Inteligência Artificial

Perceptron

Professor Alexandre “Montanha” de Oliveira

Professor Alexandre “Montanha”

- Mestre em Educação e Gestão Social
- Especialista em Educação
- Especialista em Gamification
- Analista de Sistemas
- Historiador
- Gamer

http://lattes.cnpq.br/7137425845019062
https://www.linkedin.com/in/alexandre-oliveira-montanha/
alexmontanha@hotmail.com

Sobre mim

Professor Alexandre Montanha

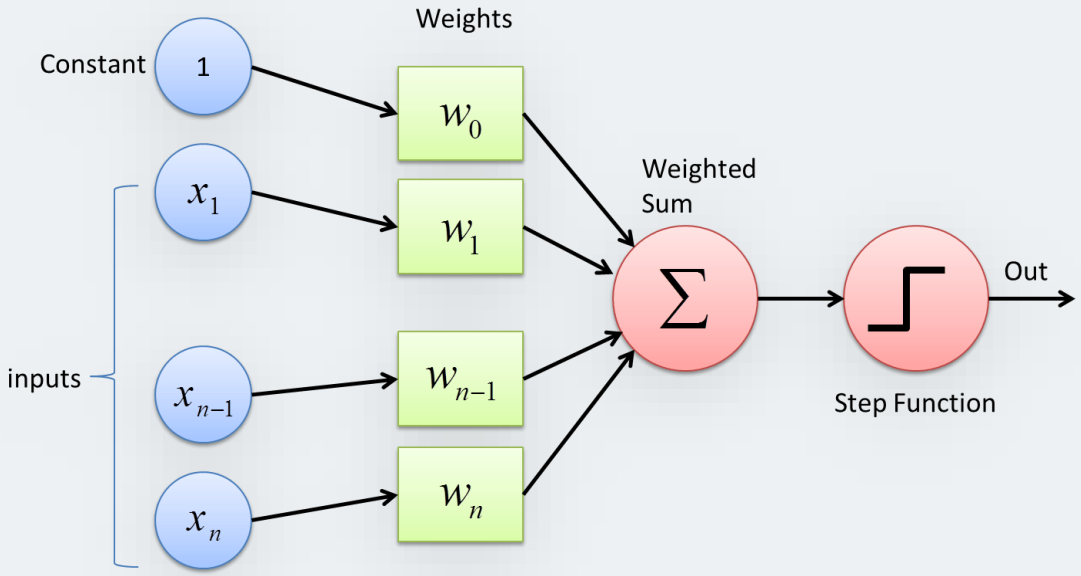

```rust
fn main() {
    let unidade_curricular = "Programação de Soluções Computacionais";
    let professor = "Alexandre 'Montanha' de Oliveira";
    let tempo_analista = 35;
    let tempo_professor = 15;
    let principais_empresas = ["CEMIG", "Belgo Mineira", "TOTVS", "Grupo Ânima", "Mobilus"];
    let principais_tecnologias = ["Assembly", "Clipper", "C++", "Pascal", "Delphi", "C#", "Dart", "Rust"];
    let principais_linguas = ["Português", "Inglês", "Grego"];
    let formacao = ["Tec. em Processamento de Dados", "História", "Mestrado em Educação e Gestão Social"];

    println!("Unidade Curricular: {}", unidade_curricular);
    println!("Professor: {}", professor);
    println!("Tempo como Analista de Sistemas: {} anos", tempo_analista);
    println!("Tempo como Professor: {} anos", tempo_professor);
    println!("Principais Empresas: {:?}", principais_empresas);
    println!("Principais Tecnologias: {:?}", principais_tecnologias);
    println!("Principais Linguas: {:?}", principais_linguas);
    println!("Formação: {:?}", formacao);
}
```

Introdução

Diversos modelos de neurônios artificiais foram propostos para simular o aprendizado humano, como o **perceptron**, que, apesar de possuir mudanças sutis em comparação ao modelo **McCulloch e Pitts (MP)**, revolucionou a inteligência artificial e o aprendizado de máquina.

Classicamente, o perceptron foi utilizado como um **classificador linear**.

No entanto, a classificação depende de uma boa representação dos dados e do treinamento a partir de **funções de custo**, que permitem à rede aprender e se adaptar para obter melhores resultados.



Modelo de neurônio — perceptron

O perceptron tem como origem o modelo proposto por McCulloch e Pitts (MP), primeiro neurônio artificial, que pode ser dividido em quatro partes: entradas, conexões, corpo da célula e saída (NORVIG; RUSSELL, 2013).

Modelo de neurônio — MP

O neurônio MP pode conter inúmeras entradas, e cada uma delas está associada a uma conexão com um peso (weight), que excita e inibe o valor da entrada a que se vincula.

Se a entrada for sempre binária (0 ou 1), o peso será positivo, nulo ou negativo (1, 0 ou –1). Apesar de os parâmetros possuírem estados parecidos, o valor do estado inibidor é diferente para a entrada e o peso.

Caso a entrada seja 0, de forma a não indicar a presença de alguma informação, o peso inibidor terá um valor de –1, significando que, quando existir uma entrada ativa por meio dele, o resultado será a diminuição da excitação, e não apenas uma não influência.

Por fim, um peso de 0 tem o mesmo efeito de se eliminar a conexão (NORVIG; RUSSELL, 2013).

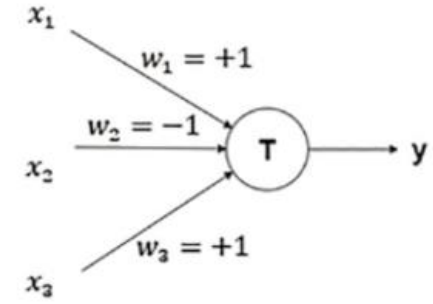


Figura 1. Modelo de neurônio MP.

Modelo de neurônio — MP

Já a saída, assim como a entrada, somente pode ser 0 ou 1, mas, para isso, no corpo da célula, deve-se realizar a soma de todas as entradas ponderadas por seus respectivos pesos.

Caso essa somatória seja maior que um valor de limiar (threshold), a saída será igual a 1, do contrário, ela permanecerá em 0.

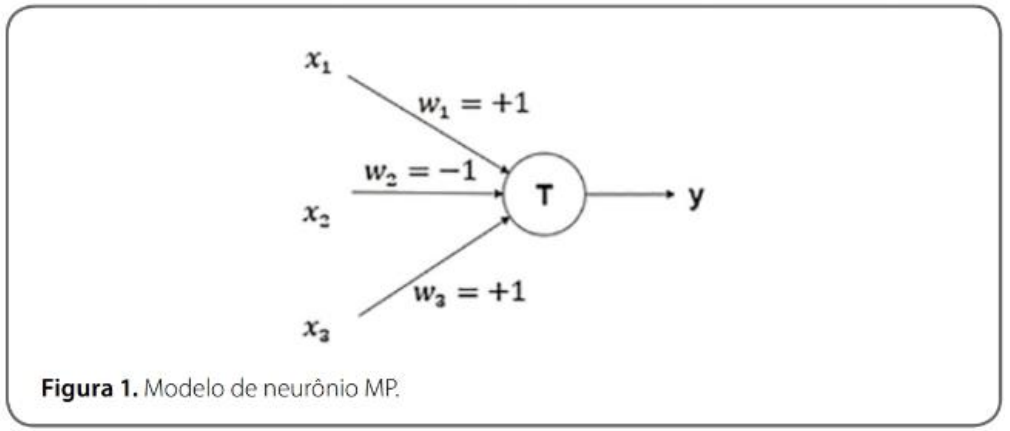


Figura 1. Modelo de neurônio MP.

Modelo de neurônio — MP

ATENÇÃO!

A grande limitação desse tipo de neurônio é a restrição aos valores de peso para servirem apenas como excitadores (1) ou inibidores (–1), restringindo a resolução de problemas, pois a relação entre várias entradas pode ser muito mais complexa do que isso.

Modelo de neurônio — MP

Modelo McCulloch e Pitts (MP)

Equação de Entrada:

$$in_j = \sum_i^n w_{i,j} \times x_i$$

Equação de Saída:

$$y = f(x) = \begin{cases} 0, & x \leq 0 \\ 1, & x > 0 \end{cases}$$

Modelo de neurônio — MP

Modelo McCulloch e Pitts

Equação de Entrada:

Equação de Saída:

```python
def perceptron_input(inputs, weights):
    """
    Calculate the perceptron input.

    :param inputs: List of input values
    :param weights: List of weights corresponding to the inputs
    :return: The result of the weighted sum of inputs plus the bias
    """
    weighted_sum = sum(i * w for i, w in zip(inputs, weights))
    return weighted_sum
```

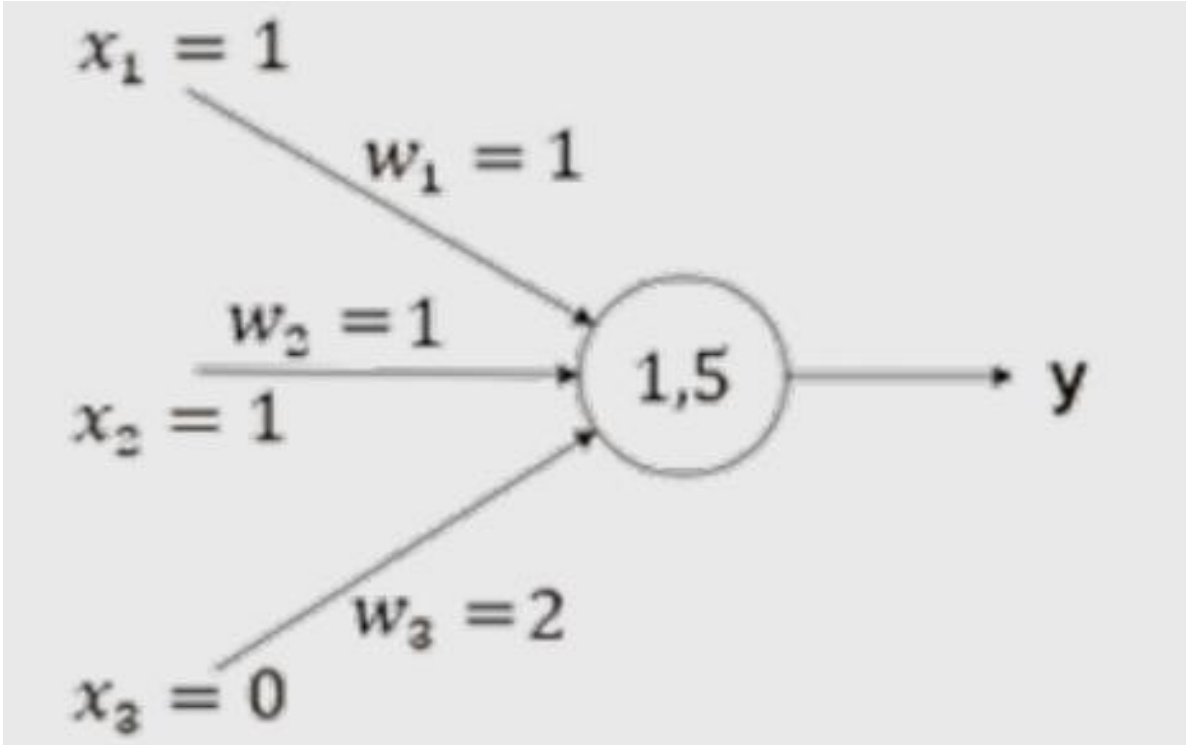
```python
def perceptron_output(inputs, weights):
    """
    Calculate the perceptron output.

    :param inputs: List of input values
    :param weights: List of weights corresponding to the inputs
    :return: The output of the perceptron
    """
    return 1 if perceptron_input(inputs, weights) >= 0 else 0
```

Exercício 1

Considere o neurônio a seguir, com seus valores de entrada (x), seu peso (w) e limiar de ativação (threshold = 1,5).

Qual será a saída, para os seguintes valores e pesos:

x = [1, 1, 2] e w = [1, 1, 0]



Modelo de neurônio — Perceptron

A partir do modelo MP e de estudos desenvolvidos pelo psicólogo Donald

Hebb, que ficaram conhecidos, posteriormente, como aprendizado de Hebb, o psicólogo Frank Rosenblatt propôs um modelo de neurônio chamado de perceptron.

Hebb afirmava que o aprendizado humano ocorria mediante o reforço das conexões entre neurônios sempre que eles eram ativados em conjunto, assim, sempre que um neurônio A era ativado ao mesmo tempo que o B, a ligação entre eles se reforçava.

É o equivalente a dizer que o peso dado a essa conexão foi ampliado, porém, quando qualquer um desses neurônios se ativava sem que o outro o fosse, a conexão era suprimida, reduzindo seu peso.

Modelo de neurônio — Perceptron

Percebendo o impacto do aprendizado de Hebb, Rosenblatt criou o modelo do perceptron mantendo boa parte da estrutura do MP, mas com uma novidade: a possibilidade de se utilizar pesos e limiares de valores fracionários (threshold).

Essa mudança aumentou, significativamente, a complexidade da possível relação entre duas ou mais entradas.

Modelo de neurônio — Perceptron

Em ambos os casos, seja para o modelo MP ou o perceptron, o somatório das entradas ponderadas deve ser superior a um limiar e, portanto, objeto de outra função antes que possam resultar no valor real de saída.

Essa função que avalia a soma é denominada de ativação. Para a equação original de saída, essa soma ponderada pode ser subtraída do limiar (equação 3) e, assim, a função de ativação precisa validar a saída sempre que o resultado total for maior que 0 (equação 4)

Equação 3 $in_j - T \geq 0$

Equação 4 $y = g(in_j - T)$

Modelo de neurônio — Perceptron

Sendo o viés (bias – b) o oposto do limiar (threshold – T), conforme

mostra a equação 5, pode-se utilizá-lo como uma entrada de valor sempre unitário, cujo peso corresponde ao oposto do limiar.

Assim, a saída é expressa como o resultado da função de ativação para as somas ponderadas das entradas em conjunto com o viés (equação 6 ou, de forma expandida, equação 7).

Equação 5 $b = -T$

Equação 6 $y = g(in_j + b)$

Equação 7 $y = g(b + \sum_{i}^{n} w_{i,j} \times x_i)$

Modelo de neurônio — Perceptron

A função de ativação do modelo perceptron foi mantida da mesma forma que o modelo MP, selecionando uma saída 1 ou 0 a partir de um limiar a ser superado pela soma ponderada das entradas.

Contudo, observe que o limiar (T) é substituído por um viés (bias – b), que terá a mesma intensidade dele, em sentido oposto (equação 5).

Modelo de neurônio — Perceptron

O uso do viés se trata de uma entrada de contribuição negativa à soma ponderada do neurônio ou, ainda, uma entrada que sempre terá um peso inibidor ($w = -1$).

Assim, a função de ativação rígida (saída 0 ou 1) torna a saída ativa para qualquer soma maior que 0 e inativa para as menores ou iguais a 0.

No entanto, essa informação é abstraída, pois a função para outros modelos de rede podem utilizar regras diferentes, o que torna a equação 6 abrangente o suficiente para representar redes que usem uma função logística de ativação, que atribui apenas 0 ou 1 para uma saída, mas também para outras funções mais complexas.

A equação 7, por sua vez, é apenas uma forma expandida e mais intuitiva de representar a mesma fórmula da 6.

Exercício 2

Dada a fórmula do Perceptron abaixo, codifique a mesma em Python, versione no GitHub e post o link no Ulife.

Adicione exemplos, utilizando o código.

$$f(x_n, w_n, b) = b + \sum_{i}^{n} w_{i,j} \times x_i$$

Referência Bibliográfica

SILVA, Fabrício M.; LENZ, Maikon L.; FREITAS, Pedro H C.; et al. **Inteligência artificial**. Porto Alegre: Grupo A, 2018. E-book. ISBN 9788595029392. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788595029392/. Acesso em: 08 set. 2024.

Thanks!

Professor Alexandre Montanha


slidesgo