o desempenho eficiente do modelo.

Passo 2: Monte a soma: $y = x_1 w_{1,0} + x_2 w_{2,0} + b_0$ (modelo 0-da soma)

Passo 3: Defina uma função de ativação f e aplicar a y, ou seja, y' = f (y)

Passo 4: Identifique o valor de saída y correspondente à entrada e calcular o erro: E = y − y'.

Passo 5: Faça a atualização de pesos e bias:

$$\begin{aligned} w_{1,1} &= w_{1,0} + x_1 \alpha E \\ w_{2,1} &= w_{2,0} + x_2 \alpha E \\ b_1 &= b_0 + \alpha E \end{aligned}$$

A regra de atualização do viés é derivada diretamente da regra de atualização dos pesos, com a diferença de que o viés não está associado a nenhuma entrada específica (é como um peso associado a uma entrada constante de valor 1). Matematicamente, ambas as regras são baseadas na minimização do erro de classificação, ajustando os parâmetros do modelo na direção que reduz o erro.

Passo 6: Proponha o novo modelo da soma: $y = x_1 w_{1,1} + x_2 w_{2,1} + b_1$ (modelo 1 - da soma)

Passo 7: Escolha outra entrada (x_1, x_2), diferente da escolhida no Passo 1 e repetir os passos 2 ao 6, considerando a nova atualização da soma, até que para toda entrada tenha o erro E = 0.

Passo 8: Uma vez atingido $E = \bar{y} - \hat{y} = 0$, para toda entrada, proponha o modelo matemático final:
$y = x_1 w_{1,n+1} + x_2 w_{2,n+1} + b_{n+1}$, em que:

$$w_{1,n+1} = w_{1,n} + x_1 \alpha E$$
$$w_{2,n+1} = w_{2,n} + x_2 \alpha E$$
$$b_{n+1} = b_n + \alpha E$$

$w_{1,n+1}, w_{2,n+1}$ e b_{n+1} são os pesos e bias atualizados e $w_{1,n}$, $w_{2,n}$ e b_n são pesos e bias anteriores.
Calcular o erro: $E = y - y' = 0 - 0 = 0.$
A seguir, atualize pesos e bias:

$$w_{1,2} = w_{1,1} + x_1 \alpha E = 0 + 1(0.4)(0) = 0$$
$$w_{2,2} = w_{2,1} + x_2 \alpha E = 0 + 0(0.4)(0) = 0$$
$$b_2 = b_1 + \alpha E = -1 + 0.4(0) = -1$$

Passo 9: Proponha o novo modelo da soma:
$y = x_1 w_{1,2} + x_2 w_{2,2} + b_2 = x_1(0) + x_2(0) - 1$ (modelo 2 - da soma).

Passo 10: Escolha outra entrada $(x_1, x_2) \neq (0,1)$, diferente das escolhidas nos Passos 1 e 6, e realize a atualização da soma:

$$y = x_1 w_{1,2} + x_2 w_{2,2} + b_2 = 0(0) + 1(0) - 1 = -1$$

Passo 11: Aplique a função degrau a y = −1:

$y' = f(-1) = 0,$ pois $y = -1 < 0.$

Passo 12: Identifique a saída y = 0, correspondente à entrada (x1, x2) = (0, 1) e calcular o erro: E = y − y′ = 0 − 0 = 0.

Atualize pesos e bias:

$$w_{1,3} = w_{1,2} + x_1 \eta E = 0 + 0(0.4)(0) = 0$$
$$w_{2,3} = w_{2,2} + x_2 \eta E = 0 + 1(0.4)(0) = 0$$
$$b_3 = b_2 + \eta E = -1 + 0.4(0) = -1$$

O erro continua nulo, então, pesos e bias mantém seus valores iniciais.

Passo 13: Proponha o novo modelo da soma:

$y = x_1 w_{1,3} + x_2 w_{2,3} + b_3 = x_1(0) + x_2(0) - 1$ (modelo 3 - da soma que, nesse caso, coincide com o modelo 2.)

Passo 14: Escolha outra entrada (x1, x2) = (1,1), diferente das escolhidas nos Passos 1, 6 e 10, e realize a atualização da soma:

$y = x_1 w_{1,3} + x_2 w_{2,3} + b_3 = 1(0) + 1(0) - 1 = -1$

Atualize pesos e bias:

$$w_{1,5} = w_{1,4} + x_1 \eta E = 0.4 + 0(0.4)(0) = 0.4$$
$$w_{2,5} = w_{2,4} + x_2 \eta E = 0.4 + 0(0.4)(0) = 0.4$$
$$b_5 = b_4 + \eta E = -0.6 + 0.4(0) = -0.6$$

Passo 15: Aplique a função degrau a y = −1:

$\hat{y} = f(-1) = 0,$ pois $y = -1 < 0.$

Passo 16: Identifique a saída ¯y = 1, correspondente à entrada $(x_1, x_2) = (1,1)$ e calcular o erro: $E = \bar{y} - \hat{y} = 1 - 0 = 1.$

Atualize pesos e bias:

$$w_{1,4} = w_{1,3} + x_1 \eta E = 0 + 1(0.4)(1) = 0.4$$
$$w_{2,4} = w_{2,3} + x_2 \eta E = 0 + 1(0.4)(1) = 0.4$$
$$b_4 = b_3 + \eta E = -1 + 0.4(1) = -0.6$$

Passo 17: Proponha o novo modelo da soma:
$y = x_1 w_{1,4} + x_2 w_{2,4} + b_4 = x_1(0.4) + x_2(0.4) - 0.6$ (modelo 4 - da soma)
OBS: Como ainda não atingiu-se E = 0 para toda entrada, e tem-se que escolher uma entrada diferente da escolhida no Passo 14, repete-se o processo nas entradas (0,0), (0,1), (1,0) e (1,1), sequencialmente.

Passo 18: Escolha a entrada $(x_1, x_2) = (0, 0)$ e atualize a soma:
$y = x_1 w_{1,4} + x_2 w_{2,4} + b_4 = 0(0.4) + 0(0.4) - 0.6 = -0.6$

Passo 19: Aplique a função degrau a y = −0.6:
$\hat{y} = f(-0.6) = 0, \quad \text{pois} \quad y = -0.4 < 0.$

Passo 20: Identifique a saída $\bar{y} = 0$, correspondente à entrada $(x_1, x_2) = (0, 0)$ e calcule o erro: $E = \bar{y} - \hat{y} = 0 - 0 = 0.$
Passo 21: Proponha o novo modelo da soma:
$y = x_1 w_{1,5} + x_2 w_{2,5} + b_5 = x_1(0.4) + x_2(0.4) - 0.6$ (modelo 5 - da soma).

Passo 22: Escolha outra entrada, $(x_1, x_2) = (0, 1)$ e realizar a atualização da soma:
$y = x_1 w_{1,5} + x_2 w_{2,5} + b_5 = 0(0.4) + 1(0.4) - 0.6 = -0.2$

Passo 23: Aplique a função degrau a y = −0.2:
$\hat{y} = f(-0.2) = 0, \quad \text{pois} \quad y = -0.2 < 0.$

Passo 24: Identifique a saída $\bar{y} = 0$, correspondente à entrada $(x_1, x_2) = (0, 1)$ e calcular o erro: $E = \bar{y} - \hat{y} = 0 - 0 = 0.$

Atualize pesos e bias:

$$w_{1,6} = w_{1,5} + x_1 \eta E = 0.4 + 0(0.4)(0) = 0.4$$
$$w_{2,6} = w_{2,5} + x_2 \eta E = 0.4 + 1(0.4)(0) = 0.4$$
$$b_6 = b_5 + \eta E = -0.6 + 0.4(0) = -0.6$$

Passo 25: Proponha o novo modelo da soma:
$y = x_1 w_{1,6} + x_2 w_{2,6} + b_6 = x_1(0.4) + x_2(0.4) - 0.6$ (modelo 6 - da soma).

Passo 26: Escolha outra entrada, (x1, x2) = (1,1) e realize a atualização da soma:
$y = x_1 w_{1,7} + x_2 w_{2,7} + b_7 = 1(0.4) + 1(0.4) - 0.6 = 0.2$

Passo 27: Aplique a função degrau a y = 0.2:
$\hat{y} = f(0.2) = 1, \quad \text{pois} \quad y = 0.2 > 0.$

Passo 28: Identifique a saída $\bar{y} = 1$, correspondente à entrada $(x_1, x_2) = (1, 1)$ e calcular o erro:
$E = \bar{y} - \hat{y} = 0 - 0 = 0.$

Atualize pesos e bias:

$$w_{1,7} = w_{1,6} + x_1 \eta E = 0.4 + 0(0.4)(0) = 0.4$$
$$w_{2,7} = w_{2,6} + x_2 \eta E = 0.4 + 1(0.4)(0) = 0.4$$
$$b_7 = b_6 + \eta E = -0.6 + 0.4(0) = -0.6$$

Passo 29: Proponha o novo modelo da soma:
$y = x_1 w_{1,7} + x_2 w_{2,7} + b_7 = x_1(0.4) + x_2(0.4) - 0.6$ (modelo 6 - da soma).

Passo 30: Escolha outra entrada, $(x_1, x_2) = (1, 1)$ e realize a atualização da soma:
$y = x_1 w_{1,7} + x_2 w_{2,7} + b_7 = 1(0.4) + 1(0.4) - 0.6 = 0.2$

Passo 31: Aplique a função degrau a y = 0.2:
$\hat{y} = f(0.2) = 1, \text{ pois } y = 0.2 > 0.$

Passo 32: Identifique a saída $\bar{y} = 1$, correspondente à entrada $(x_1, x_2) = (1, 1)$ e calcule o erro: $E = \bar{y} - \hat{y} = 1 - 1 = 0.$

Passo 33: Tendo encontrado $E = \bar{y} - \hat{y} = 0$ para toda entrada (x_1, x_2), proponha o MMO final:

$$\bar{y} = f(x_1 w_{1,7} + x_2 w_{2,7} + b_7) = f(x_1(0.4) + x_2(0.4) - 0.6)$$

Encontrando o modelo final da soma: $y = x_1(0.4) + x_2(0.4) - 0.6$

APÊNDICE H

Códigos para a Programação da Rede Perceptron AND

O código, que será mostrado a seguir, treina um perceptron simples para resolver um problema de lógica AND. Ele começa definindo funções auxiliares e parâmetros, inicializa pesos e vieses, e, em seguida, atualiza-os com base nos erros calculados durante o treinamento. Todo o processo é acompanhado por prints detalhados no console para monitorar o progresso.

```
# Importar bibliotecas necessárias
import numpy as np

# Função de ativação: degrau (step function)
def funcao_degrau(x):
    return 1 if x >= 0 else 0

# Função para treinar o perceptron
def treinar_perceptron(X, y, taxa_aprendizado, epocas):

    # Inicializar pesos e viés
    w = np.zeros(X.shape[1])  # Pesos iniciais
    b = -1  # Viés inicial
    print(f"Pesos iniciais: {w}, Viés inicial: {b}")

    # Treinamento do perceptron
    for epoca in range(epocas):
        print(f"\nÉpoca {epoca + 1}/{epocas}")
        for i in range(X.shape[0]):

            # Calcular a saída do perceptron
            saida_linear = np.dot(X[i], w) + b
            y_pred = funcao_degrau(saida_linear)

            # Calcular o erro
            erro = y[i] - y_pred
```

```
            # Atualizar os pesos e o viés
            w += taxa_aprendizado * erro * X[i]
            b += taxa_aprendizado * erro
            print(f"Entrada: {X[i]}, Saída esperada: {y[i]}, "
                f"Saída calculada: {y_pred}, Erro: {erro}")
        print(f"Atualização de pesos: {w}, Atualização de viés: {b}")
    return w, b

# Definindo o conjunto de dados (Entradas e Saídas esperadas)
X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])  # Entradas
y = np.array([0, 0, 0, 1])  # Saídas esperadas (AND lógico)

# Definir a taxa de aprendizado e o número de épocas
taxa_aprendizado = 0.4
epocas = 2

# Treinar o perceptron
pesos, vies = treinar_perceptron(X, y, taxa_aprendizado, epocas)
print(f"\nPesos finais: {pesos}, Viés final: {vies}")
```

O seguinte resultado será exibido ao executar os códigos:

Pesos iniciais: [0. 0.], Bias inicial: -1

Época 1/2
Entrada: [0 0], Saída esperada: 0, Saída calculada: 0, Erro: 0
Atualização de pesos: [0. 0.], Atualização de bias: -1.0
Entrada: [0 1], Saída esperada: 0, Saída calculada: 0, Erro: 0
Atualização de pesos: [0. 0.], Atualização de bias: -1.0
Entrada: [1 0], Saída esperada: 0, Saída calculada: 0, Erro: 0
Atualização de pesos: [0. 0.], Atualização de bias: -1.0
Entrada: [1 1], Saída esperada: 1, Saída calculada: 0, Erro: 1
Atualização de pesos: [0.4 0.4], Atualização de bias: -0.6

Época 2/2
Entrada: [0 0], Saída esperada: 0, Saída calculada: 0, Erro: 0
Atualização de pesos: [0.4 0.4], Atualização de bias: -0.6
Entrada: [0 1], Saída esperada: 0, Saída calculada: 0, Erro: 0
Atualização de pesos: [0.4 0.4], Atualização de bias: -0.6
Entrada: [1 0], Saída esperada: 0, Saída calculada: 0, Erro: 0
Atualização de pesos: [0.4 0.4], Atualização de bias: -0.6
Entrada: [1 1], Saída esperada: 1, Saída calculada: 1, Erro: 0
Atualização de pesos: [0.4 0.4], Atualização de bias: -0.6

Pesos finais: [0.4 0.4], Bias final: -0.6

APÊNDICE I

Códigos da RNA Perceptron AND, Função Sigmoide

```
# Importar bibliotecas necessárias
import numpy as np

# Função de ativação sigmoide
def funcao_sigmoide(x):
    return 1 / (1 + np.exp(-x))

# Função para treinar o perceptron
def treinar_perceptron(x, y, taxa_aprendizado, epocas):

# Inicializar pesos e viés
    w = np.zeros(x.shape[1])
    b = -1 # Viés inicial
    print(f"Pesos iniciais: {w}, Viés inicial: {b}")

# Treinamento do perceptron
    for epoca in range(epocas):
        print(f"\nÉpoca {epoca + 1}/{epocas}")
        for i in range(x.shape[0]):

# Calcular a saída do perceptron
            saida_linear = np.dot(x[i], w) + b
            y_pred = funcao_sigmoide(saida_linear)
            saida_final = 1 if y_pred >= 0.5 else 0

# Calcular o erro
            erro = y[i] - saida_final

# Atualizar os pesos e o viés
            w += taxa_aprendizado * erro * x[i]
            b += taxa_aprendizado * erro
            print(f"Entrada: {x[i]}, Saída esperada: {y[i]},
                Saída calculada (sigmoide): {y_pred},
                Saída final: {saida_final},
```

```
Erro: {erro}")
                print(f"Atualização de pesos: {w},
#Atualização de viés: {b}")
return w, b

# Definindo o conjunto de dados
# (Entradas e Saídas esperadas)
X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]]) # Entradas
y = np.array([0, 0, 0, 1]) # Saídas esperadas (AND lógico)

# Definir a taxa de aprendizado e o número de épocas
taxa_aprendizado = 0.4
epocas = 2

# Treinar o perceptron
pesos, viés = treinar_perceptron(X, y, taxa_aprendizado, epocas)
print(f"\nPesos finais: {pesos}, Viés final: {viés}")
```

Visualizamos, agora, o seguinte resultado ao executar o algoritmo, usando a função de ativação sigmoide:

```
Pesos iniciais: [0. 0.], Viés inicial: -1
Época 1/2
Entrada: [0 0], Saída esperada: 0, Saída calculada (sigmoide): 0.2689414213699951, Saída final: 0, Erro: 0 Atualização de pesos: [0. 0.], Atualização de viés: -1.0
Entrada: [0 1], Saída esperada: 0, Saída calculada (sigmoide): 0.2689414213699951, Saída final: 0, Erro: 0 Atualização de pesos: [0. 0.], Atualização de viés: -1.0
Entrada: [1 0], Saída esperada: 0, Saída calculada (sigmoide): 0.2689414213699951, Saída final: 0, Erro: 0 Atualização de pesos: [0. 0.], Atualização de viés: -1.0
Entrada: [1 1], Saída esperada: 1, Saída calculada (sigmoide): 0.2689414213699951, Saída final: 0, Erro: 1 Atualização de pesos: [0.4 0.4], Atualização de viés: -0.6
```

Época 2/2

Entrada: [0 0], Saída esperada: 0, Saída calculada (sigmoide): 0.35434369377420455, Saída final: 0, Erro: 0 Atualização de pesos: [0.4 0.4], Atualização de viés: -0.6

Entrada: [0 1], Saída esperada: 0, Saída calculada (sigmoide): 0.45016600268752216, Saída final: 0, Erro: 0 Atualização de pesos:[0.4 0.4], Atualização de viés: -0.6

Entrada: [1 0], Saída esperada: 0, Saída calculada (sigmoide): 0.45016600268752216, Saída final: 0, Erro: 0 Atualização de pesos: [0.4 0.4], Atualização de viés: -0.6

Entrada: [1 1], Saída esperada: 1, Saída calculada (sigmoide): 0.549833997312478, Saída final: 1, Erro: 0 Atualização de pesos: [0.4 0.4], Atualização de viés: -0.6

Pesos finais: [0.4 0.4], Viés final: -0.6

APÊNDICE J

Códigos da RNA Perceptron AND: Função ReLU

```
# Importar bibliotecas necessárias
import numpy as np

# Função de ativação ReLU
def funcao_relu(x):
    return max(0, x)

# Função para treinar o perceptron
def treinar_perceptron(x, y, taxa_aprendizado, epocas):

# Inicializar pesos e viés
    w = np.zeros(x.shape[1]) # Pesos iniciais
    b = -1 # Viés inicial
    print(f"Pesos iniciais: {w}, Viés inicial: {b}")

# Treinamento do perceptron
    for epoca in range(epocas):
        print(f"\nÉpoca {epoca + 1}/{epocas}")
        for i in range(x.shape[0]):

# Calcular a saída do perceptron
            saida_linear = np.dot(x[i], w) + b
            y_pred = funcao_relu(saida_linear)
            saida_final = 1 if y_pred > 0 else 0

# Calcular o erro
            erro = y[i] - saida_final

# Atualizar os pesos e o viés
            w += taxa_aprendizado * erro * x[i]
            b += taxa_aprendizado * erro
            print(f"Entrada: {x[i]}, Saída esperada: {y[i]},
                  Saída calculada (ReLU): {y_pred},
                  Saída final: {saida_final}, Erro: {erro}")
            print(f"Atualização de pesos: {w},
# Atualização de viés: {b}")
return w, b
```

```
# Definindo o conjunto de dados
# (Entradas e Saídas esperadas)
X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]]) # Entradas
y = np.array([0, 0, 0, 1]) # Saídas esperadas (AND lógico)

# Definir a taxa de aprendizado e o número de épocas
taxa_aprendizado = 0.4
epocas = 2

# Treinar o perceptron
pesos, viés = treinar_perceptron(X, y, taxa_aprendizado, epocas)
print(f"\nPesos finais: {pesos}, Viés final: {viés}")
```

Resultado encontrado:

Pesos iniciais: [0. 0.], Viés inicial: -1

Época 1/2

Entrada: [0 0], Saída esperada: 0, Saída calculada (ReLU): 0, Saída final: 0, Erro: 0 Atualização de pesos: [0. 0.], Atualização de viés: -1.0
Entrada: [0 1], Saída esperada: 0, Saída calculada (ReLU): 0, Saída final: 0, Erro: 0 Atualização de pesos: [0. 0.], Atualização de viés: -1.0
Entrada: [1 0], Saída esperada: 0, Saída calculada(ReLU): 0, Saída final: 0, Erro: 0 Atualização de pesos: [0. 0.], Atualização de viés: -1.0
Entrada: [1 1], Saída esperada: 1, Saída calculada (ReLU): 0, Saída final: 0, Erro: 1 Atualização de pesos: [0.4 0.4], Atualização de viés: -0.6

Época 2/2

Entrada: [0 0], Saída esperada: 0, Saída calculada (ReLU): 0, Saída final: 0, Erro: 0 Atualização de pesos: [0.4 0.4], Atualização de viés: -0.6
Entrada: [0 1], Saída esperada: 0, Saída calculada (ReLU): 0, Saída final: 0, Erro: 0 Atualização de pesos: [0.4 0.4], Atualização de viés: -0.6
Entrada: [1 0], Saída esperada: 0, Saída calculada (ReLU): 0, Saída final: 0, Erro: 0 Atualização de pesos: [0.4 0.4], Atualização de viés: -0.6
Entrada: [1 1], Saída esperada: 1, Saída calculada (ReLU): 0.20000000000000007, Saída final: 1, Erro: 0 Atualização de pesos: [0.4 0.4], Atualização de viés: -0.6

Pesos finais: [0.4 0.4], Viés final: -0.6

APÊNDICE K

Tarefa -Aula 7

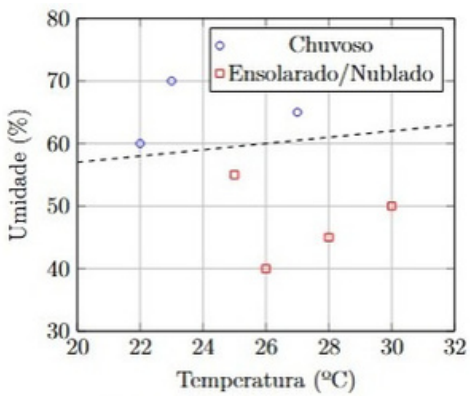
Exercício: Abaixo está a Tabela 3.10 com dados fictícios de temperatura, umidade e condição do dia para uma semana:

Temperatura e umidade de CXC por uma semana

Temperatura (ºC)	Umidade (%)	Dia
22	60	Chuvoso
28	45	Ensolarado
25	55	Nublado
30	50	Ensolarado
27	65	Chuvoso
23	70	Chuvoso
26	40	Ensolarado

Fonte: Elaborada pela autora.

Observe que estes dados são linearmente separáveis:
Separação Linear dos Dados para Temperatura e Umidade



Fonte: Elaborada pela autora.

Neste caso, podemos treinar uma rede perceptron de camada única para prever se os próximos dias serão chuvosos ou ensolarados, com base nas temperaturas e umidades previstas. A rede utilizará os dados de temperatura e umidade como entradas para determinar a condição do dia, classificando-o como "Chuvoso" ou "Ensolarado/Nublado".

Com os dados sendo linearmente separáveis, o perceptron é uma escolha adequada para realizar essa classificação, aprendendo a partir dos exemplos fornecidos na tabela acima.

A tarefa é analisar cada linha da programação deste código, apresentado na sequência, indicando suas funcionalidades:

```
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import Perceptron
from sklearn.metrics import accuracy_score
# Dados fictícios
X = np.array([
[22, 60],
[28, 45],
[25, 55],
[30, 50],
[27, 65],
[23, 70],
[26, 40]
])
y = np.array([
'Chuvoso',
'Ensolarado',
'Nublado',
'Ensolarado',
'Chuvoso',
'Chuvoso',
'Ensolarado'
])
# Codificação das classes
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)
# Criação e treinamento do modelo Perceptron
model = Perceptron()
model.fit(X, y_encoded)
# Previsões
y_pred = model.predict(X)
# Avaliação do modelo
accuracy = accuracy_score(y_encoded, y_pred)
print(f'Precisão do modelo: {accuracy:.2f}')
# Exemplo de previsão
novos_dados = np.array([
[24, 62], # Exemplo de dados
de temperatura e umidade
[29, 48]
])
previsoes = model.predict(novos_dados)
previsoes_decodificadas = label_encoder.
inverse_transform(previsoes)
print("Previsões para os novos dados:",
previsoes_decodificadas)
```

Bom trabalho!

APÊNDICE L

Códigos da RNA Perceptron XOR: Função Degrau

```
import tensorflow as tf
import matplotlib.pyplot as plt

# Definindo os dados de entrada e saída para a função XOR
entradas = tf.constant([[0, 0], [0, 1], [1, 0], [1, 1]], dtype=tf.float32)
saidas = tf.constant([[0], [1], [1], [0]], dtype=tf.float32)

# Definindo a arquitetura da rede MLP
modelo = tf.keras.Sequential([
tf.keras.layers.Dense(2, activation='relu', input_shape=(2,)),

# Camada oculta com 2 neurônios e função de ativação ReLU
tf.keras.layers.Dense(1, activation='sigmoid')

# Camada de saída com 1 neurônio e função de ativação sigmoide])
# Compilando o modelo
modelo.compile(optimizer='adam',  loss='binary_crossentropy',  metrics=
['accuracy'])

# Treinando o modelo e salvando o histórico do treinamento
historico = modelo.fit(entradas, saidas, epochs=5000, verbose=0)

# Aumentando o número de épocas para 5000
# Fazendo previsões
previsoes = modelo.predict(entradas)
previsoes_binarias = (previsoes > 0.5).astype(int)

# Aplicando threshold para obter previsões binárias
print("Previsões:")
print(previsoes_binarias)

# Plotando a função de custo durante o treinamento
plt.plot(historico.history['loss'])
plt.title('Função de Custo ao Longo do Treinamento')
plt.xlabel('Época')
plt.ylabel('Função de Custo')
plt.show()
```

plt.show()

Esses valores representam as saídas previstas pelo modelo para cada uma das entradas da função XOR. Como a camada de saída usa a função de ativação sigmoide, as previsões estão na faixa de 0 a 1.

Aqui está o que cada previsão significa:
[[0.4397117]] - Entrada [0, 0]: O modelo prevê aproximadamente 0.44.
[[0.5586208]] - Entrada [0, 1]: O modelo prevê aproximadamente 0.56.
[[0.7639964]] - Entrada [1, 0]: O modelo prevê aproximadamente 0.76.
[[0.23500887]] - Entrada [1, 1]: O modelo prevê aproximadamente 0.24.
Idealmente, para a função XOR, esperamos as seguintes saídas:
Entrada [0, 0] deve dar saída 0
Entrada [0, 1] deve dar saída 1
Entrada [1, 0] deve dar saída 1
Entrada [1, 1] deve dar saída 0
As previsões do modelo estão próximas, mas não perfeitamente alinhadas com os valores esperados, indicando que o modelo está aprendendo a função XOR, mas ainda ode não está completamente treinado ou a arquitetura pode não ser suficientemente complexa para capturar totalmente a função XOR; pode precisar de justes adicionais .
Esta rede neural é composta por uma camada oculta com a função de ativação ReLU e uma camada de saída com a função de ativação sigmoide, que é adequada para problemas de classificação binária.

Esses valores representam as saídas previstas pelo modelo para cada uma das entradas da função XOR. Como a camada de saída usa a função de ativação sigmoide, as previsões estão na faixa de 0 a 1.
Aqui está o que cada previsão significa:
[[0.4397117]] - Entrada [0, 0]: O modelo prevê aproximadamente 0.44.
[[0.5586208]] - Entrada [0, 1]: O modelo prevê aproximadamente 0.56.
[[0.7639964]] - Entrada [1, 0]: O modelo prevê aproximadamente 0.76.
[[0.23500887]] - Entrada [1, 1]: O modelo prevê aproximadamente 0.24.

Idealmente, para a função XOR, esperamos as seguintes saídas:
Entrada [0, 0] deve dar saída 0
Entrada [0, 1] deve dar saída 1
Entrada [1, 0] deve dar saída 1
Entrada [1, 1] deve dar saída 0

As previsões do modelo estão próximas, mas não perfeitamente alinhadas com os valores esperados, indicando que o modelo está aprendendo a função XOR, mas ainda pode não está completamente treinado ou a arquitetura pode não ser suficientemente complexa para capturar totalmente a função XOR. Ajustes adicionais nos hiperparâmetros ou na arquitetura do modelo podem melhorar a precisão.

EDITORA CHEFE
Profº Me. Isabele de Souza Carvalho

EDITOR EXECUTIVO
Nathan Albano Valente

AUTORES DO LIVRO
Inês Guadalupe
Juan Carlos Z. Aguilar

PRODUÇÃO EDITORIAL
Seven Publicações Ltda

EDIÇÃO DE ARTE
Evellyn Thais de Souza

EDIÇÃO DE TEXTO
Natan Bones Petitemberte

BIBLIOTECÁRIA
Bruna Heller

IMAGENS DE CAPA
Evellyn Thais de Souza

Dados Internacionais de Catalogação na Publicação (CIP)
(Câmara Brasileira do Livro, SP, Brasil)

G897i
Guadalupe, Inês.
Inteligência Artificial no Ensino Médio - Desenvolvendo o Pensamento Computacional através da Rede Neural Perceptron [recurso eletrônico] / Inês Guadalupe, Juan Carlos Z. Aguilar. – São José dos Pinhais, PR: Seven Editora, 2025.
Dados eletrônicos (1 PDF).

Inclui bibliografia.
ISBN 978-65-6109-181-7

1. Inteligência artificial. 2. Ensino médio. 3. Pensamento computacional. I. Aguilar, Juan Carlos Z. II. Título.

CDU 004.8:373.5

Bruna Heller - Bibliotecária - CRB10/2348

Índices para catálogo sistemático:

CDU: Inteligência artificial 004.8
CDU: Escola de ensino médio 373.5

DOI: 10.56238/livrosindi202523-001

Seven Publicações Ltda
CNPJ: 43.789.355/0001-14
editora@sevenevents.com.br
São José dos Pinhais/PR

REALIZAÇÃO:

SEVEN
publicações acadêmicas

ACESSE NOSSO CATÁLOGO!

WWW.SEVENPUBLI.COM

CONECTANDO O **PESQUISADOR** E A **CIÊNCIA** EM UM SÓ CLIQUE.