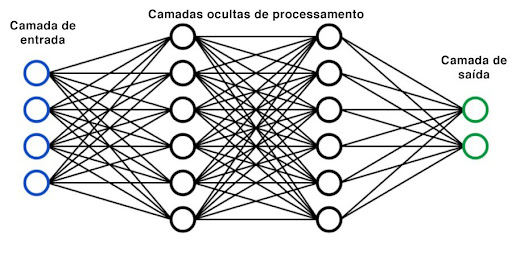
**Relatório 13 - Redes Neurais**

Guilherme Loan Schneider

**Descrição da atividade**

Funcionamento básico das Redes Neurais Artificiais

Resumidamente, uma rede neural artificial é um conjunto de neurônios (forma o pensamento da rede) utilizado para compreender um determinado conjunto de dados e com isso, tentar chegar o mais próximo de um raciocínio humano, mas com respostas mais rápidas e precisas.

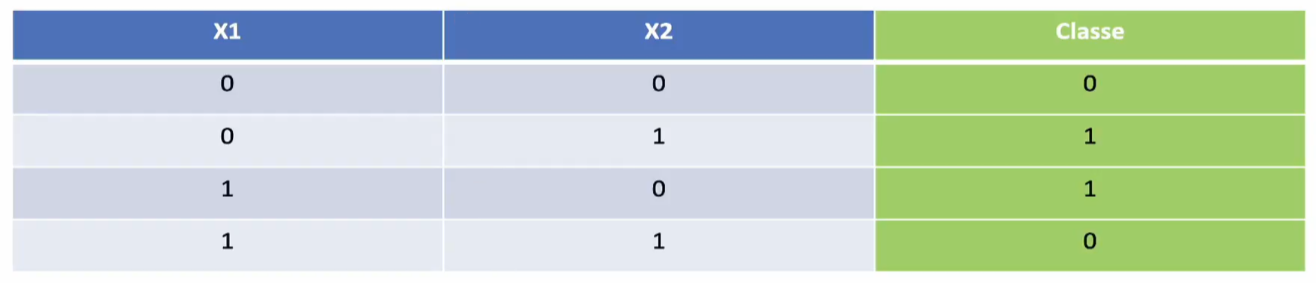


Uma Rede Neural Artificial possui os seguintes itens primordiais que irão decidir o desempenho de uma rede:

* Camada de Entrada – Ela é basicamente todas as features do seu conjunto de dados, como por exemplo, a base de dados “Iris”, que possui como features o tamanho e comprimento da sépala e tamanho e comprimento da pétala, totalizando 4 features.
* Camadas Ocultas – Essa camada é responsável por desempenhar o papel de “pensamento”, fazendo isso a partir de reajustes de pesos (linhas de um neurônio para os outros). Os pesos são fundamentais para que a rede consiga gerar uma boa decisão para uma determinada previsão. O tamanho dessa camada é calculado da seguinte forma (Numero de features + Numero de classes (camada de saída) / 2). No caso da base “Iris”, existem 4 features, e 3 saídas, logo, o cálculo resulta em 3.5, que pode ser arredondado para cima, totalizando 4. Logo, a camada oculta possuirá 4 neurônios.
* Camada de Saída – É a camada onde o resultado do pensamento da rede neural é esperado. Essa camada pode possuir desde apenas um neurônio, para problemas onde é feita a previsão de apenas uma classe, e para mais complexos (mais de duas classes), onde pode ser retornado, por exemplo, valores de previsão de venda de um produto em 3 localidades diferentes do mundo.

Aprofundando nas camadas

Para realizar o aprofundamento, utilizaremos a aplicação da tabela verdade do operador XOR (localizada abaixo), implementando-a na questão da rede neural.



Camada de Entrada

A camada de entrada é responsável por enviar os valores recebidos pelo usuário, ou seja, quando o usuário quiser requisitar uma previsão sobre algo, essa camada irá receber os valores, e enviá-los para a camada seguinte. Nas ligações entre cada neurônio (representados por segmentos de reta), existe os pesos definidos pela rede neural, que à medida que épocas são executadas, rebalanceamentos serão feitos.

*Os pesos inicias são definidos aleatoriamente pelo algoritmo.*

Camada Oculta (escondida)

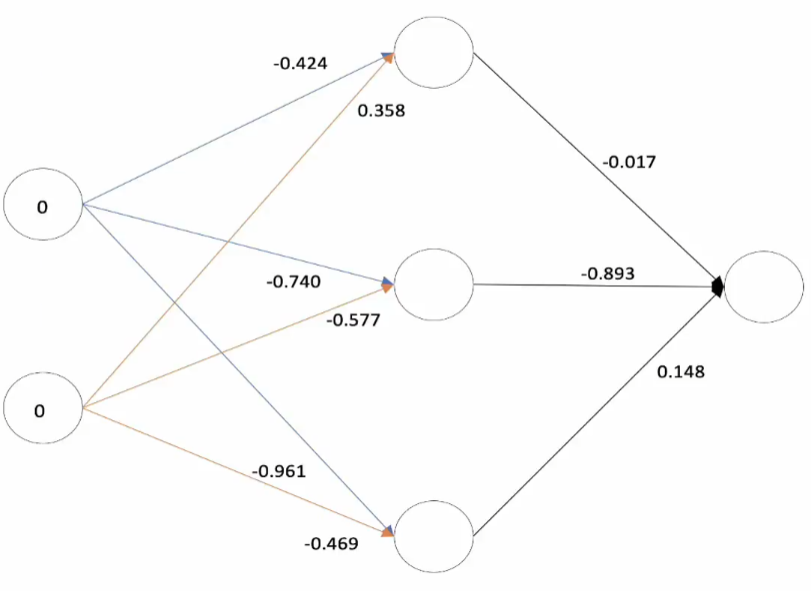
Quando a camada oculta recebe os valores da camada anterior, existem duas funções que devem ser calculadas por ela, a função de soma e a função de ativação.

A função de soma consiste em utilizar o valor passado pela camada de entrada e multiplica-lo pelo peso da aresta conectada ao neurônio atual. No exemplo abaixo, o cálculo seria realizado da seguinte forma para CADA neurônio na camada oculta.

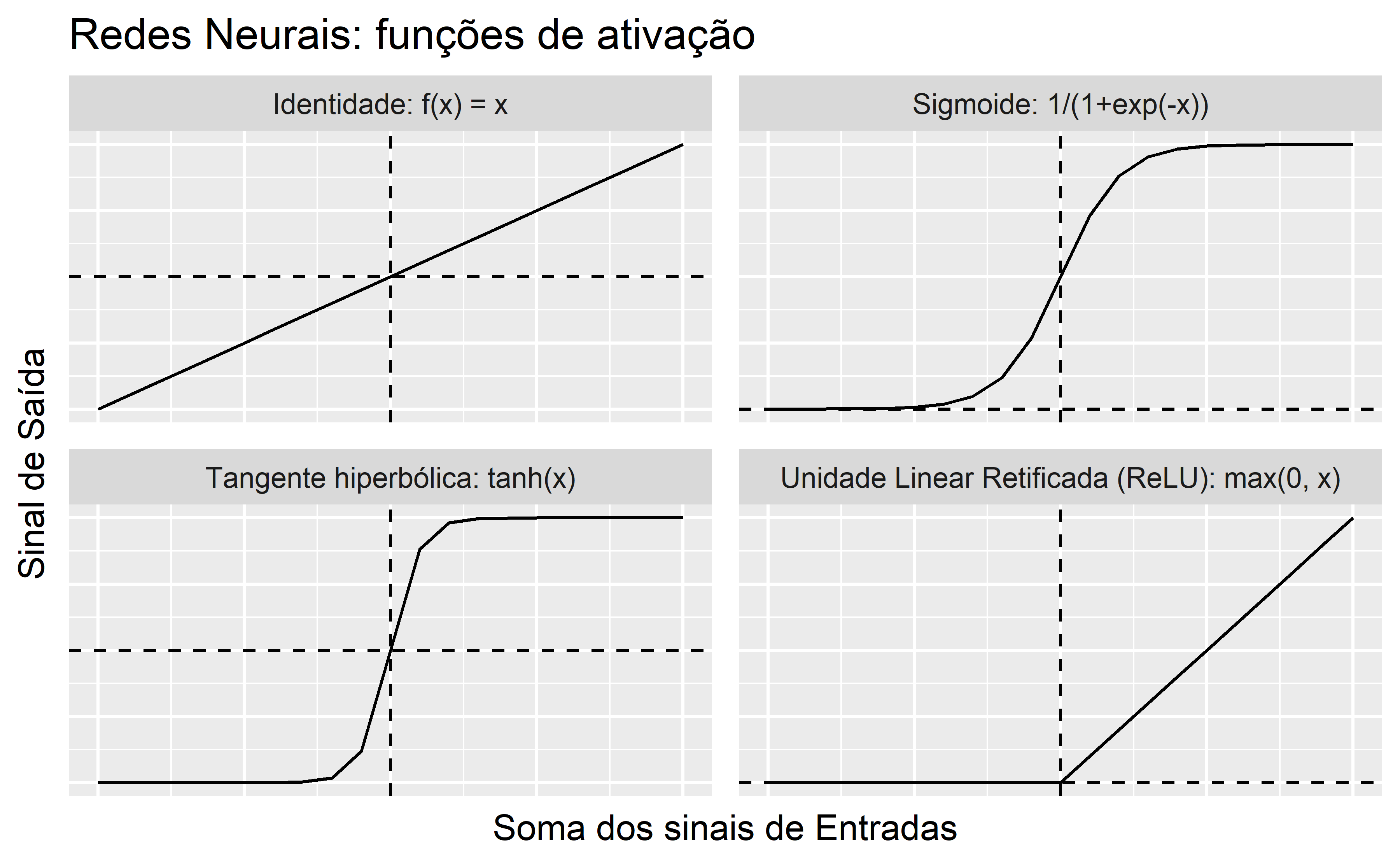
Como o primeiro neurônio possui duas ligações provenientes da camada de entrada, o cálculo é o seguinte: (0 \* -0.424 + 0 \* 0.358). O resultado é igual a 0.

O segundo neurônio é feito o mesmo cálculo, logo, (0 \* -0.740 + 0 \* -0.577). O resultado é igual a 0.

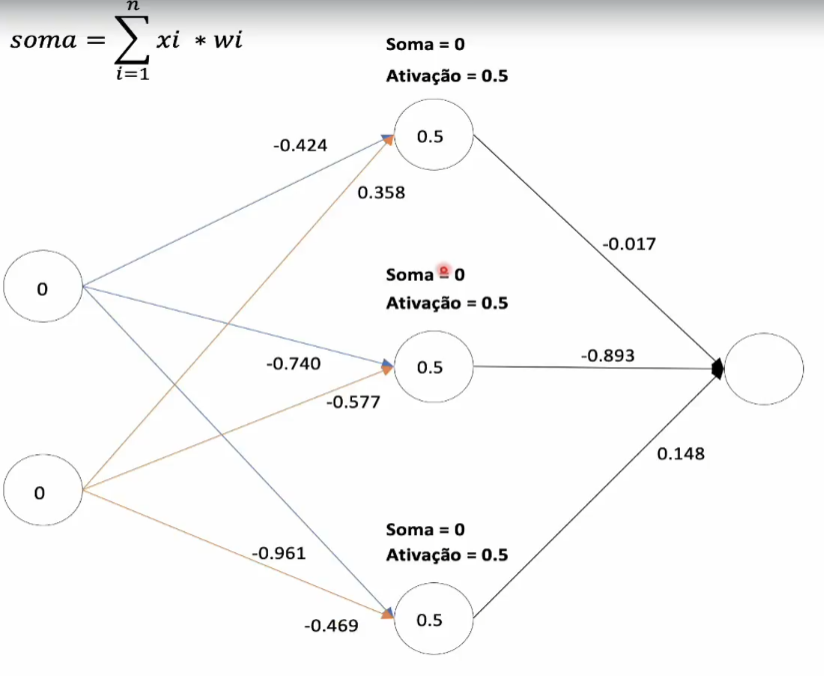
Para o terceiro neurônio, (0 \* - 0.961 + 0 \* -0.469). O resultado é igual a 0.



Após esses cálculos, deveremos aplicar a função de ativação ainda na camada oculta. Para esse problema, utilizaremos a função sigmoide, que funciona para quando deseja-se retornar a probabilidade de algo ser verdade, muito utilizado para problemas de classificação binária.



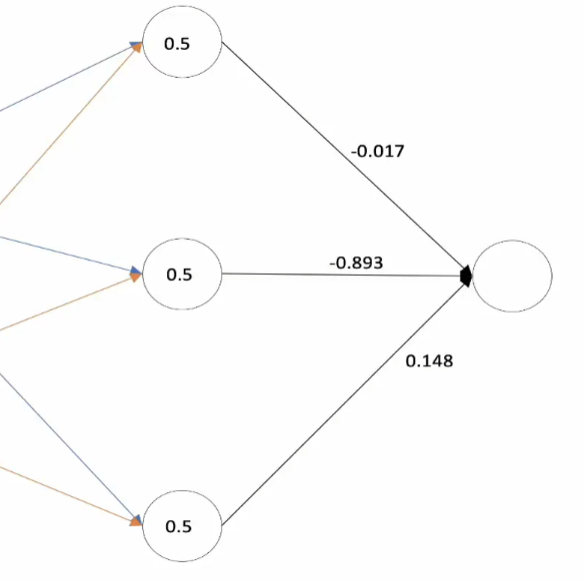
Aplicando os valores obtidos na função de soma, obtém-se o valor 0.5 para os três neurônios.



É importante salientar que é realizado esses cálculos para todas as entradas, então a (0,0), (0,1), (1,0), (1,1).

Camada de Saída

Nessa camada, o processo se repete, deverá ser aplicada a função de soma e de aplicação, utilizando os valores de ativação obtidos na camada anterior.

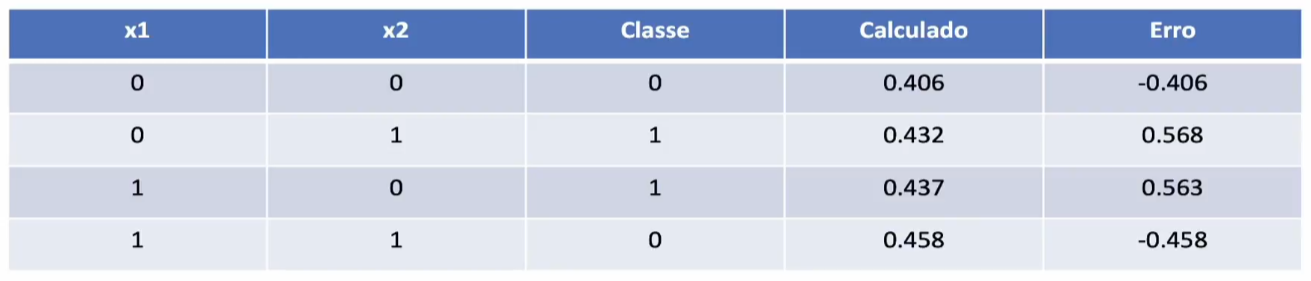


O cálculo da função de soma é o seguinte: (0.5 \* -0.017 + 0.5 \* - 0.893 + 0.5 \* 0.148). O resultado é igual a -0.381. Aplicando esse valor na função de ativação (função sigmoide), obtem-se o resultado igual a 0.406.

Nessa primeira iteração, é possível verificar que o resultado obtido é igual a 0.406, muito diferente no resultado esperado 0.

Cálculo do erro

O cálculo do erro, nesse exemplo mais simples, é calculado a partir do valor da resposta correta menos a resposta calculada.



Após isso, é calculada a média absoluta, que é o somatório em módulo, dos valores de erro obtidos (0.406 + 0.568 + 0.563 + 0.458), totalizando 0.498. O objetivo é sempre reduzir o valor do erro.

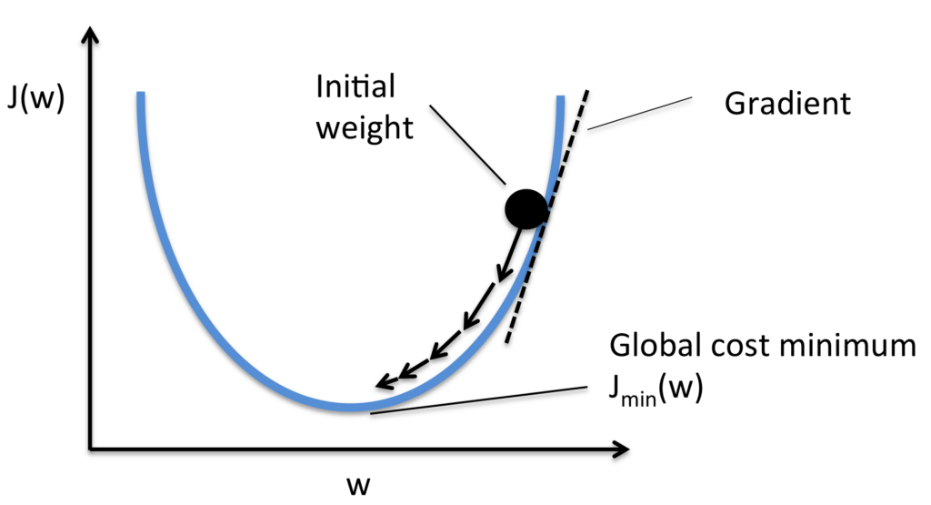
Reduzindo o valor do erro

Existem vários conjuntos de técnicas para tentar reduzir o valor do erro, abaixo estão algumas que foram utilizadas no desenvolvimento do curso:

Ajuste dos pesos: Utiliza o backpropagation para realizar os ajustes nos pesos da rede neural.

Cálculo do Erro: A rede realiza uma previsão, compara com o valor real e calcula o erro usando uma função de custo (ex: erro quadrático médio ou entropia cruzada).

Descida do Gradiente: Utiliza o gradiente da função de custo em relação aos pesos para determinar a direção e a intensidade da atualização necessária para minimizar o erro.



Descida do Gradiente Estocástico (SGD - Stochastic Gradient Descent): Em vez de calcular o gradiente com todos os dados de treinamento (descida do gradiente batch), o SGD atualiza os pesos com base em um pequeno subconjunto (batch) aleatório, tornando o treinamento mais rápido e eficiente.

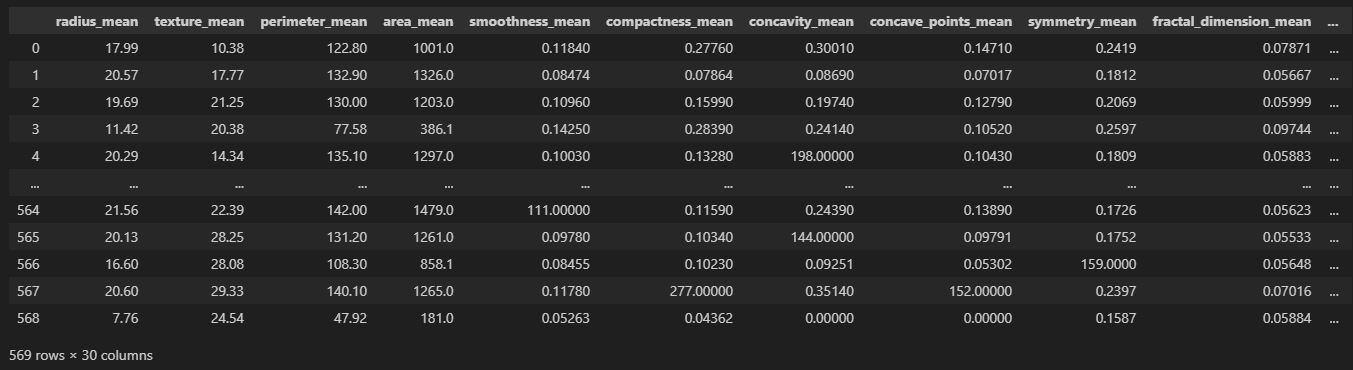
Cálculo do Parâmetro Delta: O delta representa a correção necessária para os pesos com base no erro propagado pela retropropagação (backpropagation). Ele é calculado a partir da derivada da função de ativação multiplicada pelo gradiente do erro.

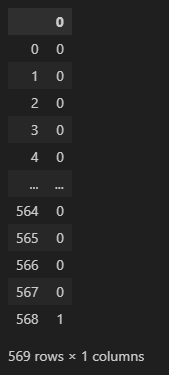
**Redes Neurais Artificiais na prática**

Criando uma rede neural a fim de ser treinada com os dados da base “Breast Cancer” e realizar previsões se um tumor é considerado benigno ou maligno.

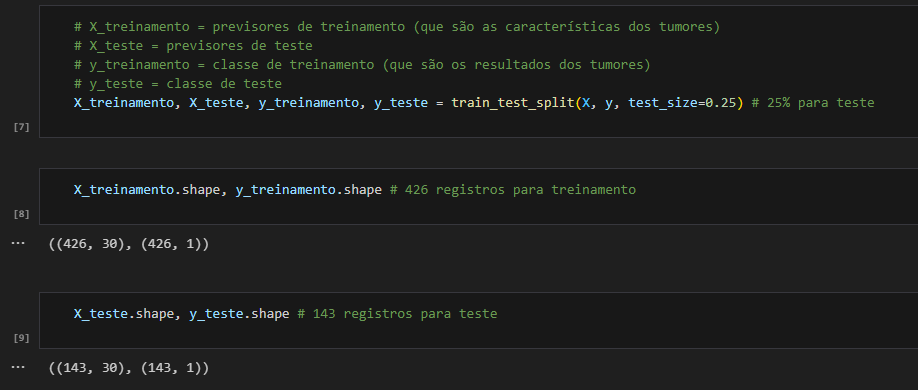
Nesse caso temos um problema de classificação binária (ou é benigno ou é maligno), logo, utilizaremos a função de ativação sigmoide.

A nossa base de dados possui 569 linhas e 31 colunas no total. Como ela está dividida em dois arquivos, o primeiro contendo 30 colunas, que são as nossas features, e o segundo contendo 1 coluna, com as respostas de cada tumor, classificando-o como benigno (0) ou maligno (1).

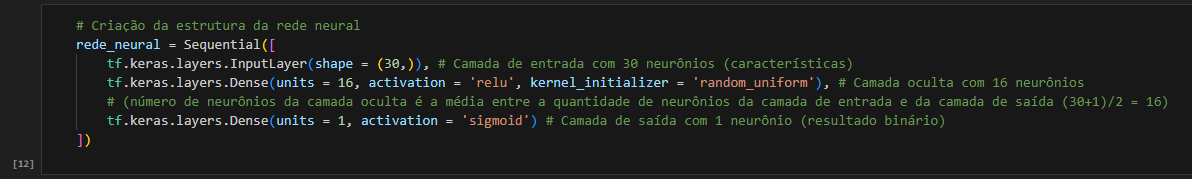




Nessa implementação, dividimos a base de dados no tipo TrainTest, 75% dos dados serão utilizados para treino, e 25% para teste e avaliação da rede neural.



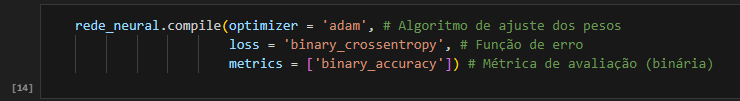
Em seguida, criamos a rede neural com 30 neurônios na camada de entrada, uma camada oculta com 16 neurônios, seguindo o cálculo para obter esse resultado, e a camada de saída com um único neurônio pois ele apenas retornará 0 ou 1.



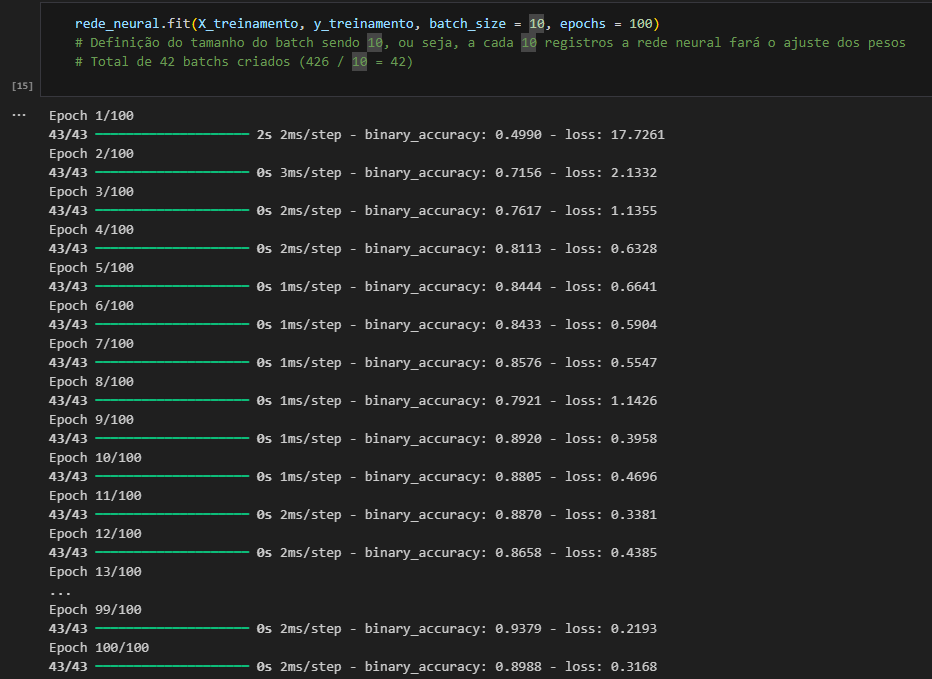
Na imagem abaixo é possível ver a camada oculta e o seu total de conexões. Totaliza 496 por conta de possui 30 neurônios na camada de entrada, e 16 na camada oculta (30 \* 16 = 480). Além desses neurônios, existem mais 16 neurônios de bias, um em cada neurônio da camada oculta, que farão com que mesmo que um peso seja zero, a rede ainda consiga compreender os padrões dos dados, totalizando 496.



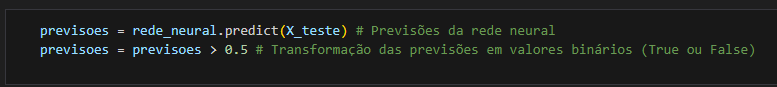
Em seguida, compilamos a rede neural com o algoritmo de ajuste dos pesos, nesse caso o Adam, juntamente com a função de erro sendo a Binary Cross Entropy (BCE), por fim a métrica de avaliação binária.



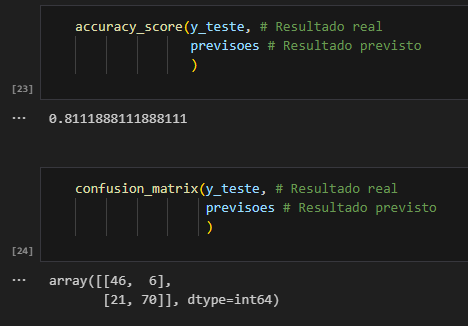
Então, executamos o algoritmo para efetuar, de fato, o treinamento da nossa rede neural. Aqui é definido o tamanho do batch como 10, então ele irá acessar 10 linhas aleatórias dos nossos dados de treinamento, e em seguida atualizar os pesos da rede. O algoritmo fará isso até não haver mais linhas na base de dados. Além disso, é definido o número de epochs, que indica quantas vezes a rede neural irá executar o algoritmo e rebalancear os pesos.



Após o termino do treinamento, passamos os dados de teste para o algoritmo fazer as predições, classificando as características passadas como um tumor benigno ou maligno. No código abaixo já existe também a transformação dos valores retornados em valores True ou False, como estamos utilizando a função sigmoide, podemos utilizar um valor 0.5 para transformar esses dados (esse valor depende do nível de confiança do algoritmo).

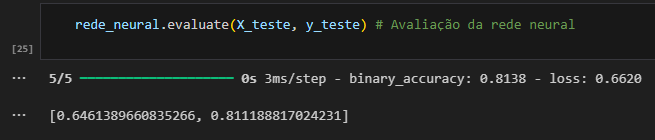


Por fim, podemos verificar a acurácia alcançada a partir da rede neural, juntamente com a matriz de confusão, que indica onde o algoritmo mais cometeu erros.



A rede neural acertou 46 tumores benignos e 70 tumores malignos, mas errou 21 tumores malignos (classificando-os como benignos) e 6 tumores benignos (classificando-os como malignos).

Por fim, temos a avaliação da rede neural, com uma acurácia de 81%.



**Referencias**

[Deep Learning com Python de A a Z – Seção 2 à 7](https://docs.google.com/document/d/1rAvJ0FiWZG5fgecj4VhHqDJcgQWh_O5CrmZCw812GGU/edit?tab=t.0#heading=h.p7j16a17rz31)

<https://www.tensorflow.org/tutorials?hl=pt-br>