Universidade Federal de São João del Rei
PROFMAT - Mestrado Profissional em Matemática em Rede Nacional

Inês Guadalupe

A Integração da Modelagem Matemática com as Redes Neurais Artificiais: Uma Proposta para Itinerário Formativo

São João del Rei
2024

Inês Guadalupe

A INTEGRAÇÃO DA MODELAGEM MATEMÁTICA COM AS REDES NEURAIS ARTIFICIAIS: UMA PROPOSTA PARA ITINERÁRIO FORMATIVO

Dissertação apresentada à Universidade Federal de São João del Rei, Campus Santo Antônio, como parte das exigências do Programa de Pós-Graduação Mestrado Profissional em Matemática em Rede Nacional, para obter o título de Mestre.

Orientador
Juan Carlos Zavaleta Aguilar

São João del Rei
2024

Ficha catalográfica elaborada pela Divisão de Biblioteca (DIBIB)
e Núcleo de Tecnologia da Informação (NTINF) da UFSJ,
com os dados fornecidos pelo(a) autor(a)

897i Guadalupe, Inês Maria Maia Pinto de.
A Integração da Modelagem Matemática com as Redes Neurais Artificiais : Uma Proposta para Itinerário Formativo / Inês Maria Maia Pinto de Guadalupe ; orientador Juan Carlos Zavaleta Aguilar. -- São João del-Rei, 2025.
153 p.

Dissertação (Mestrado - Programa de Mestrado Profissional em Matemática em Rede Nacional - PROFMAT) -- Universidade Federal de São João del Rei, 2025.

1. Modelagem Matemática. 2. Redes Neurais Artificiais. 3. Pensamento Computacional. 4. Sequência Didática. 5. Itinerário Formativo. I. Aguilar, Juan Carlos Zavaleta, orient. II. Título.

Inês Guadalupe

A INTEGRAÇÃO DA MODELAGEM MATEMÁTICA COM AS REDES NEURAIS ARTIFICIAIS: UMA PROPOSTA PARA ITINERÁRIO FORMATIVO

Dissertação apresentada à Universidade Federal de São João del Rei, Campus Santo Antônio, como parte das exigências do Programa de Pós-Graduação Mestrado Profissional em Matemática em Rede Nacional, para obter o título de Mestre.

Aprovada: 06 de dezembro de 2024.

Juan Carlos Zavaleta Aguilar
(Orientador)

Jorge Andrés Julca Avila - UFSJ
(Membro Interno do PROFMAT)

Estaner Claro Romão - USP
(Membro Externo)

São João del Rei
2024

Dedico este trabalho a todos os professores que acreditam, tal como eu, que a constante busca pela melhoria de suas práticas é essencial à excelência no ensino.

Agradecimentos

Ao ilustre Prof. Dr. Juan Carlos Zavaleta Aguilar, quero expressar minha mais sincera gratidão pela orientação sólida, dedicação e paciência. Suas orientações foram fundamentais para o sucesso deste trabalho.

Expresso também minha profunda gratidão aos professores do Mestrado, José Angel Dávalos Chuquipoma, Carla Regina Guimarães Brighenti, Francinildo Nobre Ferreira, Jorge Andrés Julca Avila e Viviane Pardini Valério. Cada um de vocês contribuiu de maneira única para o meu aprendizado e crescimento acadêmico.

Quero estender meus agradecimentos aos colegas Éder Tostes, Fátima Aguilar e Lucas Meneses. A jornada acadêmica foi enriquecida pela presença de vocês, e a amizade compartilhada tornou os desafios mais leves e as conquistas mais significativas.

À minha mãe, irmãos, marido, filhos e nora, manifesto minha profunda gratidão pelo apoio inabalável que tornou possível a conclusão deste trabalho.

À Escola Estadual Coronel Xavier Chaves e à Escola Municipal Sebastião Patrício Pinto, reconheço a importância fundamental no desenvolvimento deste trabalho. A colaboração e apoio destas instituições foram cruciais para a condução da pesquisa.

Aos membros da Banca Examinadora, por aceitarem o convite de avaliar o resultado deste trabalho.

Agradeço a todos que, de alguma forma, contribuíram para este trabalho. Este TCC é resultado de um esforço coletivo, e estou profundamente grata por cada pessoa que fez parte desta jornada.

Resumo

Este Trabalho de Conclusão de Curso (TCC) visa integrar a modelagem matemática (MM) com redes neurais artificiais (RNA) no ensino médio (EM), promovendo o desenvolvimento do pensamento computacional (PCO). A proposta foi aplicada em uma turma do 3º ano de uma escola pública de tempo integral, por meio de uma sequência didática envolvendo onze aulas. Nestas aulas, os alunos exploraram conceitos de Inteligência Artificial (IA) e RNA, vinculados ao conteúdo matemático e alinhados às diretrizes da Base Nacional Comum Curricular (BNCC). Utilizou-se uma metodologia mista, que combinou abordagens qualitativas e quantitativas; questionários iniciais avaliaram o conhecimento prévio e o interesse dos estudantes, enquanto atividades práticas e discussões interativas, como comparações entre neurônios biológicos e artificiais e o uso de programação no Google Colab, enriqueceram a experiência de aprendizado. A linguagem de programação Python facilitou a aplicação dos conceitos de RNA, como funções de ativação, ajuste de pesos e reconhecimento de padrões. Observou-se o desenvolvimento do PCO, especialmente nas habilidades de decomposição e abstração, à medida que os alunos acompanhavam a aplicação de algoritmos simples para estruturar problemas matemáticos. Embora houvesse inicialmente receio quanto ao uso de funções, os estudantes acabaram compreendendo a simplicidade lógica desses conceitos, o que aumentou o engajamento nas atividades práticas. Conclui-se que a sequência didática foi eficaz em introduzir os alunos ao campo das RNA, promovendo conexões entre conceitos matemáticos e tecnológicos e despertando o interesse pelo mundo da programação e suas aplicações na resolução de problemas complexos. Com o devido ajuste no tempo de cada etapa e um foco maior em atividades práticas, essa abordagem tem grande potencial para aproximar os estudantes das novas tecnologias e fortalecer o desenvolvimento do PCO.

Palavras-chave: : Redes Neurais Artificiais. Modelagem Matemática. Pensamento Computacional. Sequência Didática. Itinerário Formativo.

Abstract

This thesis aims to integrate mathematical modeling (MM) with artificial neural networks (ANN) in high school education, promoting the development of computational thinking (CT). The proposal was implemented in a third-year class at a full-time public high school through a didactic sequence of eleven lessons, where students explored concepts of Artificial Intelligence (AI) and ANN linked to mathematical content and aligned with the Brazilian National Common Core Curriculum (BNCC). A mixed-methods approach was employed, combining qualitative and quantitative strategies; initial questionnaires assessed students' prior knowledge and interest, while practical activities and interactive discussions, such as comparing biological and artificial neurons and programming exercises in Google Colab, enriched the learning experience. The Python programming language facilitated the application of ANN concepts, including activation functions, weight adjustment, and pattern recognition. CT development was observed, particularly in decomposition and abstraction skills, as students followed simple algorithmic applications to structure mathematical problems. Although students initially showed apprehension about functions, they gradually grasped their logical simplicity, which increased engagement in practical activities. The didactic sequence effectively introduced students to the field of ANN, fostering connections between mathematical and technological concepts and sparking interest in programming and its applications in solving complex problems. With adjustments in the timing of each stage and a greater emphasis on hands-on activities, this approach has significant potential to connect students with emerging technologies and strengthen CT development.

Keywords: : Artificial Neural Networks. Mathematical Modeling. Computational Thinking. Didactic Sequence. Formative Itinerary.

Lista de Símbolos

Símbolos e notações utilizadas neste trabalho:

η Letra grega Eta

σ Letra grega Sigma

Σ Símbolo de Somatório

$=$ Igual

$>$ Maior

$\geq$ Maior ou igual

$<$ Menor

$\leq$ Menor ou igual

$\wedge$ Conjunção lógica (and)

$\vee$ Disjunção lógica (or)

$\underline{V}$ Disjunção exclusiva (xor)

$\rightarrow$ Implicação lógica

$\neg$ Negação lógica

Lista de Abreviaturas

Abreviaturas utilizadas neste trabalho:

ABNT	Associação Brasileira de Normas Técnicas
BNCC	Base Nacional Comum Curricular
CSA	Campus Santo Antônio
EECXC	Escola Estadual Coronel Xavier Chaves
ENEM	Exame Nacional do Ensino Médio
EQM	Erro Quadrático Médio
EM	Ensino Médio
IBM	International Business Machines
INEP	Instituto Nacional de Pesquisa Anísio Teixeira
MLP	Multi-Layer Perceptron
MM	Modelagem Matemática
MMO	Modelo Matemático
NEM	Novo Ensino Médio
OCDE	Organização para a Cooperação e Desenvolvimento Econômico
PCO	Pensamento Computacional
PISA	Programa Internacional de Avaliação de Estudantes
PROFMAT	Mestrado Profissional em Matemática em Rede Nacional
RNA	Rede Neural Artificial
SAEB	Sistema de Avaliação da Educação Básica
TCC	Trabalho de Conclusão de Curso
UFSJ	Universidade Federal de São João del Rei

Lista de Figuras

Lista de Tabelas

Sumário

1 INTRODUÇÃO

A educação matemática desempenha um papel fundamental na formação dos estudantes, desenvolvendo habilidades analíticas, lógicas e críticas essenciais para enfrentar desafios tanto nas carreiras quanto na vida cotidiana. No entanto, o ensino tradicional muitas vezes parece abstrato e distante da realidade, o que pode desmotivar e dificultar a compreensão da disciplina.

No Brasil, a educação enfrenta desafios significativos, especialmente em termos de qualidade e equidade. Avaliações externas, como o Sistema de Avaliação da Educação Básica (SAEB), destacam fragilidades, como a evasão escolar e o baixo desempenho em matemática [1]. Dados do Censo Escolar de 2023 indicam um aumento preocupante na taxa de evasão escolar [2], enquanto os resultados do Programa Internacional de Avaliação de Estudantes (PISA) mostram que a maioria dos estudantes brasileiros não alcança o nível mínimo de proficiência em matemática [3].

Diante desse cenário, é imperativo que os professores repensem suas práticas profissionais. A modelagem matemática (MM) surge como uma abordagem pedagógica importante, aproximando a matemática do mundo real e tornando-a mais acessível e relevante para os estudantes.

Este TCC propõe uma sequência didática a ser aplicada em uma turma do 3º ano do ensino médio (EM) da Escola Estadual Coronel Xavier Chaves (EECXC). O objetivo é integrar a Modelagem Matemática (MM) e as Redes Neurais Artificiais (RNA) no contexto do Pensamento Computacional (PCO), visando facilitar a compreensão da Inteligência Artificial (IA) a partir de uma perspectiva matemática. A escolha da EECXC se justifica pelo perfil de seus alunos, que refletem a realidade da educação pública brasileira, e pelo interesse da instituição em promover inovações pedagógicas no ensino de matemática, buscando, de forma efetiva, melhorar as práticas nos Itinerários Formativos do Novo Ensino Médio (NEM).

A opção por este tema é motivada pela crescente relevância das RNA e a importância

dos fundamentos matemáticos na compreensão da IA e do aprendizado de máquina. Essas redes, inspiradas no funcionamento do cérebro humano, oferecem uma abordagem eficaz para resolver problemas complexos.

A introdução dos conceitos de RNA no currículo de matemática do EM não só torna a disciplina mais relevante para o mundo atual, mas também prepara os estudantes para futuras oportunidades [4]. No entanto, para que essa abordagem seja implementada de forma eficaz, é essencial contar com professores bem capacitados e motivados.

Assim, este trabalho visa desenvolver uma sequência didática, com atividades e recursos práticos para os professores dos Itinerários Formativos do EM, abordando conectivos lógicos (AND, OR e XOR) e demonstrando a construção de uma RNA para prever resultados em situações simples. A proposta inclui o uso da linguagem de programação Python para implementar esses conceitos, proporcionando uma compreensão prática e teórica das RNA.

Na literatura especializada podem ser encontrados diversos trabalhos relacionados com os temas tratados nessa dissertação, entre eles destacam-se os trabalhos de Inês Furtado, que apresenta uma abordagem prática para sala de aula envolvendo RNA [5], e outras pesquisas relevantes como "Redes Neurais no Ensino Básico", da Sociedade Brasileira de Matemática, que explora o uso de redes neurais como recurso pedagógico inovador para o ensino básico, oferecendo formação continuada para professores [6]. Outro estudo, intitulado "Redes Neurais Artificiais: Ideias Básicas no Ensino Médio", descreve estratégias para introduzir conceitos de RNA aos alunos do EM de forma acessível, utilizando álgebra linear e programação [7]. Ainda, o trabalho "Utilização de Redes Neurais Artificiais como Estímulo ao Aprendizado de Matemática", que propõe o uso de RNA para despertar o interesse dos estudantes pela matemática, destacando seu potencial pedagógico [7]. Essas contribuições teóricas e práticas enriquecem a elaboração da sequência didática proposta neste trabalho, fornecendo as bases para sua implementação.

Ao explorar a conexão entre MM e RNA na educação matemática, inserindo o PCO no EM, este trabalho busca investigar como o PCO, por meio do conhecimento das RNA, pode contribuir para o ensino de matemática e suas tecnologias, desenvolvendo competências e habilidades previstas na Base Nacional Comum Curricular (BNCC).

O TCC está estruturado em cinco capítulos: o Capítulo 1 introduz o trabalho a partir dos seus objetivos; o Capítulo 2 apresenta os referenciais teóricos, conectando

MM, RNA e PCO; o Capítulo 3 detalha a proposta da sequência didática; o Capítulo 4 discute a implementação desta sequência didática; e o Capítulo 5 traz os resultados, as considerações finais e potenciais desdobramentos.

2 REFERENCIAIS TEÓRICOS

2.1 Modelagem Matemática

A MM é uma abordagem pedagógica que visa ensinar matemática por meio de sua aplicação em contextos do cotidiano no âmbito da educação básica. Segundo Burak, esta metodologia teve origem na década de 80, a partir de estudos que objetivavam uma melhoria do ensino de matemática. Nesta metodologia, em vez de apenas aprender fórmulas e teorias, os alunos são estimulados a resolver problemas reais usando conceitos matemáticos. Desta forma, são levados a compreender e agir sobre o mundo ao qual pertence [8].

A obra deste autor aborda a incorporação da MM na prática de ensino, destacando a importância de identificar situações práticas, coletar dados relevantes, traduzir problemas para o contexto matemático, criar e resolver modelos, analisar resultados, refletir criticamente sobre os modelos e comunicar os resultados de forma clara e organizada.

De acordo com Bassanezi (2002), o ensino da matemática nas escolas é conduzido de forma descolada da realidade e do processo histórico de sua construção, evidenciado nesta observação:

> O desenvolvimento de novas teorias matemáticas e suas apresentações como algo acabado e completo acabaram conduzindo seu ensino nas escolas de maneira desvinculada da realidade, e mesmo do processo histórico de construção da matemática. Assim é que um teorema é ensinado, seguindo o seguinte esquema: "enunciado – demonstração – aplicação", quando de fato o que poderia ser feito é sua construção na ordem inversa, isto é, sua motivação, a formulação de hipóteses, a validação e novos questionamentos, e finalmente seu enunciado. Estaríamos assim reinventando o resultado juntamente com os alunos, seguindo o processo da modelagem e conjugando verdadeiramente o binômio ensino-aprendizagem [9, p. 36].

Devemos incorporar a MM na prática de ensino de várias maneiras, a partir de diferentes abordagens mas seguindo alguns procedimentos, conforme [10]:

- Identificando um problema do mundo real que envolva situações práticas e que possam ser resolvidos usando conceitos matemáticos específicos. Isso ajuda os estudantes a ver a aplicação prática da matemática em suas vidas.
- Coletando dados relevantes para o problema. Isso pode incluir atividades de pesquisa, experimentação ou análise de conjuntos de dados existentes.
- Traduzindo um problema do mundo real para o contexto matemático, identificando as variáveis envolvidas e as relações entre elas, podendo fazer suposições ou simplificações para tornar o problema mais acessível. Isso envolve traduzir a linguagem do problema para a linguagem matemática.
- Criando modelos matemáticos que representem o problema ou seja, as relações entre as variáveis identificadas. Isso pode envolver o uso de equações, inequações, sistemas de equações, funções, tabelas, gráficos, etc.
- Resolvendo equações ou manipulando as representações matemáticas criadas para encontrar uma solução para o problema. Isso pode envolver o uso de cálculos matemáticos, software de simulação ou outras ferramentas computacionais.
- Analisando os resultados do MMO. Isso significa interpretar as soluções em termos do problema original e verificar se os resultados fazem sentido na situação do mundo real. Discutir se a solução é aplicável ao problema original e quais são as implicações práticas das descobertas.
- Fazendo uma reflexão crítica sobre os modelos criados; considerando a validade de suas suposições, a precisão dos dados coletados e a relevância de suas soluções.
- Comunicando os resultados de forma clara e organizada, explicando o problema, o MMO desenvolvido, os métodos utilizados e as instruções apresentadas, por meio de relatórios, apresentações ou modelos visuais.

Pereira afirma que:

> Com modelos variando em termos de formalidade, clareza, riqueza de detalhes e relevância, dependendo de sua função e objetivo, segundo diferentes autores, o processo de modelação formalizado ao longo da história foi fundamental para estruturar o conhecimento humano e simplificar a compreensão de fenômenos complexos. [10, p. 56].

Dessa forma, a MM frequentemente envolve habilidades multidisciplinares, pois o processo de modelagem exige a integração de conhecimentos de diversas áreas, como a pesquisa científica, o uso de tecnologias, e a comunicação de resultados. Além disso, os estudantes podem se beneficiar ao trabalhar em equipe, combinando diferentes perspectivas para criar modelos mais abrangentes e eficazes. Assim, a MM se revela uma ferramenta versátil, útil em diversas disciplinas e áreas, ajudando a desenvolver habilidades essenciais, como pensamento crítico, resolução de problemas e comunicação matemática.

2.2 A Inteligência Artificial

A IA se refere à simulação de processos de inteligência humana em sistemas computacionais, permitindo que máquinas executem funções cognitivas como aprendizado, raciocínio, identificação de padrões, resolução de problemas e interpretação da linguagem natural. Essas capacidades possibilitam que os sistemas realizem tarefas complexas de forma semelhante ao raciocínio humano [11]. Este campo é interdisciplinar, combinando ciência da computação, matemática, estatística, engenharia e ciências cognitivas.

Os principais campos da IA abrangem diversas áreas interconectadas e com aplicações variadas. Segundo Russell e Norvig (2016), esses campos incluem: aprendizado de máquina, que permite a criação de sistemas capazes de melhorar seu desempenho com base em dados; RNA e aprendizado profundo, focados em simular o funcionamento do cérebro humano para resolver problemas complexos; visão computacional, para a análise de imagens e vídeos; processamento de linguagem natural, que envolve a interpretação e geração de linguagem humana; robótica, dedicada à construção de agentes físicos autônomos; e inteligência artificial generalizada, que busca desenvolver sistemas com capacidades cognitivas comparáveis às humanas. Além disso, destacam-se o processamento de sinais, voltado para o tratamento de informações em áudio e outros formatos; sistemas especialistas, projetados para tomar decisões em áreas específicas; e a lógica fuzzy, que lida com incertezas em sistemas complexos [12].

Aprendizado de Máquina se concentra no desenvolvimento de modelos que aprendem padrões a partir de dados, sem a necessidade de programação explícita de regras. RNA são uma abordagem dentro do Aprendizado de Máquina, imitando o funcionamento do cérebro humano. Deep Learning, uma forma avançada de RNA com múltiplas camadas, tem resolvido problemas complexos com sucesso [13].

As RNA são fundamentais em muitos modelos de IA, especialmente aqueles baseados em aprendizado profundo.Esse conceito foi introduzido por Frank Rosenblatt na década de 1950, com o desenvolvimento do perceptron, que visava simular o comportamento de um neurônio biológico e formar a base para o estudo das redes neurais modernas [14].

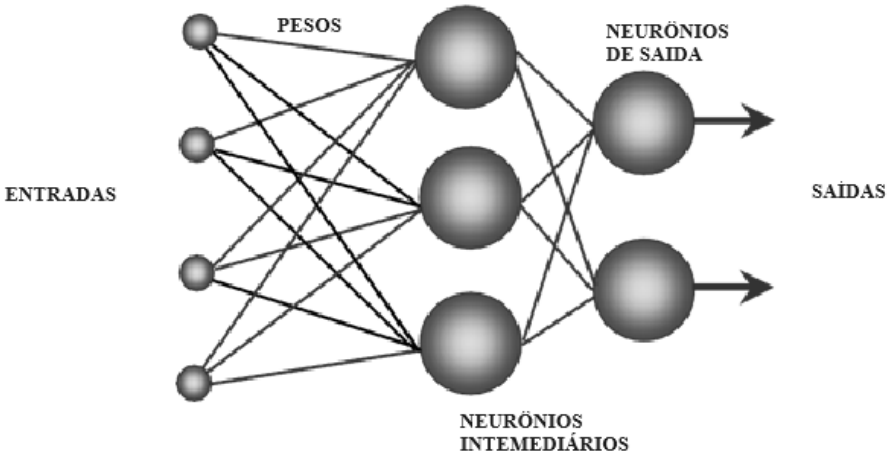
2.2.1 As Redes Neurais Artificiais

Muitos dos avanços recentes em IA, incluindo reconhecimento de voz, tradução automática e jogos de estratégia, são impulsionados por modelos baseados em RNA. Esta tecnologia encontra aplicação em diversos setores, como reconhecimento de padrões, diagnósticos médicos, aplicações financeiras, mineração de dados, jogos, reconhecimento de voz, carros autônomos e tradução de idiomas [11].

As RNA são estruturas matemáticas inspiradas no funcionamento do cérebro humano, compostas por unidades chamadas neurônios artificiais, nós ou perceptrons, organizadas em camadas. Esse conceito foi introduzido por Frank Rosenblatt na década de 1950, com o desenvolvimento do perceptron, que visava simular o comportamento de um neurônio biológico e formar a base para o estudo das redes neurais modernas [14].

Através da Figura 2.1, pode-se observar as três principais camadas de uma rede neural: entrada, oculta e saída.

Figura 2.1: Estrutura de uma RNA



Fonte: Dados da referência [5, p. 11].

Maria Inês Furtado (2019) descreve as camadas de uma RNA, detalhando suas funções e o papel crucial de cada camada, desde a entrada até a saída, para o processamento de dados e aprendizagem [5].

• Camada de entrada: é a primeira camada da rede, onde os dados do problema são introduzidos. Estes dados de entrada podem ser números, pixels de uma imagem, palavras em um texto, etc. A camada de entrada em uma RNA tem papel fundamental de receber e representar os dados iniciais do problema em formato numérico, preparando-os para serem processados pelas camadas internas da rede.

Cada unidade na camada de entrada é chamada de neurônio. Embora esses neurônios não realizem operações complexas como os neurônios nas camadas internas, são essenciais para iniciar o processo de propagação para as camadas subsequentes, onde operações mais complexas ocorrem.

• Camadas Ocultas: são camadas intermediárias entre a camada de entrada e a camada de saída. Cada camada oculta captura características mais complexas e abstratas à medida que se aprofunda na rede. O número de camadas ocultas e o número de neurônios em cada camada depende da arquitetura da rede neural e das necessidades da tarefa. Desta forma, cada nó em uma camada oculta combina as informações da camada anterior de maneira ponderada, utilizando pesos que são ajustados durante o treinamento da rede; representa uma característica ou variável do dado de entrada e está conectado a todos os nós da camada anterior.

Um nó realiza cálculos complexos com base nas entradas e emite um sinal de saída. Nestas camadas ocorre a maior parte do processamento da informação: cada nó realiza uma combinação linear das entradas ponderadas pelos pesos. Em seguida, essa combinação passa por uma função de ativação, como a função sigmoid, tangente hiperbólica, ReLU (Rectified Linear Unit) ou função degrau.

• Camada de Saída: é a última camada da rede, onde a resposta é produzida. O número de unidades fundamentais que compõem a estrutura da rede (nós) nesta camada depende do tipo de problema que a rede está resolvendo. Por exemplo, em um problema de classificação binária, haveria dois nós, a saber, um nó de saída associado à classe 0 e outro à classe 1.

O funcionamento de uma rede neural envolve também três etapas principais: propagação para frente (propagação direta), cálculo do erro e retropropagação. Durante a fase de propagação para frente, os dados de entrada são passados pela rede, camada por camada, até a camada de saída.

Após a combinação linear das entradas, seguida por uma função de ativação, cada

entrada é multiplicada pelo seu respectivo peso, e esses produtos são somados, resultando na aplicação do somatório ponderado.

Conforme descrito por Haykin, os pesos iniciais de uma rede neural são definidos aleatoriamente, com valores geralmente entre -1 e 1. Durante o treinamento da rede, esses pesos são ajustados iterativamente com o objetivo de minimizar o erro nas saídas. Esse processo de ajuste dos pesos, realizado através de algoritmos de aprendizado, como o gradiente descendente, é chamado de treinamento da rede [11].

A função de ativação apresenta não-linearidade na rede, permitindo que ela aprenda padrões complexos nos dados. Em um modelo linear, como a regressão linear simples, a relação entre as variáveis de entrada e a saída é representada por uma linha reta ou um hiperplano. Isso significa que a saída é uma combinação linear das entradas, ponderadas pelos pesos correspondentes.

Além disso, a escolha adequada da função de ativação é fundamental para o desempenho de uma RNA e para sua capacidade de aprender e generalizar a partir dos dados. Para ser utilizada em redes neurais, uma função matemática deve atender a duas condições principais: ser contínua e diferenciável. 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 por algoritmos de otimização baseados em gradientes. A diferenciabilidade é igualmente crucial, pois algoritmos como o gradiente descendente dependem do cálculo das derivadas da função de ativação para ajustar os pesos das conexões. Funções com descontinuidades ou pontos onde a derivada não existe comprometem o processo de aprendizado, tornando inviável o uso de métodos baseados em gradientes [11].

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. 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 [5].

Em resumo, as RNA são uma técnica específica da IA, sendo um método eficiente

para abordar questões complexas que envolvem reconhecimento de padrões e aprendizado a partir do treinamento dos dados. Estas redes são construídas a partir de neurônios artificiais que, por meio de ponderações e funções de ativação, possibilitam a tomada de decisões e aprendizado em cenários variados. O desenvolvimento contínuo nesse campo está expandindo constantemente os limites que as redes neurais podem alcançar, com aplicações surpreendentes em nosso mundo.

Os neurônios biológicos e os neurônios artificiais são componentes fundamentais nos sistemas nervosos biológicos e nas redes neurais artificiais, respectivamente. Eles diferenciam-se em alguns aspectos, cada um sendo adaptado para seus respectivos contextos de aplicação.

O Neurônio biológico é encontrado no sistema nervoso dos organismos vivos, incluindo humanos e animais. Os neurônios biológicos são células especializadas que transmitem sinais elétricos e químicos no cérebro e no sistema nervoso. Formam um sistema muito complexo, com estruturas especializadas, como dendritos, axônios, sinapses e organelas. Realiza uma variedade de funções, incluindo a transmissão de sinais elétricos e a liberação de neurotransmissores. A comunicação entre os neurônios ocorre através de sinais elétricos (impulsos nervosos) ao longo do axônio e a liberação de neurotransmissores nas sinapses. Exibe plasticidade sináptica e pode se adaptar a mudanças no ambiente ou no aprendizado, modificando a força das conexões sinápticas, com consumo eficiente de energia e altamente adaptado para o ambiente biológico. Apresenta-se muito pequeno, com dimensões microscópicas [15].

Um neurônio artificial é criado para simular o funcionamento dos neurônios biológicos em modelos computacionais. Estes neurônios são construídos com o propósito de realizar operações em redes neurais artificiais usadas em IA e aprendizado de máquina. Geralmente são simplificados em comparação com neurônios biológicos. Focam em operações matemáticas, como a soma ponderada de entradas e a aplicação de uma função de ativação. A comunicação entre eles é realizada por meio da passagem de valores (ou ativações) ao longo das conexões ponderadas entre neurônios em uma rede neural. A plasticidade pode ser incorporada em certos modelos, mas não é necessariamente uma característica padrão. Em algumas aplicações, as conexões podem ser ajustadas durante o treinamento. Em geral, consome mais energia e pode não ser tão eficiente quanto o neurônio biológico. São representados por parâmetros e valores armazenados em memória

de computador, operando em uma escala maior [16].

Para entender melhor o funcionamento de uma RNA, é essencial compreender a base da sua construção: o neurônio artificial.

2.2.2 Como surgiram as Redes Neurais Artificiais

A história das RNA remonta ao meio do século XX e é marcada pela aplicação de princípios matemáticos fundamentais. Esses princípios culminaram na criação do neurônio artificial, resultado de um processo histórico significativo [16].

Em 1943, o neurocientista Warren McCulloch e o matemático Walter Pitts publicaram um artigo seminal que introduziu o conceito de um neurônio artificial. O modelo que eles desenvolveram se assemelha a um dispositivo lógico binário, representando o neurônio como uma função matemática simples. Utilizando circuitos elétricos, essa abordagem permite que o neurônio seja ativado ou desativado com base em entradas ponderadas, estabelecendo as bases para a compreensão das operações neurais e a construção de redes artificiais.

Em 1949, Donald Hebb, em sua obra "The Organization of Behavior", enfatizou que os caminhos neurais são fortalecidos através da prática, o que contribuiu significativamente para os estudos sobre RNA. Hebb propôs que, quando dois neurônios disparam simultaneamente, a conexão entre eles se torna mais robusta, uma ideia que ficou conhecida como a regra de Hebb.

O Algoritmo de Hebb representou uma contribuição pioneira para a compreensão de como a aprendizagem ocorre no cérebro, influenciando o desenvolvimento de modelos de aprendizado em redes neurais artificiais. A premissa fundamental do algoritmo é que, se dois neurônios forem ativados ao mesmo tempo, a conexão sináptica entre eles será fortalecida. No entanto, a regra de Hebb é uma simplificação que não abrange todas as complexidades da aprendizagem neural. Ao longo do tempo, foram desenvolvidas outras regras mais refinadas e sofisticadas para lidar com os diferentes aspectos da aprendizagem em redes neurais [17].

Na década de 1950, Nathaniel Rochester tentou simular uma rede neural nos laboratórios de pesquisa da IBM, marcando o início do desenvolvimento dessas redes. Após o Projeto Dartmouth em 1956, John von Neumann propôs imitar funções neurais usando relés telegráficos. Em 1957, Frank Rosenblatt, inspirado pelo funcionamento

do olho de uma mosca, desenvolveu o perceptron, nome derivado do termo percepção em inglês. Esse modelo simplificado de neurônio artificial buscava replicar algumas funcionalidades do cérebro humano para realizar tarefas de classificação binária. Embora eficaz em classificar entradas, o perceptron apresentava limitações significativas.

Em 1959, Bernard Widrow e Marcian Hoff criaram os modelos ADALINE e MADALINE. ADALINE previa o próximo bit em padrões binários, enquanto o MADALINE foi a primeira RNA aplicada a um problema real, eliminando ecos em linhas telefônicas. Nesse mesmo ano, Widrow e Hoff desenvolveram a Regra Delta, também conhecida como a regra de aprendizado de Widrow-Hoff, aplicada no Adaline. A Regra Delta consiste em ajustar os pesos das conexões entre neurônios para minimizar a diferença entre a saída desejada e a saída real da rede, ou seja, o erro [5].

Esses sucessos iniciais, no entanto, levaram a expectativas exageradas, resultando em desapontamento durante o inverno da IA, até 1981, como definiu John McCarthy. Este período, que começou no final dos anos 1970 e se estendeu até o início dos anos 1980, foi marcado por uma redução significativa no financiamento e no interesse pela pesquisa em IA, incluindo RNA, conforme destaca Maria Inês Furtado, dizendo que:

> Um período de pesquisa silenciosa seguiu-se entre 1967 e 1982, com poucas publicações devido aos fatos ocorridos anteriormente. Entretanto, aqueles que pesquisavam nesta época, e todos os que se seguiram no decorrer destes anos conseguiram novamente estabelecer um campo concreto para o renascimento da área [5, p. 3].

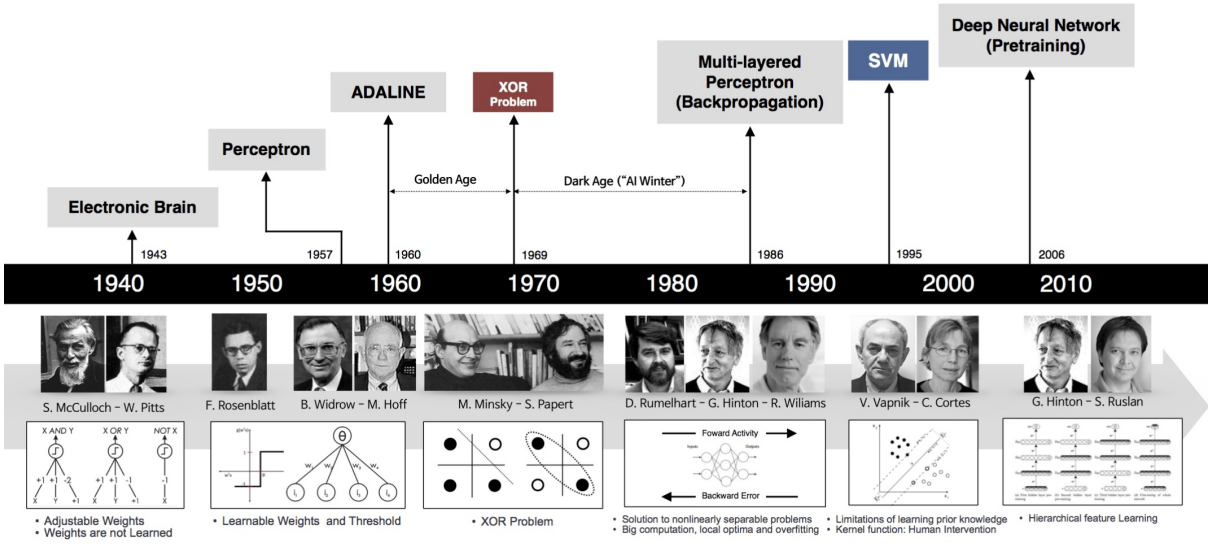
Em 1982, eventos como a apresentação de John Hopfield reacenderam o interesse, seguido pela criação de conferências e o desenvolvimento das redes de retropropagação em 1986. A década de 1990 viu as RNA demonstrarem seu potencial em várias aplicações, impulsionadas pelo aumento da capacidade de computação e conjuntos de dados mais amplos. O termo deep learning ganhou destaque, destacando o uso de redes neurais profundas com múltiplas camadas ocultas.

As RNA modernas são alicerçadas em princípios matemáticos sólidos, como álgebra linear, cálculo diferencial e otimização numérica. A pesquisa em RNA continua a explorar métodos avançados, arquiteturas complexas e aplicações em diversos setores.

A Figura 2.2, disponível em <https://www.deeplearningbook.com.br/uma-breve-historia-das-redes-neurais-artificiais/>, ilustra alguns dos marcos importantes no desen-

volvimento das RNA, destacando os principais avanços e conquistas ao longo do tempo.

Figura 2.2: Marcos no desenvolvimento das redes neurais



Fonte: [14],
https://www.deeplearningbook.com.br/uma-breve-historia-das-redes-neurais-artificiais/.

A história das RNA está inseparavelmente ligada à aplicação de princípios matemáticos, desde os neurônios artificiais iniciais até os avanços em redes neurais profundas, salientando o papel fundamental da matemática na formulação, treinamento e compreensão desses modelos [11].

2.2.3 O que é um Perceptron

Para entender o que é um perceptron, é essencial compreender a função básica dos neurônios no cérebro, uma vez que os perceptrons são inspirados na estrutura e no funcionamento dos neurônios biológicos. Maria Inês Furtado discute essa relação, destacando a relevância dos perceptrons no contexto educacional e computacional [5].

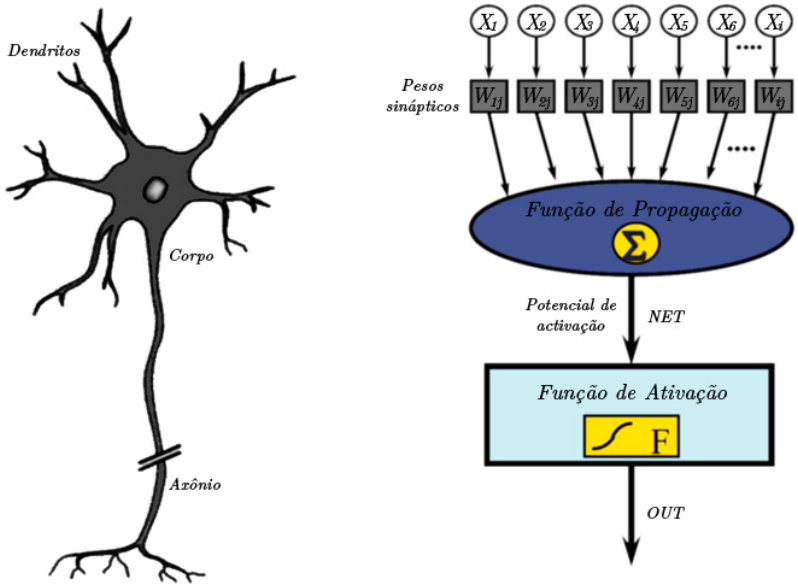
Os neurônios são células fundamentais do sistema nervoso, responsáveis por transmitir impulsos elétricos e sinais químicos ao cérebro. Cada neurônio é composto por três partes principais: os dendritos, o corpo celular e o axônio. As informações são geralmente recebidas pelos dendritos e pelo corpo celular e são transmitidas pelo axônio. Esses neurônios servem como a unidade básica de processamento de informações no cérebro. Quando um neurônio recebe estímulos elétricos de outros neurônios, ele decide se deve gerar um sinal elétrico com base na regra do "tudo ou nada". Isso significa que o neurônio só dispara um sinal se a estimulação recebida superar um determinado limiar, valor limite

usado para decidir se um neurônio será ativado ou não; caso contrário, ele permanece inativo [15].

O perceptron é uma representação computacional simplificada de um neurônio. É um componente fundamental de muitos algoritmos de aprendizado de máquina e funciona como um modelo de rede neural de uma única camada, usado principalmente para tarefas de classificação linear.

Para ilustrar a relação entre os neurônios biológicos e os perceptrons, temos a Figura 2.3.

Figura 2.3: Comparação entre Neurônio Humano e Neurônio Artificial



Fonte: Adaptado de [18, p. 68].

A Figura 2.3 ilustra, à esquerda, um neurônio biológico, destacando suas principais partes: os dendritos, o corpo celular e o axônio. Esta imagem evidencia a complexidade do sistema nervoso humano e a interação dos neurônios no processamento e transmissão de informações. À direita, apresenta um neurônio artificial, ou perceptron. Este modelo simplificado serve como bloco de construção fundamental para redes neurais artificiais, capturando a essência do neurônio biológico de forma computacional, o que possibilita sua aplicação em algoritmos de aprendizado de máquina.

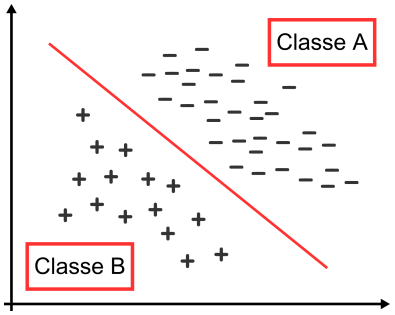
Segundo Silva, Spatti e Flauzino [19], o perceptron funciona como um classificador capaz de separar padrões em classes, desde que estas sejam linearmente separáveis.

A classificação linear refere-se à capacidade de dividir um conjunto de dados de

forma clara e precisa usando uma linha reta. Em um modelo linear, como a regressão linear simples, a relação entre as variáveis de entrada e a variável de saída é representada por uma linha reta ou um hiperplano. Isso implica que a saída é uma combinação linear das entradas, ponderadas pelos pesos correspondentes.

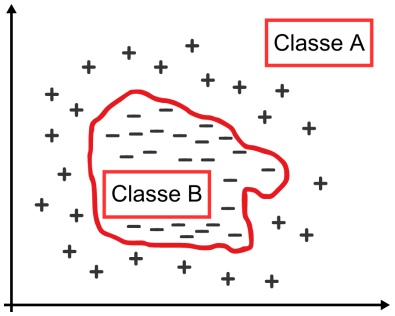
No contexto dos perceptrons e algoritmos de aprendizado de máquina, os dados são considerados linearmente separáveis se for possível traçar uma fronteira de decisão que separe duas classes de dados. Nas Figuras 2.4 e 2.5, ilustramos dados que são linearmente separáveis e dados que não são, respectivamente.

Figura 2.4: Dados Linearmente Separáveis



Fonte: Adaptado de [5, p. 48].

Figura 2.5: Dados Não Linearmente Separáveis



Fonte: Adaptado de [5, p. 49].

Se for possível encontrar uma linha (em duas dimensões), um plano (em três dimensões) ou um hiperplano (em mais de três dimensões) que separe as classes, os dados são considerados linearmente separáveis [16].

O perceptron utiliza entradas ponderadas e uma função de ativação para produzir saídas binárias. Ele é a unidade fundamental das redes neurais artificiais. É um modelo de neurônio simples, particularmente eficaz quando os dados são linearmente separáveis já que ele ajusta os pesos para encontrar a melhor linha reta que separa as duas classes ou melhor, para otimizar a capacidade de separação linear das classes. No entanto, apresenta

limitações quando os dados não são linearmente separáveis, e é aí que modelos mais avançados, como redes neurais mais profundas, tornam-se necessários para lidar com problemas mais complexos.

Segundo Haykin, um perceptron é um tipo de neurônio em redes neurais utilizado para realizar classificação binária. Este neurônio pode receber múltiplas entradas que são ponderadas para calcular uma saída:

$$(X_1,X_2,...,X_n) \tag{2.1}$$

Cada entrada é ponderada por um peso correspondente, definido, a princípio, aleatoriamente.

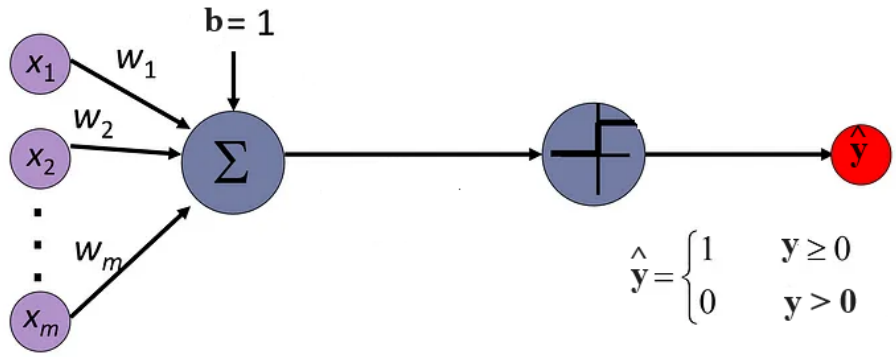
$$(W_1,W_2,...,W_n) \tag{2.2}$$

As entradas ponderadas são somadas (passam pela função soma, Σ) e por uma função de ativação (f), que pode introduzir não linearidade ao modelo.

A função de ativação determina se o neurônio deve ser ativado, com base na soma ponderada das entradas. A saída do neurônio é então determinada por esta função, podendo ser binária (0 ou 1) ou contínua, dependendo da escolha da função específica [11].

Na Figura 2.6, disponível em <https://www.deeplearningbook.com.br/funcao-de-ativacao/>, é ilustrado o funcionamento básico de um neurônio artificial dentro de uma rede neural: recepção de várias entradas $(x_1, x_2, ..., x_n)$, que são ponderadas por pesos correspondentes $(w_1, w_2, ..., w_n)$, refletindo a importância de cada entrada. Cada entrada é multiplicada por seu peso e, em seguida, os produtos dessas multiplicações são somados, formando uma soma ponderada ($\sum$). Esse valor somado é então ajustado por um parâmetro adicional conhecido como viés (b), que serve para deslocar a soma, permitindo que a rede ajuste a saída mesmo quando todas as entradas são zero. Após essa soma ajustada, a função de ativação é aplicada para introduzir a não linearidade, essencial para a capacidade da rede neural de aprender e modelar padrões complexos. A função de ativação determina o valor final da saída do neurônio e que, no caso do perceptron, pode ser uma função simples, como a função degrau.

Figura 2.6: Representação de um Perceptron



Fonte: Adaptado de [20],
https://www.deeplearningbook.com.br/funcao-de-ativacao/.

Em resumo, um perceptron é um algoritmo simples que aprende a tomar decisões com base nas entradas que recebe, ajustando seus pesos e viés para minimizar os erros e melhorar sua eficiência ao longo do tempo. Ele forma uma base para redes neurais mais complexas usadas em problemas de aprendizado de máquina. Eles são usados para resolver uma variedade de problemas de aprendizado de máquina, incluindo classificação, reconhecimento de padrões e outras aplicações específicas.

Segundo Haykin, uma rede neural pode ser vista como uma máquina adaptativa onde todos os neurônios artificiais são maciçamente interligados:

> Uma rede neural é um processador maciçamente paralelo distribuído constituído de unidades de processamento simples, que têm a propensão natural para armazenar conhecimento experimental e torná-lo disponível para uso. Ela se assemelha ao cérebro em dois aspectos: 1. O conhecimento é adquirido pela rede a partir de seu ambiente através de um processo de aprendizagem. 2. Forças de conexão entre neurônios, conhecidas como pesos sinápticos, são utilizadas para armazenar o conhecimento adquirido [11, p. 28].

A aprendizagem de uma RNA significa que ela se torna capaz de apresentar saídas adequadas a partir de novas entradas, ou seja, de dados que não estavam presentes durante o seu processo de treinamento. Isso implica que a rede ajusta seus pesos para minimizar os erros na saída, permitindo assim generalizar o conhecimento adquirido para novas situações.

Enquanto o treinamento é o processo de ajustar os pesos do perceptron com base nos exemplos fornecidos, a aprendizagem refere-se à capacidade do perceptron de se

tornar melhor na tarefa para a qual está sendo treinado à medida que mais exemplos são apresentados. Dessa forma, a rede neural aprimora sua performance e generalização com o tempo e a exposição a novos dados [11].

2.2.4 Redes Neurais Multicamadas

As redes neurais multicamadas, chamadas Multi-Layer Perceptron (MLP), representam uma evolução significativa em relação aos perceptrons simples, que consistem em apenas uma única camada de neurônios. As MLP são a base das RNA mais complexas utilizadas atualmente em diversas aplicações da IA [16].

Uma rede neural multicamadas também é composta por três tipos principais de camadas [21]:

1. **Camada de entrada**: Recebe os dados de entrada e os transmite para a próxima camada. Cada neurônio desta camada corresponde a uma característica do conjunto de dados.

2. **Camadas ocultas**: Localizadas entre a camada de entrada e a camada de saída, podem ser uma ou várias e são responsáveis por capturar padrões complexos nos dados. Cada neurônio nas camadas ocultas é conectado a todos os neurônios da camada anterior, e essa conexão é ponderada.

3. **Camada de saída**: Produz a saída final da rede, que pode ser uma classificação, uma previsão ou outra forma de resultado, dependendo do problema.

Cada conexão entre neurônios possui um peso associado, que é ajustado durante o treinamento da rede. O funcionamento básico de uma MLP pode ser descrito em três etapas principais [16]:

a. Propagação Direta (Forward Propagation): Os dados de entrada são propagados através da rede, camada por camada. Em cada neurônio, os dados são combinados de acordo com os pesos das conexões e uma função de ativação é aplicada para introduzir não-linearidade no modelo, permitindo que a rede capture relações complexas nos dados.

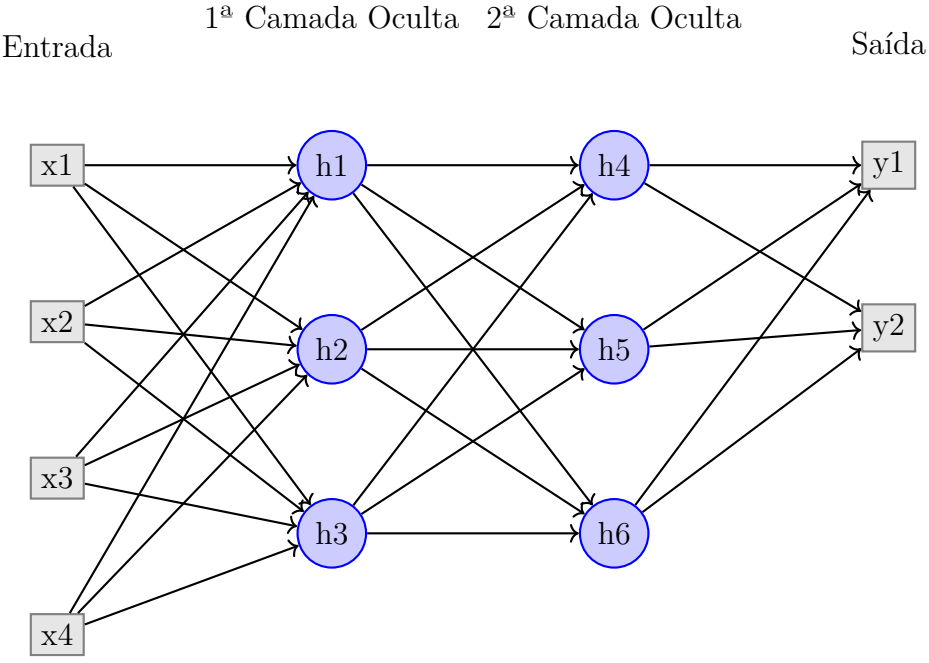
b. Erro e Retropropagação (Backpropagation): Após a propagação direta, o erro entre a saída prevista pela rede e a saída desejada é calculado. A retropropagação é um

algoritmo que ajusta os pesos das conexões de modo a minimizar esse erro. Ele faz isso propagando o erro de volta pela rede, a partir da camada de saída até a camada de entrada, ajustando os pesos de acordo com a contribuição de cada neurônio para o erro total.

c. Atualização dos Pesos: Os pesos são atualizados com base nas derivadas do erro em relação a cada peso, usando um método de otimização como o gradiente descendente. Este processo é repetido iterativamente em várias épocas até que a rede atinja um nível aceitável de desempenho.

As MLP são capazes de resolver problemas que os perceptrons simples não conseguem, como a classificação de dados não linearmente separáveis. Por exemplo, o famoso problema do XOR (ou exclusivo), que não pode ser resolvido por um perceptron de camada única, pode ser facilmente solucionado por uma rede multicamada com pelo menos uma camada oculta, conforme a Figura 2.7.

Figura 2.7: Estrutura de uma Rede Neural Multicamadas



Fonte: Elaborada pela autora.

Além disso, as redes multicamadas formam a base para arquiteturas mais avançadas, como redes convolucionais (CNNs) e redes recorrentes (RNNs), que são amplamente utilizadas em tarefas de visão computacional e processamento de linguagem natural, respectivamente [16].

As MLP têm uma ampla gama de aplicações, incluindo, mas não se limitando a:

- **Reconhecimento de Padrões**: Como reconhecimento facial, reconhecimento de voz e detecção de fraudes.
- **Previsão e Modelagem**: Como previsão de séries temporais e modelagem financeira.
- **Controle e Robótica**: Como controle de sistemas dinâmicos e navegação autônoma.

As redes neurais multicamadas representam um avanço significativo na capacidade dos modelos de IA de aprender e generalizar a partir de dados complexos. Sua estrutura em camadas e o uso de algoritmos de treinamento eficazes como a retropropagação permitem que essas redes abordem uma vasta gama de problemas práticos de forma eficaz, destacando-se como um componente crucial na evolução da IA [16].

2.3 Pensamento Computacional

O PCO é uma habilidade cognitiva que envolve a capacidade de resolver problemas de forma lógica e algorítmica, muitas vezes fazendo uso de conceitos e técnicas da ciência da computação [22].

Embora o PCO possa ser aplicado à programação e à computação, ele se refere principalmente a uma mentalidade voltada para a resolução de problemas. Como destaca Jeanette Wing, uma das principais defensoras e teóricas do PCO, esse pensamento envolve habilidades de resolução de problemas, raciocínio lógico e a capacidade de entender e aplicar conceitos computacionais em diversas situações, não se limitando ao contexto da programação. Wing afirma que: ”Pensamento computacional é uma habilidade fundamental para todos, não somente para cientistas da computação. À leitura, escrita e aritmética, deveríamos incluir pensamento computacional na habilidade analítica de todas as crianças” [22, p. 2].

Assim, o PCO não está limitado apenas ao campo da programação, mas pode ser aplicado em uma ampla variedade de contextos, incluindo matemática, ciência, engenharia, resolução de problemas do cotidiano e muito mais.

Segundo Batista [23], utilizar o PCO é uma maneira de abordar desafios complexos de forma estruturada, dividindo-os em etapas menores e desenvolvendo algoritmos para resolvê-los:

> Trata-se de uma forma de pensar a resolução de problemas de maneira sistemática e eficiente. Essa abordagem faz uso de habilidades como a decomposição de problemas complexos em partes menores e mais gerenciáveis, o reconhecimento de padrões e a busca de similaridades a partir de experiências anteriores, a abstração de conceitos para simplificação de ideias e estratégias de solução e o desenvolvimento de algoritmos e processos com o passo a passo das soluções elaboradas. Em outras palavras, o PCO busca entender e criar soluções de maneira organizada, lógica e estruturada [23, p. 7].

O termo pensamento computacional foi popularizado na década de 2000, embora suas raízes remontem a ideias e conceitos anteriores. O PCO ganhou destaque em 2006 pela professora Jeannette Marie Wing. A popularização deste termo se deu com a publicação do seu artigo intitulado ”Computational Thinking”. Wing argumentou que o PCO refere-se a ”uma forma para seres humanos resolverem problemas; não é tentar fazer com que seres humanos pensem como computadores” [22, p. 4].

Estudos como o de Costa et al. (2021) exemplificam a aplicação de conceitos computacionais em áreas como matemática, biologia, artes e língua portuguesa, demonstrando como algoritmos e cálculos computacionais podem ser úteis, mesmo sem a utilização de um computador [24].

Dentre algumas ideias que são frequentemente discutidas no contexto do PCO, podemos destacar:

- a habilidade de dividir um problema complexo em partes menores e mais gerenciáveis;
- a capacidade de identificar tendências, regularidades e padrões dentro de um problema e que é essencial para a solução de problemas e para a criação de algoritmos eficientes;
- a habilidade de reduzir a complexidade de um sistema, concentrando-se nos aspectos mais importantes e ignorando detalhes menos relevantes, o que permite criar modelos e generalizações que podem ser aplicadas em diferentes situações;
- a compreensão de algoritmos, que são passos sequenciais ou instruções precisas para resolver um problema ou realizar uma tarefa e que envolve desenvolver habilidades para criar, analisar e melhorar algoritmos;

- a aplicação dos princípios do PCO para resolver uma variedade de problemas, desde questões matemáticas até questões do mundo real;
- a compreensão de como a automação funciona, incluindo a capacidade de criar scripts simples para automatizar tarefas repetitivas;
- a capacidade de criar modelos simulados de sistemas complexos para entender seu comportamento e fazer inferências sobre seu funcionamento futuro;
- a capacidade de avaliar soluções propostas criticamente e entender as limitações das abordagens computacionais em diversos contextos.

Brackmann aborda os Quatro Pilares do PCO: Decomposição, Reconhecimento de Padrões, Abstração e Algoritmo. O autor reforça que todos os Quatro Pilares têm grande importância e são independentes durante o processo de formulação de soluções computacionais [25].

Para Brackmann et al.:

> O Pensamento Computacional tem como pressuposto identificar problemas complexos e dividi-los em partes menores e mais simples, fáceis de gerenciar (Pilar Decomposição). Ao se trabalhar com problemas menores o aluno poderá realizar uma analisar individualmente e com maior profundidade, de forma a identificar problemas parecidos já solucionados anteriormente (Pilar Reconhecimento de Padrões), focando nos detalhes importantes e ignorando informações irrelevantes (Pilar Abstração). Por fim, orientações ou regras simples podem ser criados para solucionar cada um dos subproblemas encontrados (Pilar Algoritmos) [25, p. 33].

O PCO é uma habilidade essencial no mundo moderno, pois ajuda as pessoas a abordarem problemas de maneira mais eficaz, a entenderem a lógica por trás das tecnologias e a tomarem decisões. Como resultado, é crescente a promoção do PCO na educação.

2.3.1 A BNCC e o Pensamento Computacional

A BNCC é um documento normativo que, "define o conjunto orgânico e progressivo de aprendizagens essenciais que todos os alunos devem desenvolver ao longo das etapas e modalidades da Educação Básica" [26, p.7].

A BNCC propõe que os estudantes utilizem tecnologias, como calculadoras e planilhas eletrônicas, desde os primeiros anos do Ensino Fundamental. Essa abordagem visa garantir que, ao chegar aos anos finais, os alunos estejam preparados para desenvolver o PCO através da interpretação e elaboração de algoritmos, incluindo aqueles representados por fluxogramas.

Dentre as dez competências gerais descritas na BNCC, destaca-se aqui a competência número 5:

> Compreender, utilizar e criar tecnologias digitais de informação e comunicação de forma crítica, significativa, reflexiva e ética nas diversas práticas sociais (incluindo as escolares) para se comunicar, disseminar e acessar informações, produzir conhecimentos, resolver problemas e exercer protagonismo e autoria na vida pessoal e coletiva [26, p. 9].

Embora a BNCC não defina o ensino do PCO como uma disciplina isolada, ela integra seus princípios e habilidades em diversas áreas curriculares. No EM, as competências específicas de matemática e suas tecnologias enfatizam:

> Investigar e estabelecer conjecturas a respeito de diferentes conceitos e propriedades matemáticas, empregando estratégias e recursos, como observação de padrões, experimentações e diferentes tecnologias, identificando a necessidade, ou não, de uma demonstração cada vez mais formal na validação das referidas conjecturas [26, p. 531].

A BNCC estabelece competências gerais que incluem a resolução de problemas complexos e o uso crítico e responsável das tecnologias digitais. Ela promove a integração do PCO em disciplinas como Matemática, permitindo o desenvolvimento de habilidades para resolver problemas por meio de passos lógicos e, frequentemente, utilizando algoritmos computacionais.

Dessa forma, o documento enfatiza o desenvolvimento de habilidades cognitivas, como raciocínio lógico, abstração, solução de problemas e criatividade, que são componentes essenciais do PCO. Além disso, valoriza a colaboração, o trabalho em equipe e a resolução de conflitos, habilidades socio emocionais importantes para o desenvolvimento do PCO.

A seguir, compara-se a MM e o PCO, evidenciando como ambos se complementam na resolução de problemas. Segundo Costa et al. (2021), ambos utilizam abstração e simplificação para modelar situações complexas de forma estruturada, enquanto Wing

(2006) destaca o PCO como uma habilidade essencial que transcende a programação.

Abstração e Simplificação:

- No processo de MM, problemas do mundo real são traduzidos em equações, gráficos e fórmulas matemáticas. Essas representações ajudam a compreender padrões subjacentes e relações entre diferentes variáveis.
- O PCO envolve a decomposição de problemas complexos em partes menores, a identificação de padrões e a abstração de detalhes irrelevantes. Essas habilidades são fundamentais para a criação de algoritmos, que requerem uma compreensão da estrutura básica do problema antes da formulação.

Algoritmos e Solução de Problemas:

- Na MM, algoritmos específicos são desenvolvidos para encontrar soluções para problemas complexos, frequentemente envolvendo cálculos, otimizações ou técnicas especializadas.
- No PCO, habilidades como decomposição de problemas, reconhecimento de padrões, design de algoritmos eficientes e teste de soluções são essenciais.

Manipulação de Dados:

- Modelos matemáticos frequentemente lidam com grandes conjuntos de dados, analisando características físicas, comportamentais ou econômicas para extrair significados importantes.
- No PCO, habilidades de manipulação de dados, como organização, limpeza, análise estatística e interpretação, são usadas para extrair informações úteis dos dados.

Simulações e Experimentação Virtual:

- Modelos matemáticos são usados para simular sistemas complexos, permitindo o estudo do comportamento desses sistemas em diferentes condições.
- O PCO envolve a criação de simulações computacionais, onde algoritmos e estruturas de dados são usados para desenvolver simulações que ajudam a entender sistemas complexos.

Integração de Tecnologia:

- A tecnologia, incluindo linguagens de programação (como Python e R), Machine Learning, Inteligência Artificial e bancos de dados, facilita a criação e análise de modelos matemáticos complexos.
- No PCO, a integração de tecnologia envolve não apenas o uso de software, mas também a compreensão de como os algoritmos funcionam por trás das ferramentas de software.

Em síntese, tanto a MM quanto o PCO estão profundamente associados à resolução de problemas complexos. A MM permite compreender e modelar problemas de maneira abstrata, enquanto o PCO desenvolve o raciocínio estruturado e as habilidades práticas necessárias para implementar soluções eficientes. Ambas as abordagens fornecem uma base sólida para a inovação, além de impulsionarem o avanço científico e tecnológico, preparando as novas gerações para se tornarem os desenvolvedores das tecnologias do futuro [24] e [22].

2.4 Itinerários Formativos

O conceito de itinerário formativo surgiu como parte da implementação de novas propostas pedagógicas no Brasil, especialmente com a introdução da BNCC no EM. O itinerário formativo visa personalizar e diversificar a formação dos alunos, oferecendo trajetórias de aprendizado mais flexíveis e alinhadas aos seus interesses e habilidades [27].

A Lei nº 13.415/2017 aborda os itinerários formativos especificamente no artigo 36, que trata da reorganização do currículo do EM. De acordo com essa lei, o currículo do EM deve ser estruturado para garantir tanto a formação comum quanto a formação específica, por meio dos itinerários formativos. Esses itinerários permitem que o estudante se aprofunde em áreas do conhecimento ou em campos relacionados às suas aspirações profissionais [28].

Ao contrário do modelo tradicional, em que todos os alunos seguem um currículo único e rígido, o itinerário formativo permite que os estudantes escolham áreas de aprofundamento, com o objetivo de prepará-los melhor para o mercado de trabalho ou para a continuidade de seus estudos no nível superior [26].

Os principais objetivos do itinerário formativo são promover uma educação mais

integrada, interdisciplinar e focada no desenvolvimento das competências e habilidades dos alunos. Esse modelo de ensino busca atender às necessidades individuais de aprendizagem, estimulando a autonomia dos estudantes e oferecendo oportunidades para que se envolvam em áreas de conhecimento específicas. Além disso, pretende-se que o itinerário formativo contribua para o desenvolvimento de habilidades essenciais, como o pensamento crítico, a resolução de problemas complexos e a aplicação prática do conhecimento adquirido. Entretanto, Almeida [29], ao analisar os desafios enfrentados na implementação dos itinerários formativos no NEM brasileiro, mostra que:

> ... a submissão dos itinerários formativos aos interesses do mercado de trabalho levanta preocupações sobre a formação dos jovens brasileiros, enfatizando a preparação para um mercado precário e flexível em detrimento de uma educação sólida e abrangente. A falta de avaliação adequada das necessidades dos estudantes e a ausência de integração entre as etapas da educação básica contribuem para a fragmentação do sistema educacional e para a perpetuação das desigualdades existentes [29, p. 42].

Os itinerários formativos abrangem diversas áreas, tais como Linguagens e suas Tecnologias, Matemática e suas Tecnologias, Ciências da Natureza e suas Tecnologias, Ciências Humanas e Sociais Aplicadas, além da Formação Técnica e Profissional [30]. Dentro desses itinerários, há a possibilidade de oferecer cursos, disciplinas e atividades destinadas a aprofundar os conhecimentos em áreas específicas, proporcionando, assim, uma formação mais especializada.

No contexto da MM e RNA é possível aplicar uma proposta de sequência didática que envolva esses temas e que seja orientada para o desenvolvimento do PCO. A ideia é integrar essas áreas de forma a estimular o raciocínio lógico e a criatividade dos alunos, ao mesmo tempo em que se abordam conceitos matemáticos e computacionais essenciais. A MM, ao ser aplicada ao estudo de RNA, permite que os estudantes compreendam como algoritmos complexos podem ser formulados e solucionados de maneira estruturada. A sequência didática proposta deve incluir atividades práticas, estudos de caso e desafios de programação, proporcionando uma experiência de aprendizado que desenvolva tanto as habilidades teóricas quanto as práticas necessárias para a resolução de problemas no contexto tecnológico atual.

3 PROPOSTA E DESENVOLVIMENTO DA SEQUÊNCIA DIDÁTICA

Com base no referencial teórico apresentado, este capítulo apresenta uma abordagem didática para integrar o uso de RNA às aulas de Tecnologia e Inovação, destinadas a uma turma do EM em período integral. Tais aulas integram os itinerários formativos sugeridos para o NEM, os quais têm como objetivo principal a flexibilização e diversificação do ensino.

Esses itinerários possibilitam uma formação mais especializada. Assim, a escolha do itinerário formativo Tecnologias e Inovação justifica-se pelo interesse em aplicar a MM e desenvolver o PCO no EM, por meio da elaboração didática de uma RNA.

Para tanto, recomenda-se a implementação de um modelo de rede neural conhecido como perceptron, devido à sua simplicidade quando restrito a um número mínimo de neurônios. Essa escolha possibilita uma demonstração prática das iterações da rede, permitindo que os alunos compreendam o processo de forma manual, o que facilita a abstração da complexidade dos algoritmos subjacentes às RNA. Tal abordagem foca na compreensão dos fundamentos antes de explorar estruturas mais complexas, como as redes multicamadas.

Outra justificativa para abordar o perceptron no EM é que as ferramentas matemáticas utilizadas na MM dessa RNA, como diferentes tipos de funções de ativação, compreensão da separabilidade de dados, somatórias, sequências e operações matriciais básicas, estão alinhadas às habilidades propostas pela BNCC para essa etapa da educação básica. Além disso, o estudo do perceptron estimula o desenvolvimento do pensamento computacional ao integrar conceitos de lógica, abstração e análise de dados, promovendo a solução de problemas complexos e a aplicação prática de conhecimentos matemáticos e computacionais.

Considerando os conhecimentos básicos dos estudantes sobre o sistema nervoso, funções e conectivos lógicos, é necessária uma revisão dos conceitos fundamentais necessários para se compreender os princípios do perceptron. Essa revisão deve incluir os fundamentos matemáticos essenciais e o entendimento básico sobre o funcionamento dos neurônios, que servirão como base para abordar os aspectos tecnológicos relacionados à IA. No entanto, é importante destacar que cada turma possui características únicas, e, portanto, a necessidade de revisão pode variar. Em alguns casos, a retomada desses conceitos pode demandar mais tempo, enquanto em outros, pode ser abordada de forma mais sucinta. Assim, o planejamento da sequência didática deve ser cuidadosamente ajustado para garantir que os alunos adquiram a base necessária, permitindo uma aplicação eficaz dos conteúdos e o bom andamento das atividades subsequentes.

Foi decidida a implementação da sequência didática na turma do 3º ano da EECXC, que é a única turma em tempo integral da instituição no ano de 2024 e a única escola de nível médio do município. A turma é composta por 25 adolescentes com idades entre 16 e 18 anos, e a escolha dessa turma fundamenta-se na maturidade esperada desses estudantes, que se encontram em uma fase de crescente interesse pelo mercado de trabalho e reconhecem a importância da computação e da programação. Além disso, esses jovens demonstram entusiasmo pelo uso das tecnologias, impulsionados tanto pela curiosidade quanto pela oportunidade de explorar o tema na redação do ENEM 2024.

3.1 Plano das aulas da sequência didática

Este plano propõe um conjunto de 11 encontros ao longo de três meses, com uma aula semanal. As aulas devem ser realizadas alternando entre o ambiente da sala de aula e o laboratório de informática, priorizando atividades práticas e o uso de recursos tecnológicos, como retroprojetores, computadores e internet, para enriquecer as dinâmicas.

Na primeira aula, apresente os conceitos fundamentais relacionados à IA e avalie o conhecimento prévio dos alunos. Nas aulas seguintes, introduza a teoria sobre RNA e os princípios básicos de programação. Utilize situações-problema que possibilitem a criação e análise de redes perceptron. Promova discussões que ajudem a aprofundar a compreensão dos conceitos.

Organize os alunos em equipes e incentive-os a desenvolver soluções para os desafios propostos. Oriente cada equipe a apresentar suas abordagens, destacando as estratégias

utilizadas e os conceitos matemáticos aplicados.

O principal objetivo desta sequência didática é oferecer aos estudantes uma introdução aos conceitos fundamentais de IA e RNA, conectando esses tópicos à prática pedagógica. As atividades planejadas buscam desenvolver o PCO e habilidades matemáticas, além de estimular a resolução colaborativa de problemas.

Ao final da sequência, os alunos devem ser capazes de compreender os conceitos básicos da IA e sentir-se motivados a explorar mais profundamente essa área dinâmica e transformadora. Para alcançar esses objetivos, integre atividades práticas e promova discussões em grupo, garantindo uma aprendizagem interativa e significativa, conectada aos desafios do mundo atual.

Essa abordagem equilibra teoria e prática, com ênfase no desenvolvimento de habilidades aplicadas por meio de atividades realizadas no laboratório de informática.

3.1.1 Aula 1: Motivação do Tema

Nesta primeira aula, o objetivo é introduzir aos estudantes do EM uma sequência de atividades planejadas para explorar o campo da IA. É importante focar na base deste estudo e trabalhar com modelos simplificados para facilitar o entendimento de todos.

O propósito desta aula vai além da transmissão de conhecimentos técnicos. O professor deve buscar despertar a curiosidade, inspirar o engajamento e promover o desenvolvimento de habilidades críticas e analíticas.

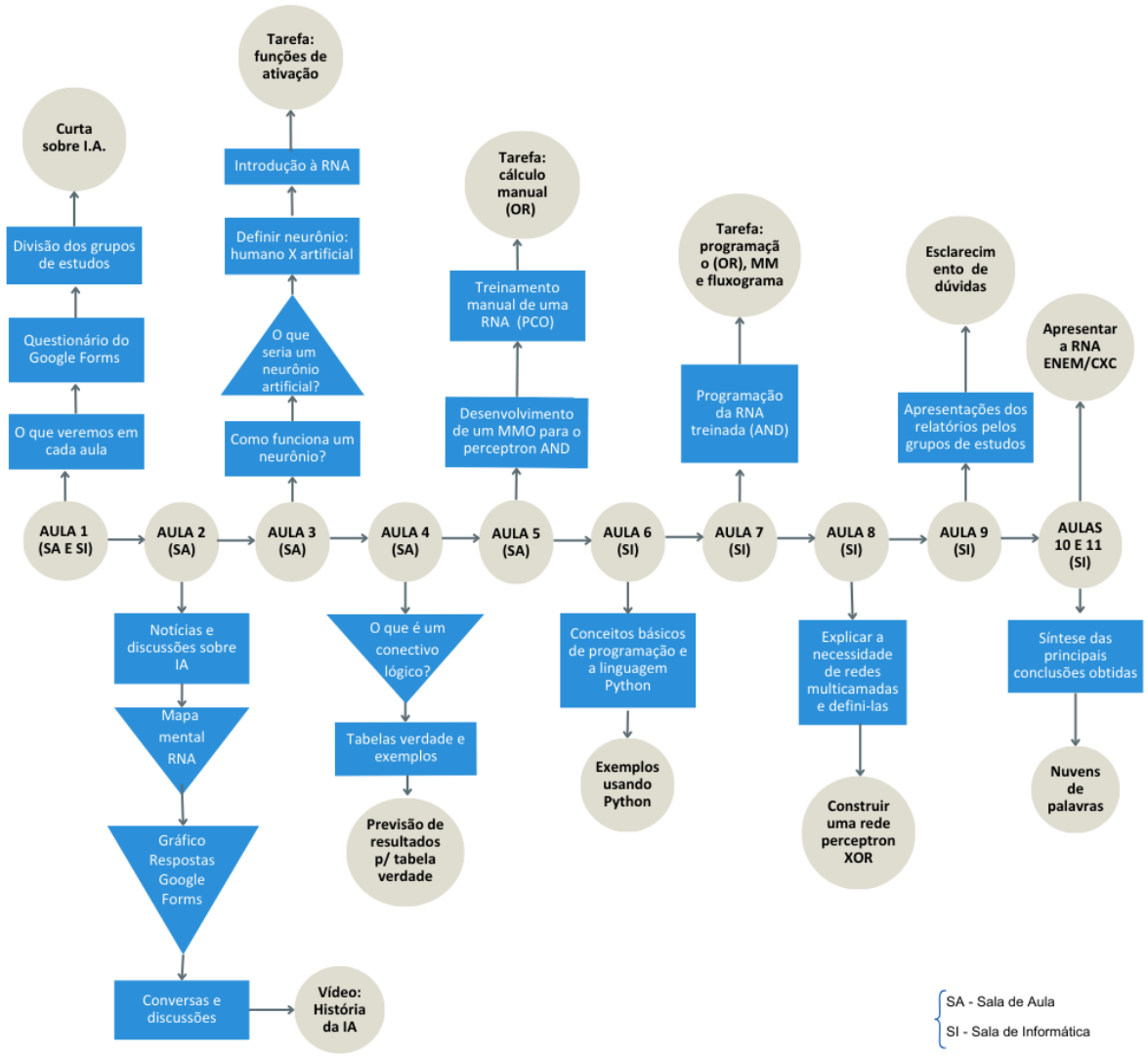
No início desta aula, cada aluno deve receber um kit com materiais de escrita para garantir a organização e a consistência dos registros ao longo de toda a sequência didática. Essa iniciativa visa não apenas facilitar o desenvolvimento eficiente do trabalho, mas também incentivar os alunos a documentarem, de forma espontânea e reflexiva, suas impressões, opiniões e aprendizados sobre cada aula.

Neste primeiro encontro, os alunos serão informados sobre o conteúdo a ser abordado nas próximas sessões da sequência didática. Na parte prática, serão desafiados a trabalhar tanto individualmente quanto em grupos de quatro pessoas. As atividades incluirão pesquisas, cálculos com funções específicas, construção de tabelas verdade, modificação de códigos já utilizados nas aulas, conforme as diretrizes estabelecidas, e execução de redes similares, além da observação de como as mudanças nas variáveis afetam o desempenho da rede. A formação dos grupos deverá ser cuidadosamente planejada, levando em

consideração as afinidades entre os estudantes e o potencial de colaboração.

Para dar esta visão abrangente do caminho que percorrerão juntos, nos próximos encontros, durante o estudo da IA, utilize o esquema delineado na Figura 3.1. Este esquema serve como um mapa para essa jornada, destacando os tópicos principais e como eles se interconectam.

Figura 3.1: Planejamento do Curso: O que Está por Vir



Fonte: Elaborada pela autora.

Ainda neste primeiro encontro, os estudantes serão conduzidos ao laboratório de informática ou orientados a utilizar seus celulares para acessar um questionário disponibilizado via Google Forms, conforme apresentado no Apêndice A deste documento. O questionário é composto por seis questões de múltipla escolha e uma questão aberta, elaboradas com o objetivo de sondar o conhecimento prévio dos alunos sobre IA. Através deste instrumento, será possível obter dados essenciais para ajustar a abordagem pedagógica de acordo com as necessidades e o nível de compreensão da turma sobre o tema.

Para a próxima aula, faça a proposta de que cada estudante leve, pelo menos, uma notícia recente sobre IA. As notícias poderão ser apresentadas em recortes de jornais ou revistas, ou em cópias de artigos encontrados na internet. Esta tarefa deve ser individual e não inclusa na nota final dos grupos de trabalho. O objetivo é incentivar a reflexão e discussão sobre o tema abordado na aula, proporcionando um entendimento mais amplo e atualizado sobre as aplicações e implicações da IA na sociedade. O professor deverá explicar, fornecer por escrito e garantir que as tarefas futuras sejam entregues impressas, para que todos compreendam e não se esqueçam das atividades.

Finalize esta aula apresentando uma curta-metragem, de 5 minutos, (<https://youtu.be/UhA_ZgI-otM?si=1HguguF0bci0dT80>) para mais uma reflexão sobre o que é a IA.

3.1.2 Aula 2: O que sabemos sobre IA?

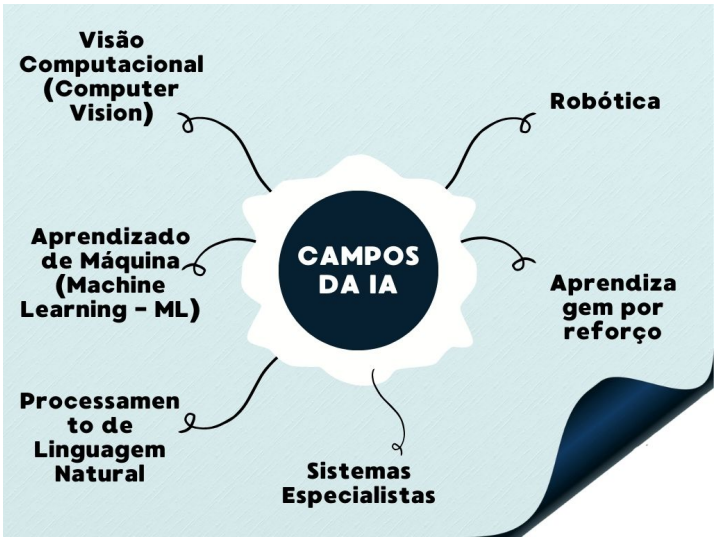
Neste encontro, explore o significado da IA e o papel das RNA na sociedade moderna. Analisando as notícias trazidas pelos estudantes, discuta sua relevância potencial e oriente-os a distinguir entre informação valiosa, sensacionalismo e preconceito. É fundamental compreender como esses conceitos inovadores estão moldando nosso presente e futuro.

Na aula anterior, foi realizada uma pesquisa através do Google Forms para explorar o conhecimento dos estudantes sobre IA e RNA. As respostas obtidas se encontram no Apêndice A, junto ao questionário, e devem ser analisadas para serem apresentadas aos alunos em forma de gráficos, com o intuito de estimular reflexões e discussões sobre o tema.

Apresente alguns dos diversos campos de estudo da IA, conforme Figura 3.2.

Mostre aos jovens que uma RNA é um destes campos de estudo e que permite que sistemas de IA aprendam e generalizem a partir de dados, aproximando o comportamento

Figura 3.2: Campos de Estudo da IA



Fonte: Elaborada pela autora.

da máquina ao do cérebro humano em termos de processamento e tomada de decisão.

Mostre a eles que o Aprendizado de Máquina (Machine Learning) foca no desenvolvimento de algoritmos que permitem aos computadores aprender a partir de dados. As RNAs são uma das técnicas mais poderosas dentro do aprendizado de máquina, especialmente eficazes para reconhecer padrões complexos em grandes volumes de dados.

Destaque também a área de visão computacional, que busca desenvolver sistemas que possam interpretar e compreender informações visuais do mundo, como imagens e vídeos. As RNAs, particularmente as redes neurais convolucionais (CNNs), são amplamente usadas aqui para tarefas como reconhecimento facial e análise de imagens médicas.

Quanto ao processamento de linguagem natural (PLN), objetiva criar sistemas capazes de entender e gerar linguagem humana. As RNAs, como as redes neurais recorrentes (RNNs) e transformadores, são fundamentais para melhorar a tradução automática, chatbots e assistentes virtuais.

Mencione o campo de estudos dos agentes autônomos e robótica, que envolve a criação de sistemas capazes de tomar decisões e agir no mundo real, muitas vezes de forma autônoma. As RNAs são usadas para controlar robôs, permitindo que eles naveguem e interajam com seu ambiente de maneira inteligente.

Explique aos alunos os sistemas especialistas, destacando que esse campo da IA foca na criação de programas que imitam a capacidade de decisão de um especialista humano em um domínio específico. Esses sistemas são projetados para resolver problemas

complexos, tomando decisões ou fornecendo conselhos em áreas como medicina, engenharia e finanças, entre outras.

Assim, na sala de aula e com o apoio de um retroprojetor, esclareça os conceitos associados à IA e às RNA por meio dos conhecimentos prévios dos estudantes e de uma análise histórica do desenvolvimento da IA.

Defina RNA como sistemas computacionais inspirados na estrutura e funcionamento do cérebro humano, especialmente no modo como os neurônios biológicos interagem e destacando que este será o assunto da próxima aula. Explique que essas redes são projetadas para simular a capacidade do cérebro de reconhecer padrões, aprender com a experiência e resolver problemas complexos.

Inicie então a discussão explorando as origens da IA, destacando marcos significativos e inovações que moldaram o campo ao longo das décadas; use a Figura 2.2. Com isso, busque conectar esses eventos históricos ao contexto atual da IA, facilitando uma compreensão mais profunda e contextualizada.

Por fim, apresente também nesta aula, um vídeo de 15 minutos sobre a história da IA: <https://youtu.be/Lhu8bdmkMCMsi=WDgwELLnLIL0h6O8>

3.1.3 Aula 3: Abordagem Geral

Na terceira aula, explore os neurônios humanos e os neurônios artificiais, comparando suas características e funções. O objetivo é proporcionar aos estudantes uma compreensão clara do que é um neurônio e de como ele pode ser reproduzido, mesmo de forma simplificada, através das RNA. Ao fazer essa comparação, é importante ressaltar a importância dos neurônios humanos na transmissão de informações e como as RNA buscam simular esse processo para resolver problemas complexos.

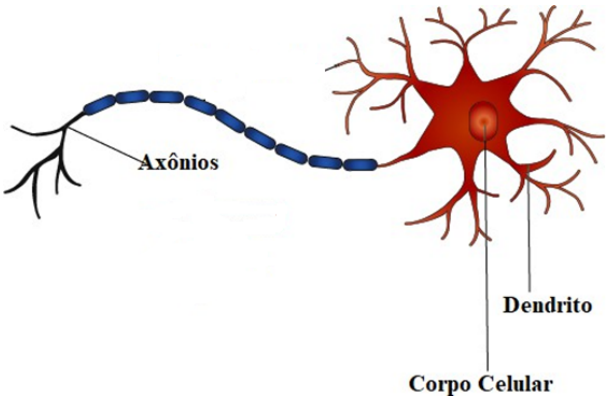
Para preparar essa aula, recomenda-se que o educador revise seus conhecimentos sobre o funcionamento dos neurônios humanos, visto que entender a biologia básica do cérebro é fundamental para compreender o funcionamento das redes neurais. Isso inclui a estrutura dos neurônios (dendritos, axônio, corpo celular), o processo de transmissão de impulsos nervosos e a sinapse, que conecta os neurônios entre si. Compreender esses conceitos básicos proporcionará aos alunos uma base sólida, que facilitará o entendimento de como as RNA tentam simular essa complexidade. Uma boa fonte de consulta pode ser livros de biologia do ensino médio ou materiais educacionais confiáveis disponíveis online,

como vídeos didáticos e artigos de instituições educacionais. Esse preparo permitirá uma abordagem mais confiante e dinâmica, além de enriquecer as discussões com os alunos, tornando o aprendizado mais interativo e interessante.

Use a MM como uma ferramenta essencial para simplificar a complexidade do cérebro, permitindo uma análise detalhada e acessível dessas estruturas fundamentais. A MM possibilita que os alunos compreendam melhor como o cérebro processa informações e como esse processo pode ser adaptado para criar sistemas computacionais, como as redes neurais artificiais.

Durante a aula, mostre aos estudantes que os neurônios humanos são células fundamentais do sistema nervoso, responsáveis por transmitir sinais elétricos e químicos por todo o corpo. De maneira simplificada, podemos dividir os neurônios em três partes principais: dendritos, corpo celular e axônios, como ilustrado na Figura 3.3.

Figura 3.3: Neurônio Humano



Fonte: [5, p. 6].

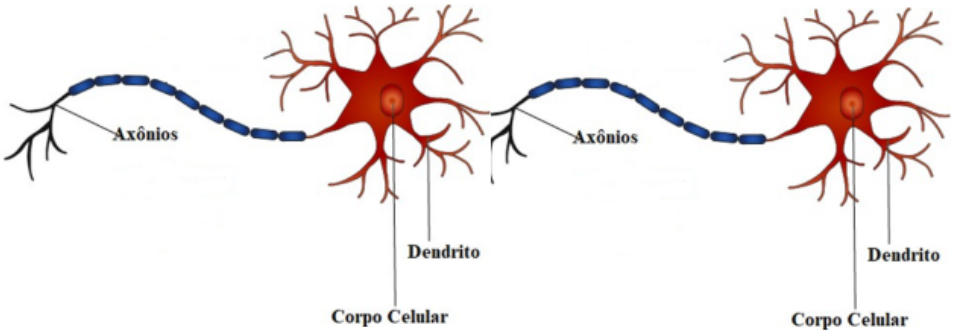
Ao apresentar essas partes, explique suas funções e como cada uma delas contribui para o funcionamento do sistema nervoso, comparando com as componentes de uma RNA.

Segundo Kandel et al. (2000), os dendritos são estruturas ramificadas curtas e finas que se estendem a partir do corpo celular do neurônio, desempenhando um papel crucial na recepção de sinais elétricos e químicos de outras células nervosas e do ambiente externo. Essas ramificações aumentam significativamente a área de superfície do neurônio, permitindo que ele receba múltiplos estímulos de diferentes fontes ao mesmo tempo. Os sinais recebidos pelos dendritos são transmitidos para o corpo celular, ou soma, que contém o núcleo e a maioria das organelas celulares. No corpo celular, esses sinais são integrados e processados, permitindo que o neurônio decida se deve ou não gerar um impulso elétrico,

conhecido como potencial de ação.

A partir do corpo celular, estende-se o axônio, uma única projeção longa e cilíndrica, que atua como a principal via de transmissão de impulsos nervosos do neurônio. Esse impulso viaja ao longo do axônio em direção às suas extremidades, onde estão localizadas pequenas estruturas especializadas chamadas terminais axônicos. Essas terminações são responsáveis por transmitir o sinal nervoso para outras células, como neurônios, células musculares ou glandulares. A comunicação entre o terminal axônico de um neurônio e os dendritos de outro ocorre através de sinapses, que são junções especializadas onde ocorre a liberação de neurotransmissores. Esses mensageiros químicos permitem a propagação do sinal, garantindo a coordenação precisa das respostas do sistema nervoso, como ilustrado na Figura 3.4 [15].

Figura 3.4: Comunicação entre dois neurônios



Fonte: Adaptado de [5, p. 6].

Quando se toca em algo quente, por exemplo, as células sensoriais na pele detectam o estímulo e enviam sinais elétricos para os dendritos dos neurônios sensoriais. Esses sinais viajam pelo corpo celular, passando ao longo do axônio até os terminais axônicos. Nos terminais axônicos, ocorre a liberação de neurotransmissores, que atravessam a sinapse e ativam o próximo neurônio na cadeia. Esta sequência de eventos resulta na sensação de calor e na ocorrência rápida de retirar a mão do objeto quente.

Para ajudar os estudantes a visualizarem essa dinâmica, faça uma simulação onde cada pessoa representa um neurônio. Imagine que seu braço direito são os dendritos, seu tronco é o corpo celular, e seu braço esquerdo é o axônio. Se você tocar uma parede quente com a mão direita, isso representa os dendritos recebendo um sinal. A informação (sinal elétrico) viaja pelo seu corpo (corpo celular) até o braço esquerdo (axônio) e a mão

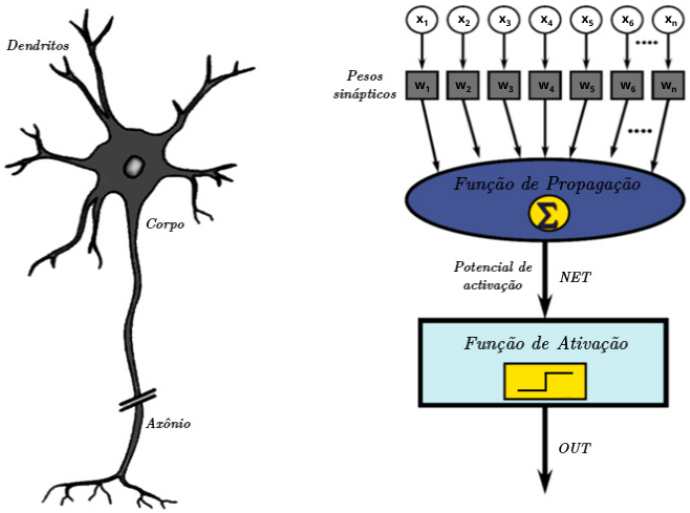
esquerda (terminais axônicos). Sua mão esquerda então se conecta à mão direita de outra pessoa, representando a sinapse. Quando a corrente se fecha e o sinal percorre essa cadeia (toda a turma com as mãos dadas, direita e esquerda) a reação, do professor, de retirar a mão direita da parede quente demonstra a resposta rápida do sistema nervoso a um estímulo.

Na comparação dos neurônios humanos com os neurônios artificiais, destaque que estes últimos são unidades básicas de processamento inspiradas no funcionamento do cérebro humano. Assim como os neurônios biológicos, eles recebem entradas (estímulos), processam essas entradas e geram uma saída (resposta).

No entanto, os neurônios artificiais são modelos matemáticos dos neurônios biológicos e não são compostos por células reais. Eles recebem entradas ponderadas, aplicam uma função de ativação e produzem uma saída. Assim como os neurônios biológicos, os neurônios artificiais também são conectados entre si em uma rede, onde a informação é processada em camadas e resulta em uma resposta ou saída.

Utilize um slide contendo a Figura 3.5 para ilustrar o funcionamento dos neurônios humanos e artificiais, comparando-os.

Figura 3.5: Neurônio Humano X Neurônio Artificial



Fonte: [18, p. 68].

Para esta exploração, reúnam-se na sala de aula tradicional, com um retroprojetor como ferramenta de ensino.

Destaque que os neurônios artificiais operam mediante o processamento de informações representadas por meio de entradas numéricas, juntamente com pesos que representam a importância relativa dessas entradas. A função de ativação determina se o

neurônio será ativado com base na soma ponderada das entradas.

Cada unidade na camada de entrada de uma RNA é chamada de neurônio, embora esses neurônios na camada de entrada não realizem operações complexas como os neurônios nas camadas internas. Lidando-se com imagens, por exemplo, cada pixel, (menor unidade de uma imagem digital exibida em uma tela) pode ser uma entrada para a rede. O número de neurônios na camada de entrada é determinado pela natureza dos dados. Se, por exemplo, a imagem tiver 28 x 28 pixels em determinado tom, teremos 28 x 28 = 784 neurônios na camada de entrada [31].

Um exemplo de neurônio artificial é o perceptron, que funciona recebendo múltiplas entradas, multiplicando-as pelos pesos correspondentes, somando esses produtos ponderados e aplicando uma função de ativação para produzir uma saída binária.

A Figura 2.6 ilustra um perceptron de n entradas, $x_1, x_2, ..., x_n$, cada uma com seus respectivos pesos: $w_1, w_2, ..., w_n$ e um viés $x_o = b = 1$.

Observar-se-á mais adiante, durante o treinamento do perceptron, que os pesos são ajustados com base no aprendizado supervisionado, onde a saída da rede é comparada com a resposta correta.

Assim, em uma RNA, que é uma espécie de cérebro artificial, tem-se vários neurônios conectados entre si. Esses neurônios são organizados em camadas e são responsáveis por processar informações e fazer previsões ou tomar decisões. Mas, para que esses neurônios funcionem corretamente, precisa-se de algo chamado função de ativação.

Existem diversos tipos de funções de ativação que podem ser utilizadas em redes neurais, cada uma com características e propriedades específicas. No entanto, é fundamental compreender que nem toda função é adequada para servir como função de ativação. Para que uma função de ativação seja eficaz, ela deve ser bem definida e suave. Isso significa que a função deve ser contínua, sem descontinuidades bruscas, quebras abruptas, ou pontos singulares que possam comprometer o fluxo de informação através da rede neural. Essas características são essenciais porque garantem que as pequenas variações nos valores de entrada resultem em pequenas e previsíveis variações nos valores de saída, o que é crucial para o processo de aprendizado da rede.

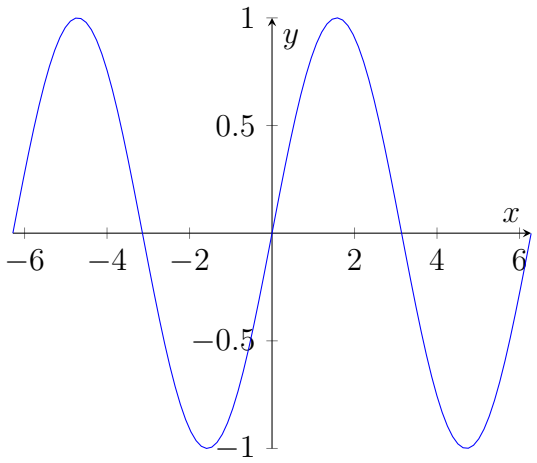
Uma função suave e contínua, por exemplo, permite que a RNA aprenda de forma mais eficiente, ajustando gradualmente os pesos durante o processo de treinamento.

Para tornar esses conceitos mais concretos e acessíveis aos alunos do ensino médio,

utilize recursos visuais com o auxílio de um retroprojetor. Por meio dele, exemplifique três tipos distintos de funções de ativação, ilustrando os casos de uma função bem definida e suave (Figura 3.6), uma função com descontinuidade brusca (Figura 3.7), e uma função que apresenta pontos singulares (Figura 3.8).

Figura 3.6: Exemplo de função definida e suave

Função Definida e Suave: $y = \text{sen}(x)$

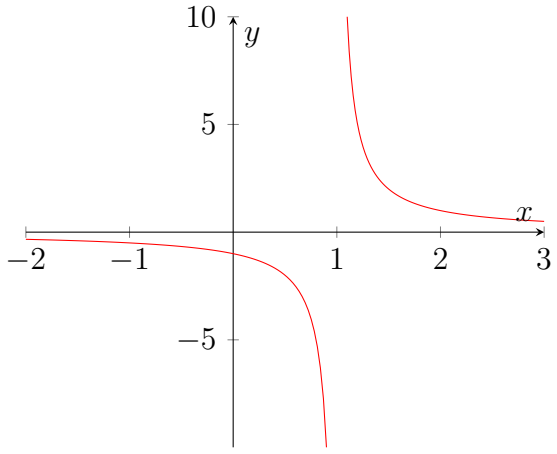


Fonte: Elaborado pela autora.

A função apresentada na Figura 3.6 é contínua em todos os pontos, não apresenta quebras bruscas ou pontos onde não se define.

Figura 3.7: Exemplo de função descontínua

Função com Descontinuidade Brusca: $y = \frac{1}{x-1}$

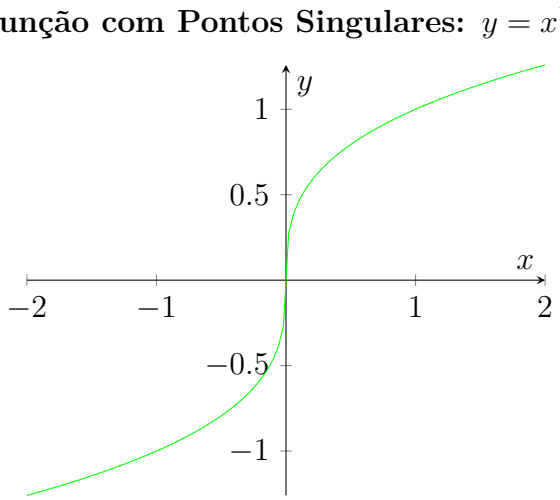


Fonte: Elaborado pela autora.

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 $x = 1$.

Figura 3.8: Exemplo de função que apresenta singularidades



Fonte: Elaborado pela autora.

No gráfico da Figura 3.8, a função é contínua, mas com uma mudança abrupta na inclinação em $x = 0$.

Portanto, 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 $y = \frac{1}{x}$, 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 3.7, 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 3.8 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 3.9 observamos 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.

A função sigmoide é definida como:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.1}$$

Observe que, quando x é muito grande, e^{-x} se aproxima de 0, pois a exponencial de um número negativo muito grande é muito próxima de zero. Assim, quando $x \to \infty$, temos $\sigma(x) \to 1$. Em outras palavras:

$$\sigma(x) \approx \frac{1}{1 + 0} = 1 \tag{3.2}$$

Quando x é muito pequeno (isto é, $x \to -\infty$), e^{-x} se torna muito grande, pois a exponencial de um número positivo muito grande cresce rapidamente. Logo, para $x \to -\infty$, temos $\sigma(x) \to 0$. Portanto:

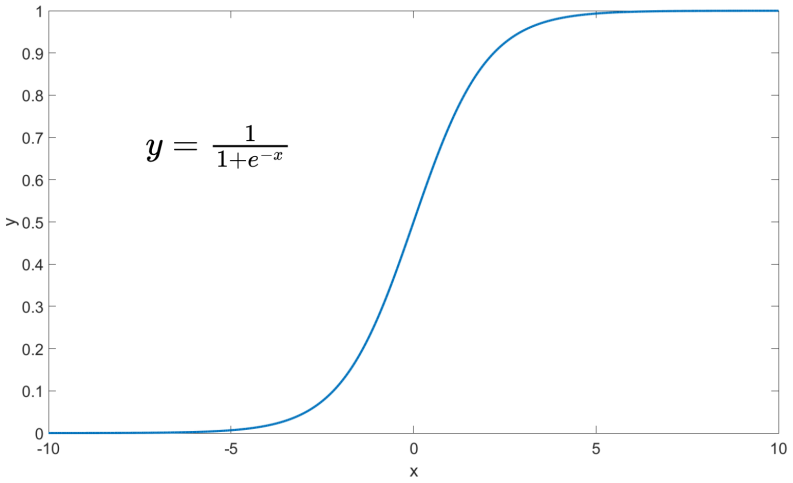
$$\sigma(x) \approx \frac{1}{1 + \infty} = 0 \tag{3.3}$$

Finalmente, quando $x = 0$, temos:

$$\sigma(0) = \frac{1}{1 + e^{0}} = \frac{1}{1 + 1} = \frac{1}{2} = 0.5 \tag{3.4}$$

Estes resultados são observados no gráfico da função sigmoide, na Figura 3.9.

Figura 3.9: Função sigmoide



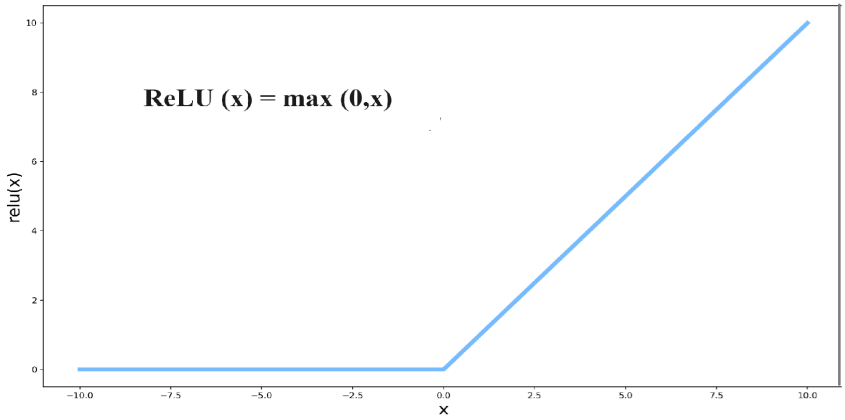
Fonte: Elaborada pela autora.

Aproveite este momento para mencionar que e é um número especial na matemática, assim como π (pi). Ressaltamos que π está relacionado aos círculos, enquanto o número de Euler, denotado por e, é uma constante matemática importante aproximadamente igual a: $e \approx 2.71828$, que aparece especialmente quando algo cresce exponencialmente, como no caso dos juros compostos ($M = P\left(1 + \frac{i}{n}\right)^{nt}$). Ambos são números irracionais, o que significa que suas representações decimais são infinitas e não periódicas.

Pode-se pensar na função sigmoid 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.

Por outro lado, 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. Isso porque, com degraus mais altos, a rede consegue fazer ajustes maiores e mais eficazes em menos tempo. É uma função simples e eficiente; muito usada em redes neurais modernas porque ajuda a resolver problemas de saturação das funções de ativação, como mostra a Figura 3.10.

Figura 3.10: Função ReLU



Fonte: Elaborada pela autora.

Enquanto a sigmoide pode ser útil em redes menores ou em casos onde é necessário uma saída entre 0 e 1, a ReLU é mais eficiente em redes profundas, facilitando o aprendizado. Assim, enquanto a sigmoid pode dificultar o aprendizado quando a rede é complexa, a ReLU ajuda a rede a aprender mais rapidamente, tornando o processo mais eficiente.

Em notação de função por partes, a função ReLU pode ser expressa como:

$$\text{ReLU}(x) = \begin{cases} x, & \text{se } x \geq 0; \\ 0, & \text{se } x < 0. \end{cases} \tag{3.5}$$

Note que, em termos mais simples, a função ReLU elimina todos os valores negativos, substituindo-os por zero, e deixa passar os valores positivos inalterados, pois está definida como:

$$\text{ReLU}(x) = \max(0, x).$$

Isso significa que, para qualquer valor de entrada:

- Se $x \geq 0$, o resultado é x.
- Se $x < 0$, o resultado é 0.

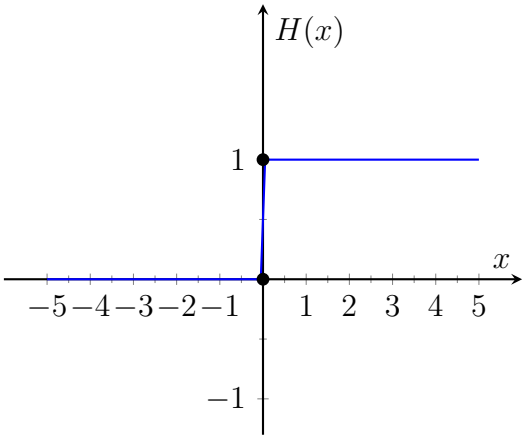
Busque, então, apresentar a função ReLU com exemplos, usando o valor zero, um valor negativo e outro positivo: Se $x = 0$, então $\text{ReLU}(0) = \max(0, 0) = 0$.
Se $x = -3$, então $\text{ReLU}(-3) = \max(0, -3) = 0$.
Se $x = 2$, então $\text{ReLU}(2) = \max(0, 2) = 2$.

Apresente também a função degrau, que é uma das funções de ativação mais simples e fundamentais utilizadas no Perceptron, uma arquitetura básica de RNA. Ela é representada pela seguinte equação:

$$H(x) = \begin{cases} 0 & \text{se } x < 0 \\ 1 & \text{se } x \geq 0 \end{cases} \tag{3.6}$$

Numa RNA, a função degrau é usada para decidir se um neurônio deve ser ativado ou não, baseado na soma ponderada de suas entradas.Ela gera uma saída binária (0 ou 1), dependendo se a soma ponderada das entradas ultrapassa um determinado limiar. Entretanto, a função degrau não é suave e possui uma descontinuidade em $x = 0$ (tem um valor constante até zero e depois salta para um, apresenta uma descontinuidade nesse ponto de salto), conforme ilustrado na Figura 3.11.

Figura 3.11: Gráfico da função degrau $H(x)$.



Fonte: Elaborada pela autora

Esta descontinuidade característica da função degrau pode dificultar o treinamento de redes neurais, especialmente em arquiteturas mais avançadas. Além disso, sua limitação a uma saída binária a torna inadequada para resolver problemas mais complexos. No entanto, essa característica não compromete o desempenho de uma RNA Perceptron, que opera em contextos de separabilidade linear.

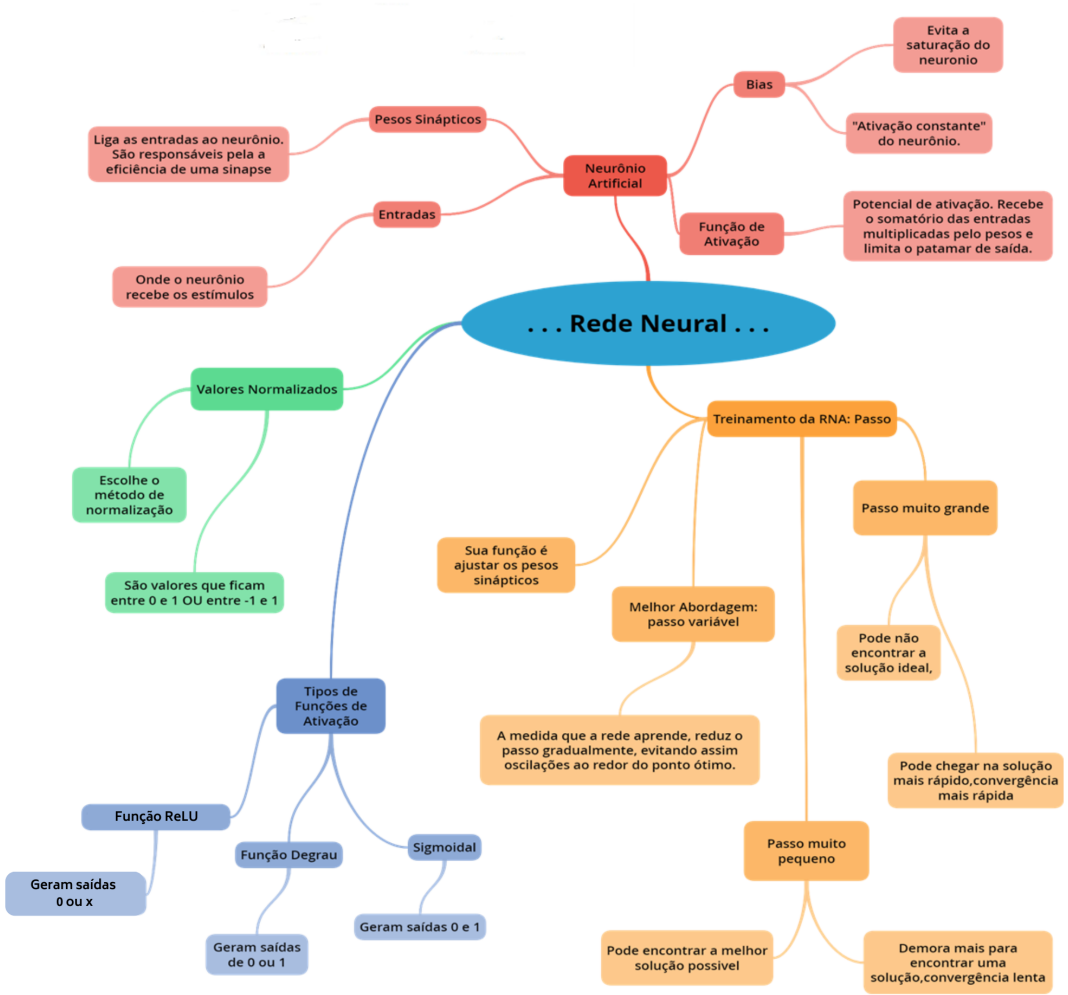
A escolha entre a função sigmoide, função degrau ou outra função de ativação depende do contexto do problema, das características dos dados e das propriedades desejadas para a rede neural em construção. Frequentemente, é necessário realizar experimentação e ajuste fino para determinar a melhor função de ativação para uma tarefa específica.

Ao longo da aula, destaque conceitos chave, como dendritos, corpo celular, axônio, função de ativação, etc., ajudando os alunos a identificarem e compreenderem os pontos mais importantes.

Reforce, portanto, que a IA é um campo vasto e em constante evolução, com aplicações que vão desde a automação industrial até o desenvolvimento de sistemas de recomendação e assistentes virtuais. Entender os princípios básicos que fundamentam a IA é essencial para qualquer estudante.

Além disso, para melhorar a interpretação e iniciar o entendimento dos conceitos ligados à IA, apresente também um mapa mental com um resumo geral sobre RNA, conforme ilustrado na Figura 3.12.

Figura 3.12: Mapa mental sobre RNA



Fonte: Adaptado de [5, p. 34].

O mapa mental apresentado na Figura 3.12 será uma ferramenta visual para

organizar e relacionar as principais ideias e componentes das RNA. Entre os elementos destacados, inclui-se as entradas, que são os dados iniciais que alimentam a RNA; os pesos sinápticos, que ajustam a influência de cada entrada na rede; o bias, um valor adicional que ajuda a ajustar a saída da rede neural permitindo que a função de ativação seja deslocada; as funções de ativação, que decide se um neurônio deve ser ativado introduzindo não-linearidade no modelo e permitindo que a rede neural aprenda a representar funções complexas; e o treinamento de uma RNA, que é o processo pelo qual a rede ajusta seus pesos e bias para minimizar o erro na saída. Este mapa mental dará aos alunos uma visão clara e estruturada do assunto, facilitando a compreensão das interconexões e funcionalidades dentro de uma RNA [5].

Conclua esta aula propondo aos estudantes uma tarefa adicional, a qual será avaliada e contará para a nota do itinerário formativo. Todas as orientações necessárias estarão disponíveis em material impresso para os grupos de trabalho. O exercício proposto, Apêndice B, visa aprofundar o entendimento das funções de ativação.

3.1.4 Aula 4: Uso dos Conectivos Lógicos

Nesta aula, explore a tabela verdade como uma ferramenta essencial para visualizar todas as possíveis combinações de valores verdade de uma expressão lógica. A lógica proposicional nos oferece uma base sólida para compreender e desenvolver raciocínios lógicos, o que é fundamental para a construção de argumentos válidos e a análise detalhada de afirmações e suas inter-relações. Além disso, a lógica proposicional é um dos pilares do pensamento computacional, sendo amplamente utilizada na programação, inteligência artificial e no desenvolvimento de sistemas baseados em regras.

Uma proposição é uma declaração que pode ser avaliada como verdadeira ou falsa, mas não pode ser ambas simultaneamente. Usando a tabela verdade, pode-se determinar de forma sistemática e clara os resultados de expressões lógicas complexas, facilitando a compreensão de como diferentes combinações de valores verdade afetam o resultado de uma proposição composta. Essa ferramenta é especialmente útil para identificar padrões de raciocínio e verificar a validade de inferências lógicas, auxiliando no desenvolvimento do pensamento crítico e analítico.

A lógica proposicional não apenas ajuda a estruturar e avaliar argumentos de forma lógica, mas também fornece uma metodologia para resolver problemas e analisar sistemas

em diversas áreas, como matemática, ciência da computação, filosofia, engenharia e até mesmo no campo jurídico. A capacidade de expressar condições e relações de forma precisa por meio de operadores lógicos possibilita a criação de modelos eficientes para tomada de decisão e automação de processos.

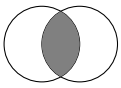
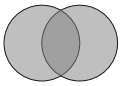
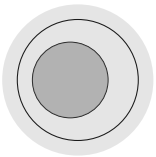
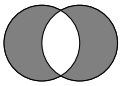
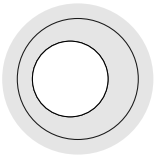
Discuta com os estudantes como a tabela verdade é fundamental para entender o comportamento de circuitos lógicos e algoritmos, facilitando a compreensão e a construção de soluções em áreas como ciência da computação e engenharia elétrica. Em circuitos lógicos, usam-se tabelas verdade para prever seu comportamento. Por exemplo, em um circuito simples com dois interruptores, a tabela verdade mostra todas as combinações possíveis de os interruptores estarem ligados ou desligados e o que acontece com a lâmpada (acende ou apaga). Esse mesmo princípio pode ser estendido para circuitos mais complexos, como somadores binários e unidades lógicas utilizadas em processadores.

Outro exemplo prático pode ser encontrado em problemas de tomada de decisão, nos quais várias condições precisam ser satisfeitas para uma ação ocorrer. A tabela verdade auxilia no mapeamento de todas as possibilidades e na identificação de quais combinações de condições levam à ação desejada. Essa abordagem é essencial em sistemas de inteligência artificial baseados em regras, onde as decisões são tomadas a partir de proposições lógicas interligadas.

Além disso, mostre exemplos práticos de como a tabela verdade pode ser aplicada na resolução de problemas e na otimização de processos lógicos. Apresente casos reais em que essa ferramenta é utilizada para simplificar expressões booleanas, reduzindo o número de operações necessárias para a execução de um determinado procedimento computacional. Esse tipo de simplificação tem impacto direto no desempenho de sistemas digitais, tornando-os mais eficientes e confiáveis.

Para facilitar a compreensão dos conectivos lógicos, apresentem a Tabela 3.1, onde são descritos o significado de cada conectivo lógico e a forma como operam sobre os valores de verdade, com o auxílio de diagramas que representam graficamente as relações entre as proposições. Esses diagramas permitem visualizar conceitos como interseção, união, inclusão, igualdade, diferença e negação, facilitando uma compreensão mais intuitiva dos conectivos lógicos. Por meio dessa abordagem visual, é possível analisar como diferentes combinações de valores de entrada geram resultados específicos, oferecendo uma compreensão clara e sistemática das operações lógicas fundamentais.

Tabela 3.1: Conectivos Lógicos: Símbolos, Significados e Representações Visuais

Conectivo	**Símbolo**	**Significado**	**Conjunto**	**Diagrama**
e (and) - conjunção	$\wedge$	Será verdadeira somente quando todas as proposições forem verdadeiras	interseção	
ou (or) - disjunção	$\vee$	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.	-	

Fonte: Elaborada pela autora.

Cite também outras situações onde as tabelas verdade podem ser usadas:

- Circuitos elétricos (projetar e analisar circuitos digitais): As entradas em um circuito digital são sinais elétricos ou lógicos fornecidos para processamento, que podem assumir dois estados básicos, frequentemente representados como 0 e 1, ou falso e verdadeiro.

- Tomadas de decisão: use a lógica para tomar decisões com base em condições. Por

exemplo, se houver previsão de chuva, leve um guarda-chuva; caso contrário, não. A tabela verdade mapeia essas situações e ações correspondentes.

- Debates e argumentações: As tabelas verdade podem ser utilizadas para representar argumentos e suas implicações lógicas. Em debates políticos, por exemplo, podem ajudar a ilustrar as consequências de diferentes políticas ou decisões.

- Programação de computadores (if, else, while): Em linguagens de programação, estruturas condicionais como `if`, `else`, e `while` são utilizadas para tomar decisões baseadas em condições, o que será abordado nas próximas aulas.

Tome então como exemplo uma sentença: "Quero café e leite". Isto é diferente de quando se diz "Quero café ou leite". No primeiro caso, só se fica satisfeito quando se recebe os dois (café e leite). No segundo caso, satisfaz-se com qualquer um dos dois ou com os dois.

A analogia com a escolha entre café e leite é especialmente útil para ilustrar a diferença entre "e"(AND) e "ou"(OR) em contextos cotidianos. Esses conceitos são fundamentais não apenas na lógica formal, mas também em muitas áreas práticas, como programação de computadores e tomada de decisões.

Neste caso, tem-se duas proposições simples:

P: Quero café.

Q Quero leite.

Pode-se então construir a Tabela 3.2 para este primeiro caso.

Tabela 3.2: Tabela AND ($\wedge$)

P	Q	P $\wedge$ Q
V	V	V
V	F	F
F	V	F
F	F	F

Fonte: Elaborada pela autora.

Na Tabela 3.2, P $\wedge$ Q é verdadeiro (V) apenas quando ambas as proposições P e Q são verdadeiras (V). Nos outros casos, onde pelo menos uma das proposições é falsa, a expressão P $\wedge$ Q também é falsa (F).

Portanto, no primeiro exemplo dado:

- Se recebo café (P é verdadeiro) e se recebo leite (Q é verdadeiro), então é verdadeira a expressão "P $\wedge$ Q".
- Se recebo café (P é verdadeiro) e não recebo leite (Q é falso), então é falsa a expressão P $\wedge$ Q.
- Se não recebo café (P é falso) e recebo leite (Q é verdadeiro), é falsa a expressão "P $\wedge$ Q".
- Se não recebo café (P é falso) e não recebo leite (Q é falso), então é falsa a expressão P $\wedge$ Q.

Mas, para o segundo exemplo, tem-se a Tabela 3.3:

Tabela 3.3: Tabela OR ($\vee$)

P	Q	P $\vee$ Q
V	V	V
V	F	V
F	V	V
F	F	F

Fonte: Elaborada pela autora.

Neste caso:

- Se recebo café (P é verdadeiro) e se recebo leite (Q é verdadeiro), é verdadeira a expressão P $\vee$ Q.
- Se recebo café (P é verdadeiro) e não recebo leite (Q é falso), é verdadeira a expressão P $\vee$ Q.
- Se não recebo café (P é falso), mas recebo leite (Q é verdadeiro), é verdadeira a expressão P$\vee$ Q.
- Se não recebo café (P é falso) e não recebo leite (Q é falso), então é falsa a expressão P $\vee$ Q.

Demonstre a relevância das tabelas verdade em situações da vida real e informe aos estudantes que haverá uma aula exclusivamente dedicada a compreender o básico da programação.

Considere então outro exemplo: "Se sou estudante e participo das aulas de matemática, então não obtenho nota maior ou igual à média". Isto pode ser, matematicamente, representado por:

$$(P \wedge Q) \rightarrow \neg R$$

onde:

- P: "Sou estudante."
- Q: "Participo das aulas."
- R: "Obtenho nota maior ou igual à média."

Neste caso, identifique primeiro a expressão $(P \wedge Q)$ a qual é referida como hipótese (sou estudante e participo das aulas de matemática) e $\neg R$ é a nossa tese (não obtenho nota maior ou igual à média). Como são três proposições nesta sentença, os valores de verdade ou falsidade são oito.

Apresente então as Tabelas 3.4 e 3.5 para explicar os conectivos lógicos se...então e negação, respectivamente, onde:

- $\rightarrow$ é o símbolo da implicação que, como visto, significa que a veracidade da proposição seguinte depende da veracidade da proposição anterior e
- $\neg R$ é a negação de R.

Tabela 3.4: Tabela Se... Então ($\rightarrow$)

$P \wedge Q$	$\neg R$	$(P \wedge Q) \rightarrow \neg R$
V	F	F
V	V	V
F	F	V
F	V	V
F	F	V
F	V	V
F	F	V
F	V	V

Fonte: Elaborada pela autora.

Tabela 3.5: Tabela NÃO (¬)

R	¬ R
V	F
F	V

Fonte: Elaborada pela autora.

Então, se R é Obtenho nota maior ou igual à média, $\neg R$ é: Não obtenho nota maior ou igual à média. Negar uma proposição significa trocar o seu valor lógico, ou seja, a negação de uma proposição verdadeira é uma proposição falsa; a negação de uma proposição falsa é uma proposição verdadeira.

È necessário entender também a implicação $(A \rightarrow B)$ que é considerada falsa apenas se (A) for verdadeira e (B) for falsa (se sou estudante e não participo das aulas de matemática, então não obter nota igual ou maior que a média é falso). Então, se $(P \wedge Q)$ é verdadeiro e $\neg R$ é falso, a sentença é falsa mas, em todos os outros casos, a sentença é verdadeira;

O objetivo aqui é determinar a verdade da proposição composta $(P \wedge Q) \rightarrow \neg R$.

Liste então, como na Tabela 3.6, todas as combinações possíveis de verdade e falsidade para P, Q, e R. Em seguida, calcule as colunas intermediárias e a coluna final para $(P \wedge Q) \rightarrow \neg R$.

Tabela 3.6: Tabela Verdade: Exemplo 2

P	Q	R	$P \wedge Q$	$\neg R$	$(P \wedge Q) \rightarrow \neg R$
V	V	V	V	F	F
V	V	F	V	V	V
V	F	V	F	F	V
V	F	F	F	V	V
F	V	V	F	F	V
F	V	F	F	V	V
F	F	V	F	F	V
F	F	F	F	V	V

Fonte: Elaborada pela autora.

Observações:

- Como há três variáveis, pelo princípio multiplicativo (2^3), a Tabela 3.6 contém oito linhas para representar todos as combinações possíveis para essa situação.
- Coluna $P \wedge Q$: É verdadeira apenas quando ambos P e Q são verdadeiros.
- Coluna $\neg R$: É o valor oposto de R.
- Coluna $(P \wedge Q) \rightarrow \neg R$: A implicação é falsa apenas quando $P \wedge Q$ é verdadeiro e $\neg R$ é falso (ou seja, R é verdadeiro).

A Tabela 3.6 permite que os estudantes vejam claramente como os valores das proposições simples P, Q e R afetam a proposição composta $(P \wedge Q) \rightarrow \neg R$. Ao trabalhar com a Tabela 3.6, os estudantes poderão praticar a aplicação de conectivos lógicos e a interpretação das tabelas-verdade, além de entender como os valores de verdade se propagam através das proposições compostas.

Assim, as tabelas verdade ajudam a entender como os operadores lógicos funcionam ao analisar as diferentes combinações de valores de verdade das proposições envolvidas. Fornecem uma maneira sistemática de analisar e entender a lógica por trás de diferentes sistemas e processos e ajudam a tomar decisões informadas, projetar sistemas eficientes e avaliar a validade de argumentos e raciocínios. Mostre assim que estas tabelas podem ter qualquer quantidade de variáveis e serão analisadas com a mesma lógica mostrada para duas variáveis. Por exemplo, a tabela AND, independente da quantidade de variáveis, só será verdadeira, se todas estas variáveis forem verdadeiras.

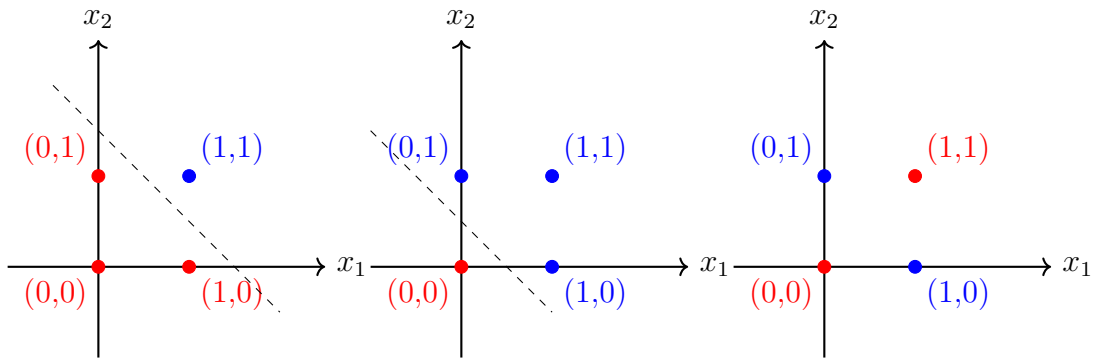
O conector AND pode servir como um exemplo introdutório para treinar uma pequena RNA e realizar operações lógicas básicas [5].

Essa abordagem permite associar a ideia de RNA à previsão de resultados com base em uma tabela verdade, demonstrando como uma RNA pode executar a operação lógica E (AND) a partir de seus dados de entrada. No perceptron, os valores de entrada são codificados como V e F, 1 e 0, respectivamente. A rede é treinada para ajustar os pesos das conexões entre os neurônios de entrada e o neurônio de saída. O objetivo é que a saída da rede seja verdadeira (1) apenas quando todos os neurônios de entrada forem verdadeiros (1), replicando o comportamento da operação lógica AND. Assim, a saída será 1 somente se todas as entradas forem 1; caso contrário, a saída será 0. É isto que será mostrado na aula seguinte.

Lembre aos estudantes que, como os dados são linearmente separáveis, é possível a construção da RNA perceptron, o modelo mais simples de neurônio.

Apresente a Figura 3.13, enfatize 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 3.13: Separabilidade dos Conectivos AND, OR e XOR



Fonte: Elaborada pela autora.

No entanto, lembre-os que, em aplicações do mundo real, RNA são geralmente usadas para tarefas mais complexas, como reconhecimento de padrões em imagens, processamento de linguagem natural, previsão, entre outros.

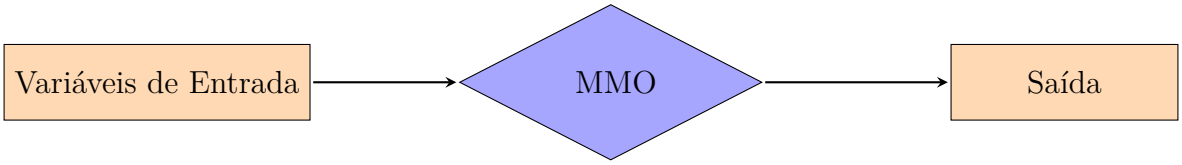
3.1.5 Aula 5: Definição e Treinamento Manual de uma RNA

Nesta aula, o objetivo é entender como construir um algoritmo de uma RNA perceptron para simular o conector lógico AND. Para isso, é preciso aplicar conceitos matemáticos e lógicos simples, sem a necessidade de computadores, utilizando instrumentos disponíveis em sala de aula: retroprojetor, quadro, cadernos e instrumentos de escrita.

Dedique-se então à elaboração manual dos algoritmos de uma rede perceptron: crie um modelo matemático (MMO) para o conectivo lógico AND utilizando o perceptron e, para isso, 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 e Q (valores de entrada) e encontrar como resposta os valores de saída $(P \wedge Q)$, tal como mostra a Tabela 3.3

O esquema do MMO pode seguir a estrutura apresentada na Figura 3.14.

Figura 3.14: Diagrama de fluxo de um processo.



Fonte: Elaborada pela autora.

Para elaborar o MMO, deve-se observar novamente o perceptron, através da Figura 2.6, 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:

$$(P, Q) \to P \wedge Q$$

$$(V, V) \to V, \quad (V, F) \to F, \quad (F, V) \to F, \quad (F, F) \to F$$

Aplicando a codificação definida, obtém-se os seguintes valores para as variáveis:

$$(1, 1) \to 1, \quad (1, 0) \to 0, \quad (0, 1) \to 0, \quad (0, 0) \to 0$$

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:

$$y = x_1 w_1 + x_2 w_2 + b \tag{3.7}$$

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, apresentada na Figura 3.11 e que pode ser definida da seguinte forma:

$$\hat{y} = f(y) = \begin{cases} 1, & y \geq 0 \\ 0, & y < 0 \end{cases} \tag{3.8}$$

ou, considerando uma soma ponderada das entradas com um viés:

$$\hat{y} = \begin{cases} 1, & x_1 w_1 + x_2 w_2 + b \geq 0 \\ 0, & x_1 w_1 + x_2 w_2 + b < 0 \end{cases} \tag{3.9}$$

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 MMO, 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 -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:

$$y = x_1 w_1 + x_2 w_2 + b = 0(0) + 0(0) - 1 = -1 \tag{3.10}$$

Agora aplique a função de ativação f (função degrau).

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

Sabe-se que para a entrada (0, 0) é esperado o valor de saída 0 (pois $(F, F) \rightarrow F$, 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: $x_1 = 1$, $x_2 = 1$, $w_1 = 0$, $w_2 = 0$ e $b = -1$.

Forme a soma:

$$y = x_1 w_1 + x_2 w_2 + b = 1(0) + 1(0) - 1 = -1 \tag{3.11}$$

Agora aplique a função degrau $\hat{y} = f(-1) = 0 \rightarrow F$. Porém esperava-se que a saída fosse 1, pois $(V), \rightarrow V$. Comprovando que, para essa entrada o modelo proposto não funciona. O que deve-se fazer?

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

Em 1959, Bernard Widrow e Marcian Hoff criaram os algoritmos ADALINE e MADALINE. ADALINE previa o próximo bit em padrões binários, enquanto o MADALINE foi a primeira RNA aplicada a um problema real, eliminando ecos em linhas telefônicas. Nesse mesmo ano, Widrow e Hoff desenvolveram a regra Delta, também conhecida como a regra de aprendizado de Widrow-Hoff, aplicada no ADALINE [32].

A Regra Delta consiste em ajustar os pesos das conexões entre neurônios para minimizar a diferença entre a saída esperada e a saída real da rede, ou seja, o erro. Enquanto o erro $E \neq 0$, atualiza-se os pesos e do bias da seguinte forma:

$$\begin{cases} w_{ji} & = w_{(j-1)i} + x_i \alpha E \\ b_j & = b_{(j-1)} + \alpha E \end{cases} \tag{3.12}$$

Passo 1: Escolha 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 aprendizagem η.

Relembre aqui que η 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. Portanto, 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, $\hat{y} = f(y)$

Passo 4: Identifique o valor de saída $\bar{y}$ correspondente à entrada (x_1, x_2) e calcular o erro:

$$E = \bar{y} - \hat{y}. \tag{3.13}$$

Passo 5: Faça a atualização de pesos e bias:

$$\begin{cases} w_{1,1} = w_{1,0} + x_1 \eta E \\ w_{2,1} = w_{2,0} + x_2 \eta E \\ b_1 = b_0 + \eta E \end{cases} \tag{3.14}$$

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:

$$y = x_1 w_{1,1} + x_2 w_{2,1} + b_1 \quad \text{(modelo 1 - da soma)} \tag{3.15}$$

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 MMO final:

$$\bar{y} = f(x_1 w_{1,n+1} + x_2 w_{2,n+1} + b_{n+1}) \tag{3.16}$$

sendo $y = x_1 w_{1,n+1} + x_2 w_{2,n+1} + b_{n+1}$ o modelo final da soma, em que:

$$\begin{cases} w_{1,n+1} = w_{1,n} + x_1 \eta E \\ w_{2,n+1} = w_{2,n} + x_2 \eta E \\ b_{n+1} = b_n + \eta E \end{cases} \tag{3.17}$$

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.

Aplique então o algoritmo matemático para resolver o problema proposto.

Passo 1: Escolha, inicialmente, a entrada $(x_1,x_2) = (0,0)$, propor os pesos iniciais $w_{1,0} = w_{2,0} = 0$, bias inicial $b_0 = -1$ e fixar a taxa de aprendizagem $\eta = 0{,}4$.

Passo 2: Monte o modelo 0 da soma:

$$y = x_1 w_{1,0} + x_2 w_{2,0} + b_0 = 0(0) + 0(0) - 1 = -1 \tag{3.18}$$

Passo 3: Defina a função degrau como função de ativação e aplicar a $y = -1$:

$$\hat{y} = f(-1) = 0, \quad \text{pois} \quad y = -1 < 0. \tag{3.19}$$

Passo 4: Identifique a saída $\bar{y} = 0$, correspondente à entrada $(x_1, x_2) = (0, 0)$ e calcular o erro: $E = \bar{y} - \hat{y} = 0 - 0 = 0$.

Atualize pesos e bias:

$$\begin{cases} w_{1,1} & = w_{1,0} + x_1 \eta E = 0 + 0(0.4)(0) = 0 \\ w_{2,1} & = w_{2,0} + x_2 \eta E = 0 + 1(0.4)(0) = 0 \\ b_1 & = b_0 + \eta E = -1 + 0.4(0) = -1 \end{cases} \tag{3.20}$$

Ou seja, o erro é nulo, então pesos e bias mantém seus valores iniciais.

Passo 5: Proponha o novo modelo da soma:

$$y = x_1 w_{1,1} + x_2 w_{2,1} + b_1 = x_1(0) + x_2(0) - 1 \quad \text{(modelo 1 - da soma)} \tag{3.21}$$

Passo 6: Escolha outra entrada $(x_1, x_2) = (1,0)$, diferente da escolhida no Passo 1 e realizar a atualização da soma:

$$y = x_1 w_{1,1} + x_2 w_{2,1} + b_1 = 1(0) + 0(0) - 1 = -1 \tag{3.22}$$

Passo 7: Aplique a função degrau a $y = -1$:

$$\hat{y} = f(-1) = 0, \quad \text{pois} \quad y = -1 < 0. \tag{3.23}$$

Passo 8: Identifique a saída $\bar{y} = 0$, correspondente à entrada $(x_1, x_2) = (1, 0)$ e

calcular o erro: $E = \bar{y} - \hat{y} = 0 - 0 = 0$. A seguir, atualize pesos e bias:

$$\begin{cases} w_{1,2} & = w_{1,1} + x_1 \eta E = 0 + 1(0.4)(0) = 0 \\ w_{2,2} & = w_{2,1} + x_2 \eta E = 0 + 0(0.4)(0) = 0 \\ b_2 & = b_1 + \eta E = -1 + 0.4(0) = -1 \end{cases} \tag{3.24}$$

Passo 9: Proponha o novo modelo da soma:

$$y = x_1 w_{1,2} + x_2 w_{2,2} + b_2 = x_1(0) + x_2(0) - 1 \quad \text{(modelo 2 - da soma)} \tag{3.25}$$

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

$$y = x_1 w_{1,2} + x_2 w_{2,2} + b_2 = 0(0) + 1(0) - 1 = -1 \tag{3.26}$$

Passo 11: Aplique a função degrau a $y = -1$:

$$\hat{y} = f(-1) = 0, \quad \text{pois} \quad y = -1 < 0. \tag{3.27}$$

Passo 12: 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:

$$\begin{cases} w_{1,3} & = w_{1,2} + x_1 \eta E = 0 + 0(0.4)(0) = 0 \\ w_{2,3} & = w_{2,2} + x_2 \eta E = 0 + 1(0.4)(0) = 0 \\ b_3 & = b_2 + \eta E = -1 + 0.4(0) = -1 \end{cases} \tag{3.28}$$

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

Passo 13: Proponha o novo modelo da soma:

$$y = x_1 w_{1,3} + x_2 w_{2,3} + b_3 = x_1(0) + x_2(0) - 1 \quad \text{(modelo 3 - da soma)} \tag{3.29}$$

que, nesse caso, coincide com o modelo 2.)

Passo 14: Escolha outra entrada $(x_1, x_2) = (1,1)$, diferente das escolhidas nos

Passos 1, 6 e 10, e realize a atualização da soma:

$$y = x_1 w_{1,3} + x_2 w_{2,3} + b_3 = 1(0) + 1(0) - 1 = -1 \tag{3.30}$$

Passo 15: Aplique a função degrau a $y = -1$:

$$\hat{y} = f(-1) = 0, \quad \text{pois} \quad y = -1 < 0. \tag{3.31}$$

Passo 16: Identifique a saída $\bar{y} = 1$, 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:

$$\begin{cases} w_{1,4} & = w_{1,3} + x_1 \eta E = 0 + 1(0.4)(1) = 0.4 \\ w_{2,4} & = w_{2,3} + x_2 \eta E = 0 + 1(0.4)(1) = 0.4 \\ b_4 & = b_3 + \eta E = -1 + 0.4(1) = -0.6 \end{cases} \tag{3.32}$$

Passo 17: Proponha o novo modelo da soma:

$$y = x_1 w_{1,4} + x_2 w_{2,4} + b_4 = x_1(0.4) + x_2(0.4) - 0.6 \quad \text{(modelo 4 - da soma)} \tag{3.33}$$

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:

$$y = x_1 w_{1,4} + x_2 w_{2,4} + b_4 = 0(0.4) + 0(0.4) - 0.6 = -0.6 \tag{3.34}$$

Passo 19: Aplique a função degrau a $y = -0.6$:

$$\hat{y} = f(-0.6) = 0, \quad \text{pois} \quad y = -0.4 < 0. \tag{3.35}$$

Passo 20: Identifique a saída $\bar{y} = 0$, correspondente à entrada $(x_1, x_2) = (0, 0)$ e calcule o erro: $E = \bar{y} - \hat{y} = 0 - 0 = 0$.

Atualize pesos e bias:

$$\begin{cases} w_{1,5} & = w_{1,4} + x_1\eta E = 0.4 + 0(0.4)(0) = 0.4 \\ w_{2,5} & = w_{2,4} + x_2\eta E = 0.4 + 0(0.4)(0) = 0.4 \\ b_5 & = b_4 + \eta E = -0.6 + 0.4(0) = -0.6 \end{cases} \tag{3.36}$$

Passo 21: Proponha o novo modelo da soma:

$$y = x_1 w_{1,5} + x_2 w_{2,5} + b_5 = x_1(0.4) + x_2(0.4) - 0.6 \quad \text{(modelo 5 - da soma)} \tag{3.37}$$

Passo 22: Escolha outra entrada, $(x_1, x_2) = (0,1)$ e realizar a atualização da soma:

$$y = x_1 w_{1,5} + x_2 w_{2,5} + b_5 = 0(0.4) + 1(0.4) - 0.6 = -0.2 \tag{3.38}$$

Passo 23: Aplique a função degrau a $y = -0.2$:

$$\hat{y} = f(-0.2) = 0, \quad \text{pois} \quad y = -0.2 < 0. \tag{3.39}$$

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:

$$\begin{cases} w_{1,6} & = w_{1,5} + x_1\eta E = 0.4 + 0(0.4)(0) = 0.4 \\ w_{2,6} & = w_{2,5} + x_2\eta E = 0.4 + 1(0.4)(0) = 0.4 \\ b_6 & = b_5 + \eta E = -0.6 + 0.4(0) = -0.6 \end{cases} \tag{3.40}$$

Passo 25: Proponha o novo modelo da soma:

$$y = x_1 w_{1,6} + x_2 w_{2,6} + b_6 = x_1(0.4) + x_2(0.4) - 0.6 \quad \text{(modelo 6 - da soma)} \tag{3.41}$$

Passo 26: Escolha outra entrada, $(x_1, x_2) = (1,1)$ e realize a atualização da soma:

$$y = x_1 w_{1,7} + x_2 w_{2,7} + b_7 = 1(0.4) + 1(0.4) - 0.6 = 0.2 \tag{3.42}$$

Passo 27: Aplique a função degrau a $y = 0.2$:

$$\hat{y} = f(0.2) = 1, \quad \text{pois} \quad y = 0.2 > 0. \tag{3.43}$$

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:

$$\begin{cases} w_{1,7} & = w_{1,6} + x_1 \eta E = 0.4 + 0(0.4)(0) = 0.4 \\ w_{2,7} & = w_{2,6} + x_2 \eta E = 0.4 + 1(0.4)(0) = 0.4 \\ b_7 & = b_6 + \eta E = -0.6 + 0.4(0) = -0.6 \end{cases} \tag{3.44}$$

Passo 29: Proponha o novo modelo da soma:

$$y = x_1 w_{1,7} + x_2 w_{2,7} + b_7 = x_1(0.4) + x_2(0.4) - 0.6 \quad \text{(modelo 6 - da soma)} \tag{3.45}$$

Passo 30: Escolha outra entrada, $(x_1, x_2) = (1,1)$ e realize a atualização da soma:

$$y = x_1 w_{1,7} + x_2 w_{2,7} + b_7 = 1(0.4) + 1(0.4) - 0.6 = 0.2 \tag{3.46}$$

Passo 31: Aplique a função degrau a $y = 0.2$:

$$\hat{y} = f(0.2) = 1, \quad \text{pois} \quad y = 0.2 > 0. \tag{3.47}$$

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:

$$\bar{y} = f(x_1 w_{1,7} + x_2 w_{2,7} + b_7) = f(x_1(0.4) + x_2(0.4) - 0.6) \tag{3.48}$$

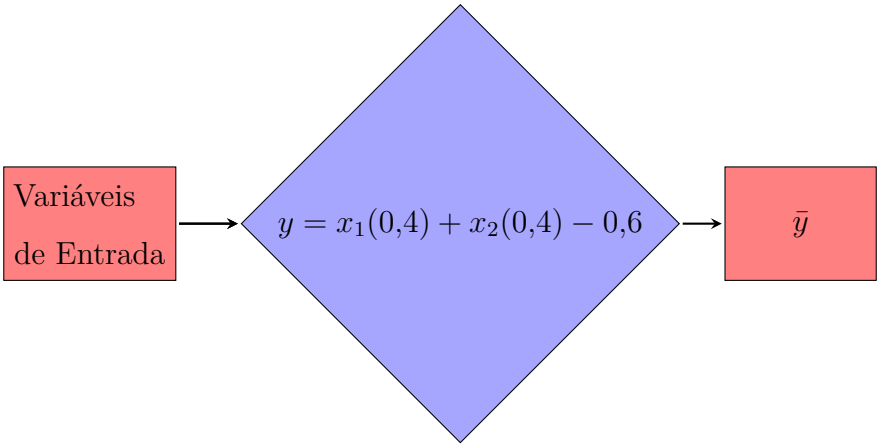
Encontrando o modelo final da soma:

$$y = x_1(0.4) + x_2(0.4) - 0.6 \tag{3.49}$$

Observe que o algoritmo matemático possibilita:

- obter um modelo matemático para resolver o problema proposto, conforme a Figura 3.15.

Figura 3.15: Diagrama de fluxo MMO.



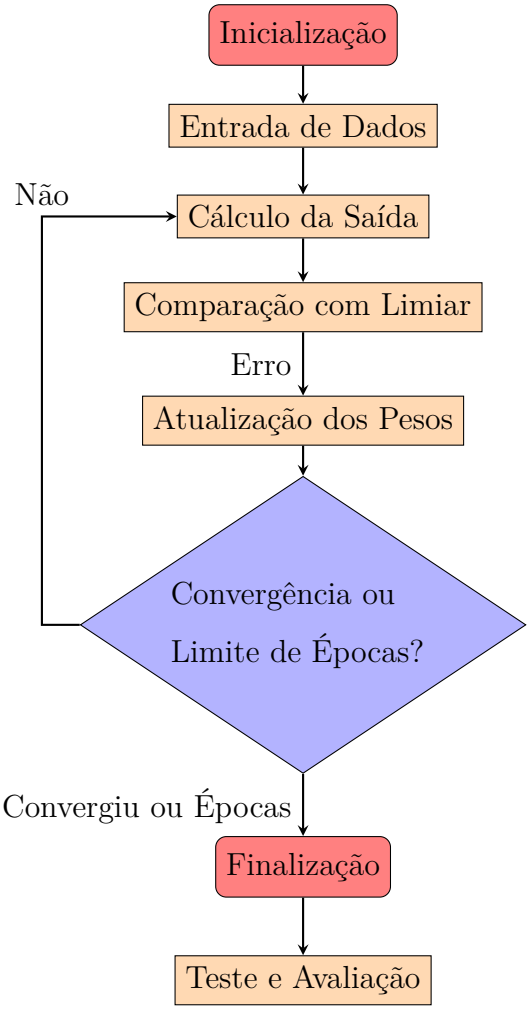
Fonte: Elaborada pela autora.

- Perceber a realização de uma concatenação de passos (fluxo) até atingir uma solução satisfatória. Isso significa que cada etapa do processo deve ser executada em uma sequência lógica, onde os resultados de uma etapa servem como base para a próxima. Assim, torna-se possível compreender o encadeamento das operações realizadas e sua importância na construção da solução.

- Detectar uma padronização de procedimentos (repetição), fundamental para a organização do pensamento computacional. 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 um fluxograma, conforme apresentado na Figura 3.16, para visualizar de forma clara e estruturada como ocorre a concatenação de passos no computador. O fluxograma servirá como um guia visual para entender a sequência lógica das operações, destacando os pontos-chave do processo e auxiliando na organização do raciocínio algorítmico.

Figura 3.16: Fluxograma de uma Rede Peceptron



Fonte: Elaborado pela autora.

Em resumo, 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 (η) e o número de épocas. Defina aleatoriamente os pesos das variáveis de entrada e o bias para a construção da rede perceptron AND: w_i, b e η, conforme mostrado abaixo:

$$w_1 = 0, \quad w_2 = 0, \quad b = -1, \quad \eta = 0{,}4$$

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 $\bar{y}$ 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.

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 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.

Aproveite também para discutir o símbolo de somatório ($\sum$), de acordo com a BNCC para o EM, que visa fortalecer o raciocínio lógico-matemático com a aplicação de notações matemáticas em problemas relacionados a sequências e séries [26].

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 $\eta = 0{,}4$. Esse 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.

A seguir apresente a Tabela 3.7 de uma rede AND que começou com b = 1 e pesos $w_1 = -0{,}1$ e $w_2 = -0{,}2$. Precisou de 7 iterações para encontrar a estabilidade:

Tabela 3.7: Ajuste de pesos e valores: RNA Perceptron AND

Época	x_1	x_2	b	y	w_1	w_2	w_0	Soma	$\hat{y}$	Erro	Ajuste de pesos
1	1	1	1	1	-0.1	-0.2	0.0	-0.3	0	1	Sim
	1	0	1	0	0.0	-0.2	0.1	0.1	1	-1	Sim
	0	1	1	0	0.0	-0.1	0.1	0.0	0	0	Não
	0	0	1	0	0.0	-0.1	0.1	0.1	0	0	Não
2	1	1	1	1	0.0	-0.1	0.1	0.0	0	1	Sim
	1	0	1	0	0.1	-0.1	0.2	0.2	1	-1	Sim
	0	1	1	0	0.1	0.0	0.2	0.2	0	0	Não
	0	0	1	0	0.1	0.0	0.2	0.2	0	0	Não
3	1	1	1	1	0.1	0.0	0.2	0.3	1	0	Sim
	1	0	1	0	0.2	0.0	0.3	0.5	1	-1	Sim
	0	1	1	0	0.2	0.1	0.3	0.4	0	0	Não
	0	0	1	0	0.2	0.1	0.3	0.3	0	0	Não
4	1	1	1	1	0.2	0.1	0.3	0.6	1	0	Sim
	1	0	1	0	0.3	0.1	0.4	0.8	1	-1	Sim
	0	1	1	0	0.3	0.2	0.4	0.7	0	0	Não
	0	0	1	0	0.3	0.2	0.4	0.6	0	0	Não
5	1	1	1	1	0.3	0.2	0.4	0.9	1	0	Sim
	1	0	1	0	0.4	0.2	0.5	1.1	1	-1	Sim
	0	1	1	0	0.4	0.3	0.5	1.0	0	0	Não
	0	0	1	0	0.4	0.3	0.5	0.9	0	0	Não
6	1	1	1	1	0.4	0.3	0.5	1.2	1	0	Sim
	1	0	1	0	0.5	0.3	0.6	1.4	1	-1	Sim
	0	1	1	0	0.5	0.4	0.6	1.3	0	0	Não
	0	0	1	0	0.5	0.4	0.6	1.2	0	0	Não
7	1	1	1	1	0.5	0.4	0.6	1.5	1	0	Sim
	1	0	1	0	0.6	0.4	0.7	1.7	0	0	Sim
	0	1	1	0	0.6	0.5	0.7	1.6	0	0	Sim
	0	0	1	0	0.6	0.5	0.7	1.5	0	0	Sim

Fonte: Elaborada pela autora.

Na Tabela 3.7, observe os valores de y (desejado) e $\hat{y}$ (obtido) em cada iteração. Quando esses valores são iguais, significa que a rede produziu a saída correta, de modo que os pesos e o bias são mantidos, e a rede passa para processar uma nova entrada. No entanto, se os valores de y e $\hat{y}$ forem diferentes, isso indica que há um erro na previsão da rede. Nesse caso, é necessário realizar um ajuste nos pesos e no bias.

O ajuste é feito de acordo com a Regra Delta, em que o erro é utilizado para corrigir os pesos de forma que a rede melhore suas previsões nas próximas iterações. Esse ciclo de ajustes ocorre continuamente até que, para todas as entradas, a rede consiga gerar as saídas corretas, ou seja, quando $y = \hat{y}$ para todas as entradas do conjunto de

treinamento.

Quando todas as entradas resultam na saída desejada e o erro se torna nulo, diz-se que a rede atingiu a convergência. Neste ponto, não há mais necessidade de reajustar os pesos e o bias, uma vez que a rede foi treinada para reconhecer corretamente os padrões apresentados.

Este processo de aprendizado se desenvolve ao longo de várias épocas (passagens por todas as entradas de treinamento). A convergência é alcançada quando, ao final de uma época, todas as previsões feitas pela rede estão corretas. A partir desse momento, a rede estará apta para realizar previsões futuras com os parâmetros ajustados adequadamente.

A ideia é que, à medida que a rede neural é treinada com mais exemplos da tabela verdade, ela ajusta os pesos de suas conexões de forma a otimizar suas previsões, aproximando-se cada vez mais dos resultados esperados.

Portanto, associar a ideia de RNA com a previsão de resultados para uma tabela verdade significa usar a tabela verdade como um conjunto de dados de treinamento para a rede neural, permitindo que ela aprenda a prever corretamente a saída com base nas entradas fornecidas.

Novamente, finalize a aula apresentando os próximos exercícios que compõem a atividade avaliativa desta sequência didática, detalhados no Apêndice C.

3.1.6 Aula 6: Conceitos Básicos de Programação

Nesta aula, o objetivo é entender os fundamentos da programação em Python para, posteriormente aplicá-los ao desenvolvimento de uma rede perceptron para a função lógica AND. Use a plataforma Google Colab para facilitar a execução de códigos que exemplificam cada caso.

O Google Colaboratory, ou Colab, é uma plataforma gratuita de notebooks Jupyter hospedada na nuvem, que permite escrever e executar código Python diretamente no navegador. O acesso à plataforma é possível para qualquer pessoa que tenha uma conta no Google.

Para configurar o ambiente no Google Colab, cada estudante deve logar a sua conta e realizar uma pesquisa na web, sendo direcionado para a página correspondente.

Informe aos estudantes que o grande poder da linguagem Phyton está nas suas bibliotecas, algoritmos que alguém já criou e que permitem, muitas vezes, com uma única

linha de código, executar códigos muito complexos. Caso precise de alguma biblioteca específica que não esteja instalada por padrão, pode-se instalá-la diretamente em uma célula do notebook Colab usando comandos como `!pip install`.

Por exemplo:

```
# Instalar o pacote NumPy
!pip install numpy
```

Para implementar uma rede perceptron, é fundamental compreender alguns conceitos básicos de programação e sua aplicação em Python. Elabore slides que deverão conter tais conceitos:

1. **Comentários:** são linhas de texto que são ignoradas pelo interpretador Python. São usados para documentar o código, explicar seu funcionamento e torná-lo mais legível. Começam com o caractere `#` e serão bastante usados nos exemplos destes conceitos básicos.

2. **Indentação:** é o espaçamento no início de uma linha de código que define um bloco de código. Em Python, a indentação é obrigatória e é usada para determinar a estrutura e a hierarquia do código.

   Estruturas como if, for, while e def, que serão explicadas mais adiante, requerem uma indentação adequada para indicar quais linhas de código fazem parte dessas estruturas e devem ser executadas sob determinadas condições. A convenção é utilizar 4 espaços para cada nível de indentação, mas o importante é ser consistente ao longo de todo o código, seja utilizando espaços ou tabulações.

   A indentação é fundamental na legibilidade e estruturação do código. Ela facilita a identificação visual dos blocos de código, permitindo que se compreenda facilmente a estrutura do programa. A falta de indentação adequada pode resultar em erros de sintaxe, já que Python depende da indentação para determinar o fluxo de execução do código.

   Exemplo de Indentação:

   ```
   x=6
   if x > 5:
   ```

```
    print("x é maior que 5")
    print("Isso está dentro do bloco do if")
print("Isso está fora do bloco do if")


y=5
   if y > 5:
        print("x é maior que 5")
        print("Isso está dentro do bloco do if")
print("Isso está fora do bloco do if")
```

Neste exemplo, as duas instruções print após o if estão indentadas, indicando que fazem parte do bloco de código que será executado se a condição x > 5 for verdadeira. A última instrução print não está indentada e, portanto, será executada independentemente da condição, pois está fora do bloco do if.

Use os exemplos acima para mostrar diferentes situações: na primeira parte, defina corretamente a identação do valor de x; na segunda parte, o Colab vai notificar um erro de identação e os estudantes precisarão ajustar o código do valor de y.

3. **Strings:** Strings são sequências de caracteres utilizadas para representar texto em linguagens de programação. Em Python, strings são delimitadas por aspas (simples ou duplas) e podem incluir letras, números, símbolos, espaços e outros caracteres. Tudo o que é escrito entre aspas em aspas uma string é exibida exatamente como foi digitada quando usada em um comando print.

4. **Print (mostrar resultado):** É uma função usada para exibir mensagens, resultados ou informações na tela. Existem diferentes tipos de prints; são úteis para entender como cada função de ativação se comporta e interpretar os resultados do modelo. O print mostra se a condição foi satisfeita conforme a função degrau, exibe o valor exato da saída, no caso da função ReLU ou mostra o valor com precisão decimal, representando uma probabilidade, no caso da função sigmoide.

```
   # Exemplos de outputs diferentes
step_output = step_function(soma_ponderada)
```

```
relu_output = relu_function(soma_ponderada)
sigmoid_output = sigmoid_function(soma_ponderada)

print(f"Saída da função degrau: {step_output}")
print(f"Saída da função ReLU: {relu_output}")
print(f"Saída da função sigmoide: {sigmoid_output:.4f}")

# 4f pede resultado com 4 casas decimais.
```

5. **Estruturas de Controle:** Permitem controlar o fluxo de execução do código. As principais estruturas incluem:

   - **Condicionais:** ‘if‘, ‘elif‘, e ‘else‘ são usados para executar blocos de código com base em condições específicas.
   - **Laços:** ‘for‘ e ‘while‘ são usados para repetir a execução de blocos de código várias vezes.

6. if (se): Executa um bloco de código se uma condição for verdadeira. Por exemplo:

```
idade = 16
if idade >= 18:
    print('Você é maior de idade.')
else:
    print('Você é menor de idade.')
```

   Se idade for maior ou igual a 18, a mensagem ’Você é maior de idade.’ será impressa. Caso contrário, será impressa a mensagem ’Você é menor de idade.’ Em sala de aula, verifique este comando usando diferentes idades.

7. else (então): Executa um bloco de código se a condição do if for falsa, como no exemplo anterior. Também pode ser usado com elif para testar condições adicionais. Por exemplo:

```
x = 10
if x > 5:
    print("x é maior que 5")
elif x == 5:
    print("x é igual a 5")
else:
    print("x é menor que 5")
```

Neste exemplo, como x é maior que 5, o primeiro bloco de código é executado e as condições seguintes são ignoradas. Se x fosse igual a 5, o segundo bloco seria executado. Se x fosse menor que 5, o terceiro bloco seria executado.

8. for: Usado para repetir um bloco de código várias vezes. Imagine que você tem uma tarefa que precisa ser repetida, como imprimir uma lista de números ou processar uma série de dados. Em vez de escrever o mesmo código várias vezes, você pode usar um loop for para automatizar esse processo. Por exemplo, para imprimir os números de 1 a 5, você usa:

```
for i in range(1, 6):
    print(i)
```

Neste caso, range(1, 6) gera a sequência [1, 2, 3, 4, 5]. O for ajuda a tornar o código mais limpo, eficiente e fácil de entender.

9. while (enquanto): Executa um bloco de código enquanto uma condição específica for verdadeira. Por exemplo:

```
contador = 0
while contador < 5:
print("Contagem:", contador)
contador += 1
```

Este loop while continua executando o bloco de código, de um em um, até que contador seja igual a cinco. Para melhor entendimento do código, pedi aos estudantes que trocassem 1, por 2, depois por 3, neste comando:

```
contador += 1
```

Assim, é possível verificar como o contador funcionaria caso fosse necessário executar o bloco de código de outra maneira.

10. **Operações matemáticas:** são cálculos realizados sobre números para obter resultados específicos, incluindo adição, subtração, multiplicação, divisão, comparações, entre outras. Elas são usadas para manipular dados, como somar elementos de uma lista, calcular médias, ou comparar valores.

11. **Variáveis:** são espaços de armazenamento na memória que contêm valores. Em Python, você pode simplesmente atribuir um valor a uma variável usando o operador de atribuição =.

```
# Exemplo de tipos de dados e variáveis
idade = 25
altura = 1.75
nome = "João"
e_maior_de_idade = True

print(nome, "tem", idade, "anos e", altura,
"metros de altura.")
if e_maior_de_idade:
    print(nome, "é maior de idade.")
else:
    print(nome, "não é maior de idade.")
```

12. **Operadores:** Python suporta vários tipos de operadores, como operadores aritméticos (+, -, *, /), operadores de comparação (==, !=, >, <, >=, <=), operadores lógicos (and, or, not), entre outros. Esses operadores são usados para realizar operações matemáticas, comparações e lógica booleana.

13. **Pacotes (ou bibliotecas):** são coleções de módulos que oferecem funcionalidades específicas e ajudam a simplificar o desenvolvimento. Utilize:

    - `Matplotlib`: Biblioteca para criação de gráficos e visualizações.

- `NumPy`: Biblioteca para manipulação de arrays, frequentemente abreviada como `np`.

14. **Manipulação de Arrays com NumPy:** Arrays são listas de números, palavras ou outros tipos de dados que se queira manipular. Permitem armazenar vários valores em uma única variável. Utilize `NumPy` para manipular esses arrays. Veja como inicializar arrays com zeros e com valores aleatórios:

```
# Inicializar arrays (pesos do perceptron) com zeros:

import numpy as np
num_features = 2
weights = np.zeros(num_features)
print("Pesos iniciais:", weights)

# Inicializar os pesos com valores aleatórios:

weights = np.random.uniform(-1, 1, num_features)
print("Pesos iniciais aleatórios:", weights)
```

Peça que executem o código mais de duas vezes e observem os valores aleatórios se alterando.

15. **inputs:** se refere às entradas que são fornecidas a um modelo ou algoritmo. No contexto de redes neurais e perceptrons, os inputs são os dados que o modelo processa para fazer previsões ou classificações. Eles fornecem os dados que o modelo usa para aprender e fazer previsões.

16. **len(inputs):** Verifica o número de entradas que um modelo está recebendo.

Ao analisar alguns princípios básicos de programação, mostre aos alunos como esses conceitos fundamentais (arrays, laços, funções e operações matemáticas) são usados para resolver problemas reais.

17. **Produto Escalar:** O comando np.dot calcula o produto escalar, que é usado para calcular a soma ponderada das entradas:

```
# Entradas e pesos do perceptron
inputs = np.array([0, 1])
weights = np.array([0, 0])
bias = -1

# Calcular a soma ponderada
weighted_sum = np.dot(inputs, weights) + bias
print("Soma ponderada:", weighted_sum)
```

18. **Função:** é um bloco de código que é executado quando é chamado. Pode aceitar argumentos (dados) e retornar um resultado. Elas ajudam a organizar o código em partes menores e reutilizáveis. As funções em Python são definidas usando a palavra-chave def, como nos exemplos a seguir:

```
[breaklines=true]
def calculo_area_retangulo(largura, altura):
    return largura * altura
# Chamando a função
area = calculo_area_retangulo(5, 10)
print(area)  # Saída: 50
```

19. **Funções de Ativação:** determinam a saída do perceptron com base na soma ponderada das entradas. Explore três funções de ativação:

20. 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}')
```

21. 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.

22. 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}')
```

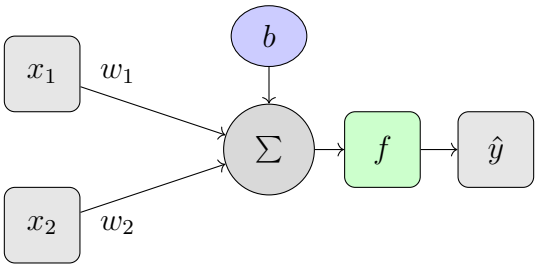
3.1.7 Aula 7: Programação da Rede Perceptron

Após apresentar o algoritmo do perceptron passo a passo e os fundamentos da linguagem de programação nas aulas anteriores, é fundamental aprofundar o entendimento prático, demonstrando como um perceptron funciona em situações concretas. A aplicação prática ajuda a consolidar os conceitos teóricos e a compreender sua utilidade no desenvolvimento de soluções baseadas em redes neurais.

No exemplo detalhado manualmente, é explorado um perceptron com duas entradas, uma configuração clássica que se aplica à função lógica AND. Esta função ilustra de forma simples e didática como os pesos, os valores de entrada, o limiar e a função de ativação interagem para produzir um resultado. Nesse contexto, apresente a Figura 3.17,

que retrata um perceptron com duas entradas e explica visualmente seu funcionamento, destacando as etapas do cálculo, desde a soma ponderada até a aplicação da função de ativação.

Figura 3.17: Diagrama do Perceptron com duas entradas



Fonte: Elaborada pela autora.

Inicie acessando o Google Colab, onde deve-se colar os primeiros comandos compartilhados anteriormente no grupo do WhatsApp da turma.

Importe a biblioteca NumPy, utilizada para criar pesos iniciais aleatórios e realizar cálculos, defina e inicialize uma classe perceptron. Implemente o cálculo do erro e o uso da função degrau, que converte a saída linear do perceptron em uma saída binária, conforme os comentários e os códigos a seguir.

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 viés, e, em seguida, atualiza-os com base nos erros calculados durante o treinamento. Todo o processo é acompanhado por prints detalhados para monitorar o progresso.

```
[breaklines=true]
# 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}")
```

Visualize então o seguinte resultado 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

É importante lembrar que reduzir o número de épocas pode economizar tempo de computação, especialmente se percebido que o erro está convergindo rapidamente para zero ou se a curva de aprendizado começar a nivelar após um certo número de épocas

mas interromper o treinamento prematuramente pode resultar em um modelo que não foi suficientemente treinado, levando a um desempenho insatisfatório e a um maior erro de previsão. Além disso, se o modelo não tiver tempo suficiente para aprender os padrões dos dados, ele pode apresentar um desempenho ruim em novos dados. Entretanto, no caso de uma rede perceptron treinando para a função lógica AND, que é um problema simples, é possível que a rede já esteja treinada adequadamente em poucas iterações. Para esse problema específico, é comum observar que, em duas iterações, a rede já atinge um desempenho satisfatório.

Os pesos foram inicializados como um vetor de zeros: $(w = np.zeros(X.shape[1]))$. Isso significa que, independentemente das entradas, a saída do perceptron no início sempre será o valor do viés, que é -1. Isso resulta em uma saída inicial sempre igual a 0, pois a função de ativação função degrau retornará $\text{função}_{\text{degrau}}$ 0 para qualquer entrada, já que a soma será sempre negativa.

No contexto de redes neurais, inicializar todos os pesos com o mesmo valor (como zero ou um valor fixo) geralmente cria simetria entre os neurônios. Isso impede que a rede aprenda adequadamente, pois todos os neurônios atualizarão seus pesos da mesma forma. Como resultado, o modelo terá dificuldades para aprender características diferenciadas, o que pode tornar o aprendizado mais lento ou até falhar. Em modelos mais complexos, como redes neurais profundas, isso é um problema conhecido.

No caso do perceptron aplicado a problemas simples, como a função AND, a simplicidade do problema e sua natureza linear podem permitir que o modelo ainda aprenda, mesmo com essa simetria inicial. Isso ocorre porque as atualizações dos pesos com base na regra de aprendizado do perceptron (como a regra delta) são eficazes o suficiente para quebrar essa simetria ao longo do tempo, possibilitando o aprendizado adequado.

A função degrau é funcional no contexto de um perceptron clássico devido à sua simplicidade e adequação a problemas lineares. No entanto, ela não é prática ou eficaz para redes neurais modernas, que requerem gradientes para treinamento e a capacidade de capturar relações não lineares. Em redes mais complexas, funções de ativação como a ReLU ou a sigmoide são preferíveis, pois oferecem melhores condições para a aprendizagem e a generalização do modelo.

O bias foi atualizado durante o treinamento do perceptron, e a atualização segue a

mesma lógica básica dos pesos, ajustando-se com base no erro entre a saída desejada e a saída prevista. Essa atualização do bias é importante para permitir que o perceptron se ajuste corretamente a diferentes funções de decisão.

Mostre também aos estudantes que a RNA pode ser treinada de diferentes formas, aplicando-se outras funções de ativação ao mesmo algoritmo. Em seguida, apresente a rede Perceptron para a função AND, utilizando também as funções sigmoide e ReLU, conforme demonstrado nos Anexos H e G, respectivamente. Observe que, mesmo ao alterar as funções de ativação, o resultado esperado foi alcançado em apenas duas iterações.

No entanto, enquanto a função degrau é excelente para introduções e problemas simples, a função sigmoide oferece um maior poder de aprendizado, apesar de exigir cálculos mais complexos, que não são práticos para serem realizados manualmente. A função ReLU, por sua vez, é útil devido à sua simplicidade e eficiência computacional. Ela ajuda a evitar problemas de saturação, que ocorrem quando a saída da função de ativação é muito próxima de seus valores máximos ou mínimos, o que pode dificultar o aprendizado da rede.

Para tornar a aula mais interessante, pode-se adicionar algumas visualizações. Por exemplo, usar a biblioteca matplotlib para plotar gráficos que mostram a separação linear realizada pela RNA, a partir dos códigos seguintes:

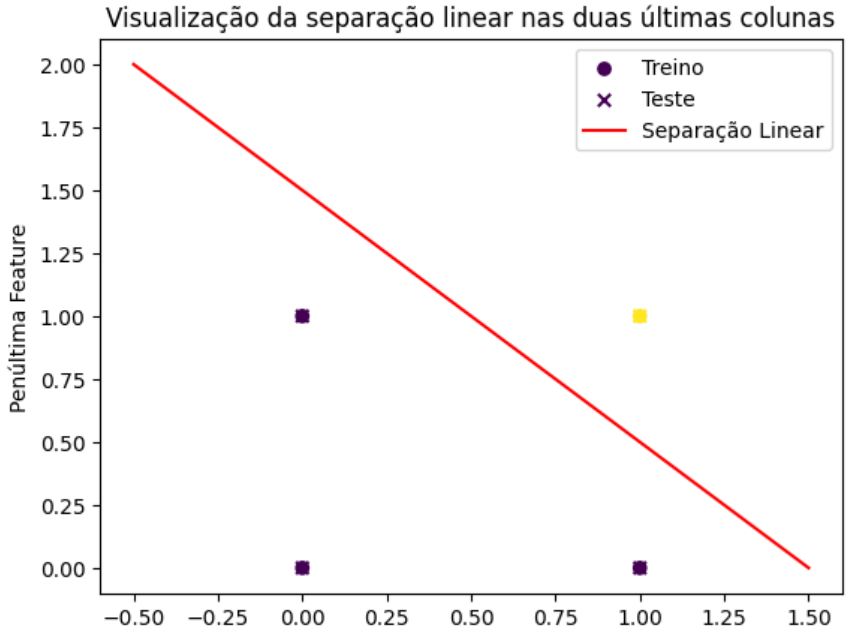
```
[breaklines=true]


import numpy as np
import matplotlib.pyplot as plt

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

# Função para treinar o perceptron
def train_perceptron(X, y, learning_rate, epochs):
    w = np.zeros(X.shape[1])  # Pesos iniciais
    b = -1  # Viés inicial
```

```
    for epoch in range(epochs):
        for i in range(X.shape[0]):
            linear_output = np.dot(X[i], w) + b
            y_pred = step_function(linear_output)
            error = y[i] - y_pred
            w += learning_rate * error * X[i]
            b += learning_rate * error

    return w, b

# Dados de exemplo (Entradas e Saídas esperadas)
X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
y = np.array([0, 0, 0, 1])  # Saídas esperadas (AND lógico)

# Definir a taxa de aprendizagem e o número de épocas
learning_rate = 0.4
epochs = 10

# Treinar o perceptron
weights, bias = train_perceptron(X, y, learning_rate, epochs)

# Dados de treinamento e teste (para visualização)
x_treino_2d = X
y_treino = y
x_teste_2d = X
y_teste = y

# Criar um gráfico de dispersão para visualizar a separação linear
plt.scatter(x_treino_2d[:, 0], x_treino_2d[:, 1],

c=y_treino, cmap='viridis', label='Treino', s=100)
```

```
plt.scatter(x_teste_2d[:, 0], x_teste_2d[:, 1], c=y_teste,

cmap='viridis', marker='x', label='Teste', s=100)

# Adicionando a linha de separação linear
x_values = np.linspace(-0.5, 1.5, 100)
y_values = (-weights[0] * x_values - bias) / weights[1]
plt.plot(x_values, y_values, color='red', label='Separação Linear')

# Configurações do gráfico
plt.title('Visualização da Separação Linear para a Função AND')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.legend()
plt.grid()
plt.show()
```

Ao executar a rede, visualize o gráfico da Figura 3.18, que representa a separação linear para a função AND, gerado no Google Colab.

Figura 3.18: Separação Linear: Função AND



Fonte: Elaborada pela autora.

Conclua esta aula explorando a programação das redes neurais, destacando como diferentes abordagens podem resolver problemas semelhantes. Para reforçar o aprendizado e consolidar os conceitos abordados, proponha uma nova tarefa que complementa a atividade da Aula 5 e culmina em um relatório final. Esta tarefa está disponível no Apêndice D.

3.1.8 Aula 8: Construção de uma Rede Neural Multicamadas

Nesta aula, busque aprofundar a compreensão sobre redes neurais, destacando a necessidade de redes multicamadas (MLP) para lidar com problemas mais complexos. Para isso, retome o exemplo da função XOR, que não possui dados linearmente separáveis, como mostrado na Figura 3.19.

É preciso lembrar aqui que a função OR (ou inclusivo) quer prever quando pelo menos uma das entradas é verdadeira (1), enquanto a função XOR (ou exclusivo) quer prever se apenas uma das entradas é verdadeira, como mostra a Tabela 3.8:

Tabela 3.8: Tabela XOR ($\oplus$) e OR ($\vee$)

P	**Q**	**P** $\vee$ **Q**	**P** $\oplus$ **Q**
V	V	V	F
V	F	V	V
F	V	V	V
F	F	F	F

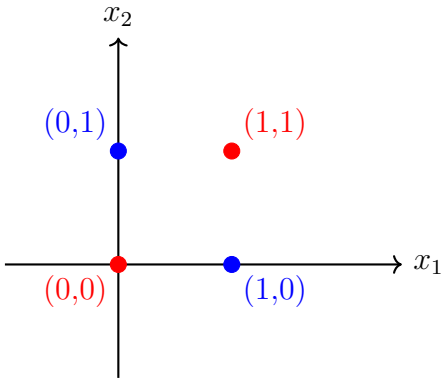
Fonte: Elaborada pela autora.

No caso da função XOR, o desafio é que um único perceptron, como implementado no código para a função AND, não consegue modelar a função XOR diretamente. Isso ocorre porque a função XOR não é linearmente separável, como ilustra o gráfico da Figura 3.19.

O conectivo lógico XOR (ou exclusivo) é verdadeiro apenas quando as duas entradas são diferentes. Em outras palavras, a função XOR retorna verdadeiro (1) somente se uma das entradas for verdadeira e a outra falsa.

Essa característica implica que não existe uma linha reta que possa separar claramente as duas classes de saída (0 e 1) na representação gráfica do XOR. Para distinguir

Figura 3.19: Inseparabilidade dos dados: função XOR.



Fonte: Elaborada pela autora.

as classes, é necessário o uso de uma fronteira de decisão não linear, pois apenas fronteiras curvas conseguem separar adequadamente os pontos das duas classes.

Observe então que, embora o perceptron tenha sido inicialmente considerado um avanço significativo no campo da IA, foi posteriormente demonstrado que o perceptron de uma única camada tem limitações significativas quando se trata de problemas não linearmente separáveis. No entanto, o perceptron ainda é uma peça fundamental na história do desenvolvimento de redes neurais e aprendizado de máquina, servindo como ponto de partida para o desenvolvimento de arquiteturas de redes neurais mais complexas e eficazes.

Para modelar com precisão a função XOR, será necessário utilizar uma rede neural mais complexa, como uma MLP. Assim, pode-se baixar a biblioteca Python tensorflow, ela automatiza muitos dos cálculos complexos necessários para treinar modelos de aprendizado de máquina.

Mostre então aos estudantes a programação da rede XOR, Apêndice I, enfatizando que esta estrutura é mais complexa do que as redes AND e OR. Explique que, devido à sua complexidade, não entraria em todos os detalhes da implementação. Em vez disso, focaria no resultado final e na plotagem do gráfico para confirmar a teoria por trás do funcionamento da rede. Assim, os alunos podem compreender a aplicação prática sem se preocupar com a complexidade da matemática envolvida.

3.1.9 Aula 9: Aplicação Prática

No final das aulas anteriores os estudantes foram desafiados a resolver algumas questões e relatar o trabalho grupal desenvolvido, com o objetivo de verificar o aprendizado

do perceptron: MMO, cálculo manual seguindo o algorítimo matemático, indução ao PCO através da elaboração de fluxogramas e do entendimento e execução do código computacional.

Nesta aula, os estudantes deverão entregar o relatório e apresentar seus trabalhos. Determine um tempo máximo para cada grupo expor suas ideias, utilizando o projetor para exibir a programação realizada. Após cada apresentação, faça um intervalo para esclarecer dúvidas ou complementar informações.

3.1.10 Aula 10: Apresentação de uma RNA com dados da EECXC

Enfatize aqui que as redes multicamadas são uma generalização do perceptron simples. Elas incluem uma ou mais camadas ocultas entre a camada de entrada e a camada de saída. Um perceptron simples não tem camadas ocultas e, por isso, tem capacidade limitada na resolução de problemas complexos. Já as redes multicamadas, com suas camadas ocultas, conseguem lidar com uma variedade muito maior de problemas, extraindo características complexas e modelando relações não lineares entre as entradas e as saídas.

Apresente então uma rede multicamadas desenvolvida para a previsão de resultados do ENEM a partir de um treinamento com dados do INEP, de estudantes da EECXC.

A necessidade de uma rede multicamadas se resume à sua capacidade de:

- Resolver problemas não linearmente separáveis.
- Extrair características complexas e em vários níveis de abstração.
- Modelar relações não lineares entre entradas e saídas.

Essas capacidades são fundamentais para enfrentar desafios reais de aprendizado de máquina, como reconhecimento de voz, visão computacional, e muitas outras aplicações que requerem a interpretação e a análise de grandes volumes de dados complexos.

Tendo em vista a limitação oferecida por uma rede perceptron, torna-se interessante desenvolver uma rede multicamadas com dados que façam parte da história dos estudantes ou que lhes despertem maior curiosidade. Contextualize então o problema abordado destacando a importância de prever o desempenho no ENEM e discutindo como isso

pode beneficiar os alunos (prever o desempenho no ENEM não apenas lhes dá uma visão mais clara de suas habilidades e áreas de melhoria, mas também os capacita a tomar decisões educacionais e profissionais mais bem fundamentadas, reduzindo a ansiedade e aumentando suas chances de sucesso acadêmico e profissional). Inclua também uma breve discussão sobre as implicações éticas de se usar dados pessoais para prever o desempenho de alguém no exame.

Utilizou-se para esta rede multicamadas dados dos alunos da EECXC que participaram do ENEM, de 2009 a 2022. Esses dados foram obtidos a partir da base do INEP, que disponibiliza informações individuais de cada aluno, ano a ano, desde 2005. Embora a base de dados do INEP esteja disponível desde 2005, optamos por focar nos dados a partir de 2009 devido a alterações significativas no formato e na aplicação do exame, o que garante uma maior comparabilidade e relevância para o contexto atual da avaliação.

Ao realizar a filtragem dos dados, foi construído um conjunto de dados (dataset) exclusivo para os estudantes desta escola, num total de 348 candidatos, compreendendo informações pessoais, resultados de provas, questionários socioeconômicos e notas do ENEM. Durante esse processo, eliminou-se os dados considerados irrelevantes para o estudo, especificamente aqueles que se repetiam em todas as linhas, como nomes e códigos da cidade, do estado e da escola.

É válido ressaltar que todo esse procedimento de organização de dados foi conduzido por meio da linguagem de programação Python, cujos códigos estão disponíveis gratuitamente na internet. Adicionalmente, é importante notar que o ChatGPT pode ser utilizado como uma ferramenta para auxiliar na revisão do código e oferecer sugestões de melhorias, mas a execução real do código é realizada na plataforma Python.

Após a filtragem dos dados, foi calculada a média das notas dos exames (Redação, Ciências Humanas, Ciências da Natureza, Linguagens, Matemática e Língua Estrangeira) e armazenada na coluna MEDIA-NOTAS. Criou-se então a coluna PROBABILIDADE, que classifica os estudantes como 1 ou 0, dependendo se sua média é igual ou superior a 500 pontos, respectivamente. Essa transformação de dados permite que o problema seja tratado como uma classificação binária, facilitando a análise e a modelagem.

Dividiu-se os dados em variáveis preditoras e variável alvo (classificação binária) e foram aplicadas técnicas de imputação de dados ausentes, removendo outliers e normalizando os dados. A divisão dos dados em 80% para treinamento e 20% para teste

foi realizada para avaliar a capacidade de generalização do modelo, permitindo que ele consiga prever corretamente os resultados em dados que não foram usados durante o treinamento.

A rede neural foi construída com camadas Dense (um tipo de camada em redes neurais onde cada neurônio está conectado a todos os neurônios da camada anterior (se houver) e da camada seguinte) e funções de ativação ReLU para as camadas ocultas, e Sigmoid para a camada de saída. A função Sigmoid é especialmente útil em problemas de classificação binária, pois ela transforma as saídas em probabilidades entre 0 e 1, o que permite classificar os estudantes como tendo ou não uma média superior a 500 pontos.

Finalmente, o modelo foi treinado, e avaliou-se a acurácia nos conjuntos de treino e teste, utilizando a função de perda binary-crossentropy (que mede a diferença entre as probabilidades previstas e os valores reais em problemas de classificação binária), a otimização com o Adam (um algoritmo eficiente para ajuste dos pesos da rede neural, combinando as vantagens do Gradient Descent com técnicas de adaptação de taxas de aprendizado) e o monitoramento do treinamento por meio de EarlyStopping (uma técnica que interrompe o treinamento quando o desempenho no conjunto de validação não melhora após um número definido de épocas, prevenindo overfitting). Isso garantiu que a melhor versão do modelo fosse salva.

Esta rede deve ser apresentada aos estudantes para que possam entender melhor como o perceptron evolui para uma rede multicamadas.

No laboratório de informática, os alunos podem ter acesso aos dados reais da escola usados para o desenvolvimento do modelo e acompanhar o desenvolvimento da rede neural multicamadas executada no Google Colab. Além disso, explique o código, sem se ater à parte complexa da matemática ali aplicada; desde a importação dos dados até a avaliação do modelo, para garantir que compreendam como os conceitos teóricos se traduzem em prática.

Mostre que, na realização deste projeto, foram percorridas as seguintes etapas:

1. **Coleta e Pré-processamento dos Dados**

   Certificou-se de ter todos os dados necessários do INEP para cada estudante, analisando a relevância das variáveis para o processo. Realizou-se a limpeza de dados, tratando valores ausentes, removendo outliers e normalizando os dados para garantir que todas as variáveis tenham um peso semelhante durante o treinamento

do modelo. A normalização ajuda a acelerar o treinamento e a evitar que certas características tenham um impacto desproporcional nos resultados do modelo. Este processo envolveu programação, utilizando a linguagem Python.

2. **Exploração dos Dados**

   Realizou-se a análise exploratória dos dados para entender melhor a distribuição e as relações entre as variáveis.

3. **Divisão dos Dados**

   Separou-se os dados em conjuntos de treinamento e teste para avaliar o desempenho do modelo. Treinou-se a rede com 80% dos dados e reservo-se 20% para teste.

4. **Construção do Modelo**

   Escolheu-se uma arquitetura de rede neural adequada para o problema, definindo a rede com duas camadas e utilizando as funções ReLU e Sigmoide como funções de ativação e critério de perda.

5. **Treinamento do Modelo**

   Treinou-se o modelo usando os dados de treinamento, ajustando os hiperparâmetros para otimizar o desempenho.

6. **Avaliação do Modelo**

   Usou-se os dados de teste, avaliou-se o modelo para garantir que ele generalizasse bem para novos dados e fez-se a análise métricas de desempenho, como precisão, recall, F1-score e a curva ROC.

7. **Código Python e Comentários**

   Para desenvolver este modelo, siga os passos seguintes, que se resumem em preparação dos dados, normalização, imputação e divisão dos conjuntos de treino e teste:

   - Importe as bibliotecas `pandas`, `numpy` e `matplotlib.pyplot`;
   - Carregue e pré-processe os dados;
   - Calcule a média das notas dos exames e adicione uma nova coluna `MEDIA-NOTAS`;
   - Crie uma coluna `PROBABILIDADE` com base na condição da média >= 500;

- Divida os dados em variáveis preditoras (`x`) e variável alvo (`y`);
- Lide com valores não numéricos: calcule o escore Z para todas as colunas numéricas e remova instâncias com outliers;
- Normalizar, imputar e dividir os conjuntos de treino e teste, essenciais para a construção e avaliação do modelo de rede neural.

Esses passos poderão ser realizados no Google Colab.

Após a apresentação da rede neural, ela poderá ser compartilhada com os alunos, que perceberão a necessidade de indicar um novo diretório para o arquivo salvo em seus respectivos computadores, a fim de executar a rede corretamente. Para que os estudantes consigam importar o arquivo salvo para o Google Colab, será necessário realizar uma pequena modificação na célula de código correspondente. Precisarão importar o arquivo salvo para o Colab e, para isso, editar na célula:

```
    from google.colab import files
uploaded = files.upload()
```

.

Quando este código for executado, uma janela de diálogo aparecerá, permitindo que os alunos naveguem até o local em seus computadores onde o arquivo está armazenado e o selecionem para upload. Após a conclusão do upload, o arquivo ficará acessível no ambiente do Colab, podendo ser utilizado diretamente pelo nome do arquivo. Por exemplo, se o arquivo se chamar redeneural.cxc, eles poderão carregá-lo com o código:

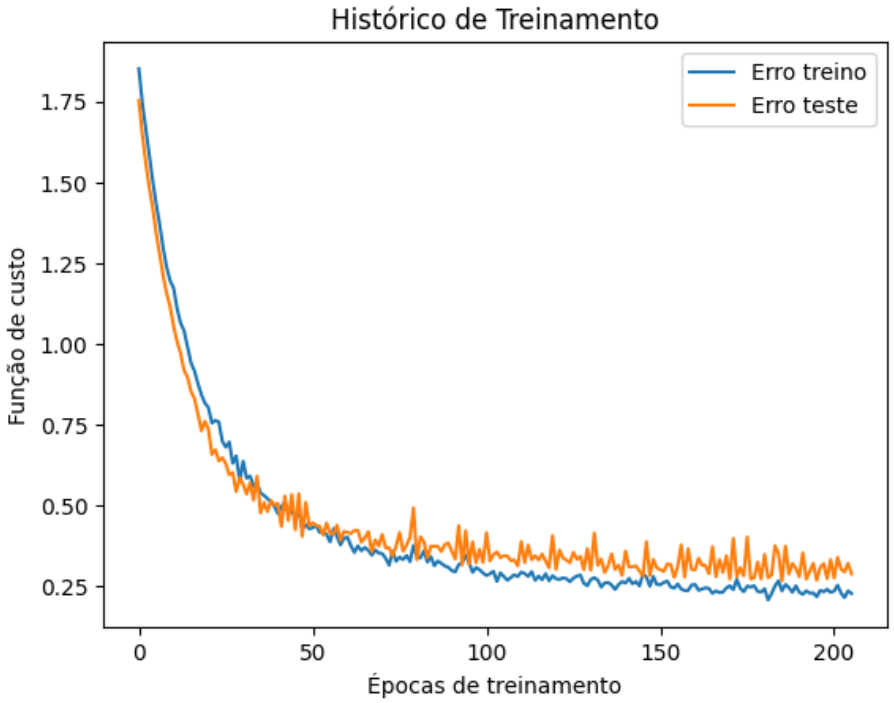
```
    import keras
model = keras.models.load_model('redeneural.cxc')
```

.

Em seguida, os estudantes devem ser incentivados a explorar diferentes parâmetros e a sensibilidade do modelo a diferentes tipos de dados: reduzir o número de iterações, ampliar a faixa de interesse, passando a prever notas menores ou maiores ou igual a 600, modificar percentual no conjunto de dados para treino e teste e alterar outros aspectos da programação para comparar resultados. Isso os ajudará a entender melhor como ajustar e otimizar o modelo para obter os melhores resultados.

Os resultados apresentados mostram o desempenho de um modelo de aprendizado de máquina ao longo de várias épocas de treinamento com as métricas de perda (loss) e acurácia (accuracy) para o conjunto de treinamento e validação e demonstram o desempenho de uma rede neural multicamadas em uma tarefa de classificação. Durante o treinamento, a rede atingiu uma acurácia de 98,2% no conjunto de dados de treinamento, o que indica que o modelo está ajustado de forma eficaz aos dados com os quais foi treinado. Entretanto, a perda de 0,2711 sugere que, embora a rede tenha aprendido a classificar bem os dados de treinamento, ainda existem erros que precisam ser minimizados. A acurácia no conjunto de teste, por sua vez, foi de 90%, refletindo um bom desempenho, mas uma queda significativa em relação à acurácia do conjunto de treino, conforme o gráfico da Figura 3.20:

Figura 3.20: Treinamento da RNA CXC



Fonte: Elaborada pela autora.

Esses resultados ressaltam a importância de avaliar não apenas a performance nos dados de treinamento, mas também a capacidade de generalização do modelo. Uma acurácia de 90% no conjunto de teste é encorajadora, mas destaca a necessidade de considerar estratégias adicionais para melhorar a robustez do modelo, como a regularização ou o aumento da diversidade do conjunto de dados. Para a implementação de redes neurais em contextos práticos, pode ser necessário ter esse cuidado, entretanto, esse é apenas

um modelo contextualizado de RNA multicamadas e não é o objetivo aprofundar nesse estudo.

Os dados fornecem, de forma complementar, as médias das notas em diferentes áreas para estudantes que não são treineiros, removendo valores não disponíveis (NaN):

- Ciências da Natureza (CN): 502.417341
- Ciências Humanas (CH): 557.193103
- Linguagens e Códigos (LC): 528.451163
- Matemática (MT): 538.505263
- Redação: 607.514451

Promova a interatividade, encoraje os jovens a fazer previsões com base nos dados fornecidos e comparando seus resultados com os da rede neural. Utilizando Python e Pandas, pode-se ter flexibilidade para manipular e inserir dados de forma eficiente em sua tabela, preparando-os para posterior análise e previsão com a rede neural desenvolvida.

```
# Supondo que df seja seu DataFrame carregado
novo_dado = {
    'Coluna1': valor1,
    'Coluna2': valor2,
    'Coluna3': valor3,
    # Adicione todas as colunas, conforme necessário,
    com seus nomes e valores.
}
# Adicionando uma nova linha ao DataFrame
df = df.append(novo_dado, ignore_index=True)
```

Isso os ajudará a entender melhor os conceitos apresentados dando-lhes a oportunidade de aplicá-los na prática, apesar das limitações de tempo e dos objetivos iniciais da sequência didática.

Esta aula está pervista para acontecer no laboratório de informática. Sintetize as principais conclusões obtidas ao longo da sequência didática, ouvindo as considerações e percepções de cada estudante.

3.1.11 Aula 11: Revisão dos conceitos estudados

A fim de obter feedback detalhado dos alunos sobre as aulas e o conteúdo abordado, realize uma atividade interativa de criação de nuvens de palavras. Nesta atividade, os alunos serão convidados a expressar suas opiniões e experiências de forma livre e informal, destacando as palavras-chave que melhor representem o que consideram a respeito da IA e, posteriormente, sobre esta sequência didática. Além da nuvem de palavras, incentive os alunos a criar um fluxograma no Canva, representando o que aprenderam sobre RNA. Essa abordagem permitirá que eles organizem visualmente os conceitos e relacionamentos discutidos em aula, facilitando a compreensão e a reflexão sobre o aprendizado, além de proporcionar mais uma forma de avaliação do conhecimento adquirido.

A nuvem de palavras é uma representação visual que destaca as palavras mais frequentemente mencionadas pelos alunos com tamanhos maiores pela maior frequência de ocorrência. Essa abordagem nos permite identificar a tendência das respostas.

Após a criação da nuvem de palavras, será aberto espaço para uma conversa amigável e informal, onde os alunos poderão compartilhar suas opiniões e considerações sobre a aula e o assunto. Essa troca de ideias permitirá uma compreensão mais profunda das necessidades e expectativas dos alunos, contribuindo significativamente para o aprimoramento contínuo do planejamento e execução das aulas.

Os resultados obtidos nas nuvens de palavras, os fluxogramas criados, juntamente com as considerações orais e os registros diários dos estudantes, serão apresentados no próximo capítulo, intitulado Resultados da Sequência Didática.

4 RESULTADOS DA SEQUÊNCIA DIDÁTICA

Primeira Aula: 16 de agosto de 2024

A primeira aula da sequência didática sobre IA ocorreu numa sexta-feira, com a participação de 20 dos 25 alunos matriculados. O objetivo dessa sequência é introduzir conceitos fundamentais de IA e sua aplicação em diversas áreas.

Após uma conversa inicial sobre os objetivos do trabalho, apresentei a sequência planejada das aulas, Figura 3.1. Notei que a turma demonstrou um interesse significativo pelo tema, mas também percebi que o conhecimento prévio sobre IA era quase inexistente. Durante a discussão, os alunos expressaram suas associações com IA. Uma aluna, ao ouvir o termo pela primeira vez, afirmou ter pensado em "robôs que destruiriam o mundo". Essa resposta provocou uma discussão interessante sobre os estereótipos relacionados à IA, permitindo que os alunos refletissem sobre suas opiniões.

Outra estudante mencionou ter participado de um curso de Python para iniciantes no ano anterior, e seis alunos da turma, que formam o grupo responsável pela assistência tecnológica da escola, mostraram-se especialmente entusiasmados com o tema. Este grupo é ativo na resolução de problemas tecnológicos que surgem na escola, lidando com computadores e retroprojetores.

Durante a aula, solicitei que os alunos completassem um questionário no "Google Forms", Apêndice A. No entanto, enfrentamos problemas com o sinal de internet na sala de aula, o que impossibilitou a realização da atividade naquele momento. Como o laboratório de informática já estava reservado para outro professor, optei por enviar o link do questionário no grupo de WhatsApp da turma, delegando aos alunos a responsabilidade de completá-lo o mais rápido possível. Devido ao tempo limitado, a curta de 5 minutos planejada foi adiada para a próxima aula.

Nesse dia, expliquei aos estudantes que a partir do terceiro encontro realizaríamos

atividades em grupo. Dividimos a turma em equipes de estudo, observando que a classe estava bastante fragmentada, com trios e quartetos que não se integravam bem aos demais. Dei-lhes liberdade para se organizarem da maneira que achassem mais conveniente e, ao final, registrei a formação de cinco grupos, compostos por três a sete integrantes cada. Fui informada de que uma das alunas estava em licença médica por tempo indeterminado, e, por isso, preparei uma atividade remota para que ela pudesse acompanhar as aulas. Elaborei orientações de pesquisa sobre o tema, solicitando que a aluna relatasse suas descobertas a partir dessa pesquisa.

Para concluir a aula, pedi que os estudantes buscassem notícias recentes sobre Inteligência Artificial para discutirmos na semana seguinte. A aula não apenas apresentou conceitos básicos sobre IA, mas também incentivou os alunos a refletirem sobre suas próprias percepções e experiências com tecnologia. Na próxima aula, espero que eles tragam notícias relevantes sobre IA para enriquecer nossa discussão e aprofundar o entendimento do tema.

Segunda Aula: 23 de agosto de 2024

Na segunda aula da sequência didática , estavam presentes 13 estudantes, representando 52% da turma. O foco desta aula foi aprofundar a discussão sobre as notícias relacionadas à IA e introduzir o conceito de RNA.

Apenas dois alunos trouxeram uma notícia relacionada à IA, enquanto os demais mencionaram que não tiveram tempo ou se esqueceram. As notícias trazidas foram uma de abril de 2024, publicada pela revista Veja, intitulada "Cientistas ensinam cão-robô a andar na Lua", que discute inovações em robótica, e outra da revista Superinteressante sobre "IA na medicina". Iniciamos a aula ouvindo essas notícias e abrindo espaço para que os estudantes comentassem sobre o assunto.

Alguns alunos demonstraram grande entusiasmo com a evolução da tecnologia, discutindo a relevância dos avanços mencionados e trazendo outras notícias que conheciam, como o desenvolvimento de robôs, carros autônomos e assistentes virtuais. Outros, porém, preferiram apenas ouvir, sem expressar suas opiniões, o que pode indicar timidez, insegurança sobre o tema ou a necessidade de estratégias que estimulem o engajamento. Para promover maior participação, uma opção seria incorporar debates em pequenos grupos, embora isso exija mais tempo de aula para uma discussão mais aprofundada.

Em seguida, apresentei o vídeo "O que é IA", que oferece uma explicação clara e

acessível sobre o conceito. O vídeo destaca como a IA, como qualquer ferramenta, pode ser utilizada de maneira positiva ou negativa, dependendo dos objetivos e da ética envolvidos. Aproveitei este ponto para discutir com os alunos a importância do uso ético da tecnologia, o que gerou algumas reflexões como o uso da IA para as campanhas eleitorais, o aumento de deepfakes e conteúdos manipulados que podem dificultar a distinção entre informações reais e falsas. Ressaltei, então, que a IA pode ser extremamente útil no processo de coleta, análise e classificação de informações sobre pessoas, grupos ou comportamentos para identificar características específicas. Isso permite realizar o perfilamento de eleitores, recurso muito útil em eleições acirradas. Essa estratégia permite adaptar mensagens para diferentes segmentos de eleitores, aumentando o impacto e a eficácia das campanhas.

Após o vídeo, comentei os resultados da pesquisa realizada por meio do questionário do Google. Quatro alunos não responderam ao questionário, o que me levou a refletir sobre possíveis obstáculos à participação, como falta de acesso à internet ou tempo. Muitos dos alunos são da zona rural do município, onde a internet é bastante instável, mas passam o dia todo na escola, o que sugere falta de prioridade ou curiosidade em relação à atividade. A estudante em licença médica, entretanto, participou ativamente e respondeu ao questionário.

As respostas ao questionário, conforme Anexo A, revelaram que embora a grande maioria dos alunos afirmassem ter alguma noção sobre IA, apenas 28,6% disseram ter um entendimento sólido de como ela funciona. Outro dado importante foi que 90,5% dos alunos nunca ouviram falar de RNA ou têm apenas uma ideia vaga, sem entender sua relação com a IA. Por outro lado, mais de 90% dos alunos demonstraram interesse em aprender mais sobre o tema, enquanto apenas um estudante afirmou não ter certeza se precisaria realmente entender IA.

Aproveitei os resultados do questionário para introduzir, pela primeira vez, o conceito de RNA. Expliquei que a IA engloba vários campos de estudo, e a RNA é um deles. Por meio de slides, apresentei a Figura 3.2, destacando alguns campos da IA.

Durante a explicação, mencionei o termo "neurônio", mas percebi que muitos estudantes não souberam associá-lo imediatamente a algo conhecido. Para facilitar a compreensão, expliquei que o neurônio é a célula do sistema nervoso e que a IA se inspirou no funcionamento dos neurônios biológicos. Usei imagens, lado a lado, de um neurônio humano e um neurônio artificial, comparando-os.

Adiantei que, na próxima aula, estudaríamos em mais detalhes como a RNA funciona e que ela é composta por neurônios artificiais interconectados. Também expliquei que a matemática desempenha um papel central na construção dessas redes, preparando os alunos para o conteúdo da próxima aula. Para tornar a explicação mais clara, planejei usar diagramas e analogias visuais, o que deverá facilitar a compreensão.

A aula despertou curiosidade em parte dos alunos, os meninos em sua totalidade e algumas meninas, mas ainda percebo a necessidade de incentivar uma participação mais ativa, especialmente em temas complexos como IA e RNA. A introdução de termos técnicos, como neurônio e RNA, será acompanhada por recursos visuais e exemplos mais concretos nas próximas aulas. Também buscarei novas maneiras de integrar a participação daquelas meninas que se mostraram mais tímidas, visando ampliar o debate e a compreensão coletiva.

Mais uma vez, o tempo foi insuficiente e, para a apresentação do vídeo História da IA, a turma sugeriu que eu o enviasse no grupo da sala para que assistissem durante o horário de almoço, e assim o fiz.

Terceira Aula: 30 de agosto de 2024

Nesta aula, novamente enfrentamos baixa frequência de estudantes, o que pode impactar a compreensão de conceitos importantes, como os neurônios artificiais e suas aplicações.

Em conversa com os demais professores da escola, compreendi que essa baixa frequência vem ocorrendo, em todas as turmas do ensino médio, de forma recorrente desde o primeiro semestre. Muitos estudantes demonstram insatisfação com o regime de tempo integral, que exige que permaneçam na escola por 9 horas diárias. Essa insatisfação tem levado a um elevado número de faltas ou a pedidos frequentes de saída antecipada, justificados por responsáveis com motivos como consultas médicas, mal-estar ou outros compromissos.

Iniciei a aula relembrando que um neurônio artificial é uma modelagem simplificada de um neurônio humano. Para ilustrar isso, apresentei uma imagem de um neurônio biológico, destacando suas principais partes: dendritos, núcleo e axônio 3.17. Expliquei a função de cada uma dessas estruturas no funcionamento do sistema nervoso humano e simulei uma sinapse para demonstrar a transmissão de impulsos nervosos. Enfatizei que os sinais elétricos, chamados potenciais de ação, percorrem o axônio e desencadeiam a

liberação de neurotransmissores nas sinapses, permitindo a comunicação entre neurônios e o envio de informações ao sistema nervoso central. Essa analogia é fundamental, pois a inspiração para criar redes neurais artificiais surgiu exatamente do funcionamento do cérebro humano, e entender essa conexão pode ajudá-los a ver a relevância do que estamos aprendendo.

Em seguida, utilizei um slide onde comparei o neurônio biológico com o neurônio artificial 2.3. Nesta comparação, defini os principais elementos de um neurônio artificial: entradas (inputs), pesos, somatório, função de ativação e saída (output). Expliquei como esses componentes se inter-relacionam e como a função de ativação desempenha um papel crucial no processamento das informações dentro de uma rede neural.

Prosseguindo, introduzi o conceito de funções de ativação, ressaltando que iríamos conhecer três novas funções: a função degrau, a função sigmoide e a função ReLU. Neste momento, percebi que muitos estudantes se mostraram preocupados com a ideia de aplicar funções neste estudo. Acredito que este tema tão importante ainda não lhes trouxe a segurança do aprendizado. Contudo, disse-lhes que se tranquilizassem, pois eram conceitos simples e básicos. Redefini o termo função, recordando como o assunto deve ter-lhes sido apresentado, assumindo alguns exemplos simples, como o valor a pagar em função da quantidade, a nota da prova em função do número de acertos de questões e o tempo usado para um trajeto em função da velocidade do automóvel. A seguir, apresentei-lhes, uma a uma, as funções que usaríamos nas próximas aulas, matemática e visualmente: 3.9, 3.10, 3.11. Com a projeção de imagens dos gráficos, ilustrei como cada função se comporta em relação às entradas, facilitando a compreensão.

A comparação entre essas funções foi importante para que os alunos entendessem como diferentes mecanismos podem ser utilizados para processar informações em um neurônio artificial. Neste momento, já mostravam-se tranquilos em relação a estas funções.

Por fim, anunciei que, na próxima aula, explicaria como essas funções de ativação são aplicadas na criação de um perceptron, o modelo mais simples de neurônio artificial. Ressaltei que essa compreensão é uma base sólida para estudos futuros em redes neurais mais complexas.

Para consolidar o conteúdo apresentado, deixei uma tarefa prática de cálculos envolvendo as três funções de ativação discutidas em aula e detalhadas no Apêndice B. O objetivo da tarefa é que os alunos compreendam, na prática, como cada função transforma

as entradas e o impacto que isso tem na saída do neurônio artificial. Também incentivei-os a trazer dúvidas para a próxima aula, pois funções de ativação podem ser um conceito novo e desafiador.

Quarta Aula: 4 de setembro de 2024

No início de setembro, consegui ajustar meu horário ao trocar com a professora da disciplina de Tecnologia e Inovação, que também leciona Filosofia para a mesma turma. Agora, posso ministrar as aulas no primeiro período da tarde, em vez do último horário da semana. Essa mudança foi motivada tanto pela minha disponibilidade quanto pela intenção de melhorar a frequência dos estudantes, que vinha sendo um desafio até então.

Com a turma completa nesta aula, aproveitei a oportunidade para retomar alguns conceitos essenciais. Reforcei que um perceptron é um neurônio artificial simplificado, utilizando novamente a imagem comparativa do neurônio humano e do artificial, Figura 2.3, para explicar o fluxo de dados, desde as entradas até a saída gerada pela função de ativação.

Revisamos os gráficos das três funções de ativação estudadas anteriormente: Figuras 3.11, 3.9 e 3.10. Ao solicitar que os alunos entregassem os cálculos da tarefa proposta na última aula, notei que nenhum grupo havia concluído a atividade, e percebi que muitos sequer haviam começado. Sugeri que os grupos de estudo se encontrassem durante os intervalos entre as aulas para concluir a tarefa até a semana seguinte. Essa estratégia não só estimula a colaboração entre os alunos, mas também reforça o aprendizado coletivo.

Em seguida, avancei com o conteúdo da aula, explicando que o perceptron é utilizado exclusivamente em situações onde os dados são linearmente separáveis. Para tornar esse conceito mais claro, mostrei exemplos visuais que ilustravam a separabilidade linear, citando os exemplos dos conectivos lógicos AND e XOR: Figura 3.13.

Introduzi o conceito de rede perceptron, utilizando o conectivo lógico AND como exemplo. Para simplificar, apresentei as possíveis combinações de valores de verdade para duas proposições simples. Usei o exemplo "quero café"e "quero leite"como as proposições básicas, e a combinação delas resulta em "quero café com leite", que analisamos pela lógica. Esta analogia ajudou os alunos a visualizarem melhor o conceito, tornando-o mais relevante e próximo do cotidiano deles.

Para ajustar o tempo da aula e retomar conceitos iniciais, replanejei o conteúdo, optando por não abordar todos os exemplos que havia preparado para explicar esses

conceitos. Resumi o uso dos conectivos lógicos, focando principalmente em AND, OR e XOR, que seriam os mais relevantes para esta sequência didática.

Expliquei a importância da tabela-verdade no estudo da lógica, ressaltando que esse conteúdo é amplamente aplicável em diversas áreas profissionais e frequentemente cobrado em concursos, especialmente pela sua utilidade na tomada de decisões. Assim como as quatro operações são a base da matemática, a tabela-verdade é a base do raciocínio lógico.

Além disso, esclareci que as entradas de um neurônio artificial são representadas por valores numéricos e que, para isso, usaríamos uma codificação. Neste caso, atribuímos o valor 1 para verdadeiro e 0 para falso. Apliquei essa codificação nas possíveis entradas de uma rede perceptron para o conectivo AND, destacando a lógica por trás das decisões tomadas por esse tipo de rede. Informei que continuaríamos com esse estudo na próxima aula.

Quinta Aula: 11 de setembro de 2024

Nesta aula, o objetivo foi descrever manualmente o funcionamento de uma rede perceptron usando o conectivo lógico AND. Expliquei que o problema poderia ser encarado como a busca por uma fórmula matemática que, baseada no modelo do perceptron, ensinaria a rede a prever corretamente o valor lógico AND. Ou seja, a rede seria treinada para retornar um resultado verdadeiro apenas quando todas as entradas fossem verdadeiras.

Relembramos a codificação padrão para as entradas, onde 1 representa verdadeiro e 0 representa falso. Registrei as entradas da rede perceptron e sugeri, inicialmente, pesos nulos e um bias de -1. Aproveitei para reforçar que, ao iniciar o treinamento, os pesos geralmente são atribuídos de forma aleatória e ajustados a cada etapa com base no erro calculado. Em redes neurais mais complexas, valores de pesos idênticos podem causar um problema de simetria, dificultando o aprendizado da rede. No entanto, como estamos lidando com uma estrutura simplificada com apenas duas entradas, começaremos com pesos iguais a zero para facilitar o entendimento do processo em apenas duas iterações.

Para facilitar o processo de treinamento manual, sugeri uma taxa de aprendizagem de 0,4. Expliquei que, na prática, essa taxa precisa ser cuidadosamente ajustada para garantir um aprendizado eficiente, evitando tanto uma convergência muito lenta quanto resultados imprecisos. A taxa de aprendizagem interfere diretamente no tamanho do ajuste que a rede fará nos pesos e bias sempre que houver uma divergência entre a saída da rede e a saída esperada.

Apliquei então o primeiro par de entradas, somando o produto dos pesos pelas entradas e adicionando o bias. Registrei o resultado e, em seguida, passei pela função de ativação degrau para determinar a saída. No próximo par de entradas, houve uma discrepância entre o valor calculado e o esperado segundo a tabela verdade do AND, o que indicou a presença de um erro. Nesse momento, mostrei aos alunos como reajustar os pesos e o bias, reintroduzindo os conceitos da regra Delta, que corrige os pesos com base na diferença entre o valor esperado e o valor obtido, para guiar o processo de ajuste.

Percebi que, à medida que os cálculos avançavam, alguns estudantes começaram a se dispersar. Um deles comentou que preferia avançar diretamente para a programação da rede, considerando o processo manual tedioso e dispensável. No entanto, decidi retomar os cálculos de forma mais simplificada, enfatizando a importância de uma compreensão sólida dos conceitos. Expliquei que identificar os pontos onde o algoritmo matemático pode ser automatizado é fundamental para a programação eficiente. Isso também serviu como preparação para a introdução à base do Python, que será abordada na próxima aula.

Para essa aula, estava planejada uma atividade prática em grupos: refazer os cálculos, mas desta vez criando um perceptron para o conectivo lógico OR, conforme o AneC. O objetivo era permitir que analisassem o processo, realizando os ajustes necessários.

Deixei as orientações com os grupos, mas expliquei que o conteúdo seria retomado na próxima aula, ajudando-os a se organizar melhor para a atividade. Pedi que, por enquanto, apenas lessem as instruções e refletissem sobre como poderiam agir para resolver o problema.

No entanto, precisei lembrar a turma que ninguém havia entregue a primeira tarefa, o que já indicava uma falta de compromisso com o estudo. Além disso, observei que, durante as aulas, ninguém fazia registros ou anotações, o que dificultava o acompanhamento da aula. Percebi que muitos alunos estavam preocupados com a avaliação de química que teriam na aula seguinte, o que desviou sua atenção. Embora a sala estivesse cheia, senti que, novamente, eu estava falando com uma minoria dos estudantes.

Para otimizar o ensino da construção manual de um perceptron e evitar que a aula se torne cansativa, sugiro focar nas atualizações de pesos e bias apenas quando houver erro, ignorando as atualizações desnecessárias em casos de erro zero. Além disso, é recomendado intercalar as explicações teóricas com visualizações gráficas, para tornar

a aula mais dinâmica e manter o engajamento dos alunos. Destacar as correções mais significativas e lembrá-los que, na sula seguinte, utilizarão simulações digitais para acelerar o processo, mantendo o equilíbrio entre teoria e prática.

Sexta Aula: 18 de setembro de 2024

Nesta aula, precisei retomar e resumir a explicação dada na aula anterior, quando utilizei um quadro para relembrar os conceitos estudados, pois nenhum dos alunos fez registros durante a última aula. Neste momento, eu tinha em mão uma síntese do processo de construção de uma RNA perceptron AND, impressa para cada um dos estudantes. Essa base era essencial para que pudessem avançar para a programação desta RNA. Com quase todos os estudantes presentes, organizei-os em grupos para que, sob minha supervisão, iniciassem a resolução da atividade sobre o conectivo OR, a partir da análise da RNA do conectivo AND.

Entretanto, percebi novamente que os grupos enfrentaram dificuldades em manter o foco. Eles não se concentravam para ler o material de apoio que preparei e incluí no Apêndice F para complementar o planejamento inicial, mostrando ainda uma grande dependência da minha orientação. A dispersão foi evidente, com muitos estudantes distraídos, preocupados com a prova de biologia que teriam na aula seguinte. Infelizmente, a aula se mostrou pouco produtiva, repetindo o cenário da aula anterior.

Concluí a aula recomendando que, ao menos, lessem o material de apoio e tentassem realizar a atividade proposta.

De acordo com o planejamento inicial, essa sexta aula já seria realizada no laboratório de informática para darmos início à programação, mas essa etapa precisou ser adiada para o próximo encontro.

Sétima Aula: 25 de setembro de 2024

Na aula de hoje, trabalhei com os alunos os conceitos básicos de Python no Google Colab, com o objetivo principal de que compreendessem o conceito de perceptron e sua aplicação como modelo de neurônio artificial. Iniciei projetando no quadro alguns códigos simples que deveriam ser reproduzidos no Colab, explicando passo a passo os comandos essenciais de Python conforme o planejamento inicial. A turma pôde acompanhar e replicar os códigos no ambiente, fazendo pequenas alterações, executando-os e observando como essas mudanças influenciavam os resultados. Essa abordagem prática reforçou o entendimento sobre entradas, saídas e a função de ativação.

Fiquei satisfeita ao ver alguns estudantes modificando outras variáveis, explorando além do sugerido e analisando as novas saídas apresentadas. Isso mostrou que, por meio da programação, eles poderiam adquirir uma compreensão sólida do funcionamento de uma RNA.

Para concluir a explicação, introduzi o conceito de função de ativação e programei três delas: degrau, sigmoide e ReLU. Confira os códigos nos Apêndices F, G e H, respectivamente.

Projetei novamente os códigos no quadro e os enviei pelo WhatsApp para que os alunos pudessem dedicar mais tempo a explorar cada função E. Posteriormente, projetei novamente os gráficos destas funções que comprovavam os resultados apresentados pela programação.

Pedi que executassem os scripts e observassem o comportamento de cada função de ativação. A tarefa inicial, proposta na terceira aula, envolvia calcular essas funções manualmente; agora, na sétima aula, os alunos repetiriam o cálculo utilizando a programação em Python. Como o cálculo manual não foi realizado anteriormente, sugeri que comparassem os resultados obtidos com os valores da tarefa e experimentassem ajustar os parâmetros para observar diferentes saídas. Assim, puderam perceber a simplicidade da execução programada e visualizar os resultados em gráficos projetados, discutindo as diferenças entre as funções.

A atividade foi bem recebida pela turma, que demonstrou grande envolvimento ao longo da aula, participando ativamente, testando o código e propondo suas próprias variações. Ao final, o interesse e o progresso dos alunos em relação aos conceitos abordados foram evidentes. Na próxima aula, concluiremos nossas observações sobre as funções de ativação na construção de uma rede perceptron e daremos continuidade a outros conceitos essenciais para a programação da rede perceptron AND.

Durante toda a aula, mantive o foco no entendimento da lógica de programação, retomando sempre os conceitos fundamentais da construção de uma rede neural e preparando a turma para o entendimento completo da programação da RNA perceptron AND.

Oitava Aula: 2 de outubro de 2024

Nesta aula, concentrei-me no entendimento da lógica de programação por trás do perceptron, destacando a importância de cada linha de código. À medida que os alunos

executavam os programas e faziam ajustes, discutimos as implicações de suas modificações e como elas influenciavam o funcionamento da rede neural.

No entanto, observei que muitos estudantes se perdiam em seus códigos devido a pequenos erros, como uma indentação incorreta, a seleção inadequada de um símbolo ao copiar, ou até mesmo um espaço extra que impedia o Colab de executar os comandos corretamente para a construção do perceptron AND. Esses problemas consumiram boa parte do tempo, não consegui avançar até o ponto em que pretendia introduzir a necessidade de redes multicamadas.

Por outro lado, no contato direto com os alunos, percebi um grande interesse em entender como tudo funciona na programação. Eles se surpreendiam com o impacto que pequenos detalhes tinham no resultado esperado. Aproveitei essa oportunidade para esclarecer a diferença entre programação e inteligência artificial, retomando o conceito de aprendizagem da rede. Também expliquei que, em redes mais complexas, são usadas múltiplas funções de ativação e que o objetivo final é que a rede seja capaz de tomar decisões, de forma "autônoma".

Esta aula corresponde ao planejamento da Aula 7, que incluía a Tarefa 7, descrita no Apêndice D. No entanto, como nenhuma das tarefas anteriores foi realizada em casa, a Tarefa 7, planejada como uma atividade complementar destinada a reforçar os conceitos de programação e RNA, não pode ser executada. Não tínhamos tempo disponível para executá-la em sala de aula.

Nona Aula: 9 de outubro de 2024

Iniciamos a aula retomando o uso do Colab para revisar os códigos da rede perceptron AND e observar as saídas após duas iterações que permitiriam a convergência da rede.

Reforcei a importância do planejamento antes da programação: definir claramente o objetivo, considerar as melhores abordagens e tomar decisões fundamentadas. Compartilhei com eles meu próprio processo de escolha de pesos e bias para o cálculo manual. Expliquei que foi um trabalho experimental, no qual testei diferentes configurações, mantendo o bias fixo em algumas tentativas, avaliando a taxa de aprendizagem ideal para acelerar o processo e ainda garantir um bom resultado. Dada a natureza manual do trabalho, optei por valores de pesos e bias aleatórios para simplificar e agilizar o processo.

Mostrei que, para adaptar o exemplo da rede AND e programar a rede perceptron

OR, é preciso apenas considerar as diferenças lógicas entre esses conectivos e analisar a separabilidade dos dados. Juntos, revisamos esses pontos, fizemos as alterações necessárias nas saídas esperadas, executamos o código e discutimos os novos resultados.

Aproveitamos para esclarecer a tarefa solicitada na quinta aula: a construção manual do perceptron OR com base no perceptron AND.

De acordo com o planejamento inicial, esta nona aula seria dedicada às apresentações dos resultados dos trabalhos em grupo. No entanto, esta parte do planejamento não se concretizou como esperado, o que dificultou a avaliação do aprendizado dos estudantes, uma vez que não desenvolveram as atividades propostas.

Décima Aula: 23 de outubro de 2024

Nesta aula, o objetivo foi introduzir a estrutura de uma rede neural multicamadas. Começamos revisitando o conceito de uma RNA que resolve o problema do XOR, cujos dados de entrada não são linearmente separáveis, como vimos anteriormente. Retomei a Figura 3.19, reforçando que, nesse caso, o modelo de perceptron simples não é suficiente, e por isso precisávamos explorar a arquitetura de uma rede multicamadas. Em seguida, analisamos a imagem de uma RNA com múltiplas entradas 2.1, para entender melhor essa estrutura.

O objetivo foi demonstrar aos estudantes que um perceptron de camada única tem, de fato, as limitações já mencionadas, sendo capaz de lidar apenas com dados linearmente separáveis. Em contraste, uma rede neural multicamadas é capaz de superar essas limitações, resolvendo problemas complexos que os métodos tradicionais de programação não conseguem abordar de forma eficaz.

Em continuidade, introduzi um exemplo prático de rede neural construída com dados históricos do INEP, especificamente de estudantes da nossa escola desde 2009. Mostrei a planilha do Excel com os dados consolidados: 348 linhas e 436 colunas, correspondentes às variáveis de entrada da rede. Observei a reação dos estudantes, alguns demonstraram surpresa e até receio com o tamanho e complexidade da base de dados — o que talvez os ajudou a perceber a importância da programação para esse tipo de análise. Também lhes apresentei o dicionário de variáveis do ENEM 2022, e discutimos as variáveis não quantitativas e como elas poderiam ser transformadas e utilizadas na programação.

Expliquei que, ao explorar essa grande tabela, poderíamos identificar informações específicas, como a maior e a menor nota, médias, valores modais e medianos, o que desper-

tou bastante curiosidade na turma. No entanto, alguns alunos expressaram preocupação com a dificuldade de incluir seus próprios dados na rede e obter uma previsão preliminar de suas notas no ENEM. Esclareci que, embora esse processo seja viável com programação, ele demanda tempo, especialmente devido à quantidade de variáveis envolvidas. Propus que, para facilitar, experimentassem construir uma rede neural com um número reduzido de variáveis, selecionando as mais relevantes. Comentei, no entanto, que uma rede tende a produzir melhores resultados quando treinada com uma maior diversidade de variáveis.

Introduzi os conceitos de dados para treinamento e dados para teste, e apresentei a Figura 3.18, destacando que a rede, conforme programada, oferece um bom grau de confiabilidade nas previsões. Porém, ressaltei a importância de uma análise especializada para validar os resultados, considerando que meu objetivo foi apenas contextualizar o uso de tecnologias avançadas em nossa prática de aula.

Os estudantes mostraram grande curiosidade em relação à programação da rede, embora inicialmente tenham ficado intimidados com a quantidade de códigos. Eles ficaram eufóricos ao vislumbrar a possibilidade de prever resultados de provas, como as do ENEM, antes mesmo de realizá-las. Os mais otimistas se empolgaram com a ideia dessas previsões, enquanto outros, que se sentiram desmotivados desde o início da sequência didática, consideraram o conteúdo muito complexo. Reforcei, mais uma vez, que este era apenas o primeiro contato deles com programação e redes neurais e que, com a continuidade dos estudos, esses conceitos certamente se tornariam mais familiares e práticos, sendo assimilados com mais naturalidade.

Finalizei explicando que, no próximo e último encontro, faríamos uma revisão rápida dos principais tópicos abordados, sintetizando tudo o que discutimos ao longo das aulas.

Aula 11: 30 de outubro de 2024

Nesta última aula da sequência didática, iniciei abrindo espaço para que os alunos compartilhassem suas percepções e experiências ao longo deste Itinerário Formativo em Tecnologia e Inovação. De modo geral, eles expressaram que, embora considerem o tema interessante e necessário, também o veem como muito complexo e desafiador. Alguns até afirmaram não ter afinidade com esses estudos. Aproveitei para relatar minha própria trajetória, lembrando-lhes que, no início, também encontrei dificuldades em programação e no entendimento de redes neurais — uma área que, em um primeiro

momento, parecia distante e inatingível para mim. Ressaltei, porém, a importância de persistirmos: a tecnologia avança rapidamente e, embora o aprendizado demande esforço, é uma habilidade essencial para nos mantermos atualizados.

Em seguida, compartilhei um link e pedi que resumissem suas experiências em três palavras, incluindo emoções, aprendizados e desafios enfrentados ao longo do curso. Com as palavras escolhidas, construímos juntos uma nuvem de palavras, que projetei no quadro. Ao ler algumas delas, confesso que senti um leve desapontamento ao ver expressões como "não gosto", "cansativo", "confuso", "sono". Porém, senti também satisfação ao notar que as palavras "interessante", "difícil", foram as que mais se destacaram, indicando que, apesar dos desafios, a turma reconheceu o valor do aprendizado, como é possível observar a partir da Figura 4.1:

Figura 4.1: Percepções dos Alunos sobre a Experiência de Aprendizagem

Fonte: Elaborada pela autora.

Para concluir, reafirmei que este foi apenas o primeiro contato com uma área nova e desafiadora, e que é natural encontrar obstáculos. Encorajei-os a continuar explorando, mantendo a curiosidade e a disposição para o aprendizado, pois a jornada tecnológica é longa, mas gratificante. Finalizamos com uma breve revisão das principais etapas que percorremos juntos, reforçando a importância do esforço contínuo e da resiliência em um campo que oferece inúmeras oportunidades para quem se dispõe a aprender.

5 CONSIDERAÇÕES FINAIS

A proposta de integrar a MM e as RNA no EM, com o objetivo de desenvolver o PCO, mostrou-se desafiadora e promissora ao mesmo tempo. A sequência didática, implementada ao longo de onze aulas, permitiu aos alunos do 3º ano da EECXC um primeiro contato com conceitos fundamentais de IA e RNA, conectando-os ao conteúdo matemático e tecnológico de forma teórica e prática. Uma das maiores contribuições da matemática para esse trabalho foi a modelagem das funções de ativação, que proporcionou uma melhor compreensão do funcionamento das redes neurais. Quando os alunos observaram o processo manual de construção da RNA, eles não apenas repetiram cálculos, mas também buscaram padrões, permitindo de forma natural o desenvolvimento do PCO. Esse processo os preparou para a próxima etapa: a programação das redes neurais, uma vez que entender as repetições e padrões é um caminho direto para a implementação algorítmica.

Ensinar conceitos complexos, como redes neurais e programação, para alunos de EM é desafiador, e suas dificuldades já eram esperadas. Isso, por si só, é uma contribuição importante ao campo da educação, pois revela o que funciona e o que ainda precisa ser ajustado para abordar temas complexos em contextos escolares.

Desde a primeira aula, ficou claro que os estudantes tinham pouco ou nenhum conhecimento prévio sobre IA e RNA, o que era esperado. No entanto, eles demonstraram interesse em estudar esses conceitos. Vídeos, diagramas e exemplos comparativos entre neurônios biológicos e artificiais foram fundamentais para facilitar a compreensão de termos técnicos como neurônio, funções de ativação e perceptron. Contudo, a falta de uma base sólida ou mesmo interesse em matemática causou, em diversos momentos, uma certa insegurança entre os alunos, dificultando a participação ativa. Essa lacuna impactou diretamente a compreensão de conceitos mais avançados, como a aplicação das funções de ativação e o reajuste de pesos nas redes neurais, tornando o ritmo de aprendizado mais lento. Por outro lado, a curiosidade em relação à IA gerou interesse natural pela discussão, apesar de alguns tópicos apresentarem maior dificuldade de compreensão.

Diante da percepção de que alguns estudantes tiveram dificuldade em compreender conceitos fundamentais das RNA, como funções de ativação e reajuste de pesos, devido à falta de uma base sólida em matemática, é importante ressaltar que a matemática utilizada nas atividades foi bastante simples e poderia ser aplicada em qualquer turma do EM, do primeiro ao terceiro ano. Os cálculos envolvidos foram acessíveis e compatíveis com o nível de ensino, sendo planejados de modo a não sobrecarregar os alunos com fórmulas complexas. A dificuldade, portanto, não esteve na complexidade matemática, mas na aceitação e familiaridade com a disciplina. Isso evidencia a necessidade de um maior investimento na construção de uma base matemática consistente ao longo do ensino básico, de forma a garantir que os alunos estejam mais preparados para enfrentar desafios que envolvam aplicações tecnológicas, como RNA.

Momentos de entusiasmo foram especialmente evidentes entre os alunos que fazem parte da equipe de assistência tecnológica da escola, que se engajaram com mais facilidade nas discussões. No entanto, a não adesão às tarefas externas indicou a necessidade de repensar as estratégias de incentivo e avaliação. Fatores como o acesso limitado à internet e a falta de tempo para atividades complementares sugerem que a adaptação ao contexto dos alunos, especialmente aqueles da zona rural, poderia melhorar o engajamento.

A coleta de dados, por meio de questionários sobre o conhecimento prévio e o interesse dos alunos, gerou informações valiosas. A pesquisa revelou que a maioria dos estudantes nunca havia ouvido falar de redes neurais artificiais, mas demonstrou curiosidade em aprender mais sobre o tema. As respostas à pergunta: Você faz uso da inteligência artificial no seu dia a dia? Conte-me quando ou como, destacaram uma clara insegurança quanto ao entendimento desse tópico tão atual, revelaram o quanto ele ainda é pouco compreendido. Esse resultado reforça a necessidade de continuar explorando o tema em sala de aula, contextualizando seu uso prático e ético, e evidenciando sua importância no cotidiano e no cenário global.

Outro desafio observado foi a tendência dos alunos em priorizar disciplinas que exigem avaliação formal, o que afetou diretamente o engajamento com a sequência didática. Em alguns momentos, metade da turma estava concentrada em cadernos de química ou biologia, preocupada com as avaliações dessas disciplinas. Isso levanta questionamentos sobre a adequação dos itinerários formativos para a aplicação dessa sequência, já que muitos alunos concentram seus esforços nas matérias que consideram mais importantes

para sua aprovação. Esse cenário reforça a necessidade de repensar o ambiente e o contexto em que a sequência didática é aplicada.

Além disso, foi necessário ajustar os horários de aula com a turma para aumentar a frequência dos alunos. Após conversar com outros professores e supervisores, percebi que a baixa frequência é um problema recorrente no NEM. Muitos alunos recebem autorização familiar para sair mais cedo ou faltam às aulas sem justificativa, devido à dificuldade de se adaptarem à carga horária exigida. Isso destaca a importância de encontrar soluções que equilibrem as exigências curriculares com a realidade dos estudantes, para que o aprendizado ocorra de maneira eficaz e motivadora.

Os resultados finais desta sequência didática indicam que, apesar dos desafios, a abordagem tem grande potencial para contribuir significativamente para a formação dos alunos. A integração de temas inovadores, como IA e RNA, ao conteúdo de matemática e tecnologia desperta o interesse dos estudantes e os conecta a questões contemporâneas, preparando-os para desafios futuros, tanto acadêmicos quanto profissionais.

Apesar das dificuldades, a sequência didática foi uma oportunidade para os alunos exercitarem a resiliência, que é fundamental para o aprendizado em áreas técnicas.

Por fim, as considerações para o aprimoramento deste projeto incluem o fortalecimento das estratégias de participação e engajamento dos alunos, o ajuste das atividades práticas para garantir maior adesão e a continuidade do uso de recursos visuais e analogias que facilitem a compreensão dos conceitos técnicos. Para os alunos do ensino integral, uma alternativa seria incluir aulas específicas para o desenvolvimento de atividades em grupo, que nesta sequência foram designadas como tarefas para casa. Outra possibilidade seria eliminar as tarefas manuais e, após a demosntração do processo manual, focar diretamente na programação, solicitando que todos os cálculos e experimentos fossem realizados no Google Colab. No entanto, para que isso fosse eficaz, os estudantes precisariam primeiro identificar claramente os padrões de repetição que possibilitariam a programação da rede. Além disso, a introdução de quizzes como forma de avaliação ao longo da sequência pode incentivar a reflexão e a busca pelo conhecimento, substituindo as atividades escritas que os alunos não aceitam bem. Essas estratégias, porém, demandariam mais tempo e mais aulas, o que infelizmente não se encaixou no cronograma disponível.

Outra possibilidade para fomentar a participação dos alunos nas atividades próprias desta sequência é através de projetos, em que os alunos que gostariam de trilhar o

ensino superior em áreas relacionadas a matemática aplicada ou ciências da computação, sejam escolhidos. Finalmente, com esse trabalho, espero ter contribuído e incentivado a abordagem das RNA desde o EM, em vista que esse conhecimento está impactando os ambientes de trabalho, de ensino, gestão, entre outros de maneira crescente e desafiadora.

REFERÊNCIAS

1 EDUCAçãO, B. *Estudo alerta para a urgência de ações em relação ao ensino de Matemática no País*. 2023. Disponível em: <https://www.b3.com.br/pt_br/noticias/estudo-alerta-para-a-urgencia-de-acoes-em-relacao-ao-ensino-de-matematica-no-pais.htm>. Acesso em: 26 dez. 2024.

2 Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira. *Publicados resultados preliminares do Censo Escolar 2023*. 2023. Disponível em: <https://www.gov.br/inep/pt-br/assuntos/noticias/censo-escolar/publicados-resultados-preliminares-do-censo-escolar-2023>. Acesso em: 17 out. 2023.

3 Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira. *Divulgados os resultados do PISA 2022*. 2023. Disponível em: <https://www.gov.br/inep/pt-br/assuntos/noticias/acoes-internacionais/divulgados-os-resultados-do-pisa-2022>. Acesso em: 17 out. 2023.

4 OLIVEIRA, C. R.; SILVA, M. L. *A inteligência artificial no ensino*: personalização e acessibilidade para o século xxi. *Cadernos Uninter de Práticas Educativas*, v. 10, n. 2, p. 45–56, jul 2022. Disponível em: <https://www.cadernosuninter.com/index.php/intersaberes/article/view/2964/2145>. Acesso em: 17 jan. 2025.

5 FURTADO, M. I. V. *Redes neurais artificiais: uma abordagem para sala de aula*. Ponta Grossa, PR: Atena Editora, 2019.

6 Sociedade Brasileira de Matemática. *Redes neurais no ensino básico*. 2022. Disponível em: <https://pmo.sbm.org.br/wp-content/uploads/sites/5/sites/5/2022/09/art32_vol10_SBM_PMO_2022.pdf>. Acesso em: 17 jan. 2025.

7 COLLE, J. E.; RODRIGUES, W. M.; SILVA, W. R. da. Redes neurais artificiais: ideias básicas no ensino médio. *Brazilian Journal of Development*, v. 8, n. 3, p. 21795–21810, 2022. Disponível em: <https://doi.org/10.34117/bjdv8n3-399>. Acesso em: 21 jan. 2025.

8 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: <https://mid-educacao.curitiba.pr.gov.br/2017/10/pdf/00156542.pdf>. Acesso em: 17 jan. 2025.

9 BASSANEZI, R. C. *Ensino-aprendizagem com modelagem matemática: uma nova estratégia*. São Paulo: Unicamp, 2002.

10 PEREIRA, L. E. *Contribuições da modelagem matemática para o desenvolvimento de habilidades de resolução de problemas no ensino médio*. 2023. Disponível em: <https://www.bdtd.ueg.br/handle/tede/1462>. Acesso em: 17 jan. 2025.

11 HAYKIN, S. *Redes neurais: princípios e prática*. Porto Alegre: Editora Bookman, 2001.

12 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*.

13 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.

14 Data Science Academy. *Uma breve história das redes neurais artificiais*. Disponível em: <https://www.deeplearningbook.com.br/uma-breve-historia-das-redes-neurais-artificiais/>. Acesso em: 14 jan. 2025.

15 KANDEL, E. R.; SCHWARTZ, J. H.; JESSELL, T. M. *Princípios de neurociência*. Porto Alegre, Brasil: AMGH Editora, 2000.

16 HAYKIN, S. *Redes neurais e máquinas de aprendizado*. 3ª. ed. Upper Saddle River, NJ: Pearson Education, 2008. Título original: Neural Networks and Learning Machines.

17 PEREIRA, F. G. S. G. *Redes neurais artificiais: fundamentos e aplicações*. Rio de Janeiro: Editora LTC, 2010.

18 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.

19 SILVA, D. H. S. e. R. A. F. Ivan Nunes da. *Redes neurais artificiais para engenharia e ciências aplicadas*. São Paulo: Editora Artliber, 2010.

20 Deep Learning Book Brasil. *Função de Ativação*. 2025. Disponível em: <https://www.deeplearningbook.com.br/funcao-de-ativacao/>. Acesso em: 17 jan. 2025.

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

22 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.

23 BATISTA, E. J. S. *Pensamento Computacional: Teoria e Prática*. Campo Grande, MS: Universidade Federal de Mato Grosso do Sul, 2023. Disponível em: <https://repositorio.ufms.br/handle/123456789/8876>. Acesso em: 14 jan. 2025.

24 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: <https://doi.org/10.1234/rbet.v12i3.5678>. Acesso em: 21 jan. 2025.

25 BRACKMANN, C. P. *Pensamento Computacional Desplugado*: Ensino e avaliação na educação primária espanhola. *Journal on Computational Thinking (JCThink)*, v. 2, n. 1, p. 36–36, 2018. Disponível em: <https://periodicos.univali.br/index.php/IJCThink/article/view/12415>. Acesso em: 16 jan. 2025.

26 Ministério da Educação. *Base Nacional Comum Curricular: Ensino Médio - Matemática*. 2018. Disponível em: <http://basenacionalcomum.mec.gov.br/>. Acesso em: 22 dez. 2023.

27 SILVA, E. F. Itinerários formativos no novo ensino médio: maiores oportunidades ou aligeiramento da formação? *Projeção e Docência*, v. 15, n. 1, p. e1524DO08, 2018. Disponível em: <https://projecaociencia.com.br/index.php/Projecao3/article/view/2389>. Acesso em: 14 jan. 2025.

28 BRASIL. *Lei nº 13.415, de 16 de fevereiro de 2017*. <http://www.planalto.gov.br/ccivil_03/_ato2015-2018/2017/lei/l13415.htm>. Acesso em: 26 dez. 2024.

29 ALMEIDA, R. S. d. et al. Itinerários formativos no novo ensino médio e os desafios para a educação no brasil. *IOSR Journal of Humanities and Social Science (IOSR-JHSS)*, v. 29, n. 3, Series 4, p. 37–42, 2024. Disponível em: <https://doi.org/10.9790/0837-2903043742>. Acesso em: 21 jan. 2025.

30 PACHECO, J. F.; CORDEIRO, M. J. de J. A. Ensino médio com itinerários formativos: uma análise da nova proposta sob a perspectiva da formação integral. *Revista Ft*, v. 27, n. 120, p. 29–29, mar. 2023. Disponível em: <https://revistaft.com.br/ensino-medio-com-itinerarios-formativos-uma-analise-da-nossa-proposta-sob-a-perspectiva-da-formacao-integral/>. Acesso em: 17 jan. 2025.

31 SIGMOIDAL. Redes neurais multicamadas com python e keras. *Blog Sigmoidal*, 2021. Disponível em: <https://sigmoidal.ai/redes-neurais-python-keras-2/>. Acesso em: 28 jan. 2025.

32 MITCHELL, T. M. *Aprendizado de Máquina*. São Paulo: McGraw-Hill, 1997.

APÊNDICE A
Questionário sobre IA

Formulário sobre IA e Respostas dos Alunos

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.

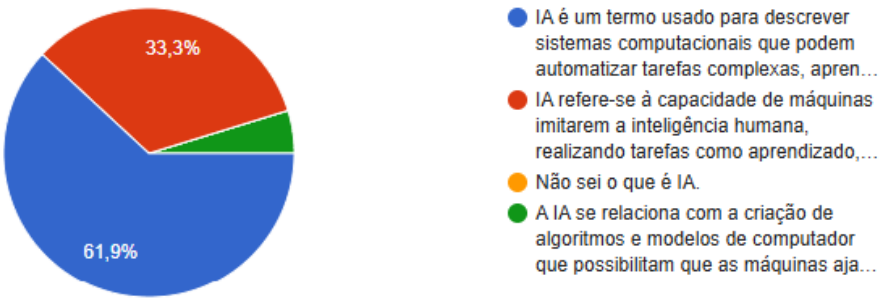
Por favor, responda honestamente cada uma das perguntas a seguir, escolhendo opções que condizem com o seu conhecimento neste momento.

Obrigada por participar!

Marque a(s) opção(ões) correta(s):

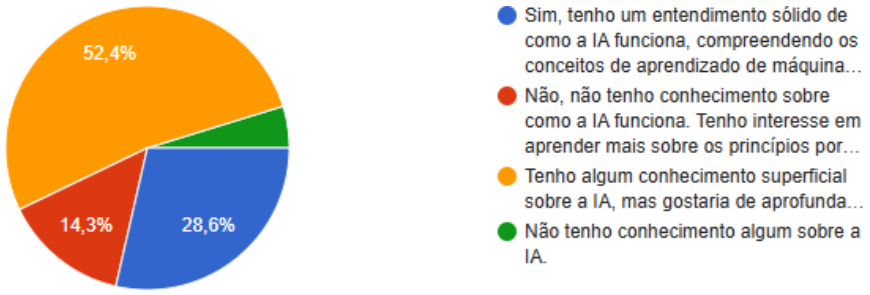
1. **O que você entende por inteligência artificial (IA)?**

   - 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.

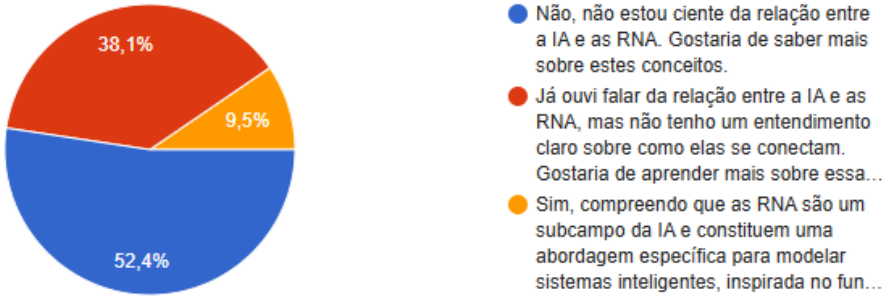


3. **Você já ouviu falar das redes neurais artificiais (RNA)?**

   - 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?**

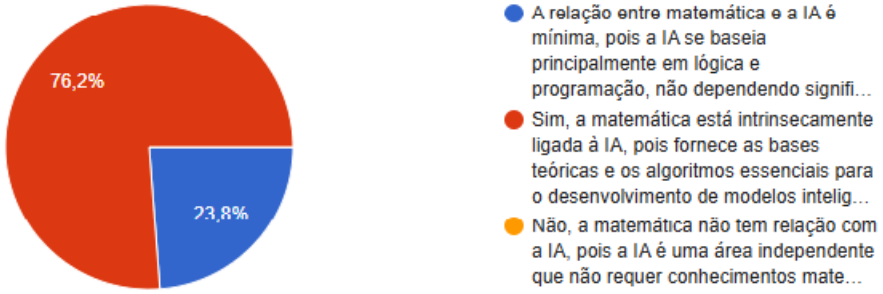
   - Não, não estou ciente da relação entre a IA e as RNA. Gostaria de saber mais sobre estes conceitos.
   - Já ouvi falar da relação entre a IA e as RNA, mas não tenho um entendimento claro sobre como elas se conectam. Gostaria de aprender mais sobre essa associação específica.
   - Sim, compreendo que 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?**

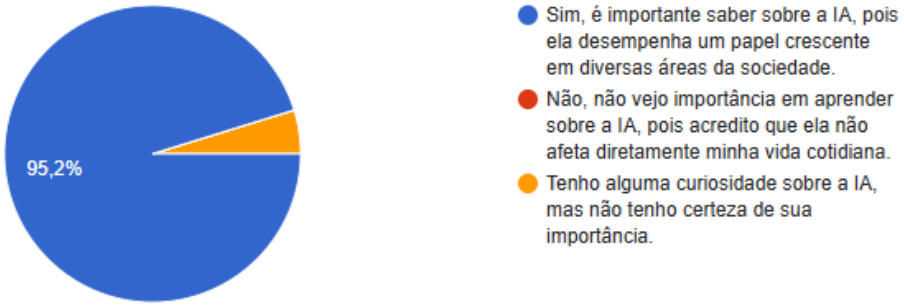
   - A relação entre matemática e a IA é mínima, pois a IA se baseia principalmente em lógica e programação, não dependendo significativamente de conceitos matemáticos.
   - Sim, a matemática está intrinsecamente ligada à IA, pois fornece as bases teóricas e os algoritmos essenciais para o desenvolvimento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



H

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 B
Tarefa- Aula 3

Exercício:

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, a soma total de pontos avaliativos para o terceiro bimestre será de 10 pontos.

Data de entrega: 4/9/2024

Dadas as funções de ativação:

1. **Sigmoid**: $\sigma(x) = \dfrac{1}{1+e^{-x}}$

2. **ReLU (Rectified Linear Unit)**: $\text{ReLU}(x) = \max(0, x)$

3. **Degrau**: $\text{degrau}(x) = \begin{cases} 0 & \text{se } x < 0 \\ 1 & \text{se } x \geq 0 \end{cases}$

Para cada um dos valores fornecidos, calcule, no seu caderno de registros, o resultado das funções de ativação sigmoid, ReLU e degrau: (use uma calculadora, se necessário)

- **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:** $x = -1.5, -1, -0.5, 0, 0.5$
- **Grupo 5:** $x = -1, -0.5, 0, 0.5, 1$

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

x	$\sigma(x)$	ReLU(x)	Degrau(x)

Exemplos de Cálculos:

1. **Para** $x = -2$:

$$\sigma(-2) = \frac{1}{1 + e^{-(-2)}} = \frac{1}{1 + e^{2}} \approx 0.119$$

$$\text{ReLU}(-2) = \max(0, -2) = 0$$

$$\text{degrau}(-2) = 0$$

2. **Para** $x = 1$:

$$\sigma(1) = \frac{1}{1 + e^{-1}} \approx 0.731$$

$$\text{ReLU}(1) = \max(0, 1) = 1$$

$$\text{degrau}(1) = 1$$

APÊNDICE C
Tarefa- Aula 5

Exercícios Desafio sobre a Aplicação Manual e Computacional da RNA Perceptron

Data da entrega: 30/10/2024

- **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 C.1: Tabela OR $(\vee)$

P	Q	$P \vee Q$
V	V	V
V	F	V
F	V	V
F	F	F

Fonte: Elaborada pela autora.

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 algorítimo matemático desenvolvido em sala de aula para o conectivo $(P \wedge Q)$. Depois, proponha um modelo matemático final, usando as especificações de entradas dadas.

A tarefa específica de cada grupo é:

1. **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.
2. **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.

3. **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.
4. **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.
5. **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!

Tabela de Atividades por Grupo

Tabela C.2: Tabela 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 D
Tarefa- Aula 7

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

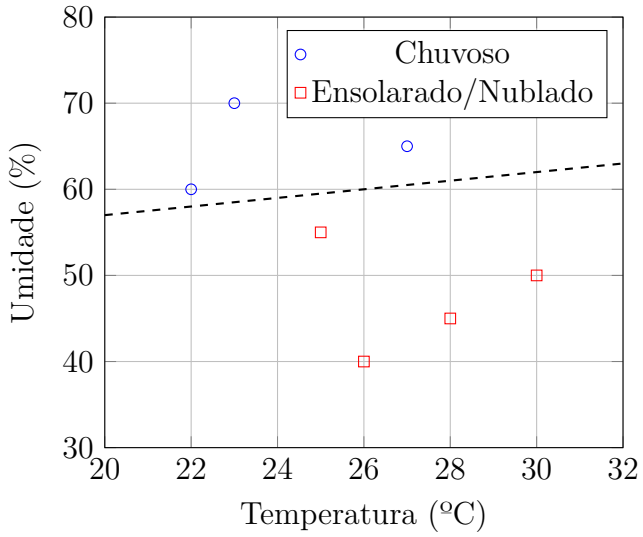
Tabela D.1: Temperatura e umidade de CXC por uma semana

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.

Mostramos, através da Figura D.1, que estes dados são linearmente separáveis:

Figura D.1: 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 E
Códigos para programação das funções de ativação

Função

1. 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}')
```

2. 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, trocamos o comando resultado: $.2f$ para $.4f$ e o executamos novamente, observando o resultado apresentado: agora, valores com quatro casas decimais.

3. 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}')
```

APÊNDICE F
Construção de uma Rede Perceptron para o Conectivo Lógico AND

Introdução

Este anexo apresenta a construção de uma rede Perceptron simples para resolver o problema do conectivo lógico AND. A seguir, será descrito o processo de inicialização, treinamento e ajuste de pesos e bias, utilizando uma taxa de aprendizagem fixa.

Usando a codificação $V = 0$ e $F = 1$, a tabela verdade do AND é dada por:

Entrada 1 (x_1)	Entrada 2 (x_2)	Saída (d)
0	0	0
0	1	0
1	0	0
1	1	1

O Perceptron somente retorna verdadeiro (1) quando ambas as entradas são verdadeiras (1).

Inicialização dos Pesos e Bias

- Pesos: $w_1 = 0$ e $w_2 = 0$
- Bias: $b = -1$
- Taxa de Aprendizagem: $\alpha = 0.4$

Usamos a função de ativação degrau, que verifica se a soma ponderada das entradas é suficiente para ativar a saída.

Passo a Passo do Treinamento

Para cada combinação de entradas, o Perceptron calcula a saída e ajusta os pesos e bias caso haja erro.

Passo 1: Cálculo da Saída

Para cada entrada, calculamos a saída y do Perceptron usando a equação:

$$y = w_1 \cdot x_1 + w_2 \cdot x_2 + b$$

Passo 2: Atualização dos Pesos e Bias

Se houver erro, o ajuste dos pesos w_i e do bias b é feito pelas fórmulas:

$$w_i = w_i + \alpha \cdot (d - y) \cdot x_i$$

$$b = b + \alpha \cdot (d - y)$$

Onde:

- w_i são os pesos reajustados
- α é a taxa de aprendizagem
- d é a saída desejada
- y é a saída calculada
- b é o bias

Exemplo de Treinamento

Época 1

- Entrada (0, 0), Saída desejada: $d = 0$

$$y = 0 \cdot 0 + 0 \cdot 0 - 1 = -1 \quad \text{(saída calculada: } \hat{y} = 0)$$

Não há erro, então não há ajustes.

- Entrada (0, 1), Saída desejada: $d = 0$

$$y = 0 \cdot 0 + 0 \cdot 1 - 1 = -1 \quad \text{(saída calculada: } \hat{y} = 0)$$

Não há erro, então não há ajustes.

- Entrada (1, 0), Saída desejada: $d = 0$

$$y = 0 \cdot 1 + 0 \cdot 0 - 1 = -1 \quad \text{(saída calculada: } \hat{y} = 0)$$

Não há erro, então não há ajustes.

- Entrada (1, 1), Saída desejada: $d = 1$

$$y = 0 \cdot 1 + 0 \cdot 1 - 1 = -1 \quad \text{(saída calculada: } \hat{y} = 0)$$

Como houve erro ($E = 1 - 0 = 1$), ajustamos os pesos e bias:

$$w_1 = 0 + 0.4 \cdot (1 - 0) \cdot 1 = 0.4$$

$$w_2 = 0 + 0.4 \cdot (1 - 0) \cdot 1 = 0.4$$

$$b = -1 + 0.4 \cdot (1 - 0) = -0.6$$

Época 2

Repetimos o processo com os novos valores de $w_1 = w_2 = 0.4$ e $b = -0.6$:

- Entrada (0, 0): $y = 0.4 \cdot 0 + 0.4 \cdot 0 - 0.6 = -0.6 \quad (\hat{y} = 0)$
- Entrada (0, 1): $y = 0.4 \cdot 0 + 0.4 \cdot 1 - 0.6 = -0.2 \quad (\hat{y} = 0)$
- Entrada (1, 0): $y = 0.4 \cdot 1 + 0.4 \cdot 0 - 0.6 = -0.2 \quad (\hat{y} = 0)$
- Entrada (1, 1): $y = 0.4 \cdot 1 + 0.4 \cdot 1 - 0.6 = 0.2 \quad (\hat{y} = 1)$

Após a segunda época, o Perceptron convergiu.

Modelo Final

Após o treinamento, o Perceptron foi capaz de aprender o conectivo AND. Os pesos e bias finais são:

$$w_1 = 0.4, \quad w_2 = 0.4, \quad b = -0.6$$

O modelo final é dado pela equação:

$$y = x_1 \cdot 0.4 + x_2 \cdot 0.4 - 0.6$$

APÊNDICE G
Códigos da RNA Perceptron AND, Função Sigmoide

```
[breaklines=true]

# 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])
    # 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_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

`\annex`.

APÊNDICE H
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}")
```

Novamente, obtemos o resultado esperado:

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

`\annex`.

APÊNDICE I
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()
```

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.

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.