



OPTUNA

Optuna:
A hyperparameter
optimization
framework



Contenido



- Introducción
- Optuna
- Espacios de búsqueda, algoritmos de muestreo y pruners
- Visualización para el análisis de optimización
- Ejemplo en Tensorflow

Introducción



Los *hiperparámetros* determinan el comportamiento y desempeño de los algoritmos de ML.

Suelen establecerse de forma manual antes del proceso de entrenamiento.

Algunos ejemplos son el *learning rate*, *número de capas*, *número de unidades*, etc.

Introducción



El problema de la búsqueda de hiperparámetros es el proceso de seleccionar un conjunto de parámetros óptimos para un algoritmo de aprendizaje.

Es un problema desafiante por varias razones:

- Espacio de búsqueda grande
- Evaluaciones costosas
- Interacciones complejas
- Falta de gradientes

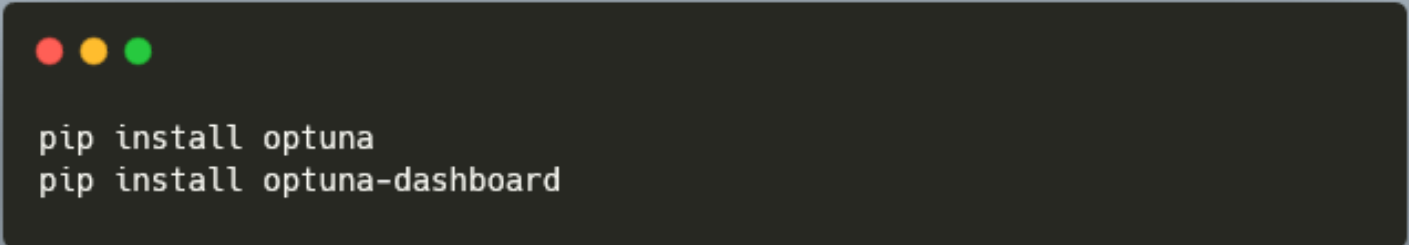
¿Qué es Optuna?

- Optuna es una biblioteca de Python diseñada para optimizar los hiperparámetros de modelos de machine learning.
- El usuario puede construir dinámicamente los espacios de búsqueda para los hiperparámetros.
- Optuna reduce tiempo y recursos necesarios para encontrar los hiperparámetros óptimos.

¿Y cómo funciona?

1. Definimos el espacio de búsqueda: hay que especificar los parámetros que deseamos optimizar, así como el rango de valores que pueden tomar.
2. Creamos una función objetivo: esta función toma un conjunto de hiperparámetros como argumento y devuelve el valor que queremos minimizar (o maximizar).
3. Optimización: Optuna realiza una búsqueda eficiente en el espacio de hiperparámetros, probando diferentes combinaciones para encontrar la que maximiza o minimiza la función objetivo
4. Resultados: Una vez finalizada la optimización, podemos obtener los mejores hiperparámetros encontrados y utilizarlos para entrenar el modelo final.

Instalación

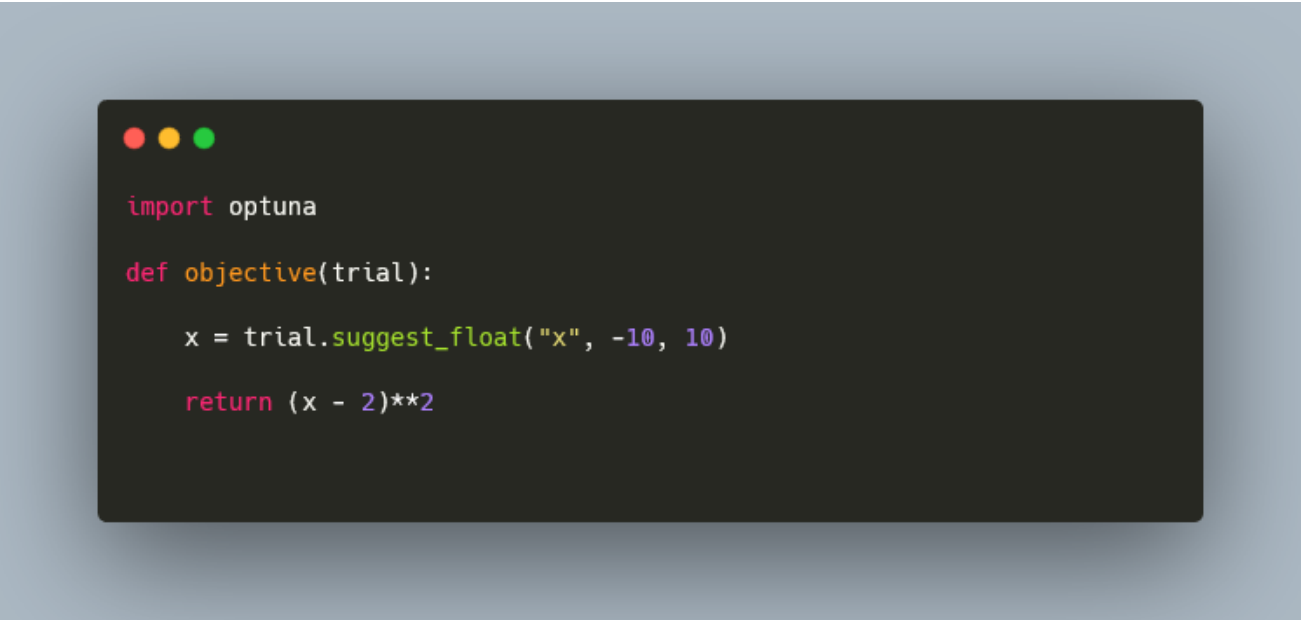
A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. It contains two lines of text: 'pip install optuna' and 'pip install optuna-dashboard'.

```
pip install optuna  
pip install optuna-dashboard
```

Ejemplo:

Supongamos que deseamos optimizar: $(x - 2)^2$

En Optuna, las funciones a ser optimizadas llevan el nombre `objective`

A code editor window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. It contains Python code for an Optuna objective function.

```
import optuna

def objective(trial):
    x = trial.suggest_float("x", -10, 10)
    return (x - 2)**2
```


Ejemplo:

Durante la optimización, Optuna llama y evalúa repetidamente a la función objetivo con distintos valores de x .

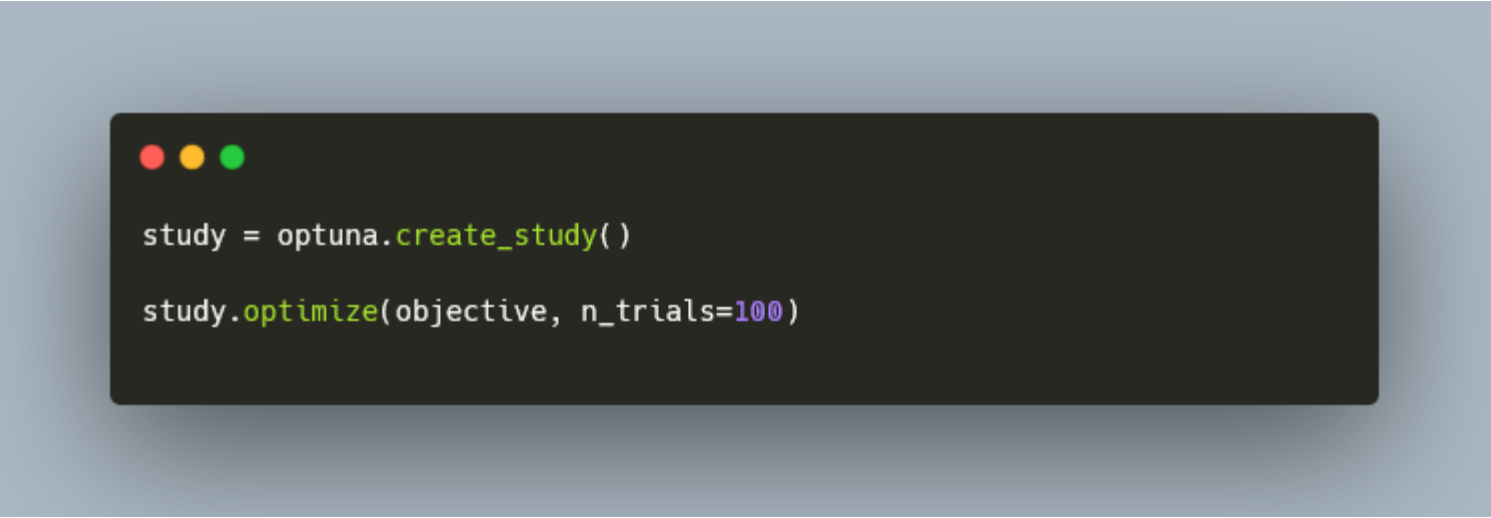
Un objeto `trial` corresponde a una sola ejecución de la función objetivo y es inicializado internamente sobre cada invocación de la función.

Los `suggest` APIs (por ejemplo, `suggest_float()`) son llamados dentro de la función objetivo para obtener parámetros para un trial.

`suggest_float()` selecciona los parámetros uniformemente dentro del rango establecido.

Ejemplo:

Para comenzar la optimización, creamos un objeto `study` y pasamos la función objetivo al método `optimize()` como sigue:

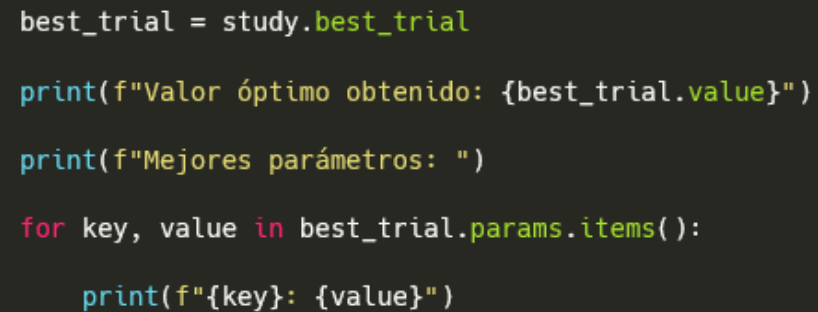
A code snippet is displayed within a dark-themed rectangular box that has a subtle drop shadow. The box is set against a light blue-grey background. At the top left of the box are three small colored circles (red, yellow, green) resembling a window's title bar. The code is written in a monospaced font with syntax highlighting: `study = optuna.create_study()` and `study.optimize(objective, n_trials=100)`.

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

```
[I 2023-09-14 23:04:58,854] A new study created in memory with name: no-name-f7c4144d-0e88-40de-a358-96ca73ae3fb0
[I 2023-09-14 23:04:58,860] Trial 0 finished with value: 137.86515493249345 and parameters: {'x': -9.741599334523958}. Best is trial 0 with value: 137.86515493249345.
[I 2023-09-14 23:04:58,864] Trial 1 finished with value: 2.602838966831088 and parameters: {'x': 3.613331635724995}. Best is trial 1 with value: 2.602838966831088.
[I 2023-09-14 23:04:58,868] Trial 2 finished with value: 58.276724034542084 and parameters: {'x': 9.633919310193296}. Best is trial 1 with value: 2.602838966831088.
[I 2023-09-14 23:04:58,868] Trial 3 finished with value: 5.686159312413056 and parameters: {'x': 4.384566902482096}. Best is trial 1 with value: 2.602838966831088.
[I 2023-09-14 23:04:58,872] Trial 4 finished with value: 10.248457509667325 and parameters: {'x': -1.2013212131348716}. Best is trial 1 with value: 2.602838966831088.
[I 2023-09-14 23:04:58,876] Trial 5 finished with value: 25.482527683042832 and parameters: {'x': -3.048022155561803}. Best is trial 1 with value: 2.602838966831088.
[I 2023-09-14 23:04:58,877] Trial 6 finished with value: 27.98023092203777 and parameters: {'x': -3.289634290008882}. Best is trial 1 with value: 2.602838966831088.
[I 2023-09-14 23:04:58,880] Trial 7 finished with value: 58.89376981064494 and parameters: {'x': 9.674227636097651}. Best is trial 1 with value: 2.602838966831088.
[I 2023-09-14 23:04:58,880] Trial 8 finished with value: 5.316657494558098 and parameters: {'x': -0.30578782513875247}. Best is trial 1 with value: 2.602838966831088.
[I 2023-09-14 23:04:58,887] Trial 9 finished with value: 0.11553563912045624 and parameters: {'x': 1.660094661529925}. Best is trial 9 with value: 0.11553563912045624.
[I 2023-09-14 23:04:58,908] Trial 10 finished with value: 4.475643720758226 and parameters: {'x': 4.115571724323765}. Best is trial 9 with value: 0.11553563912045624.
[I 2023-09-14 23:04:58,923] Trial 11 finished with value: 1.199208972906664 and parameters: {'x': 3.095084002671331}. Best is trial 9 with value: 0.11553563912045624.
[I 2023-09-14 23:04:58,940] Trial 12 finished with value: 0.06124329651003651 and parameters: {'x': 2.2474738299498283}. Best is trial 12 with value: 0.06124329651003651.
[I 2023-09-14 23:04:58,952] Trial 13 finished with value: 0.18862495661259324 and parameters: {'x': 1.56569025268526}. Best is trial 12 with value: 0.06124329651003651.
[I 2023-09-14 23:04:58,972] Trial 14 finished with value: 20.746209747407022 and parameters: {'x': 6.554800736300878}. Best is trial 12 with value: 0.06124329651003651.
[I 2023-09-14 23:04:58,988] Trial 15 finished with value: 1.2115518448645972 and parameters: {'x': 0.8992948419923721}. Best is trial 12 with value: 0.06124329651003651.
[I 2023-09-14 23:04:59,004] Trial 16 finished with value: 11.967625102902343 and parameters: {'x': -1.4594255452173477}. Best is trial 12 with value: 0.06124329651003651.
[I 2023-09-14 23:04:59,016] Trial 17 finished with value: 19.224101323485236 and parameters: {'x': 6.384529772220191}. Best is trial 12 with value: 0.06124329651003651.
[I 2023-09-14 23:04:59,032] Trial 18 finished with value: 0.08175004878045616 and parameters: {'x': 1.714080345585589}. Best is trial 12 with value: 0.06124329651003651.
[I 2023-09-14 23:04:59,040] Trial 19 finished with value: 41.30494617406795 and parameters: {'x': -4.426892419674376}. Best is trial 12 with value: 0.06124329651003651.
[I 2023-09-14 23:04:59,052] Trial 20 finished with value: 2.9864937934907516 and parameters: {'x': 0.2718524966048901}. Best is trial 12 with value: 0.06124329651003651.
[I 2023-09-14 23:04:59,068] Trial 21 finished with value: 5.69012855394554e-05 and parameters: {'x': 2.0075432940774873}. Best is trial 21 with value: 5.69012855394554e-05.
[I 2023-09-14 23:04:59,078] Trial 22 finished with value: 0.4503377240813476 and parameters: {'x': 2.671072070705783}. Best is trial 21 with value: 5.69012855394554e-05.
[I 2023-09-14 23:04:59,092] Trial 23 finished with value: 12.898627880407084 and parameters: {'x': 5.591465979291337}. Best is trial 21 with value: 5.69012855394554e-05.
...
[I 2023-09-14 23:05:00,136] Trial 96 finished with value: 1.350981759438984 and parameters: {'x': 0.8376825909249299}. Best is trial 24 with value: 4.085198291744798e-07.
[I 2023-09-14 23:05:00,152] Trial 97 finished with value: 0.022162928982228026 and parameters: {'x': 1.8511278099098827}. Best is trial 24 with value: 4.085198291744798e-07.
[I 2023-09-14 23:05:00,164] Trial 98 finished with value: 0.7321793671400878 and parameters: {'x': 2.8556748022117326}. Best is trial 24 with value: 4.085198291744798e-07.
[I 2023-09-14 23:05:00,178] Trial 99 finished with value: 2.611849405965285 and parameters: {'x': 0.3838782824411755}. Best is trial 24 with value: 4.085198291744798e-07.
```

Ejemplo:

Podemos obtener los mejores parámetros de la siguiente forma:



```
best_trial = study.best_trial

print(f"Valor óptimo obtenido: {best_trial.value}")

print(f"Mejores parámetros: ")

for key, value in best_trial.params.items():
    print(f"{key}: {value}")
```

Ejemplo:

Una característica notable de Optuna es que si ejecutamos nuevamente `optimize()`, podemos continuar con la optimización.

Espacios de búsqueda

Muestreo de hiperparámetros

Podemos utilizar las siguientes funciones para el muestreo:

- `optuna.trial.suggest_categorical()`
- `optuna.trial.suggest_int()`
- `optuna.trial.suggest_float()`

Con argumentos opcionales de `step` y `log`, podemos discretizar o calcular el logaritmo de los enteros y flotantes

Espacios de búsqueda

```
def objective(trial):  
    # Parámetro categórico  
    optimizer = trial.suggest_categorical("optimizer", ["SGD",  
"Adam"])  
  
    # Parámetro entero  
    num_layers = trial.suggest_int("num_layers", 1, 3)  
  
    # Parámetro entero (con log)  
    num_channels = trial.suggest_int("num_channels", 32, 512, log =  
True)  
  
    # Parámetro entero (discretizado)  
    num_units = trial.suggest_int("num_units", 10, 100, step = 5)  
  
    # Parámetro de punto flotante  
    dropout_rate = trial.suggest_float("dropout_rate", 0.0, 1.0)  
  
    # Parámetro de punto flotante (log)  
    learning_rate = trial.suggest_float("learning_rate", 1e-5, 1e-2,  
log = True)  
  
    # Parámetro de punto flotante (discretizado)  
    drop_path_rate = trial.suggest_float("drop_path_rate", 0.0, 1.0,  
step = 0.1)
```

Algoritmos de muestreo

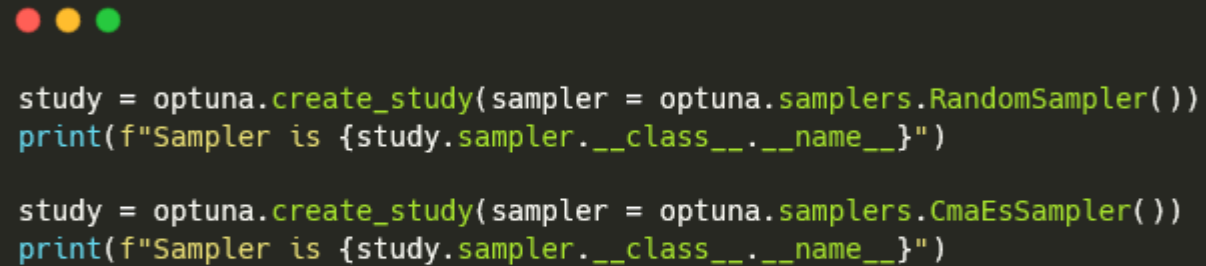
Algoritmos de optimización eficientes

Los samplers reducen continuamente el espacio de búsqueda, utilizando los registros de los valores de los parámetros sugeridos y los valores objetivo evaluados.

Optuna provee los siguientes algoritmos de muestreo:

- Grid Search se implementa en `GridSampler`
- Random Search se implementa en `RandomSampler`
- Tree-Structured Parzen Estimator Algorithm en `TPESampler` (por defecto)
- CMA-ES Based Algorithm en `CmaEsSampler`
- Non-Dominated Sorting Genetic Algorithm II en `NSGAIISampler`
- Quasi Monte-Carlo Sampling Algorithm en `QMCSampler`

Algoritmos de muestreo



```
study = optuna.create_study(sampler = optuna.samplers.RandomSampler())
print(f"Sampler is {study.sampler.__class__.__name__}")

study = optuna.create_study(sampler = optuna.samplers.CmaEsSampler())
print(f"Sampler is {study.sampler.__class__.__name__}")
```

Algoritmos de pruning

En Optuna, pruning se refiere al proceso de detener prematuramente los ensayos de optimización poco prometedores.

Los pruners son los objetos que implementan este proceso de “poda” de ensayos, lo que permite que el proceso de optimización se centre más en las regiones prometedoras en el espacio de búsqueda.

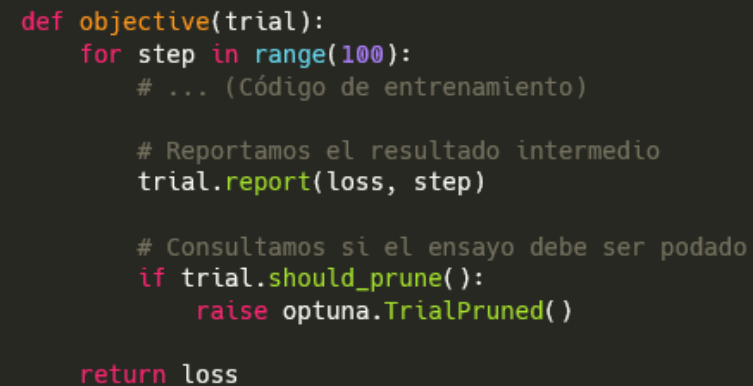
También se conoce como early-stopping.

Algoritmos de pruning

Optuna ofrece los siguientes algoritmos:

- **MedianPruner:** Detiene el ensayo en una etapa específica si el valor de la métrica está por debajo de la mediana de los valores de los ensayos anteriores.
- **SuccessiveHalvingPruner:** Reduce a la mitad el número de ensayos en consideración en cada etapa de la optimización.
- **HyperbandPruner:** Es una extensión del Successive Halving Pruner, busca adaptarse a los recursos disponibles para la optimización.

Implementación



```
def objective(trial):  
    for step in range(100):  
        # ... (Código de entrenamiento)  
  
        # Reportamos el resultado intermedio  
        trial.report(loss, step)  
  
        # Consultamos si el ensayo debe ser podado  
        if trial.should_prune():  
            raise optuna.TrialPruned()  
  
    return loss
```

Samplers y Pruners

¿Qué sampler y pruner usar?

Para tareas distintas a Deep Learning, se recomienda:

- Para RandomSampler, MedianPruner es el mejor.
- Para TPESampler, HyperbandPruner es el mejor.

Para tareas de Deep Learning:

Recurso de cómputo en paralelo	Hiperparámetros Categóricos/Condicionales	Algoritmos recomendados
Limitado	No	TPE, GP-EI si el espacio de búsqueda es de baja dimensión y continuo
	Sí	TPE, GP-EI si el espacio de búsqueda es de baja dimensión y continuo
Suficiente	No	CMA-ES, Random Search
	Sí	Random Search o Genetic Algorithm

Análisis visual de optimización

Optuna tiene disponibles distintas funciones de visualización en el módulo `optuna.visualization` para analizar los resultados de la optimización.



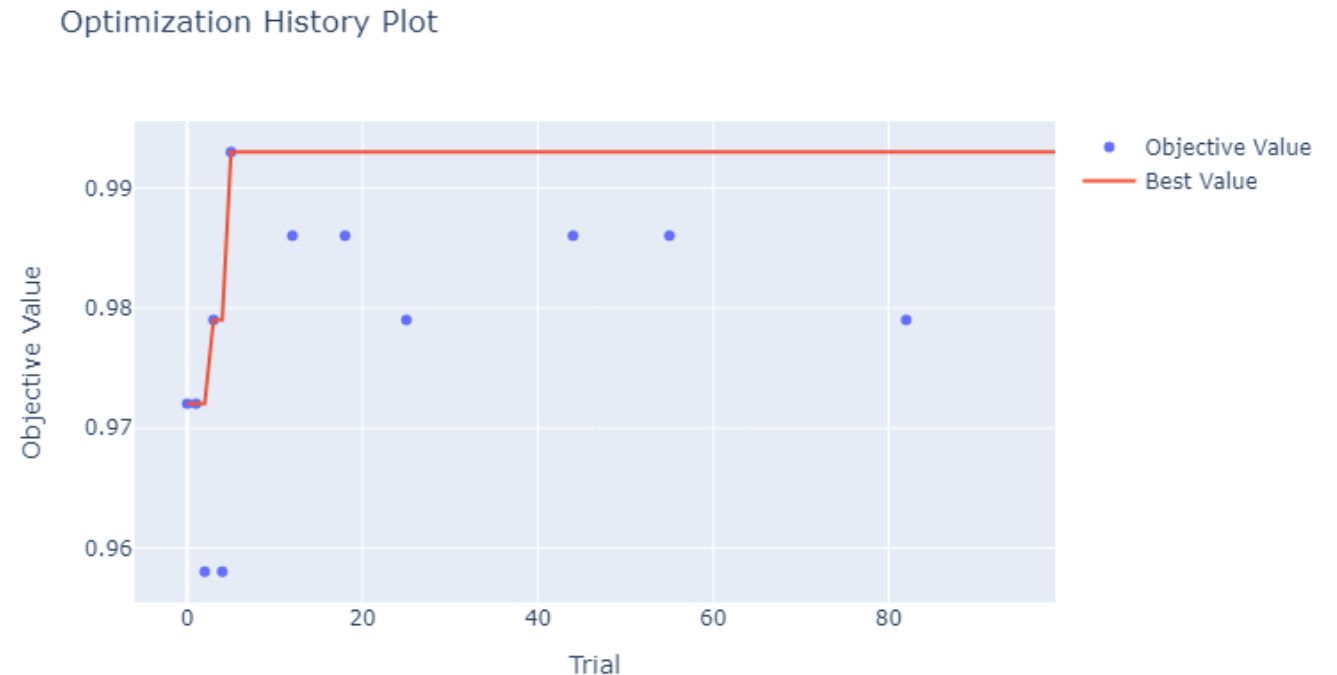
```
# Se puede utilizar Matplotlib en vez de Plotly para visualizar, al
# reemplazar 'optuna.visualization'
# con 'optuna.visualization.matplotlib'

from optuna.visualization import plot_contour
from optuna.visualization import plot_edf
from optuna.visualization import plot_intermediate_values
from optuna.visualization import plot_optimization_history
from optuna.visualization import plot_parallel_coordinate
from optuna.visualization import plot_param_importances
from optuna.visualization import plot_rank
from optuna.visualization import plot_slice
from optuna.visualization import plot_timeline
```

Visualización

`plot_optimization_history(study)`

Genera una gráfica de la historia de la optimización, mostrando el valor de la función objetivo de cada ensayo a lo largo del tiempo.

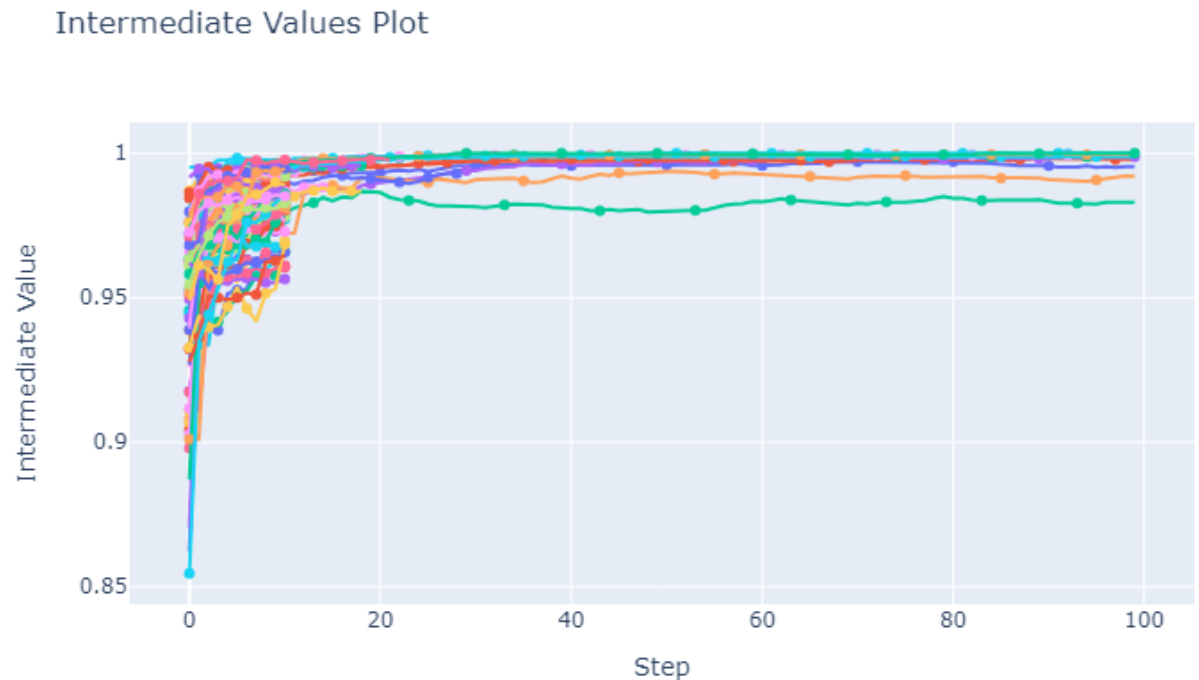


Visualización

`plot_intermediate_values(study)`

Nos genera un gráfico de los valores intermedios de todos los ensayos.

Ayuda a ver cómo los valores de la función objetivo cambian durante el entrenamiento, podría ayudar a identificar si existe convergencia o no.



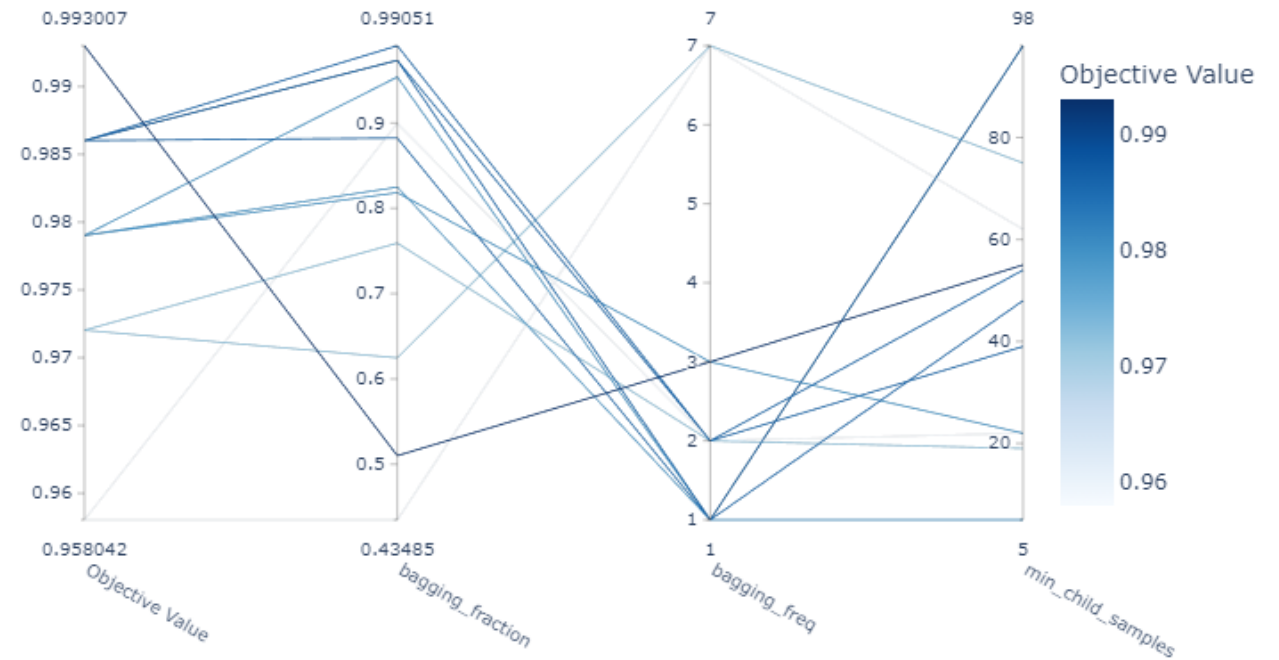
Visualización

`plot_parallel_coordinate(study)`

Genera un gráfico de coordenadas paralelas que permite visualizar las relaciones entre los parámetros y la función objetivo.

Es útil para visualizar cómo cada parámetro afecta la función objetivo y cómo están relacionados los diferentes parámetros entre sí.

Parallel Coordinate Plot

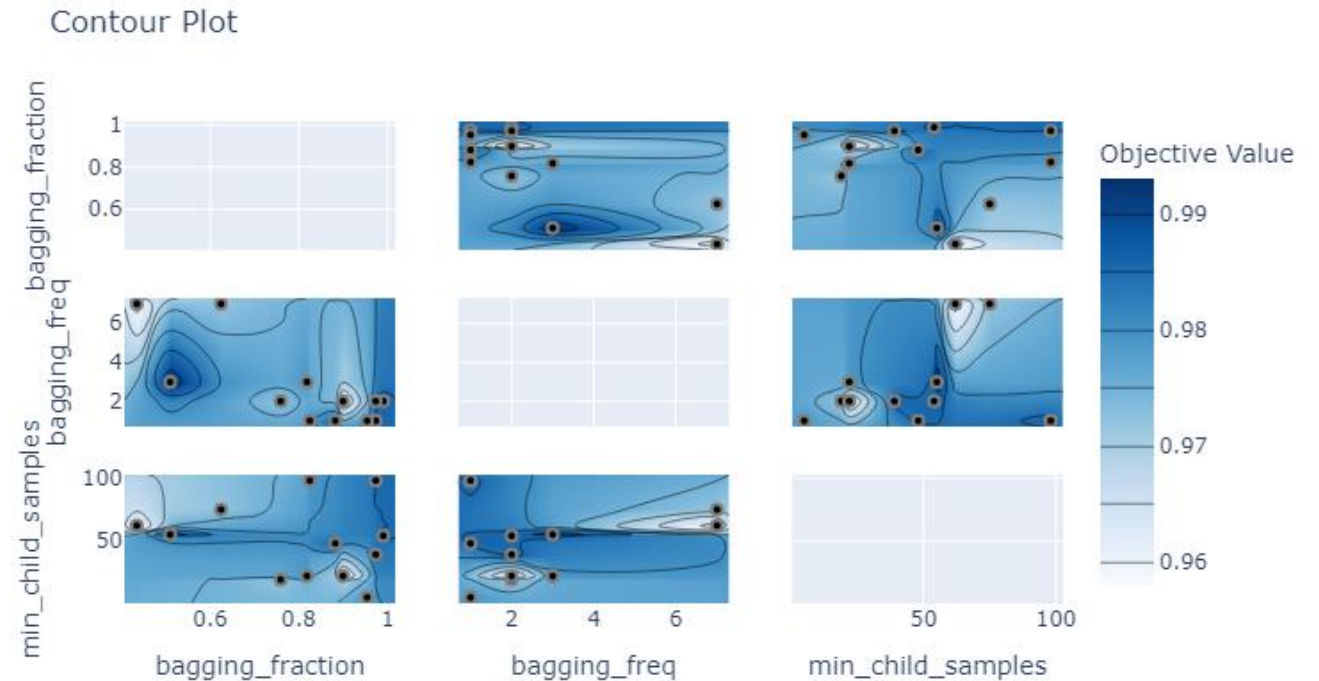


Visualización

plot_contour(study)

Genera una gráfica de contorno que muestra las interacciones entre los pares de parámetros.

Ayuda a visualizar cómo las combinaciones de dos parámetros afectan el resultado de la función objetivo.

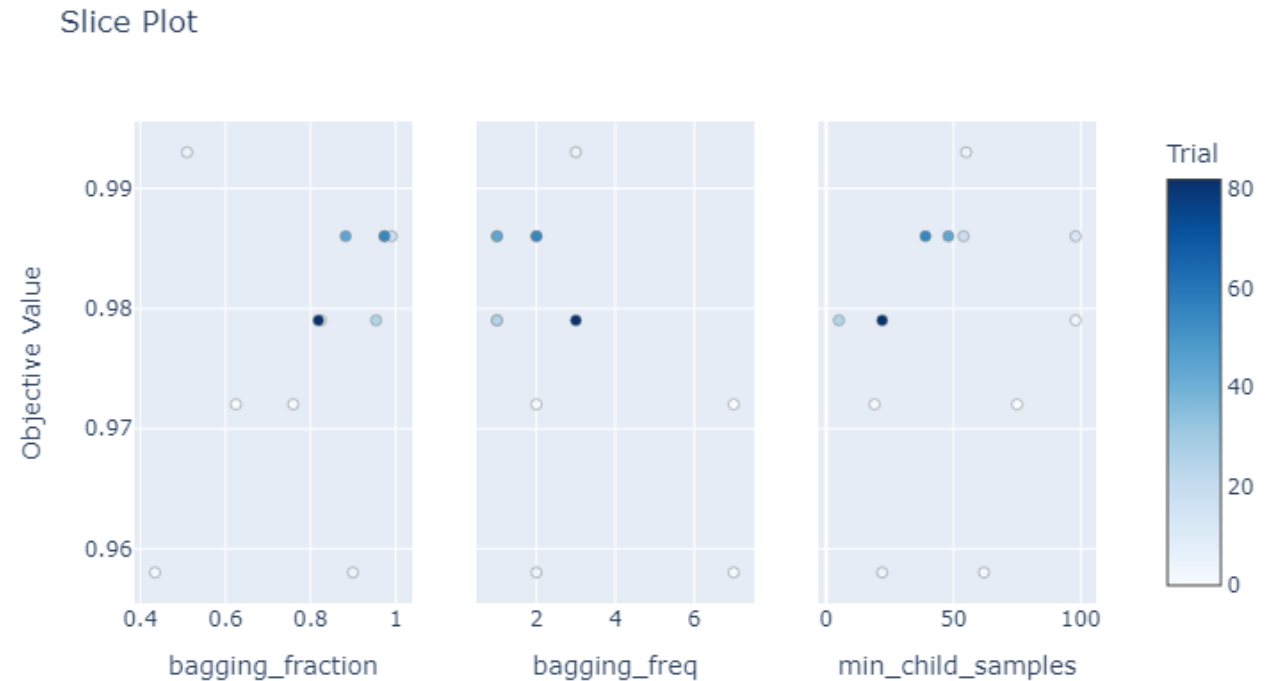


Visualización

`plot_slice(study)`

Genera un gráfico de slice que muestra cómo cada parámetro individual afecta la función objetivo.

Útil para visualizar cómo cada parámetro individual afecta la función objetivo, lo que puede ayudarnos a entender la sensibilidad de la función objetivo a cada parámetro.

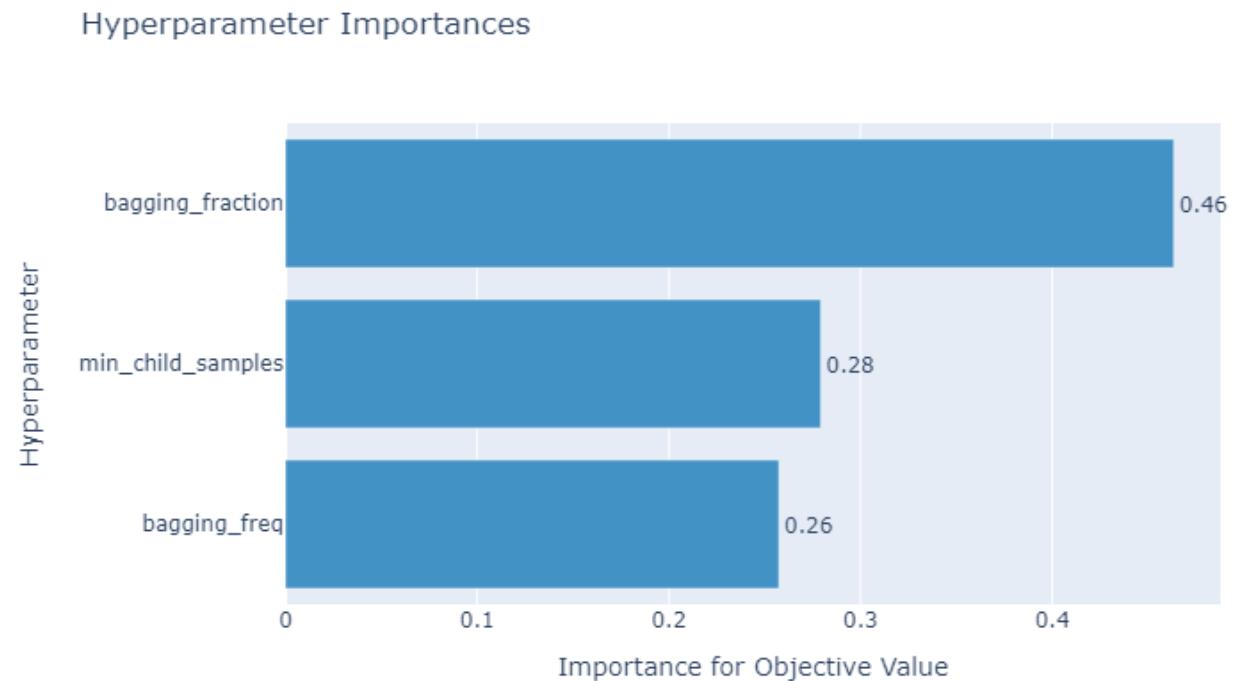


Visualización

`plot_param_importances(study)`

Genera un gráfico de slice que muestra cómo cada parámetro individual afecta la función objetivo.

Útil para visualizar cómo cada parámetro individual afecta la función objetivo, lo que puede ayudarnos a entender la sensibilidad de la función objetivo a cada parámetro.

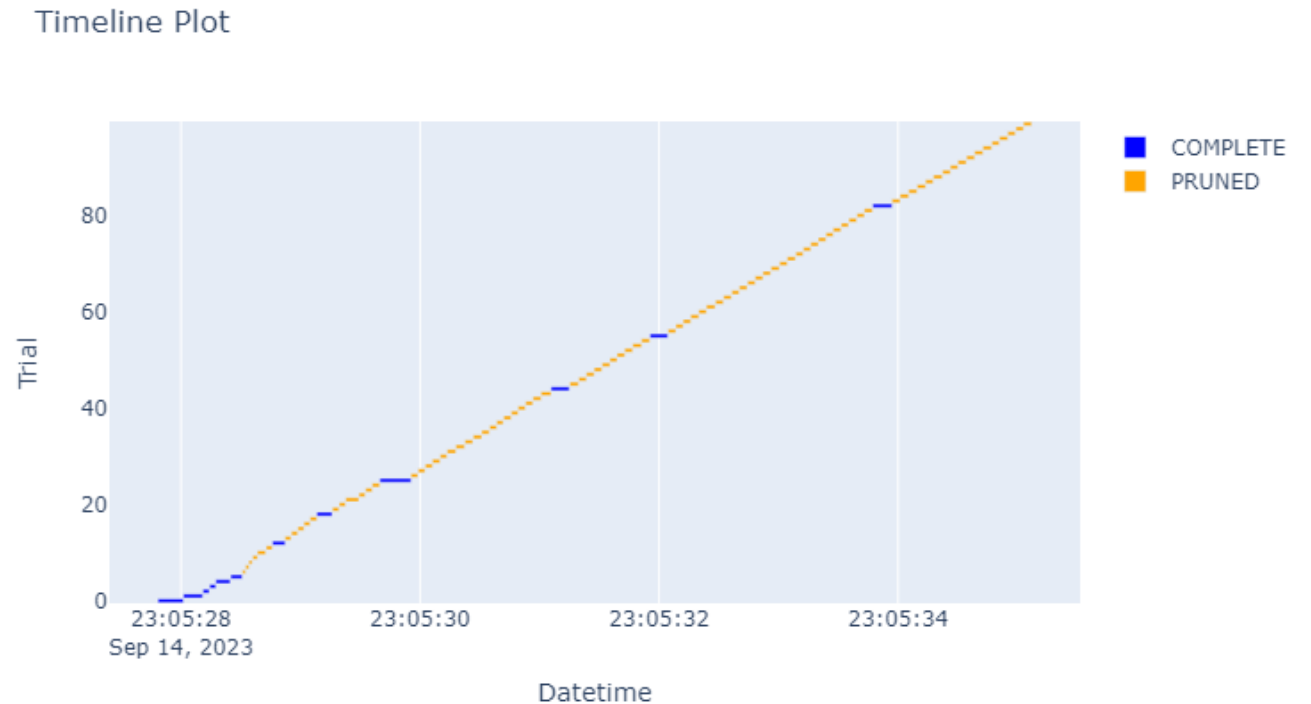


Visualización

```
plot_timeline(study)
```

Genera un gráfico de serie de tiempo que muestra cuándo se realizó cada ensayo a lo largo del tiempo.

Permite visualizar cómo se han distribuido los ensayos a lo largo del tiempo, lo que puede ser útil para evaluar el rendimiento y la eficiencia de la optimización.



Ejemplo en Tensorflow y Keras

Clasificación de dígitos escritos a mano

Para poner en práctica y ver a Optuna en acción, haremos el clásico ejemplo de clasificar imágenes de dígitos escritos a mano.

Utilizamos el dataset MNIST. A continuación, mostramos el código completo:

[Repositorio con el código y la presentación](#)

Conclusiones

- Optuna nos sirve para la optimización automática de hiperparámetros.
- Tiene buena integración con otros ecosistemas de machine learning como TF, Keras o ML Flow.
- Las herramientas de pruning permiten una optimización más rápida y eficiente.
- Las capacidades de visualización nos ayudan a realizar un análisis más profundo sobre el rendimiento del modelo y la importancia de distintos hiperparámetros.

Referencias

- [Optuna: Read the Docs](#)
- [Optuna: Code Examples](#)
- [A Comprehensive Guide on Hyperparameter Tuning and its Techniques](#)
- [Best Tools for Model Tuning and Hyperparameter Optimization](#)

Gracias

