Uma introdução a redes neurais

Luís Fernandes Saucedo Souza[1]
Bárbara Denicol do Amaral Rodriguez [1]
Cristiana Andrade Poffal[1]

[1]Universidade Federal do Rio Grande – Instituto de Matemática, Estatística e Física

Sessões

Sessões

Por que redes neurais?

O cérebro é um computador (sistema de processamento de informação) altamente complexo, não-linear e paralelo. Ele tem a capacidade de organizar seus constituintes estruturais conhecidos por neurônios de forma a realizar certos processamentos (por exemplo, reconhecimento de padrões, percepção e controle motor) muito mais rapidamente que o mais rápido computador digital hoje existente (Haykin, 2007).

Por que redes neurais?

Um neurônio em "desenvolvimento" é sinônimo de um cérebro plástico: a *plasticidade* permite que o sistema nervoso em desenvolvimento se adapte ao seu meio ambiente. Assim como a plasticidade parece ser essencial para o funcionamento dos neurônios como unidades de processamento de informação do cérebro humano, também ela o é com relação às redes neurais artificiais (Haykin, 2007).

Por que redes neurais?

Uma rede neural é um processador maciçamente paralelamente distribuído constituído de unidades de processamento simples, que têm a propensão natural para armazenar conhecimento experimental e torná-lo disponível para o uso. Ela se assemelha ao cérebro em dois aspectos:

- O conhecimento é adquirido pela rede a partir de seu ambiente por meio de um processo de aprendizagem.
- Forças de conexão entre neurônios, conhecidas como pesos sinápticos, são utilizados para armazenar o conhecimento adquirido.

Benefícios das redes neurais

É evidente que uma rede neural extrai seu poder computacional através, primeiro, de sua estrutura maciçamente paralelamente ,distribuída e segundo de sua habilidade de aprender e portanto de generalizar. A generalização se refere ao fato de a rede neural produzir saídas adequadas para entradas que não estavam presentes durante o treinamento (aprendizagem). Estas duas capacidades de processamento de informação tornam possível para as redes neurais resolver problemas complexos (de grande escala) que são atualmente intratáveis. O uso de redes neurais oferece as seguintes propriedades úteis e capacidades:

Benefícios das redes neurais

- **Generalização**: Produzir saídas adequadas para entradas que não estavam presentes durante o treinamento.
- **Paralelização**: Um problema complexo é decomposto em um conjunto de tarefas relativamente simples.
- **Não-linearidade**: Uma propriedade importante caso o mecanismo responsável pela geração do sinal de entrada seja inerentemente não-linear.

Benefícios das redes neurais

- **Analogia neurobiológica**: O projeto de uma rede neural é motivado pela analogia com o cérebro, que é uma prova viva de que o processamento paralelo tolerante a falhas é não somente possível fisicamente, mas também rápido e poderoso. O sistema nervoso humano pode ser visto como um sistema de três estágios, como mostrado no diagrama em blocos

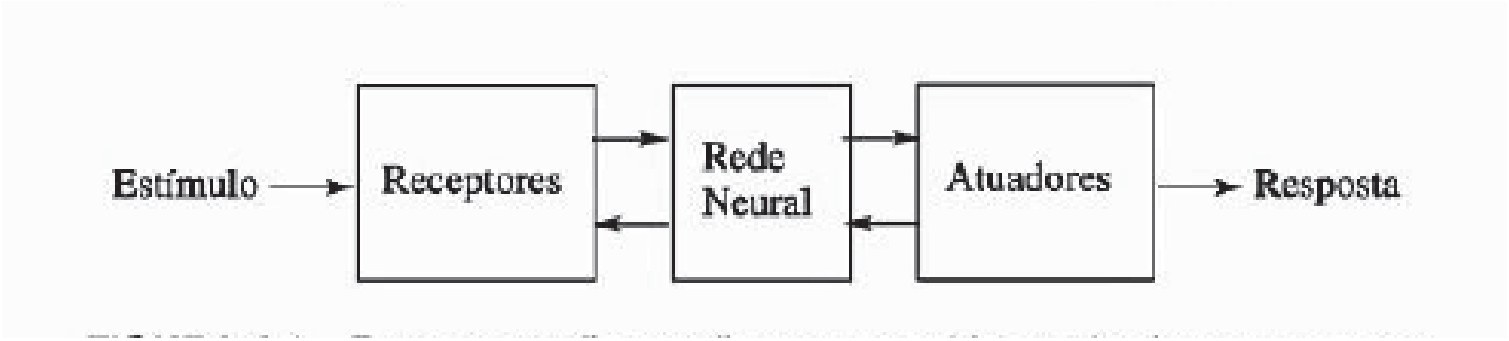

Figura: Representação em diagrama em blocos do sistema nervoso.

Fonte: Adaptado de Haykin (2007).

Sessões

Neurônio

Em termos matemáticos, pode-se descrever um neurônio k escrevendo o seguinte par de equações:

$$u_k = \sum_{i=1}^{j} w_{ki} x_i, \tag{1}$$

e

$$y_k = \varphi(u_k + b_k), \tag{2}$$

onde x_1, x_2, ..., x_j são os sinais de entrada; w_{k1}, w_{k2}, ..., w_{kj} são os pesos sinápticos do neurônio k; u_k é a saída do combinador linear devido aos sinais de entrada; b_k é o bias; $\varphi(\cdot)$ é a função de ativação; e y_k é o sinal de saída do neurônio.

Perceptron

O comportamento inteligente de uma Rede Neural Artificial (RNA) vem das interações entre as unidades de processamento da rede (Carvalho, 2020). O diagrama em blocos a seguir mostra o modelo de um neurônio, que forma a base para o projeto de redes neurais (artificiais).

Perceptron

O modelo é constituído por um conjunto de entradas (sinapses), cada uma caracterizada por um *peso* ou força própria.

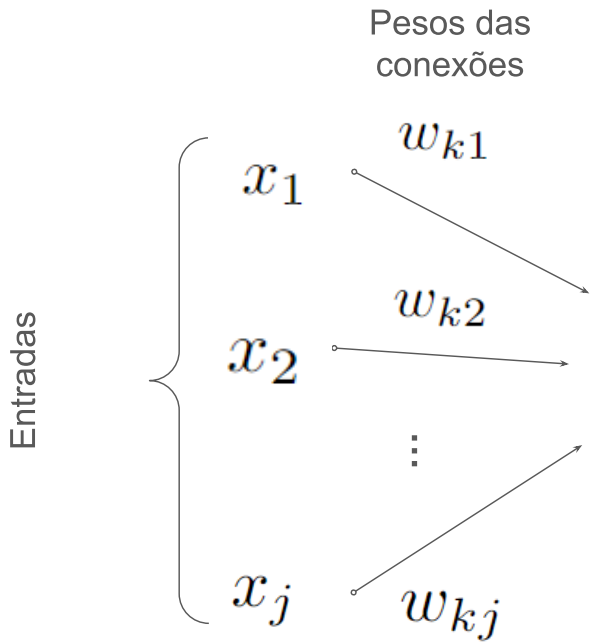

Figura: Modelo não-linear de um neurônio.

Fonte: Adaptado de Haykin (2007).

Perceptron

Um somador para somar os sinais de entrada, ponderados pelas respectivas sinapses do neurônio; as operações descritas aqui constituem um combinador linear.

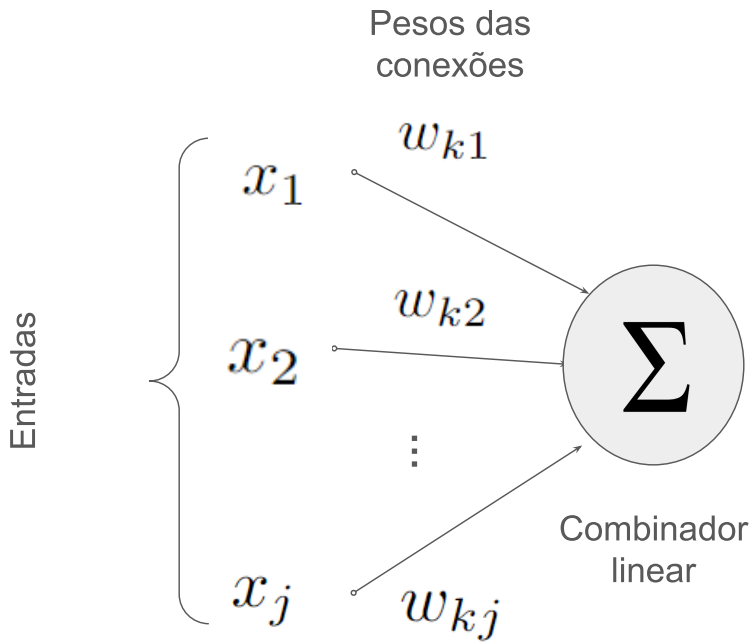

Figura: Modelo não-linear de um neurônio.

Fonte: Adaptado de Haykin (2007).

Perceptron

Um bias b_k que tem o efeito de aumentar ou diminuir a entrada líquida da função de ativação, dependendo se ele é positivo ou negativo, respectivamente.

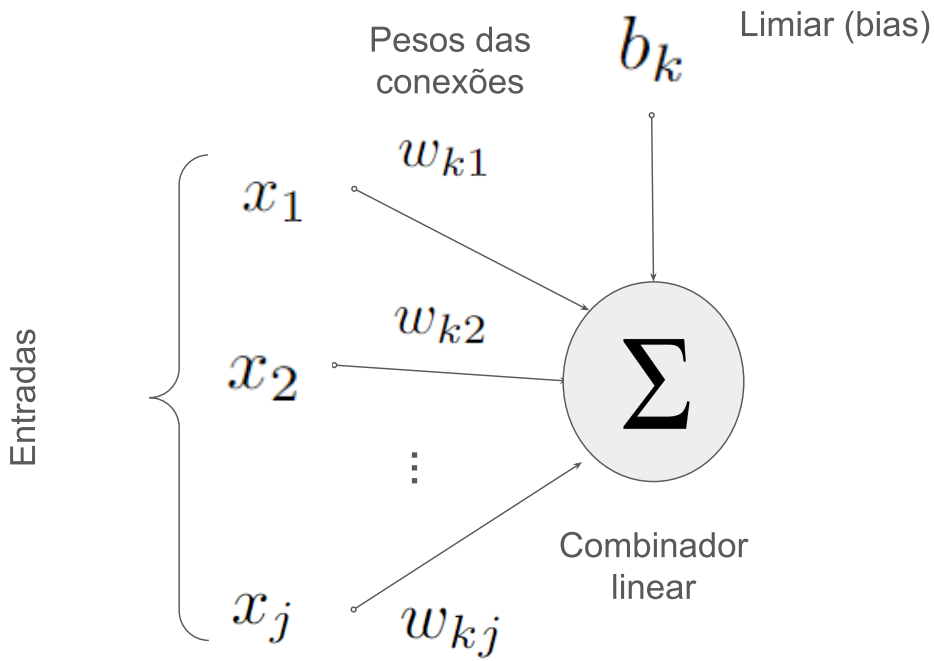

Figura: Modelo não-linear de um neurônio.

Fonte: Adaptado de Haykin (2007).

Perceptron

Uma função de ativação φ para restringir a amplitude da saída de um neurônio, o intervalo normalizado da amplitude da saída de um neurônio é escrito como o intervalo unitário fechado $[0, 1]$.

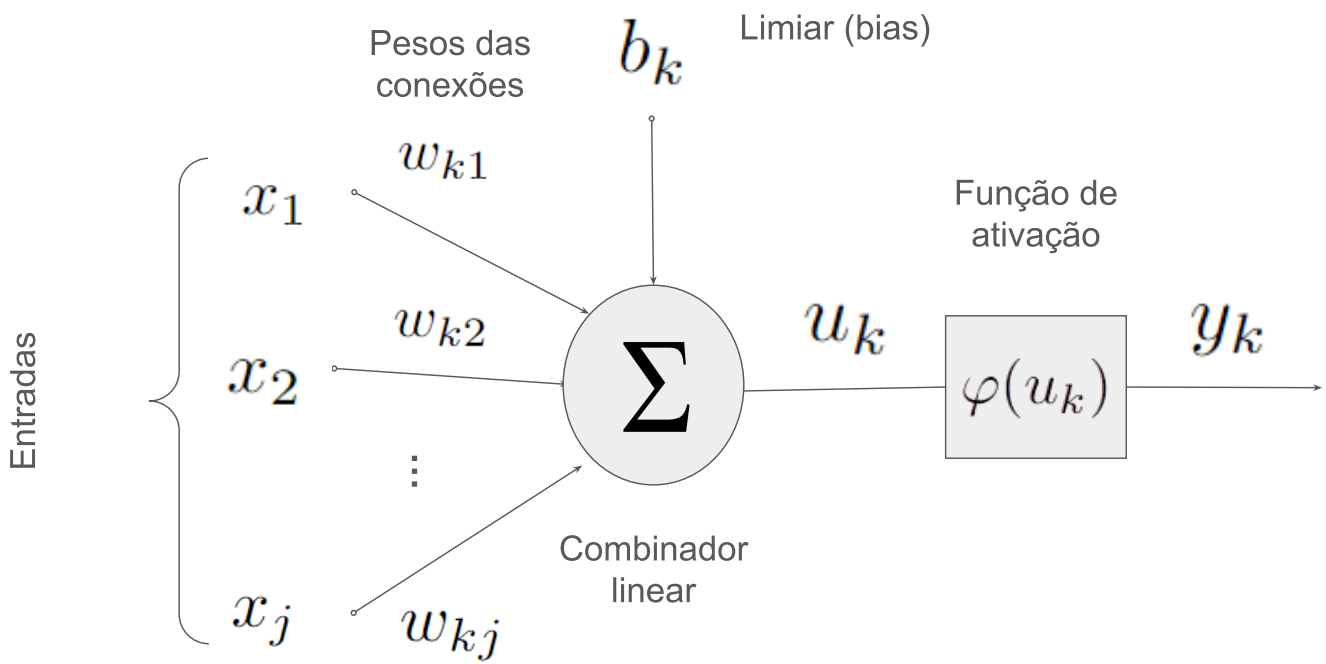

Figura: Modelo não-linear de um neurônio.

Fonte: Adaptado de Haykin (2007).

Exemplo de função de ativação

Função de limiar ou Heaviside

$$\varphi(v) = \begin{cases} 1 & \text{se } v \geq 0 \\ 0 & \text{se } v < 0 \end{cases} \tag{3}$$

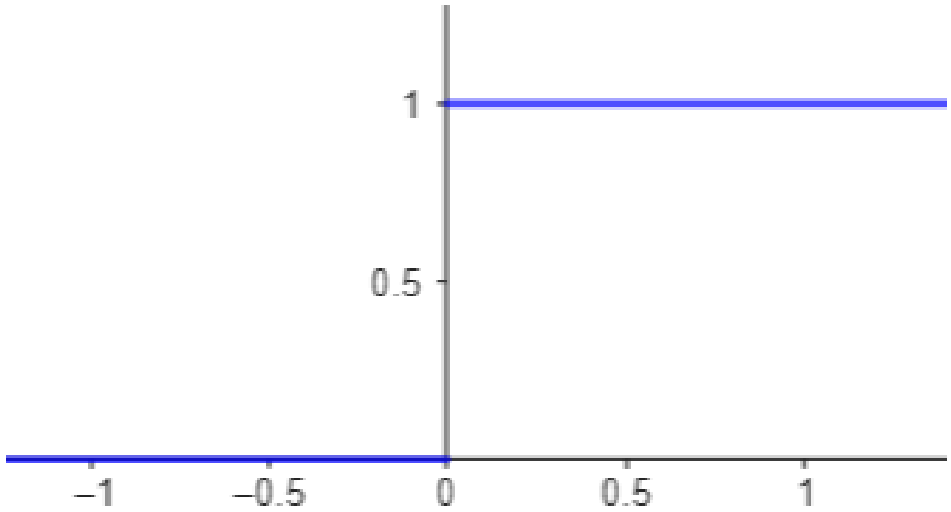

Figura: Função de limiar.

Fonte: Adaptado de Haykin (2007).

Redes Alimentadas Adiante com Camada Única

Na forma mais simples de uma rede em camadas, temos uma camada de entrada de nós de fonte que se projeta sobre uma camada de saída de neurônios (nós computacionais), mas não vice-versa. Em outras palavras, esta rede é estritamente do tipo alimentada adiante ou acíclica.

Redes Alimentadas Adiante com Camada Única

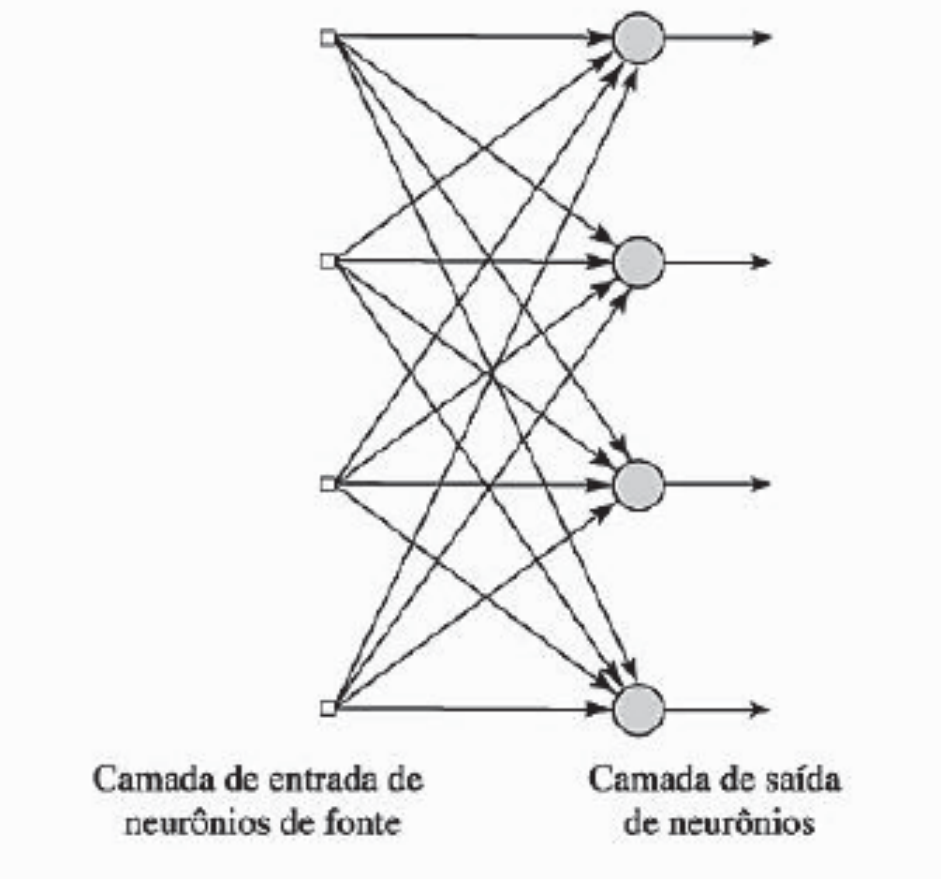

Figura: Rede alimentada adiante ou acíclica com uma única camada de neurônios

Fonte: Adaptado de Haykin (2007).

Sessões

Adaline

Para ilustrar nossa primeira regra de aprendizagem, considere o caso simples de um neurônio k que constitui o único nó computacional da camada de saída de uma rede neural alimentada adiante.

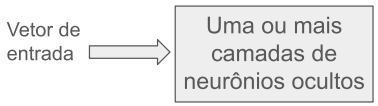

Figura: Modelo Adaline.

Fonte: Adaptado de Haykin (2007).

Adaline

O neurônio k é acionado por um vetor de sinal $x(n)$ produzido por uma ou mais camadas de neurônios ocultos. O argumento n representa o instante de tempo discreto, ou mais precisamente, o passo de tempo de um processo iterativo envolvido no ajuste dos pesos sinápticos, do neurônio k.

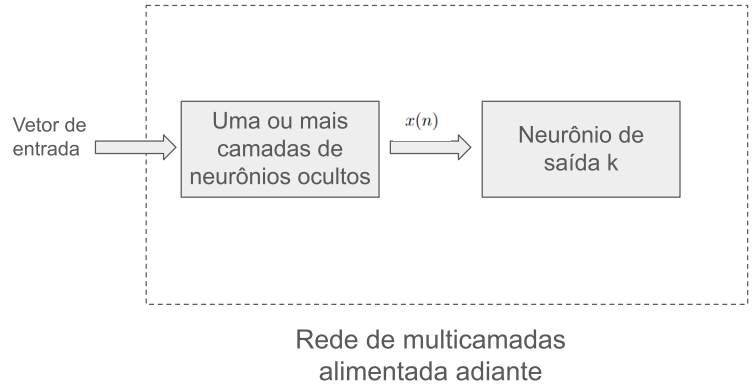

Figura: Modelo Adaline.

Fonte: Adaptado de Haykin (2007).

Adaline

O sinal de saída do neurônio k é representado por $y_k(n)$. Este sinal de saída, representando a única saída da rede neural, é comparado com uma resposta desejada ou saída-alvo, representada por $d_k(n)$.
Consequentemente, é produzido um sinal de erro, representado por $e_k(n)$.

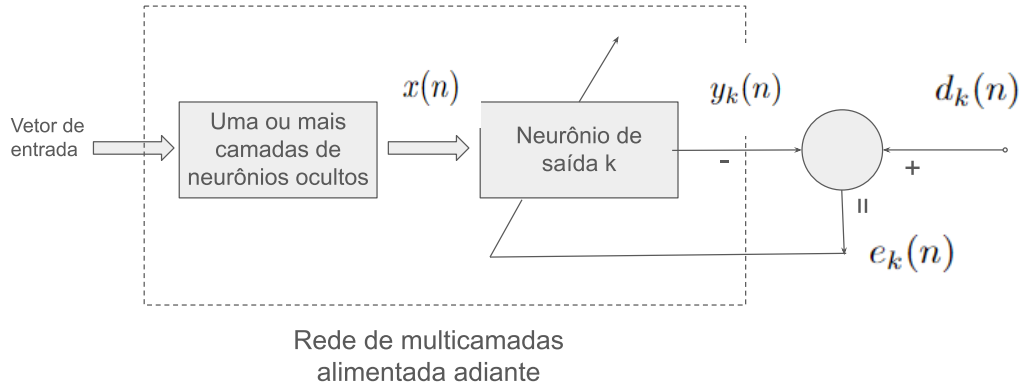

Figura: Modelo Adaline.

Fonte: Adaptado de Haykin (2007).

Sessões

Implementação em Python

De acordo com Feltrin (2020), o modelo mais básico usado na literatura para fins didáticos é o modelo onde é dada a representação de 3 espaços alocados para entradas, digamos, (x_1, x_2 e x_3).

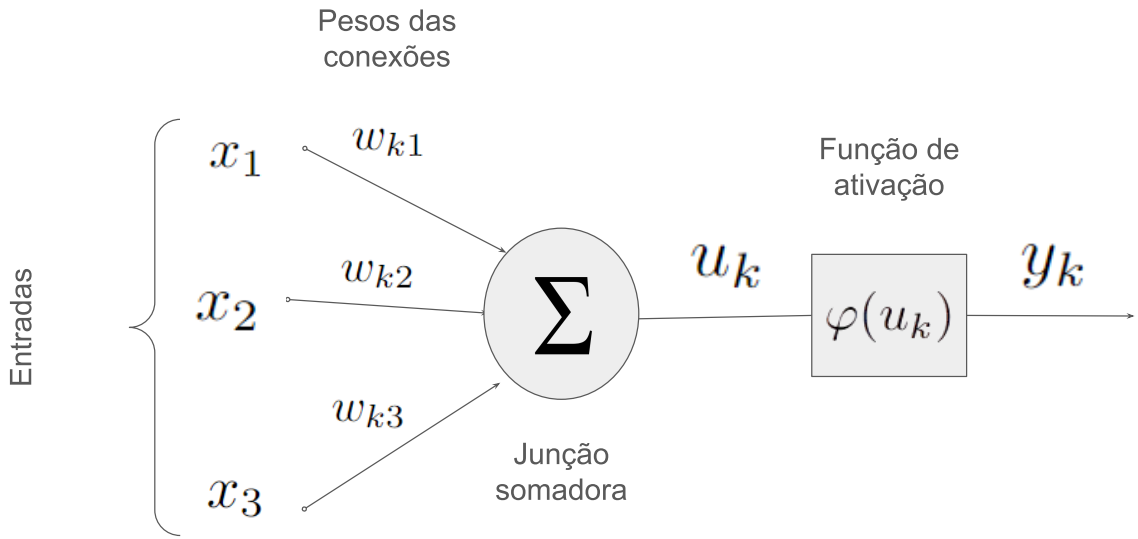

Figura: Modelo básico de um neurônio.

Fonte: Adaptado de Feltrin (2020).

Implementação em Python

Partindo para a prática, agora atribuindo valores e funções para essa estrutura, pode-se finalmente começar a entender de forma objetiva o processamento da mesma:

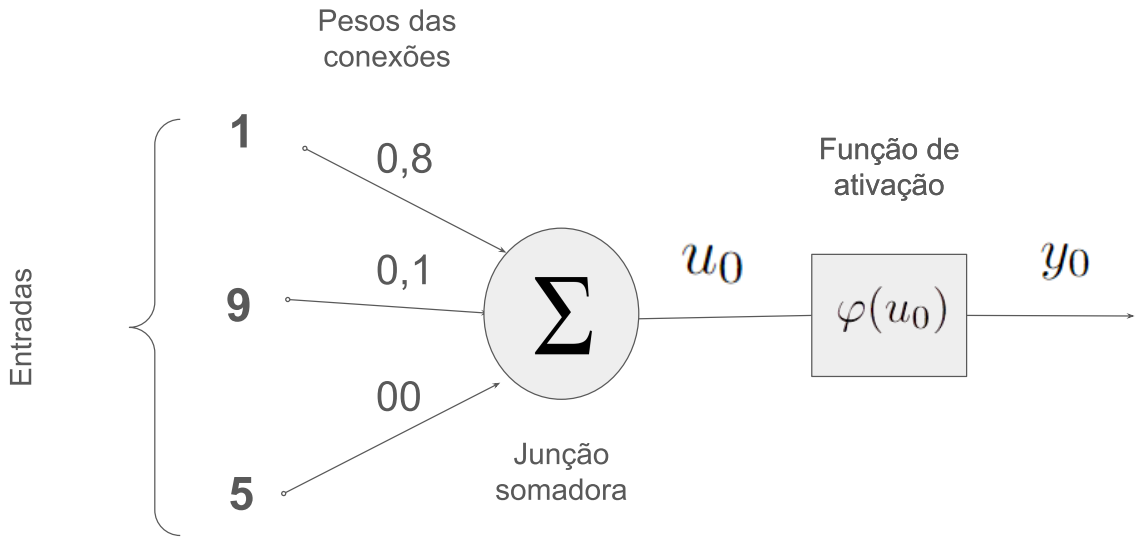

Figura: Modelo com entradas e pesos atribuídos.

Fonte: Adaptado de Feltrin (2020).

Implementação em Python

```
1   #Codigo em Python
2   #Atribuicao das entradas e pesos
3
4   x = [1, 9, 5]
5   w = [0.8, 0.1, 0]
```

Implementação em Python

Em seguida, a junção somadora aplica um produto escalar entre o vetor de entradas e pesos e atribui a u_0, pois é a primeira iteração.

$$
\begin{aligned}
u &= \mathbf{x} \cdot \mathbf{w} \\
&= (1, 9, 5) \cdot (0, 8, 0, 1, 0) \\
&= (1 \cdot 0, 8) + (9 \cdot 0, 1) + (5 \cdot 0) \\
&= 0, 8 + 0, 9 + 0 \\
&= 1, 7
\end{aligned} \tag{4}
$$

Implementação em Python

```
1  #implementacao funcao de soma
2
3  def ProdutoEscalar(x, w): #recebe entradas e pesos
4
5    u = 0 #inicia a variavel s
6
7    for k in range(3): #for do tamanho dos vetores de pesos
       e entradas
8
9      u += x[k]*w[k] #multiplicar cada entrada por seu peso
       e somar
10
11   return u
```

Implementação em Python

Por fim, a função de ativação neste caso, chamada de Função de Heaviside, possui uma regra bastante simples: se o valor resultado da função de soma for 1 ou maior que 1, o neurônio será ativado, se for uma valor abaixo de 1 (mesmo aproximado, mas menor que 1) este neurônio em questão não será ativado.

```
1  #definicao da funcao de ativacao
2
3  def Heaviside(u): #recebe o valor da soma
4
5    if u >= 1: #verifica se e maior que 1
6
7      return 1
8
9    return 0
10
11 Heaviside(u)
```

Sessões

Perceptron de uma camada - Tabela AND

Partindo para prática, será criado do zero uma rede neural que aprenderá o que é o mecanismo lógico de uma tabela AND.

X1	X2	Saída
0	0	0
0	1	0
1	0	0
1	1	1

Perceptron de uma camada - Tabela AND

Inicialmente é criada uma estrutura lógica que irá processar os valores, mas de forma errada, sendo assim, na chamada fase supervisionada, será treinada a rede para que ela aprenda o padrão referencial correto e de fato solucione o que é o processamento das entradas e as respectivas saídas de uma tabela AND.

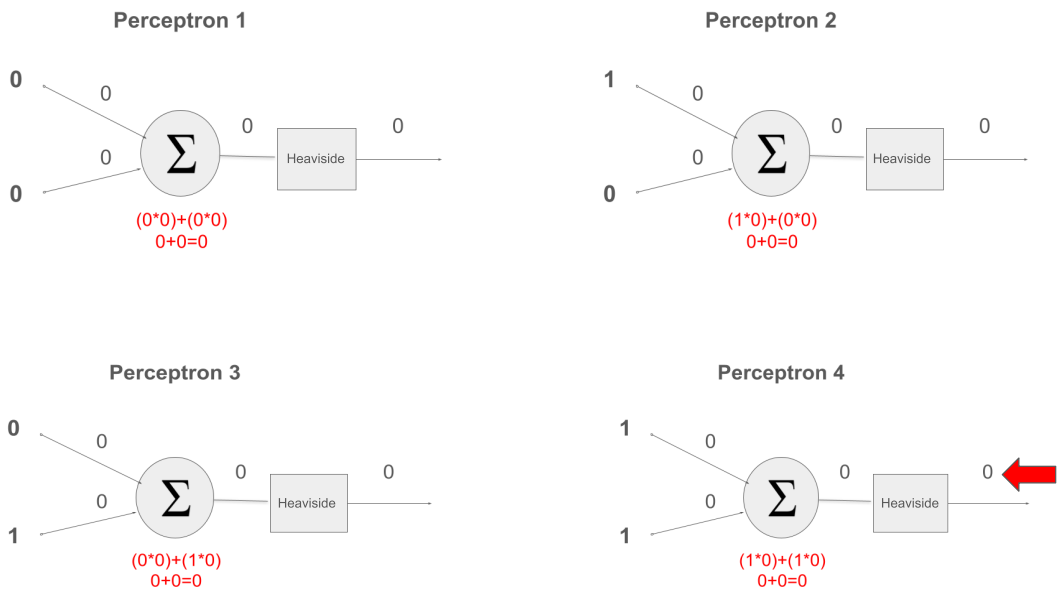

Figura: Modelo inicial.

Perceptron de uma camada - Tabela AND

No perceptron 4 a multiplicação das entradas pelos respectivos pesos como manda a regra resultou em 0, mas de acordo com a tabela verdade este resultado deveria ser 1. Em redes neurais básicas como estas, será feito o simples reajuste dos valores dos pesos e a repetição do processo para ver se o mesmo é corrigido.

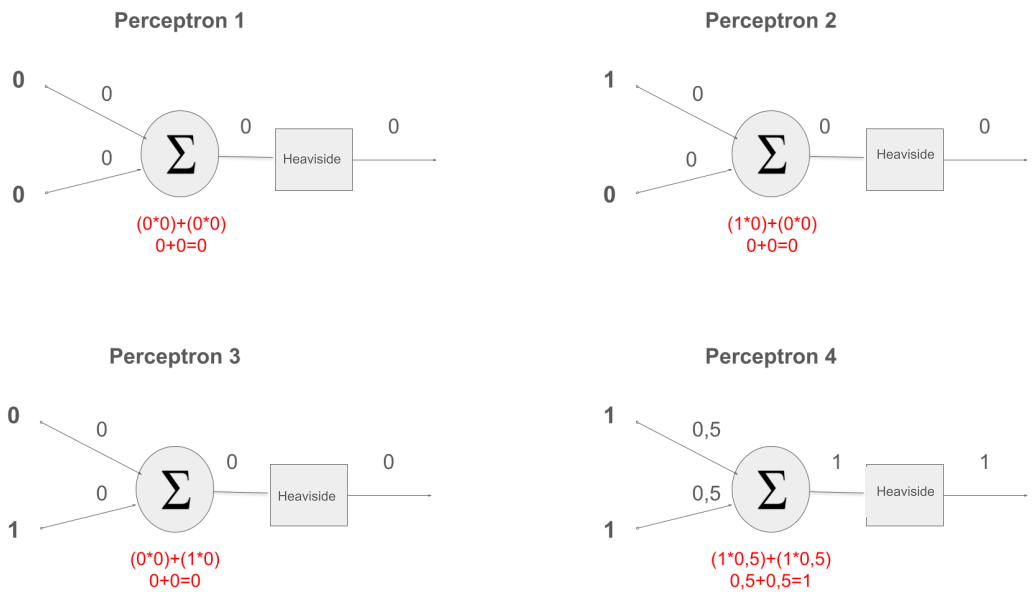

Figura: Primeira iteração.

Perceptron de uma camada - Tabela AND

Revisando apenas o perceptron 4, alterando os valores dos pesos de 0 para 0.5 e repetindo a função de soma, temos o valor esperado 1 (Tabela AND 1 e 1 = 1). Uma vez encontrado o valor de pesos que solucionou o problema, é hora de aplicar este mesmo valor de pesos a toda rede neural e verificar se este valor funciona para tudo.

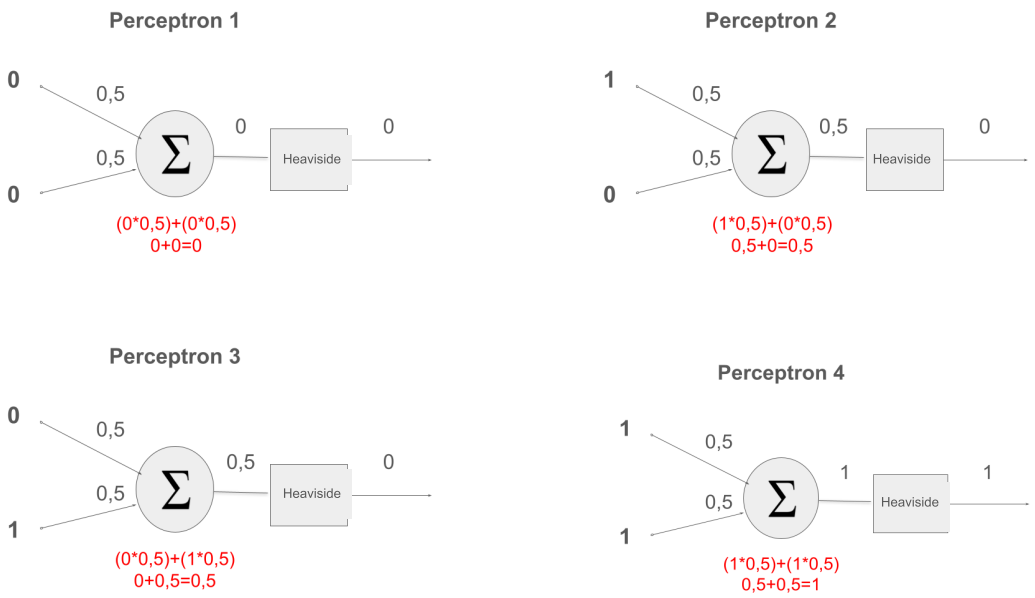

Figura: Segunda iteração.

Como exemplo de aplicação de rede neural na tomada de decisão imagina-se uma base de dados sobre diferentes shows para o mesmo cliente que leva em consideração se é longe (x_1), se é caro (x_2), se tem amigos para ir junto (x_3) e a decisão tomada (d). Onde 0 significa não e 1 significa sim. As decisões serão usadas de rótulo (solução desejada) na fase supervisionada da rede. Diante disso tem-se a seguinte tabela:

Show	x_1	x_2	x_3	d
1	0	0	1	1
2	1	0	1	1
3	1	1	1	0
4	1	1	0	0

A rede então será estruturada da seguinte forma:

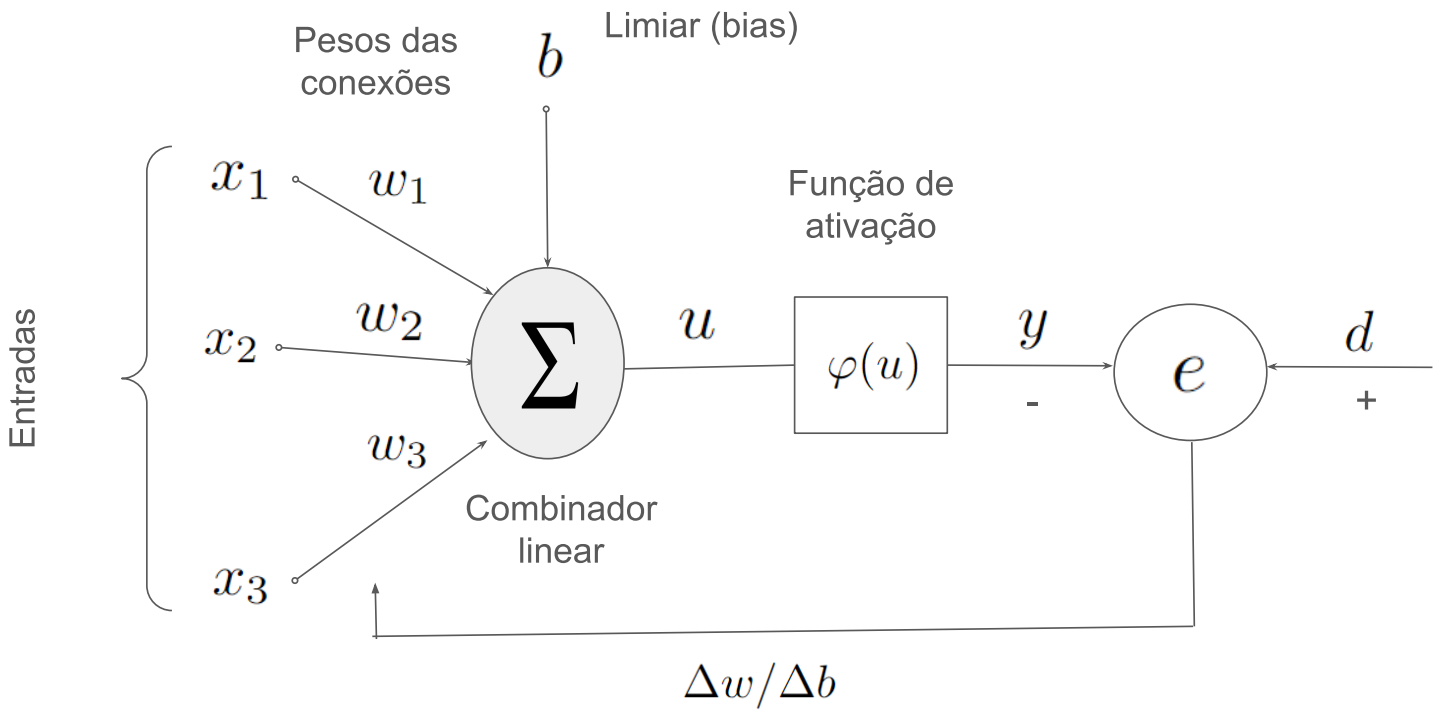

Figura: Rede de exemplo.

Exemplo escolha de show

Os cálculos serão feitos para cada show até que o erro total seja zero. Percorrer todos os shows e começar no primeiro de novo é chamada de uma época. Então começam os cálculos na época 1 show 1 com os pesos (w) zerados, bias (b) igual a 1 e taxa de aprendizado (η) igual a $0,1$.

$$
\begin{aligned}
u &= 0 \cdot 0 + 0 \cdot 0 + 1 \cdot 0 + 1 = 1 \\
y &= \varphi(1) = 1 \\
e &= 1 - 1 = 0 \\
\mathbf{w} &= (0,0,0) + [0,1 \cdot 0 \cdot (0,0,1)] = (0,0,0) \\
b &= 1 + (0,1 \cdot 0 \cdot 1) = 1
\end{aligned}
\tag{5}
$$

Sessões

Referências e materiais

Figura: Link do repositório.