Eduardo Oliveira Coelho

Código disponível em: https://github.com/eduardoocoelho/ai-lists

Questão 1 -

- **Testes e resultados:**
  - **Porta AND:**
    - Nº Entradas: 4
    - Entradas: 1 1 1 1
    - Resultado Esperado: 1

```
Número de entradas: 4
Tipo de operação (AND, OR, XOR): AND
Insira a entrada 1 (0 ou 1): 1
Insira a entrada 2 (0 ou 1): 1
Insira a entrada 3 (0 ou 1): 1
Insira a entrada 4 (0 ou 1): 1
Resultado da operação AND para as entradas [1 1 1 1]: 1
```

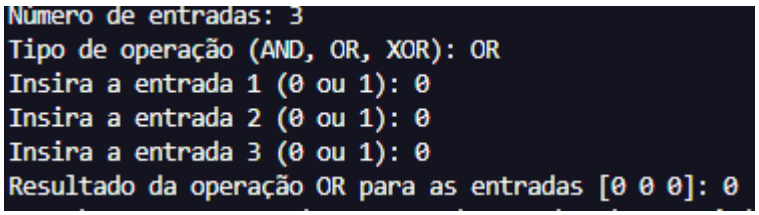
    - Nº Entradas: 6
    - Entradas: 0 0 0 0 1 0
    - Resultado Esperado: 0

```
Número de entradas: 6
Tipo de operação (AND, OR, XOR): AND
Insira a entrada 1 (0 ou 1): 0
Insira a entrada 2 (0 ou 1): 0
Insira a entrada 3 (0 ou 1): 0
Insira a entrada 4 (0 ou 1): 0
Insira a entrada 5 (0 ou 1): 1
Insira a entrada 6 (0 ou 1): 0
Resultado da operação AND para as entradas [0 0 0 0 1 0]: 0
```

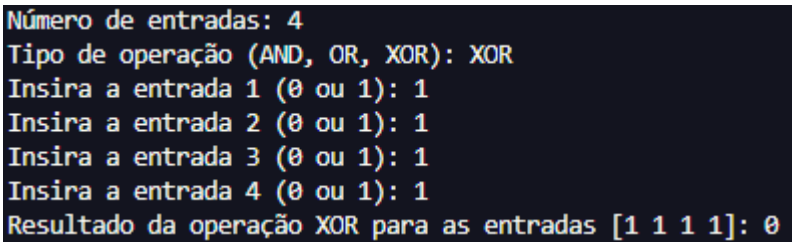
  - **Porta OR:**
    - Nº Entradas: 9
    - Entradas: 0 0 0 0 0 0 1 0 0
    - Resultado Esperado: 1

```
Número de entradas: 9
Tipo de operação (AND, OR, XOR): OR
Insira a entrada 1 (0 ou 1): 0
Insira a entrada 2 (0 ou 1): 0
Insira a entrada 3 (0 ou 1): 0
Insira a entrada 4 (0 ou 1): 0
Insira a entrada 5 (0 ou 1): 0
Insira a entrada 6 (0 ou 1): 0
Insira a entrada 7 (0 ou 1): 1
Insira a entrada 8 (0 ou 1): 0
Insira a entrada 9 (0 ou 1): 0
Resultado da operação OR para as entradas [0 0 0 0 0 0 1 0 0]: 1
```

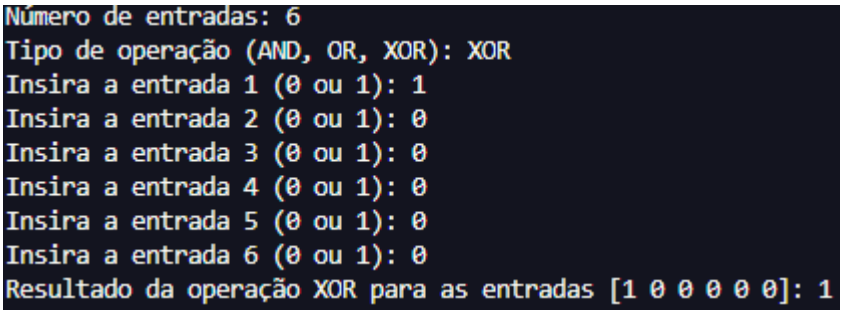
- Nº Entradas: 3
- Entradas: 0 0 0
- Resultado Esperado: 0



- **Porta XOR:**
  - Nº Entradas: 4
  - Entradas: 1 1 1 1 1
  - Resultado Esperado: 0



  - Nº Entradas: 6
  - Entradas:1 0 0 0 0 0
  - Resultado Esperado: 1



- **Explicações da Implementação:**
  - **Definição da função de ativação sigmoidal:**
    - Função de ativação usada para introduzir não linearidades na rede neural. É utilizada para transformar as saídas das camadas ocultas e de saída para um intervalo entre 0 e 1.
  - **Definição da função de previsão**
    - Realiza a propagação para frente (feedforward) da rede neural. Ela calcula as saídas da camada oculta (A1) e da camada de saída (A2) usando as entradas (X) e os pesos (W1 e W2).
  - **Definição da função de perda - Binary Cross Entropy**
    - Calcula a entropia cruzada binária entre as previsões (y_pred) e os rótulos reais (y_true). Essa função de perda é utilizada para

avaliar o quão bem a rede está performando durante o treinamento.

- **Definição da função de treinamento:**
  - Treinamento da rede neural usando o algoritmo de Backpropagation. Ela inicializa os pesos aleatoriamente, realiza a propagação para frente, calcula o erro, realiza a retropropagação para calcular os gradientes e, finalmente, atualiza os pesos.
- **Função main:**
  - Guia a interação com o usuário, cria o conjunto de treinamento com base na operação escolhida, treina a rede neural e testa a rede neural com entradas fornecidas pelo usuário. O código permite que o usuário insira valores para as entradas e exibe o resultado da operação escolhida pela rede neural.

- **Importância da Taxa de Aprendizado:**
  - Hiperparâmetro crucial em algoritmos de otimização utilizados no treinamento de redes neurais. Ela determina a magnitude dos ajustes feitos nos pesos durante a atualização. Uma taxa de aprendizado muito alta pode resultar em oscilações e até mesmo em divergência, enquanto uma taxa de aprendizado muito baixa pode levar a convergência lenta ou ficar presa em mínimos locais.
- **Importância do Bias:**
  - Constante adicionada aos inputs de cada camada de uma rede neural. Ele permite que a rede faça ajustes na saída mesmo quando todos os inputs são zero. Isso é crucial para garantir que a rede neural possa modelar corretamente padrões mais complexos e não fique restrita apenas a funções lineares.
- **Função de Ativação:**
  - Crucial para introduzir não linearidades nos modelos, permitindo a aprendizagem de representações hierárquicas complexas dos dados. Ela facilita o mapeamento não linear de inputs para outputs, possibilitando a modelagem eficiente de relações complexas e a realização de tarefas como classificação e reconhecimento de padrões.
  - **Sigmoid:**
    - Função não linear que mapeia qualquer valor real para o intervalo entre 0 e 1. Essa função é usada para introduzir não linearidades nas camadas ocultas da rede neural. A saída da sigmoid está próxima de 0 para entradas muito negativas, próxima de 1 para entradas muito positivas e se aproxima de 0.5 para entradas próximas a zero. Pode apresentar problemas como o "vanishing gradient" em redes mais profundas.

  - **Tangente - Tanh:**

- Mapeia valores reais para o intervalo entre -1 e 1. Similar à sigmoid, ela introduz não linearidades nas camadas ocultas, mas tem a vantagem de ter uma saída que varia de -1 a 1, o que pode facilitar o treinamento de modelos, especialmente em problemas de classificação onde as classes são balanceadas em torno de zero.

Questão 2 -

O uso de funções de ativação não lineares no algoritmo Backpropagation é crucial para o funcionamento eficaz das redes neurais artificiais. Essas funções permitem que as redes aprendam funções complexas dos dados e construam representações hierárquicas, essenciais para lidar com relacionamentos não lineares e modelar tarefas complexas.

As redes neurais combinam características complexas e abstratas graças às funções de ativação não lineares. Essas funções são essenciais para o treinamento de retropropagação e permitem o ajuste dos pesos da rede. Sem essas funções, as redes neurais estariam restritas a representar apenas transformações lineares dos dados.

As funções de ativação não lineares no algoritmo Backpropagation são essenciais para que as redes neurais artificiais aprendam e representem funções complexas, construam representações hierárquicas dos dados e se assemelhem ao funcionamento dos sistemas biológicos, tornando-as poderosas para resolver tarefas de aprendizado de máquina complexas e diversas.

Questão 3 -

- **Sigmoide (Função Logística):**
  - **Equação:** $\sigma(x) = \frac{1}{1 + e^{-x}}$
  - Transforma a entrada em um intervalo entre 0 e 1, o que é útil para problemas de classificação binária. Ela possui uma forma suave e é diferenciável em todo o seu domínio.
  - **Importância:** é essencial em problemas de classificação binária, mapeando a entrada para o intervalo de 0 a 1. Ela é diferenciável e permite o ajuste suave dos pesos durante o treinamento da rede. Porém, pode causar "desvanecimento de gradiente" em entradas muito positivas ou negativas, o que pode tornar o treinamento de redes profundas mais lento.

- **Tangente Hiperbólica (tanh):**
    - **Equação:** $\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$
    - Transforma a entrada em um intervalo entre -1 e 1. Ele é útil em redes neurais quando se deseja ter saídas que variam de forma simétrica em relação a zero.
    - **Importância:** é uma função semelhante à sigmoide, mas mapeia a entrada para o intervalo de -1 a 1. É útil quando se deseja saídas centradas em zero e é diferenciável para backpropagation. No entanto, pode sofrer de desvanecimento de gradiente para entradas extremas.
- **Unidade Linear Retificada (ReLU):**
    - **Equação:** $f(x) = \max(0, x)$
    - A função ReLU é não linear, mas possui uma forma mais simples. Ela retorna zero para entradas negativas e a própria entrada para entradas positivas. O ReLU é amplamente utilizado em redes neurais profundas devido à sua simplicidade e à sua capacidade de mitigar o problema de desvanecimento de gradiente.
    - **Importância:** é amplamente utilizada em redes neurais devido à sua simplicidade e eficácia. Ela ajuda a mitigar o problema de desvanecimento de gradiente e é especialmente útil para treinar redes profundas. No entanto, não é diferenciável em zero, o que é tratado com aproximações em algoritmos de treinamento.