UNIVERSIDAD DE GRANADA

Aprendizaje Automático

Apuntes de la asignatura

Ismael Sallami Moreno

Recursos Ingeniería Informática y Ade

Licencia

Aprendizaje Automático

Ismael Sallami Moreno

https://elblogdeismael.github.io

Índice general

1 Teoría **9**

1.1 Tema 1. Conceptos básicos **9**

1.2 Tema 2. Aprendizaje Supervisado **11**

1.3 Tema 3. Calidad de Datos **13**

1.4 Tema 4. Aprendizaje No Supervisado **15**

1.5 Tema 5. Aprendizaje Profundo. Fundamentos **17**

1.6 Tema 6. Aprendizaje Profundo en Diferentes Tipos de Datos **19**

1.7 Tema 7: Aprendizaje por Refuerzo (Reinforcement Learning) **21**

1.8 Tema 8: Optimización en Aprendizaje Automático, Hiperparámetros, Selección de Modelos y Regularización **22**

1.9 Tema 9. Aspectos Avanzados, Retos y Desafíos **23**

2 Glosario **27**

Capítulo 1

Teoría

Tema 8 y 9 no entran en el examen

1.1 Tema 1. Conceptos básicos

1.1.1 ¿Qué es el Aprendizaje Automático (AA)?

- **Definición clásica (Arthur Samuel, 1959):** ciencia y arte de programar ordenadores para que aprendan de los datos sin ser programados explícitamente.
- **Definición de ingeniería (Tom Mitchell, 1997):** un programa aprende de la **experiencia E**, respecto a una **tarea T** y una **medida de rendimiento P**, si su rendimiento en T (medido por P) mejora con la experiencia E.
- **Formalización:** consiste en estimar una función $f(x, w)$ que minimice el riesgo o pérdida promedio sobre un conjunto de datos de entrenamiento (x_i, y_i).
- **¿Cuándo conviene usarlo?** En problemas complejos donde las reglas explícitas son ineficientes, en entornos fluctuantes (los datos cambian) y para descubrir patrones ocultos en grandes volúmenes de información (minería de datos).

1.1.2 Tipos de Aprendizaje Automático

- **Aprendizaje Supervisado:** se entrena con datos que incluyen las soluciones deseadas, llamadas **etiquetas**. Dos tareas principales:
  - **Clasificación:** predecir una clase o categoría (p. ej. filtro antispam).
  - **Regresión:** predecir un valor numérico objetivo a partir de unas características o predictores (p. ej. el precio de un coche).
- **Aprendizaje No Supervisado:** los datos no están etiquetados; el sistema aprende sin "profesor". Tareas:
  - **Agrupamiento (Clustering):** detectar grupos similares (K-Means, DBSCAN; segmentación de clientes).
  - **Detección de anomalías:** identificar casos inusuales (fraude, defectos; Isolation Forest).
  - **Reducción de dimensionalidad y visualización:** simplificar datos complejos (PCA, t-SNE).
  - **Reglas de asociación:** descubrir relaciones entre atributos (algoritmo **Apriori**).
- **Aprendizaje Semi-supervisado:** combina pocos datos etiquetados con muchos sin etiquetar; útil cuando etiquetar es caro.
- **Aprendizaje por Refuerzo:** un **agente** observa un **entorno**, ejecuta **acciones** y recibe **recompensas** o penalizaciones, aprendiendo una **política** que maximiza la recompensa acumulada (AlphaGo, bots de videojuegos).

1.1.3 Enfoques: Instancias vs. Modelos

- **Basado en instancias:** memoriza los ejemplos y clasifica los casos nuevos por una medida de **similitud** (distancia) frente a los datos almacenados. No requiere entrenamiento previo (p. ej. k-NN).
- **Basado en modelos:** construye un modelo que generaliza, usando una **función de pérdida** (error) y un método de **optimización**.

1.1.4 Evaluación y Generalización

- Los datos se dividen en **conjunto de entrenamiento** y **conjunto de test**.
- **Generalización:** mide el comportamiento ante datos nuevos (no vistos en entrenamiento). Para estimarla se usa la **validación cruzada (k-fold)**, que particiona los datos en k bloques y rota cuál actúa como test.
- **Estimación de densidad:** estimar la función de densidad de probabilidad (PDF) que generó los datos; útil para detectar anomalías (zonas de baja densidad).
- **Criterios de evaluación:** además de la precisión, se valoran velocidad, robustez (tratar valores desconocidos), escalabilidad, interpretabilidad y complejidad del modelo.

1.1.5 Retos en el Entorno Real

- **Teorema "No Free Lunch" (NFL):** sin suposiciones sobre los datos, no hay razón para preferir un modelo sobre otro. No existe un algoritmo óptimo para todos los problemas.
- **Sobreajuste (Overfitting):** el modelo va muy bien en entrenamiento pero generaliza mal. Se combate con **regularización** y ajuste de hiperparámetros.
- **Subajuste (Underfitting):** el modelo es demasiado simple para capturar la estructura de los datos.
- **Datos no representativos:** si validación y test no representan la realidad, el modelo falla en producción.

1.1.6 Calidad de Datos y Enfoques de Desarrollo

- Era del **Big Data**: el dato es el recurso clave y abarca múltiples tipos (texto, imagen, audio, series temporales).
- **Data-Centric vs Model-Centric:** transición desde mejorar el modelo (Model-Centric) hacia el enfoque **Data-Centric**, donde la arquitectura se estandariza y el esfuerzo se concentra en iterar, adquirir, limpiar y auditar los datos.

1.1.7 Regulación y Ética

- **AI Act:** marco regulatorio europeo que clasifica la IA por nivel de riesgo (inaceptable, alto, limitado, mínimo).
- **IA Fiable (Trustworthy AI):** exige sistemas robustos, lícitos y éticos: transparencia, privacidad y supervisión humana durante todo el ciclo de vida.
- **AI Safety:** estudio de la seguridad y vulnerabilidades frente a adversarios (p. ej. patrones en ropa que actúan como "capa de invisibilidad" ante detectores de objetos).

1.2 Tema 2. Aprendizaje Supervisado

1.2.1 Evaluación de Modelos (Métricas)

Antes de construir modelos, es necesario saber cómo evaluarlos.

- **Para Regresión:**
  - **RMSE (Raíz del Error Cuadrático Medio):** medida preferida; penaliza más los errores grandes y se basa en la norma Euclídea (L_2).
  - **MAE (Error Medio Absoluto):** con muchos valores atípicos (*outliers*), el MAE (norma Manhattan, L_1) es preferible por ser más robusto.
- **Para Clasificación:**
  - **Matriz de confusión:** recoge verdaderos positivos (TP), falsos positivos (FP), verdaderos negativos (TN) y falsos negativos (FN).
  - **Precisión (Precision):** exactitud de las predicciones positivas, $TP/(TP + FP)$.
  - **Recuperación / Sensibilidad (Recall):** capacidad de encontrar todos los positivos, $TP/(TP + FN)$. Existe un **compromiso (trade-off) precision/recall**: subir el umbral de decisión sube la precisión y baja el recall.
  - **F1:** media armónica de precisión y recall.
  - **Curva ROC y AUC:** la ROC traza sensibilidad frente a 1-especificidad; el **AUC** es el área bajo la curva (1 = perfecto, 0,5 = azar).

1.2.2 Aprendizaje Basado en Instancias vs. Modelos

- **Basado en instancias (k-NN):** memoriza el entrenamiento y clasifica midiendo similitud (distancia) con los vecinos más cercanos. Sin fase de entrenamiento real. Clasificación: "regla de la mayoría" de los vecinos; regresión: media del vecindario.
- **Basado en modelos:** construye una función matemática que minimiza un error.

1.2.3 Modelos Lineales y Regularización (Regresión)

- **Regresión Lineal:** predice una suma ponderada de las características más un término de sesgo (bias). Se entrena minimizando el MSE.
  - Resolución directa con la **Ecuación Normal** (pseudoinversa o SVD) o iterativa con **Descenso de Gradiente**. Como el coste es convexo, el descenso de gradiente garantiza el mínimo global.
- **Regresión Polinómica:** ajusta datos no lineales añadiendo potencias de las características. Un grado alto provoca **sobreajuste**.
- **Regularización** (restringir pesos para evitar el sobreajuste):
  - **Ridge:** penalización L_2 sobre el MSE. Mantiene los pesos pequeños. Buena opción por defecto.
  - **Lasso:** penalización L_1. Elimina características inútiles forzando pesos exactamente a cero (modelo disperso).
  - **Elastic Net:** término medio entre Ridge y Lasso; preferible a Lasso cuando hay características muy correlacionadas.
  - **Parada Anticipada (Early Stopping):** regulariza métodos iterativos deteniendo el entrenamiento cuando el error de validación alcanza un mínimo.

1.2.4 Clasificación Probabilística: Regresión Logística y Naïve Bayes

- **Regresión Logística:** estima la probabilidad de pertenencia a una clase (si $> 50\,\%$, predice positivo). Coste convexo: **Pérdida Logarítmica (Log Loss)**.
- **Regresión Softmax:** generaliza la logística a múltiples clases sin clasificadores binarios separados. Coste: **Entropía Cruzada**.
- **Naïve Bayes:** se basa en el Teorema de Bayes y la hipótesis (ingenua) de independencia entre características; aplica la regla **MAP (Maximum A Posteriori)** para devolver la clase más probable.

1.2.5 Máquinas de Vectores de Soporte (SVM)

Basadas en la Teoría del Aprendizaje Estadístico.

- **SVM Lineal (Clasificación):** busca la "calle" (margen) más ancha posible entre clases.
  - **Vectores de soporte:** instancias situadas en el borde de la calle que determinan el margen. Muy sensibles a la **escala** de las características.
  - **Margen Duro vs Blando:** el duro no permite violaciones (sensible a outliers, exige separabilidad lineal). El **margen blando** usa el hiperparámetro C para equilibrar: permite algunas violaciones para generalizar mejor. C grande $\rightarrow$ menos regularización.
- **Regresión SVM:** invierte el objetivo: mete el mayor número de instancias *dentro* de la calle, cuyo ancho controla el hiperparámetro ϵ (modelo ϵ-insensible).
- **SVM No Lineal (truco del Kernel):** para datos no separables linealmente. En vez de añadir características polinómicas (costoso), usa funciones **Kernel** (Polinómico, **Gaussiano RBF**, Sigmoide) apoyándose en el **Teorema de Mercer** para operar en dimensiones superiores sin calcularlas explícitamente.

1.2.6 Árboles de Decisión y Ensembles

- **Árboles de Decisión (CART):** árbol de preguntas (¿pétalo $< 2{,}45$ cm?). Clasificación: minimizan impureza **Gini** o **Entropía**; regresión: minimizan el MSE. Son **no paramétricos**, por lo que sin restricciones (`max_depth`, `min_samples_leaf`) tienden fuertemente al sobreajuste. Tienen **alta varianza** y son sensibles a la orientación de los ejes.
- **Ensembles (sabiduría de la multitud):** combinar varios modelos.
  - **Voto duro vs blando:** el duro agrega por mayoría de clases predichas; el blando promedia probabilidades (suele rendir mejor si los modelos las estiman bien).
  - **Bagging y Pasting:** entrena un mismo modelo en subconjuntos aleatorios. *Bagging* con reemplazo; *Pasting* sin él. Agregación final por moda (clasificación) o media (regresión).
  - **Evaluación Out-of-Bag (OOB):** en bagging, cada predictor se valida con las instancias que no le tocaron en su muestreo, sin necesitar conjunto de validación aparte.
  - **Random Patches / Random Subspaces:** muestrean también las características (no solo las instancias).
  - **Random Forests:** ensemble (bagging) de árboles que además aleatoriza el subconjunto de características en cada división. Promediar muchos árboles reduce drásticamente la varianza. Permite medir la **importancia de las características**.
  - **Extra-Trees (Extremely Randomized Trees):** como Random Forest pero con umbrales de división aleatorios; más rápido y con más sesgo y menos varianza.
  - **Boosting:** entrena modelos en secuencia, cada uno corrigiendo los fallos del anterior. **AdaBoost** da más peso a las instancias mal clasificadas; **Gradient Boosting** ajusta

cada nuevo predictor a los errores residuales del conjunto.
- **Stacking:** entrena un meta-modelo (*blender*) que aprende a combinar las predicciones de varios modelos base.

1.2.7 Estrategias Multiclase

Para que un clasificador binario puro (como SVM) prediga sobre 3+ clases:

- **OvO (One-vs-One):** un modelo binario por cada par de clases.
- **OvA / OvR (One-vs-All / One-vs-Rest):** un clasificador por clase frente al resto; gana la clase con mayor confianza. Suele preferirse por entrenar menos modelos.

Claves de examen (Tema 2). Frente al sobreajuste, los recursos principales son la **regularización** (hiperparámetros como C, α, profundidades de árbol; Ridge/Lasso/Elastic Net) y el **ensamblado** que reduce varianza (Random Forests). RMSE penaliza errores grandes; MAE es más robusto a outliers. Precision mide acierto de los positivos; recall, cobertura de los positivos.

1.3 Tema 3. Calidad de Datos

Principio rector: **"Garbage In, Garbage Out"** (datos basura → resultados basura). El preprocesamiento convierte datos brutos en *Smart Data* y llega a ocupar el **80 % del tiempo** de un proyecto de minería de datos.

1.3.1 Integración, Limpieza y Transformación

- **Integración:** combinar datos de múltiples fuentes, detectar duplicados y eliminar atributos redundantes (p. ej. por correlación).
- **Limpieza de datos imperfectos:**
  - **Valores perdidos (Missing values):** ignorarlos (poco recomendable), rellenar con una constante, usar media/desviación, o imputarlos con algoritmos de AA (KNNI, SVMI, etc.).
  - **Ruido:** ejemplos mal etiquetados o erróneos. Se filtran con *ensembles* de clasificadores por votación: **Ensemble Filter (EF)**, **Cross-Validated Committees Filter (CVCF)**, **Iterative-Partitioning Filter (IPF)**.
  - **Outliers:** datos muy diferentes del resto; se detectan por distancias o por *clustering* parcial.
- **Transformación:**
  - **Agregación y generalización:** resumir (ventas diarias → mensuales) o subir en la jerarquía (edad numérica → "joven/adulto").
  - **Normalización:** escalar valores numéricos para métodos sensibles a la distancia (k-NN, redes neuronales), llevando los rangos a $[0, 1]$ o $[-1, 1]$ (escala min-max, escala decimal, estandarización z-score).

1.3.2 Reducción de Datos

Objetivo: un conjunto más pequeño y manejable que conserve la relevancia, para aprender más rápido y generalizar mejor.

A. Discretización

Convierte valores numéricos continuos en intervalos nominales; facilita la comprensión y es obligatoria para algunos algoritmos.

- **No supervisada:** sin usar la clase. Igual amplitud (intervalos del mismo tamaño) o igual frecuencia (mismo número de datos por intervalo).
- **Supervisada:** usa la clase para cortes más informados. Destacan el algoritmo de **Fayyad e Irani** basado en **Entropía (criterio MDLP)** y los métodos basados en **Chi-cuadrado**.

B. Selección de Características (Feature Selection)

Busca el subconjunto óptimo de variables; demasiadas aumentan la complejidad y empeoran la generalización. Combina estrategias de búsqueda (añadir o quitar variables) con funciones de evaluación:

- **Filtro (Filter):** métricas matemáticas (correlación, entropía, distancias). Rápido, pero tiende a seleccionar muchas variables.
- **Envolvente (Wrapper):** usa el propio algoritmo de aprendizaje para evaluar el subconjunto. Más exacto, pero **muy costoso** y sesgado hacia ese clasificador concreto.
- **Embebido (Embedded):** la selección ocurre dentro del entrenamiento (p. ej. Lasso, importancia en árboles).

C. Selección de Instancias

Elige las filas (ejemplos) más relevantes, eliminando ruido y redundancia; útil para métodos perezosos (k-NN) y para reducir tamaño en árboles.

- **Condensación (CNN):** retiene instancias de las fronteras de decisión y descarta las interiores. Rápido (incremental) pero dependiente del orden.
- **Edición (ENN):** elimina instancias mal clasificadas por sus k vecinos. Suaviza fronteras (quita ruido) pero no elimina redundancia.
- **Híbridos y evolutivos:** combinan ambos. Destacan **DROP3**, algoritmos genéticos (**CHC**) y **SSMA**.
- **Muestreo (Sampling):** seleccionar un subconjunto aleatorio representativo.

1.3.3 Reducción de Dimensionalidad

- **Maldición de la dimensionalidad:** a más dimensiones, los datos se vuelven dispersos, las distancias pierden significado y se necesita exponencialmente más datos. Proyectar a menos dimensiones ayuda.
- **Proyección:** los datos reales suelen vivir cerca de un subespacio de menor dimensión; proyectar sobre él reduce dimensiones (falla si el subespacio se "retuerce").
- **Aprendizaje de variedades (Manifold Learning):** asume que los datos yacen sobre una variedad de baja dimensión "enrollada" en el espacio (p. ej. desenrollar el *Swiss roll* de 3D a 2D).
- **PCA (Análisis de Componentes Principales):** método estrella. Halla el hiperplano que preserva la **máxima varianza** y proyecta sobre él, apoyándose en la **Descomposición en Valores Singulares (SVD)**.
  - **PCA incremental:** procesa los datos por lotes (mini-batches), apto para conjuntos que no caben en RAM o flujos.

  - **PCA aleatorio / Proyección aleatoria:** proyecta sobre un subespacio aleatorio; muy rápido y con garantías teóricas (lema de Johnson-Lindenstrauss).
- **LLE (Locally Linear Embedding):** técnica no lineal de *manifold learning*; conserva las relaciones lineales locales entre vecinos al reducir dimensiones. Bueno con datos no lineales sin ruido.

Claves de examen (Tema 3). Conviene distinguir **Selección de Características** (eliminar *columnas/variables*) de **Selección de Instancias** (eliminar *filas/ejemplos*). Los enfoques *Wrapper* son más exactos pero mucho más lentos que los *Filter*. PCA preserva varianza vía SVD; reduce dimensiones, no instancias.

1.4 Tema 4. Aprendizaje No Supervisado

1.4.1 Tareas Principales

Se buscan patrones naturales sin etiquetas. Tres áreas:

- **Clustering (Agrupamiento):** asignar instancias similares a grupos (segmentación de clientes, motores de búsqueda, segmentación de imágenes).
- **Detección de anomalías:** aprender el aspecto de los datos "normales" (inliers) para identificar instancias raras (fraude, defectos).
- **Estimación de densidad:** estimar la PDF del proceso generador; útil para análisis y para detectar anomalías (baja densidad).

1.4.2 Algoritmos de Clustering

A. Familia K-Means

Rápido y sencillo: inicializa k centroides, asigna cada punto al más cercano y recalcula el centroide (media) hasta converger. Minimiza la **inercia** (suma de distancias al cuadrado a su centroide).

- **Límites:** exige conocer k de antemano; debe ejecutarse varias veces para no caer en óptimos locales; falla con clusters de distinta densidad, tamaño o forma no esférica.
- **Variantes:**
  - **K-Means++:** inicialización inteligente: separa al máximo los centroides iniciales, reduciendo soluciones subóptimas.
  - **K-Means acelerado (Elkan):** usa la desigualdad triangular para evitar cálculos de distancia innecesarios.
  - **Mini-Batch K-Means:** actualiza con pequeños lotes; mucho más rápido y apto para datos que no caben en RAM, con inercia algo peor.

B. Número óptimo de clusters (k)

- **Método del Codo (Elbow):** la inercia siempre baja al subir k; se busca el "codo" (punto de inflexión) en la curva inercia vs k.
- **Puntuación de Silueta (Silhouette):** más precisa. Para cada punto, $(b-a)/\max(a,b)$, con a = distancia media intra-cluster y b = distancia media al cluster más cercano. Va de -1 a $+1$ (cerca de $+1$ = bien agrupado).

C. DBSCAN (Basado en Densidad)

Define clusters como regiones continuas de alta densidad. Hiperparámetros: radio de vecindad ϵ y mínimo de muestras (`min_samples`).

- Clasifica los puntos como **centrales** (con el mínimo de vecinos), **de borde** o **anomalías**.
- **Ventajas:** no exige k, detecta formas arbitrarias y es robusto a outliers.
- **Desventajas:** falla con densidades muy distintas y escala mal, $O(m^2 n)$.

D. Otros algoritmos

- **Clustering Aglomerativo:** árbol jerárquico de abajo arriba, emparejando los clusters más cercanos.
- **BIRCH:** para datasets enormes; construye un árbol sobre la marcha limitando memoria.
- **Mean-Shift:** desplaza círculos hacia la zona de mayor densidad (máximo local). Cualquier forma, pero lento.
- **Affinity Propagation:** los puntos intercambian mensajes "votando" por un ejemplar representativo; calcula k automáticamente.
- **Clustering Espectral:** reduce la dimensionalidad de una matriz de similitud y aplica k-means. Excelente para grafos (redes sociales).

1.4.3 Medidas de Calidad del Clustering

- **Índice de Davies-Bouldin:** relación dispersión intra-cluster / separación entre centroides. **Mejor cuanto más pequeño.**
- **Silueta:** mide cercanía a los propios y lejanía del borde. **Mejor cuanto más próximo a +1.**
- **Índice de Dunn:** distancia mínima entre clusters / diámetro máximo de un cluster. **Mejor cuanto más alto.**
- **Estadística pseudo-F:** distancias entre-clusters / intra-clusters. **Mejor cuanto más alto.**

1.4.4 Detección de Anomalías

Eventos raros (fraude, intrusiones, enfermedades raras). Retos: la "normalidad" cambia con el tiempo, la frontera es imprecisa y el ruido se camufla.

A. Tipos de Anomalías

- **Puntuales:** un punto individual alejado del resto.
- **Contextuales / Condicionales:** anómalo solo en su contexto (un bajón de temperatura es normal en invierno, anómalo en verano).
- **Colectivas:** una secuencia o grupo es anómalo en conjunto (arritmia en un ECG) aunque cada punto parezca normal.

B. Técnicas de Detección

- **Basadas en distancia (k-NN):** anomalía = punto con mayor distancia a su k-ésimo vecino. Falla con densidades distintas.

- **Basadas en densidad (LOF, Local Outlier Factor):** compara la densidad de un punto con la de sus vecinos; densidad mucho menor → outlier. Resuelve el problema de densidad variable de k-NN.
- **Basadas en modelos (One-Class SVM):** envuelve las instancias normales (maximizando el margen desde el origen o con una hiperesfera mínima) para aislar la clase objetivo.
- **Basadas en ensembles (Isolation Forest, iForest):** bosques de "árboles de aislamiento" con divisiones aleatorias. Clave: **las anomalías son más fáciles de aislar** y caen en ramas más cortas (menor profundidad) → score de anomalía alto.

1.5 Tema 5. Aprendizaje Profundo. Fundamentos

1.5.1 Del Perceptrón al MLP

- **Redes Neuronales Artificiales (RNA):** núcleo del aprendizaje profundo; versátiles, potentes y escalables. Primer modelo: **McCulloch y Pitts (1943)**, con neuronas de entradas/salida binarias capaces de calcular proposiciones lógicas.
- **Perceptrón (Rosenblatt, 1957):** basado en la **unidad lógica de umbral (TLU/LTU)**. Calcula $z = w^\top x + b$ y le aplica una **función escalón** (Heaviside o signo). Equivale a una clasificación lineal binaria. Una **capa densa / totalmente conectada** conecta cada TLU con todas las entradas.
- **Regla de aprendizaje del perceptrón:** refuerza las conexiones que ayudan a reducir el error; converge **solo si los datos son linealmente separables**.
- **Limitación (Minsky y Papert, 1969):** el perceptrón no resuelve problemas no lineales (p. ej. la función **XOR**), como cualquier modelo lineal.
- **Perceptrón Multicapa (MLP):** apila una **capa de entrada**, una o más **capas ocultas** y una **capa de salida**. Resuelve el XOR y patrones complejos.

1.5.2 Retropropagación (Backpropagation)

- **Retropropagación:** combinación de **diferenciación automática en modo inverso** y **descenso de gradiente**. En dos pasadas calcula el gradiente del error respecto a cada parámetro.
- **Paso hacia delante (forward):** calcula y guarda las salidas de cada capa hasta la salida final.
- **Paso hacia atrás (backward):** mide, vía **regla de la cadena**, cuánto contribuye cada peso al error, propagando el gradiente de salida a entrada.
- **Paso de descenso de gradiente:** ajusta pesos y sesgos para reducir el error.
- **Detalles clave:** se procesa por **minilotes** (p. ej. 32 instancias); una pasada completa al conjunto es una **época**. Los pesos ocultos se **inicializan aleatoriamente** (si no, el entrenamiento falla). La **función escalón** se sustituyó por la **sigmoide** $\sigma(z) = 1/(1 + e^{-z})$ porque tiene derivada no nula y permite progresar al gradiente.
- **Tipos de salida del MLP:** regresión (una neurona por valor a predecir, sin activación o con ReLU); clasificación binaria (1 neurona sigmoide); multiclase (una neurona por clase + **softmax**, con pérdida de **entropía cruzada**).

1.5.3 Problema de los Gradientes Inestables

- **Gradientes que se desvanecen (vanishing):** al retropropagar, los gradientes se hacen cada vez más pequeños en las capas inferiores, que casi no se actualizan y no convergen.

Causa típica: sigmoide + mala inicialización (saturación).
- **Gradientes que explotan (exploding):** crecen sin control y el algoritmo diverge; frecuente en redes recurrentes.
- **Inicialización Glorot (Xavier):** mantiene igual la varianza de entradas y salidas (usa fan_{avg}). Variante **LeCun** (fan_{in}). **Inicialización He (Kaiming):** para ReLU y variantes.

1.5.4 Funciones de Activación

Necesarias y **no lineales**: sin ellas, una pila de capas equivale a una sola transformación lineal.

- **Sigmoide / tanh:** forma de S. La **tanh** va de -1 a 1 (centrada en 0, converge algo mejor que la sigmoide). Saturan en los extremos $\rightarrow$ gradientes que se desvanecen.
- **ReLU** $=$ máx$(0, z)$: rápida y sin saturación para valores positivos; opción por defecto. Problema de las **ReLU moribundas** (neuronas que solo dan 0).
- **Leaky ReLU** $=$ máx$(\alpha z, z)$: pendiente pequeña para $z < 0$ evita la muerte de neuronas. Variantes: **RReLU** (α aleatorio, regulariza) y **PReLU** (α aprendible).
- **ELU:** toma valores negativos (media más cercana a 0), suave en $z = 0$; mejora ReLU pero más lenta.
- **SELU (ELU escalada):** en un MLP de solo capas densas, **autonormaliza** (media 0, desviación 1) y resuelve los gradientes inestables; sujeta a restricciones estrictas.
- **GELU / SiLU / Swish:** variantes suaves de ReLU, no convexas ni monótonas; suelen rendir mejor en tareas complejas a mayor coste. $\text{Swish}_\beta(z) = z\,\sigma(\beta z)$.
- **Recomendación práctica:** ReLU por defecto en tareas simples; Swish/GELU en tareas complejas; SELU en MLP profundos.

1.5.5 Estabilización del Entrenamiento

- **Normalización por lotes (Batch Normalization, BN):** centra en cero y normaliza las entradas de cada capa usando la media y desviación del minilote, y luego las **reescala y desplaza** con dos parámetros aprendibles (γ, β). En test usa medias móviles. Acelera el entrenamiento, reduce la sensibilidad a la inicialización y actúa como **regularizador**.
- **Recorte de gradiente (Gradient Clipping):** recorta los gradientes para que no superen un umbral; mitiga la explosión, sobre todo en redes recurrentes.

1.5.6 Reutilización de Capas y Transfer Learning

- **Transfer Learning:** reutilizar capas de una red preentrenada en un problema similar; ahorra datos y tiempo y mejora la generalización.
- **Pretraining no supervisado:** con muchos datos sin etiquetar, se preentrena capa a capa (p. ej. con autoencoders) y luego se ajusta.
- **Pretraining en tarea auxiliar:** entrenar primero en una tarea con datos etiquetados abundantes y reutilizar lo aprendido.

1.5.7 Optimizadores Más Rápidos

- **SGD (Descenso de Gradiente Estocástico):** actualización básica con la tasa de aprendizaje.
- **Momentum:** acumula un "impulso" de gradientes anteriores; acelera en direcciones consistentes.
- **Nesterov (NAG):** mide el gradiente un poco más adelante en la dirección del momentum; suele converger algo más rápido.

- **AdaGrad:** adapta la tasa por parámetro; buena para problemas simples, pero se frena demasiado pronto en redes profundas.
- **RMSProp:** corrige a AdaGrad usando una media móvil de los gradientes al cuadrado.
- **Adam (Adaptive Moment Estimation):** combina Momentum + RMSProp; opción por defecto muy robusta. Variante **AdaMax** (norma L_∞).
- **Programación del ritmo de aprendizaje (learning rate scheduling):** reducir la tasa durante el entrenamiento. Estrategias: **power scheduling**, **exponencial**, **constante a trozos**, **performance scheduling** (baja al estancarse la validación) y **1-cycle** (sube y luego baja; logra *superconvergencia*).

1.5.8 Regularización en Redes Neuronales

- **Parada anticipada (early stopping)** y la propia **Batch Normalization** ya regularizan.
- **Regularización ℓ_1 / ℓ_2:** ℓ_2 restringe los pesos; ℓ_1 produce modelos dispersos (muchos pesos a 0).
- **Dropout:** en cada paso, cada neurona (salvo las de salida) se "apaga" con probabilidad p (tasa típica 10–50 %). Impide la coadaptación y hace la red más robusta; equivale a promediar un enorme ensemble de subredes. Tras entrenar, se compensan los pesos por $(1-p)$.
- **MC Dropout:** aplica dropout también en test y promedia varias pasadas; mejora el rendimiento y estima la **incertidumbre** sin reentrenar.
- **Max-Norm:** restringe la norma de los pesos entrantes de cada neurona, $\|w\|_2 \leq r$.

1.5.9 Autoencoders

- **Autoencoder:** arquitectura **encoder–decoder** que aprende a reconstruir su entrada pasando por un **espacio latente** comprimido. Usos: compresión, reducción de dimensionalidad, detección de anomalías y pretraining no supervisado.
- **Denoising Autoencoder:** se corrompe la entrada a propósito (ruido o enmascarado) y la red aprende a reconstruir la versión limpia, forzando una codificación robusta.
- **ML vs Deep Learning:** el DL aprende las características automáticamente (*feature learning*) en lugar del diseño manual (*feature engineering*) del ML clásico.

1.6 Tema 6. Aprendizaje Profundo en Diferentes Tipos de Datos

1.6.1 Redes Neuronales Profundas (DNN)

- **DNN:** modelos con muchas capas que aprenden **representaciones jerárquicas** de los datos (características de bajo nivel → medio → alto). Más capas → funciones más complejas.

1.6.2 Redes Neuronales Convolucionales (CNN / ConvNets)

- **CNN:** red con al menos una capa de **convolución**; pensada para datos en forma de matriz/rejilla (imágenes).
- **Convolución:** operación local lineal con una **máscara / filtro / kernel**. El filtro se multiplica elemento a elemento sobre una región, se suman los productos y el resultado forma el **mapa de activación / mapa de características (feature map)**. Ejemplos clásicos: filtro Gaussiano (suavizado), Sobel (bordes).
- **Idea clave:** en una CNN los **coeficientes del filtro se aprenden** (son pesos), no los fija

un experto. (LeNet-5, LeCun 1998.)
- **Capa convolucional vs densa:** la densa conecta todo con todo (muchos pesos); la convolucional opera localmente con muchos menos pesos.

Ventajas de las ConvNets:

- **Conectividad dispersa:** cada unidad "ve" solo una región local → menos operaciones.
- **Compartición de parámetros (weight sharing):** el mismo filtro recorre toda la imagen → pocos parámetros (actúa como **regularización**).
- **Equivarianza a la traslación:** si el objeto se traslada, su representación se traslada igual ($f(g(x)) = g(f(x))$). No es equivariante a escala ni rotación.

Elementos y dimensiones:

- Las capas convolucionales se intercalan con **funciones de activación no lineales** (ReLU y variantes). Los filtros tienen **la misma profundidad** que el volumen de entrada.
- **Stride (paso):** desplazamiento del filtro. **Padding (relleno):** añadir bordes (p. ej. *zero-padding*) para conservar el tamaño y no reducir la dimensionalidad demasiado rápido.
- **Tamaño de salida:** Output $= \big((N + 2P - F)/S\big) + 1$, con N entrada, F filtro, P padding, S stride.
- **Pooling (max / average):** reduce dimensionalidad e introduce **invarianza** a pequeñas traslaciones.
- **Flatten:** aplana el volumen de feature maps a un vector para las capas densas finales.
- **Softmax:** activación final que da la probabilidad de cada clase.

1.6.3 Transformers

Para **secuencias** (texto, series temporales, audio, vídeo): elementos + orden + contexto. Cada elemento se convierte primero en un vector (**embedding**).

Antecedentes y sus límites:

- **Red densa:** trata la entrada como bloque fijo, ignora el orden y no admite longitud variable.
- **CNN:** capta patrones **locales** pero no bien las dependencias largas.
- **RNN (recurrentes):** procesan paso a paso con una memoria (estado oculto); pierden información lejana en secuencias largas.
- **LSTM:** añaden puertas (**olvido**, **entrada**, **salida**) para regular qué recordar; siguen siendo secuenciales (no paralelizables) y lentas.

Mecanismo de Atención:

- **Atención:** cada elemento (token) consulta directamente al resto y combina la información según su relevancia (suma ponderada), en lugar de comprimir todo en una memoria secuencial.
- **Query (Q), Key (K), Value (V):** de cada embedding se generan tres vectores. Q = "qué busco", K = "qué ofrezco", V = "qué información transmito". Se comparan Q con K para decidir a quién atender y se combinan los V.
- **Scaled Dot-Product Attention:** compara Q con K, escala para estabilizar, aplica **softmax** (pesos que suman 1) y mezcla los V.
- **Self-attention:** Q, K y V salen de la misma secuencia; cada token se contextualiza con todos los demás (dependencias largas y directas).
- **Cross-attention:** Q de una secuencia, K y V de otra (conecta lo que se genera con la entrada).
- **Máscaras de atención:** impiden mirar tokens futuros (generación de texto).
- **Multi-head attention:** varias atenciones en paralelo, cada "cabeza" capta un tipo de relación (sintáctica, semántica, posiciones, dependencias lejanas).

Bloque y arquitectura Transformer:

- **Positional encoding:** la atención no conoce el orden → se suma información de posición al embedding. Entrada = contenido + posición.
- **Bloque Transformer:** Multi-head Attention (mezcla info entre tokens) + **Feed-Forward Network (FFN)** (transforma cada token por separado).
- **Conexiones residuales:** suman la entrada original a la transformada (estabilizan el gradiente, evitan olvidar).
- **Layer Normalization:** normaliza cada representación para un entrenamiento estable.
- **Encoder:** Self-attention + Add&Norm + FFN + Add&Norm; produce representaciones contextuales.
- **Decoder:** Masked self-attention + Cross-attention + FFN; genera la salida paso a paso.
- **Arquitecturas:** *encoder-only* (comprender/embeddings), *decoder-only* (generación; base de los LLM), *encoder-decoder* (completa, traducción).

Propiedades, coste y aplicaciones:

- **Ventajas:** capturan dependencias a largo plazo, **procesan en paralelo** (clave para escalar a miles de millones de parámetros) y aprenden relaciones flexibles.
- **Coste:** la atención es **cuadrática** $O(T^2)$ en la longitud T de la secuencia; exige mucha memoria y datos; sin posición no entiende el orden; difícil de interpretar.
- **LLM modernos:** Transformers escalados, con ajuste por instrucciones, alineamiento con preferencias humanas, contexto largo, multimodalidad y uso de herramientas.
- **Aplicaciones:** texto, **series temporales** (Informer, Autoformer, PatchTST, Chronos), **visión** (Vision Transformer: dividir la imagen en parches → embeddings → Transformer).

1.7 Tema 7: Aprendizaje por Refuerzo (Reinforcement Learning)

1. Fundamentos e interfaz agente-entorno

- **Definición:** a diferencia del aprendizaje supervisado, no existen etiquetas previas; el aprendizaje ocurre mediante la interacción entre un agente y su entorno.
- **Elementos clave:** un **agente** observa un **entorno**, ejecuta **acciones** y recibe **recompensas** o penalizaciones.
- **Objetivo:** el agente debe aprender una política π que maximice la recompensa acumulada a lo largo del tiempo.
- **Exploración vs. explotación:** equilibrio entre probar acciones nuevas (exploración) y aprovechar las acciones conocidas que proporcionan recompensa (explotación).

2. Procesos de Decisión de Markov (MDP)

- **Concepto:** formulación matemática que asume la propiedad de Markov (el futuro depende únicamente del estado actual y de la acción tomada).
- **Componentes:** estados (S_t), acciones (A_t), recompensas (R_{t+1}) y probabilidades de transición.
- **Tareas episódicas:** secuencias finitas de interacción que comienzan en un estado inicial y terminan en un estado terminal.

3. Funciones de valor y ecuación de Bellman

- **Retorno (G_t):** suma ponderada y descontada de las recompensas futuras.
- **Función de valor de estado ($V(s)$):** medida de la bondad de un estado bajo una política

dada.
- **Función de valor de acción ($Q(s,a)$):** recompensa esperada tras tomar la acción a en el estado s y seguir la política.
- **Ecuación de Bellman:** relación recursiva fundamental que permite resolver el MDP mediante programación dinámica.

4. Algoritmos clásicos (model-free)

- **Métodos de Monte Carlo (MC):** actualizan valores usando el retorno real al terminar episodios completos.
- **Diferencia Temporal (TD):** actualizaciones paso a paso tras cada acción, sin esperar al final del episodio.
- **Q-Learning (off-policy):** aprende el valor de la política óptima mientras el agente sigue otra política exploratoria.
- **Estrategia ϵ-greedy:** con probabilidad $1-\epsilon$ se elige la mejor acción conocida; con probabilidad ϵ se elige una acción aleatoria para explorar.

5. Aprendizaje por refuerzo profundo (Deep RL)

- **Motivación:** los métodos tabulares no escalan en entornos con espacios de estados muy grandes.
- **Approximate Q-Learning:** usar funciones que aproximen los valores Q en lugar de tablas.
- **Deep Q-Networks (DQN):** redes profundas que aproximan $Q(s,a)$; su éxito permitió superar a humanos en juegos Atari.
- **Replay memory:** búfer de experiencias pasadas para entrenar con mini-lotes aleatorios y estabilizar el aprendizaje.

Consejo práctico para la máxima nota: comprender la ecuación de Bellman y dominar la implementación del bucle de interacción de Q-Learning (entorno → estado → acción → recompensa → actualización), incluyendo el uso de la estrategia ϵ-greedy.

1.8 Tema 8: Optimización en Aprendizaje Automático, Hiperparámetros, Selección de Modelos y Regularización

1. Selección de modelos y ajuste de hiperparámetros

- **Concepto:** los hiperparámetros (p. ej. tasa de aprendizaje, profundidad de un árbol) no se aprenden durante el entrenamiento y requieren búsqueda/exploración.
- **Riesgo de sobreajuste en test:** afinar exclusivamente sobre el conjunto de test induce sobreajuste y degrada el rendimiento en datos nuevos.
- **Técnicas de búsqueda:** Grid Search (exhaustivo), búsqueda aleatoria y optimización bayesiana.
- **Herramientas:** Keras Tuner, Hyperopt, skopt y enfoques AutoML (incluidos algoritmos evolutivos) para explorar arquitecturas y parámetros eficientemente.

2. Regularización

- **Propósito:** restringir la complejidad del modelo para reducir la varianza a cambio de aumentar ligeramente el sesgo, evitando el sobreajuste.
- **Modelos lineales:**
  - **Ridge:** penalización L_2 que mantiene los pesos pequeños.

  - **Lasso:** penalización L_1 que fuerza muchos pesos a cero (modelos dispersos).
- **Redes neuronales:**
  - **Early stopping:** detener el entrenamiento cuando el error de validación deja de mejorar.
  - **Dropout:** apagar aleatoriamente neuronas durante el entrenamiento; **MC Dropout** permite estimar incertidumbre en test.
  - **Max-Norm:** restringe la norma L_2 máxima de los pesos entrantes.
  - **Data augmentation:** generar variaciones realistas del conjunto de entrenamiento.
  - **Batch normalization:** estabiliza gradientes y funciona como regularizador práctico.

3. Optimizadores avanzados

- **Momentum:** acumula gradientes anteriores como impulso.
- **RMSProp:** tasa de aprendizaje adaptativa por dimensión usando la media móvil de los gradientes al cuadrado.
- **Adam:** combina momentos del gradiente y del cuadrado del gradiente (opción por defecto). Variantes: AdaMax, Nadam y AdamW.

4. Tasa de aprendizaje (learning rate)

- **Importancia:** una tasa demasiado baja retrasa la convergencia; una tasa demasiado alta puede provocar divergencia.
- **Búsqueda:** experimentar subiendo exponencialmente la tasa para identificar el umbral antes de que la pérdida se dispare.
- **Schedulers y estrategias:** reducir la tasa durante el entrenamiento; estrategias como **1cycle** (subida y bajada controlada del learning rate y ajuste del momentum) permiten superconvergencia.

Consejo práctico: distinguir que SGD y Momentum usan una tasa fija compartida, mientras que RMSProp y Adam usan tasas adaptativas por dimensión; Grid Search sufre en espacios grandes y conviene optimización bayesiana o librerías específicas.

1.9 Tema 9. Aspectos Avanzados, Retos y Desafíos

1.9.1 Inteligencia Artificial Explicable (XAI)

- **Motivación:** la precisión no basta. Ejemplo clásico **LIME** (husky vs lobo): un clasificador con 94 % de acierto en test miraba la **nieve del fondo**, no el animal (*shortcut learning*). Hay que auditar **qué** aprende el modelo.
- **Tres motivos para XAI: confianza** (medicina, conducción autónoma), **depuración** (sesgos ocultos, *data leakage*, atajos) y **cumplimiento** legal (EU AI Act, derecho a explicación del GDPR).
- **RED vs BLUE XAI:** RED (orientada al modelo): investigar, explorar, depurar y garantizar seguridad. BLUE (orientada al humano): confianza, ética y explicaciones para el usuario.
- **Taxonomía (3 ejes):**
  - **Temporal: Pre-hoc** (modelo interpretable de origen: árboles, reglas, modelos lineales) vs **Post-hoc** (caja negra explicada a posteriori: LIME, SHAP).
  - **Alcance: Local** (una predicción) vs **Global** (todo el modelo).
  - **Modelo: Agnóstico** (cualquier arquitectura) vs **Específico** (requiere acceso interno).
- **Métodos clave:**
  - **LIME:** perturba la entrada localmente y ajusta un modelo simple que aproxima la caja negra en esa vecindad. Agnóstico pero **inestable**.
  - **SHAP:** valores de **Shapley** (teoría de juegos); reparte la contribución de cada caracte-

rística de forma justa (eficiencia, simetría, linealidad, dummy). Coste **exponencial** en número de características.
  - **Contraejemplos (Counterfactuals):** "si tu ingreso fuera 5.000 € mayor, se aprobaría"; intuitivos y alineados con el razonamiento causal.
  - **Integrated Gradients:** integra el gradiente desde una *baseline* hasta la entrada real (específico de redes neuronales).
  - **LRP (Layer-wise Relevance Propagation):** propaga la relevancia hacia atrás capa a capa con reglas de conservación; produce *heatmaps* por píxel.
  - **CRP:** extiende LRP de píxeles a **conceptos** de alto nivel.
  - **ProtoPNet (basado en prototipos):** clasifica diciendo "esto se parece a este prototipo"; la explicación **es** el razonamiento del modelo (XAI intrínseca / RED).
  - **Métricas:** *Quantus* (fidelidad, localización, complejidad, robustez); propuestas DASCI: **REVEL**, **X-SHIELD** (regulariza ocultando características espurias), **STOOD-X** (XAI + detección OOD), **CUBIC** (detecta sesgos conceptuales).
- **EU AI Act:** para sistemas de **alto riesgo** exige trazabilidad, explicabilidad, supervisión humana y documentación. La XAI deja de ser opcional.
- **Desafíos:** las explicaciones post-hoc pueden ser engañosas (*confirmation bias*), no hay métrica única, alto coste (SHAP exponencial), XAI multimodal y de cadenas de agentes.

1.9.2 AI Safety y Detección Out-of-Distribution (OOD)

- **AI Safety:** campo interdisciplinar que previene accidentes, usos indebidos y daños de los sistemas de IA. **AI safety ≠ AI security.**
- **Robustez:** capacidad de mantener el rendimiento frente a perturbaciones y entradas **adversarias** a lo largo del ciclo de vida (anomalías, ataques adversarios, usos maliciosos, puertas traseras).
- **Mundo cerrado vs mundo abierto:** el supuesto de "mundo cerrado" (todas las clases vistas en entrenamiento) provoca sobreocupación del espacio y no se adapta a datos nuevos. El mundo real es **abierto**.
- **Detección Out-of-Distribution (OOD):** identificar entradas que no pertenecen a la distribución de entrenamiento (**ID = in-distribution**). Se distingue **Near-OOD** (parecido, p. ej. CIFAR-10 vs CIFAR-100) de **Far-OOD** (muy distinto, p. ej. EMNIST). Relacionado con el **Open Set Recognition**.

1.9.3 Aprendizaje Continuo (Continual Learning)

- **Definición:** paradigma donde el modelo aprende **secuencialmente** tareas a lo largo del tiempo. Los humanos lo hacen de forma natural; las redes, no.
- **Olvido catastrófico (Catastrophic Forgetting):** al entrenar la Tarea B, el rendimiento en la Tarea A cae drásticamente (caída de *accuracy*).
- **Escenarios: Task-Incremental** (se sabe a qué tarea pertenece el test), **Domain-Incremental** (misma tarea, distinta distribución), **Class-Incremental** (nuevas clases en cada tarea, el más difícil).
- **Backward Transfer (BWT):** métrica del olvido: $\text{BWT} = \frac{1}{N-1}\sum_{j=1}^{N-1}[A(N,j) - A(j,j)]$. $\text{BWT} < 0$ olvido; $= 0$ sin olvido; > 0 transferencia positiva.
- **Estrategias:**
  - **EWC (Elastic Weight Consolidation):** penaliza cambiar los pesos importantes para tareas previas, ponderando por la **Matriz de Información de Fisher (FIM)**. A diferencia de ℓ_2 puro (que penaliza todos los pesos por igual), protege selectivamente.
  - **LoRA y variantes:** adaptadores de bajo rango ($B \in \mathbb{R}^{d\times r}, A \in \mathbb{R}^{r\times k}$) que congelan los

pesos base W_0 y entrenan un adaptador por tarea. Variantes: **O-LoRA** (ortogonaliza adaptadores), **LoRA-Pro**, *Adapter Merging*.

- **Machine Unlearning:** problema inverso: hacer que un modelo **olvide** información específica (p. ej. por el "derecho al olvido", GDPR Art. 17). Reentrenar desde cero es inviable.
  - **SISA (Sharded, Isolated, Sliced, Aggregated):** trocea los datos; para olvidar un dato basta reentrenar el *shard/slice* afectado, dejando intactos los demás.
  - **Algoritmo SSD (Selective Synaptic Dampening):** usa la FIM para identificar y **amortiguar (dampening)** los pesos específicos del dato a olvidar.
  - **Reto:** verificar el olvido (distinguir un dato olvidado de uno nunca visto).

1.9.4 Causalidad en Aprendizaje Automático

- **Correlación ≠ Causalidad:** ejemplo clásico helados vs ahogamientos: su correlación positiva es **espuria**; el **calor** es una **variable confusora (confounder)** que aumenta ambos por separado.
- **Shortcut learning / correlaciones espurias:** el modelo aprende atajos (la nieve para "lobo", el fondo "playa" para una vaca) que no generalizan.
- **Causal Discovery:** inferir la estructura del **grafo causal (DAG, grafo acíclico dirigido)** a partir de dependencias e independencias condicionales, cuando no se puede intervenir. Requisitos: **suficiencia causal** (haber observado todos los confusores) y ausencia de ciclos.
- **Causal Representation Learning:** separar el **espacio latente** en **contenido** Z_C (mecanismos invariantes: la vaca) y **estilo** Z_S (factores espurios: el fondo/playa). Mediante **supervisión débil** e intervenciones sobre Z_S, se aprenden representaciones invariantes.
- **V-REx (Variance Risk Extrapolation):** $R_{\text{V-REx}} = \sum_e R_e + \beta \cdot \text{Var}(\{R_e\})$. Minimiza el riesgo medio y la **varianza** del rendimiento entre entornos, forzando representaciones Z_C invariantes que ignoran los factores espurios Z_S.

Claves de examen (Tema 9). XAI: distinguir pre-hoc/post-hoc, local/global, agnóstico/específico; LIME (inestable, local, agnóstico), SHAP (Shapley, exponencial). OOD: mundo abierto, Near vs Far. Continual learning: olvido catastrófico, EWC+Fisher, BWT negativo = olvido. Causalidad: confounder → correlación espuria; separar contenido invariante del estilo espurio.

Capítulo 2

Glosario

A

- **Aprendizaje Automático (AA):** Ciencia de programar ordenadores para que aprendan de los datos sin ser programados explícitamente. Se enfoca en algoritmos cuyo rendimiento en una tarea mejora con la experiencia.
- **Aprendizaje Basado en Instancias:** Sistema que memoriza los ejemplos de entrenamiento y clasifica nuevos casos empleando una medida de similitud (como la distancia matemática), sin requerir una fase de entrenamiento previo.
- **Aprendizaje Basado en Modelos:** Sistema que construye una función matemática (modelo) a partir de los datos para generalizar y hacer predicciones minimizando un error.
- **Aprendizaje Continuo (Continual Learning):** Paradigma donde el modelo aprende múltiples tareas secuencialmente en el tiempo. Su mayor reto es mantener el rendimiento en tareas antiguas sin reentrenar desde cero.
- **Aprendizaje No Supervisado:** Entrenamiento con datos no etiquetados donde el algoritmo busca patrones ocultos. Sus tareas principales incluyen agrupamiento (clustering), reducción de dimensionalidad y detección de anomalías.
- **Aprendizaje por Refuerzo:** Método donde un agente interactúa con un entorno tomando acciones y recibiendo recompensas o penalizaciones, optimizando una política que maximice las recompensas acumuladas a largo plazo.
- **Aprendizaje Supervisado:** Método que utiliza datos previamente etiquetados con las soluciones deseadas, principalmente para resolver tareas de clasificación y regresión.
- **Árboles de Decisión (CART):** Modelos no paramétricos y basados en reglas que dividen el espacio buscando reducir la impureza (Gini/Entropía). Son altamente propensos al sobreajuste si no se restringen sus hiperparámetros (ej. controlando su profundidad máxima).
- **Atención (Mecanismo de):** Permite conexiones directas entre los elementos de una secuencia, calculando matemáticamente (mediante Queries, Keys y Values) cuánto debe atender cada componente al resto, reemplazando así la memoria secuencial por comunicación directa.
- **AUC (Área Bajo la Curva):** Métrica asociada a la curva ROC. Un clasificador perfecto obtiene un AUC de 1, mientras que un clasificador puramente aleatorio obtiene un 0.5.
- **Autoencoders:** Arquitecturas de red neuronal divididas en encoder y decoder que aprenden representaciones latentes al comprimir la información y reconstruirla, siendo muy eficaces en tareas como la reducción de ruido.

B

- **Bagging:** Técnica de ensamblado (ensemble) que reduce la varianza entrenando el mismo algoritmo en diversos subconjuntos de datos muestreados con reemplazo (bootstrap).
- **Batch Normalization (Normalización por Lotes):** Capa en redes profundas que centra, normaliza y escala las entradas según su minilote. Previene la inestabilidad de los gradientes, acelera la convergencia y actúa como regularizador.

- **Boosting:** Método de ensamblado donde los modelos débiles se entrenan secuencialmente, y cada nuevo predictor intenta corregir los fallos residuales de su predecesor (ej. AdaBoost).

C

- **Causal Representation Learning:** Intento de descomponer el espacio latente de la IA separando los factores causales invariantes (el contenido real) de los factores espurios o engañosos (el entorno o estilo).
- **Clustering (Agrupamiento):** Técnica no supervisada que asigna instancias a grupos subyacentes de alta similitud. Algoritmos destacados incluyen K-Means y DBSCAN.
- **Curva ROC:** Gráfico que evalúa el rendimiento de clasificadores binarios mostrando el compromiso entre la tasa de verdaderos positivos (recuperación) y la tasa de falsos positivos a través de distintos umbrales de decisión.

D

- **Data-Centric AI:** Enfoque de la ingeniería de IA donde, manteniendo arquitecturas de modelo estandarizadas, el esfuerzo principal se dedica a limpiar, adquirir y mejorar la calidad de los datos.
- **Detección de Anomalías:** Tarea no supervisada encargada de aprender la frontera de normalidad de un dataset para identificar valores raros, atípicos o ruidosos.
- **Discretización:** Transformación de características numéricas continuas en variables categóricas de intervalos, vital para algoritmos incapaces de procesar campos numéricos puros.
- **Dropout (Abandono):** Poderoso regularizador para redes neuronales en el que cada neurona tiene una probabilidad específica de apagarse en cada paso del entrenamiento, previniendo coadaptaciones frágiles y forzando a aprender características más generalizables.

E

- **Early Stopping (Parada Anticipada):** Estrategia de regularización para optimizaciones iterativas que detiene el algoritmo de entrenamiento cuando el error en el conjunto de validación llega a su mínimo y comienza a empeorar.
- **Ecuación de Bellman:** Ecuación recursiva en el corazón del aprendizaje por refuerzo que conecta el valor de un estado con el valor esperado de los estados futuros subsiguientes.
- **Ensembles (Ensamblados):** Consisten en fusionar múltiples modelos individuales (aprendices) con el propósito de generar una predicción final con menor error y mayor robustez, guiados por la "sabiduría de la multitud".
- **EWC (Elastic Weight Consolidation):** Técnica usada en el aprendizaje continuo que penaliza el cambio de los pesos críticos pertenecientes a tareas pasadas (usando la Matriz de Información de Fisher) para evadir el olvido catastrófico.

F

- **F1 Score:** Métrica de clasificación construida como la media armónica entre la precisión y la recuperación. Favorece a los modelos que logran un balance equilibrado entre ambas, penalizando fuertemente si una de ellas es baja.

G

- **Gradient Clipping (Recorte de Gradientes):** Limitar artificialmente la magnitud de los gradientes a un umbral preestablecido durante la retropropagación. Fundamental en redes recurrentes para evitar explosiones matemáticas.

H

- **Hiperparámetro:** Un parámetro estructural del propio algoritmo de aprendizaje o de la regularización, definido externamente antes del entrenamiento, y que el modelo no puede

aprender ni actualizar de los datos.

I

- **IA Explicable (XAI):** Dominio orientado a auditar y entender los razonamientos internos de cajas negras. Incluye taxonomías como local/global, pre-hoc/post-hoc y agnóstico/específico.

L

- **Lasso (Norma L1):** Regularización de modelos lineales que restringe los pesos aplicando penalizaciones absolutas. Posee la propiedad natural de reducir a cero exacto los atributos inútiles, logrando selección automática de características.
- **LIME:** Algoritmo agnóstico de explicabilidad local (XAI) que perturba las entradas alrededor de una predicción y ajusta un modelo muy simple sobre ella para descubrir en qué se basó la caja negra.

M

- **MAE (Error Medio Absoluto):** Métrica de evaluación para regresiones cimentada en la norma Manhattan (L1); su gran ventaja respecto al MSE es que no es tan sensible al ruido generado por los valores atípicos (outliers).
- **Model-Centric AI:** Enfoque tradicional que centra sus iteraciones en modificar la arquitectura y los hiperparámetros del código del modelo, asumiendo el dataset como estático.

O

- **Olvido Catastrófico (Catastrophic Forgetting):** Defecto severo inherente a las redes neuronales estándar, en el cual sobrescriben el conocimiento de una tarea anterior al aprender una tarea secuencial nueva mediante el descenso de gradiente.
- **OOD (Out-of-Distribution):** Situaciones en las que los datos proporcionados al modelo provienen de distribuciones marginales distintas a las vistas en entrenamiento, suponiendo vulnerabilidades enormes en un contexto de mundo abierto.
- **Overfitting (Sobreajuste):** Circunstancia en la que un modelo es excesivamente complejo y se ajusta meticulosamente al ruido subyacente de la muestra de entrenamiento, impidiéndole generalizar correctamente ante instancias inexploradas.

P

- **Pasting:** Algoritmo de ensamblado homólogo al bagging, con la particularidad de que los subconjuntos aleatorios de entrenamiento se seleccionan estricta y obligatoriamente sin reemplazo.
- **PCA (Análisis de Componentes Principales):** Técnica canónica no supervisada para reducir la dimensionalidad. Proyecta los datos matemáticamente (vía SVD) hacia un hiperplano que consigue preservar su máxima varianza.
- **Precisión (Precision):** Métrica en clasificación que cuantifica la proporción de aciertos exclusivamente entre todos aquellos elementos que el modelo predijo como positivos.

R

- **Recuperación / Sensibilidad (Recall):** Fracción matemática de instancias que efectivamente pertenecían a la clase positiva que fueron correctamente identificadas por el modelo.
- **Redes Neuronales Convolucionales (CNN):** Modelos caracterizados por su conectividad dispersa y compartición de pesos mediante kernels, arquitectónicamente superiores para extraer características automatizadas de matrices (imágenes).
- **Regularización:** Constreñir intencionalmente los grados de libertad de un modelo paramétrico para restringir su complejidad o reducir los pesos, actuando como cura principal frente al sobreajuste.

- **Ridge (Norma L2):** Método de regularización lineal que restringe la magnitud de los pesos penalizándolos cuadráticamente. A diferencia del Lasso, minimiza los valores pero raramente los anula por completo.
- **RMSE (Raíz del Error Cuadrático Medio):** Medida evaluativa prioritaria en regresión apoyada en la norma Euclídea; es fuertemente sensible ya que eleva al cuadrado y amplifica los errores grandes.

S

- **Selección de Características (Feature Selection):** Proceso mediante algoritmos Filtro o Envolventes que determina qué subconjunto óptimo de variables (columnas) debe utilizarse, minimizando la complejidad del sistema final.
- **Selección de Instancias:** Optimización orientada al recorte de filas en la base de datos, reteniendo instancias cruciales de las fronteras de decisión y descartando duplicados o ruidosos (vital en el k-NN).
- **Self-Attention:** Mecanismo transformacional donde los elementos (Queries, Keys y Values) son extraídos de la misma secuencia de entrada, forzando a los tokens a interactuar unos con otros sin la dependencia temporal del orden lineal.
- **SHAP:** Sistema XAI fundamentado en la teoría de juegos (Valores de Shapley); asegura explicaciones justas garantizando el peso y contribución marginal exacta que cada variable brindó a la respuesta final.
- **SISA:** Paradigma arquitectónico de Machine Unlearning que segmenta el aprendizaje en fragmentos independientes (Sharded) para admitir aislar y olvidar características concretas dictadas por las leyes de privacidad, sin incurrir en costes inabarcables por reentrenamiento completo.

T

- **Teorema "No Free Lunch" (NFL):** Postulado que garantiza que sin adoptar asunciones previas sobre un conjunto de datos, ningún algoritmo computacional ostenta superioridad sobre los demás; se ha de probar o asumir empíricamente.
- **Transformer:** Módulo fundacional paralelo en aprendizaje profundo orientado a secuencias; descarta toda recurrencia interna estructurándose en mecanismos Self-Attention paralelos en conjunto con codificación posicional para conservar el orden.

U

- **Underfitting (Subajuste):** Circunstancia opuesta al sobreajuste en donde el modelo estadístico es demasiado ingenuo (baja capacidad representativa o excesiva regularización) como para aprender o ajustarse ni siquiera a los datos de entrenamiento base.