Universidad Nacional Mayor de San Marcos

Facultad de Ingeniería de Sistemas e Informática

Escuela Profesional de Ingeniería de Sistemas

Inteligencia Artificial

Exposición de la semana 9 del silabo: Fundamentos del Deep Learning

Docente:

Luis Guerra Grados

Alumnos:

- Gonzales Marca Jesus Miguel
- Suarez Bautista Pablo Israel
- Lliuya Villagaray Joaquin Lazaro
- Pachas Oshiro Kojiro Andre

Semestre académico 2025-I

ÍNDICE

1. Fundamentos del Deep Learning

1.1. ¿Qué es el Deep Learning?

El Deep Learning o aprendizaje profundo es una subrama del Machine Learning que utiliza redes neuronales artificiales con múltiples capas (capas profundas) para modelar y resolver problemas complejos como reconocimiento de voz, imágenes, texto y más. Estas redes aprenden directamente de los datos, sin necesidad de que un humano defina explícitamente las reglas o características.

1.2. Evolución histórica

- **Década de 1940-1950**: Se introdujo el concepto de la neurona artificial (modelo de McCulloch-Pitts) y el perceptrón simple (Rosenblatt, 1958).
- **Década de 1980**: Se desarrolló el algoritmo de retropropagación (backpropagation), clave para entrenar redes multicapa.
- **Década de 2000-2010**: Gracias al aumento del poder computacional (GPU) y grandes volúmenes de datos (*big data*), el deep learning resurgió con gran éxito.
- **Actualidad**: Es el núcleo de tecnologías como GPT, ChatGPT, DALL·E, sistemas de recomendación, visión por computador y asistentes virtuales.

Características principales

- **Arquitecturas profundas**: Redes con muchas capas ocultas que permiten representar funciones altamente no lineales.
- **Aprendizaje jerárquico**: Aprende representaciones desde lo simple (bordes en una imagen) hasta lo complejo (rostros, objetos).
- **Extracción automática de características**: No requiere ingeniería manual de atributos.
- **Gran capacidad de generalización**: Cuando se entrena con suficiente información, supera a muchos métodos tradicionales.

1.3. Cómo aprende una red neuronal

Cada neurona recibe entradas, las pondera mediante pesos, las suma y aplica una función de activación.

La red predice una salida, compara con el valor real (función de pérdida), y ajusta los pesos usando *backpropagation* y *gradient descent*.

Este proceso se repite en múltiples *epochs* hasta que la red minimiza el error y generaliza bien sobre nuevos datos.

2. Comparación con el Machine Learning clásico

2.1. Diferencias técnicas y conceptuales

2.1.1. Definición de Machine Learning clásico

El Machine Learning (ML) es una aplicación de inteligencia artificial en la que los programas informáticos utilizan algoritmos para encontrar patrones en los datos. Donde estos dependen de sí mismos sin la interacción humana.Este modelo se encuentra en casi todos los avances tecnológicos y aplicaciones de inteligencia artificial que hay en el mercado. Algunos ejemplos son:

- Cuando tu proveedor de correo electrónico reconoce un email como spam;
- Cuando las redes sociales te recomiendan amigos, grupos y videos;
- Cuando el GPS anticipa qué partes de tu ruta tendrán mucho tráfico y te redireccionan utilizando estos algoritmos.

2.1.2. Diferencias principales

Por definición:

Machine Learning (ML) es un subcampo de la inteligencia artificial que permite a las máquinas aprender de los datos mediante algoritmos estadísticos. Los modelos de ML requieren que el programador defina qué características o atributos son importantes para que el algoritmo los utilice en el proceso de aprendizaje.

Deep Learning (DL) es una subárea del Machine Learning que utiliza redes neuronales profundas (deep neural networks) para aprender representaciones jerárquicas de los datos. El modelo es capaz de aprender automáticamente las características relevantes sin intervención humana.

Por tipo de extracción:

ML tradicional: Requiere **extracción manual de características**. El rendimiento depende en gran parte de la calidad y la relevancia de las variables seleccionadas (por ejemplo: bordes, color, tamaño en visión computacional).

DL: Utiliza **extracción automática de características** a través de múltiples capas (por ejemplo: convoluciones en una CNN) que identifican patrones desde lo más simple a lo más complejo.

Por arquitectura del algoritmo

ML: Utiliza algoritmos como Random Forest, K-Nearest Neighbors, entre otros. Son algoritmos generalmente más simples y con menos parámetros.

DL: Utiliza arquitecturas complejas como redes neuronales convolucionales (CNN), redes recurrentes (RNN), etc. Estas redes pueden tener millones de parámetros y requieren entrenamiento profundo y optimización intensiva.

Característica	**Aprendizaje Supervisado Clásico**	**Deep Learning**
Extracción de características	Manual (Ingeniería de atributos)	Automática (aprende de datos)
Requiere muchos datos	No necesariamente	Sí, requiere grandes volúmenes
Rendimiento en tareas complejas	Limitado	Muy alto con datos adecuados
Interpretabilidad	Alta (modelos más simples)	Baja
Tiempo de entrenamiento	Rápido (en general)	Lento y computacionalmente costoso

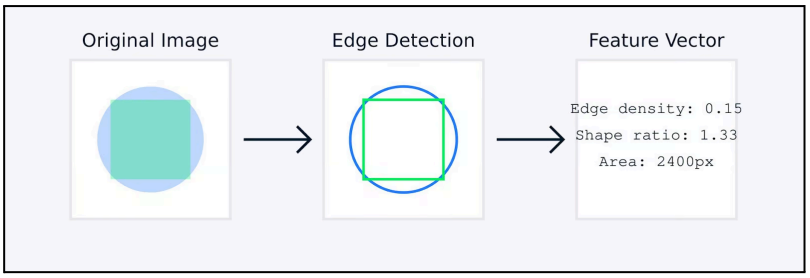
2.2. Extracción de características: manual vs automática

Los métodos de extracción de rasgos pueden clasificarse a grandes rasgos en dos enfoques principales: la ingeniería manual de rasgos y la extracción automatizada de rasgos.

2.2.1. Ingeniería manual de funciones

La ingeniería manual de rasgos implica utilizar la experiencia del dominio para identificar y crear rasgos relevantes a partir de datos brutos. Este enfoque práctico se basa en nuestra comprensión del problema y de los datos para elaborar características significativas.

Por ejemplo, en procesamiento de imágenes la ingeniería manual de rasgos puede implicar técnicas como la detección de bordes para identificar los límites del objeto, histogramas de color para captar la distribución del color, análisis de texturas para cuantificar patrones y descriptores de forma para caracterizar la geometría del objeto.



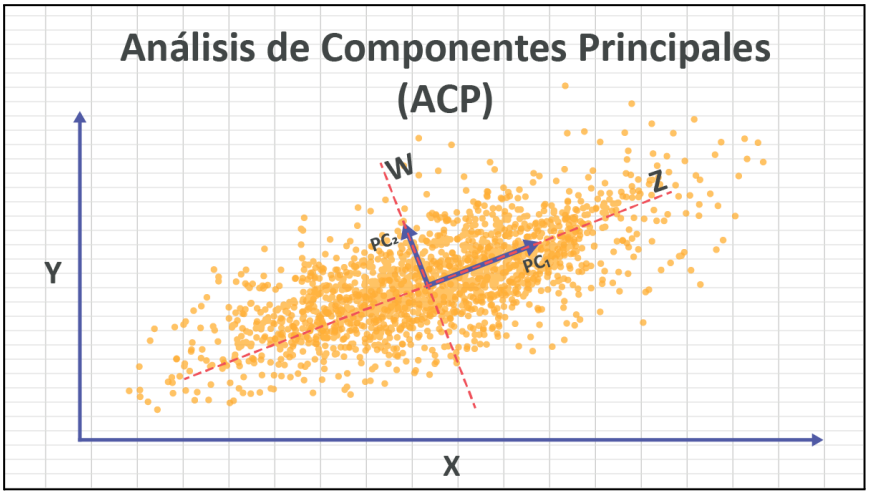
Estas técnicas, guiadas por la experiencia en el dominio, aumentan la calidad de la representación de los datos y pueden mejorar significativamente el rendimiento del modelo.

2.2.2. Extracción automatizada de rasgos

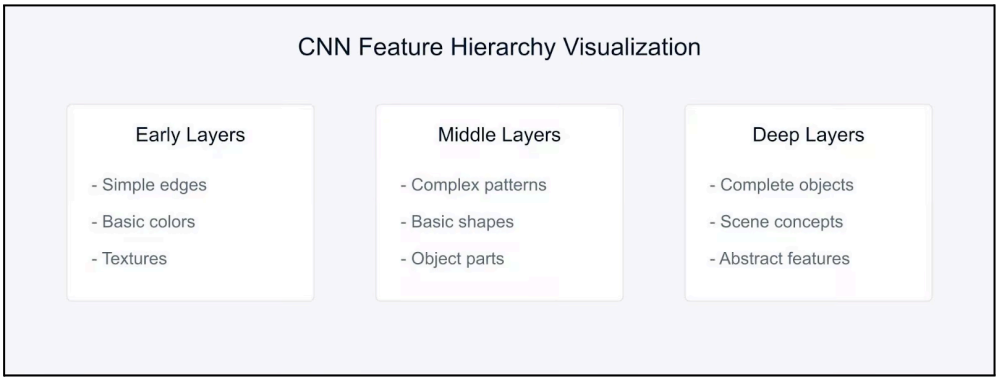
La extracción automatizada de rasgos utiliza algoritmos para descubrir y crear rasgos sin orientación humana explícita. Estos métodos son especialmente útiles cuando se trata de conjuntos de datos complejos en los que la ingeniería manual de rasgos puede resultar poco práctica o ineficaz.

Análisis de Componentes Principales (ACP): Transforma los datos en un conjunto de componentes no correlacionados, en el que cada componente capta la máxima varianza restante. Este enfoque es especialmente útil para reducción de la

dimensionalidad ya que conserva la información esencial de los datos simplificando su estructura.



Para el caso de imágenes, las CNN aprenden características a través de su estructura jerárquica. En las primeras capas, detectan elementos visuales básicos, como bordes y colores. A continuación, las capas intermedias combinan estos elementos para reconocer patrones y formas, mientras que las capas más profundas captan objetos complejos y permiten comprender la escena.



2.3. Requisitos de datos y procesamiento

Los requerimientos para cada modelo de lenguajes varían conforme a la necesidad del problema. De la siguiente manera serían distribuido estos modelos:

Enfoque	Requisitos de Datos	Requisitos de Procesamiento

SVM (Machine Learning clásico)	Requiere menos datos (puede funcionar con cientos o pocos miles de muestras). Necesita que las características sean extraídas manualmente.	• Computacionalmente ligero. • Entrenamiento rápido en CPU.
CNN (Deep Learning)	Requiere gran cantidad de datos para generalizar (decenas de miles o millones). Extrae características automáticamente.	• Alto costo computacional. • Necesita GPU para un entrenamiento eficiente. • Entrenamiento más lento pero escalable.

2.4. Casos de ejemplo de extracción de características: SVM vs CNN

Para este caso, se ejemplifica la extracción de características manualmente y automáticamente. Teniendo en cuenta las diferentes técnicas de extracción, las cuales son:

- Extracción de características de la imagen
- Extracción de características de audio
- Extracción de características de series temporales

Para este ejemplo se realizará la **extracción de características de la imagen** en sus diferentes tipos de extracción.

2.4.1. Para SVM

Se importarán las librerías cv2, numpy y matplotlib.pyplot para subir un archivo jpg y para realizar las transformaciones necesarias a la imagen.

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

Una vez teniendo cargada una imagen, se realiza un prueba

```
image = cv2.imread('godzilla.jpg')
# Check if the image was loaded successfully
```

```
if image is None:
    print("Error: Could not load the image. Please
ensure 'godzilla.jpg' is in the correct directory.")
else:
    # Convert BGR to RGB (OpenCV loads in BGR format)
    image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    # Display the original image
    plt.imshow(image_rgb)
    plt.title('Original Image')
    plt.axis('off')
    plt.show()
```

Antes de aplicar la detección de bordes, necesitamos preprocesar nuestra imagen. Lo hacemos de la siguiente manera:

```
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
blurred = cv2.GaussianBlur(gray_image, (5, 5), 0)
```

Por último, aplicamos el algoritmo de detección de bordes Canny y visualicemos los resultados:

```
edges = cv2.Canny(blurred, threshold1=100,
threshold2=200)
# Display the results
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.imshow(image_rgb)
plt.title('Original Image')
plt.axis('off')
plt.subplot(1, 2, 2)
plt.imshow(edges, cmap='gray')
plt.title('Edge Detection')
plt.axis('off')
plt.tight_layout()
plt.show()
```

Resultando así:

2.4.2. Para CNN

Se importarán las librerías cv2, numpy y matplotlib.pyplot para subir un archivo jpg y para realizar las transformaciones necesarias a la imagen. Además, se implementarán librerías para la ejecución de la IA.

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras import Input, Model
from tensorflow.keras.layers import Conv2D, MaxPooling2D
```

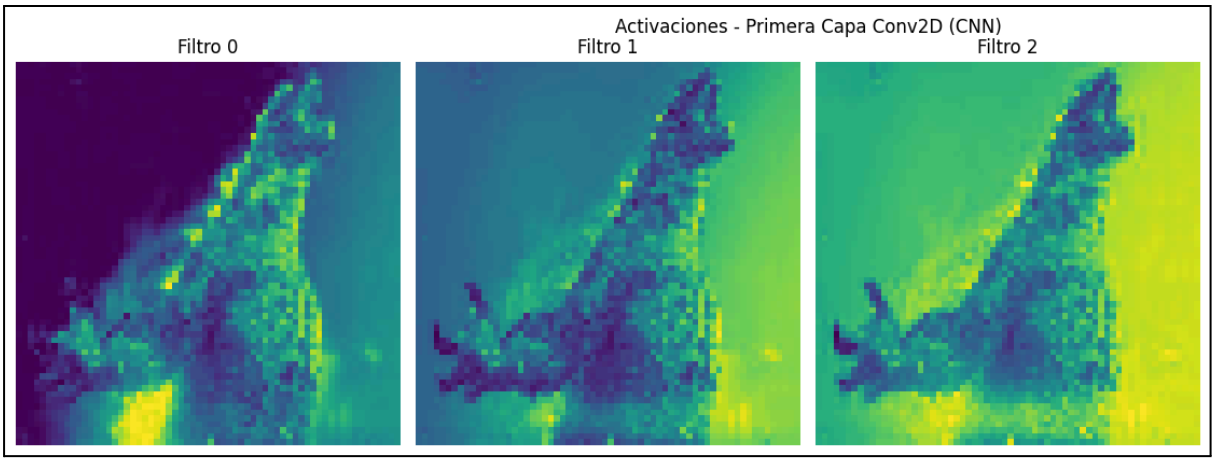
Este código utiliza una **red neuronal convolucional (CNN)** para analizar y visualizar las activaciones de los filtros en la primera capa convolucional.

```
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
image_resized = cv2.resize(image_rgb, (64, 64))  #
Tamaño fijo para la CNN
image_input = image_resized / 255.0              #
Normalizar
image_input = np.expand_dims(image_input, axis=0)  #
(1, 64, 64, 3)
# 3. Construir red CNN con Functional API
input_img = Input(shape=(64, 64, 3))
x = Conv2D(16, (3, 3), activation='relu')(input_img)
x = MaxPooling2D((2, 2))(x)
x = Conv2D(32, (3, 3), activation='relu')(x)
x = MaxPooling2D((2, 2))(x)
x = Conv2D(64, (3, 3), activation='relu')(x)
model = Model(inputs=input_img, outputs=x)
# 4. Crear modelo intermedio para ver activaciones
activation_model = Model(inputs=input_img,
outputs=[layer.output for layer in model.layers])
# 5. Obtener activaciones
activations = activation_model.predict(image_input)
# 6. Visualizar activaciones de la primera capa
(Conv2D con 16 filtros)
first_layer_activation = activations[0]
num_filters = first_layer_activation.shape[-1]
plt.figure(figsize=(15, 8))
for i in range(min(num_filters, 8)):  # Mostrar
primeros 8 filtros
  plt.subplot(2, 4, i + 1)
  plt.imshow(first_layer_activation[0, :, :, i],
cmap='viridis')
  plt.title(f'Filtro {i}')
  plt.axis('off')
```

```
plt.suptitle('Activaciones - Primera Capa Conv2D
(CNN)')
plt.tight_layout()
plt.show()
```

Extrae las salidas de cada capa y muestra los mapas de activación de los primeros 8 filtros de la primera capa convolucional. Cada subplot representa cómo un filtro específico responde a características de la imagen (bordes, texturas, etc.)

Este algoritmo busca entender qué características aprenden los filtros iniciales de la CNN, clave para tareas como clasificación o detección de objetos.



3. Aplicaciones y Limitaciones del Deep Learning

3.1. Aplicaciones destacadas (visión, lenguaje, voz)

El Deep Learning ha revolucionado múltiples campos mediante la automatización de tareas cognitivas complejas que anteriormente eran dominio exclusivo de humanos. Entre las aplicaciones más destacadas se encuentran:

- **Visión por computadora**: Las redes neuronales convolucionales (CNNs) permiten identificar objetos, rostros, patrones y escenas en imágenes y videos. Se utilizan en vigilancia, automóviles autónomos, diagnóstico médico por imágenes, reconocimiento facial, y más.
- **Procesamiento del lenguaje natural (NLP)**: Modelos como BERT, GPT y T5 permiten comprender, traducir, resumir y generar texto humano. Se emplean en asistentes virtuales, chatbots, análisis de sentimiento y motores de búsqueda.

- **Reconocimiento de voz y síntesis de audio**: Modelos como DeepSpeech de Mozilla o Whisper de OpenAI permiten convertir voz en texto con alta precisión. A su vez, redes generativas como WaveNet permiten sintetizar voces humanas casi indistinguibles de las reales.

3.2. Ejemplos reales (industria, salud, transporte, etc.)

- Industria: En la manufactura, se emplean redes neuronales para inspección de calidad mediante imágenes, predicción de mantenimiento de maquinaria (predictive maintenance) y optimización de la cadena de suministro.
- Salud: El deep learning permite diagnosticar enfermedades a partir de radiografías, resonancias o incluso patrones en el habla y escritura. Ejemplo: detección de cáncer de mama con precisión superior a radiólogos (Estudio de Google Health, 2020).
- Transporte: En los vehículos autónomos, los sistemas de conducción utilizan deep learning para interpretar señales de tráfico, peatones, carriles y obstáculos en tiempo real.
- Finanzas: Se emplea para detectar fraudes, analizar riesgo crediticio, predecir movimientos del mercado y automatizar asesoramiento financiero mediante asistentes virtuales.
- Agricultura: Uso de drones con visión por computadora para detectar plagas, enfermedades o deficiencias en cultivos.

3.3. Limitaciones del Deep Learning:

Requiere grandes cantidades de datos

Para que los modelos generalicen bien, necesitan grandes volúmenes de datos etiquetados.

Esto no solo es costoso, sino que en algunos sectores (como medicina o derecho) puede

ser difícil de obtener por privacidad o escasez.

Alta demanda computacional

El entrenamiento de modelos profundos requiere GPUs o TPUs de alto rendimiento,

consumo energético elevado y tiempos prolongados, lo que representa una barrera para muchos desarrolladores e instituciones pequeñas.

Problemas de explicabilidad

Muchos modelos funcionan como cajas negras: generan predicciones precisas, pero no es claro cómo llegaron a esas conclusiones. Esto es un riesgo en sectores como salud, justicia o finanzas, donde la interpretabilidad es clave.

3.4. Ética y riesgos potenciales

El uso indiscriminado o mal controlado del Deep Learning puede acarrear riesgos sociales, éticos y legales:

Privacidad y vigilancia

Aplicaciones como el reconocimiento facial masivo o la predicción de comportamiento pueden invadir la privacidad y generar sociedades de vigilancia.

Uso malicioso de IA

Desde la creación de deepfakes hasta la automatización de ciberataques, los modelos de deep learning pueden ser utilizados para fines dañinos o ilegales.

4. Redes Neuronales Artificiales

4.1. ¿Qué es una red neuronal artificial?

Las redes neuronales artificiales surgieron como un intento de imitar el funcionamiento del cerebro humano. En el cerebro, las neuronas biológicas se conectan a través de sinapsis formando una compleja red. Una neurona biológica:

- Recibe señales químicas a través de las dendritas
- Procesa estas señales en el cuerpo celular (soma)
- Si la suma de señales supera cierto umbral, genera un impulso eléctrico
- Transmite este impulso a otras neuronas a través del axón

Este mecanismo básico de "recibir, procesar, y transmitir" es lo que las redes neuronales artificiales intentan simular matemáticamente.

Evolución histórica

- 1943: Warren McCulloch y Walter Pitts proponen el primer modelo matemático de una neurona artificial
- 1958: Frank Rosenblatt desarrolla el "Perceptrón", el primer algoritmo de aprendizaje para redes neuronales
- 1969: Minsky y Papert publican "Perceptrons", señalando limitaciones (como la imposibilidad de resolver el problema XOR)
- 1980s: Desarrollo del algoritmo de retropropagación (backpropagation), revitalizando el campo
- 1990s-2000s: Avances incrementales pero limitados por capacidad computacional
- 2010s-presente: Explosión del aprendizaje profundo (deep learning) gracias a:

Funcionamiento

Una red neuronal artificial funciona como un sistema de procesamiento de información que:

1. Recibe datos de entrada en forma de vectores o matrices
2. Transforma estos datos a través de operaciones matemáticas en cada capa
3. Produce una salida que representa la solución al problema (clasificación, predicción, etc.)

El proceso de transformación implica:

- Multiplicación matricial (datos × pesos)
- Adición de términos de sesgo
- Aplicación de funciones no lineales (activación)

Tipos de redes neuronales artificiales

1. Perceptrón multicapa (MLP)

- La arquitectura más básica y tradicional
- Neuronas organizadas en capas totalmente conectadas
- Buenas para problemas de clasificación y regresión simples
- Limitaciones con datos secuenciales o espaciales

2. Redes neuronales convolucionales (CNN)

- Especializadas en procesamiento de imágenes y datos con estructura espacial
- Utilizan filtros (kernels) que se desplazan por los datos
- Detectan características locales como bordes, texturas y patrones
- Ejemplo: ResNet, VGG, Inception (usadas en reconocimiento facial, detección de objetos)

3. Redes neuronales recurrentes (RNN)

- Diseñadas para datos secuenciales (texto, audio, series temporales)
- Tienen "memoria" de entradas anteriores
- Variantes importantes: LSTM (Long Short-Term Memory) y GRU
- Aplicaciones: traducción automática, generación de texto, análisis de sentimientos

4. Redes generativas adversarias (GAN)

- Compuestas por dos redes que compiten entre sí
- Un generador crea datos sintéticos
- Un discriminador intenta distinguir entre datos reales y generados
- Aplicaciones: generación de imágenes realistas, deepfakes, arte digital

Diferencias clave entre redes neuronales biológicas y artificiales

Aspecto	Redes Neuronales Biológicas	Redes Neuronales Artificiales
Complejidad	Extremadamente complejas (~86 mil millones de neuronas con ~1000 billones de conexiones)	Simplificadas (desde cientos hasta billones de parámetros)
Velocidad	Relativamente lentas (~100Hz)	Rápidas (miles de millones de operaciones por segundo)
Eficiencia energética	Altamente eficientes (~20W para todo el cerebro)	Ineficientes (pueden requerir kilowatts)
Aprendizaje	Continuo y adaptativo	Por lotes y con datos específicos
Robustez	Muy tolerantes a fallos	Sensibles a pequeñas perturbaciones
Procesamiento	Paralelo por naturaleza	Simulación de paralelismo

4.2. Componentes: capas, pesos, funciones de activación

Capas

Las redes neuronales están organizadas en capas, cada una con un propósito específico en el procesamiento de información:

Capa de entrada

- Función: Recibe los datos originales del problema
- Características:
    - Tiene tantas neuronas como características (features) de entrada
    - No realiza cálculos, solo distribuye la información
    - Ejemplo: en una red para reconocer dígitos escritos a mano (MNIST), tendría 784 neuronas (una por cada píxel de la imagen 28×28)

Capas ocultas

- Función: Realizan la extracción y transformación de características
- Características:
    - Su número y tamaño determinan la capacidad de la red
    - Cada capa adicional permite aprender representaciones más abstractas

    - Las redes con muchas capas ocultas se denominan "profundas" (deep learning)
    - Diseño variable según el problema (densas, convolucionales, recurrentes)

Capa de salida

- Función: Produce el resultado final de la red
- Características:
    - Su estructura depende del tipo de problema:
        - Clasificación binaria: 1 neurona con activación sigmoid
        - Clasificación multiclase: tantas neuronas como clases (con softmax)
        - Regresión: 1 o más neuronas con activación lineal

Pesos y sesgos

Pesos (weights)

- Definición: Valores numéricos que determinan la fuerza de conexión entre neuronas
- Características:
    - Son los parámetros principales que la red ajusta durante el entrenamiento
    - Valores grandes indican conexiones importantes
    - Valores cercanos a cero indican conexiones débiles
    - Pueden ser positivos (excitatorios) o negativos (inhibitorios)
    - Inicialización: típicamente aleatoria pero con métodos específicos (Xavier/Glorot, He, etc.)

Sesgos (bias)

- Definición: Valores constantes añadidos a la suma ponderada en cada neurona
- Características:
    - Permiten desplazar la función de activación horizontalmente

- Añaden un grado de libertad adicional al modelo
- Funcionan como un umbral de activación ajustable
- Matemáticamente, equivalen a una entrada adicional con valor constante 1

Representación matemática

Para una neurona, la operación básica es:

$$output = f(\Sigma(w_i * x_i) + b)$$

Donde:

w_i son los pesos

x_i son las entradas

b es el sesgo (bias)

f es la función de activación

Funciones de activación

Las funciones de activación introducen no-linealidad en el modelo, permitiendo aprender relaciones complejas entre las variables.

Principales funciones de activación

1. Sigmoid (Logística)

Fórmula: $\sigma(x) = 1/(1+e^{(-x)})$

Rango de salida: (0, 1)

Características:

- Suave y diferenciable
- Útil en la capa de salida para clasificación binaria
- Problemas: desvanecimiento del gradiente en redes profundas, salidas no centradas en cero

2. Tanh (Tangente hiperbólica)

Fórmula: tanh(x) = (e^x - e^(-x))/(e^x + e^(-x))

Rango de salida: (-1, 1)

Características:

- Similar a sigmoid pero centrada en cero
- Gradientes más pronunciados
- Sufre también de desvanecimiento del gradiente

3. ReLU (Rectified Linear Unit)

Fórmula: ReLU(x) = max(0, x)

Rango de salida: [0, ∞)

Características:

- Computacionalmente eficiente
- Ayuda a mitigar el problema del desvanecimiento del gradiente
- Problema: "neuronas muertas" cuando los gradientes son cero para entradas negativas

4. Leaky ReLU

Fórmula: Leaky_ReLU(x) = max(αx, x), donde α es un valor pequeño (≈0.01)

Características:

- Soluciona el problema de las "neuronas muertas" de ReLU
- Permite gradientes pequeños para valores negativos

5. Softmax

Fórmula: softmax(x_i) = e^(x_i) / Σ_j e^(x_j)

Características:

- Usada en la capa de salida para clasificación multiclase

- Convierte valores en probabilidades (suman 1)
- Acentúa las diferencias entre los valores más altos

4.3. Proceso de entrenamiento: forward + backpropagation

Visión general del entrenamiento

El entrenamiento de una red neuronal es el proceso mediante el cual se ajustan sus pesos y sesgos para minimizar una función de error o pérdida. Este proceso se divide principalmente en dos fases que se repiten iterativamente:

1. Forward propagation (propagación hacia adelante): cálculo de la predicción
2. Backpropagation (retropropagación): cálculo de errores y actualización de parámetros

Forward Propagation (Propagación hacia adelante)

Es el proceso por el cual la información fluye desde la capa de entrada hasta la capa de salida, generando una predicción.

Pasos detallados:

1. Entrada de datos: Un lote de datos (batch) se proporciona a la capa de entrada.
2. Operaciones en cada capa:

   Para cada neurona j en la capa l:

$$z_j^{(l)} = \Sigma(w_{ji}^{(l)} * a_i^{(l-1)}) + b_j^{(l)}$$

Donde:

$z_j^{(l)}$ es la entrada neta a la neurona

$w_{ji}^{(l)}$ es el peso de la conexión

$a_i^{(l-1)}$ es la activación de la neurona en la capa anterior

$b_j^{(l)}$ es el sesgo

3. Aplicación de la función de activación:

$$a_j^{(l)} = f(z_j^{(l)})$$

- Donde f es la función de activación seleccionada

4. Propagación secuencial: La salida de cada capa se convierte en la entrada de la siguiente
5. Salida final: La capa de salida genera la predicción ŷ (y-hat)

Ejemplo simplificado:

Para una red de 3 capas con entradas X, tenemos:

$Z1 = W1 \cdot X + b1$

$A1 = f(Z1)$

$Z2 = W2 \cdot A1 + b2$

$A2 = f(Z2)$

$Z3 = W3 \cdot A2 + b3$

$A3 = f(Z3) = \hat{y}$ (predicción)

Backpropagation (Retropropagación)

Es el algoritmo que calcula el gradiente de la función de error con respecto a cada peso en la red, permitiendo actualizarlos en la dirección que minimiza el error.

Pasos detallados:

1. Cálculo del error: Se compara la predicción con el valor real.

$$E = L(\hat{y}, y)$$

   Donde L es la función de pérdida (loss function)

2. Cálculo del gradiente en la capa de salida:

$$\delta^{(L)} = \nabla_a L \odot f'(z^{(L)})$$

Donde:

- $\delta^{(L)}$ es el error en la capa de salida
- $\nabla_a L$ es el gradiente de la función de pérdida respecto a las activaciones
- f' es la derivada de la función de activación
- $\odot$ representa el producto elemento por elemento (Hadamard)

3. Propagación del error hacia atrás:

Para cada capa l desde la última capa oculta hasta la primera:

$$\delta^{(l)} = ((W^{(l+1)})^T \cdot \delta^{(l+1)}) \odot f'(z^{(l)})$$

4. Cálculo de gradientes para pesos y sesgos:

$$\nabla_{W^{(l)}} E = \delta^{(l)} \cdot (a^{(l-1)})^T$$

$$\nabla_{b^{(l)}} E = \delta^{(l)}$$

5. Actualización de parámetros mediante el algoritmo de optimización:

$$W^{(l)} = W^{(l)} - \eta \cdot \nabla_{W^{(l)}} E$$

$$b^{(l)} = b^{(l)} - \eta \cdot \nabla_{b^{(l)}} E$$

- Donde η es la tasa de aprendizaje (learning rate)

Ciclo completo de entrenamiento

1. Inicialización: Asignación aleatoria de pesos iniciales (usando métodos como inicialización de He o Xavier)
2. Bucle de épocas: Se itera sobre todo el conjunto de datos múltiples veces (épocas)
    - Para cada época:
        - Dividir datos en mini-lotes
        - Para cada mini-lote:
            - Forward propagation
            - Calcular la pérdida

- Backpropagation
- Actualizar parámetros

3. Evaluación: Después de cada época, se evalúa el rendimiento en un conjunto de validación
4. Parada temprana (Early stopping): Se detiene el entrenamiento cuando el error en validación deja de mejorar

4.4. Visualización del error durante el entrenamiento

¿Qué es el error en redes neuronales?

El error (también llamado pérdida o "loss") es una medida de qué tan inexactas son las predicciones de la red neuronal en comparación con los valores reales esperados. Es esencialmente una cuantificación de la diferencia entre lo que la red predice y la respuesta correcta.

Tipos comunes de funciones de error

1. **Error cuadrático medio (MSE):** Calcula el promedio de los cuadrados de las diferencias entre las predicciones y los valores reales. Es muy común para problemas de regresión.
2. **Entropía cruzada (Cross-Entropy)**: Mide la divergencia entre la distribución de probabilidad predicha y la distribución real. Es ampliamente utilizada en problemas de clasificación.
3. **Error absoluto medio (MAE):** Calcula el promedio de los valores absolutos de las diferencias entre predicciones y valores reales.

¿Por qué se produce el error?

El error se produce porque:

- La red neuronal está en proceso de aprendizaje y aún no ha encontrado los pesos óptimos
- Los datos son complejos y la arquitectura de la red podría no ser suficiente

- Existe ruido inherente en los datos
- La red puede estar sobreajustando (overfitting) o subajustando (underfitting) los datos

Visualización del error durante el entrenamiento

Curvas de aprendizaje

Las curvas de aprendizaje son gráficas que muestran cómo evoluciona el error a lo largo del proceso de entrenamiento. Típicamente:

- El eje X representa las épocas (iterations o epochs) - cada vez que la red procesa todo el conjunto de datos de entrenamiento
- El eje Y representa el valor del error

Interpretación de las curvas de error

1. Curva ideal: El error de entrenamiento disminuye gradualmente y se estabiliza. El error de validación sigue un patrón similar y se acerca al error de entrenamiento.
2. Sobreajuste (Overfitting): El error de entrenamiento continúa disminuyendo, pero el error de validación comienza a aumentar después de cierto punto. Esto indica que la red está "memorizando" los datos de entrenamiento en lugar de generalizar.
3. Subajuste (Underfitting): Tanto el error de entrenamiento como el de validación se mantienen altos. Esto sugiere que la red es demasiado simple para capturar los patrones en los datos.
4. Aprendizaje lento: La curva de error disminuye muy lentamente, lo que puede indicar una tasa de aprendizaje demasiado baja o una arquitectura inadecuada.
5. Inestabilidad: Si la curva de error muestra oscilaciones pronunciadas, podría indicar una tasa de aprendizaje demasiado alta o inestabilidad numérica.

Importancia de la visualización del error

Visualizar el error durante el entrenamiento es crucial porque:

1. Diagnóstico de problemas: Permite identificar problemas como sobreajuste o subajuste

2. Ajuste de hiperparámetros: Ayuda a determinar cuándo detener el entrenamiento (early stopping) y a ajustar hiperparámetros como la tasa de aprendizaje

3. Evaluación del rendimiento: Proporciona una manera de evaluar qué tan bien está aprendiendo la red

4. Comparación de modelos: Facilita la comparación entre diferentes arquitecturas o configuraciones

Técnicas de visualización comunes

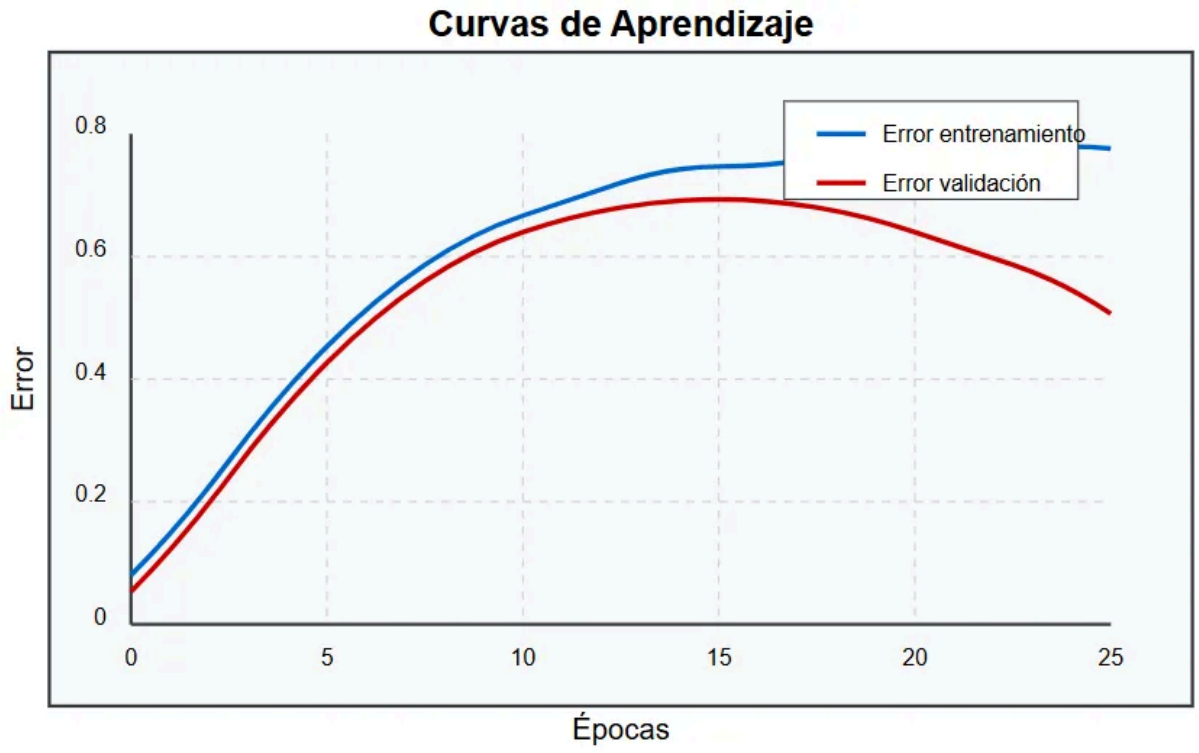
1. Gráficas de error por época: La visualización más básica y común

2. Comparación de error de entrenamiento vs validación: Fundamental para detectar sobreajuste

3. Gráficas de distribución del error: Muestra cómo se distribuye el error entre diferentes muestras

4. Mapas de calor del error: Visualización más avanzada que muestra qué áreas o características contribuyen más al error

Gráficas de visualización del error mediante métodos específicos



El gráfico representa la curva de aprendizaje típicas durante el entrenamiento de una red neuronal.

Elementos básicos de la gráfica

- Eje X (Épocas): Representa el número de iteraciones completas sobre todo el conjunto de datos de entrenamiento.
- Eje Y (Error): Muestra la magnitud del error o pérdida.
- Línea azul: Representa el error en los datos de entrenamiento.
- Línea roja: Representa el error en los datos de validación (datos que la red no ha visto durante el entrenamiento).

Interpretación de estas curvas específicas

Esta gráfica muestra un caso interesante porque ambas curvas inicialmente aumentan (de la época 0 a aproximadamente la época 15), lo cual no es lo típico. Esto puede representar un escenario particular:

1. Fase inicial (épocas 0-15):

- Tanto el error de entrenamiento como el de validación aumentan
- Esto podría indicar que la red está "desaprendiendo" patrones iniciales o que hay una configuración de entrenamiento particular

2. Fase posterior (épocas 15-25):

- El error de entrenamiento (azul) se estabiliza
- El error de validación (rojo) comienza a disminuir

¿De dónde sale esta curva?

Esta curva se obtiene durante el proceso de entrenamiento de la red neuronal:

1. Cálculo del error: En cada época, la red:

   Procesa todos los datos de entrenamiento (forward pass)

   Calcula la diferencia entre sus predicciones y los valores reales (error)

Hace lo mismo con un conjunto separado de datos de validación

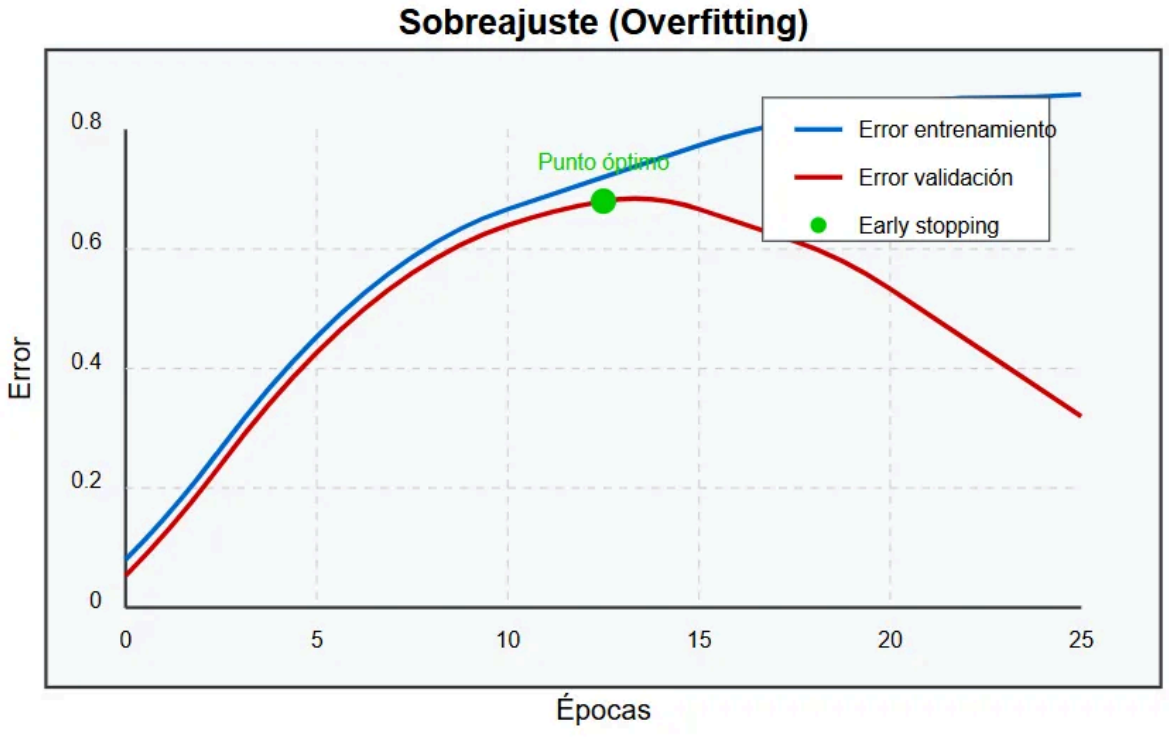
2. Almacenamiento: Estos valores de error se guardan para cada época
3. Visualización: Se grafican estos valores almacenados, con las épocas en el eje X y los valores de error en el eje Y

¿Qué significa esto para el modelo?

Esta gráfica particular muestra un comportamiento atípico que podría indicar:

1. Posible subajuste inicial: La red podría estar luchando por encontrar patrones útiles al principio.
2. Mejora gradual en generalización: La disminución del error de validación al final sugiere que el modelo está comenzando a generalizar mejor.
3. Comportamiento interesante: El hecho de que el error aumente antes de disminuir podría indicar que el modelo está pasando por una fase de "reorganización" de sus pesos antes de encontrar una configuración más óptima.

Gráfica de sobreajuste



Esta curva muestra el patrón clásico de overfitting:

1. Fase inicial (épocas 0-12):

- Ambas curvas (entrenamiento y validación) disminuyen juntas
- La red está aprendiendo patrones útiles que generalizan bien

2. Punto crítico (época ~12-13):

- Ambas curvas alcanzan su punto mínimo
- Este es el punto óptimo donde el modelo generaliza mejor

3. Fase de sobreajuste (épocas 13-25):

- Error de entrenamiento (azul): Continúa disminuyendo (incluso llega casi a 0.85)
- Error de validación (rojo): Comienza a aumentar significativamente
- Esta divergencia es la señal clara de overfitting

¿Qué está pasando?

Después del punto óptimo, la red neuronal:

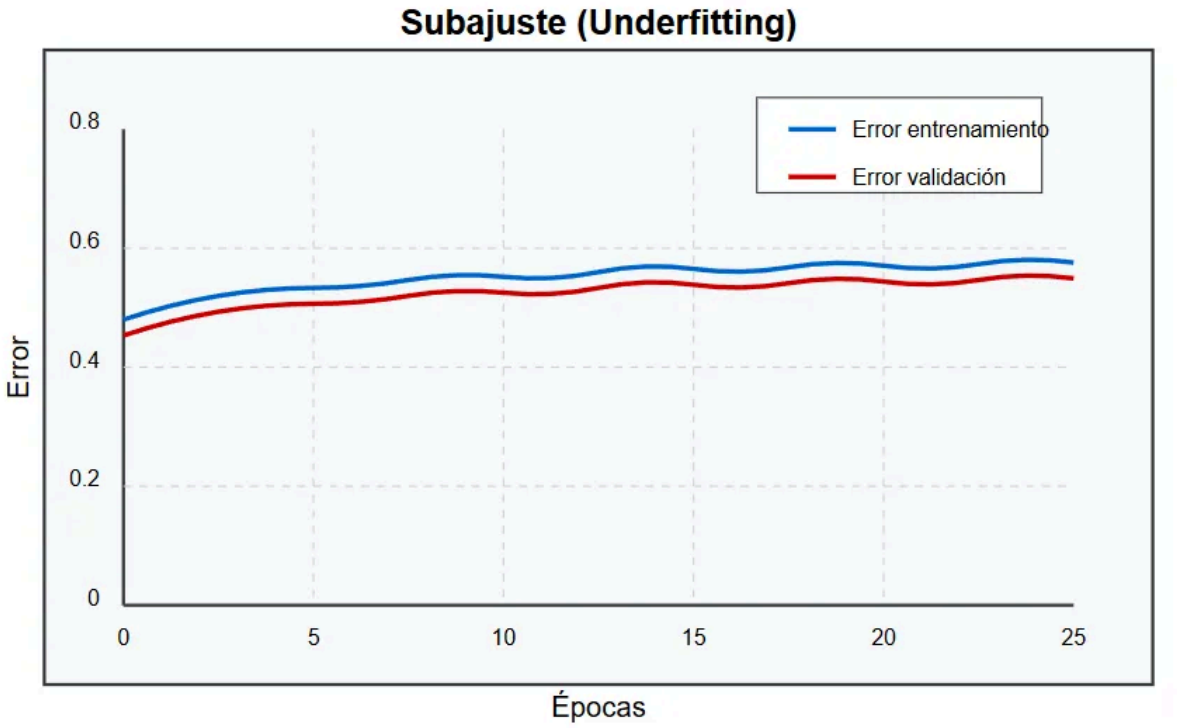
- Sigue "memorizando" los datos de entrenamiento (por eso el error de entrenamiento mejora)
- Pero pierde capacidad de generalizar a datos nuevos (por eso el error de validación empeora)
- La red se está especializando demasiado en los datos que ha visto

Early Stopping (Punto verde)

El punto verde marca la estrategia de early stopping:

- Es donde deberíamos haber detenido el entrenamiento
- Representa el mejor balance entre aprender patrones útiles y mantener la capacidad de generalización
- Entrenar más allá de este punto solo empeora el rendimiento del modelo

Gráfica de subajuste



Características del Subajuste en esta gráfica

1. Errores altos y estables

- Tanto el error de entrenamiento (azul) como el de validación (rojo) se mantienen altos (alrededor de 0.5-0.55)
- No hay una mejora significativa a lo largo de las 25 épocas

2. Curvas paralelas

- Ambas curvas están muy próximas entre sí
- No hay divergencia como en el overfitting

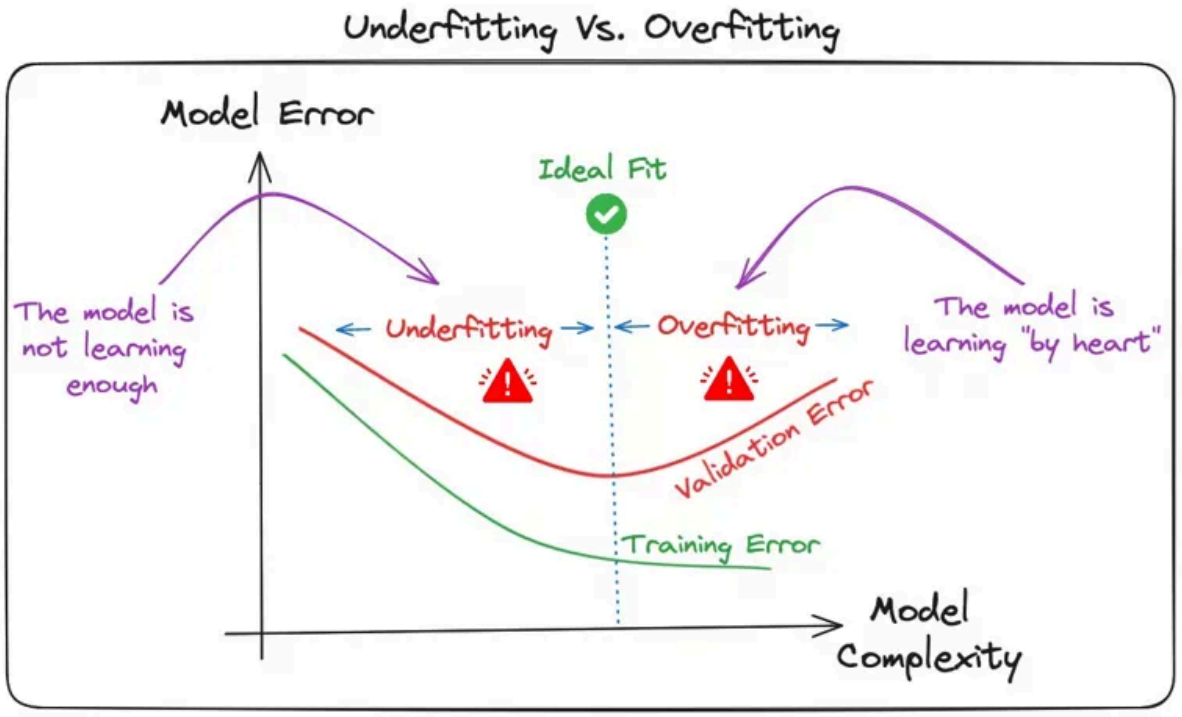
Esto indica que el modelo tiene un rendimiento similar en datos de entrenamiento y validación, pero ambos son pobres

3. Falta de aprendizaje

- Las curvas son prácticamente planas

- La red no está captando los patrones en los datos
- El modelo es demasiado simple para la complejidad del problema

Balance entre underfitting y overfitting



Estructura de la gráfica

Ejes:

- Eje X (Model Complexity): Representa qué tan complejo es el modelo (pocas neuronas/capas → muchas neuronas/capas)
- Eje Y (Model Error): Representa la magnitud del error del modelo

Curvas:

- Curva verde (Training Error): Error en datos de entrenamiento
- Curva roja (Validation Error): Error en datos de validación

Explicación dell gráfico

1. Zona de Underfitting (Izquierda)

- El training error es alto (curva verde alta)
- El validation error también es alto (curva roja alta)
- Ambas curvas están cerca → el modelo "no está aprendiendo lo suficiente"
- Es como darle un problema de cálculo a un niño de primaria

2. Punto Ideal (Centro)

- Marcado con la ✓ verde y "Ideal Fit"
- Ambos errores están en su punto más bajo
- Las curvas están próximas pero el modelo sí está aprendiendo
- Este es nuestro objetivo: generaliza bien sin memorizar

3. Zona de Overfitting (Derecha)

- El training error continúa bajando (curva verde baja)
- Pero el validation error comienza a subir (curva roja sube)
- El modelo está "aprendiendo de memoria"
- Es como un estudiante que memoriza las respuestas pero no entiende el concepto

La gráfica nos muestra que existe un punto óptimo de complejidad del modelo. Ni muy simple, ni muy complejo.

Conclusiones

Síntesis General del Deep Learning como Paradigma Revolucionario

El análisis exhaustivo de los fundamentos del Deep Learning revela una disciplina que ha transformado radicalmente el panorama de la inteligencia artificial contemporánea. Esta tecnología no representa simplemente una evolución incremental de técnicas preexistentes, sino que constituye un cambio paradigmático fundamental en la forma en que concebimos el aprendizaje automático, el procesamiento de información compleja y la resolución de problemas que anteriormente se consideraban intratables computacionalmente.

Evolución Histórica y Consolidación Tecnológica

La perspectiva histórica del Deep Learning demuestra un desarrollo que, aunque tiene raíces conceptuales en décadas anteriores, ha experimentado su verdadero florecimiento en la era contemporánea gracias a la convergencia de tres factores críticos: el incremento exponencial en la capacidad de procesamiento computacional, la disponibilidad masiva de datos digitales, y los avances algorítmicos en técnicas de optimización. Esta confluencia ha permitido que conceptos teóricos desarrollados en los años 1940-1980 finalmente alcancen su potencial práctico.

Las características principales del Deep Learning - su capacidad de aprendizaje jerárquico, la extracción automática de características, y la aproximación universal de funciones - representan logros que han requerido décadas de investigación y desarrollo. El proceso de aprendizaje de una red neuronal, basado en la propagación hacia adelante y hacia atrás del error, constituye un mecanismo elegante que permite la optimización automática de millones o incluso billones de parámetros de manera eficiente y efectiva.

Diferenciación Fundamental con el Machine Learning Clásico

La comparación sistemática entre Deep Learning y Machine Learning clásico revela diferencias que van mucho más allá de lo meramente técnico, abarcando aspectos conceptuales y metodológicos fundamentales. Mientras que el Machine Learning tradicional requiere una definición explícita y manual de características relevantes por parte de expertos del dominio, el Deep Learning introduce la capacidad revolucionaria de descubrimiento automático de representaciones óptimas.

Esta diferencia en la extracción de características representa un salto cualitativo extraordinario. Los métodos clásicos, ejemplificados por algoritmos como Support Vector Machines, dependen críticamente de la ingeniería manual de funciones, un proceso que no solo consume recursos significativos sino que también introduce limitaciones inherentes basadas en el conocimiento y la intuición humana. En contraste, las redes neuronales profundas, como las Convolutional Neural Networks, desarrollan automáticamente filtros y representaciones que frecuentemente superan las características diseñadas manualmente, descubriendo patrones y relaciones que podrían pasar desapercibidos en el análisis humano tradicional.

Los requisitos de datos y procesamiento también marcan una distinción crucial. Mientras que los métodos clásicos pueden funcionar efectivamente con conjuntos de datos relativamente pequeños, el Deep Learning requiere volúmenes masivos de información para alcanzar su potencial óptimo, compensando esta exigencia con capacidades de generalización y rendimiento que justifican ampliamente la inversión computacional adicional.

Aplicaciones Transformadoras y Limitaciones Inherentes

El espectro de aplicaciones del Deep Learning abarca prácticamente todos los dominios de la actividad humana donde existe información procesable. En visión por computadora, ha alcanzado y frecuentemente superado el rendimiento humano en tareas como clasificación de imágenes, detección de objetos y análisis de escenas complejas. En procesamiento de lenguaje natural, ha democratizado capacidades antes reservadas a sistemas altamente especializados, desde traducción automática hasta generación de texto coherente y análisis de sentimientos. En el reconocimiento de voz, ha eliminado virtualmente las barreras entre la comunicación humana y la interacción máquina.

Los ejemplos reales en industrias como la medicina, donde facilita diagnósticos más precisos y tempranos, el transporte, donde impulsa el desarrollo de vehículos autónomos, y las finanzas, donde optimiza la detección de fraudes y la gestión de riesgos, demuestran que no estamos ante una tecnología de nicho sino ante una herramienta fundamental que está redefiniendo los estándares de eficiencia y precisión en múltiples sectores económicos.

Sin embargo, es crucial reconocer las limitaciones inherentes del Deep Learning. Su naturaleza de "caja negra" presenta desafíos significativos en términos de interpretabilidad y explicabilidad, aspectos críticos en aplicaciones donde la comprensión del proceso de toma

de decisiones es tan importante como la precisión del resultado. Los requisitos computacionales masivos pueden representar barreras de entrada significativas, y la dependencia de grandes volúmenes de datos plantea desafíos en dominios donde la información es escasa o costosa de obtener.

Las consideraciones éticas emergen como un aspecto fundamental que no puede ser ignorado. Los riesgos potenciales incluyen sesgos algorítmicos que pueden perpetuar o amplificar desigualdades existentes, preocupaciones sobre privacidad y uso indebido de datos personales, y el impacto socioeconómico de la automatización masiva en el mercado laboral.

Arquitectura Neural como Fundamento Tecnológico

Las redes neuronales artificiales constituyen el sustrato tecnológico que hace posible las capacidades excepcionales del Deep Learning. La estructura fundamental de estos sistemas - compuesta por capas de neuronas artificiales con pesos sinápticos, funciones de activación no lineales, y mecanismos de propagación de información - representa una abstracción computacional elegante inspirada en el funcionamiento del cerebro biológico pero optimizada para el procesamiento digital.

El proceso de entrenamiento, articulado a través de los algoritmos de forward propagation y backpropagation, constituye un mecanismo de optimización que permite el ajuste automático de millones o billones de parámetros de manera eficiente. Este proceso iterativo de refinamiento, guiado por funciones de pérdida específicamente diseñadas, permite que la red neuronal evolucione gradualmente desde un estado inicial aleatorio hasta una configuración que puede realizar tareas complejas con precisión excepcional.

La visualización del error durante el entrenamiento proporciona insights valiosos sobre el comportamiento del modelo, permitiendo la identificación de problemas como sobreajuste, subajuste, o convergencia prematura. Esta capacidad de monitoreo y diagnóstico es esencial para el desarrollo efectivo de sistemas de Deep Learning robustos y confiables.

Perspectivas Futuras y Consideraciones Estratégicas

Los fundamentos estudiados establecen claramente que el Deep Learning no es simplemente una técnica más en el arsenal del machine learning, sino que representa una nueva era en la inteligencia artificial. Su capacidad para aprender representaciones complejas de manera autónoma, combinada con su versatilidad arquitectónica y su aplicabilidad transversal,

sugiere que continuará siendo el motor principal de innovación en inteligencia artificial en las próximas décadas.

La integración creciente del Deep Learning en infraestructuras críticas y procesos de toma de decisiones importantes subraya la necesidad de desarrollar marcos robustos para la gobernanza, la ética y la responsabilidad en el despliegue de estos sistemas. La comprensión profunda de estos fundamentos es esencial no solo para desarrolladores e investigadores, sino para cualquier profesional que busque navegar efectivamente en un mundo cada vez más influenciado por la inteligencia artificial.

Reflexión Final

En última instancia, el estudio comprehensivo de los fundamentos del Deep Learning revela que estamos presenciando no solo una evolución tecnológica, sino una revolución en nuestra comprensión de cómo las máquinas pueden aprender, adaptarse y resolver problemas complejos de manera cada vez más autónoma y eficiente. Esta transformación plantea tanto oportunidades extraordinarias como responsabilidades significativas, requiriendo un enfoque equilibrado que maximice los beneficios mientras mitiga los riesgos potenciales.

El Deep Learning ha demostrado ser una tecnología transformadora que continuará moldeando el futuro de la inteligencia artificial y, por extensión, el futuro de la sociedad humana. La comprensión sólida de sus fundamentos constituye una base esencial para participar constructivamente en esta transformación y para contribuir al desarrollo de sistemas inteligentes que sean no solo poderosos y eficientes, sino también éticos, responsables y beneficiosos para la humanidad en su conjunto.

Bibliografía

Rajesh, K. (2025, February 11). Extracción de características en el aprendizaje automático: Guía completa. *DataCamp*. https://www.datacamp.com/es/tutorial/feature-extraction-machine-learning

Badillo, F. L., Hernández, C. A. R., Narváez, B. M., & Trillos, Y. E. A. (2021). Redes neuronales convolucionales: un modelo de Deep Learning en imágenes diagnósticas. Revisión de tema. *Revista colombiana de radiología*, *32*(3), 5591-5599.

Tablada, C. J., & Torres, G. A. (2009). Redes neuronales artificiales. *Revista de Educación Matemática (RevEM)*, *24*(3), 2.

Izaurieta, F., & Saavedra, C. (2000). Redes neuronales artificiales. *Departamento de Física, Universidad de Concepción Chile*.

Rojas, R., & Rojas, R. (1996). The backpropagation algorithm. *Neural networks: a systematic introduction*, 149-182.

Ferrán, E., & Perazzo, R. (1990). COMPORTAMIENTO ASINTÓTICO DEL APRENDIZAJE EN REDES NEURONALES. In *ANALES AFA* (Vol. 1, No. 1).