Clase 050 — Panorama del ML: tipos, batch vs online, instance vs model-based

Parte: 1 — Machine Learning Clásico · Fuente: Géron, cap. 1. Duración estimada: 60 min.

Clase 050 — Panorama del ML: tipos, batch vs online, instance vs model-based

Parte: 1 — Machine Learning Clásico · Fuente: Géron, cap. 1. Duración estimada: 60 min.

Objetivo

Que el alumno arme un mapa mental claro del campo de ML antes de entrar en algoritmos concretos: qué tipos de aprendizaje existen, en qué se diferencia entrenar de una sola vez (batch) vs en streaming (online), y por qué algunos modelos "memorizan ejemplos" (instance-based) y otros "abstraen una regla" (model-based). Sirve de andamio para ubicar cada algoritmo de las próximas clases dentro de esta taxonomía.

Resultados de aprendizaje

Al finalizar la clase, el alumno podrá:

1. Clasificar un problema dado como supervisado, no supervisado, semi-supervisado o reforzado, justificando con la pinta de los datos (¿hay etiquetas?, ¿hay recompensa?).
2. Decidir batch vs online según el volumen de datos, la velocidad a la que cambia la distribución y el costo de reentrenar.
3. Distinguir instance-based de model-based y reconocer cuál usa scikit-learn por debajo en un algoritmo dado (KNN vs regresión lineal, por ejemplo).
4. Detectar señales de mala generalización (overfitting / data drift) en términos del marco del cap. 1 — anticipo de la clase 048.
5. Ubicar cualquier algoritmo del curso dentro de la grilla (supervisión × batch/online × instance/model) sin googlear.

Temas

#	Tema	Por qué importa
1	Qué es ML (Samuel 1959, Mitchell 1997)	Definición operativa: aprender = mejorar e
2	Aprendizaje supervisado	El 80% de lo que vas a tocar en la industr
3	Aprendizaje no supervisado	Clustering, reducción de dim, detección de
4	Semi-supervisado y auto-supervisado	Etiquetar es caro; mezclás un poco de labe
5	Aprendizaje por refuerzo	Otro paradigma (agente + recompensa); útil
6	Batch vs online learning	Reentrenar de cero vs aprender incremental
7	Instance-based vs model-based	Dos formas de generalizar: por similitud c

Definiciones y características

Aprendizaje supervisado

: El dataset trae cada instancia con su etiqueta (y). El algoritmo aprende un mapeo f: X → y. Sub-divisiones: clasificación (y categórica) y regresión (y continua). Ejemplos: spam filter, predicción de precio de casa.

Aprendizaje no supervisado

: Solo hay X, sin y. El algoritmo busca estructura: clusters (k-means), componentes (PCA), densidad (DBSCAN), anomalías (isolation forest). Métricas de éxito son menos directas que en supervisado.

Aprendizaje semi-supervisado

: Mezcla — pocas instancias etiquetadas + muchas sin etiqueta. Típico cuando etiquetar es caro (imagenes médicas, audio transcripto). Google Photos lo usa para clusterear caras y pedirte el nombre solo del cluster.

Aprendizaje por refuerzo (RL)

: Un agente observa el entorno, toma acciones, recibe recompensa (positiva o negativa), aprende una política que maximiza recompensa acumulada. Distinto de supervisado: no hay "respuesta correcta", solo señal de recompensa. Ej: AlphaGo, robots, trading bots.

Batch learning (offline)

: Se entrena con todo el dataset de una sola pasada. El modelo queda congelado en producción. Para incorporar datos nuevos hay que reentrenar desde cero. Simple, reproducible, pero pesado si el dataset crece o la distribución cambia.

Online learning (incremental)

: El modelo se actualiza con mini-batches o instancias una a la vez (partial_fit en sklearn). Ideal para streaming (precios, clicks) o datasets que no caben en RAM (out-of-core). El parámetro clave es el learning rate: alto = se adapta rápido pero olvida; bajo = estable pero lento.

Instance-based learning

: El sistema "memoriza" las instancias de entrenamiento y generaliza comparando una nueva instancia con las vistas, vía una medida de similitud. Ejemplo canónico: KNN. No hay parámetros aprendidos — el "modelo" es el propio dataset.

Model-based learning

: Se asume una forma funcional (lineal, árbol, red neuronal) con parámetros θ, y se ajustan los θ minimizando una función de costo sobre el train set. Una vez entrenado, podés tirar el dataset: el modelo es autocontenido. Ejemplos: regresión lineal/logística, random forest, redes neuronales.

Generalización

: Capacidad del modelo de funcionar bien en datos nuevos (no vistos en entrenamiento). Es el objetivo real de ML — no la performance en train. Se mide con un test set separado. Falla por dos motivos: overfitting (memorizó ruido) o underfitting (modelo demasiado simple). Tema central de la clase 048.

Dataset / recursos

Para los ejercicios prácticos: California Housing (sklearn.datasets.fetch_california_housing) — regresión, supervisado, ~20k filas. Y Iris (sklearn.datasets.load_iris) — clasificación clásica, 150 filas, perfecto para comparar KNN (instance-based) vs LogisticRegression (model-based) en pocos segundos.

Lectura de fondo: Géron, cap. 1 entero (~25 páginas). Mirá especialmente las figuras 1-13 a 1-17, que son el resumen visual de toda la taxonomía.

Ejercicios

1. Clasificá 6 problemas reales. Para cada uno indicá supervisado/no-supervisado/semi/RL y por qué: (a)

detectar fraude en transacciones con tarjeta; (b) segmentar clientes de un e-commerce; (c) traducir inglés→español; (d) jugar al ajedrez; (e) detectar caras duplicadas en una galería de 10k fotos donde solo 50 están taggeadas; (f) predecir el precio del dólar a 7 días.

2. Batch o online. Decidí qué corresponde y justificá en una línea: (a) modelo de scoring crediticio que se reentrena trimestralmente; (b) recomendador de noticias que reacciona a clicks en tiempo real; (c) clasificador de imágenes médicas en un hospital; (d) detector de spam en Gmail.

3. KNN vs LogReg en Iris. Entrená un KNeighborsClassifier(n_neighbors=5) y un LogisticRegression(max_iter=1000) sobre iris. Compará: accuracy en test, tamaño del modelo serializado (pickle.dumps), tiempo de inferencia sobre 1000 predicciones. ¿Cuál es instance-based? Verificalo mirando el tamaño.

4. Out-of-core con SGDRegressor. Cargá California Housing y entrená un SGDRegressor en mini-batches de 500 filas usando partial_fit en un loop. Plotteá el MSE en train a medida que pasan los batches. Es el patrón canónico de online learning.

5. Mapa mental. En papel (o draw.io), dibujá una grilla 2×2×2 con los ejes supervisión / batch-online / instance-model. Ubicá: KNN, regresión lineal, k-means, random forest, SGDClassifier, AlphaGo, autoencoder. Algunos van a quedar en celdas raras — anotá por qué.

Homework verificable

Notebook que: (a) carga Iris y California Housing; (b) entrena 4 modelos — KNeighborsClassifier, LogisticRegression, KNeighborsRegressor, SGDRegressor con partial_fit en 10 batches; (c) reporta para cada uno: tipo de aprendizaje (sup/no-sup), batch o online, instance o model-based, métrica en test (accuracy o RMSE) y tamaño en bytes del modelo serializado; (d) produce una tabla resumen en markdown al final del notebook.

Criterio de aceptación: La tabla final clasifica correctamente los 4 modelos en los 3 ejes y muestra que el KNN ocupa significativamente más bytes que LogReg para el mismo problema (evidencia empírica de que es instance-based).

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
"Mi modelo da 99% en train y 60% en test"	Overfitting clásico. El modelo memorizó el
partial_fit da resultados muy distintos a	Online learning depende del orden de los b
KNeighborsClassifier tarda una eternidad e	Es instance-based — comparar contra todo e
"El test accuracy fluctúa cada vez que cor	No fijaste random_state en el train_test_s
Confundir "no supervisado" con "sin entren	k-means igual entrena (ajusta centroides).

Preguntas frecuentes

¿Semi-supervisado es lo mismo que self-supervised?

No. Semi-supervisado = pocas etiquetas + muchos datos sin etiquetar, en un problema supervisado clásico. Self-supervised (auto-supervisado) = el modelo se inventa la etiqueta a partir del propio dato (ej: BERT enmascara palabras y predice la enmascarada). Self-supervised es el motor de los LLMs modernos.

¿Cuándo elijo online learning en serio, no como juguete?

Tres casos: (1) streaming real donde los datos llegan continuamente y la distribución cambia (clicks, precios, sensores IoT); (2) out-of-core — el dataset no entra en RAM y entrenar offline en disco es prohibitivo; (3) adaptación rápida a concept drift sin reentrenar de cero. Para el 90% de problemas tabulares estáticos, batch es más simple y suficiente.

¿KNN realmente no entrena nada?

Técnicamente fit solo guarda el train set (con un índice como KD-tree opcional). Todo el trabajo se hace en predict. Por eso se le dice "lazy learner". Contraste total con regresión lineal, donde fit calcula los coeficientes y predict es una multiplicación.

¿Random forest es instance o model-based?

Model-based. Aunque "guarda árboles" — esos árboles son la forma funcional aprendida, no las instancias originales. Una vez fiteado, podés tirar el train set y el modelo sigue funcionando solo con los árboles. KNN no — sin el train set, KNN no existe.

¿Por qué arrancamos con este panorama antes de meternos en código?

Porque sin el mapa, cada algoritmo nuevo se siente desconectado del anterior. Con el marco del cap. 1, cuando lleguemos a SVM (clase ~055) ya sabés que es supervisado, batch, model-based — y eso te dice automáticamente qué API de sklearn esperar, cómo evaluarlo y cuáles son sus debilidades genéricas. El marco ahorra horas más adelante.

Referencias

- Géron, A. Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow (3rd ed., O'Reilly, 2022), cap. 1 — The Machine Learning Landscape.
- scikit-learn User Guide — Supervised learning
- scikit-learn User Guide — Unsupervised learning
- scikit-learn — Scaling strategies (out-of-core) — base teórica de partial_fit.
- Mitchell, T. Machine Learning (McGraw-Hill, 1997) — definición clásica de aprendizaje (T, P, E).
- Distill — A visual exploration of Gaussian Processes — bonus para entender model-based no paramétrico.

Siguiente clase

Clase 051 — Desafíos del ML: overfitting, underfitting, datos insuficientes

Apéndice: notebook (primer bloque)

Primera celda ejecutable del notebook de la clase.

```
# Imports y configuración inicial
```

Archivos complementarios

- notebook.ipynb