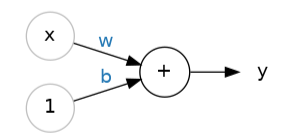
**​O QUE É DEEP LEARNING?**

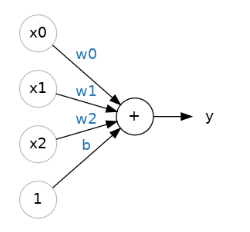
É um tipo de aprendizado de máquina caracterizado por camadas profundas de cálculos matemáticos. Sendo assim, tem mais facilidade de encontrar padrões escondidos nos problemas do mundo real. Uma rede neural é semelhante ao cérebro humano, tendo vários neurônios, cada um executando seu próprio cálculo matemático e se conectando com outros neurônios para formar a rede neural completa.

**A UNIDADE FUNDAMENTAL DE UMA REDE NEURAL: O NEURÔNIO**



Basicamente ele vai pegar as suas entradas multiplicadas pelos seus respectivos pesos e realizar uma operação entre eles, nesse caso acima, X é uma entrada, W é um peso e B é um tipo especial de peso que representa o viés de aprendizado. No caso desse neurônio, a saída dele seria: y = x\*w + b, ou seja, uma função linear.

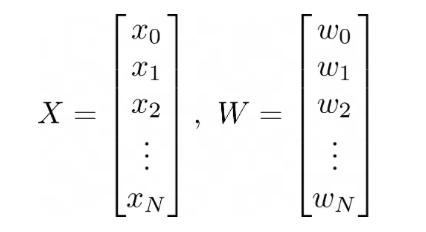
No mundo real, sabemos que um modelo não tem apenas uma entrada, ele possui várias ou milhões, para isso, temos que fazer com que o neurônio suporte múltiplas entradas:



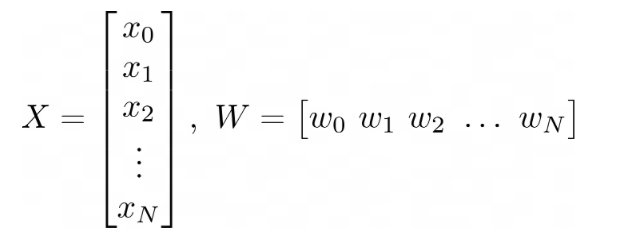


Cada entrada representa uma feature e cada peso representa um valor que determina como e quanto a feature vai impactar no resultado final, a depender da sua multiplicação pelo peso.

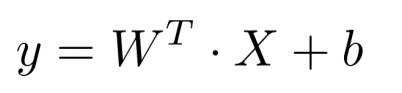
Representação em forma de matriz(X são as features e W são os pesos):



Vindo da álgebra linear, sabemos que só é possível efetuar uma operação entre matrizes quando o número de colunas da primeira é igual ao número de linhas da segunda. Então temos que usar a transposta na segunda(pesos):

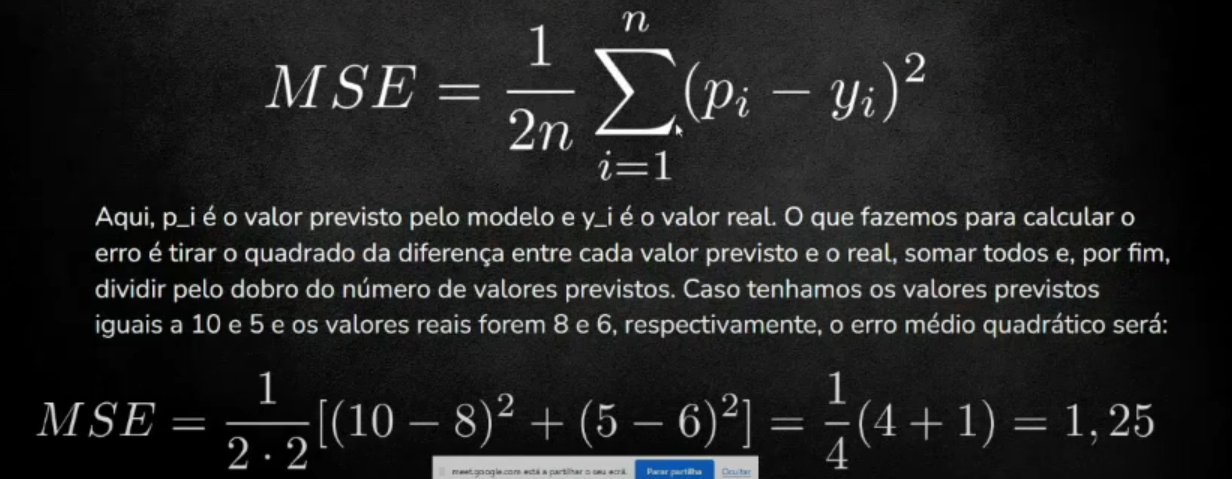


Como queremos um escalar como resultado, que seria a saída do neurônio, podemos apenas multiplicar as novas matrizes e somar com o viés de aprendizado, gerando um escalar como saída:

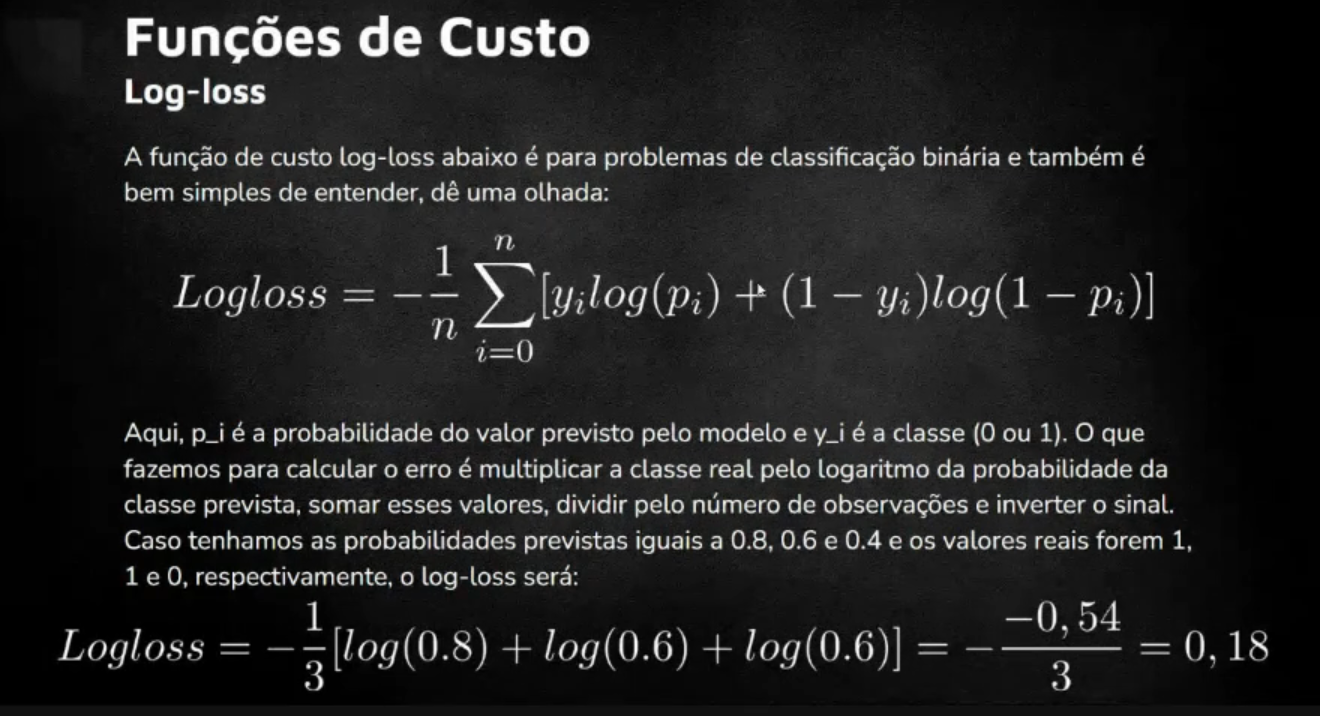


**Gradiente Descendente**

**Função de custo:** É a forma de quantificar e qualificar o erro, é a forma usada para medir o erro, como MSE, acurácia, recall, entre outras.

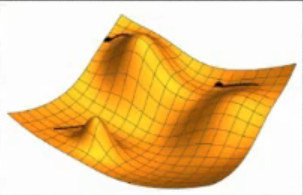


Log-loss utiliza uma função logarítmica baseada no valor real e na probabilidade de ter previsto a classe que foi prevista, primeiro vamos multiplicar a classe real pelo log da probabilidade da previsão feita, e depois faremos a mesma coisa só que invertendo os sinais.



**Definição Gradiente descendente**

Basicamente é uma forma de otimização em machine learning, ou seja, determinar os melhores parâmetros de forma a minimizar ao máximo a Loss Function.

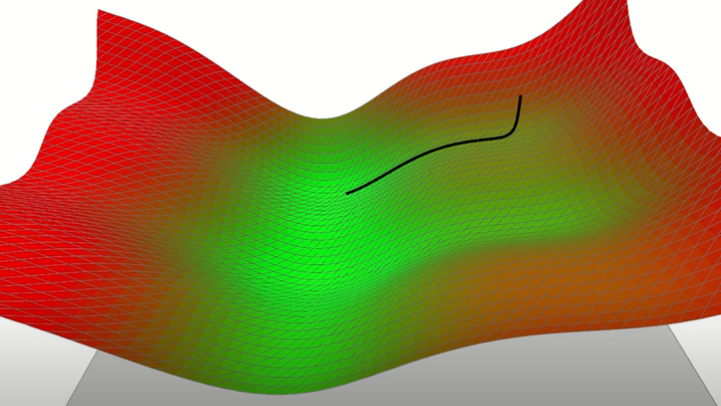


Essa malha laranja representa o resultado da Loss Function com base em todas as combinações de parâmetros possíveis, quanto maior o pico, maior a taxa de erro, e quanto menor o pico, menor o erro. Nosso objetivo com o gradiente descendente é achar o ponto mínimo global, que é o menor ponto de todo o espaço. O mínimo local é o ponto mínimo em uma determinada região do espaço.

O funcionamento básico do algoritmo é que em cada iteração, ele vai procurar o caminho mais íngreme, como se quisessem descer uma montanha e sempre fossemos pelo caminho que descesse mais, ou seja, o mais íngreme.

O objetivo é chegar no mínimo global, mas antes disso, podemos passar por um mínimo local, quando acontece isso –Derivada = 0– a bolinha que achou um mínimo local vai parar(podendo voltar a procurar depois), enquanto as outras vão continuar normalmente.

[Gradient Descent 3D - Visualization](https://www.youtube.com/watch?v=kJgx2RcJKZY&ab_channel=ChristopherGondek)

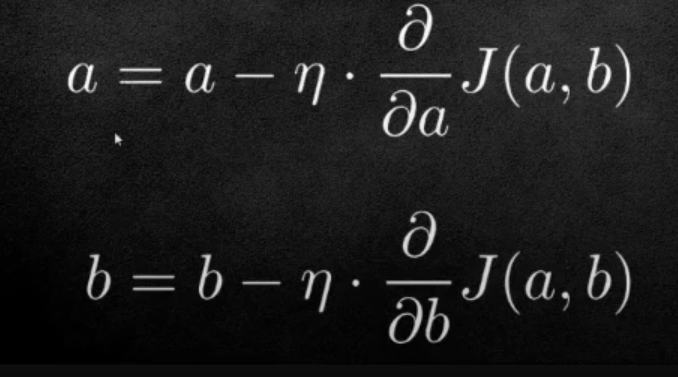


Essa função anterior é côncava, o que significa que temos que nos preocupar com os mínimos locais, em contrapartida, temos funções como as MAE e MSE que são convexas, ou seja, o único mínimo existente é o global.

Temos que escalonar as features antes de aplicar o gradiente descendente.

As funções de custos mais usadas em deep learning são côncavas, ou seja, tem mínimos locais que não são os melhores –o mínimo global–.

**Como é feita a atualização de parâmetros no algoritmo**

****

* **a,b:** Parâmetros que queremos otimizar com gradiente descendente
* **eta(n):** Learning rate: Determina o tamanho do passo de aprendizagem, se for muito pequeno, ele vai aprender(mudar os parâmetros) pouco por vez e se for maior vai aprender(mudar os parâmetros)r muito por vez(Geralmente determinamos o melhor eta por teste, como grid search e outros tipos)
* **del(∂):** Derivada parcial, ou seja ∂/∂a representa a derivada parcial da função em relação a a
* **J(a,b):** É a função de custo(loss function) que vamos utilizar, iremos deriva-las em relação a a e b