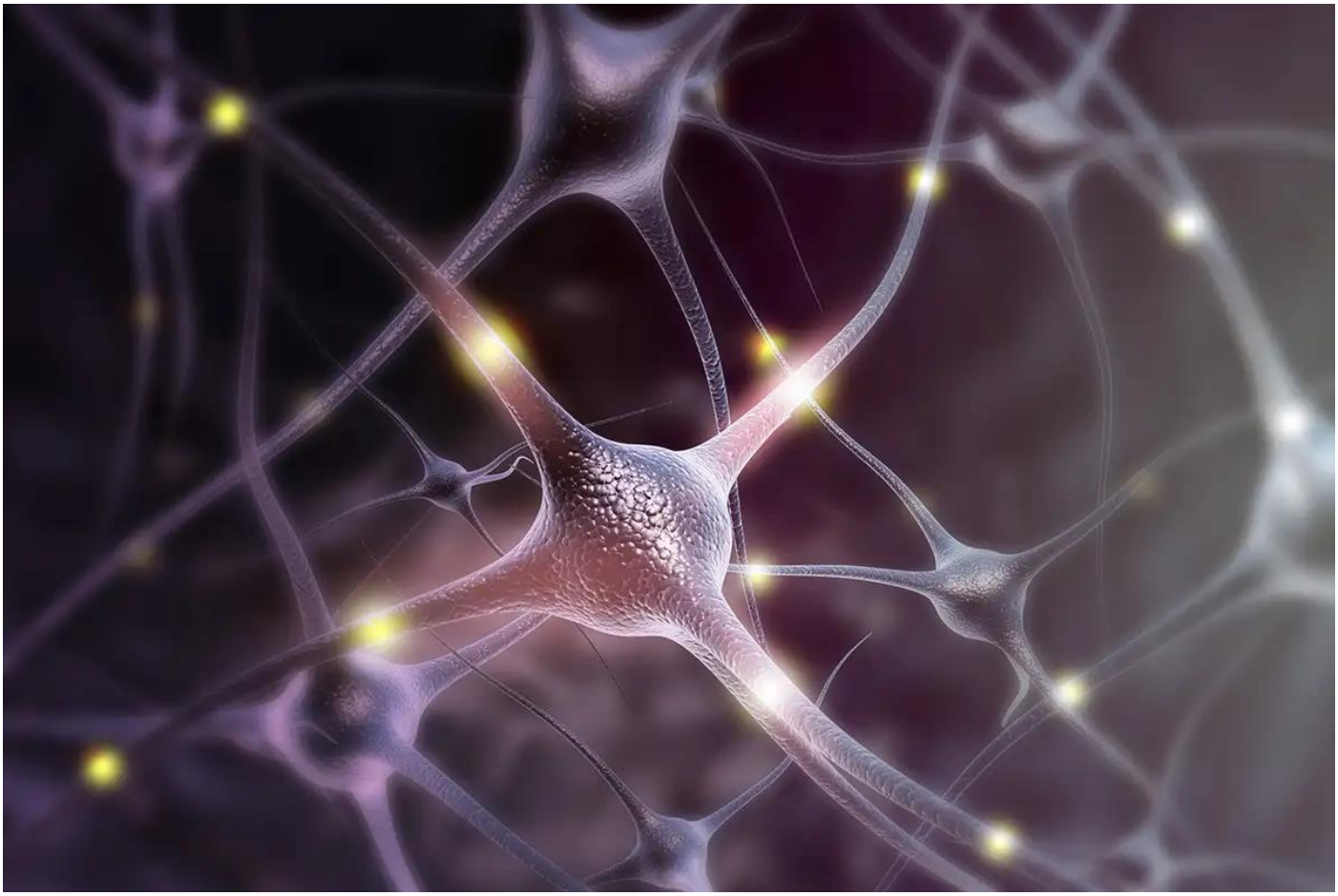
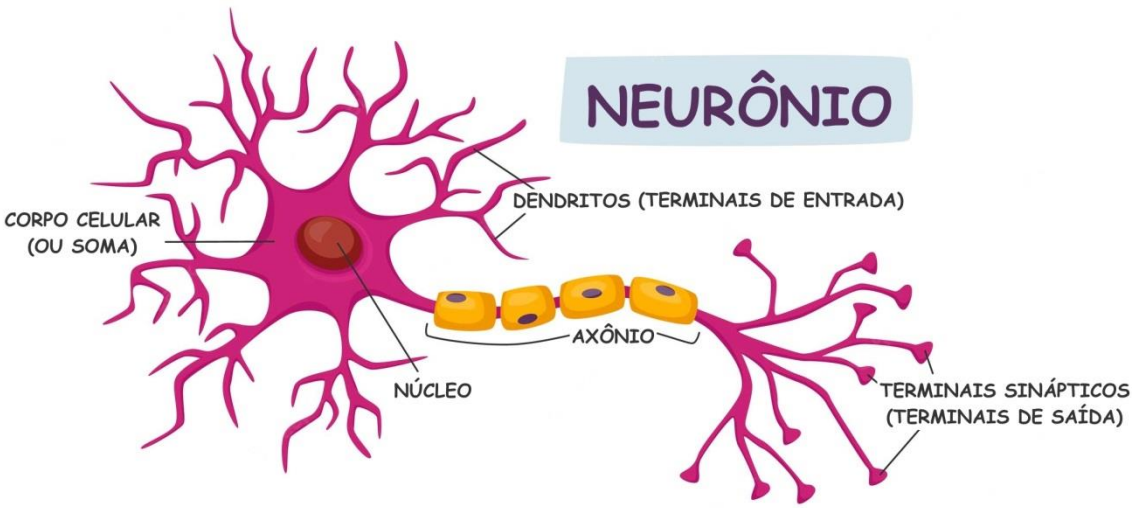
REDES NEURAIS BIOLÓGICAS E ARTIFICIAIS



Redes Neurais Biológicas



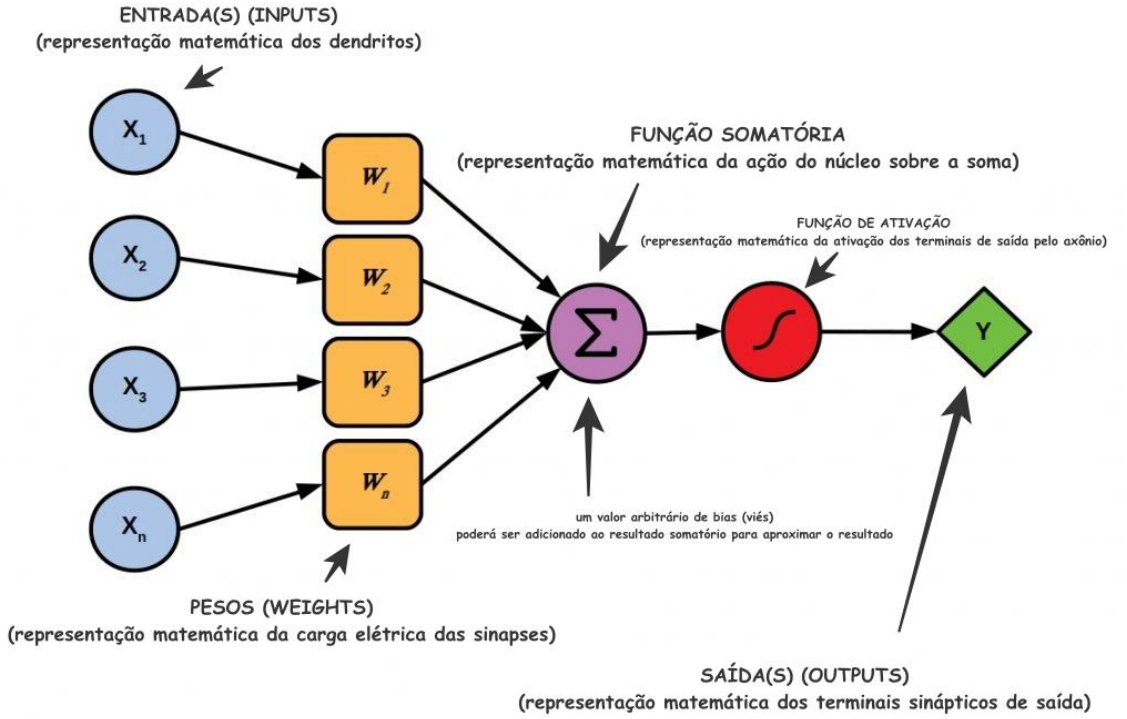
Os principais elementos que compõe um neurônio são:

- **Dendritos** – os dendritos são os terminais de entrada por onde o neurônio receptor recebe as informações vindas dos terminais sinápticos de saída do neurônio transmissor em forma de carga elétrica.
- **Corpo Celular** – o corpo celular ou soma como também é conhecido, é a matéria que envolve o núcleo. Ela recebe as cargas elétricas vindas de um ou mais dendritos e ao receber um sinal do núcleo transforma as múltiplas cargas recebidas em uma única carga elétrica sintetizada para que ela possa ser transmitida pelo canal do axônio por uma via única.
- **Núcleo** – o núcleo em conjunto com a soma sintetiza os sinais elétricos contidos no corpo celular somando as cargas e as transformando em um sinal de carga única por que o axônio só pode transmitir um sinal por vez para os seus terminais de saída.
- **Axônio** – o axônio transmite o sinal sintetizado para um ou mais terminais sinápticos que irão redistribuir a única carga elétrica recebida em múltiplos sinais a depender da quantidade de terminais sinápticos do neurônio. O número de terminais de entrada e saída poderá variar entre um neurônio e outro, de um a múltiplos terminais.
- **Terminais Sinápticos** – os terminais sinápticos são os terminais de saída por onde a carga elétrica vinda do axônio será transmitida para o próximo neurônio.

Para o evento que transmite o sinal elétrico dos terminais sinápticos de um neurônio para os dendritos de outro damos o nome de sinapse que também pode ser entendida como a região carregada eletricamente entre um neurônio e outro por onde passa a informação. Esse processo de transmissão de informação elétrica de um neurônio para outro poderá envolver até bilhões de neurônios comunicando-se entre si em uma fração de segundo. Essa interação entre os neurônios forma uma rede de informação conhecida como rede neural.

Resumo: os dendritos recebem a informação dos terminais sinápticos do neurônio anterior, transmitem a informação para o corpo celular que com a ajuda do núcleo sintetiza a informação que é passada para os terminais sinápticos através do axônio e reenviada para os dendritos do próximo neurônio.

Redes Neurais Artificiais



As redes neurais artificiais são abstrações matemáticas da arquitetura neuronal biológica e possuem uma representação matemática para cada um dos principais elementos biológicos do neurônio.

- **Entradas** – as entradas ou “inputs” são variáveis únicas ou vetoriais que representam os dendritos. São elas os valores para os quais queremos uma ou mais respostas.
- **Pesos** – os pesos ou “weights” são a representação da conversão das informações em sinais elétricos que serão sintetizados por uma função somatória que representa a soma do neurônio biológico. Os pesos são inicializados com números aleatórios entre 0 e 1. Os valores de entrada então serão multiplicados pelos pesos que irão transferir o resultado para a função somatória.
- **Função Somatória** – a função somatória é a representação matemática da ação do núcleo sobre o corpo celular (soma). Ela irá somar todos os resultados multiplicativos das entradas pelos pesos transformando os resultados multiplicativos em um único número. Um número arbitrário de viés poderá ser adicionado ao resultado somatório caso haja a necessidade de aproximar o valor.
- **Função de Ativação** – a função de ativação é a representação matemática do axônio que só pode receber uma única carga, aqui um único número resultante da função somatória. A função de ativação irá formatar o resultado somatório dentro de um intervalo específico, geralmente entre 0 e 1 e irá ativar a saída transferindo o resultado para uma ou mais variáveis de resposta.
- **Saídas** – as saídas ou “outputs” são variáveis únicas ou vetoriais que representam os terminais sinápticos de saída. Se o resultado recebido estiver longe do resultado esperado, esse resultado será reenviado para a entrada representando a transmissão de informação elétrica de um neurônio para outro e os pesos serão recalculados por um método matemático conhecido como descida gradiente que irá diminuir ou aumentar os pesos para que assim, o produto da nova entrada pelos pesos resulte em um valor mais próximo do ideal. A descida gradiente recebe esse nome por que o erro diminui gradativamente a cada novo cálculo.

- **Backpropagation** – o “backpropagation” ou retro propagação é o nome dado para cada novo reenvio dos valores de saída para a entrada que será multiplicada por pesos cada vez mais precisos conforme a quantidade de backpropagations aumenta. Para o número referente à quantidade de backpropagations damos o nome de épocas de treinamento.

Quando as saídas de um neurônio artificial são enviadas para as entradas de outro surge uma rede de informação conhecida como rede neural artificial.

Resumo: os valores de entrada são multiplicados pelos pesos, os produtos resultantes são enviados para a função somatória que irá somar todos os produtos e enviá-los para a função de ativação que irá formata-los e enviá-los para a saída. Se a saída não for a que desejamos o processo se repete com novos pesos corrigidos até que o resultado de saída seja um resultado satisfatório. Quando o resultado satisfatório for obtido os pesos do último backpropagation serão salvos para que em execuções futuras o algoritmo não precise mais de backpropagations emitindo assim resultados em uma única etapa.

Por exemplo, suponhamos que temos duas entradas para os dendritos artificiais (inputs) compostas por dois elementos numéricos de um vetor **[2, 3]** e queremos que a nossa rede neural artificial nos devolva uma única saída **0.08** pelos terminais sinápticos artificiais. Se os pesos aleatórios iniciais para cada entrada forem **[0.5, 0.5]** teremos a seguinte estrutura de cálculo...

Multiplicação das entradas pelos pesos: **[2, 3] x [0.5, 0.5] = [1, 1.5]**.

Este tipo de multiplicação é uma multiplicação conhecida como esparsa por que multiplicamos cada entrada por seu peso correspondente, mas também existem arquiteturas de conexão do tipo densa onde aplicamos uma multiplicação matricial para aumentar a variabilidade nos resultados.

Função somatória: **1 + 1.5 = 2.5**.

Função de ativação com intervalo entre 0 e 1: para formatar nosso resultado somatório no intervalo entre **0 e 1** podemos usar a fórmula da função sigmoide que é **1 / (1 + 2.71828 ^ 2.5) = 0.07585**, neste caso **2.71828** é uma constante. Observe que **0.07585** que é o nosso output matemático para o terminal sináptico está muito próximo de **0.08**, logo poderíamos considera-lo um resultado satisfatório e encerrar a execução. Caso contrário nós repetiríamos o processo com novos pesos até encontrar um resultado próximo do que estávamos esperando, cada nova repetição seria um backpropagation e o total de backpropagations necessários até encontrarmos o resultado seriam as nossas épocas.

Vejamos como ficaria esse backpropagation agora usando o valor da última saída como a entrada atual...

Como a nossa resposta foi ligeiramente inferior ao valor esperado os próximos pesos seriam ligeiramente ajustados para cima: **0.07585 x [0.8, 0.9] = [0.06068, 0.06826]**.

Função somatória: **0.06068 + 0.06826 = 0.12894**.

Observe que o resultado da função somatória é inferior ao que precisamos então neste caso usaremos o valor de bias de **2.27106**, e teremos **0.12894 + 2.27106 = 2.4**.

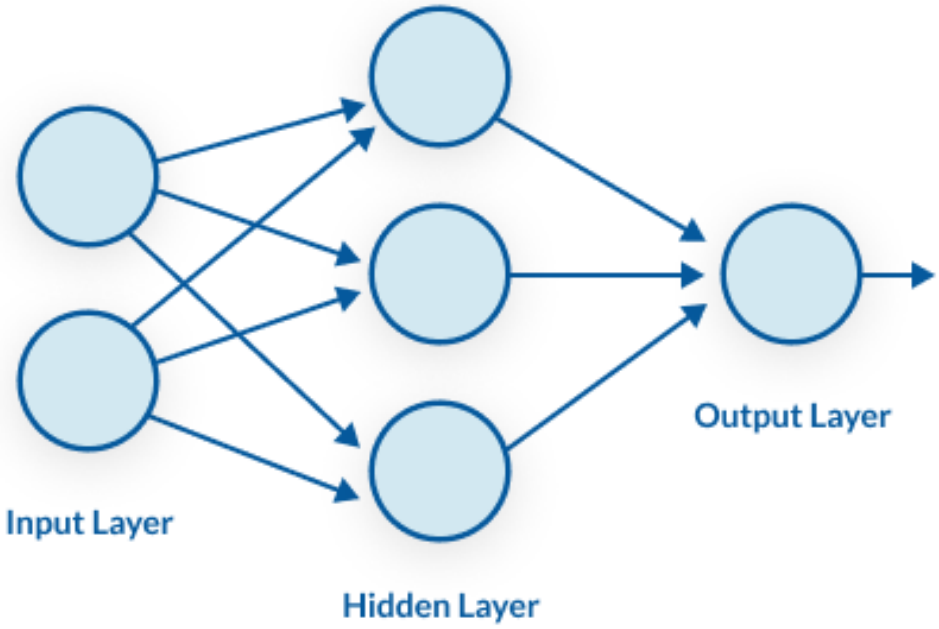
Assim como a mente humana que é permeada de vieses de diversos tipos as redes neurais artificiais também se adaptam a vieses a fim de usá-los para se chegar ao melhor resultado possível.

Formatando esse valor com a função sigmoide teremos: **1 / (1 + 2.71828 ^ 2.4) = 0.08317**.

Observe que agora a nossa saída de **0.08317** está muito mais próxima de **0.08** do que a saída anterior de **0.07585**. E dessa forma a cada novo backpropagation o nosso resultado se torna cada vez mais preciso certo? Calma, não é bem assim, da mesma forma que nós aumentamos os nossos vieses quando nos tornamos

expostos repetidas vezes aos mesmos cenários e isso torna nosso raciocínio tendencioso, nas redes neurais artificiais acontece o mesmo. Se o número de épocas para as execuções de backpropagations for muito superior ao necessário, como os pesos continuarão sendo alterados o resultado começará a exceder o valor que esperamos. Da mesma forma se usarmos épocas insuficientes o valor poderá ficar muito abaixo do valor esperado. Então nunca devemos exagerar para mais ou para menos buscando resultados exatos de mais.

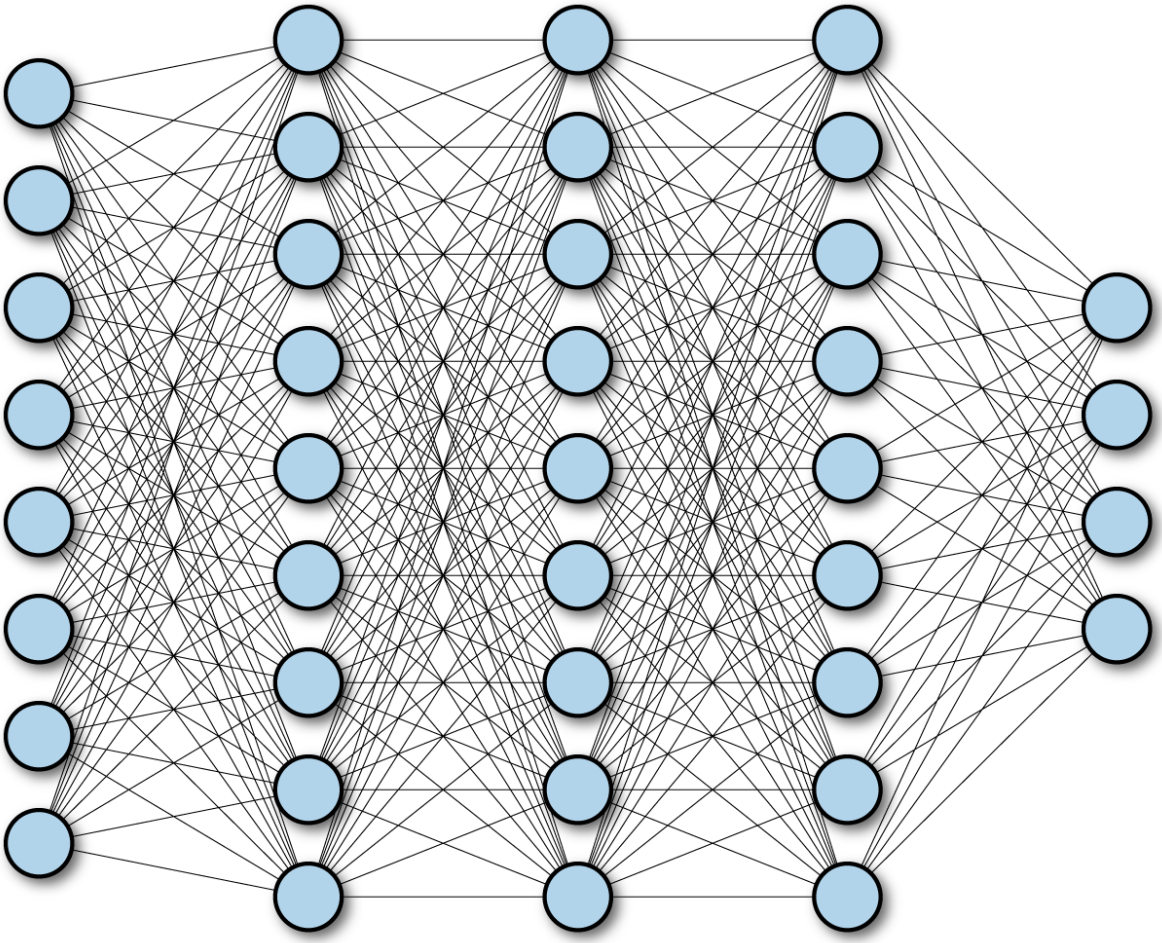
Este modelo de rede neural artificial que utilizamos como exemplo é uma função de aproximação simples que trabalha para aproximar as entradas das saídas, também conhecida como Perceptron (ou Perceptron Simples). Mas existe uma variante desse modelo que nos permite adicionar camadas de neurônios interligados entre a camada de inputs e de outputs que conseguem extrair uma quantidade superior de padrões dos nossos dados que é o Multilayer Perceptron (ou Perceptron Multicamadas). A essas camas extras damos o nome de Hidden Layers (ou Camadas Ocultas). Veja como seria a arquitetura do Multilayer Perceptron:



Observe que agora temos de fato uma rede com diversos neurônios trocando informações entre si. Cada camada possui seus valores específicos, a camada de entrada contém os valores brutos antes do processamento, a camada oculta os seus valores dos pesos e a camada de saída o resultado do processamento. Esses valores de ligação entre as camadas são chamados de nós e no exemplo acima temos uma rede neural com 2 nós na camada de entrada, ou seja, um vetor na entrada, 3 nós na camada oculta de pesos extras e um único nó na camada de saída representando uma variável simples, mas também podemos usar vetores com múltiplos elementos na saída da mesma forma que fazemos na entrada. Hoje em dia já existem redes neurais artificiais com milhões de neurônios com tanta precisão que são capazes de estabelecer diálogos por texto ou voz, identificar padrões estatísticos, realizar diagnósticos por imagem e incontáveis outras funções que vem sendo descobertas a cada dia.

O que vêm se tornando um mistério nos últimos anos com o aumento da complexidade das redes neurais artificiais que estão recebendo cada vez mais camadas com um número cada vez maior de neurônios/nós é a impossibilidade de se mapear tantos parâmetros e como essas redes estão assimilando a informação. Por que como os pesos sinápticos são calculados de forma totalmente aleatória e ajustados com base na aleatoriedade dos números gerados, quanto maior for a rede, menor será a nossa capacidade de monitorar o seu processo de aprendizagem fazendo com que esse tipo de algoritmo se torne uma caixa preta matemática. Empresas como o Google e a IBM, por exemplo, já possuem redes neurais com milhões de camadas e conexões neuronais onde o seu funcionamento é um mistério até mesmo para os desenvolvedores. É por esse mesmo motivo que cientistas não conseguem mapear a nossa mente que possui bilhões de neurônios distribuídos em uma quantidade exorbitante de camadas conectivas.

Observe por exemplo à arquitetura a seguir que já é bem mais complexa do que a anterior e por isso consegue operar com uma quantidade muito superior de parâmetros e reconhecer ainda mais padrões:



Se você entendeu tudo até aqui você já deve ter percebido que a informação em uma rede neural Perceptron sempre irá se propagar da esquerda para a direita, ou seja, sempre para frente. Iniciando na camada de entrada a esquerda, passando pelas camadas ocultas (caso elas existam), até chegar à camada de saída na extremidade direita e o processo se repete sempre na mesma direção a cada backpropagation durante as épocas de treinamento. Por esse motivo redes neurais que seguem esse tipo de arquitetura são tipificadas como redes neurais do tipo Feedforward (propagação/alimentação para frente).

O problema do operador lógico XOR

Na computação utilizamos operadores lógicos que acionam as portas lógicas do processador para que possamos programar, cada operador computa duplas de binários **(0 ou 1)** de forma diferente. No caso do operador **AND** teremos como retorno **1 (verdadeiro)** somente se ambas as entradas das portas lógicas forem "verdadeiras" ou seja, iguais a **1**. Já o operador lógico **OR** consegue retornar **1** como condição "verdadeira" quando pelo menos uma das entradas conter **1**, isso faz com que ele admita como "verdadeiro" **[0, 1], [1, 0] ou [1, 1]**. Porém no **XOR** que também é conhecido como **OR exclusivo**, só teremos "verdadeiro" para **[0, 1] e [1, 0]** excluindo a entrada **[1, 1]** do **OR** (daí o nome **OR exclusivo**).

Vejamos como isso ficaria em uma tabela gráfica...

AND

x	y	F
0	0	0
0	1	0
1	0	0
1	1	1

OR

x	y	F
0	0	0
0	1	1
1	0	1
1	1	1

XOR

x	y	F
0	0	0
0	1	1
1	0	1
1	1	0

Como as redes neurais artificiais irão sempre somar os produtos das entradas pelos pesos e os pesos sempre estarão no intervalor entre **0 e 1**, isso fará com que os resultados da função somatória sejam sempre resultados lineares, o que quer dizer que esses valores formariam uma linha reta (horizontal, vertical ou diagonal) se distribuídos em um plano cartesiano. Então tente separar as saídas do tipo **0** das saídas do tipo **1** de cada um dos operadores usando uma única linha reta, no caso uma linha horizontal já que as nossas entradas estão distribuídas nas linhas de uma tabela. Este é um caso simples e bem conhecido de classificação lógica onde temos duas classes diferentes **(0 e 1)** e precisamos classificar os dados fazendo a separação de um tipo de binário do outro.

Com os operadores **AND e OR** conseguimos fazer essa separação linear facilmente:

AND

x	y	F
0	0	0
0	1	0
1	0	0
1	1	1

OR

x	y	F
0	0	0
0	1	1
1	0	1
1	1	1

Agora tente fazer isso com o operador **XOR** e verá que é impossível usando apenas uma única linha, neste caso precisaríamos de no mínimo mais uma linha extra e essa linha extra é conseguida através do acréscimo de uma camada oculta à nossa rede Perceptron fazendo com que ela evolua para uma rede Multilayer Perceptron que é capaz de reconhecer muito mais padrões. O Perceptron no caso dos nossos operadores conseguiria reconhecer apenas dois padrões já que uma única linha separa os dados em dois grupos distintos, mas esse valor aumenta à medida que acrescentamos camadas ocultas a nossa rede Multilayer Perceptron. Como aqui precisaríamos reconhecer de **2 a 3** padrões então acrescentando uma camada oculta a nossa rede conseguiríamos chegar ao resultado desejado. No exemplo abaixo a linha na cor verde representa a camada oculta acrescentada ao Perceptron o transformando em uma rede Multilayer Perceptron capaz de reconhecer além de dois padrões:

AND

x	y	F
0	0	0
0	1	0
1	0	0
1	1	1

OR

x	y	F
0	0	0
0	1	1
1	0	1
1	1	1

XOR

x	y	F
0	0	0
0	1	1
1	0	1
1	1	0

Isso é tudo pessoal, temos aqui tudo o que você precisa saber para iniciarmos a codificação dos nossos algoritmos de redes neurais artificiais. Boa sorte!