# Agenda

**UNIDADE 5: Redes Neurais**

5.1. Estrutura básica de uma rede neural
5.2. Arquitetura de redes neurais
5.2.1. Perceptron
5.2.2. Multilayer Perceptron
5.3. Treinamento de redes neurais
5.3.1. Algoritmos de retropropagação
5.3.2. Otimização de pesos
5.3.3. Função de ativação
5.4. Arquiteturas de Redes Neurais Profundas

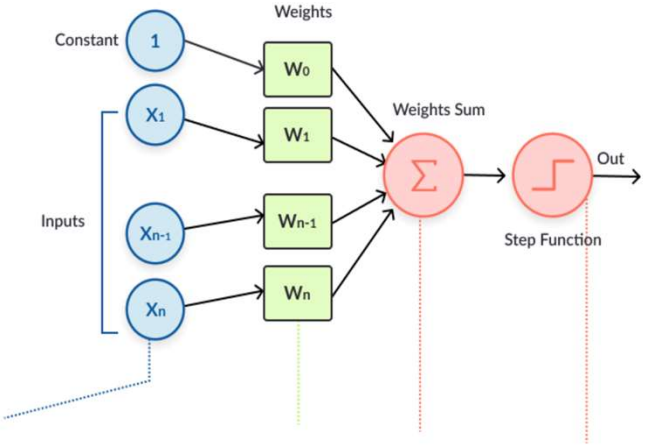
# Predição de Tempo de Atravessamento de Ações na Justiça

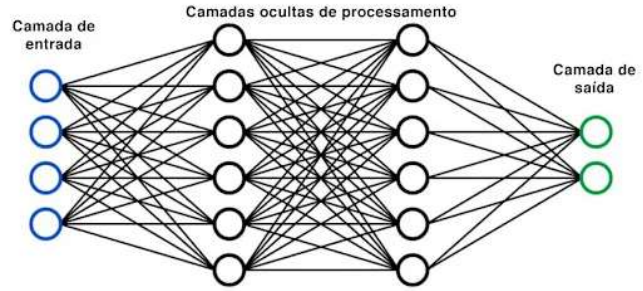
Descrição: Prever o tempo de atravessamento do processo (da solicitação até a efetiva entrega) utilizando informações históricas e considerando as características do processo no momento de autuação.

- Rito (valores: trabalhista ou sumaríssimo)
- Tempo de Serviço do Reclamante (valores: tempo em meses até a data da despensa)
- Último salário do reclamante (valores: número real)
- Profissão do reclamante (valores: comércio, indústria ou serviço)
- Cargo do reclamante (valores: direção ou execução)
- Objeto do processo (valores: falta de registro em carteira, diferença salarial, verbas recisórias, multa do Art, 477, multa do Art. 467, horas extras e reflexos, fundo de garantia por tempo de serviço, indenização por dados morais, seguro desemprego, vale transporte, adicional de insalubridade, adicional noturno, plano de saúde)
- Quantidade de depoimentos em cada audiência (valores: número inteiro entre 1 e 200)
- Acordo (valores: presença ou ausência)
- Necessidade de perícia (valores: S para Sim e N para Não)
- Solicitação de recurso ordinário contra sentença emitida pelo Juiz de 1 grau (valores: S para Sim e N para Não)
- Solicitação de recurso de revista contra acordão (valores: S para Sim e N para Não)
- Número de audiências até a emissão da sentença (valores: número inteiro entre 1 e 200)
- Tempo médio de cada audiência (valor inteiro em minutos entre 30 e 1000)
- Duração do processo (valor inteiro em meses entre 1 e 500)

# Redes Neurais

- **Self-learning treinados**, não são explicitamente programados
- No centro das redes neurais básicas está a unidade chamada de **nó ou neurônio matemático.**
- Um neurônio recebe uma ou mais **entradas.**
- Cada entrada é multiplicada por um **peso**.
- Os valores da entrada ponderada é somada a algum **valor de polarização.**
- Em seguida, o valor é alimentado em uma **função de ativação.**
- Essa **saída** é enviada para outros **neurônios mais profundos na rede neural.**

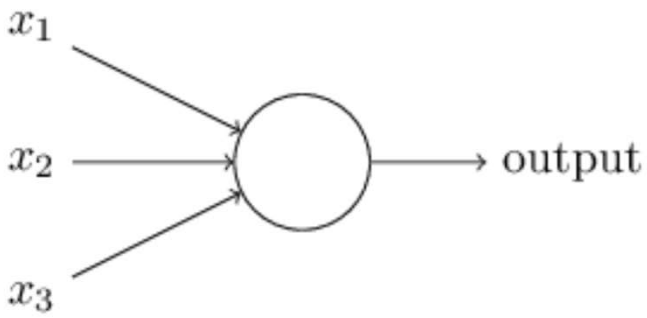


# Perceptron

- Um Perceptron é um modelo matemático que recebe várias entradas, x1, x2, … e produz uma única saída binária (duas classes).
- Resolve problemas linearmente separáveis

Exemplo do Perceptron: z = w1x1 + w2x2 + w3x3 + w0

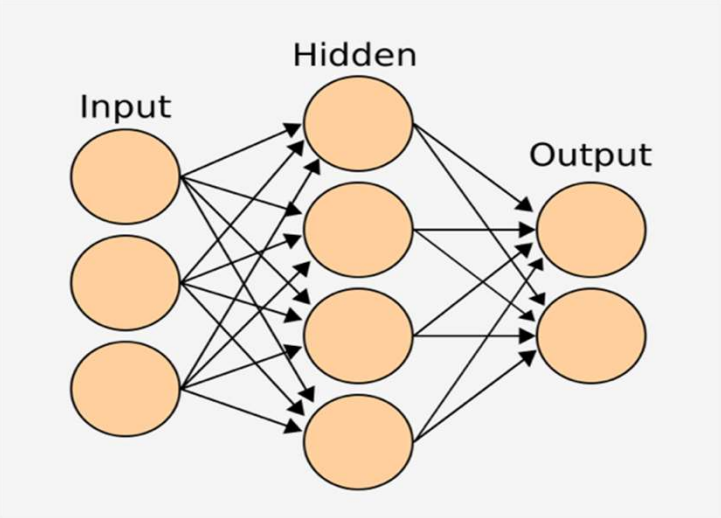


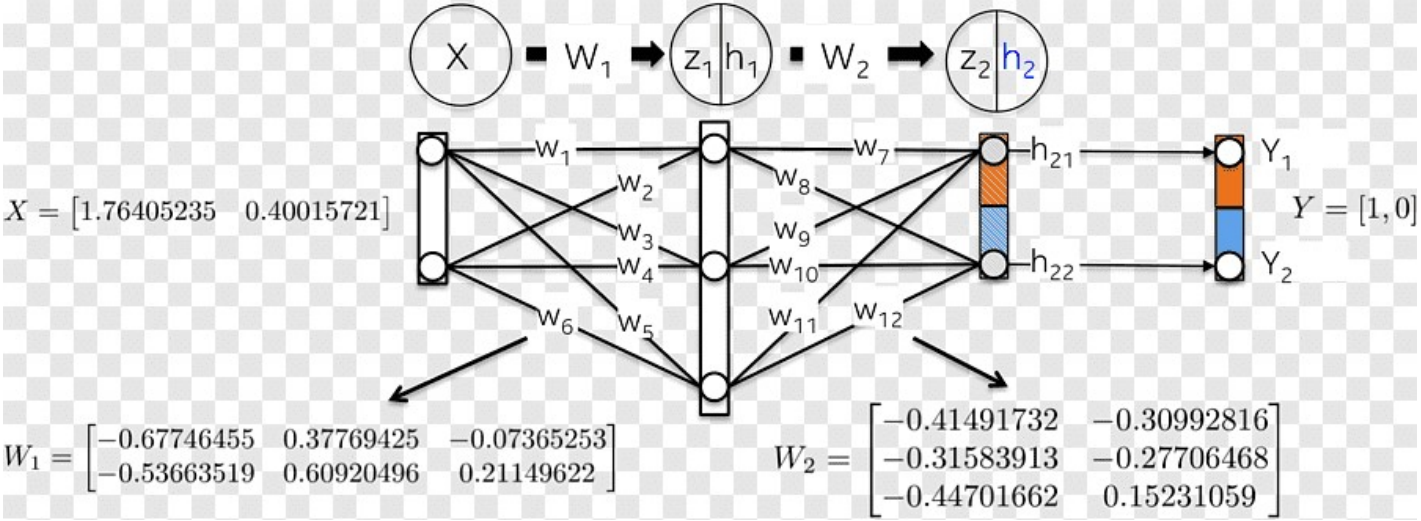

Perceptron produz uma única saída com base em várias entradas de valor real

$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{threshold} \end{cases}$$

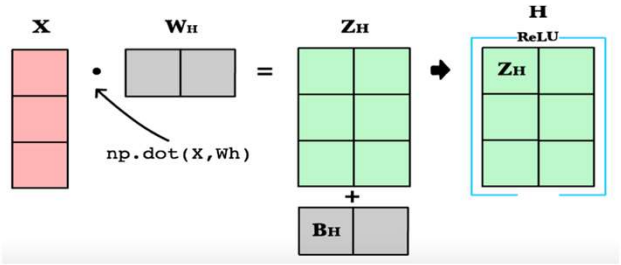
# Redes Multilayer Perceptrons

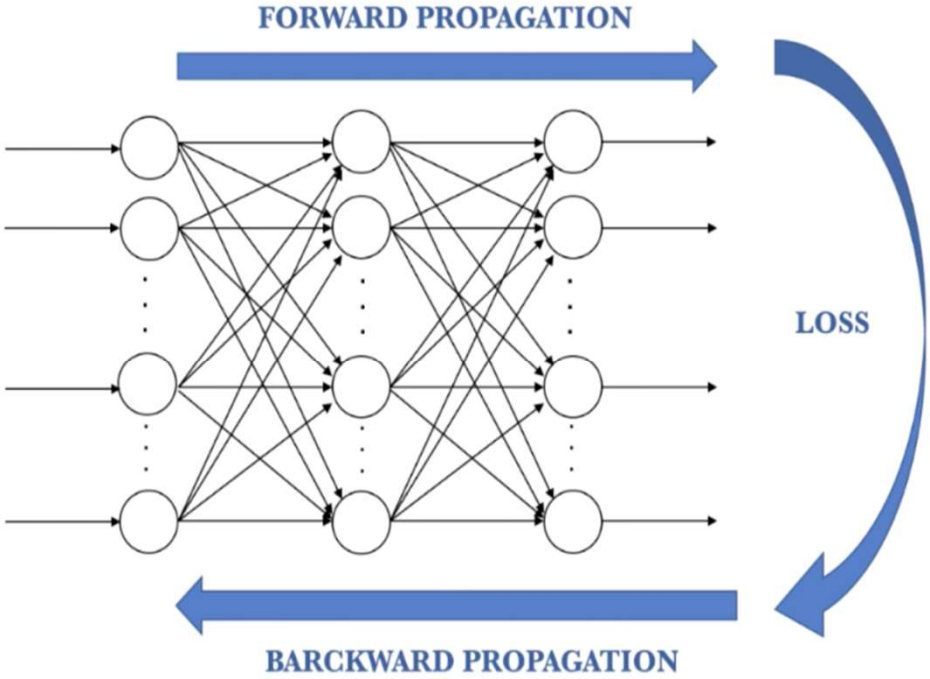
- Multilayer Perceptron (MLP) é uma rede neural artificial composta por mais de um Perceptron

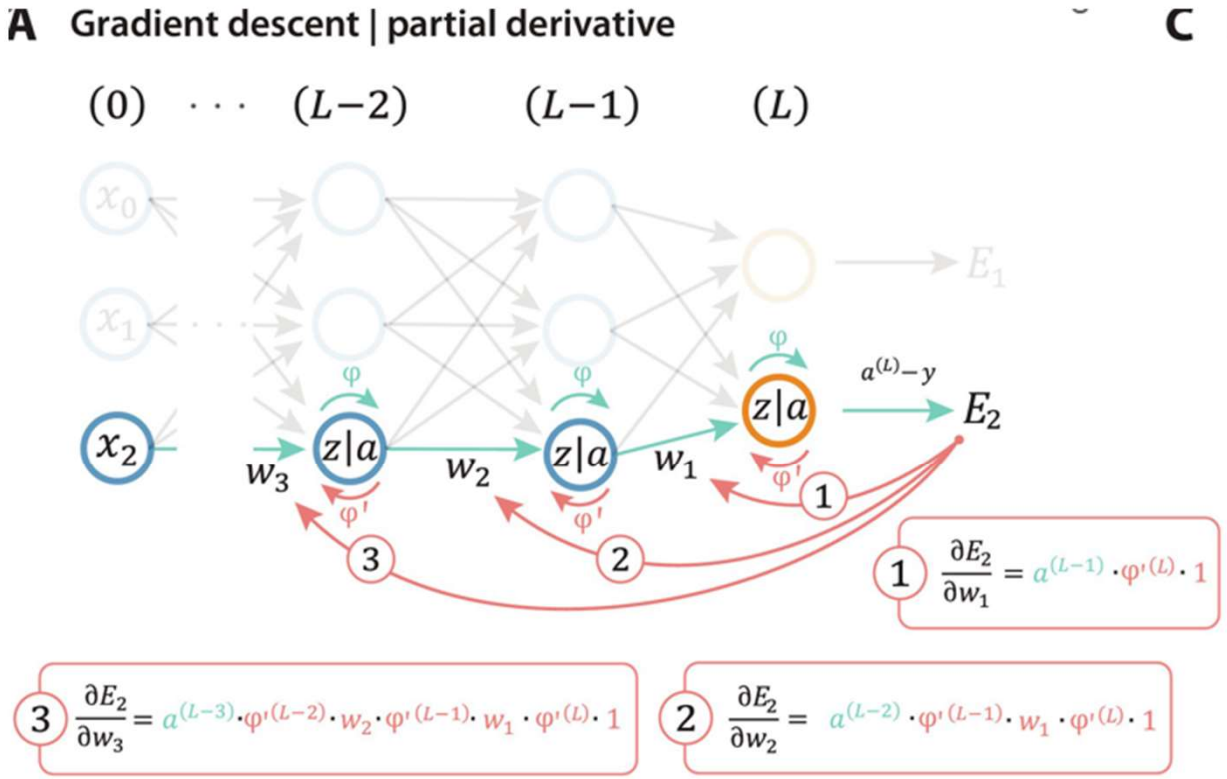


# Algoritmo de Backpropagation



# Aprendizagem

- Processo interativo de otimização dos pesos, que são modificados com base no desempenho do modelo.

LR 0,01

| Foward Feed + New Weights | N | y1 | y2 | y1+y2 | w1 | w2 | z=w1*y1+w2*y2 | Erro=(y1 + y2) - (w1*y1+w2*y2) | w1 + LR*y1 + Erro | w2 + LR*y2 + Erro |
|---|---|---|---|---|---|---|---|---|---|---|
| Primeiro Treinamento | 0 | 1 | 1 | 2 | 0,4 | 0,6 | 1,00 | 1,00 | 0,41 | 0,61 |
| | 1 | 2 | 2 | 4 | 0,41 | 0,61 | 2,04 | 1,96 | 0,45 | 0,65 |
| | 2 | 3 | 3 | 6 | 0,45 | 0,65 | 3,30 | 2,70 | 0,53 | 0,73 |
| | 3 | 4 | 4 | 8 | 0,53 | 0,73 | 5,04 | 2,96 | 0,65 | 0,85 |
| | 4 | 5 | 5 | 10 | 0,65 | 0,85 | 7,49 | 2,51 | 0,77 | 0,97 |
| | 5 | 6 | 6 | 12 | 0,77 | 0,97 | 10,49 | 1,51 | 0,86 | 1,06 |
| | 6 | 7 | 7 | 14 | 0,86 | 1,06 | 13,51 | 0,49 | 0,90 | 1,10 |
| | 7 | 8 | 8 | 16 | 0,90 | 1,10 | 15,99 | 0,01 | 0,90 | 1,10 |
| | 8 | 9 | 9 | 18 | 0,90 | 1,10 | 18,00 | 0,00 | 0,90 | 1,10 |
| | 9 | 10 | 10 | 20 | 0,90 | 1,10 | 20,00 | 0,00 | 0,90 | 1,10 |
| Teste | 0 | 11 | 11 | 22 | 0,90 | 1,10 | 22,00 | 0,00 | 0,90 | 1,10 |
| | 1 | 3 | 2 | 5 | 0,90 | 1,10 | 4,90 | 0,10 | 0,90 | 1,10 |
| | 2 | 3 | 5 | 8 | 0,90 | 1,10 | 8,20 | -0,20 | 0,89 | 1,09 |
| | 3 | 2 | 1 | 3 | 0,90 | 1,10 | 2,90 | 0,10 | 0,90 | 1,10 |
| | 4 | 6 | 7 | 13 | 0,90 | 1,10 | 13,10 | -0,10 | 0,89 | 1,09 |
| | 5 | 8 | 3 | 11 | 0,90 | 1,10 | 10,50 | 0,50 | 0,94 | 1,11 |

1. Recebe um número de entrada (x1,x2,x3,4)

1. **Multiplica** cada entrada pelo peso wi (aleatório primeira vez)

1. Soma todas as entradas **ponderadas com o bias**.

1. Calcula o custo com base em uma **função de perda (custo)** e meta da rede (depende do tipo de problema)
   - *Classificação binária:* **Binary cross-entropy**

$$\text{Loss} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(p_i) + (1-y_i)\log(1-p_i)]$$

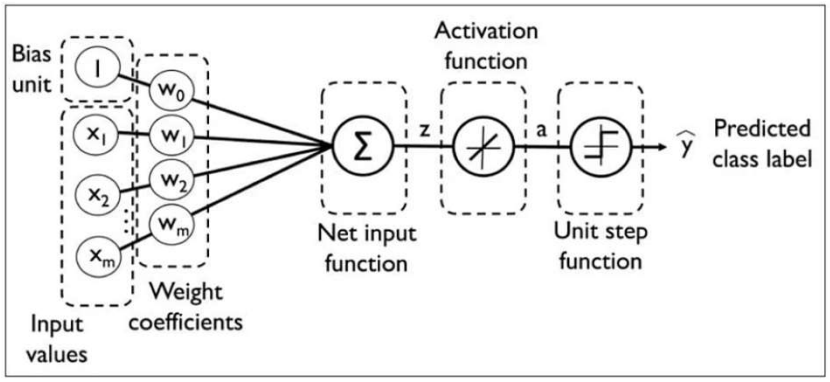
   - *Classificação multiclasse:* **Categorical cross-entropy**

$$\text{Loss} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij}\log(p_{ij})$$

   - *Regreção:* **Mean square error**

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

- Rito (valores: trabalhista ou sumaríssimo)
- Tempo de Serviço do Reclamante (valores: tempo em meses até a data da despensa)
- Último salário do reclamante (valores: número real)
- Profissão do reclamante (valores: comércio, indústria ou serviço)
- Cargo do reclamante (valores: direção ou execução)
- Objeto do processo (valores: falta de registro em carteira, diferença salarial, verbas recisórias, mult horas extras e reflexos, fundo de garantia por tempo de serviço, indenização por dados morais, se adicional de insalubridade, adicional noturno, plano de saúde)
- Quantidade de depoimentos em cada audiência (valores: número inteiro entre 1 e 200)
- Acordo (valores: presença ou ausência)
- Necessidade de perícia (valores: S para Sim e N para Não)
- Solicitação de recurso ordinário contra sentença emitida pelo Juiz de 1 grau (valores: S para Sim e
- Solicitação de recurso de revista contra acordão (valores: S para Sim e N para Não)
- Número de audiências até a emissão da sentença (valores: número inteiro entre 1 e 200)
- Tempo médio de cada audiência (valor inteiro em minutos entre 30 e 1000)
- Duração do processo (valor inteiro em meses entre 1 e 500)



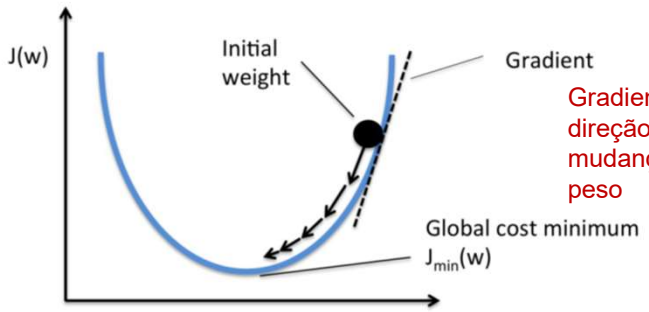
Os otimizadores são algoritmos utilizados para ajustar os pesos do modelo de aprendizado de máquina de forma a minimizar a função de erro (ou função de perda). Eles aplicam técnicas matemáticas para encontrar os parâmetros ótimos do modelo que resultam no menor erro possível na previsão dos dados

5. Aplica o **otimizador** para encontrar o **menor erro**
   - *Escolhas comuns:*
     - *stochastic gradient descent,*
     - *stochastic gradient descent com momentum,*
     - *root mean square propagation, e*
     - *adaptive moment estimation*

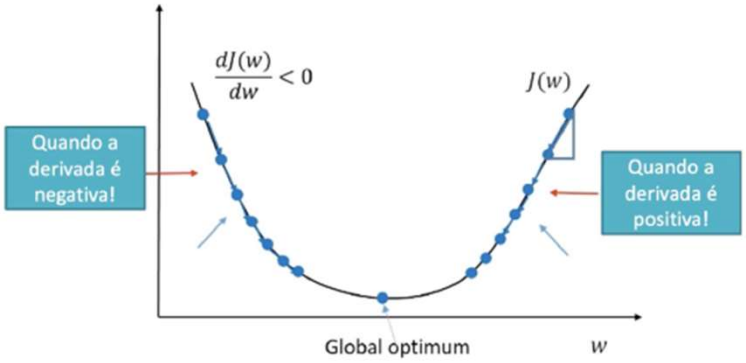
$$z = w_1x_1 + w_2x_2 + \ldots + w_mx_m + w_0$$

$$w_{12} = w_1 - learning\_rate \times \frac{\partial custo}{\partial w_1}$$

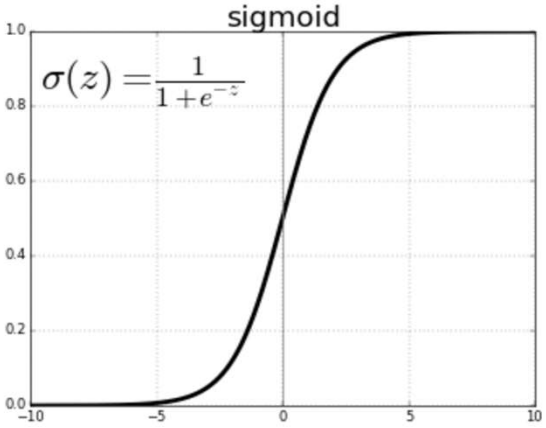
$$z = w_{12}x_1 + w_{22}x_2 + \ldots + w_{m2}x_m + w_0$$

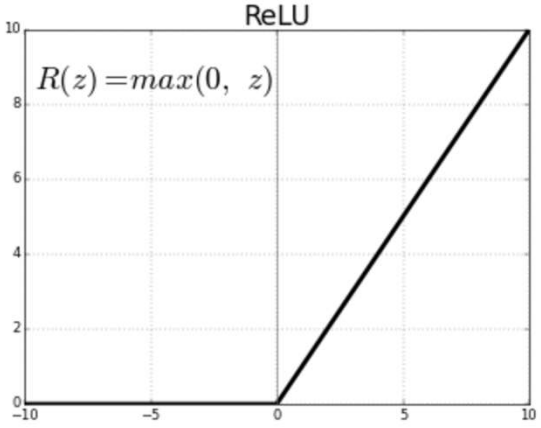


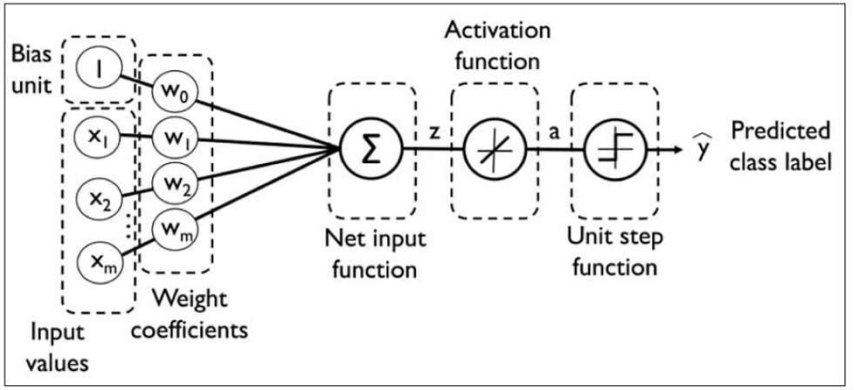

Gradiente indica a direção e magnitude d mudança para cada peso



6. Aplica uma **função de ativação**
   - Uma função de ativação popular é a *rectified linear unit* (ReLU):
     $f(z) = \max(0, z)$
   - se *z é maior que 0*, a função de ativação retorna z, caso contrário 0.



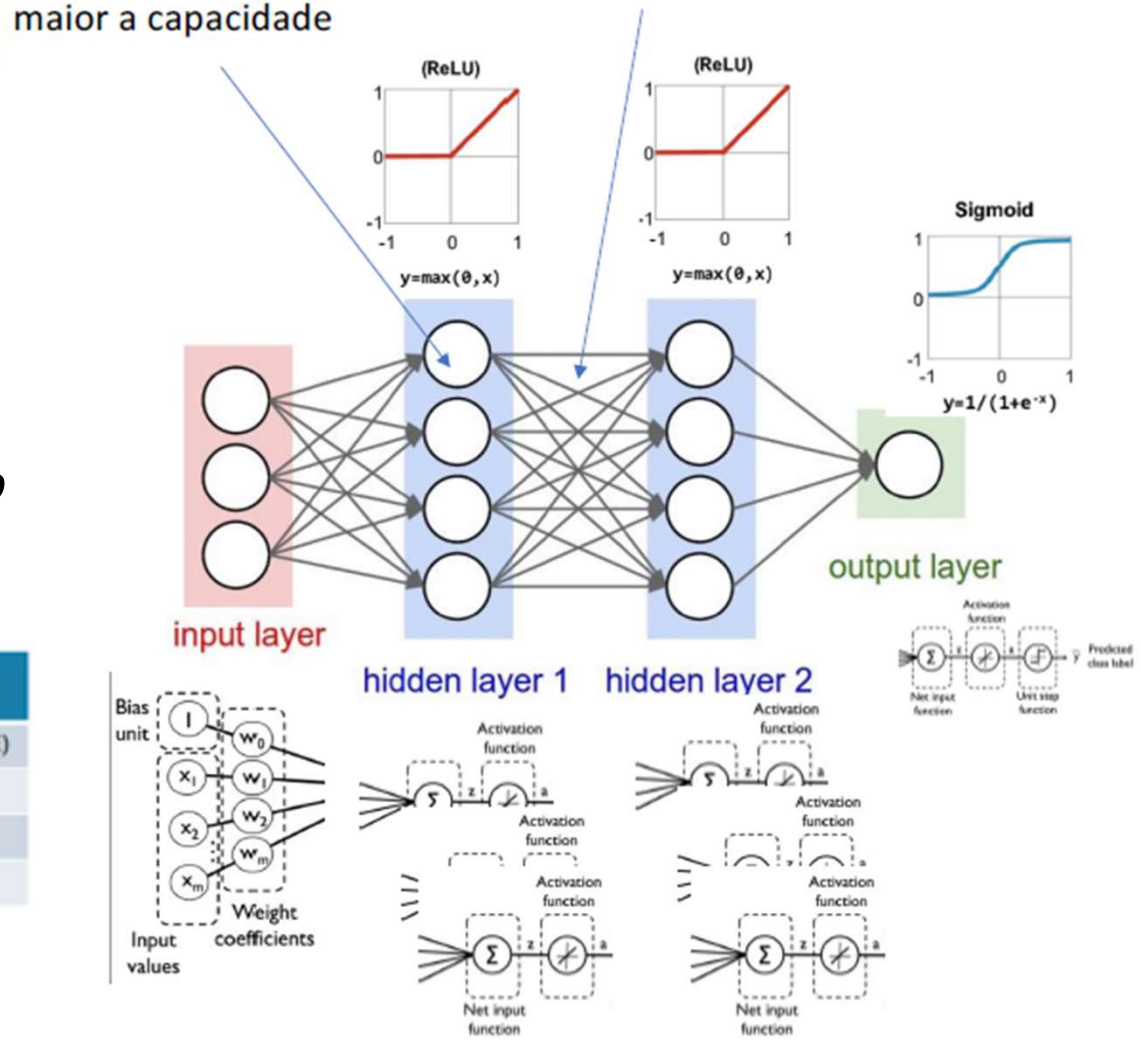



7. Envia a saída para a próxima camada
8. Aplica a **função de ativação da saída** com base na meta da rede
   - *Classificação binária: sigmoid*
   - *Classificação multiclasse: softmax*
   - *Regression: sem função de ativação*

| Problem Type | Output Type | Final Activation Function | Loss Function |
|---|---|---|---|
| Regression | Numerical value | Linear | Mean Squared Error (MSE) |
| Classification | Binary outcome | Sigmoid | Binary Cross Entropy |
| Classification | Single label, multiple classes | Softmax | Cross Entropy |
| Classification | Multiple labels, multiple classes | Sigmoid | Binary Cross Entropy |

Quanto mais nenuônios nas camadas intermediárias maior a capacidade



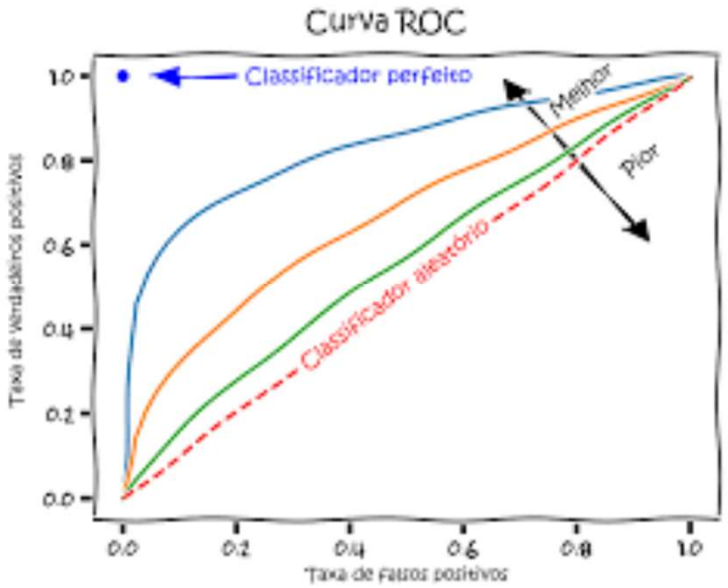
## 9. Avalia a performance

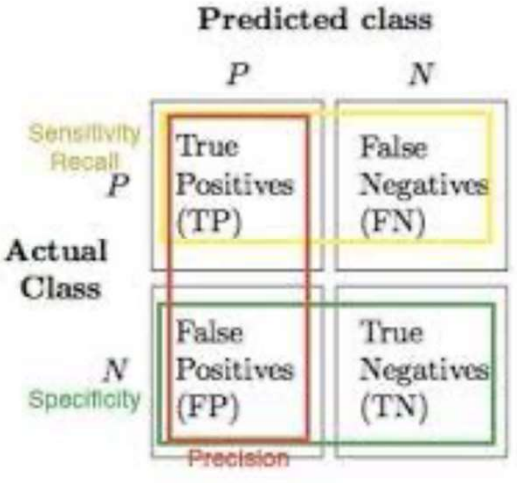
$$Accuracy = \frac{T_p + T_n}{T_p + T_n + F_p + F_n}$$

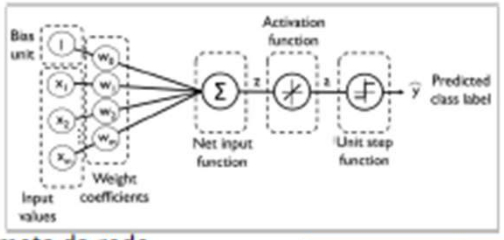
$$Precision = \frac{T_p}{T_p + F_p}$$

$$Recall = \frac{T_p}{T_p + T_n}$$

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$





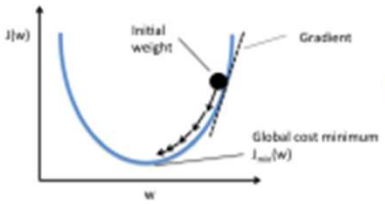

- 1. Recebe um número de entrada
- 2. **Multiplica** cada entrada pelo peso
- 3. Soma todas as entradas **ponderadas com o bias**.

- 4. Calcula o custo com base em uma **função de perda** e meta da rede
  - *Classificação binária:* Binary cross-entropy
  - *Classificação multiclasse:* Categorical cross-entropy
  - *Regression:* Mean square error

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

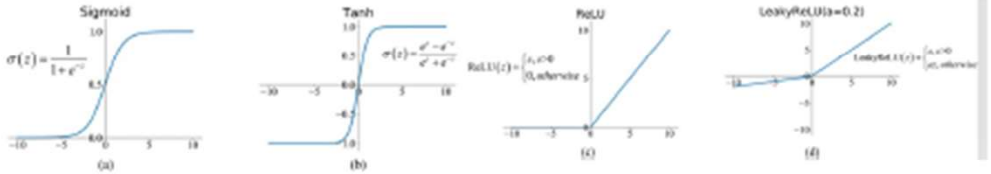
| Problem Type | Output Type | Final Activation Function | Loss Function |
|---|---|---|---|
| Regression | Numerical value | Linear | Mean Squared Error (MSE) |
| Classification | Binary outcome | Sigmoid | Binary Cross Entropy |
| Classification | Single label, multiple classes | Softmax | Cross Entropy |
| Classification | Multiple labels, multiple classes | Sigmoid | Binary Cross Entropy |

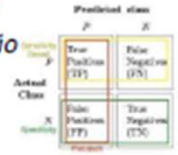
- 5. Aplica o otimizador para encontrar o **menor erro**
  - *Escolhas comuns: stochastic gradient descent, stochastic gradient descent com momentum, root mean square propagation, e adaptive moment estimation*




$$w_{12} = w_1 - learning\_rate \times \frac{\partial custo}{\partial w_1}$$

- 6. Aplica uma função de ativação
  - Uma função de ativação popular e a *rectified linear unit* (ReLU): $f(z) = \max(0, z)$
  - *z é a soma das entradas ponderada e bias*, se *z é maior que 0*, a função de ativação retorna z, caso contrario 0.
- 7. Envia a saída para a próxima camada
- 8. Aplica a função de ativação da saída com base na meta da rede
  - *Classificação binária: sigmoid*
  - *Classificação multiclasse: softmax*
  - *Regression: sem função de ativação*



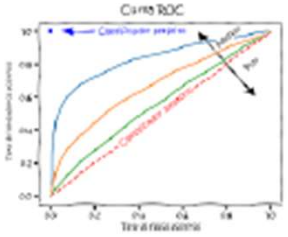




$$Accuracy = \frac{T_p + T_n}{T_p + T_n + F_p + F_n}$$

$$Precision = \frac{T_p}{T_p + F_p}$$

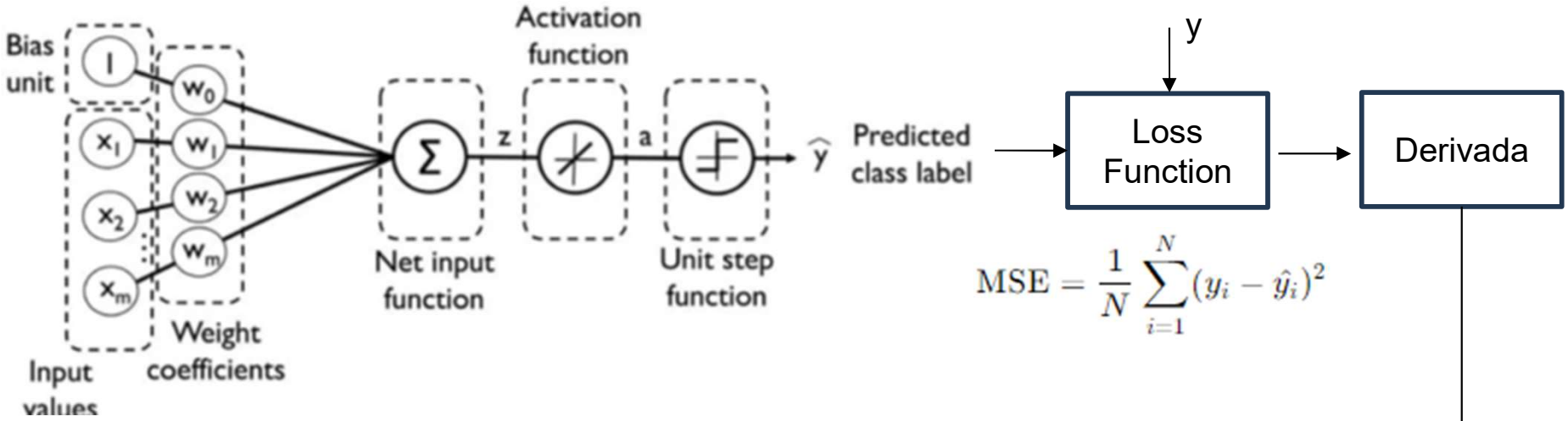
$$Recall = \frac{T_p}{T_p + T_n}$$

$$F_1 = 2\frac{precision \cdot recall}{precision + recall}$$




- **9. Avalia a performance**

**Forward pass**




$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

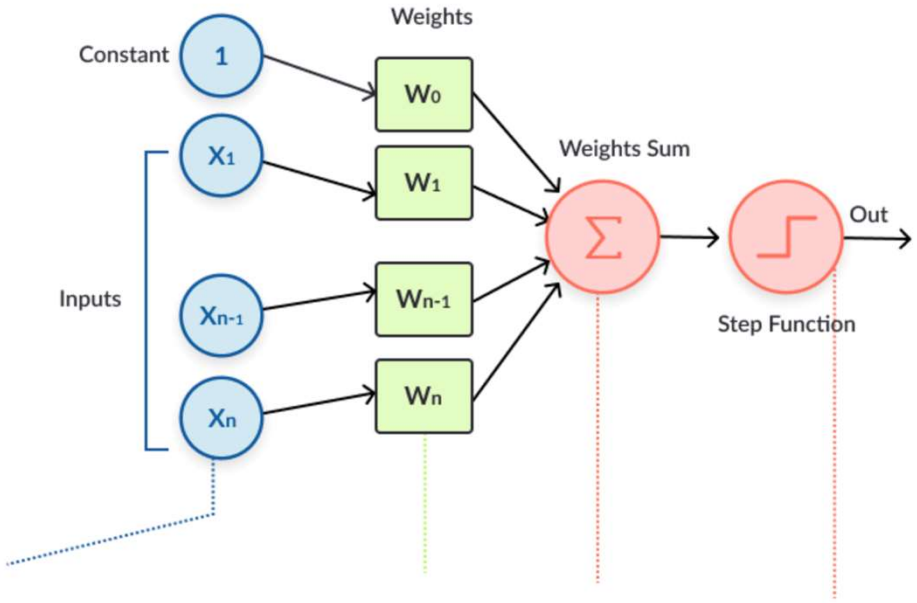
$$w_{12} = w_1 - learning\_rate \times \frac{\partial custo}{\partial w_1}$$

$$\frac{dJ(w)}{dw}$$

**Backward pass**

# Função de ativação

- Decide se um neurônio deve ser ativado ou não (informação relevante ou não)
- Podem ajudar na regularização do modelo, tornando-o menos propenso a overfitting
- Servem para trazer a não-linearidades ao sistema




$$Y = \text{Activation}(\Sigma(weight * input) + bias)$$

Uma rede neural sem função de ativação é essencialmente apenas um modelo de regressão linear.

# Função de ativação

- **Binary Step Function:** função de ativação baseado em limiar (threshold) decide se o neurônio deve ou não ser ativado.

- $$\phi(x) = \begin{cases} 1, \text{if } x \geq 0.5. \\ 0, \text{otherwise.} \end{cases}$$

**f(x) = ax**

A função linear apenas aplica um fator de multiplicação ao valor que recebe.

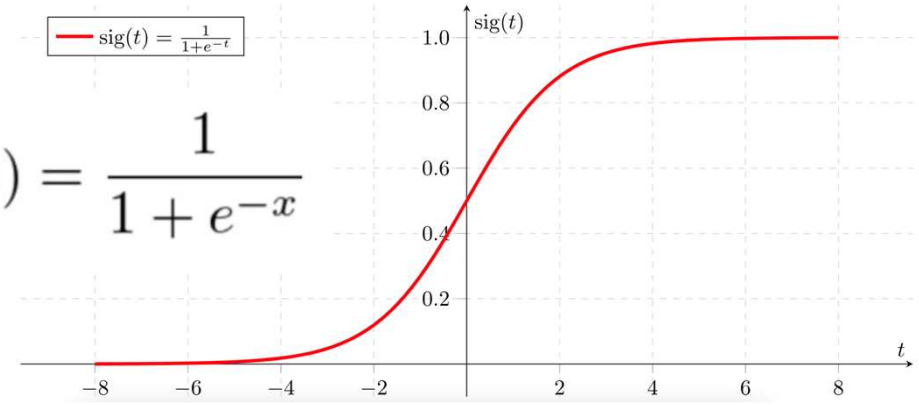
# Função de ativação

A não-linearidade permite que a rede neural modele funções mais complexas e capture padrões nos dados.

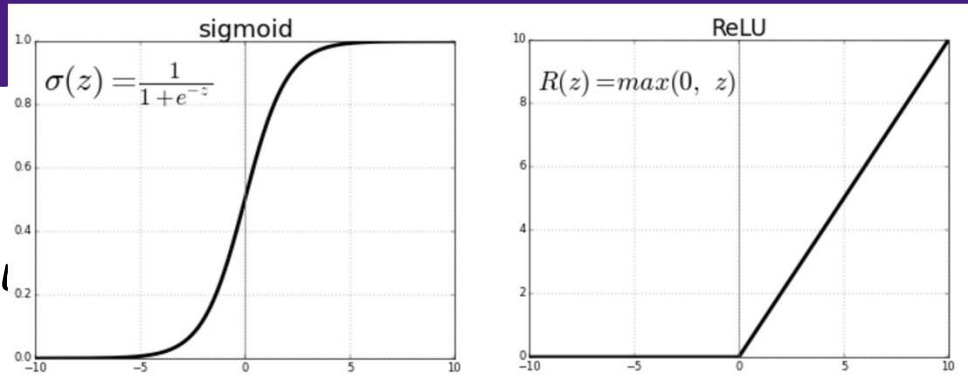
- **Função de ativação Sigmóide**
  - Função não linear suave que é capaz de lidar com apenas duas classes.
  - A função sigmóide permite converter números para **valores entre 0 e 1**
  - **Podemos definir regras para a função de ativação, como:**
    - **Se a saída do neurônio sigmóide for maior que ou igual a 0,5, gera 1;**
    - **Se a saída for menor que 0,5, gera 0.**

```
# Função sigmóide
def sigmoid(Z):
    A = 1 / (1 + np.exp(-Z))
    return A, Z
```

$$\phi(x) = \frac{1}{1 + e^{-x}}$$



# Função de ativação




- **Função de ativação ReLU**
  - ReLU é uma abreviação para *rectified linear*
  - Produz resultados no **intervalo [0, ∞[.**
  - É uma função não linear.

$f(x)$ = max (0, $x$).

```
# Função de ativação ReLu (Rectified Linear Unit)
def relu(Z):
    A = abs(Z * (Z > 0))
    return A, Z
```

  - Essa função não deixa passar funções negativas, transforma em 0. Um valor positivo ela retorna o próprio valor.
  - Ela não ativa todos os neurônios ao mesmo tempo, se entrada for negativa, ela será convertida em zero e o neurônio não será ativado

# Regularização

- **Regularização L1 e L2**
  - L1: soma absoluta dos pesos,
  - L2 soma dos quadrados dos pesos.
- **Dropout**
  - durante o treinamento, alguns neurônios são "desligados" aleatoriamente
- **Early Stopping**
  - monitorar o desempenho do modelo em um conjunto de validação e interromper o treinamento assim que o desempenho começar a piorar