Clase 072 — Curvas de aprendizaje y bias-variance tradeoff

Parte: 1 — Machine Learning Clásico · Fuente: Géron, cap. 4. Duración estimada: 60 min.

Clase 072 — Curvas de aprendizaje y bias-variance tradeoff

Parte: 1 — Machine Learning Clásico · Fuente: Géron, cap. 4. Duración estimada: 60 min.

Objetivo

Diagnosticar si un modelo sufre de alto sesgo o alta varianza leyendo curvas de aprendizaje (sklearn.model_selection.learning_curve) y decidir, con criterio, si conviene conseguir más datos, aumentar la capacidad del modelo o regularizar.

Resultados de aprendizaje

Al finalizar, el estudiante podrá:

1. Graficar una curva de aprendizaje (RMSE train vs. RMSE validación en función de train_size) con learning_curve.
2. Identificar patrones canónicos: underfitting (curvas altas y juntas) vs. overfitting (gap persistente).
3. Descomponer conceptualmente el error esperado en bias² + variance + irreducible noise.
4. Decidir acción correctiva apropiada: más datos, más capacidad, features nuevas, o regularización.
5. Justificar por qué "más datos" no siempre es la solución (caso de high bias).

Temas

- Curva de aprendizaje: qué se plotea y cómo se lee.
- Patrón de underfitting: ambas curvas convergen alto → más datos no ayuda.
- Patrón de overfitting: gap grande train/val → más datos sí ayuda, o regularizar.
- Descomposición bias-variance del error de generalización.
- Error irreducible (ruido de Bayes): cota inferior inevitable.
- Tradeoff: aumentar capacidad ↓ bias pero ↑ variance.
- Diagnóstico operacional con learning_curve y validation_curve.

Definiciones y características

- Learning curve (curva de aprendizaje) — gráfico del error de entrenamiento y validación en función del tamaño del conjunto de entrenamiento. Permite diagnosticar capacidad vs. datos.
- Bias (sesgo) — error por supuestos erróneos del modelo (p. ej., asumir lineal lo no-lineal). Modelos con alto sesgo subajustan.
- Variance (varianza) — sensibilidad del modelo a pequeñas variaciones en los datos de entrenamiento. Modelos con alta varianza sobreajustan.
- Irreducible error — ruido intrínseco de los datos (Var(ε)); ninguna mejora de modelo lo elimina. Es la cota inferior.
- Diagnóstico high-bias — train_error alto y val_error alto, ambas curvas convergen a un valor alto cuando m crece. Síntoma: más datos no mueven la aguja.
- Diagnóstico high-variance — train_error bajo, val_error notablemente más alto, gap persistente. Síntoma: el modelo memoriza; más datos o regularización deberían cerrar el gap.
- Bias-variance tradeoff — $E[(y - \hat{y})^2] = \text{Bias}^2 + \text{Var} + \sigma^2$. Reducir uno suele aumentar el otro; el óptimo

está en el medio.

- Capacidad del modelo — grados de libertad efectivos (grado del polinomio, profundidad del árbol, nº de parámetros). Más capacidad menos bias, más variance.

Dataset / recursos

- Dataset sintético tipo "noisy quadratic": y = 0.5·x² + x + 2 + ε con np.random.randn. Permite controlar la complejidad
- sklearn.datasets.fetch_california_housing (subset) para una corrida sobre datos reales.
- API clave: sklearn.model_selection.learning_curve, validation_curve.

Ejercicios

1. Curva base. Generá 200 puntos del dataset cuadrático ruidoso. Ajustá una regresión lineal y graficá la curva de aprendiz
1. Aumentar capacidad. Repetí con PolynomialFeatures(degree=2) + LinearRegression. Comparalo con degree=10. Identificá cu
1. ¿Más datos ayudan? Para el polinomio de grado 10, extendé el dataset a 2000 puntos y volvé a plotear. ¿Se cierra e
1. Validation curve. Usá validation_curve para barrer degree de 1 a 15 sobre el mismo dataset. Encontrá el "sweet spot
1. Bias-variance empírico. Entrená 100 modelos degree=10 sobre bootstraps del dataset y calculá, para una grilla de x de test

bias² + var se acerca al MSE total menos σ².

Homework verificable

Sobre el dataset california_housing:

1. Entrená DecisionTreeRegressor(max_depth=d) para d {2, 5, 10, None}.
2. Para cada uno, generá la curva de aprendizaje con learning_curve(cv=5, scoring='neg_root_mean_squared_error').
3. Clasificá cada modelo como underfit, fit razonable o overfit justificando con el gap final y el nivel de error.
1. Recomendá explícitamente, para cada caso, una de tres acciones: ["más datos", "más capacidad", "regularizar/podar"].

Criterio de aceptación: un .py o notebook que, al ejecutarse, imprima una tabla con columnas depth | train_rmse_final | val_rmse_final | gap | diagnostico | accion_recomendada y guarde los 4 gráficos en figs/learning_curve_depth_{d}.png.

Errores comunes

- Confundir curva de aprendizaje con validation curve. La primera varía m (tamaño de train); la segunda varía un hiperparám
- Leer el error de train absoluto como diagnóstico. Lo relevante es el gap y la tendencia asintótica, no un punto ai
- Recomendar "más datos" frente a high bias. Si ambas curvas ya convergieron alto, sumar filas no baja el error; hay que ca
- No promediar sobre CV. Una sola partición es ruidosa; learning_curve ya hace CV interno — usalo.
- Olvidar el irreducible error. Apuntar a val_rmse = 0 es absurdo si σ del ruido es positivo; siempre hay piso.

Preguntas frecuentes

- ¿Más datos o modelo más complejo? Mirá el gap. Gap grande con train bajo overfit más datos o regularización ayudan. Ga
- ¿Cuántos puntos uso para train_sizes? Por defecto np.linspace(0.1, 1.0, 10) está bien. Para datasets grandes, usá menos
- ¿Por qué el train_error sube al aumentar m? Con pocos datos el modelo memoriza (train_error ≈ 0); al sumar filas le cues
- ¿Sirve para clasificación? Sí. Usá scoring='accuracy' o 'neg_log_loss'; la interpretación del gap es análoga.
- ¿Bias-variance se mide en la práctica? Conceptualmente sí (vía bootstrap, como el ejercicio 5), pero en producción

Referencias

- Géron, A. Hands-On Machine Learning, 3ª ed., cap. 4 — "Learning Curves" y cap. 7 — "Bias/Variance Tradeoff".
- scikit-learn user guide: Validation curves: plotting scores to evaluate models.
- API: sklearn.model_selection.learning_curve.

Siguiente clase

Clase 073 — Regularización: Ridge, Lasso, Elastic Net

Apéndice: notebook (primer bloque)

Primera celda ejecutable del notebook de la clase.

```
# Imports y configuración inicial
```

Archivos complementarios

- notebook.ipynb