Clase 058 — Optuna y HPO bayesiano dedicado

Parte: 1 — Machine Learning Clásico · Fuente: Akiba et al. (2019) + Optuna docs. Duración estimada: 80 min.

Clase 058 — Optuna y HPO bayesiano dedicado

> *Parte: 1 — Machine Learning Clásico · Fuente: Akiba et al. (2019) + Optuna docs. Duración estimada: 80 min.*

Objetivo

Profundizar en Optuna —el framework de hyperparameter optimization estándar industrial 2026— aplicado a ML clásico (sklearn, XGBoost, LightGBM, CatBoost). Pasar de Grid/Random Search (clase 052) a TPE (Tree-structured Parzen Estimator) + Hyperband Pruner + persistencia con SQLite. Aprender a interpretar plot_optimization_history, plot_param_importances y plot_slice para entender qué hiperparámetros mueven la aguja.

Resultados de aprendizaje

Al finalizar, el estudiante podrá:

- Definir un objective(trial) con suggest_int, suggest_float, suggest_categorical, suggest_float('lr', 1e-5, 1e-1, log=True).
- Aplicar TPE (default) vs CmaEs vs NSGAIISampler (multi-objective).
- Aplicar pruners (MedianPruner, HyperbandPruner) para cortar trials malos temprano.
- Persistir el study con storage='sqlite:///study.db' y resumir trials.
- Visualizar e interpretar los 5 plots de Optuna.
- Comparar costo total: Grid Search (1000 trials) vs Optuna (100 trials) llegan a mismo accuracy.

Temas

- TPE: modela $P(x \mid y < \gamma)$ y $P(x \mid y \geq \gamma)$ con KDE; samplea de la primera.
- Pruning: callback que reporta progreso intermedio; si va mal vs históricos, kill.
- Multi-objective: optimizar accuracy AND latencia.
- Distributed: varios workers contra el mismo SQLite/PostgreSQL.
- Integration con sklearn, XGBoost, LightGBM, CatBoost.

Definiciones y características

- Trial: una corrida del objective.
- Study: contenedor de trials; se persiste opcional.
- TPE: Tree-structured Parzen Estimator. Modelo bayesiano con KDE bi-modal.
- HyperbandPruner: combina successive halving con varios brackets.
- storage: backend de persistencia. SQLite default; Postgres para distributed.
- plot_param_importances: importancia fANOVA — cuánto contribuye cada hiperparámetro a la varianza del objetivo.

Dataset / recursos

- sklearn.datasets.fetch_california_housing (regresión) o fetch_openml('credit-g') (clasificación).

- Librerías: optuna, optuna-integration, scikit-learn, xgboost, lightgbm.

Ejercicios

1. Objective básico: tunear LogisticRegression(C, penalty) y RandomForest(n_estimators, max_depth) con TPE. 50 trials.
2. Search space compuesto: hiperparámetros condicionales (e.g., solver=liblinear solo permite ciertos penalty). Optuna lo maneja con if.
3. Pruning en XGBoost: usar XGBoostPruningCallback que reporta validación por boosting round → mata trials malos.
4. Persistencia: optuna.create_study(study_name='exp1', storage='sqlite:///opt.db', load_if_exists=True). Re-correr y agregar trials.
5. Multi-objective: maximizar accuracy AND minimizar inference time; obtener Pareto front con optuna.create_study(directions=['maximize', 'minimize']).

Homework verificable

HPO con Optuna sobre XGBoost en credit-g:

1. Espacio: n_estimators, max_depth, lr, subsample, colsample_bytree, reg_alpha, reg_lambda.
2. 100 trials con TPE + Hyperband pruning.
3. Reportar best_params, best_value, plot history + importances.
4. Comparar contra RandomizedSearchCV(50 trials).

Criterio de aceptación: Optuna ≥ RandomizedSearch en AUC; plot_param_importances identifica lr y/o max_depth como las más importantes.

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
LR muestreado uniforme	Fix: suggest_float('lr', 1e-5, 1e-1, log=T
TPE no parece "inteligente" en < 20 trials	El modelo interno necesita warmup. Fix: ≥
study.optimize(n_jobs=4) se cuelga	Race conditions con SQLite. Fix: usar Post
Pruner muy agresivo mata trials buenos	Fix: ajustar MedianPruner(n_startup_trials
Olvido reportar valor intermedio para prun	El pruner no funciona sin trial.report(val

Preguntas frecuentes

¿TPE o CmaEs?

TPE para search spaces mezclados (cat + cont). CmaEs para puramente continuo, con poca cardinalidad — converge más rápido.

¿Cuántos trials?

Para ML clásico, 50-200 alcanza. Para DL caro (cada trial 1 h), 20-50 con pruning agresivo.

¿Distributed HPO?

Postgres storage + varios workers (Python processes en distintas máquinas). Optuna lo soporta nativo.

¿Pruning siempre?

Solo si el objective expone progreso intermedio (cada época en NN, cada boosting round en XGBoost). Para fit().score() directo, no aplica.

¿Visualizaciones para reportar al cliente?

plot_param_importances y plot_slice son las más comunicables. Muestran "qué cambió y cuánto".

Referencias

- Akiba et al. (2019), Optuna, KDD.
- Optuna docs — tutorials y examples.
- Bergstra, Bardenet, Bengio & Kégl (2011), Algorithms for Hyper-Parameter Optimization, NeurIPS — paper original TPE.

Siguiente clase

Clase 059 — Launch, monitoreo y mantenimiento de modelos

Apéndice: notebook (primer bloque)

TPE sampler + MedianPruner sobre GradientBoostingClassifier. Comparamos vs GridSearchCV con presupuesto similar. Instalar: pip install optuna.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import StratifiedKFold, GridSearchCV, cross_val_score
from sklearn.metrics import roc_auc_score
import time

try:
    import optuna
    optuna.logging.set_verbosity(optuna.logging.WARNING)
    OPTUNA_OK = True
except ImportError:
    print('optuna no instalado: pip install optuna')
    OPTUNA_OK = False

np.random.seed(42)
```

Archivos complementarios

- notebook.ipynb