

## ▼ Rede Neural Multicamadas (MPL)

Uma rede MPL é uma classe de rede neural artificial *feedforward* (ANN). Um MLP consiste em pelo menos três camadas de nós: uma camada de entrada, uma camada oculta e uma camada de saída. Exceto para os nós de entrada, cada nó é um neurônio que usa uma função de ativação não linear. O MPL utiliza uma técnica de aprendizado supervisionado chamada *backpropagation* para treinamento.

### Implementando uma RNA multicamadas


A imagem a seguir mostra a nossa rede, com as unidades de entrada marcadas como Input1, Input2 e Input3 (**Input Layer**) conectadas com os nós da camada oculta (**Hidden Layer**). Por sua vez as saídas dos nós da camada oculta servem como entrada para os nós da camada de saída (**Output Layer**). 

Diagrama de uma MPL

Lembrando que em cada nó temos:

$$f(h) = \text{sigmoid}(h) = \frac{1}{1 + e^{-h}}$$

onde

$$h = \frac{1}{n} \sum_{i=1}^n (w_i * x_i) + b$$

## ▼ Configuração da MPL

```
#Importando a biblioteca
import numpy as np

#Função do cálculo da sigmóide
def sigmoid(x):
    return 1/(1+np.exp(-x))

#Arquitetura da MPL
N_input = 3
N_hidden = 5
N_output = 4

#Vetor dos valores de entrada
x = np.array([0.1, 0.2, -0.6])
target = np.array([0.7, 0.2])
learnrate = 0.5
```



```
weights_hidden_output = np.array([[ 0.15, -0.11],  
                                   [-0.08,  0.05],  
                                   [-0.04,  0.07],  
                                   [-0.03,  0.07]])
```

## Forward Pass

```
hidden_layer_input = np.dot(x, weights_input_hidden)  
hidden_layer_output = sigmoid(hidden_layer_input)  
  
output_layer_in = np.dot(hidden_layer_output, weights_hidden_output)  
  
output = sigmoid(output_layer_in)  
  
print('As saídas da rede são',output)
```

```
As saídas da rede são [0.45825438 0.537853  ]
```

## Backward Pass

```
error = target - output  
  
output_error_term = error * output * (1 - output)  
  
hidden_error = np.dot(weights_hidden_output,output_error_term)  
  
hidden_error_term = hidden_error * hidden_layer_output * (1 - hidden_layer_output)  
  
delta_w_h_o = learnrate * output_error_term*hidden_layer_output[:, None]  
print('delta_w_h_o: ',delta_w_h_o)  
  
delta_w_i_h = learnrate * hidden_error_term * x[:, None]  
print('delta_w_i_h: ',delta_w_i_h)
```

```
delta_w_h_o: [[ 0.0153563 -0.02148808]  
 [ 0.01501878 -0.02101578]  
 [ 0.01516131 -0.02121523]  
 [ 0.01501128 -0.02100529]]
```

```
weights_input_hidden: [[-1.25183529e-04 -5.62509878e-05 -5.17389553e-05 -4
[-2.50367059e-04 -1.12501976e-04 -1.03477911e-04 -9.59874061e-05]
[ 7.51101176e-04  3.37505927e-04  3.10433732e-04  2.87962218e-04]]
weights_hidden_output: [[ 0.00767815 -0.01074404]
[ 0.00750939 -0.01050789]
[ 0.00758066 -0.01060761]
[ 0.00750564 -0.01050264]]
```