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

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

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

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

Objetivo

Que el alumno identifique los seis problemas que hacen fracasar un proyecto de ML — datos insuficientes, no representativos, de mala calidad, features irrelevantes, overfitting y underfitting — y sepa qué herramienta aplicar a cada uno (más datos, mejor muestreo, limpieza, feature engineering, regularización, o un modelo más expresivo). El eje conceptual es el bias-variance tradeoff y la intuición de que "el modelo memorizó vs. generalizó".

Resultados de aprendizaje

Al finalizar la clase, el alumno podrá:

1. Distinguir overfitting de underfitting mirando la brecha entre error de entrenamiento y error de validación.
2. Diagnosticar cuál de los seis desafíos de Géron está rompiendo un pipeline concreto.
3. Aplicar regularización (L1/L2, reducir capacidad del modelo, más datos) como contramedida al overfitting.
4. Detectar sampling bias y data snooping bias antes de medir performance.
5. Justificar por qué "más datos" suele ganarle a "modelo más complejo" (Banko & Brill 2001 / "The Unreasonable Effectiveness of Data").

Temas

#	Tema	Por qué importa
1	Datos insuficientes	Hasta el algoritmo más simple necesita vol
2	Datos no representativos (sampling bias)	Si el train no se parece al deploy, el mod
3	Datos de mala calidad (outliers, NaN, ruid	Garbage in, garbage out. La limpieza es 60
4	Features irrelevantes / feature engineerin	Modelos buenos con features malos pierden
5	Overfitting	El modelo aprendió el ruido del train. Baj
6	Underfitting	El modelo es demasiado rígido para captar
7	Regularización (L1/L2, early stopping, dro	Solución estándar al overfitting cuando no

Definiciones y características

Overfitting

: El modelo performa muy bien en train y mal en test. Memorizó ejemplos en vez de extraer patrones. Síntoma típico: train_score test_score. Causas: modelo demasiado complejo, pocas observaciones, ruido en los labels. Fixes: más datos, regularización, reducir capacidad, early stopping.

Underfitting

: El modelo es demasiado simple para la estructura del dato. Performa mal tanto en train como en test. Síntoma: ambos scores bajos y parecidos. Fix: modelo más expresivo, más features, menos regularización, o aceptar que la señal no está.

Regularización

: Restricción explícita sobre los parámetros del modelo para que no se ajuste al ruido. En lineales: penalizar la norma de los coeficientes (Ridge = L2, Lasso = L1). En árboles: limitar profundidad. En redes: dropout, weight decay. Controla la variance a costa de aceptar algo más de bias.

Bias-variance tradeoff

: Descomposición del error de generalización en tres términos: error = bias² + variance + irreducible_noise. Bias = error por supuestos del modelo (un lineal sobre una curva). Variance = sensibilidad a la muestra de entrenamiento (cambias el train un poco y el modelo cambia mucho). Reducir uno suele subir el otro — el arte del ML es encontrar el sweet spot.

Sampling bias

: La muestra de entrenamiento no es representativa de la población. Caso clásico: encuesta Literary Digest 1936, predijo Landon ganador porque encuestaron a suscriptores con teléfono (sesgo socioeconómico). En ML moderno: entrenar reconocimiento facial con caras mayormente blancas y desplegarlo global.

Data snooping bias

: Mirar el test set para tomar decisiones de modelado (qué features incluir, qué hiperparámetros probar). Inflás artificialmente la performance reportada porque el "test" ya contaminó tus elecciones. Regla: separá test al inicio y no lo toques hasta el final.

Feature engineering

: Proceso de construir features informativos a partir del raw data. Incluye selección (descartar irrelevantes), extracción (combinar features existentes), y creación (fecha → día_de_semana, es_finde, mes). Históricamente, más palanca que cambiar de modelo.

Dataset / recursos

Demos sintéticas con sklearn.datasets.make_regression y make_classification para visualizar curvas de aprendizaje (train vs. validation a medida que crece N) y curvas de validación (score vs. hiperparámetro de capacidad). No hace falta dataset externo — la clase es conceptual + diagnóstica.

Ejercicios

1. Diagnóstico visual. Generá un dataset con make_regression(n_samples=50, noise=20). Ajustá PolynomialFeatures(degree=15) + LinearRegression. Compará train_score y test_score. ¿Es overfitting o underfitting? Repetí con degree=1 sobre datos no lineales.

2. Learning curve. Usá sklearn.model_selection.learning_curve sobre un dataset. Plotea train_score y val_score vs. tamaño del train. Identificá: (a) si la brecha se cierra con más datos → vale la pena conseguir más, (b) si las dos convergen bajo → modelo demasiado simple.

3. Ridge vs. Lasso. Sobre el mismo dataset polinomial del ej. 1, ajustá Ridge(alpha=...) para alpha [0.001, 0.01, 0.1, 1, 10, 100]. Plotea train_score y test_score vs. alpha (curva de validación). Encontrá el sweet spot.

4. Sampling bias simulado. Generá un dataset clasificación binaria balanceado (n_samples=10000). Entrená

un modelo con un train sesgado (90% clase 0, 10% clase 1) y testealo en un test balanceado. Reportá accuracy global y por clase. ¿Qué te dice el accuracy global solo?

5. Feature engineering manual. Para un dataset (fecha_timestamp, monto), predecí monto (a) usando solo timestamp como número, (b) extrayendo día_semana, mes, es_finde. Compará R^2 — la diferencia es el valor de feature engineering.

Homework verificable

Notebook que tome sklearn.datasets.fetch_california_housing, entrene tres modelos: (a) LinearRegression baseline, (b) LinearRegression sobre PolynomialFeatures(degree=3) sin regularizar, (c) Ridge(alpha=10) sobre las mismas polynomial features. Para cada uno reportá train_R^2 y test_R^2 con train_test_split(random_state=42). Plotea las tres barras lado a lado.

Criterio de aceptación: El modelo (b) debe mostrar overfitting evidente (train alto, test bajo o negativo). El modelo (c) debe tener test_R^2 mejor que (b) y train_R^2 menor que (b) — evidencia de que la regularización trabajó. El test set se separa al principio y no se usa para tunear alpha (eso va por CV en el train).

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
train_score = 0.99, test_score = 0.40	Overfitting clásico. Fix: regularizá (Ridg
Ambos scores bajos (0.30 train, 0.28 test)	Underfitting. Fix: modelo más expresivo, m
Reportás accuracy=0.95 en clases desbalanc	El modelo predice siempre la mayoritaria.
Tuneás alpha mirando el test set	Data snooping. El test ya no es test. Fix:
Ridge no cambia nada vs. LinearRegression	No escalaste los features. Ridge penaliza

Preguntas frecuentes

¿Más datos o modelo más complejo?

Más datos, casi siempre. Banko & Brill (2001) mostraron que algoritmos mediocres con mucho dato superan a algoritmos sofisticados con poco. Solo cuando estás bloqueado para conseguir más datos vale la pena pelear con la arquitectura.

¿Ridge o Lasso?

Ridge (L2) si querés conservar todos los features y solo bajarles la magnitud. Lasso (L1) si querés selección automática — fuerza coeficientes a 0 exactos. ElasticNet mezcla las dos.

¿Cómo sé si tengo "datos insuficientes"?

Plotea la learning curve. Si val_score sigue subiendo cuando agregás más train → te faltan datos. Si ya hizo plateau → el cuello de botella es otro (modelo, features, ruido).

¿El test set se puede usar más de una vez?

No para tomar decisiones. Sí al final, para reportar performance. Si tunear hiperparámetros y elegir features mirando el test, ese número deja de ser un estimador honesto. Para tuning usá CV sobre el train o un validation set aparte.

¿Overfitting siempre es malo?

Casi siempre. La excepción rara: cuando deploy y train son el mismo universo cerrado (overfitting a una tabla fija que nunca crece). En ML real, donde llegan datos nuevos, overfitting = modelo frágil que falla en producción.

Referencias

- Géron, cap. 1 § "Main challenges of machine learning" y § "Testing and validating".
- sklearn — Validation curves
- sklearn — Underfitting vs. overfitting
- Banko & Brill (2001), Scaling to very very large corpora for natural language disambiguation.
- Halevy, Norvig & Pereira (2009), The Unreasonable Effectiveness of Data.

Siguiente clase

Clase 052 — Testing, validación, hyperparameter tuning, no free lunch theorem

Apéndice: notebook (primer bloque)

Primera celda ejecutable del notebook de la clase.

```
# Imports y configuración inicial
```

Archivos complementarios

- notebook.ipynb