Historia

- 1943: McCulloch y Pitts proponen un modelo matemático de neurona.
- 1958: Rosenblatt desarrolla el perceptrón, un algoritmo de aprendizaje supervisado.
- 1969: Minsky y Papert publican *Perceptrons*, destacando limitaciones del perceptrón.
- 1986: Rumelhart, Hinton y Williams introducen el algoritmo de retropropagación.
- 1989: LeCun y sus colegas aplican redes neuronales (LeNet) a la clasificación de dígitos manuscritos.
- 2010: Popularización del uso de GPUs para el entrenamiento de redes neuronales.
- 2012: Hinton y sus colegas ganan el concurso ImageNet con una red profunda.
- 2017: Vaswani et al. introducen el modelo Transformer.

Definición de perceptrón

Consideremos un problema de aprendizaje supervisado con un conjunto de datos

$$\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N},$$

donde $x^{(i)} \in \mathbb{R}^d$ representa el vector de características y $y^{(i)} \in \mathbb{R}$ la etiqueta asociada.

Un **perceptrón** se define mediante la composición de las siguientes funciones:

- **Función de combinación**

$$a\colon \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R} \to \mathbb{R},$$

  Por ejemplo,

$$a(x, w, b) = w^\top x + b.$$

- **Función de activación**

$$\sigma\colon \mathbb{R} \to \mathbb{R},$$

  Por ejemplo,

$$\sigma(a) = a.$$

- **Función de salida**

$$h\colon \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R} \to \mathbb{R},$$

definida como la composición

$$\hat{y} = h(x, w, b) = \sigma(a(x, w, b)).$$

- **Función de pérdida**

$$L\colon \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}.$$

Por ejemplo,

$$L(y, \hat{y}) = \frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - \hat{y}^{(i)})^2.$$

Con esto,

$$L(y, \hat{y}) = \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - h(x^{(i)}, w, b)\right)^2 = \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - \sigma(a(x^{(i)}, w, b))\right)^2.$$

El objetivo del aprendizaje es encontrar los parámetros (w^*, b^*) que minimizan la función de pérdida sobre el conjunto de datos:

$$(w^*, b^*) = \arg\min_{w\in\mathbb{R}^d,\, b\in\mathbb{R}} \frac{1}{N}\sum_{i=1}^{N} L\left(y^{(i)}, \sigma(a(x^{(i)}, w, b))\right).$$

En el ejemplo, esto se traduce en:

$$(w^*, b^*) = \arg\min_{w\in\mathbb{R}^d,\, b\in\mathbb{R}} \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - (w^\top x^{(i)} + b)\right)^2,$$

lo cual es equivalente a una regresión lineal.

Entrenamiento

El entrenamiento del perceptrón implica ajustar los parámetros w y b utilizando un algoritmo de optimización, como el descenso por gradiente.

Para esto, calculamos las derivadas parciales de la función de pérdida con respecto a los parámetros:

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \sigma}{\partial a} \cdot \frac{\partial a}{\partial w} = \frac{2}{N}\sum_{i=1}^{N}(\hat{y}^{(i)} - y^{(i)}) \cdot 1 \cdot x^{(i)},$$

$$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial b} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \sigma}{\partial a} \cdot \frac{\partial a}{\partial b} = \frac{2}{N}\sum_{i=1}^{N}(\hat{y}^{(i)} - y^{(i)}) \cdot 0 \cdot 1.$$

De esta manera, tomamos valores aleatorios iniciales para w y b, y actualizamos iterativamente:

$$\begin{pmatrix} w \\ b \end{pmatrix} \leftarrow \begin{pmatrix} w \\ b \end{pmatrix} - \eta \begin{pmatrix} \frac{\partial L}{\partial w} \\ \frac{\partial L}{\partial b} \end{pmatrix},$$

donde η es la tasa de aprendizaje. A cada actilización se le denomina **época**.