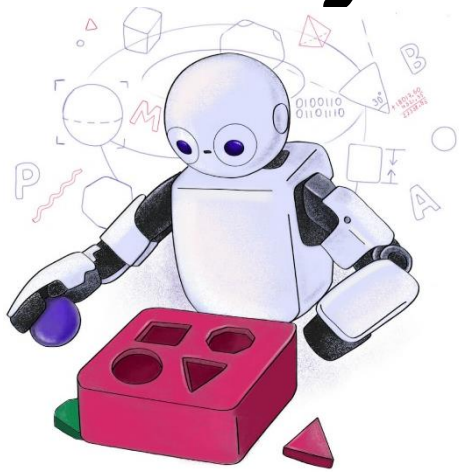
T320 - Introdução ao Aprendizado de Máquina II:

Redes Neurais Artificiais (Parte II)



Inatel

Felipe Augusto Pereira de Figueiredo
felipe.figueiredo@inatel.br

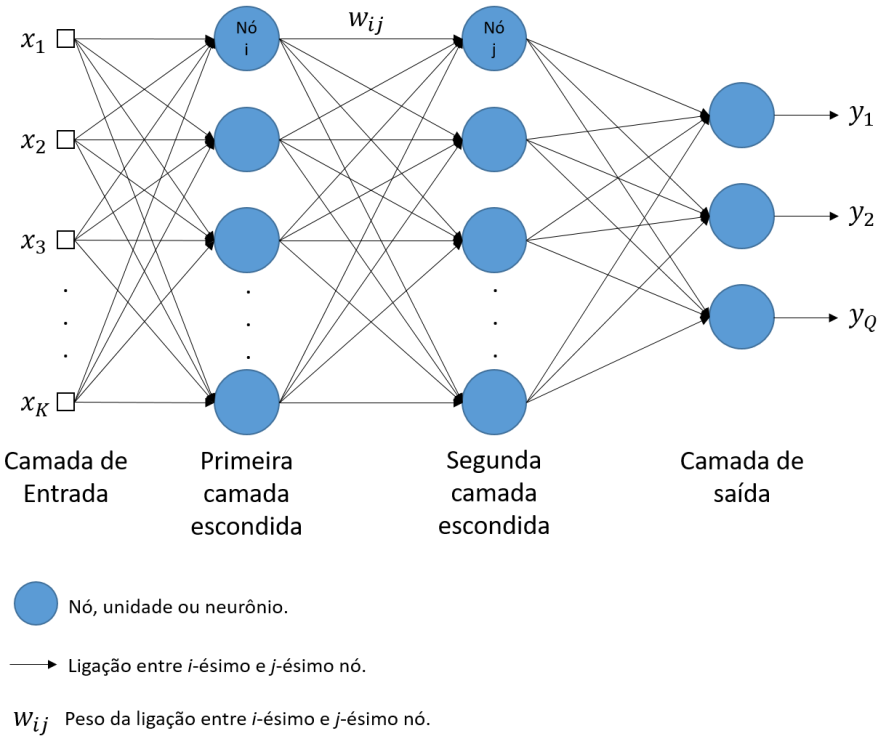
Recapitulando

- Fizemos uma analogia entre um neurônio e os modelos de McCulloch e Pitts e do Perceptron.
- Vimos a evolução dos modelos de McCulloch e Pitts para o Perceptron.
- Aprendemos suas características, diferenças e como ambos funcionam.
- Verificamos que um Perceptron é semelhante ao regressor logístico.
- Constatamos que um único Perceptron não é capaz de separar classes não-lineares, como por exemplo, o problema do XOR.
- Porém, quando combinamos vários deles, conseguimos criar um separador não-linear.
- Neste tópico, veremos que esta união de Perceptrons origina o que chamamos de **redes neurais artificiais**.

Perceptron de Múltiplas Camadas

- Em termos gerais, uma ***rede neural*** nada mais é do que uma coleção de ***neurônios*** conectados entre si através de ***ligações direcionadas*** (ou seja, as conexões têm uma direção associada).
- As propriedades da ***rede neural*** são determinadas por sua ***topologia*** e pelas propriedades dos ***neurônios*** (e.g., função de ativação e pesos).
- Algumas das limitações dos ***perceptrons*** (e.g., classificação apenas de classes linearmente separáveis) podem ser eliminadas adicionando-se camadas intermediárias (também chamadas de ocultas ou escondidas) de ***perceptrons***.
- A RNA resultante é denominada ***Perceptron de Múltiplas Camadas*** (em inglês, *Multilayer Perceptron* - MLP).

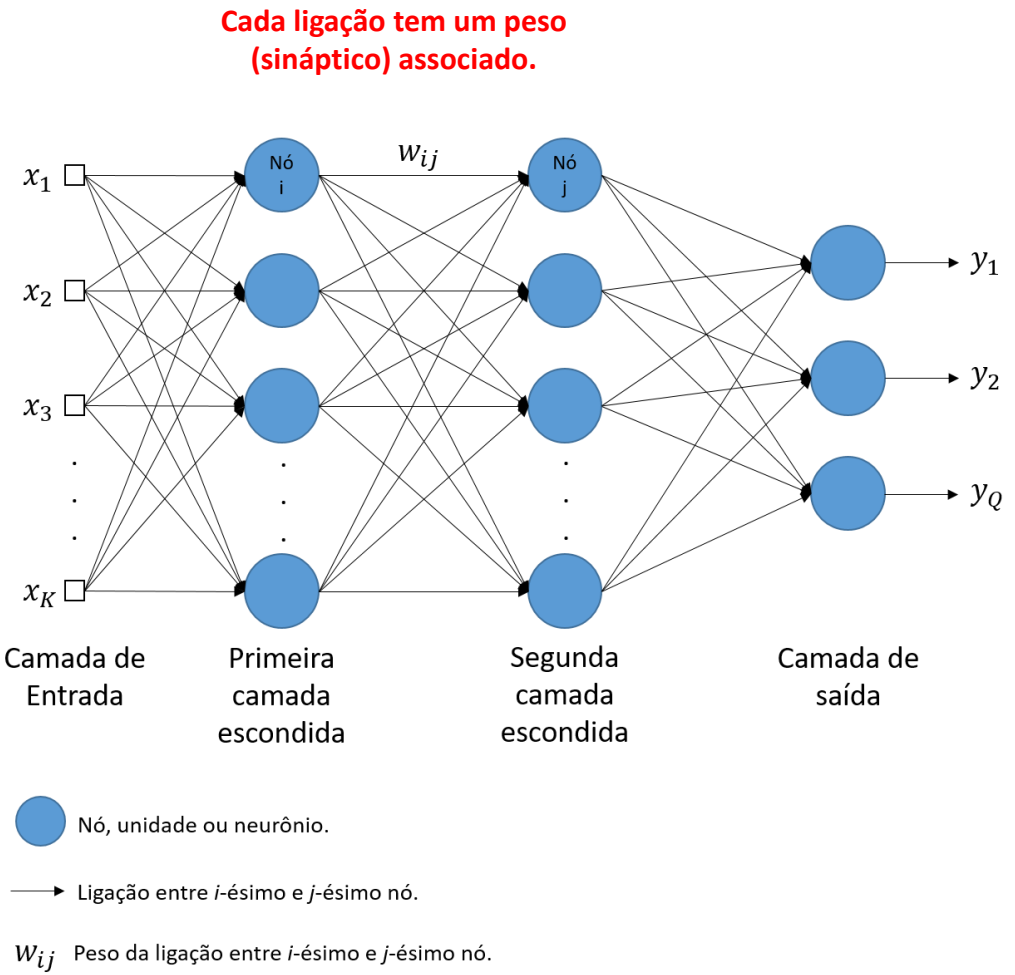
Cada ligação tem um peso (sináptico) associado.



OBS.: Neurônios também são chamados de ***nós*** ou ***unidades***.

Perceptron de Múltiplas Camadas

- Um exemplo de rede MLP com duas camadas intermediárias é mostrado na figura ao lado.
- As RNAs são o coração do Deep Learning.
  - Quando uma RNA tem duas ou mais camadas escondidas, ela é chamada de ***rede neural profunda*** (ou em inglês *Deep Neural Network* - DNN).
- **OBS**.: Em particular, uma MLP pode resolver o problema do XOR (lembre-se que um ***perceptron*** não é capaz de realizar essa tarefa).



Perceptron de Múltiplas Camadas

- A ***camada de entrada*** é o ponto de transferência dos ***atributos*** à rede.
- As ***camadas intermediárias*** realizam ***mapeamentos não-lineares*** que, idealmente, vão tornando a informação contida nos dados mais ***"explícita"*** do ponto de vista da tarefa que se deseja realizar.
  - Os mapeamentos são não-lineares devido às funções de ativação utilizadas não serem lineares.
- Por fim, os ***neurônios*** da ***camada de saída*** combinam a informação que lhes é oferecida pela última camada intermediária para formar as saídas.
- Redes MLPs são formadas por múltiplas camadas de ***Perceptrons***:
  - Portanto, tais redes têm por base o ***modelo de neurônio do Perceptron***.
- Esse modelo, discutido anteriormente, é mostrado na figura seguinte.

Perceptron de Múltiplas Camadas

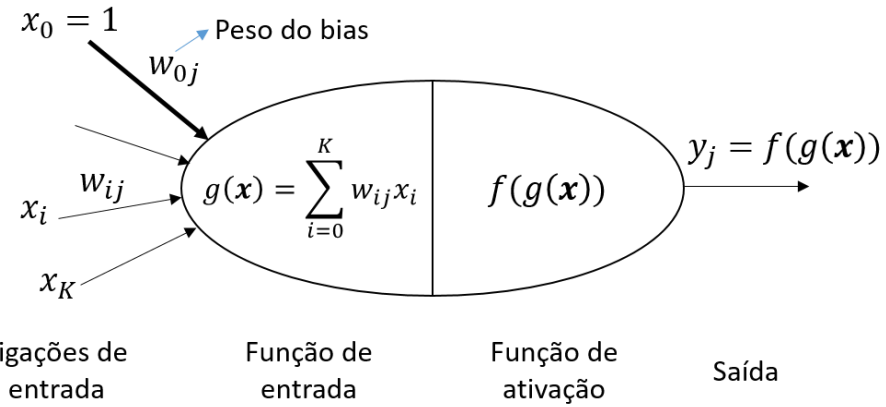
- Uma ***ligação*** do ***nó*** i para o ***nó*** j serve para propagar o sinal de ativação do ***nó*** i para o ***nó*** j. Cada ***ligação*** tem um ***peso*** associado, w_{ij}, que determina a ***força*** e o ***sinal*** da ***ligação***.
- Cada ***nó*** tem a entrada x_0 sempre com valor igual a 1 e um peso associado w_{0j}. Ou seja, esta entrada não está conectada a nenhum outro ***nó***.
- Cada ***nó*** j, calcula a soma ponderada de suas entrada da seguinte forma

$$g(\boldsymbol{x}) = \sum_{i=0}^{K} w_{ij} x_i.$$

- Em seguida, o ***nó*** aplica uma ***função de ativação*** (ou de limiar), $f(.)$, ao somatório acima para obter sua saída

$$y_j = f(g(\boldsymbol{x})) = f\left(\sum_{i=0}^{N} w_{ij} x_i\right) = f(\boldsymbol{w}^T \boldsymbol{x}).$$

- Existem vários tipos de ***funções de ativação*** que podem ser utilizadas pelos ***nós*** de uma rede MLP.
- Cada camada da rede pode usar funções de ativação diferentes.



$$y_j = f\left(\sum_{i=0}^{K} w_{ij} x_i\right),$$

onde x_i é a saída do nó i e w_{ij} é o peso conectando a saída do nó i para este nó, o nó j.

Funções de ativação

- Devido às suas características, não se utiliza a ***função degrau*** como função de ativação em MLPs.
- Até o surgimento das ***redes neurais profundas***, a regra era utilizar as ***funções logística*** ou ***tangente hiperbólica***, que são versões suavizadas da função degrau.
  - Essas funções possuem derivada definida e diferente de 0 em todos os pontos.
- A ***função logística*** tem a seguinte expressão:

$$y_j = f(z_j) = \frac{e^{pz_j}}{e^{pz_j} + 1} = \frac{1}{1 + e^{-pz_j}}.$$

- Sua derivada é dada por

$$\frac{dy_j}{dz_j} = \frac{df(z_j)}{dz_j} = py_j(1 - y_j) > 0,$$

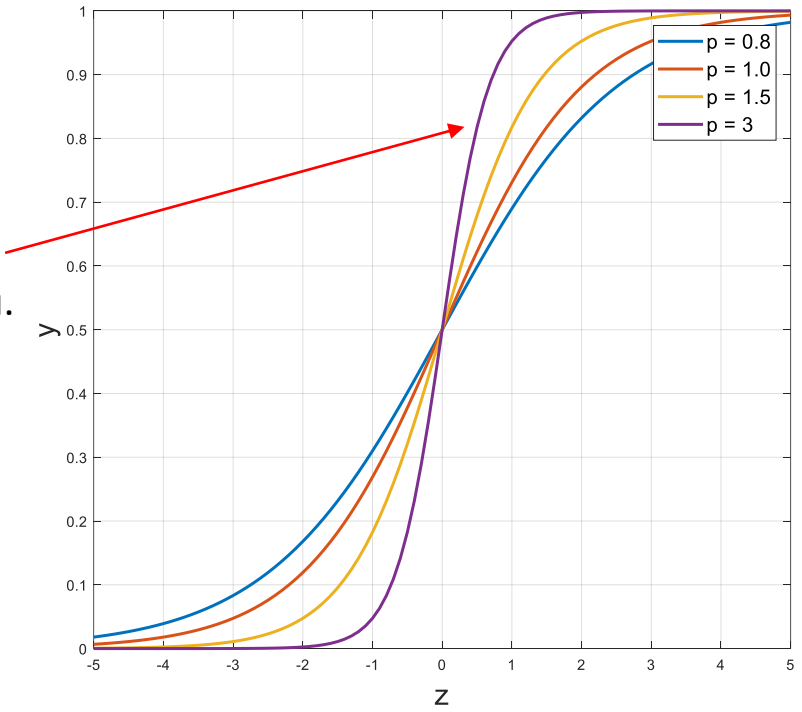
onde p é o ***fator de suavização*** da função de ativação logística.

- A derivada será importante durante o processo de aprendizado da rede neural.

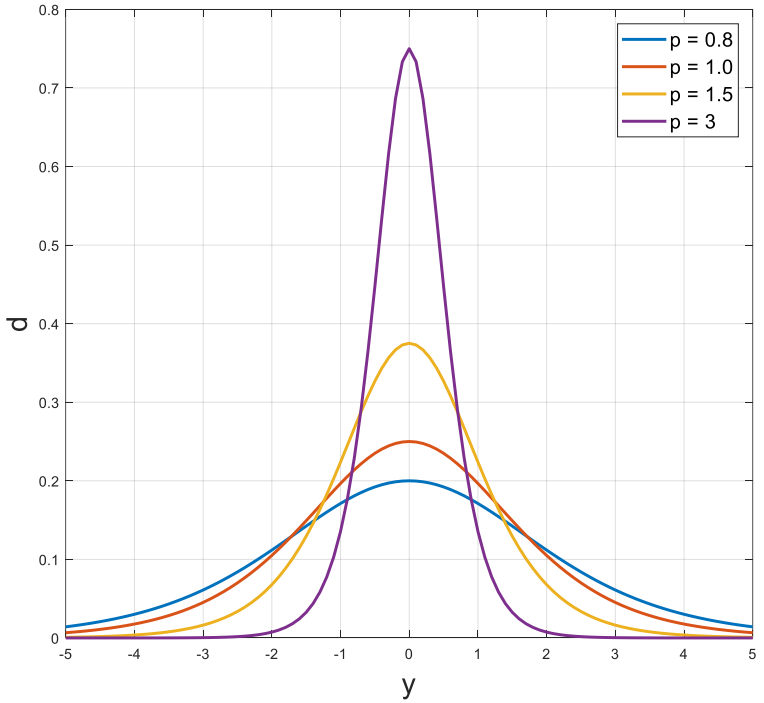
Funções de ativação

- A ***função logística*** e sua derivada para alguns valores do ***fator de suavização*** são mostradas nas figuras ao lado.
  - Normalmente, se utiliza $p = 1$.

OBS.: Quanto maior p, mais próxima ela fica da função degrau.



Função Logística



Derivada da Função Logística

OBS.: tende ao impulso conforme p aumenta.

Funções de ativação

- A ***função tangente hiperbólica*** tem sua expressão dada por:

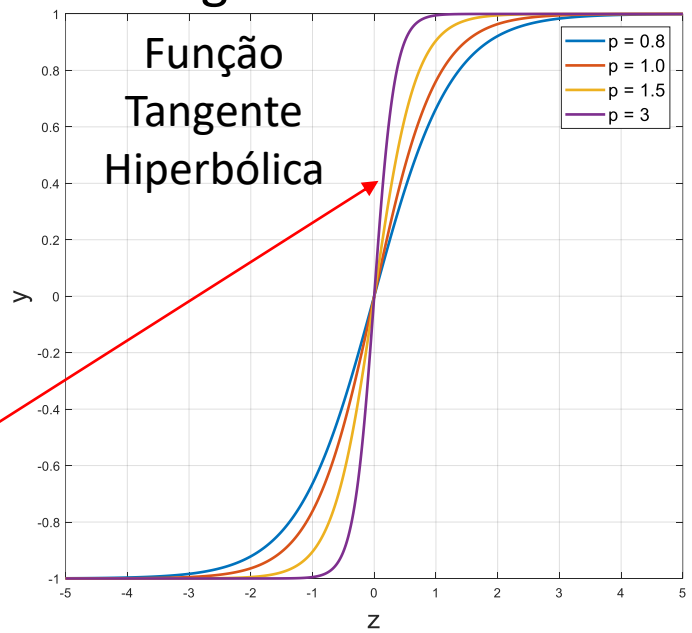
$$y_j = f(z_j) = \tanh(pz_j) = \frac{e^{pz_j} - e^{-pz_j}}{e^{pz_j} + e^{-pz_j}}.$$

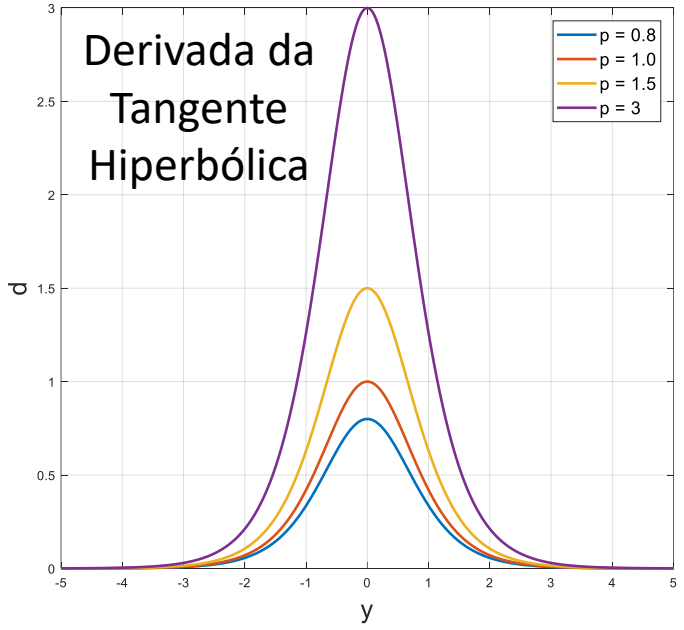
- Sua derivada é dada por

$$\frac{dy_j}{dz_j} = \frac{df(z_j)}{dz_j} = p\left(1 - \tanh^2(pz_j)\right) > 0,$$

onde mais uma vez, o parâmetro p controla a suavidade da função. Essa função e sua derivada são mostradas nas figuras abaixo.

OBS.: Quanto maior p, mais próxima ela fica da função degrau. Porém, normalmente, se usa $p = 1$.

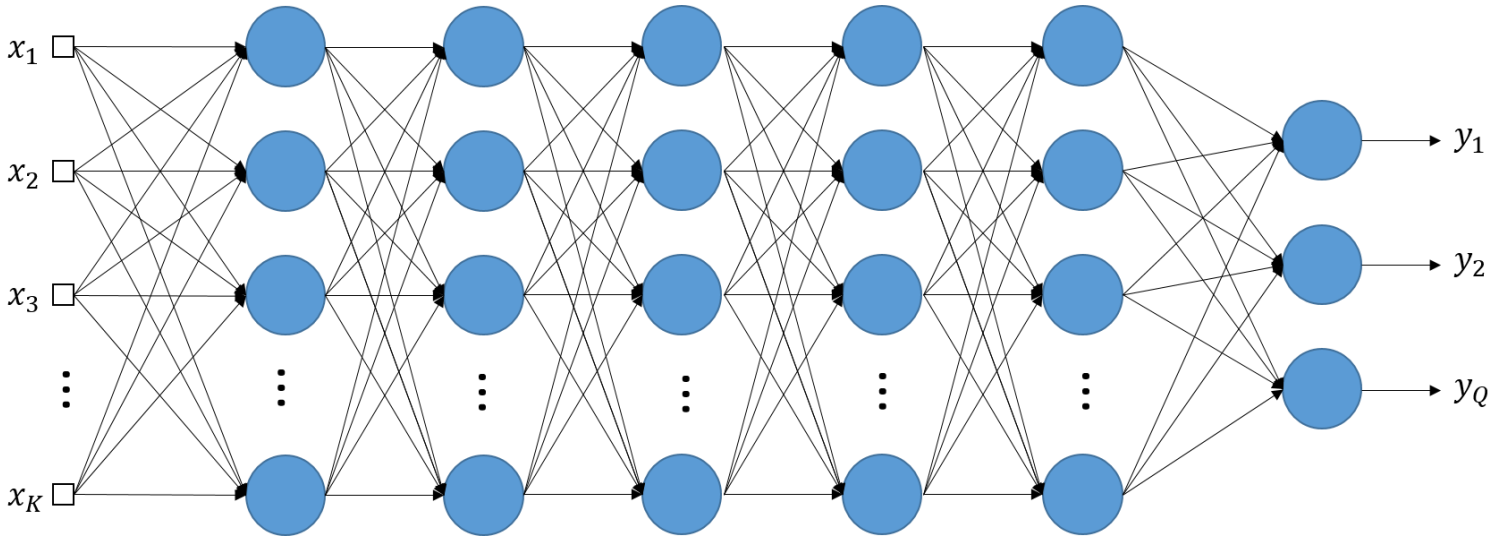




OBS.: tende ao impulso conforme p aumenta.

O Problema da Dissipação do Gradiente

- É um problema encontrado quando treinamos ***redes neurais profundas***, ou seja, com muitas camadas escondidas, com métodos de aprendizagem baseados em informações do gradiente e funções de ativação sigmóide.
- Ocorre devido à natureza do ***algoritmo de retropropagação*** usado para treinar a rede neural.



O Problema da Dissipação do Gradiente

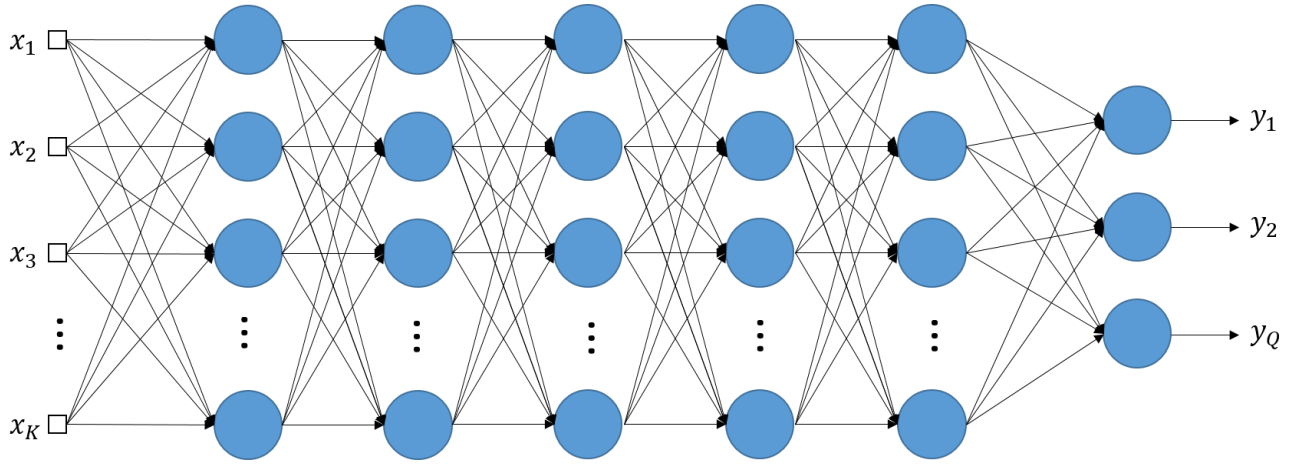
- Lembrem-se que as ***funções de ativação*** como ***tangente hiperbólica*** ou ***logística***, têm gradientes (i.e., derivadas parciais) no intervalo de 0 até aproximadamente 1.
- Durante o treinamento, para atualizar os pesos de cada camada da ***rede neural***, o ***algoritmo de retropropagação*** calcula os gradientes através da ***regra da cadeia***.

$$\frac{\partial f\big(g(h(x))\big)}{\partial x} = \frac{\partial f\big(g(h(x))\big)}{\partial g(h(x))}\frac{\partial g(h(x))}{\partial h(x)}\frac{\partial h(x)}{\partial x}$$

- Em outras palavras, a derivada de uma função de ativação em uma dada camada da rede neural torna-se o produto das derivadas das funções de ativação no caminho desde a camada final até a camada atual.
- Ou seja, no caminho inverso, da camada de saída para a camada de atual.

O Problema da Dissipação do Gradiente

- Isso tem o efeito de multiplicar ***M*** desses pequenos valores para calcular os gradientes das primeiras camadas em uma rede com ***M*** camadas.
- O que significa que o gradiente (i.e., o erro propagado) diminui exponencialmente com ***M***.
- Isso significa que os nós das camadas iniciais aprendem muito mais lentamente do que os nós das camadas finais, pois o valor do gradiente é muito pequeno, fazendo com que a atualização dos pesos também seja pequena (i.e., lenta).



Funções de ativação

- Com o surgimento das ***redes neurais profundas***, uma outra função, conhecida como ***função retificadora***, passou a ser a bastante utilizada por questões ***numéricas*** e ***computacionais***.
- A ***função retificadora*** tem sua expressão dada por

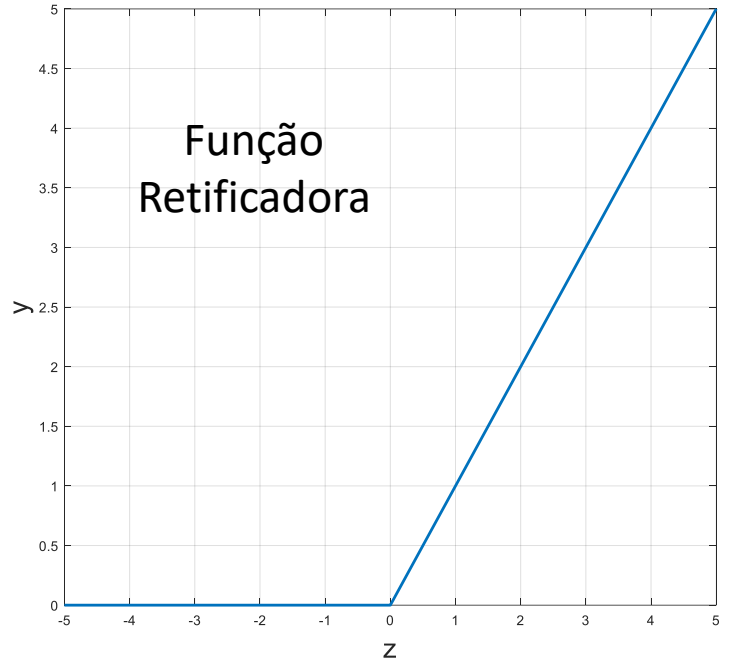
$$y_j = f(z_j) = \max(0, z_j).$$

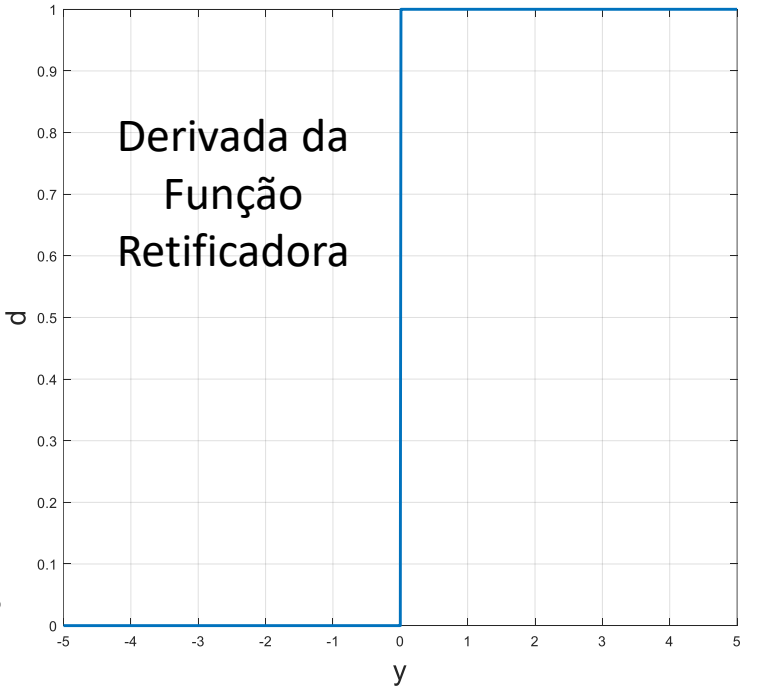
- Sua derivada é dada por

$$\frac{dy_j}{dz_j} = \frac{df(z_j)}{dz_j} = \begin{cases} 0, \text{se } y_j < 0 \\ 1, \text{se } y_j > 0' \end{cases}$$

e é indefinida para $y_j = 0$, porém o valor da derivada em zero pode ser arbitrariamente escolhido como 0 ou 1.

- Um ***nó*** que emprega uma ***função de ativação retificadora*** é chamado de ***rectified linear unit*** (ReLU)
- A ***função retificadora*** e sua derivada são mostradas nas figuras ao lado.





Funções de ativação

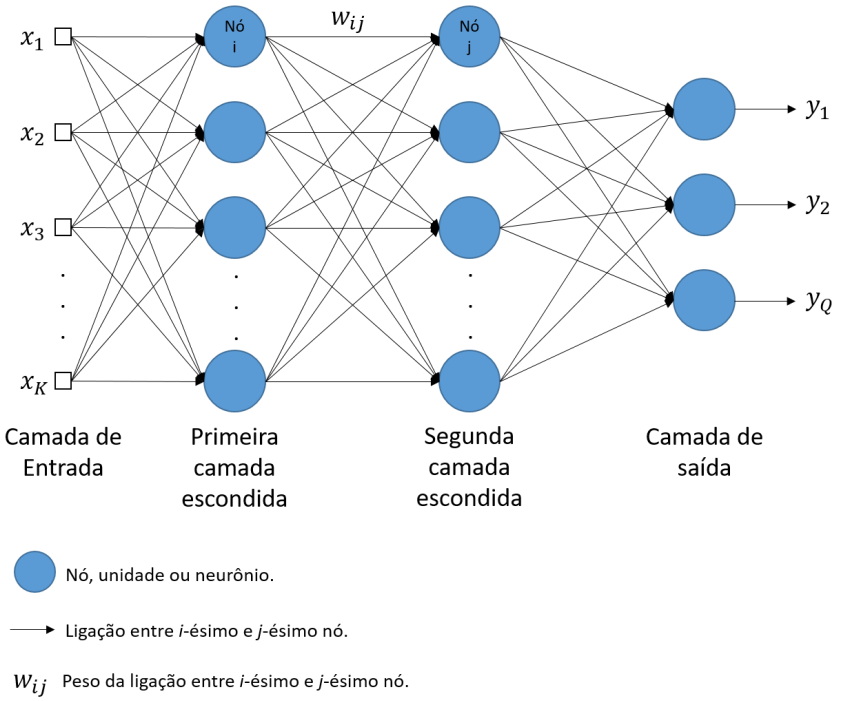
- Vantagens da ***função retificadora***:
  - A função e sua derivada são **mais rápidas de se calcular** do que as funções sigmóide e tangente hiperbólica.
  - Não sofre com o **problema da dissipação do gradiente** pois seu gradiente é igual a 0 ou 1. Mesmo se multiplicarmos vários gradientes de várias camadas, não haverá diminuição do seu valor.
- Outras funções de ativação são:
  - Identidade ou linear.
  - Gaussian Error Linear Unit (GELU).
  - Leaky rectified linear unit (Leaky ReLU).
  - Gaussiana.
  - https://en.wikipedia.org/wiki/Activation_function#Sign_equivalence_to_identity_function

Tarefa

- **Quiz**: “*T320 - Quiz – Redes Neurais Artificiais (Parte III)*” que se encontra no MS Teams.

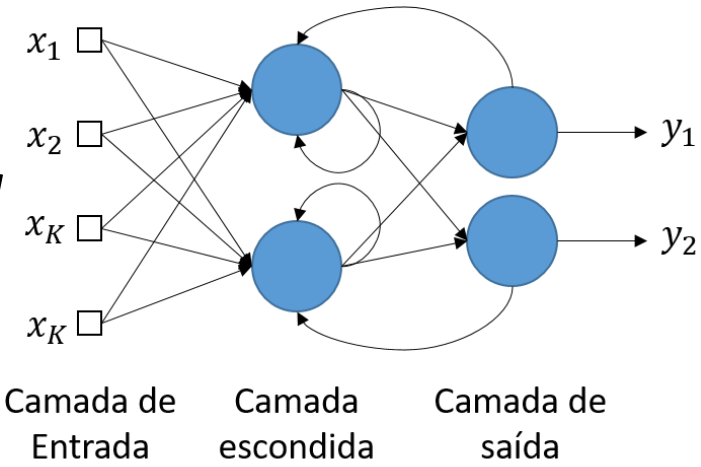
Conectando Neurônios

- Existem basicamente duas maneiras distintas para se conectar os ***nós*** de uma rede.
- Na figura ao lado, os ***nós*** da rede têm conexões em apenas uma única direção.
- Esse tipo de rede é conhecida como ***rede de alimentação direta*** (*feedforward*) ou ***sem realimentação***.
- O sinal percorre a rede em uma única direção, da entrada para a saída.
- Os ***nós*** da mesma camada não são conectados.
- Esse tipo de rede representa uma ***função de suas entradas atuais*** e, portanto, não possui um estado interno além dos próprios pesos.

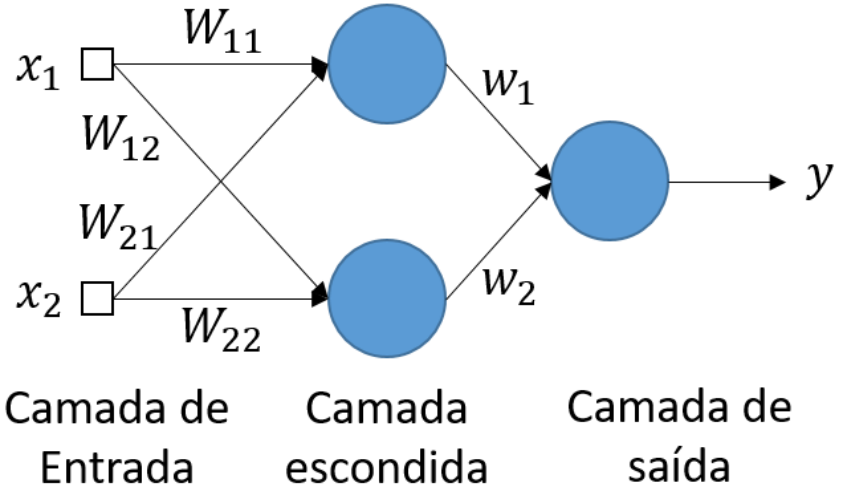


Conectando Neurônios

- Na figura ao lado, os ***nós*** da rede tem conexões em 2 direções, desta forma, o sinal percorre a rede nas direções direta e reversa.
- Este tipo de rede é conhecida como ***rede recorrente*** ou ***rede com realimentação***.
- Nessas redes, a saída de alguns ***nós*** alimentam ***nós*** da mesma camada (inclusive o próprio ***nó***) ou de camadas anteriores.
- Isso significa que os níveis de ativação da rede formam um ***sistema dinâmico*** que pode atingir um estado estável, exibir oscilações ou mesmo um comportamento caótico, ou seja, divergir.
- Além disso, a resposta da rede a uma determinada entrada depende do seu estado inicial, que pode depender das entradas anteriores.
- Portanto, ***redes recorrentes*** podem suportar memória de curto prazo.
- Essas redes são úteis para o ***processamento de dados sequenciais***, como som, dados de séries temporais ou linguagem natural (escrita e fala).



Regressão Não-Linear



- A rede MLP ao lado tem sua saída definida por

$$y = f(f(\boldsymbol{W}\boldsymbol{x})\boldsymbol{w}),$$

onde f é a ***função de ativação*** escolhida.

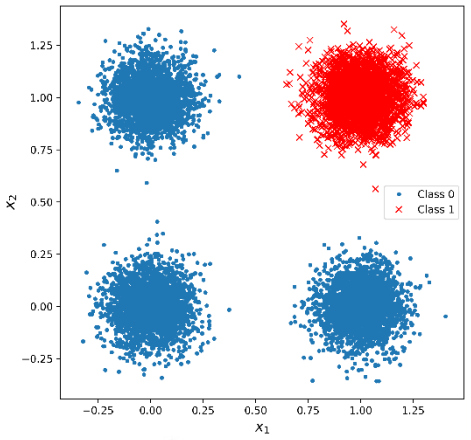
- Percebam que a saída da rede é dada pelo ***aninhamento*** das saídas de ***funções de ativação não-lineares***.
- Sendo assim, as funções que uma rede neural pode representar podem ser ***altamente não-lineares*** dependendo da quantidade de camadas e nós.
- Portanto, redes neurais podem ser vistas como ferramentas para a realização de ***regressão não-linear***, mas também podemos resolver outros problemas como os de classificação.
- Com uma única camada oculta suficientemente grande, é possível representar qualquer função contínua das entradas com uma precisão arbitrária.
- Com duas camadas ocultas, até funções descontínuas podem ser representadas.
- Portanto, dizemos que as redes neurais possuem ***capacidade de aproximação universal*** de funções.
- Veremos alguns exemplos a seguir desta capacidade de aproximação.

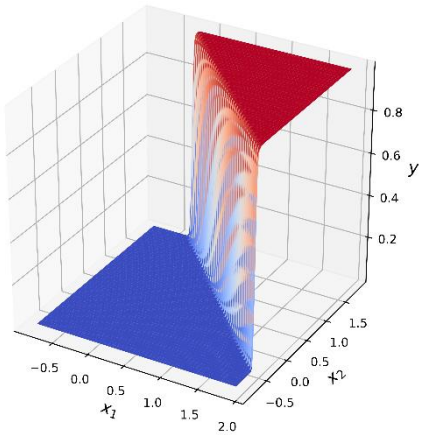
Aproximação universal de funções

- Um nó aproxima uma função de limiar suave.
- Combinando duas funções de limiar suave com direções opostas, podemos obter uma função em formato de onda.
- Combinando duas ondas perpendiculares, nós obtemos uma função em formato cilíndrico.

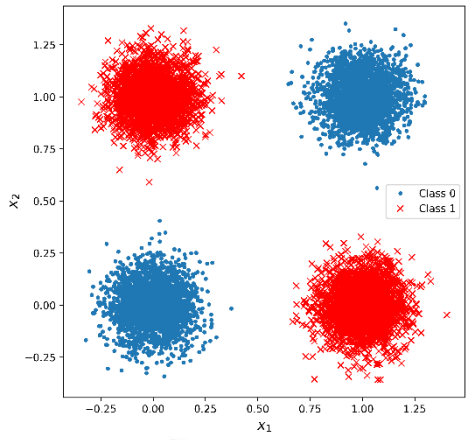
[Exemplo: FunctionApproximationWithMLP.ipynb]

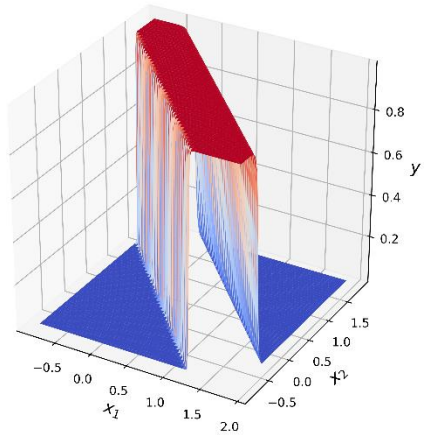
Função AND: MLP com 0 camadas escondidas, apenas um neurônio na camada de saída. Total: 1 nó.



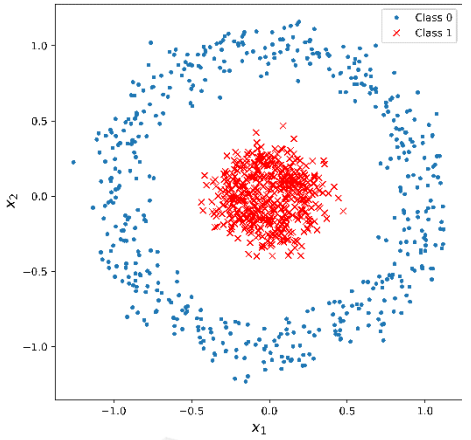


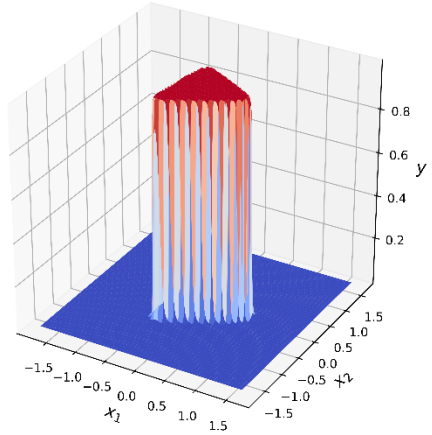
Função XOR: MLP com 1 camada escondida com 2 nós. Total: 3 nós.





Círculos concêntricos: MLP com 1 camada escondida com 4 nós. Total: 5 nós.



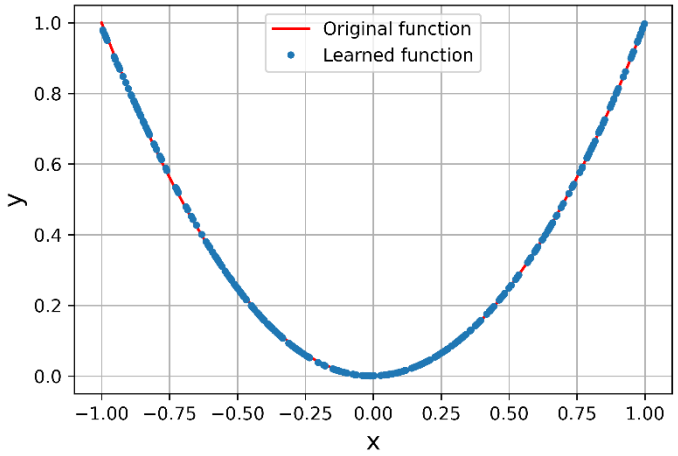


Aproximação universal de funções

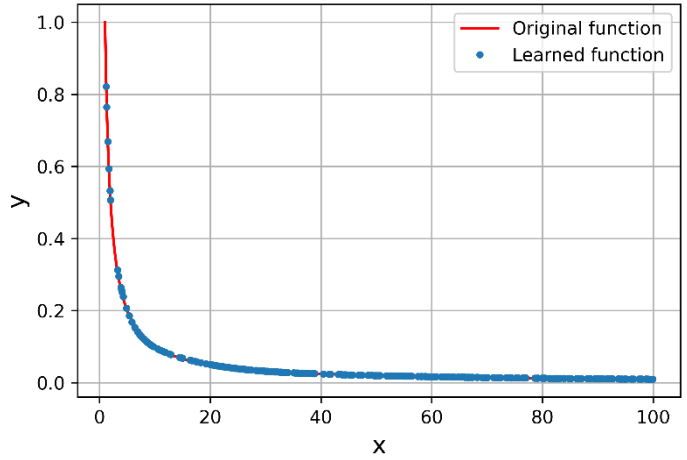
- Redes neurais podem ser usadas para aproximar funções como as mostradas abaixo:
  - $f(x) = x^2, -1 \leq x \leq 1,$
  - $f(x) = \frac{1}{x}, 1 \leq x \leq 100,$
  - $f(x) = \sin(x), 1 \leq x \leq 2\pi.$

[Exemplo: function_approximation.ipynb]

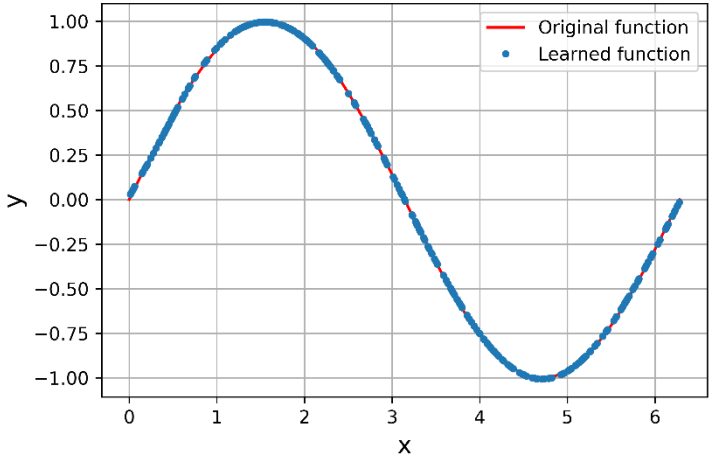
$f(x) = x^2$



$f(x) = \frac{1}{x}$



$f(x) = \sin(x)$

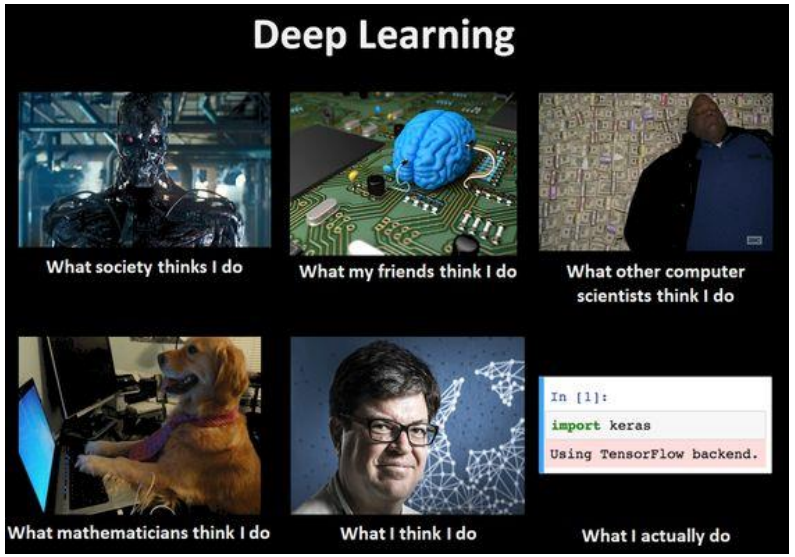


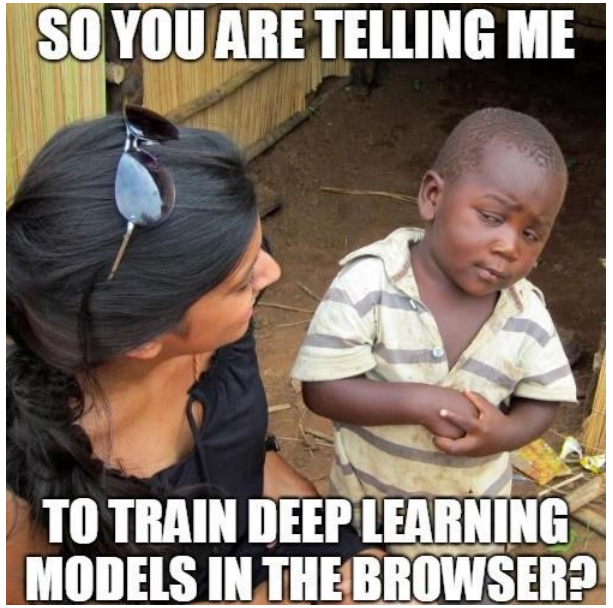
Tarefas

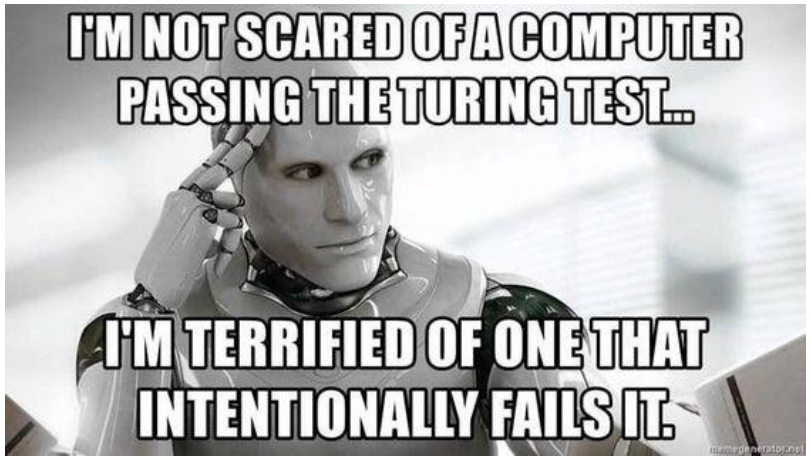
- **Quiz**: “*T320 - Quiz – Redes Neurais Artificiais (Parte IV)*” que se encontra no MS Teams.
- **Exercício Prático**: **Laboratório #7**.
  - Pode ser baixado do MS Teams ou do GitHub.
  - Pode ser respondido através do link acima (na nuvem) ou localmente.
  - Instruções para resolução e entrega dos laboratórios.
  - **Laboratórios podem ser feitos em grupo, mas as entregas devem ser individuais.**

Obrigado!

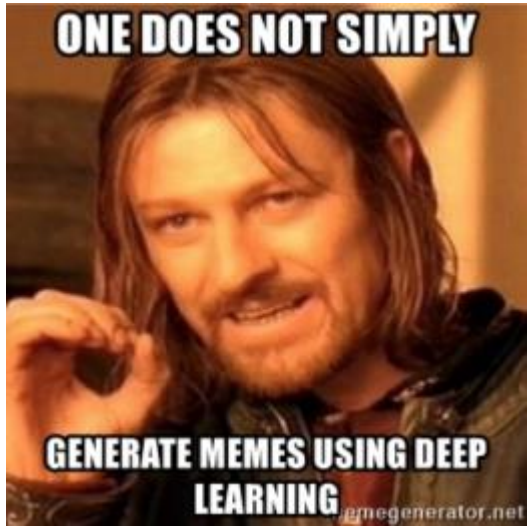

People with no idea about AI, telling me my AI will destroy the world
Me wondering why my neural network is classifying a cat as a dog..
Dog


Deep Learning
What society thinks I do
What my friends think I do
What other computer scientists think I do
What mathematicians think I do
What I think I do
In [1]:
import keras
Using TensorFlow backend.
What I actually do


SO YOU ARE TELLING ME
TO TRAIN DEEP LEARNING MODELS IN THE BROWSER?


I'M NOT SCARED OF A COMPUTER PASSING THE TURING TEST...
I'M TERRIFIED OF ONE THAT INTENTIONALLY FAILS IT.


Dog
I NEED GPU FOR MY DUMB NEURAL NETWORK


ONE DOES NOT SIMPLY
GENERATE MEMES USING DEEP LEARNING
memegenerator.net

Figuras