

## ▼ 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 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

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

## ▼ Vamos implementar uma RNA de apenas um neurônio!

Importando a biblioteca

```
import numpy as np
```

## ▼ Função do cálculo da sigmóide

```
def sigmoid(x):
    return 1/(1+np.exp(-x))
```

## ▼ Arquitetura da MPI



Vetor dos valores de entrada

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

Pesos da Camada Oculta

```
weights_in_hidden = np.array([[-0.18,  0.18, -0.03, 0.03],  
                               [ 0.05,  0.10,  0.07, 0.02],  
                               [-0.07,  0.04, -0.05, 0.05]])
```

Pesos da Camada de Saída

```
weights_hidden_out = np.array([[-0.18,  0.11],  
                                [-0.09,  0.05],  
                                [-0.04,  0.05],  
                                [-0.02,  0.07]])
```

Passagem *forward* pela rede

Camada oculta

```
#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

```
#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)
```

