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 MLP 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 saída dos *nós* da camada oculda servem como entrada para os *nós* da camada de saída (**Output Layer**).

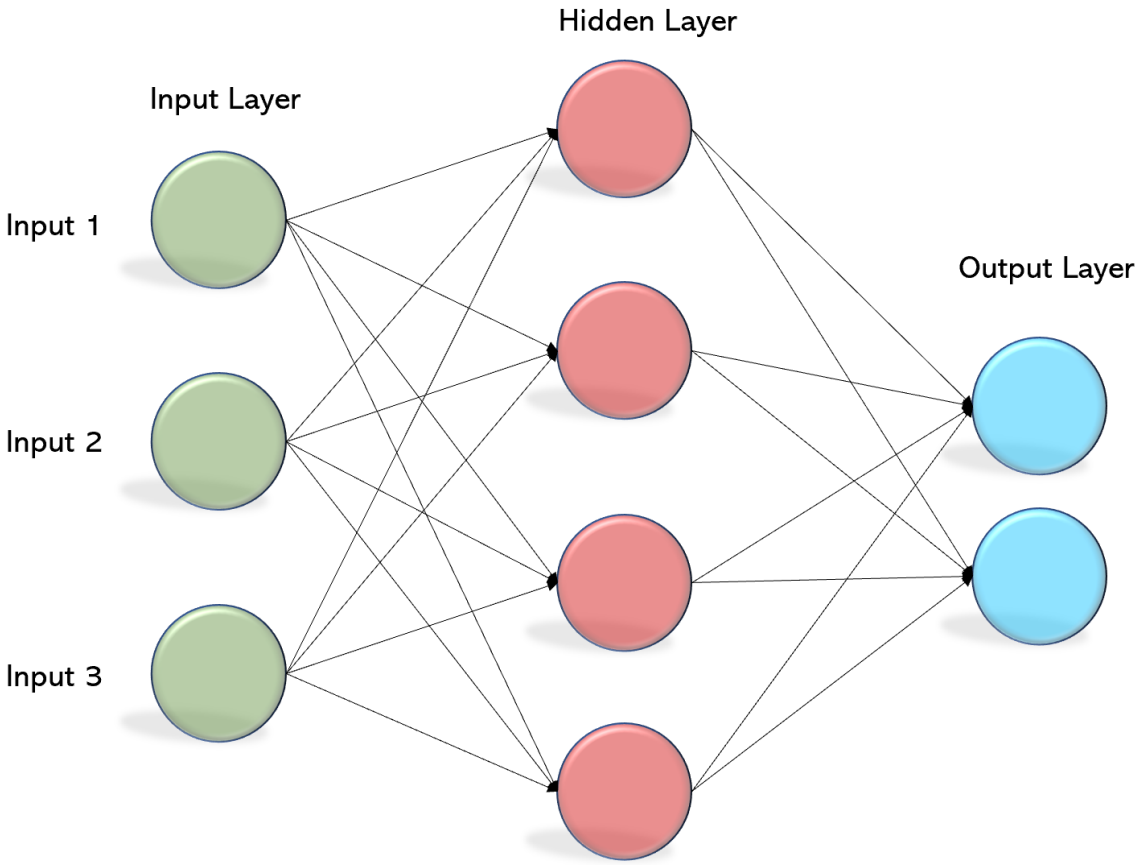

Diagrama de uma MPL

Lembrando que em cada *nó* temos:

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

onde

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

Vamos implementar uma RNA de apenas um neurônio!

Importando a biblioteca

In [1]:

```
import numpy as np
```

Função do cáculo da sigmóide

In [2]:

```
def sigmoid(x):
    result = (1+np.exp(-x)) # 1 + (e^ -x)
    return 1/result         # 1 / (1 + (e^ -x))
```

Arquitetura da MPL

In [3]:

```
N_input = 3
N_hidden = 4
N_output = 2
```

Vetor dos valores de entrada

In [4]:

```
X = np.array([1, 2, 3])
```

Pesos da Camada Oculta

In [5]:

```
weights_in_hidden = np.array([[-0.07,  0.04, -0.05, 0.07],
                              [ 0.04,  0.10,  0.02, 0.01],
                              [-0.03,  0.04, -0.11, 0.06]])
```

Pesos da Camada de Saída

In [6]:

```
weights_hidden_out = np.array([[-0.10,  0.09],
                               [-0.04,  0.12],
                               [-0.02,  0.04],
                               [-0.01,  0.09]])
```

Passagem *forward* pela rede

Camada oculta

In [7]:

```
#Calcule a combinação linear de entradas e pesos sinápticos
hidden_layer_in = np.dot(X, weights_in_hidden)

#Aplicado a função de ativação
hidden_layer_out = sigmoid(hidden_layer_in)
```

Camada de Saída

In [8]:

```
#Calcule a combinação linear de entradas e pesos sinápticos
output_layer_in = np.dot(hidden_layer_out, weights_hidden_out)

#Aplicado a função de ativação
output_layer_out = sigmoid(output_layer_in)
```

In [9]:

```
print('O input da camada oculta é:',hidden_layer_in)
```

```
O input da camada oculta é: [-0.08  0.36 -0.34  0.27]
```

In [10]:

```
print('O output da camada oculta é:',hidden_layer_out)
```

```
O output da camada oculta é: [0.48001066 0.58904043 0.41580948 0.56709
29 ]
```

In [11]:

```
print('O input da camada de output é:',output_layer_in)
```

```
O input da camada de output é: [-0.0855498   0.18155655]
```

In [12]:

```
print('As saídas da rede são',output_layer_out)
```

```
As saídas da rede são [0.47862558 0.54526487]
```

In [ ]:

```

```