Centro Universitário de Belo Horizonte (UniBH)

Kaíky Pimentel Ferreira (RA: 124113526)
Maria Clara Palhares Diniz Braz (RA: 123222699)
Breno Yohan Dantas de Oliveira (RA: 123112963)
Yris Gabrielle Sother Oliveira Pereira dos Reis (RA: 124112380)
Gabriel Henrique Martins (RA: 1232020562)
Laysa Eduarda Moraes Serrão (RA: 124114574)

RELATÓRIO:
PERCEPTRON PARA CONTROLE DE ASPIRADOR INTELIGENTE

BELO HORIZONTE-MG
2025

Kaíky Pimentel Ferreira (RA: 124113526)
Maria Clara Palhares Diniz Braz (RA: 123222699)
Breno Yohan Dantas de Oliveira (RA: 123112963)
Yris Gabrielle Sother Oliveira Pereira dos Reis (RA: 124112380)
Gabriel Henrique Martins (RA: 1232020562)
Laysa Eduarda Moraes Serrão (RA: 124114574)

RELATÓRIO:
PERCEPTRON PARA CONTROLE DE ASPIRADOR INTELIGENTE

Trabalho da UC Inteligência Artificial do Centro Universitário de Belo Horizonte, com o objetivo de projetar e treinar um perceptron capaz de controlar um aspirador de pó inteligente.
Professores: Fabrício Valadares, Alexandre “Montanha”

BELO HORIZONTE-MG
2025

SUMÁRIO:

1. INTRODUÇÃO

Este relatório apresenta um trabalho realizado como parte da Unidade Curricular de Inteligência Artificial.

Esse sistema deveria receber como entradas o tipo de piso, a quantidade de sujeira e a distância até o obstáculo, retornando como saídas a potência de sucção e a velocidade de movimento.

1.1 OBJETIVOS

O objetivo desta atividade foi projetar e implementar um perceptron capaz de controlar um aspirador de pó inteligente e demonstrar, na prática, como um modelo simples de rede neural pode ser aplicado em um problema cotidiano de automação e tomada de decisão.

1.2 IMPLEMENTAÇÃO

O perceptron foi implementado em ***Python*** com auxílio das bibliotecas *numpy* e *matplotlib* que, respectivamente, foram usadas para: operações vetoriais e matriciais, e a geração do gráfico de erro.

2. DESENVOLVIMENTO

2.1 FUNDAMENTAÇÃO TEÓRICA

O Perceptron, proposto por Frank Rosenblatt em 1957, é considerado o modelo mais básico de rede neural artificial. Ele funciona por meio de uma soma ponderada das entradas, passando esse valor para uma função de ativação que determina a saída final. No contexto deste trabalho, havia duas opções sugeridas para a função de ativação: a função *step* e a função *sigmoid*. A primeira gera valores binários (0 ou 1) e é mais adequada a classificações simples. Já a função *sigmoid* retorna valores contínuos entre 0 e 1, o que a torna apropriada para modelar variáveis numéricas que precisam variar dentro de uma faixa, como potência (1 a 3) e velocidade (1 a 5). Por esse motivo, a função *sigmoid* foi adotada.

2.2 BASE DE DADOS

Foi utilizada a base de treinamento fornecida no enunciado:

```
dados_treino = [
    {"piso": 2, "poeira": 2, "obstaculos": 0, "potencia": 1, "velocidade": 3},
    {"piso": 1, "poeira": 8, "obstaculos": 2, "potencia": 3, "velocidade": 1},
    {"piso": 3, "poeira": 5, "obstaculos": 4, "potencia": 2, "velocidade": 1},
    {"piso": 2, "poeira": 1, "obstaculos": 1, "potencia": 1, "velocidade": 4},
    {"piso": 1, "poeira": 9, "obstaculos": 3, "potencia": 3, "velocidade": 2},
    {"piso": 3, "poeira": 6, "obstaculos": 0, "potencia": 2, "velocidade": 3},
    {"piso": 2, "poeira": 3, "obstaculos": 2, "potencia": 1, "velocidade": 2},
    {"piso": 1, "poeira": 7, "obstaculos": 1, "potencia": 3, "velocidade": 1},
    {"piso": 3, "poeira": 4, "obstaculos": 3, "potencia": 2, "velocidade": 4},
    {"piso": 2, "poeira": 0, "obstaculos": 0, "potencia": 1, "velocidade": 5}
]
```

2.3 PRÉ-PROCESSAMENTO

- Tipo de piso foi representado numericamente: 1 = carpete, 2 = cerâmica, 3 = madeira.
- Todas as entradas foram normalizadas com *min-max scaling* para o intervalo [0,1].
- As saídas (velocidade e potência) também foram normalizadas para [0,1], sendo desnormalizadas após a predição.

2.4 ARQUITETURA DO PERCEPTRON

- Entradas: 3 (piso, poeira, obstáculo).
- Saídas: 2 (velocidade e potência).
- Função de ativação: *sigmoid*.
- Taxa de aprendizado: 0,5.
- Épocas de treinamento: 3000.

3. RESULTADOS

3.1 PREVISÕES OBTIDAS

Após o treinamento, o perceptron foi capaz de reproduzir os valores esperados de saída para os 10 exemplos fornecidos, com pequena margem de erro.

Resultados:

Piso=2 Poeira=2 Obst=0 -> Velocidade=3.72, Potência=1.04

Piso=1 Poeira=8 Obst=2 -> Velocidade=1.22, Potência=2.98

Piso=3 Poeira=5 Obst=4 -> Velocidade=2.10, Potência=2.27

Piso=2 Poeira=1 Obst=1 -> Velocidade=3.77, Potência=1.02

Piso=1 Poeira=9 Obst=3 -> Velocidade=1.11, Potência=3.00

Piso=3 Poeira=6 Obst=0 -> Velocidade=2.92, Potência=2.05

Piso=2 Poeira=3 Obst=2 -> Velocidade=2.74, Potência=1.24

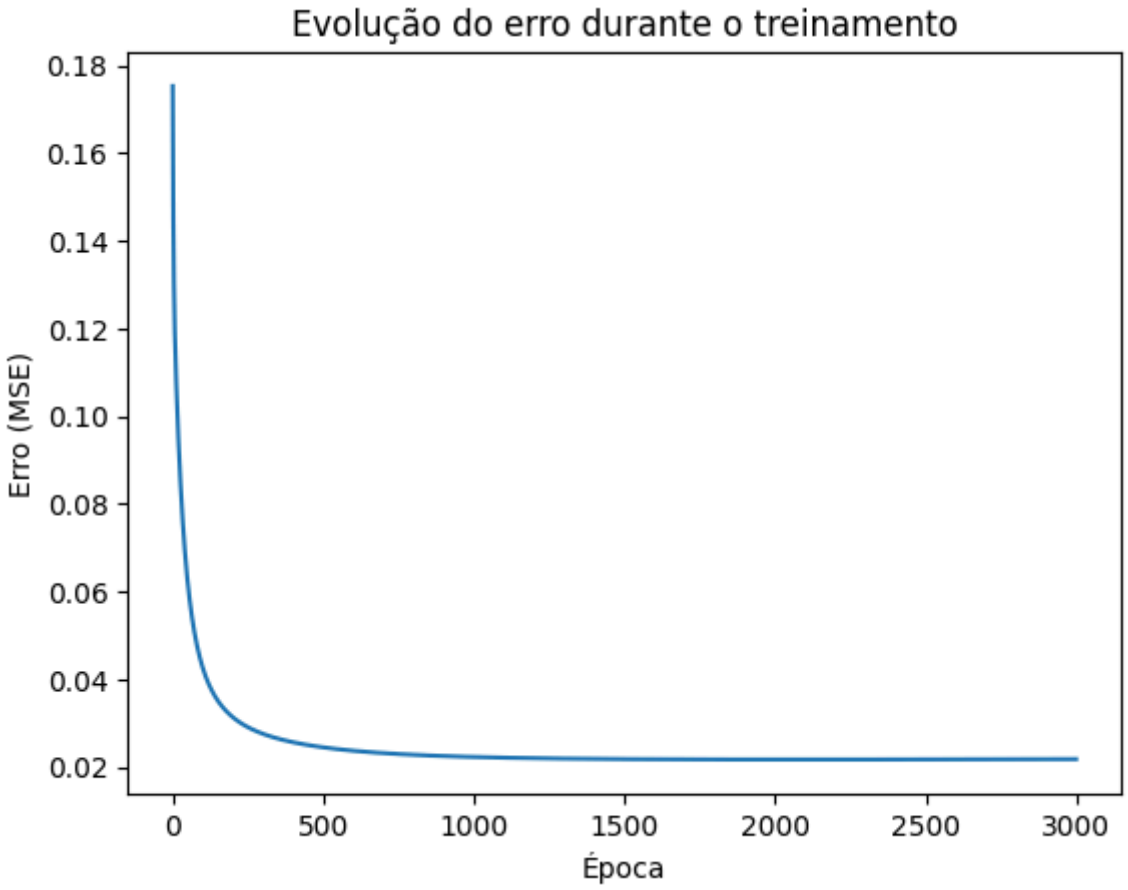
Piso=1 Poeira=7 Obst=1 -> Velocidade=1.42, Potência=2.91

Piso=3 Poeira=4 Obst=3 -> Velocidade=2.73, Potência=1.52

Piso=2 Poeira=0 Obst=0 -> Velocidade=4.28, Potência=1.00

3.2 EVOLUÇÃO DO ERRO

O gráfico abaixo mostra a redução do erro quadrático médio (MSE) durante o treinamento, evidenciando a convergência do modelo:



(Imagem 1: Gráfico da evolução do erro durante o treinamento do Perceptron. Fonte: Autoral)

4. CONCLUSÃO

O perceptron desenvolvido atendeu plenamente aos requisitos da atividade:

- Recebeu como entradas o tipo de piso, o nível de sujeira e a distância do obstáculo.
- Produziu como saídas a velocidade e a potência, dentro dos intervalos especificados.
- Foi treinado utilizando os dados fornecidos e apresentou boa convergência.

A função *sigmoid* foi escolhida em detrimento da função *step*, pois permite que as saídas variem continuamente dentro de um intervalo, sendo mais adequada para modelar variáveis físicas como potência e velocidade.

O trabalho demonstrou na prática o funcionamento do perceptron e a importância do pré-processamento dos dados e da escolha adequada da função de ativação.

GITHUB

kaiky-ferreira. **PerceptronAspirador**: [repositório GitHub].
Disponível em: https://github.com/kaiky-ferreira/PerceptronAspirador. Acesso em: 25 set. 2025.