Algoritmo de Backpropagation

1. **Input x:** Seleccionar la arquitectura de la red, definiendo el número de neuronas por capa j y el número de capa L, tipo de funciones de activación por capa de neuronas σ^l. Donde $1 \leq l \leq L$.
2. **Feedforward:** Para cada capa de neuronas propague la entrada desde la entrada hasta la salida (capa de salida). $z^l = w^l a^{l-1} + b^l$ teniendo en cuenta que $a^l = \sigma(z^l)$.
3. **Error en la salida δ^L**: Calcule el error en la salida utilizando el producto Hadamard entre el gradiente de la función de costo respecto a las salidas (si el sistema es multi - output) y la derivada de la función de activación respecto a z^l, matemáticamente esto es

$$\delta^L = \frac{\partial C}{\partial a_j^L}\, \sigma\left(z_j^L\right)$$

$$\delta^L = \nabla_a C \odot \sigma'(z^l)$$

4. **Backpropagate del Error:** Para cada capa de neurona calcule el vector de error computando:

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$$

5. **Calcule los gradientes respectos a los pesos y los bias:**

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l$$

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l$$

6. **Aplique step-descent estocástico para actualizar los pesos y los bias del modelo:**

$$w^l \leftarrow w^l - \frac{\mu}{m} \sum_{\forall x} \frac{\partial C}{\partial w_{jk}^{l,x}}$$

$$b^l \leftarrow b^l - \frac{\mu}{m} \sum_{\forall x} \delta_j^{l,x}$$

7. **Evalué la función de costo o de error actualizados** w^l y b^l:

$C = \frac{1}{2} \sum_{\forall x} \left(y_x - y^*(w^l, b^l, x)\right)^2$ // Regresión

$$C = \sum_{\forall x} \left\{ y_x ln\left(y^*(w^l, b^l, x)\right) + (1 - y_x) ln\left(1 - y^*(w^l, b^l, x)\right) \right\} //$$

Clasificación

8. **Volver al paso 2 en caso de que el algoritmo no converja (Número de iteraciones, se estabiliza C o C es muy pequeño)**