Ficha Técnica: Funciones de Activación en RNA

Jordi Pozo
I.E.S. Ribera de Castilla

Curso 2025/26

1 Introducción: ¿Por qué son necesarias?

Una red neuronal sin funciones de activación es, matemáticamente, equivalente a un solo modelo de **Regresión Lineal**, sin importar cuántas capas tenga.

Las funciones de activación introducen la **no-linealidad** en el modelo. Esto es lo que permite a la red aprender patrones complejos (como reconocer una cara en una foto o entender el sarcasmo en un texto) que no pueden ser separados por una simple línea recta.

La operación básica de una neurona es:

$$Salida = Activacion(\sum(Pesos \cdot Entradas) + Sesgo)$$

2 Funciones para Capas Ocultas

Estas funciones se utilizan en las capas intermedias ("hidden layers") para transformar los datos antes de pasarlos a la siguiente capa.

2.1 1. ReLU (Rectified Linear Unit)

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

Características:

- Si la entrada es positiva, la deja pasar tal cual. Si es negativa, devuelve cero.
- Es computacionalmente muy eficiente (solo comparaciones, no hay exponenciales).

¿Cuándo usarla?

- **Casi siempre.** Es la opción por defecto para capas ocultas en MLP y redes convolucionales (CNN).
- Ayuda a evitar el problema del "desvanecimiento del gradiente" (vanishing gradient) que sufren Sigmoid y Tanh.

Desventaja: "Dead ReLU". Si una neurona cae en la zona negativa, deja de aprender (gradiente cero). Para solucionar esto existen variantes como *Leaky ReLU*.

2.2 2. Tanh (Tangente Hiperbólica)

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

Características:

- Similar a la Sigmoide, pero su salida va de **-1 a 1**.

- Está centrada en cero, lo que facilita el aprendizaje en las capas siguientes.

¿Cuándo usarla?

- A veces funciona mejor que ReLU en redes recurrentes (RNN) o cuando necesitamos que las activaciones negativas sean explícitas.
- Generalmente preferible a la Sigmoide en capas ocultas.

3 Funciones para la Capa de Salida

La elección de la función en la última capa depende exclusivamente del **tipo de problema** que estemos resolviendo.

3.1 1. Sigmoide (Logistic / Sigmoid)

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

Características:

- Aplasta cualquier valor real a un rango entre **0 y 1**.
- Se interpreta como una probabilidad.

¿Cuándo usarla?

- **Clasificación Binaria:** (Ej. ¿Es spam o no?). Se usa en la neurona de salida única.
- **NO usar en capas ocultas** (hoy en día) debido al desvanecimiento del gradiente.

3.2 2. Softmax

$$f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

Características:

- Toma un vector de números reales y lo convierte en una distribución de probabilidad.
- La suma de todas las salidas es siempre **1.0**.

¿Cuándo usarla?

- **Clasificación Multiclase:** (Ej. Clasificar dígitos 0-9, o especies de pingüinos).
- La neurona con el valor más alto es la clase predicha.

3.3 3. Lineal (Identity)

$$f(x) = x$$

Características:

- No hace nada, deja pasar el valor tal cual.

¿Cuándo usarla?

- **Regresión:** Cuando queremos predecir un valor numérico continuo (Ej. Precio de una casa).

4 Resumen Visual

Figure 1: Representación de las funciones: Sigmoide (curva S), Tanh (curva S centrada en 0) y ReLU (linea plana en negativos, diagonal en positivos).

5 Implementación en `scikit-learn`

En `scikit-learn`, la clase `MLPClassifier` permite definir la función de las capas ocultas con el parámetro `activation`.

Nota: En scikit-learn, la función de salida se determina automáticamente según el target (y), pero en librerías como Keras/TensorFlow debemos definirla manualmente.

```python
from sklearn.neural_network import MLPClassifier

# --- Caso 1: Capas ocultas con ReLU (Estandar) ---
# activation='relu' afecta a las capas OCULTAS.
# La salida sera Softmax automaticamente si hay >2 clases.
mlp_standard = MLPClassifier(hidden_layer_sizes=(50,),
activation='relu',
solver='adam')

# --- Caso 2: Capas ocultas con Tanh ---
# A veces util si los datos estan normalizados entre -1 y 1.
mlp_tanh = MLPClassifier(hidden_layer_sizes=(50,),
activation='tanh',
solver='adam')

# --- Caso 3: Capas ocultas con Sigmoide (Logistic) ---
# Raramente usado hoy en dia para capas profundas,
# pero util para fines didacticos.
mlp_legacy = MLPClassifier(hidden_layer_sizes=(50,),
activation='logistic',
solver='sgd')

# --- Como saber que funcion de salida uso? ---
# Scikit-learn lo gestiona internamente:
# .out_activation_ : Devuelve el nombre de la funcion de salida
# 'logistic' -> para binaria
# 'softmax'  -> para multiclase
# 'identity' -> para regresion (MLPRegressor)

# Ejemplo de uso:
# mlp_standard.fit(X_train, y_train)
# print(f"Funcion de salida usada: {mlp_standard.out_activation_}")

```

Listing 1: Configuracion de funciones de activacion en MLP.

6 Guía Rápida de Selección

Capa	**Tipo de Problema**	**Función Recomendada**
Oculta	(Cualquiera)	**ReLU** (Empezar siempre aquí)
Oculta	RNN / Datos -1 a 1	Tanh
Salida	Clasificación Binaria	**Sigmoide**
Salida	Clasificación Multiclase	**Softmax**
Salida	Regresión (Valor numérico)	**Lineal (Identity)**

Table 1: Tabla resumen para la selección de funciones de activación.