RNAs: uma perspectiva mais profunda

1001513 – Aprendizado de Máquina 2
Turma A – 2023/2
Prof. Murilo Naldi

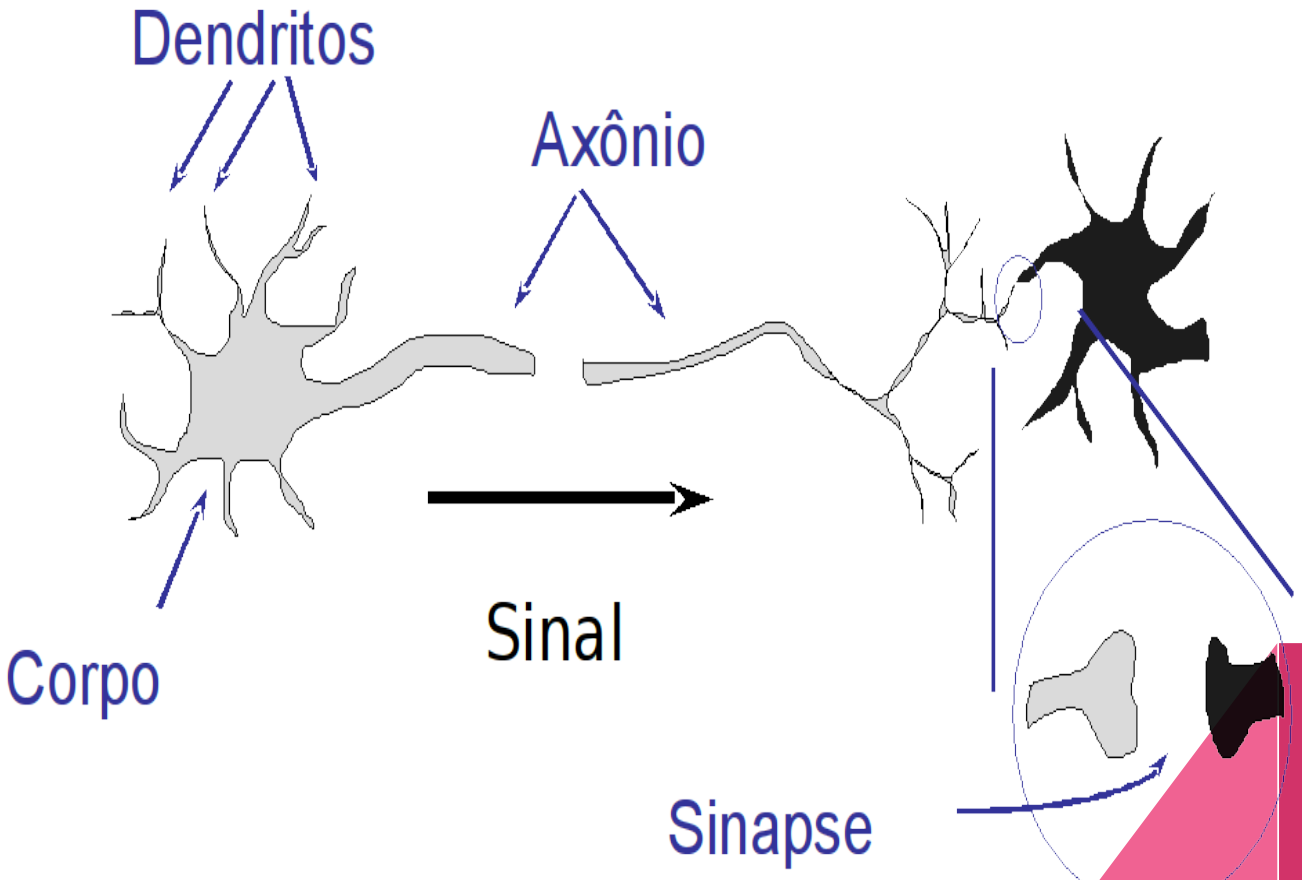
naldi@ufscar.br

Agradecimentos

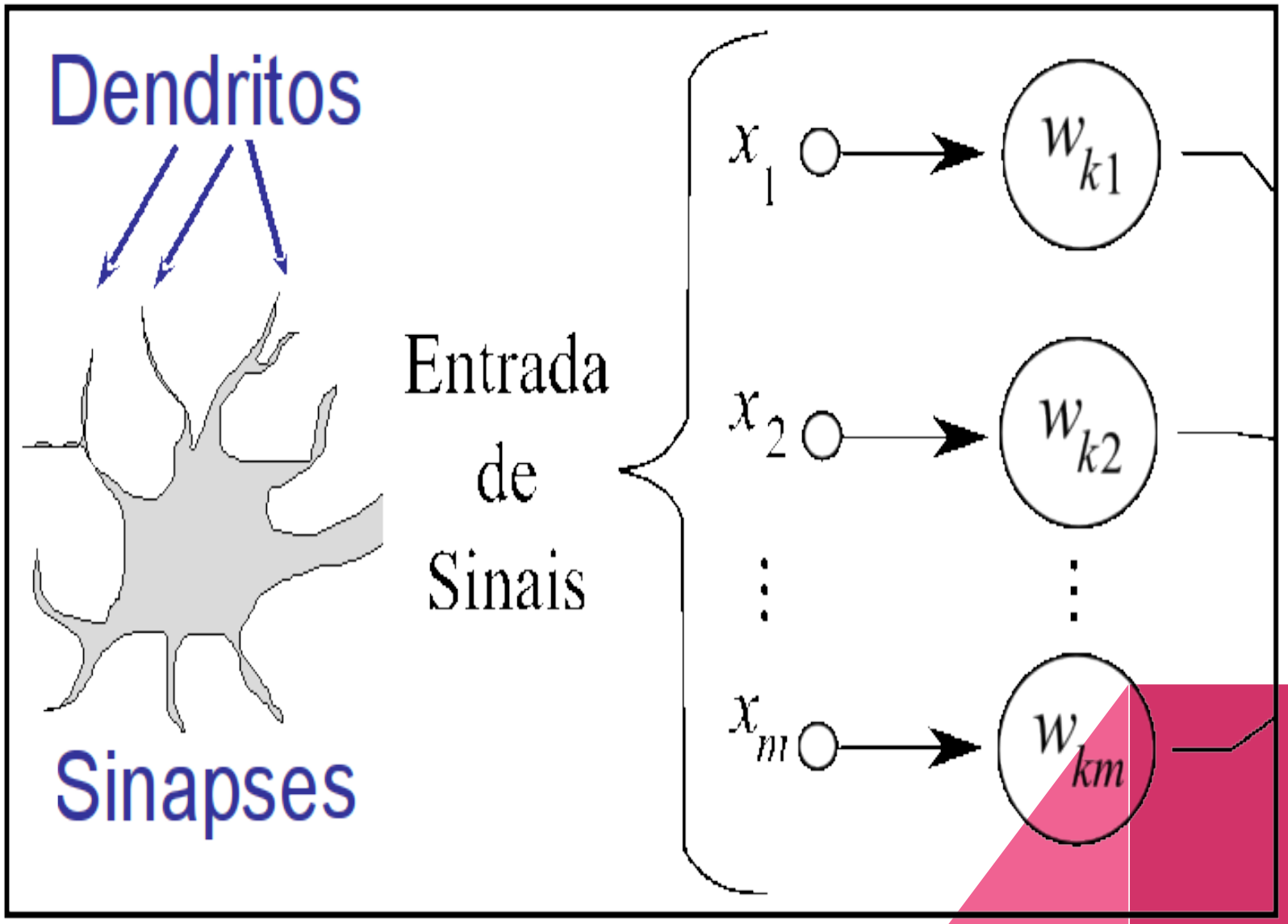
- O material desta aula é inspirado no trabalho dos seguintes professores: Ricardo Cerri, João Marcos Meirelles da Silva, Diego Silva, André Carvalho
- Intel IA Academy

Retomando

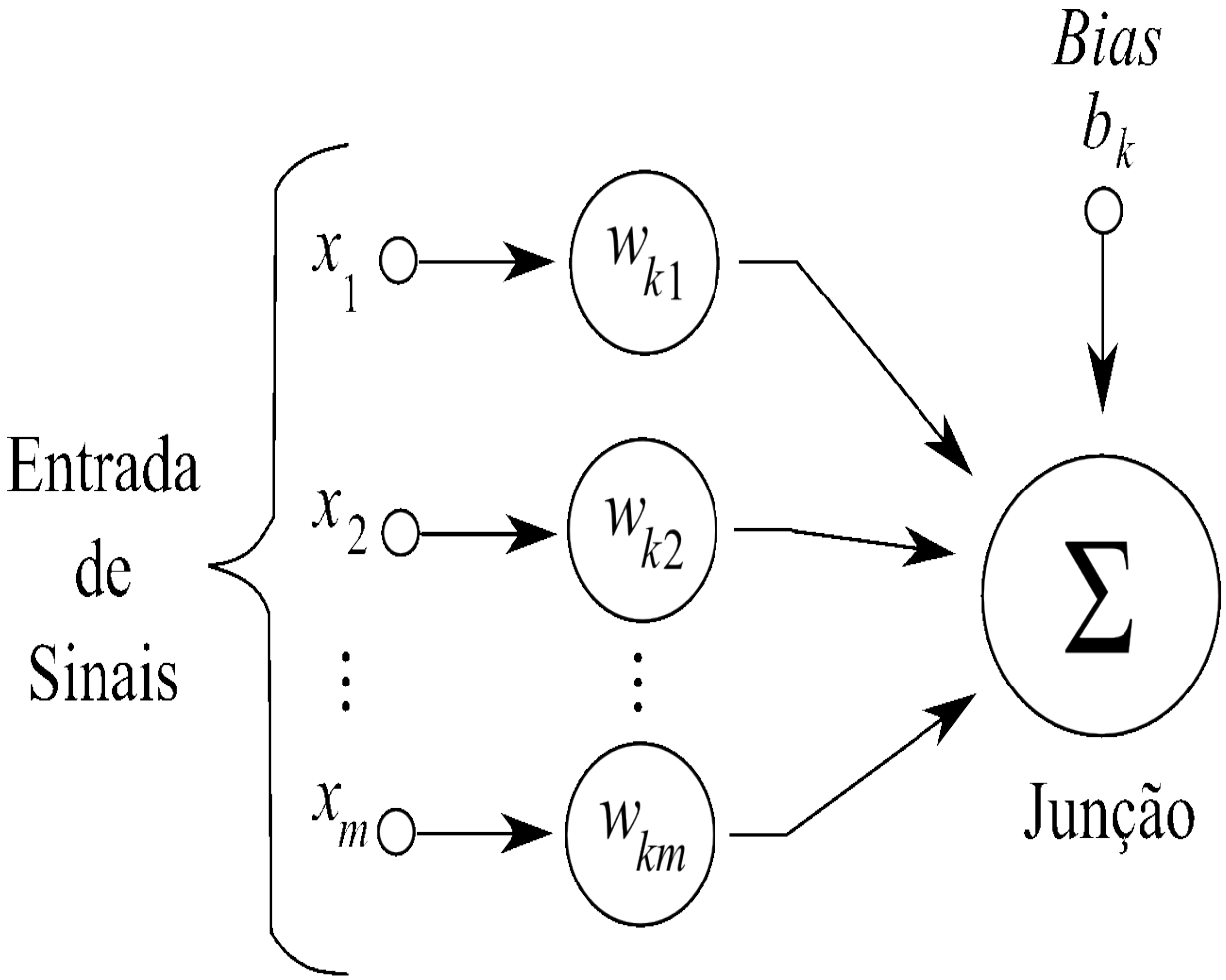
Neurônios possuem inspiração biológica



Entrada



Induced Local Field (v_k)

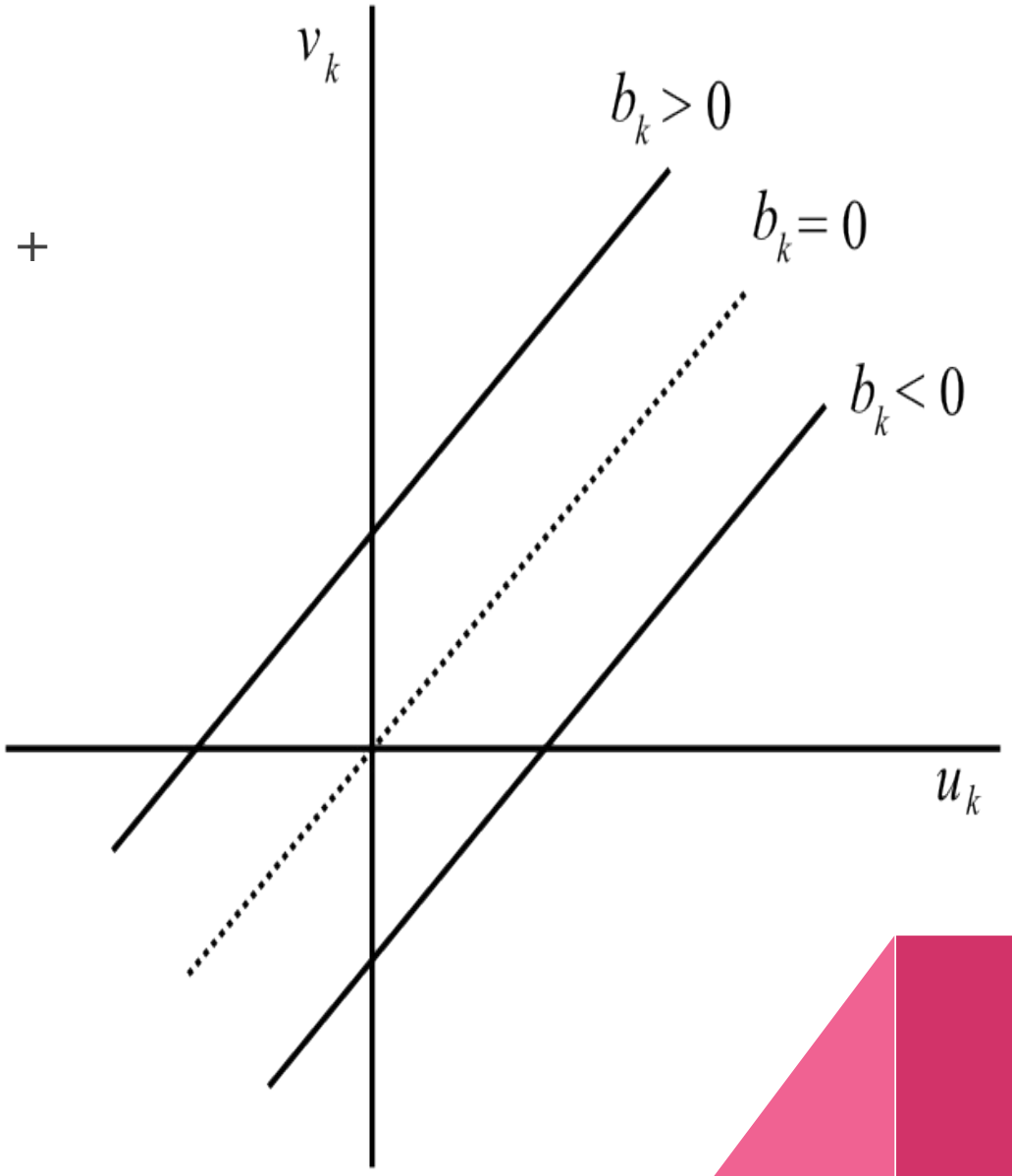


$$u_k = \sum_{j=1}^{m} w_{kj} x_j$$

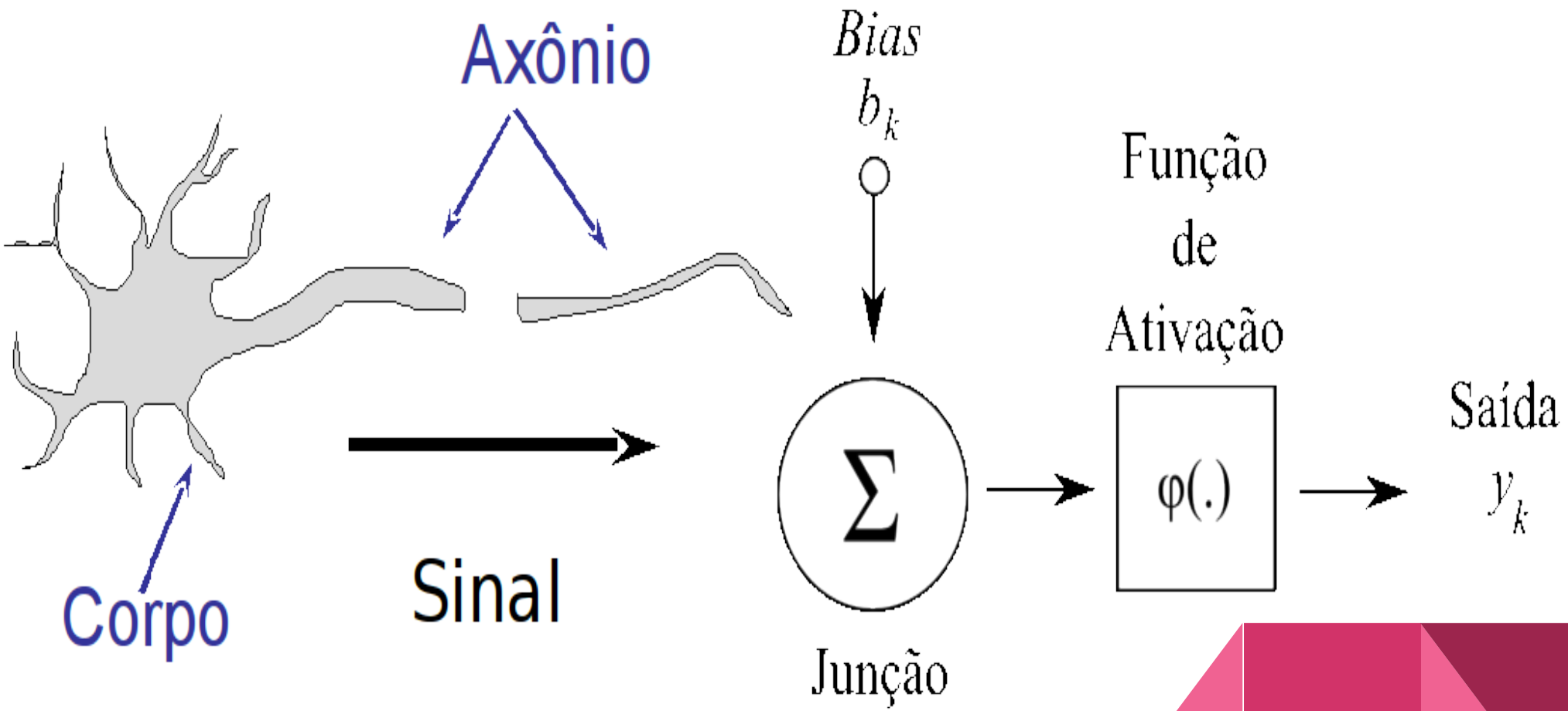
$$v_k = u_k + b_k$$

Induced Local Field (v_k)

Linear combination (u_k) +
Translation (b_k)

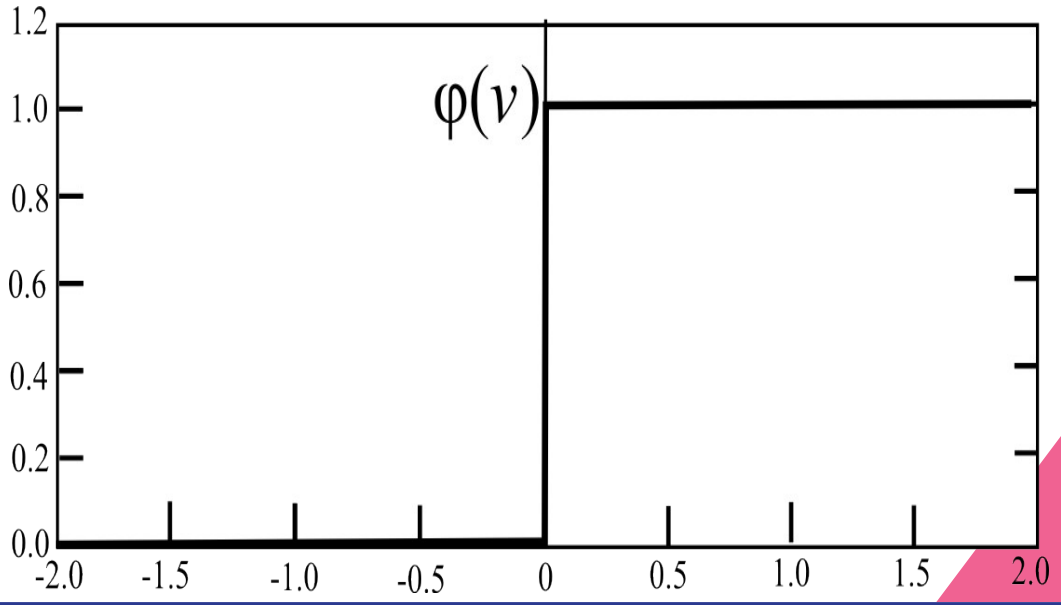


Função de ativação



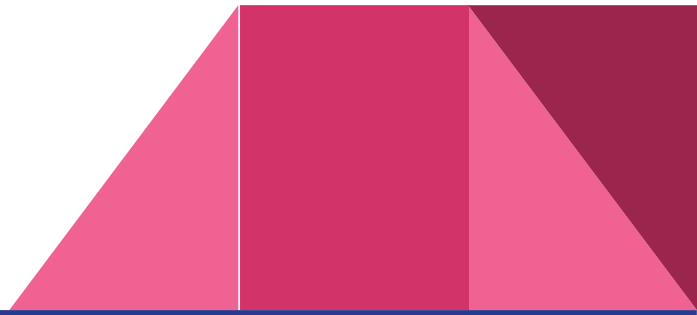
Limiar / *Binary Step*

$$\varphi(v_k) = \begin{cases} y_k = 1 & se \quad v_k \geq 0 \\ y_k = 0 & se \quad v_k < 0 \end{cases}$$



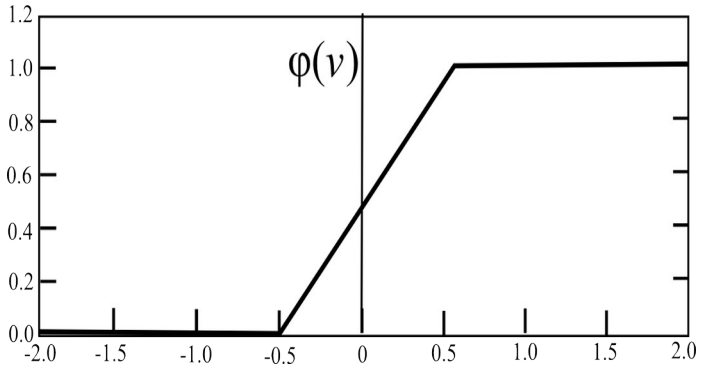
Limiar / *Binary Step*

- Limitações
  - não pode fornecer resultados com vários valores – por exemplo, não pode ser usado para problemas de classificação multiclasse
  - O gradiente da função degrau é zero
    - Dificultando aprendizado



Linear / Identidade

$$\varphi(v_k) = \begin{cases} y_k = 1 & se \quad v_k \geq \frac{1}{2} \\ y_k = v_k & se \quad \frac{1}{2} > v_k > -\frac{1}{2} \\ y_k = 0 & se \quad v_k \leq -\frac{1}{2} \end{cases}$$



Linear / Identidade

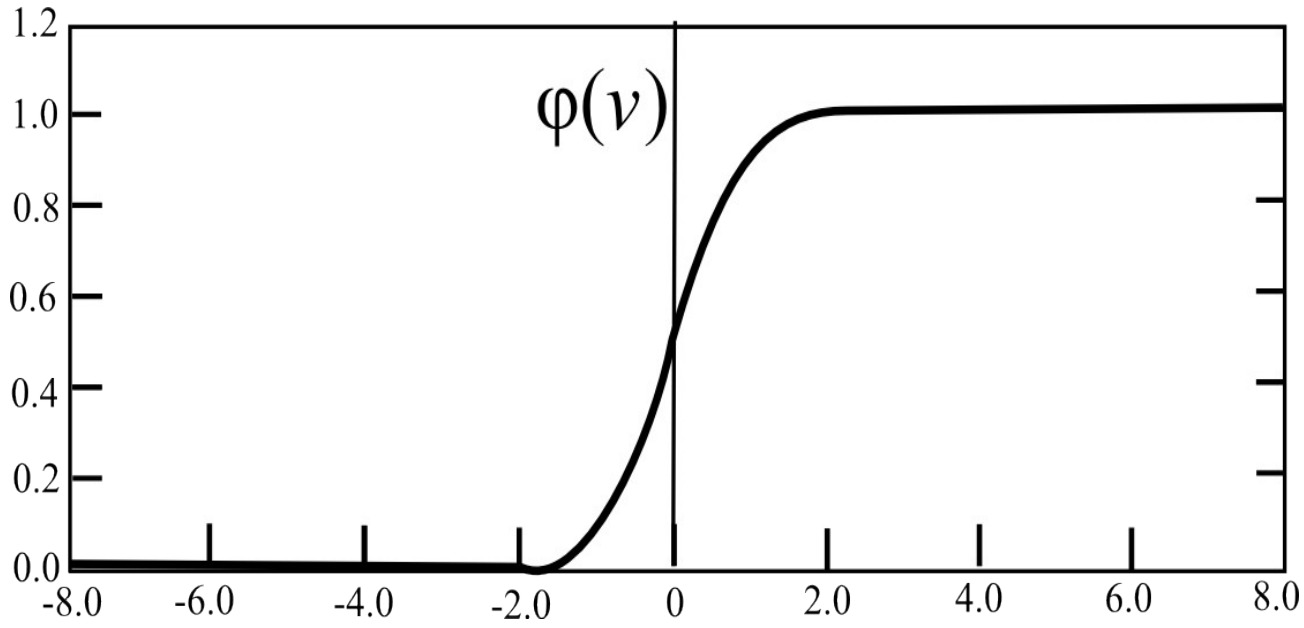
- A função não faz nada (nenhuma transformação) com a soma ponderada da entrada, ela simplesmente retorna o valor que foi fornecido (dentro de um intervalo
- Problemas:
  - A derivada da função é uma constante e não tem relação com a entrada
  - Todas as camadas da rede neural entrarão serão colapsados em uma.
    - Não importa o número de camadas da rede neural, a última camada ainda será uma função linear da primeira camada.
    - Então, essencialmente, uma função de ativação linear transforma a rede neural em apenas uma camada.

Funções não-lineares

- Muito mais interessantes para redes com várias camadas
  - Como a derivada da função está relacionada à entrada (anterior), é possível entender quais pesos nos neurônios das camadas anteriores podem fornecer uma melhor previsão
    - Pois podemos relacionar a função com o erro através do gradiente
  - Elas permitem o empilhamento de múltiplas camadas de neurônios, já que a saída seria agora uma combinação não linear de entrada passada por múltiplas camadas
    - Qualquer saída pode ser representada como um cálculo funcional em uma rede neural

Sigmóide / Logistic

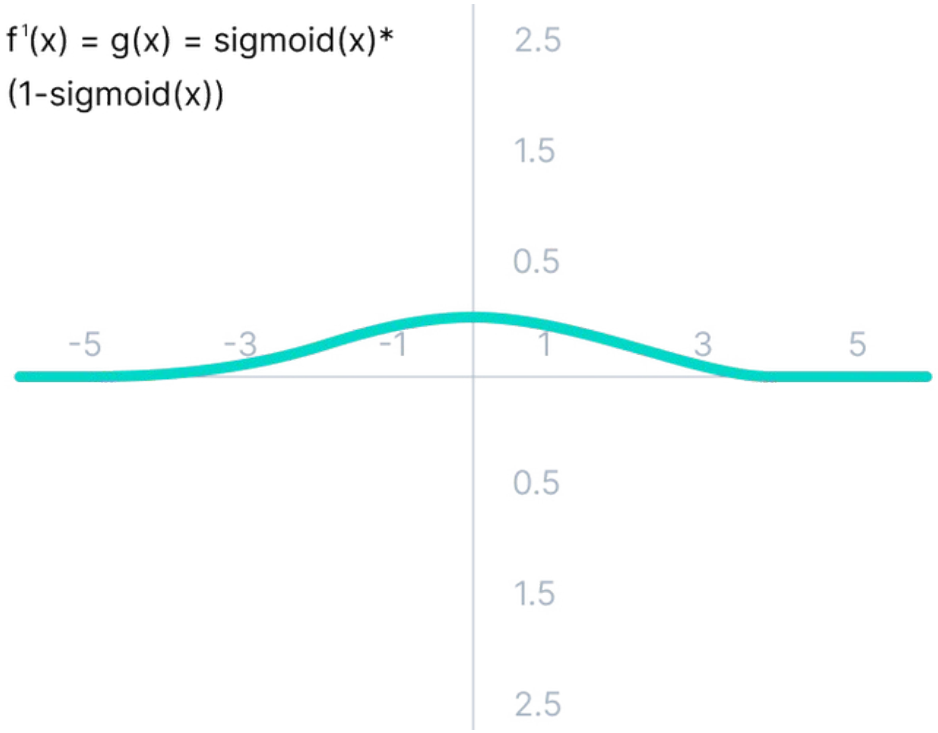
$$\varphi(v_k) = \frac{1}{1 + e^{(-av_k)}}$$



Sigmóide / Logistic

- Função recebe qualquer valor real e retorna valores entre 0 e 1
- Quanto maior a entrada (mais positiva), mais próximo o valor da saída estará de 1, enquanto quanto menor a entrada (mais negativa), mais próxima a saída estará de 0
- Vantagens:
  - É comumente usado para modelos onde temos que prever uma probabilidade como saída, pois seus valores então no intervalo [0,1]
  - A função é diferenciável e fornece um gradiente suave, ou seja, evita saltos nos valores de saída. Isto é representado na forma de S da função

Sigmóide / Logistic

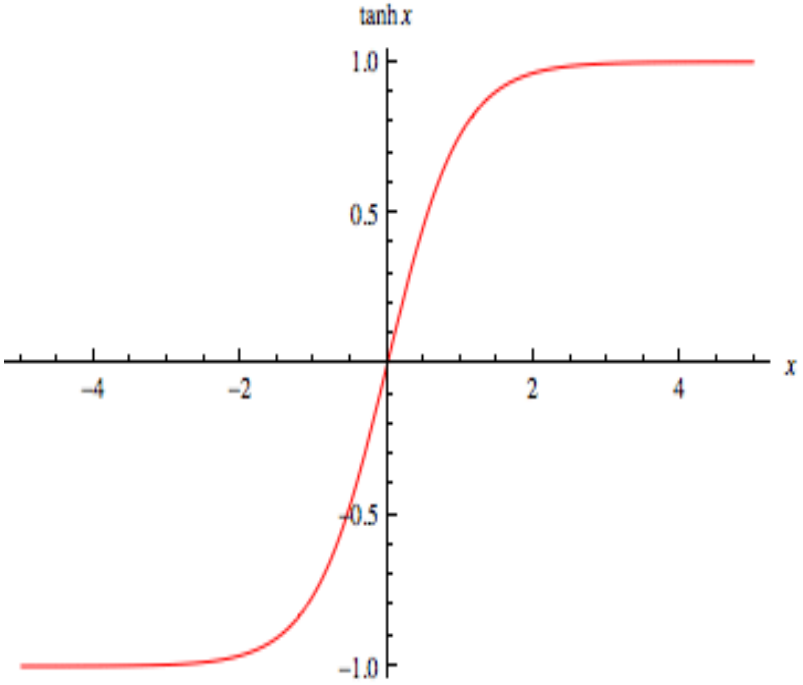


Activation Functions in Neural Networks - Pragati Baheti – V7 - Microsoft -www.v7labs.com/blog/neural-networks-activation-functions

- Desvantagens
  - A Figura mostra os valores de do gradiente para a função sigmóide
  - É possível perceber que valores de entrada maiores que 3 e menores que -3, a função possui valor de gradiente pequeno
  - Isso porque causar o problema de “desaparecimento do gradiente”
    - A saída da função é assimétrica em torno de zero
      - O que faz com que sejam de mesmo sinal e dificulta o treinamento

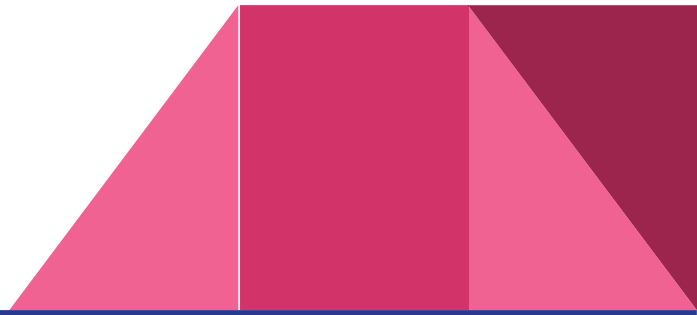
Tangente Hiperbólica

$$\varphi(v) = \frac{\left(e^{v} - e^{-x}\right)}{\left(e^{x} + e^{-x}\right)}$$



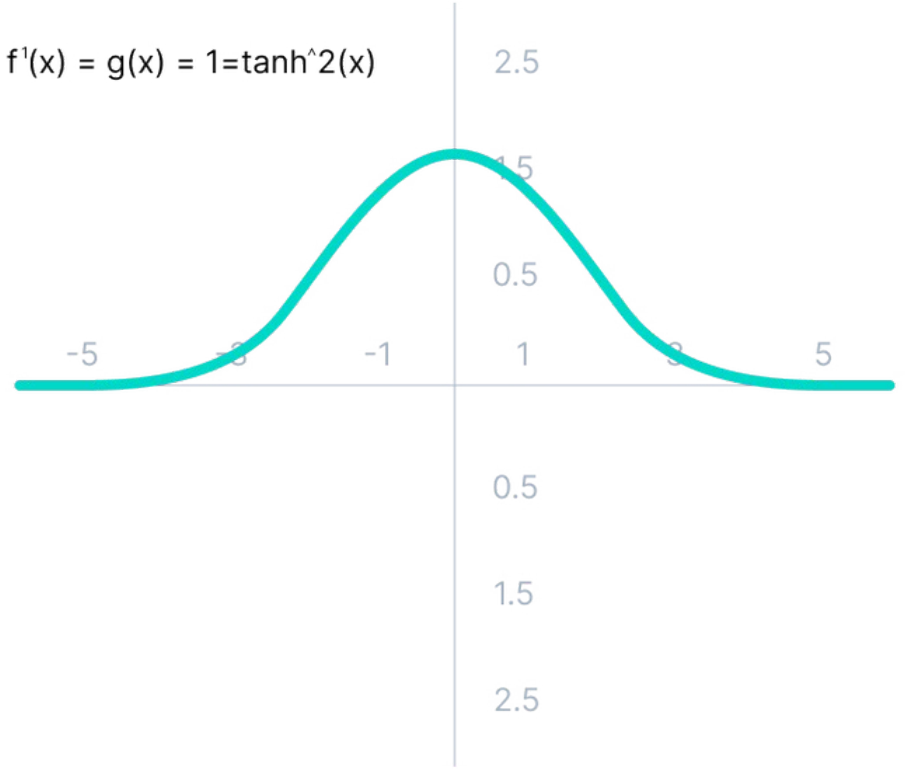
Tangente Hiperbólica

- A saída da função tangente é centralizada em zero
  - Podemos mapear os valores de saída como fortemente negativos, neutros ou fortemente positivos
- Bom para camadas intermediárias de uma rede neural, pois seus valores possuem média 0 ou muito próxima disso
  - Ajuda a centralizar os dados e torna o aprendizado para a próxima camada muito mais fácil
  - Isso é uma vantagem em relação a função sigmóide



Tangente Hiperbólica

- Desvantagens
  - Como pode ser visto, ele também enfrenta o problema de desaparecimento de gradientes
  - Além disso, o gradiente da função tanh é muito mais acentuado em comparação com a função sigmóide

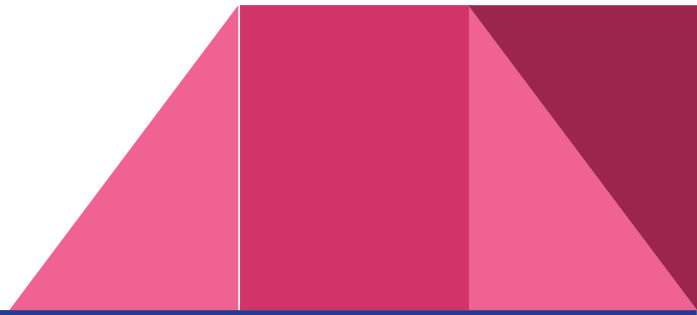


Activation Functions in Neural Networks - Pragati Baheti – V7 - Microsoft -www.v7labs.com/blog/neural-networks-activation-functions

Limitações

Funções clássicas, como a linear ou sigmóide, possuem limitações

- Funções não lineares podem modelar estruturas mais complexas
  - E dificultam o aprendizado, pois sua derivada não permite utilizar gradiente para indicar erro
- Sigmóide e tanh são saturadas e sensivelmente limitadas
  - As funções só são sensíveis a mudanças em torno do ponto médio de sua entrada

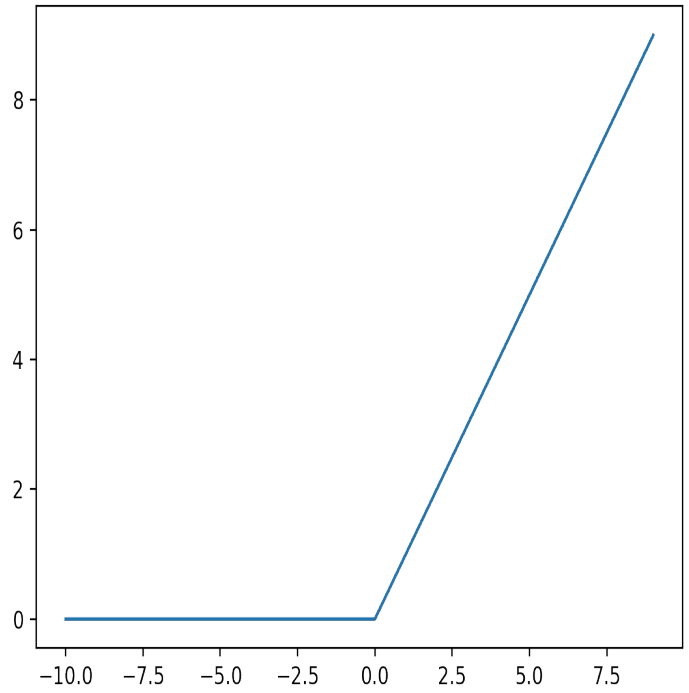


Rectified Linear Activation Function (ReLU)

Parece uma função linear, mas é uma função não linear que permite que relacionamentos complexos nos dados sejam aprendidos

- Fornece mais sensibilidade e evita a saturação fácil
  - A função de ativação linear retificada é um cálculo simples que retorna o valor fornecido como entrada diretamente ou o valor 0 se a entrada for 0 ou menos

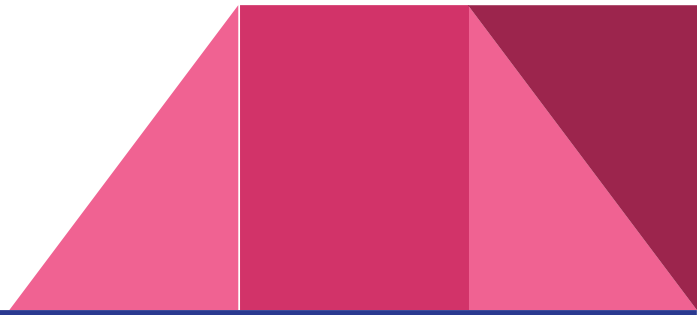
$$\varphi(v) = max(0, v)$$



Rectified Linear Activation Function (ReLU)

Vantagens:

- Fácil de implementar, não satura facilmente
- A derivada da função linear retificada também é fácil de calcular.
  - Lembre-se de que a derivada da função de ativação é necessária ao atualizar os pesos de um neurônio
  - A derivada da função é a inclinação que é 0 para valores negativos e 1 para valores positivos

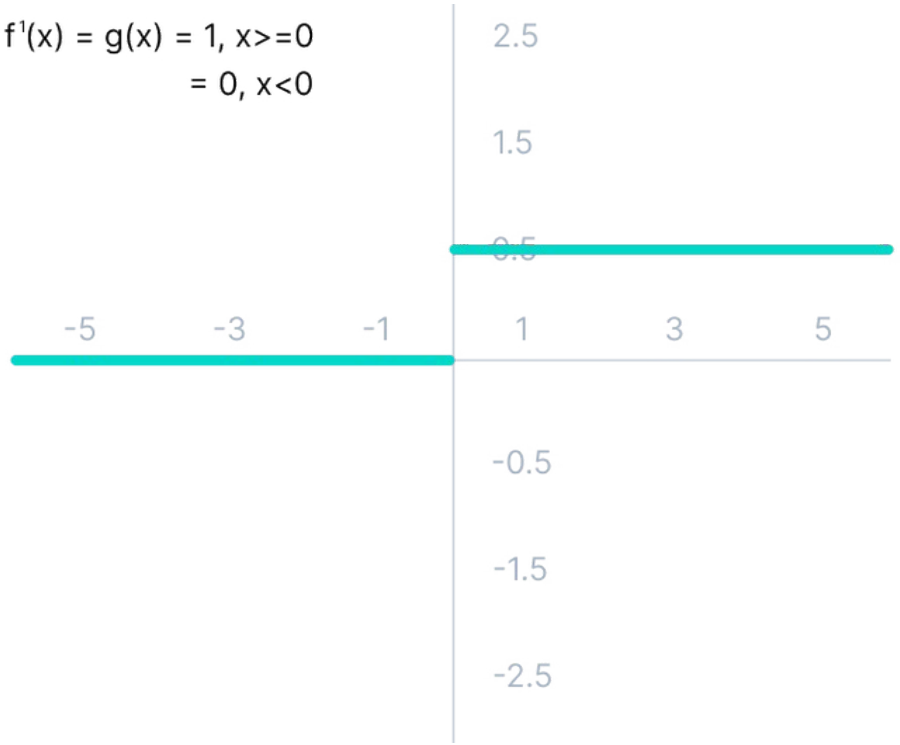


Rectified Linear Activation Function (ReLU)

Desvantagens:

- A parte negativa faz com que o gradiente vai a zero
  - Nessa parte não há aprendizado
  - Pode fazer com que alguns neurônios não seja atualizados
    - E portanto não aprendam

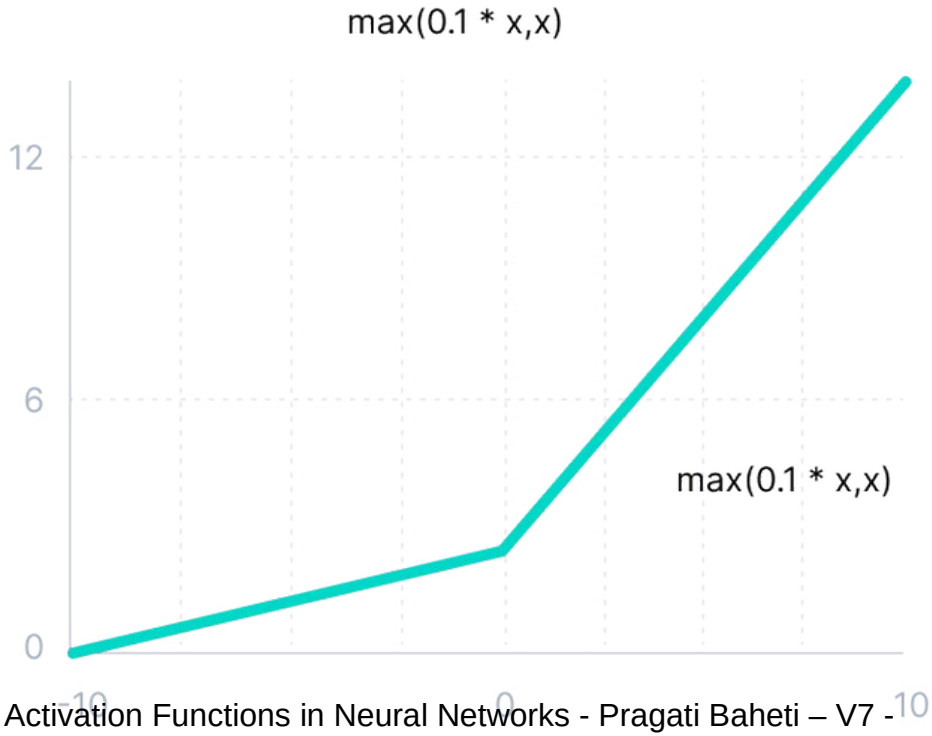
Conhecido como “*dying ReLU problem*”



Activation Functions in Neural Networks - Pragati Baheti – V7 - Microsoft -www.v7labs.com/blog/neural-networks-activation-functions

Leaky ReLU

- *Leaky ReLU* é uma versão melhorada da função ReLU para resolver o problema *Dying ReLU,* pois tem uma pequena inclinação positiva na área negativa.
- A vantagem é possibilitar a atualização de pesos mesmo para valores de entrada negativos



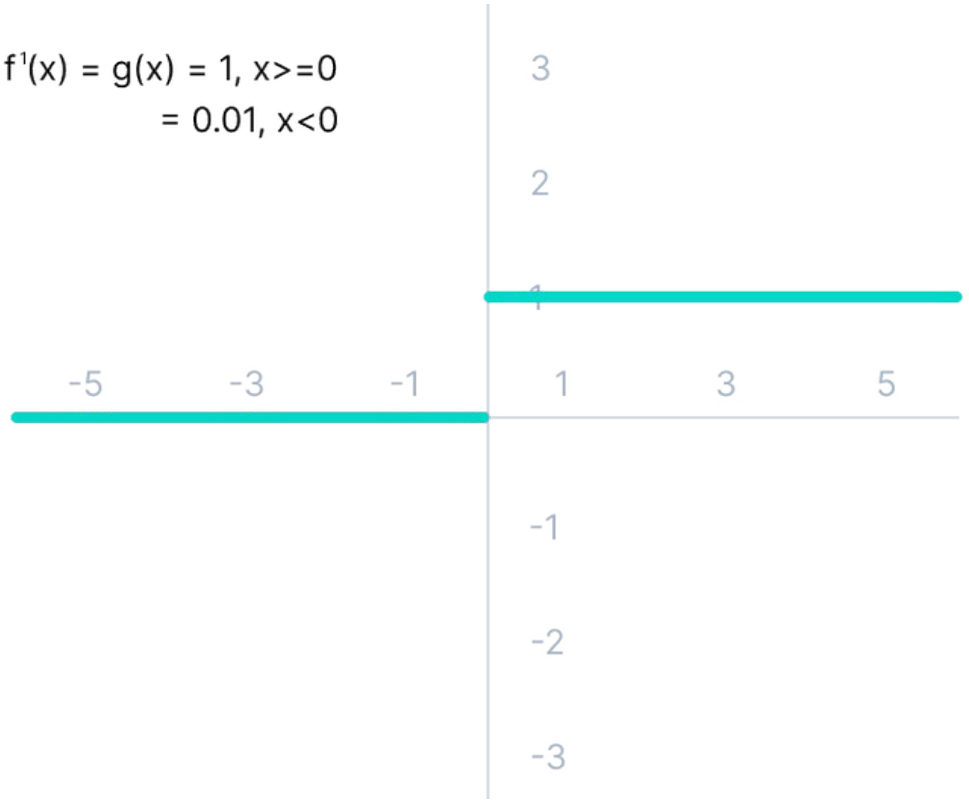
Activation Functions in Neural Networks - Pragati Baheti – V7 - Microsoft -www.v7labs.com/blog/neural-networks-activation-functions

$$\varphi(v) = max(0.1v, v)$$

Leaky ReLU

Desvantagens:

- A parte negativa faz com que o gradiente vá a 0.01
  - Predições podem ficar inconsistentes para valores negativos de entrada
  - Atualização demorada, computacionalmente custoso (valor é baixo)

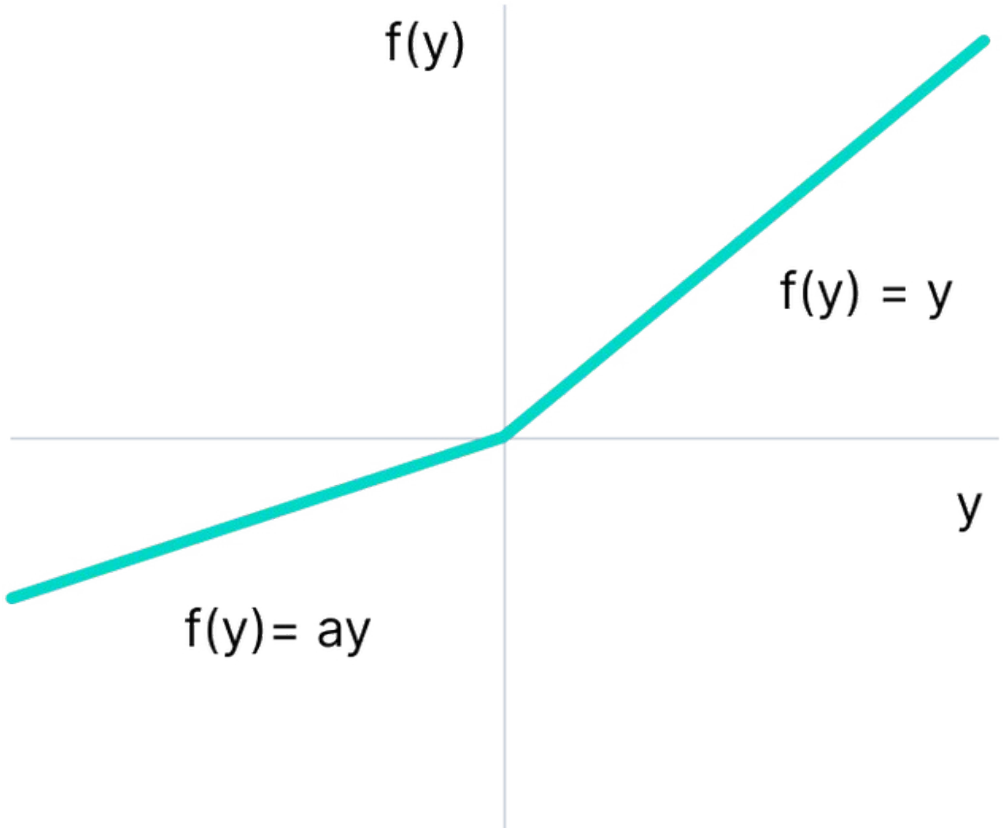


Activation Functions in Neural Networks - Pragati Baheti – V7 - Microsoft -www.v7labs.com/blog/neural-networks-activation-functions

Parametric ReLU

- *Parametric ReLU* insere um parâmetro de inclinação *a* no lugar da constante utilizada pela versão *Leaky*
- Usado para aumentar a inclinação da função em casos aonde o ajuste feito pela Leaky é insuficiente para evitar o "*dying Relu"* ou quer maior ajuste
- Pode causas instabilidade no treino

$$\varphi(v) = max(av, v)$$



Activation Functions in Neural Networks - Pragati Baheti – V7 - Microsoft -www.v7labs.com/blog/neural-networks-activation-functions

Exponential Linear Unit (ELU)

- ELU usa uma curva logarítmica para definir os valores negativos com uma linha reta, ao contrário das funções Leaky ReLU e ReLU paramétrica
- Isso causa suavização da função ReLU para valores negativos enquanto sua saída tende a -α
- Evita "*dying ReLU*" por causa de sua curva logarítmica

Activation Functions in Neural Networks - Pragati Baheti – V7 - Microsoft -www.v7labs.com/blog/neural-networks-activation-functions

$$\varphi(v)=\begin{cases} y=v & se \quad v\geq 0 \\ y=\alpha\left(e^{x}-1\right) & se \quad v<0 \end{cases}$$

Exponential Linear Unit (ELU)

- Desvantagens:
  - Função exponencial necessita de maior custo computacional
  - Pode causar problema do gradiente explosivo para redes muito profundas

Activation Functions in Neural Networks - Pragati Baheti – V7 - Microsoft -www.v7labs.com/blog/neural-networks-activation-functions

Softmax

- Muitas vezes queremos que a saída da rede neural seja uma probabilidade
  - Mesmo para uma função sigmóide que está no intervalo [0,1]
  - Ele receba as saídas da camada anterior e calcula as probabilidades relativas para cada classe
  - Por isso, comumente usado como uma função de ativação para a última camada da rede neural no caso de classificação multiclasse

$$\text{softmax}(z_i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$