Optimización de Hiperparámetros con Optuna

Miguel Gonzalo Guevara Mamani
Melissa Jessica Macedo Ramos

February 12, 2025

GitHub: `https://github.com/Melissa984/MO.git`

1 Introducción a Optuna: Optimización de Hiperparámetros

Optuna es una poderosa biblioteca de optimización de hiperparámetros para machine learning, que facilita la búsqueda eficiente de los mejores valores para los parámetros de un modelo. El objetivo de esta práctica es mostrar cómo utilizar Optuna para optimizar un modelo de clasificación aplicado al dataset Titanic de Kaggle. Optuna se basa en un enfoque de optimización basado en muestreo que se adapta dinámicamente según los resultados de las evaluaciones anteriores.

Esta herramienta permite encontrar combinaciones de parámetros que maximizan el rendimiento del modelo de manera mucho más eficiente que las búsquedas exhaustivas tradicionales.

1.1 Tabla de Contenidos

La tabla de contenidos está diseñada para ayudar al lector a navegar por el informe. Las secciones incluyen desde la instalación de la librería hasta la visualización de los resultados de la optimización. Este flujo facilita que el lector entienda tanto los aspectos teóricos como los prácticos de la optimización de hiperparámetros.

1. Instalación de Optuna: Instrucciones para instalar la librería y verificar su versión.
2. Optimización de Hiperparámetros: El proceso detallado de optimización del modelo, desde la definición de la función objetivo hasta la optimización con condicionales.
   (a) Definición del Modelo y Función Objetivo: Cómo establecer un modelo base con un conjunto de parámetros iniciales.
   (b) Búsqueda Básica de Hiperparámetros: Cómo usar Optuna para explorar diferentes configuraciones de los hiperparámetros.
   (c) Optimización Avanzada con Condicionales: Mejora de la optimización introduciendo diferentes modelos y parámetros condicionados.
3. Visualización de Resultados: Herramientas para analizar los resultados de la optimización.
   (a) Historial de Optimización: Gráficas sobre el progreso de la optimización a lo largo de las iteraciones.
   (b) Análisis de Hiperparámetros: Gráficas y análisis sobre qué hiperparámetros tuvieron mayor impacto en el rendimiento del modelo.
4. Conclusión: Resumen de los resultados y posibles aplicaciones futuras.

2 1. Instalación de Optuna

Optuna es una librería fácil de instalar que se puede integrar en cualquier entorno de desarrollo basado en Python. Para instalarla, simplemente utilizamos 'pip', el gestor de paquetes de Python. Esta es una instalación estándar que permite obtener la versión más reciente de Optuna.

```
!pip install --quiet optuna
```

Una vez instalada, se debe verificar que la instalación haya sido exitosa. A continuación, se muestra cómo obtener la versión de Optuna para confirmar que está correctamente instalada.

```
import optuna
print("Versi n␣de␣Optuna:", optuna.__version__)
```

3 2. Optimización de Hiperparámetros

En esta sección se presenta el flujo de trabajo para optimizar los hiperparámetros de un modelo utilizando Optuna.

3.1 2.1 Definición del Modelo y Función Objetivo

El primer paso en el proceso de optimización es la definición de la función objetivo. En este caso, usaremos el dataset Titanic de Kaggle y un modelo Random Forest para clasificar si un pasajero sobrevivió o no.

La función objetivo debe ser tal que reciba un objeto 'trial' de Optuna, el cual es responsable de sugerir diferentes valores de los hiperparámetros. En nuestro caso, los hiperparámetros son el número de estimadores ('n$_e$*stimators*')*ylaprofundidadmáximadelárbol*('*max*$_d$

```
import pandas as pd
import sklearn.ensemble
import sklearn.model_selection

# Cargar el dataset Titanic de Kaggle
data = pd.read_csv("train.csv")

# Preprocesamiento b sico de datos
X = data[['Pclass', 'Age', 'SibSp', 'Parch', 'Fare']]  #
    Caracter sticas
y = data['Survived']  # Etiqueta

# Manejar valores nulos de las caracter sticas
X['Age'].fillna(X['Age'].mean(), inplace=True)
X['Fare'].fillna(X['Fare'].mean(), inplace=True)

# Funci n objetivo
def objective(trial):
    n_estimators = trial.suggest_int("n_estimators", 10, 100)
    max_depth = trial.suggest_int("max_depth", 1, 32)

    clf = sklearn.ensemble.RandomForestClassifier(
        n_estimators=n_estimators,
        max_depth=max_depth
    )

    return sklearn.model_selection.cross_val_score(
        clf, X, y, n_jobs=-1, cv=3
    ).mean()

study = optuna.create_study(direction="maximize")
study.optimize(objective, n_trials=50)

print("Mejor precisi n:", study.best_value)
print("Mejores hiperpar metros:", study.best_params)
```

3.2 2.2 Búsqueda Básica de Hiperparámetros

En esta parte, se profundiza en cómo seleccionar los hiperparámetros adecuados y cómo el uso del objeto 'trial' facilita la exploración de diferentes valores.

```
def objective(trial):
    X = data[['Pclass', 'Age', 'SibSp', 'Parch', 'Fare']]
    y = data['Survived']

    X['Age'].fillna(X['Age'].mean(), inplace=True)
    X['Fare'].fillna(X['Fare'].mean(), inplace=True)

    n_estimators = trial.suggest_int("n_estimators", 2, 20)
    max_depth = trial.suggest_int("max_depth", 1, 32)

    clf = sklearn.ensemble.RandomForestClassifier(
        n_estimators=n_estimators,
        max_depth=max_depth
    )

    return sklearn.model_selection.cross_val_score(
        clf, X, y, n_jobs=-1, cv=3
    ).mean()

study = optuna.create_study(direction="maximize")
study.optimize(objective, n_trials=50)

print("Mejor precisi n:", study.best_value)
print("Mejores hiperpar metros:", study.best_params)
```

3.3 2.3 Optimización Avanzada con Condicionales

En esta sección, se introduce un enfoque más avanzado donde se pueden seleccionar diferentes tipos de clasificadores, como Random Forest o Support Vector Machine (SVM), dependiendo de la configuración. Esto añade complejidad a la optimización, permitiendo explorar combinaciones de modelos y parámetros.

```
def objective(trial):
    X = data[['Pclass', 'Age', 'SibSp', 'Parch', 'Fare']]
    y = data['Survived']

    X['Age'].fillna(X['Age'].mean(), inplace=True)
    X['Fare'].fillna(X['Fare'].mean(), inplace=True)

    classifier = trial.suggest_categorical("clasificador", ["
        RandomForest", "SVM"])

    if classifier == "RandomForest":
        n_estimators = trial.suggest_int("n_estimators", 10, 100)
        max_depth = trial.suggest_int("max_depth", 2, 32)
        clf = sklearn.ensemble.RandomForestClassifier(
            n_estimators=n_estimators,
            max_depth=max_depth
        )
    else:
        C = trial.suggest_float("svm_c", 1e-5, 1e5, log=True)
        clf = sklearn.svm.SVC(C=C, gamma="auto")

    return sklearn.model_selection.cross_val_score(
        clf, X, y, cv=3, n_jobs=-1
    ).mean()

study = optuna.create_study(direction="maximize")
study.optimize(objective, n_trials=100)
```

4 3. Visualización de Resultados

La visualización es una parte fundamental del proceso de optimización, ya que permite analizar el rendimiento y la influencia de los hiperparámetros.

4.1 3.1 Historial de Optimización

Optuna proporciona herramientas para visualizar cómo ha evolucionado la optimización a lo largo de las pruebas. El siguiente bloque de código muestra cómo generar una gráfica del historial de optimización.

```
optuna.visualization.plot_optimization_history(study)
```

4.2 3.2 Análisis de Hiperparámetros

- **Relación entre parámetros**: La siguiente visualización muestra cómo varían los hiperparámetros en función del rendimiento obtenido.

```
optuna.visualization.plot_parallel_coordinate(study)
```

- **Importancia de Hiperparámetros**: Muestra qué tan relevantes son los diferentes hiperparámetros para mejorar el modelo.

```
optuna.visualization.plot_param_importances(study)
```

5 4. Conclusión

La optimización de hiperparámetros con Optuna permite mejorar significativamente el rendimiento de los modelos al encontrar la configuración ideal de parámetros. Este enfoque automatizado y eficiente es una herramienta poderosa para proyectos de machine learning y puede ser aplicado a una variedad de modelos y datasets. El uso de condiciones en los parámetros también agrega flexibilidad y permite optimizar más de un modelo de manera simultánea.