Redes Neuronales Artificiales: Fundamentos y Aplicaciones

Prof. Dr. BARSEKH-ONJI Aboud

Universidad Anáhuac México
Facultad de Ingeniería

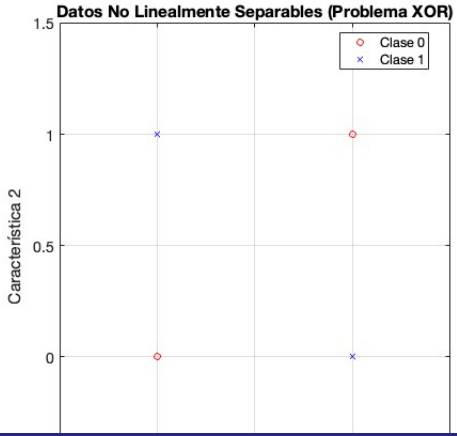
22 de diciembre de 2025

Contenido

¿Por qué Redes Neuronales Artificiales?

- Ya conocen algoritmos de Clasificación y Regresión (SVM, Árboles, etc.).
- También han explorado la inspiración biológica (Algoritmos Genéticos).
- Los problemas complejos a menudo requieren modelos que aprendan patrones no lineales.
- Las ANNs son potentes para modelar relaciones complejas en datos de alta dimensión.

ejemplo: Datos No Linealmente Separables



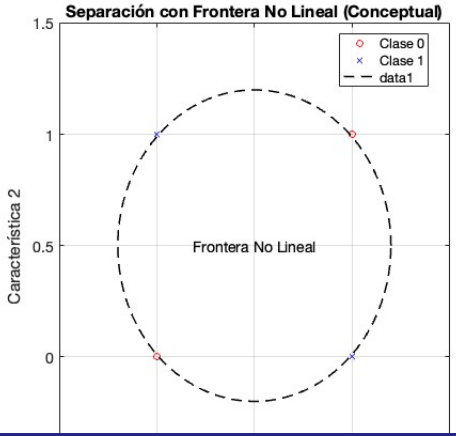
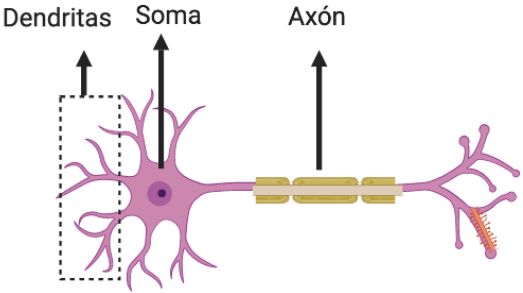
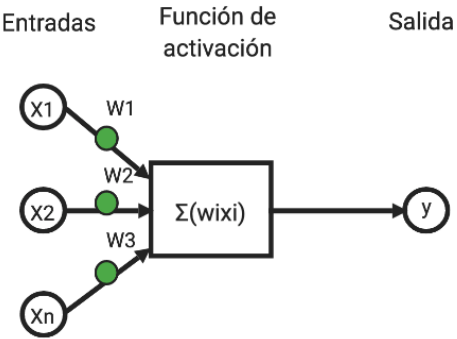


Diagrama de una red neuronal

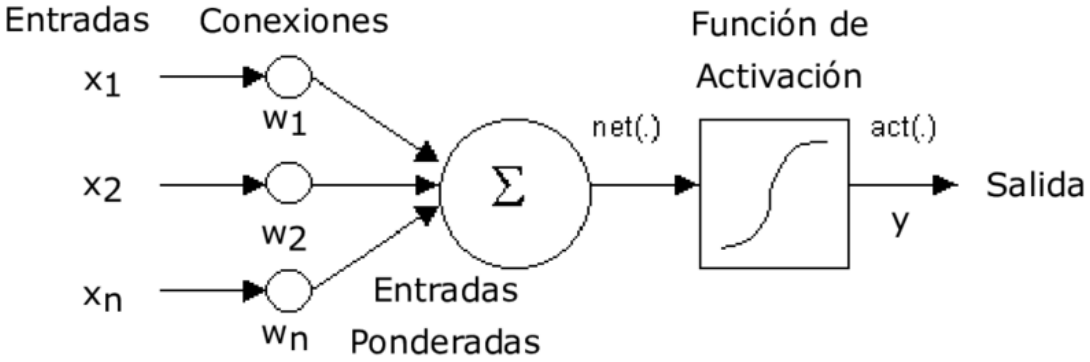




La Neurona Artificial: La Unidad Básica

- Inspirada en la neurona biológica (dendritas, soma, axón).
- Modelo matemático simple:
- Entradas: $x_1, x_2, ..., x_n$
- Pesos: $w_1, w_2, ..., w_n$ (fuerza de la conexión)
- Suma Ponderada: $z = \sum_{i=1}^{n} w_i x_i + b$ (con bias b)
- Función de Activación: f
- Salida: $y = f(z)$

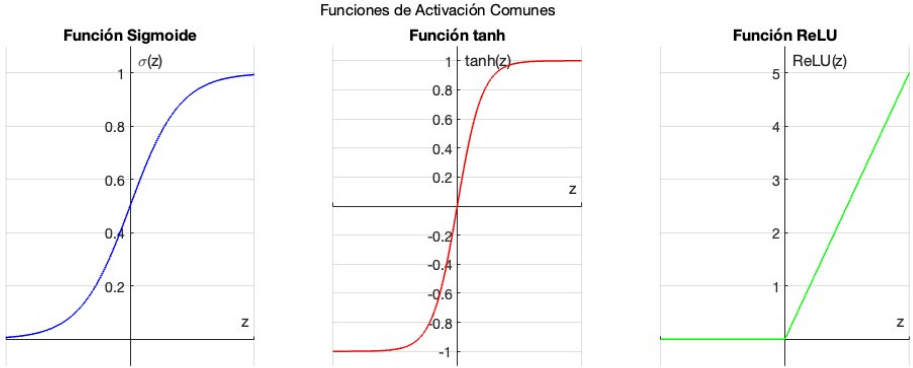
Diagrama de una red neuronal



Funciones de Activación Comunes

- Introducen no-linealidad, permitiendo aprender patrones complejos.
- Algunas comunes:
- **Sigmoide (Logística):** $f(z) = \frac{1}{1+e^{-z}}$
  (Salida entre 0 y 1)
- **Tangente Hiperbólica (tanh):** $f(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$
  (Salida entre -1 y 1)
- **ReLU (Rectified Linear Unit):** $f(z) = \text{máx}(0, z)$
  (Muy popular, eficiente)

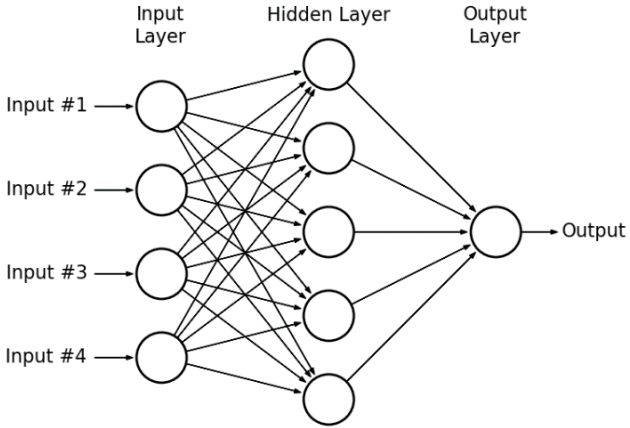
Funciones de Activación Comunes



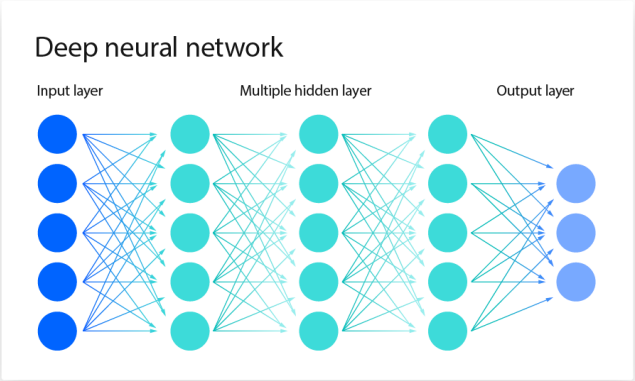
Arquitectura de la Red Neuronal: Capas

- Las neuronas se organizan en capas.
- **Capa de Entrada:** Recibe los datos brutos.
- **Capas Ocultas:** Procesan la información (una o más). Aquí ocurre la magia de aprender representaciones complejas.
- **Capa de Salida:** Produce el resultado final de la red.
- La 'profundidad' de la red se refiere al número de capas ocultas.

Arquitectura de la Red Neuronal: Capas



Arquitectura de la Red Neuronal: Capas



Tipos de Arquitecturas (Básicas)

- **Perceptrón:** Una sola capa de salida (sin capa oculta). Limitado a problemas linealmente separables.
- **Perceptrón Multicapa (MLP):** Con una o más capas ocultas.
  - Arquitectura 'Feedforward': La información fluye solo hacia adelante.
  - Capaces de aproximar cualquier función continua (Teorema de Aproximación Universal).
- Existen otras arquitecturas para tareas específicas (CNN, RNN - para cursos avanzados).

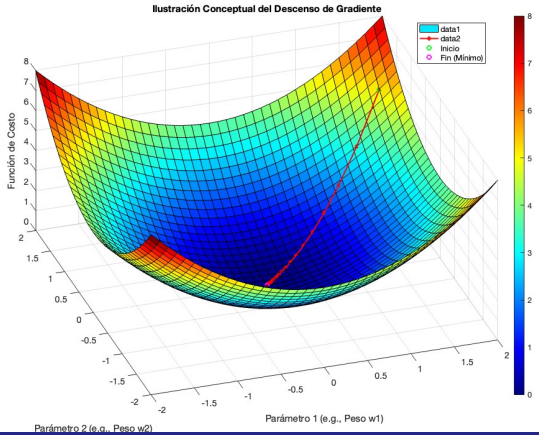
¿Cómo Aprende una Red Neuronal? Entrenamiento

- El objetivo es ajustar los pesos (w) y bias (b).
- Se busca minimizar el error entre la salida de la red y la salida deseada.
- **Función de Costo (o Pérdida):** Mide qué tan mal lo está haciendo la red (ej. MSE, Cross-Entropy).
- Queremos encontrar los w y b que minimizan esta función de costo.

Descenso de Gradiente: El Motor del Aprendizaje

- Algoritmo de optimización iterativo.
- Ajusta los parámetros (w, b) en la dirección opuesta al gradiente de la función de costo.
- El gradiente indica la dirección de máximo aumento de la función.
- Queremos 'descender' en la superficie de costo.
- **Tasa de Aprendizaje (α):** Controla el tamaño del paso en cada iteración.

Ilustración conceptual del descenso de gradiente en 2D



Backpropagation: Calculando los Gradientes

- Algoritmo eficiente para calcular los gradientes en redes multicapa.
- Proceso en dos fases:
- **1. Forward Pass:** Calcular la salida de la red para una entrada dada.
- **2. Backward Pass:** Calcular el error en la capa de salida y propagarlo hacia atrás.
- Se usa la regla de la cadena del cálculo para encontrar la contribución de cada peso/bias al error total.
- Permite saber cómo ajustar cada peso y bias para reducir el error.

Hiperparámetros Importantes

- Parámetros que NO se aprenden, se configuran antes del entrenamiento.
- Influyen crucialmente en el rendimiento de la red.
- Ejemplos:
  - Número de capas ocultas y neuronas por capa.
  - Función de activación.
  - Tasa de aprendizaje (α).
  - Número de épocas (iteraciones de entrenamiento).
  - Tamaño del batch.
- La validación cruzada ayuda a seleccionarlos (conexión con conocimiento previo).

Sobreajuste (Overfitting) y Mitigación

- La red aprende demasiado bien los datos de entrenamiento, pero generaliza mal a datos nuevos.
- Señal de que el modelo es demasiado complejo para la cantidad de datos.
- Técnicas para mitigar (breve mención):
  - Regularización (L1, L2).
  - Dropout.
  - Detención temprana (Early Stopping).

Aplicaciones de las ANNs

- Extremadamente versátiles.
- **Reconocimiento de Imágenes:** Clasificación, detección de objetos.
- **Procesamiento del Lenguaje Natural (PLN):** Traducción, análisis de sentimiento.
- **Predicción y Series de Tiempo:** Finanzas, demanda.
- **Robótica y Control:** Navegación, manipulación.
- **Medicina:** Diagnóstico basado en imágenes.
- **Juegos:** Superando a expertos humanos.

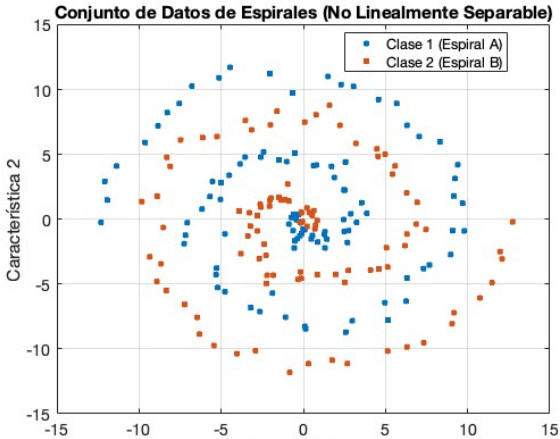
ANNs en MATLAB

- MATLAB cuenta con herramientas poderosas para ANNs.
- **Neural Network Toolbox** (o Deep Learning Toolbox).
- Permite:
  - Crear diferentes arquitecturas de red.
  - Configurar parámetros y funciones de activación.
  - Entrenar redes con varios algoritmos de optimización.
  - Evaluar y visualizar el rendimiento.
- Funciones clave: `feedforwardnet`, `train`, `sim`, `plotperform`.

Ejemplo en MATLAB

1. **Preparación de Datos:**
   - Generar o cargar el conjunto de datos (Ej: Espirales).
   - Organizar entradas (X) y salidas objetivo (T).
2. **Creación de la Red:**
   - Definir la arquitectura (Ej: `feedforwardnet`).
   - Especificar número de neuronas en capas ocultas.
   - Configurar funciones de activación y división de datos.
3. **Entrenamiento:**
   - Usar la función `train(net, X, T)`.
   - La red ajusta pesos y bias iterativamente (Descenso de Gradiente, Backpropagation).
4. **Evaluación:**
   - Usar la función `sim(net, X_test)` en datos no vistos.
   - Calcular métricas de rendimiento (Ej: Precisión, Matriz de Confusión).
   - Visualizar resultados (Ej: `plotperform`, `plotconfusion`).

Conjunto de datos



Ventajas y Desventajas

Ventajas:

- Modelado de relaciones no lineales complejas.
- Robustas a datos ruidosos/incompletos.
- Alta capacidad de generalización.
- Útiles para datos de alta dimensión.

Desventajas:

- 'Caja Negra' (dificultad de interpretación).
- Requieren grandes cantidades de datos.
- Entrenamiento computacionalmente costoso.
- Selección de hiperparámetros a veces empírica.

Conclusiones y Futuro

- Las ANNs son modelos potentes inspirados biológicamente.
- Compuestas por neuronas artificiales, organizadas en capas.
- Aprenden ajustando pesos y bias, típicamente con Descenso de Gradiente y Backpropagation.
- Amplias aplicaciones en diversos campos.
- MATLAB es una herramienta útil para experimentar con ANNs.
- El campo del Deep Learning sigue en constante evolución.