Optuna Documentation

发布 *2.7.0*

Optuna Contributors.

2022 年 05 月 26 日

目录

Optuna 是一个特别为机器学习设计的自动超参数优化软件框架. 它具有命令式的, *define-by-run* 风格的 API. 由于这种 API 的存在, 用 Optuna 编写的代码模块化程度很高, Optuna 的用户因此也可以动态地构造超参数的搜索空间.

CHAPTER 1

主要特点

Optuna 有如下现代化的功能：

- 轻量级、多功能和跨平台架构
  - 只需少量依赖, 简单安装完成后便可处理各种任务.
- *Python* 式的搜索空间
  - 利用熟悉的 Python 语法, 如条件语句和循环来定义搜索空间.
- 高效的优化算法
  - 采用了最先进的超参数采样和最有效的对无望 trial 进行剪枝的算法.
- 易用的并行优化
  - 仅需少量甚至无需代码修改便可将 study 扩展到数十甚至数百个 worker 上.
- 便捷的可视化
  - 查询优化记录.

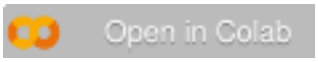
CHAPTER 2

基本概念

我们以如下方式使用 *study* 和 *trial* 这两个术语：

- Study: 基于目标函数的优化过程
- Trial: 目标函数的单次执行过程

请参考下面的示例代码. 一个 *study* 的目的是通过多次 *trial* (例如 `n_trials=100`) 来找出最佳的超参数值集（比如选择 `classifier` 还是 `svm_c`）. 而 Optuna 旨在加速和自动化此类 *study* 优化过程.




```
import ...

# Define an objective function to be minimized.
def objective(trial):

    # Invoke suggest methods of a Trial object to generate hyperparameters.
    regressor_name = trial.suggest_categorical('classifier', ['SVR', 'RandomForest'])
    if regressor_name == 'SVR':
        svr_c = trial.suggest_float('svr_c', 1e-10, 1e10, log=True)
        regressor_obj = sklearn.svm.SVR(C=svr_c)
    else:
        rf_max_depth = trial.suggest_int('rf_max_depth', 2, 32)
        regressor_obj = sklearn.ensemble.RandomForestRegressor(max_depth=rf_max_depth)

    X, y = sklearn.datasets.load_boston(return_X_y=True)
```

(下页继续)

(续上页)

```
    X_train, X_val, y_train, y_val = sklearn.model_selection.train_test_split(X, y,␣
↪random_state=0)

    regressor_obj.fit(X_train, y_train)
    y_pred = regressor_obj.predict(X_val)

    error = sklearn.metrics.mean_squared_error(y_val, y_pred)

    return error  # An objective value linked with the Trial object.

study = optuna.create_study()  # Create a new study.
study.optimize(objective, n_trials=100)  # Invoke optimization of the objective␣
↪function.
```

CHAPTER 3

Communication

- 可在 GitHub Issues 报告 bug、提 feature request 和问问题.
- 可在 Gitter 与开发者互动.
- 可在 StackOverflow 提问.

CHAPTER 4

贡献

欢迎大家对 Optuna 的一切贡献！但是在发送 pull request 时请遵从 contribution guide 的规范.

CHAPTER 5

License

MIT License (see LICENSE).

CHAPTER 6

Reference

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. In KDD (arXiv).

6.1 安装

Optuna 支持 3.6 或者更新版本的 Python.

我们推荐使用 pip 来安装 Optuna：

```
$ pip install optuna
```

你也可以在 Git 仓库的 master 分支下用开发模式安装 Optuna：

```
$ pip install git+https://github.com/optuna/optuna.git
```

你也可以通过 conda 来安装 Optuna：

```
$ conda install -c conda-forge optuna
```

6.2 教程

如果你不熟悉 Optuna 或者需要一个一般的介绍，我们强烈推荐下面这个视频。

6.2.1 关键特性

展现 Optuna 的关键特性

轻量级、多功能和跨平台架构

Optuna 完全由 Python 写成，并且只有少量依赖。这意味着一旦你开始对 Optuna 感兴趣，我们可以快速上手真实的例子。

二次函数的例子

通常，Optuna 是用于优化超参数的，但是作为例子，我们这里来优化一个简单的而此函数：$(x-2)^2$.

首先，导入*optuna*.

```
import optuna
```

在 Optuna 中，待优化函数一般被命名为 *objective*.

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

该函数的返回值是 $(x-2)^2$. 我们的目标是找到一个 `x`，使 `objective` 函数的输出最小。这被称为“optimization”（优化）。在优化过程中，Optuna 反复调用目标函数，在不同的 `x` 下对其进行求值。

一个*Trial* 对应着目标函数的单次执行。在每次调用该函数的时候，它都被内部实例化一次。

而 *suggest* API (例如*suggest_uniform()*) 在目标函数内部被调用。它被用于获取单个 trial 的参数。在上面的例子中，*suggest_uniform()* 在给定的范围（`-10` 到 `10`）内均匀地选择参数。

为了开始优化过程，我们将创建一个 study 对象，并将目标函数传递给它的一个方法*optimize()*：

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

You can get the best parameter as follows.

```
best_params = study.best_params
found_x = best_params["x"]
print("Found x: {}, (x - 2)^2: {}".format(found_x, (found_x - 2) ** 2))
```

Out:

```
Found x: 2.0007453774691357, (x - 2)^2: 5.555875714951739e-07
```

最佳参数可以通过如下方式获得：

备注： 当 Optuna 被用于机器学习时，目标函数通常返回模型的损失或者准确度。

Study 对象

下面是几个常用术语：

- **Trial**: 目标函数的单次调用
- **Study**: 一次优化过程，包含一系列的 trials.
- **Parameter**: 待优化的参数，比如上面例子中的 `x`.

在 Optuna 中，我们用 study 对象来管理优化过程。*create_study()* 方法会返回一个 study 对象。该对象包含若干有用的属性，可以用于分析优化结果。

获得参数名和参数值的字典：

```
study.best_params
```

Out:

```
{'x': 2.0007453774691357}
```

获得最佳目标函数值：

```
study.best_value
```

Out:

```
5.555875714951739e-07
```

获得最佳 trial:

```
study.best_trial
```

Out:

```
FrozenTrial(number=46, values=[5.555875714951739e-07], datetime_start=datetime.
↪datetime(2022, 5, 26, 12, 5, 35, 266572), datetime_complete=datetime.datetime(2022,␣
↪5, 26, 12, 5, 35, 269818), params={'x': 2.0007453774691357}, distributions={'x':␣
↪UniformDistribution(high=10.0, low=-10.0)}, user_attrs={}, system_attrs={},␣
↪intermediate_values={}, trial_id=46, state=TrialState.COMPLETE, value=None)
```

(下页继续)

(续上页)

获得所有 trials:

```
study.trials
```

Out:

```
[FrozenTrial(number=0, values=[0.17511330772143616], datetime_start=datetime.
↪datetime(2022, 5, 26, 12, 5, 35, 127027), datetime_complete=datetime.datetime(2022,␣
↪5, 26, 12, 5, 35, 127432), params={'x': 2.4184654199828657}, distributions={'x':␣
↪UniformDistribution(high=10.0, low=-10.0)}, user_attrs={}, system_attrs={},␣
↪intermediate_values={}, trial_id=0, state=TrialState.COMPLETE, value=None),␣
↪FrozenTrial(number=1, values=[5.55018460646496], datetime_start=datetime.
↪datetime(2022, 5, 26, 12, 5, 35, 127925), datetime_complete=datetime.datetime(2022,␣
↪5, 26, 12, 5, 35, 128255), params={'x': 4.355882978092282}, distributions={'x':␣
↪UniformDistribution(high=10.0, low=-10.0)}, user_attrs={}, system_attrs={},␣
↪intermediate_values={}, trial_id=1, state=TrialState.COMPLETE, value=None),␣
↪FrozenTrial(number=2, values=[8.18262262981584], datetime_start=datetime.
↪datetime(2022, 5, 26, 12, 5, 35, 128683), datetime_complete=datetime.datetime(2022,␣
↪5, 26, 12, 5, 35, 128995), params={'x': -0.8605283829767956}, distributions={'x':␣
↪UniformDistribution(high=10.0, low=-10.0)}, user_attrs={}, system_attrs={},␣
↪intermediate_values={}, trial_id=2, state=TrialState.COMPLETE, value=None),␣
↪FrozenTrial(number=3, values=[13.502562948612052], datetime_start=datetime.
↪datetime(2022, 5, 26, 12, 5, 35, 129426), datetime_complete=datetime.datetime(2022,␣
↪5, 26, 12, 5, 35, 129744), params={'x': -1.6745833707526696}, distributions={'x':␣
↪UniformDistribution(high=10.0, low=-10.0)}, user_attrs={}, system_attrs={},␣
↪intermediate_values={}, trial_id=3, state=TrialState.COMPLETE, value=None),␣
↪FrozenTrial(number=4, values=[4.097234965300769], datetime_start=datetime.
↪datetime(2022, 5, 26, 12, 5, 35, 130166), datetime_complete=datetime.datetime(2022,␣
↪5, 26, 12, 5, 35, 130482), params={'x': 4.024162781324854}, distributions={'x':␣
↪UniformDistribution(high=10.0, low=-10.0)}, user_attrs={}, system_attrs={},␣
↪intermediate_values={}, trial_id=4, state=TrialState.COMPLETE, value=None),␣
↪FrozenTrial(number=5, values=[53.56822845316632], datetime_start=datetime.
↪datetime(2022, 5, 26, 12, 5, 35, 130904), datetime_complete=datetime.datetime(2022,␣
↪5, 26, 12, 5, 35, 131227), params={'x': 9.319031934154019}, distributions={'x':␣
↪UniformDistribution(high=10.0, low=-10.0)}, user_attrs={}, system_attrs={},␣
↪intermediate_values={}, trial_id=5, state=TrialState.COMPLETE, value=None),␣
↪FrozenTrial(number=6, values=[18.267873372441038], datetime_start=datetime.
↪datetime(2022, 5, 26, 12, 5, 35, 131649), datetime_complete=datetime.datetime(2022,␣
↪5, 26, 12, 5, 35, 131980), params={'x': -2.2740932807369845}, distributions={'x':␣
↪UniformDistribution(high=10.0, low=-10.0)}, user_attrs={}, system_attrs={},␣
↪intermediate_values={}, trial_id=6, state=TrialState.COMPLETE, value=None),␣
↪FrozenTrial(number=7, values=[18.646084119202367], datetime_start=datetime.
↪datetime(2022, 5, 26, 12, 5, 35, 132429), datetime_complete=datetime.datetime(2022,␣
↪5, 26, 12, 5, 35, 132750), params={'x': -2.318111174947024}, distributions={'x':␣
↪UniformDistribution(high=10.0, low=-10.0)}, user_attrs={}, system_attrs={},␣
↪intermediate_values={}, trial_id=7, state=TrialState.COMPLETE, value=None),␣
↪FrozenTrial(number=8, values=[20.994735530038245], datetime_start=datetime.
↪datetime(2022, 5, 26, 12, 5, 35, 133174), datetime_complete=datetime.datetime(2022,␣
```

(下页继续)

(续上页)

获得 trial 的数目：

```
len(study.trials)
```

Out:

```
100
```

再次执行*optimize()*，我们可以继续优化过程。

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

获得更新后的的 trial 数量：

```
len(study.trials)
```

Out:

```
200
```

由于此目标函数非常简单，所以后面的 100 个 trial 并没有提升结果。然而，我们可以再一次检查该结果：

```
best_params = study.best_params
found_x = best_params["x"]
print("Found x: {}, (x - 2)^2: {}".format(found_x, (found_x - 2) ** 2))
```

Out:

```
Found x: 2.0007453774691357, (x - 2)^2: 5.555875714951739e-07
```

Total running time of the script: (0 minutes 0.799 seconds)

Python 式的搜索空间

对于超参数采样，Optuna 提供了以下特性：

- *optuna.trial.Trial.suggest_categorical()* 用于类别参数
- *optuna.trial.Trial.suggest_int()* 用于整形参数
- *optuna.trial.Trial.suggest_float()* 用于浮点型参数

通过可选的 `step` 与 `log` 参数，我们可以对整形或者浮点型参数进行离散化或者取对数操作。

```
import optuna


def objective(trial):
    # Categorical parameter
    optimizer = trial.suggest_categorical("optimizer", ["MomentumSGD", "Adam"])

    # Integer parameter
    num_layers = trial.suggest_int("num_layers", 1, 3)

    # Integer parameter (log)
    num_channels = trial.suggest_int("num_channels", 32, 512, log=True)

    # Integer parameter (discretized)
    num_units = trial.suggest_int("num_units", 10, 100, step=5)

    # Floating point parameter
    dropout_rate = trial.suggest_float("dropout_rate", 0.0, 1.0)

    # Floating point parameter (log)
    learning_rate = trial.suggest_float("learning_rate", 1e-5, 1e-2, log=True)

    # Floating point parameter (discretized)
    drop_path_rate = trial.suggest_float("drop_path_rate", 0.0, 1.0, step=0.1)
```

定义参数空间

在 Optuna 中，我们使用和 Python 语法类似的方式来定义搜索空间，其中包含条件和循环语句。

类似地，你也可以根据参数值采用分支或者循环。

更多用法见 examples.

- 分支：

```
import sklearn.ensemble
import sklearn.svm


def objective(trial):
    classifier_name = trial.suggest_categorical("classifier", ["SVC", "RandomForest"])
    if classifier_name == "SVC":
        svc_c = trial.suggest_float("svc_c", 1e-10, 1e10, log=True)
        classifier_obj = sklearn.svm.SVC(C=svc_c)
```

(下页继续)

(续上页)

```
    else:
        rf_max_depth = trial.suggest_int("rf_max_depth", 2, 32, log=True)
        classifier_obj = sklearn.ensemble.RandomForestClassifier(max_depth=rf_max_
↪depth)
```

- 循环:

```
import torch
import torch.nn as nn


def create_model(trial, in_size):
    n_layers = trial.suggest_int("n_layers", 1, 3)

    layers = []
    for i in range(n_layers):
        n_units = trial.suggest_int("n_units_l{}".format(i), 4, 128, log=True)
        layers.append(nn.Linear(in_size, n_units))
        layers.append(nn.ReLU())
        in_size = n_units
    layers.append(nn.Linear(in_size, 10))

    return nn.Sequential(*layers)
```

关于参数个数的注意事项

随着参数个数的增长，优化的难度约呈指数增长。也就是说，当你增加参数的个数的时候，优化所需要的 trial 个数会呈指数增长。因此我们不推荐增加不必要的参数。

Total running time of the script: (0 minutes 0.001 seconds)

高效的优化算法

通过采用最先进的超参数采样算法和对无望 trial 的剪枝，Optuna 使得高效的超参数优化成为可能。

采样算法

利用 suggested 参数值和评估的目标值的记录，采样器基本上不断缩小搜索空间，直到找到一个最佳的搜索空间，其产生的参数会带来更好的目标函数值。关于采样器如何 suggest 参数的更详细的解释见*optuna.samplers.BaseSampler*.

Optuna 提供了下列采样算法：

- *optuna.samplers.TPESampler* 实现的 Tree-structured Parzen Estimator 算法
- *optuna.samplers.CmaEsSampler* 实现的 CMA-ES 算法
- *optuna.samplers.GridSampler* 实现的网格搜索
- *optuna.samplers.RandomSampler* 实现的随机搜索

默认的采样器是*optuna.samplers.TPESampler*.

切换采样器

```
import optuna
```

默认情况下，Optuna 这样使用*TPESampler*.

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

Out:

```
Sampler is TPESampler
```

如果你希望使用其他采样器，比如*RandomSampler* 和*CmaEsSampler*,

```
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__}")
```

Out:

```
Sampler is RandomSampler
Sampler is CmaEsSampler
```

剪枝算法

Pruners 自动在训练的早期（也就是自动化的 early-stopping）终止无望的 trial.

Optuna 提供以下剪枝算法：

- *optuna.pruners.SuccessiveHalvingPruner* 实现的 Asynchronous Successive Halving 算法。
- *optuna.pruners.HyperbandPruner* 实现的 Hyperband 算法。
- *optuna.pruners.MedianPruner* 实现的中位数剪枝算法
- *optuna.pruners.ThresholdPruner* 实现的阈值剪枝算法

在大多数例子中我们采用的*optuna.pruners.MedianPruner*，尽管其性能基本上会被*optuna.pruners.SuccessiveHalvingPruner*和*optuna.pruners.HyperbandPruner*超过，就像在 这个基准测试结果 中那样.

激活 Pruner

要打开剪枝特性的话，你需要在迭代式训练的每一步后调用*report()*和*should_prune()*. *report()* 定期监控目标函数的中间值. *should_prune()* 确定终结那些没有达到预先设定条件的 trial.

我们推荐在主流机器学习框架中使用集成模块，全部的模块列表在*optuna.integration* 里，用例见 optuna/examples.

```
import logging
import sys

import sklearn.datasets
import sklearn.linear_model
import sklearn.model_selection


def objective(trial):
    iris = sklearn.datasets.load_iris()
    classes = list(set(iris.target))
    train_x, valid_x, train_y, valid_y = sklearn.model_selection.train_test_split(
        iris.data, iris.target, test_size=0.25, random_state=0
    )

    alpha = trial.suggest_float("alpha", 1e-5, 1e-1, log=True)
```

(下页继续)

(续上页)

```
    clf = sklearn.linear_model.SGDClassifier(alpha=alpha)

    for step in range(100):
        clf.partial_fit(train_x, train_y, classes=classes)

        # Report intermediate objective value.
        intermediate_value = 1.0 - clf.score(valid_x, valid_y)
        trial.report(intermediate_value, step)

        # Handle pruning based on the intermediate value.
        if trial.should_prune():
            raise optuna.TrialPruned()

    return 1.0 - clf.score(valid_x, valid_y)
```

将中位数终止规则设置为剪枝条件。

```
# Add stream handler of stdout to show the messages
optuna.logging.get_logger("optuna").addHandler(logging.StreamHandler(sys.stdout))
study = optuna.create_study(pruner=optuna.pruners.MedianPruner())
study.optimize(objective, n_trials=20)
```

Out:

```
A new study created in memory with name: no-name-2550f085-0161-4804-aabe-dafd3d2a69fc
Trial 0 finished with value: 0.23684210526315785 and parameters: {'alpha': 0.
↪0056457619503721985}. Best is trial 0 with value: 0.23684210526315785.
Trial 1 finished with value: 0.10526315789473684 and parameters: {'alpha': 0.
↪0004415029135399882}. Best is trial 1 with value: 0.10526315789473684.
Trial 2 finished with value: 0.2894736842105263 and parameters: {'alpha': 0.
↪003557613371651606}. Best is trial 1 with value: 0.10526315789473684.
Trial 3 finished with value: 0.10526315789473684 and parameters: {'alpha': 1.
↪9394951442030858e-05}. Best is trial 1 with value: 0.10526315789473684.
Trial 4 finished with value: 0.39473684210526316 and parameters: {'alpha': 2.
↪6138481035133756e-05}. Best is trial 1 with value: 0.10526315789473684.
Trial 5 pruned.
Trial 6 finished with value: 0.26315789473684215 and parameters: {'alpha': 0.
↪006413605309772462}. Best is trial 1 with value: 0.10526315789473684.
Trial 7 pruned.
Trial 8 pruned.
Trial 9 pruned.
Trial 10 finished with value: 0.3421052631578947 and parameters: {'alpha': 0.
↪0003506682152386342}. Best is trial 1 with value: 0.10526315789473684.
Trial 11 pruned.
```

(下页继续)

(续上页)

```
Trial 12 pruned.
Trial 13 pruned.
Trial 14 pruned.
Trial 15 pruned.
Trial 16 pruned.
Trial 17 pruned.
Trial 18 pruned.
Trial 19 pruned.
```

如你所见，有几个 trial 在其迭代完成之前被剪枝（终止）了。消息格式是 "Trial <Trial Number> pruned.".

应该使用哪个 pruner 呢？

对于非深度学习来说，根据 optuna/optuna - wiki "Benchmarks with Kurobako" 里的基准测试结果，我们推荐

- 对*optuna.samplers.RandomSampler* 而言*optuna.pruners.MedianPruner* 是最好的。
- 对于*optuna.samplers.TPESampler* 而言 optuna.pruners.Hyperband 是最好的。

不过，注意这个基准测试不是深度学习。对于深度学习而言，请参考 Ozaki et al, Hyperparameter Optimization Methods: Overview and Characteristics, in IEICE Trans, Vol.J103-D No.9 pp.615-631, 2020 里的这张表格。

Parallel Compute Resource	Categorical/Conditional Hyperparameters	Recommended Algorithms
Limited	No	TPE. GP-EI if search space is low-dimensional and continuous.
	Yes	TPE. GP-EI if search space is low-dimensional and continuous
Sufficient	No	CMA-ES, Random Search
	Yes	Random Search or Genetic Algorithm

用于剪枝的集成模块

为了用最简单的形式实现剪枝算法，Optuna 为以下库提供了集成模块。

关于 Optuna 集成模块的完整列表，参见*optuna.integration*.

For example, *XGBoostPruningCallback* introduces pruning without directly changing the logic of training iteration. (See also example for the entire script.)

```
pruning_callback = optuna.integration.XGBoostPruningCallback(trial, 'validation-error
↪')
bst = xgb.train(param, dtrain, evals=[(dvalid, 'validation')], callbacks=[pruning_
↪callback])
```

Total running time of the script: (0 minutes 1.443 seconds)

简单的并行化

并行化非常直接*optuna.study.Study.optimize()*.

如果你想要手动执行 Optuna 的优化：

1. 启动一个 RDB 服务器（在本例中我们用 MySQL）
2. 通过 *–storage* 参数创建 study
3. 在多个节点和进程之间共享 study

当然，你可以像 the kubernetes examples 里一样使用 Kubernetes.

要看并行优化怎么进行的化，请查看下面这个视频。

创建 study

你可以通过 `optuna create-study` 来创建 study. 或者在 Python 脚本中通过*optuna.create_study()*也行。

```
$ mysql -u root -e "CREATE DATABASE IF NOT EXISTS example"
$ optuna create-study --study-name "distributed-example" --storage "mysql://
↪root@localhost/example"
[I 2020-07-21 13:43:39,642] A new study created with name: distributed-example
```

然后写一个优化脚本。假设 `foo.py` 包含了以下代码。

```
import optuna


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


if __name__ == "__main__":
    study = optuna.load_study(
        study_name="distributed-example", storage="mysql://root@localhost/example"
```

(下页继续)

(续上页)

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

在多个节点和进程之间分享该 study

最后，在多个进程中运行这个共享的 study. 比如，在一个终端中运行 `Process 1` 而在另一个终端中运行 `Process 2`. 它们基于共享参数的历史记录来获取参数 suggestion.

Process 1:

```
$ python foo.py
[I 2020-07-21 13:45:02,973] Trial 0 finished with value: 45.35553104173011 and␣
↪parameters: {'x': 8.73465151598285}. Best is trial 0 with value: 45.35553104173011.
[I 2020-07-21 13:45:04,013] Trial 2 finished with value: 4.6002397305938905 and␣
↪parameters: {'x': 4.144816945707463}. Best is trial 1 with value: 0.
↪028194513284051464.
...
```

Process 2 (the same command as process 1):

```
$ python foo.py
[I 2020-07-21 13:45:03,748] Trial 1 finished with value: 0.028194513284051464 and␣
↪parameters: {'x': 1.8320877810162361}. Best is trial 1 with value: 0.
↪028194513284051464.
[I 2020-07-21 13:45:05,783] Trial 3 finished with value: 24.45966755098074 and␣
↪parameters: {'x': 6.945671597566982}. Best is trial 1 with value: 0.
↪028194513284051464.
...
```

备注： 我们不建议在大规模的分布式优化中采用 SQLite 因为其可能造成死锁和烟 zhngde 性能问题。请考虑其他数据库引擎，比如 PostgreSQL 或者 MySQL.

备注： 在分布式优化里，请不要将 SQLite 数据库放在网络文件系统上。参见 https://www.sqlite.org/faq.html#q5

Total running time of the script: (0 minutes 0.000 seconds)

用于超参数优化分析的快速可视化

Optuna 在*optuna.visualization*里提供了各种可视化特性用于分析优化结果。

通过可视化乳腺癌数据集上的 lightgbm 历史记录，本教程将带你体验此模块。

```
import lightgbm as lgb
import numpy as np
import sklearn.datasets
import sklearn.metrics
from sklearn.model_selection import train_test_split

import optuna
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_slice

SEED = 42

np.random.seed(SEED)
```

定义目标函数

```
def objective(trial):
    data, target = sklearn.datasets.load_breast_cancer(return_X_y=True)
    train_x, valid_x, train_y, valid_y = train_test_split(data, target, test_size=0.
↪25)
    dtrain = lgb.Dataset(train_x, label=train_y)
    dvalid = lgb.Dataset(valid_x, label=valid_y)

    param = {
        "objective": "binary",
        "metric": "auc",
        "verbosity": -1,
        "boosting_type": "gbdt",
        "bagging_fraction": trial.suggest_float("bagging_fraction", 0.4, 1.0),
        "bagging_freq": trial.suggest_int("bagging_freq", 1, 7),
        "min_child_samples": trial.suggest_int("min_child_samples", 5, 100),
    }

    # Add a callback for pruning.
```

(下页继续)

(续上页)

```
    pruning_callback = optuna.integration.LightGBMPruningCallback(trial, "auc")
    gbm = lgb.train(
        param, dtrain, valid_sets=[dvalid], verbose_eval=False, callbacks=[pruning_
↪callback]
    )

    preds = gbm.predict(valid_x)
    pred_labels = np.rint(preds)
    accuracy = sklearn.metrics.accuracy_score(valid_y, pred_labels)
    return accuracy
```

```
study = optuna.create_study(
    direction="maximize",
    sampler=optuna.samplers.TPESampler(seed=SEED),
    pruner=optuna.pruners.MedianPruner(n_warmup_steps=10),
)
study.optimize(objective, n_trials=100, timeout=600)
```

Out:

```
/home/docs/checkouts/readthedocs.org/user_builds/optuna-zh-cn/envs/stable/lib/python3.
↪7/site-packages/lightgbm/engine.py:239: UserWarning:

'verbose_eval' argument is deprecated and will be removed in a future release of␣
↪LightGBM. Pass 'log_evaluation()' callback via 'callbacks' argument instead.

/home/docs/checkouts/readthedocs.org/user_builds/optuna-zh-cn/envs/stable/lib/python3.
↪7/site-packages/lightgbm/engine.py:239: UserWarning:

'verbose_eval' argument is deprecated and will be removed in a future release of␣
↪LightGBM. Pass 'log_evaluation()' callback via 'callbacks' argument instead.

/home/docs/checkouts/readthedocs.org/user_builds/optuna-zh-cn/envs/stable/lib/python3.
↪7/site-packages/lightgbm/engine.py:239: UserWarning:

'verbose_eval' argument is deprecated and will be removed in a future release of␣
↪LightGBM. Pass 'log_evaluation()' callback via 'callbacks' argument instead.

/home/docs/checkouts/readthedocs.org/user_builds/optuna-zh-cn/envs/stable/lib/python3.
↪7/site-packages/lightgbm/engine.py:239: UserWarning:

'verbose_eval' argument is deprecated and will be removed in a future release of␣
↪LightGBM. Pass 'log_evaluation()' callback via 'callbacks' argument instead.
```

(下页继续)

(续上页)

```
/home/docs/checkouts/readthedocs.org/user_builds/optuna-zh-cn/envs/stable/lib/python3.
↪7/site-packages/lightgbm/engine.py:239: UserWarning:

'verbose_eval' argument is deprecated and will be removed in a future release of␣
↪LightGBM. Pass 'log_evaluation()' callback via 'callbacks' argument instead.

/home/docs/checkouts/readthedocs.org/user_builds/optuna-zh-cn/envs/stable/lib/python3.
↪7/site-packages/lightgbm/engine.py:239: UserWarning:

'verbose_eval' argument is deprecated and will be removed in a future release of␣
↪LightGBM. Pass 'log_evaluation()' callback via 'callbacks' argument instead.

/home/docs/checkouts/readthedocs.org/user_builds/optuna-zh-cn/envs/stable/lib/python3.
↪7/site-packages/lightgbm/engine.py:239: UserWarning:

'verbose_eval' argument is deprecated and will be removed in a future release of␣
↪LightGBM. Pass 'log_evaluation()' callback via 'callbacks' argument instead.

/home/docs/checkouts/readthedocs.org/user_builds/optuna-zh-cn/envs/stable/lib/python3.
↪7/site-packages/lightgbm/engine.py:239: UserWarning:

'verbose_eval' argument is deprecated and will be removed in a future release of␣
↪LightGBM. Pass 'log_evaluation()' callback via 'callbacks' argument instead.

/home/docs/checkouts/readthedocs.org/user_builds/optuna-zh-cn/envs/stable/lib/python3.
↪7/site-packages/lightgbm/engine.py:239: UserWarning:

'verbose_eval' argument is deprecated and will be removed in a future release of␣
↪LightGBM. Pass 'log_evaluation()' callback via 'callbacks' argument instead.

/home/docs/checkouts/readthedocs.org/user_builds/optuna-zh-cn/envs/stable/lib/python3.
↪7/site-packages/lightgbm/engine.py:239: UserWarning:

'verbose_eval' argument is deprecated and will be removed in a future release of␣
↪LightGBM. Pass 'log_evaluation()' callback via 'callbacks' argument instead.

/home/docs/checkouts/readthedocs.org/user_builds/optuna-zh-cn/envs/stable/lib/python3.
↪7/site-packages/lightgbm/engine.py:239: UserWarning:

'verbose_eval' argument is deprecated and will be removed in a future release of␣
↪LightGBM. Pass 'log_evaluation()' callback via 'callbacks' argument instead.
```

(下页继续)

(续上页)

```
/home/docs/checkouts/readthedocs.org/user_builds/optuna-zh-cn/envs/stable/lib/python3.
↪7/site-packages/lightgbm/engine.py:239: UserWarning:

'verbose_eval' argument is deprecated and will be removed in a future release of␣
↪LightGBM. Pass 'log_evaluation()' callback via 'callbacks' argument instead.

/home/docs/checkouts/readthedocs.org/user_builds/optuna-zh-cn/envs/stable/lib/python3.
↪7/site-packages/lightgbm/engine.py:239: UserWarning:

'verbose_eval' argument is deprecated and will be removed in a future release of␣
↪LightGBM. Pass 'log_evaluation()' callback via 'callbacks' argument instead.

/home/docs/checkouts/readthedocs.org/user_builds/optuna-zh-cn/envs/stable/lib/python3.
↪7/site-packages/lightgbm/engine.py:239: UserWarning:

'verbose_eval' argument is deprecated and will be removed in a future release of␣
↪LightGBM. Pass 'log_evaluation()' callback via 'callbacks' argument instead.

/home/docs/checkouts/readthedocs.org/user_builds/optuna-zh-cn/envs/stable/lib/python3.
↪7/site-packages/lightgbm/engine.py:239: UserWarning:

'verbose_eval' argument is deprecated and will be removed in a future release of␣
↪LightGBM. Pass 'log_evaluation()' callback via 'callbacks' argument instead.
```

绘图函数

对优化历史进行可视化。细节见*plot_optimization_history()*.

```
plot_optimization_history(study)
```

绘制各 trial 的学习曲线。细节见*plot_intermediate_values()*.

```
plot_intermediate_values(study)
```

绘制高维参数关系。细节见*plot_parallel_coordinate()*.

```
plot_parallel_coordinate(study)
```

选择要绘制的参数

```
plot_parallel_coordinate(study, params=["bagging_freq", "bagging_fraction"])
```

绘制超参数关系。细节见*plot_contour()*.

```
plot_contour(study)
```

选择要绘制的参数

```
plot_contour(study, params=["bagging_freq", "bagging_fraction"])
```

绘制各超参数的切片图。细节见*plot_slice()*.

```
plot_slice(study)
```

选择要绘制的参数

```
plot_slice(study, params=["bagging_freq", "bagging_fraction"])
```

绘制超参数重要性。细节见*plot_param_importances()*.

```
plot_param_importances(study)
```

绘制经验分布函数。*plot_edf()*.

```
plot_edf(study)
```

Total running time of the script: (0 minutes 4.634 seconds)

6.2.2 用法

展现可能对你轻松使用 Optuna 更有帮助的一些用法

用 RDB 后端保存/恢复 Study

RDB 后端可以实现持久化实验（即保存和恢复 study）以及访问 study 的历史记录。此外，我们还可以利用这个特点来进行分布式优化。具体描述见简单的并行化.

在本部分中，我们将尝试一个在本地环境下运行 SQLite DB 的简单例子。

备注： 通过设置 DB 的 storage URL 参数，你也可以使用其他的 RDB 后端，比如 PostgreSQL 或者 MySQL. 设置 URL 的方式参见 SQLAlchemy 的文档.

新建 Study

通过调用函数*create_study()*，我们可以创建一个持久化的 study. 创建新 study 会自动初始化一个 SQLite 文件 example.db.

```
import logging
import sys

import optuna

# Add stream handler of stdout to show the messages
optuna.logging.get_logger("optuna").addHandler(logging.StreamHandler(sys.stdout))
study_name = "example-study"  # Unique identifier of the study.
storage_name = "sqlite:///{}.db".format(study_name)
study = optuna.create_study(study_name=study_name, storage=storage_name)
```

Out:

```
A new study created in RDB with name: example-study
```

为了运行一个 study, 我们需要将目标函数传入*optimize()* 方法并调用它。

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


study.optimize(objective, n_trials=3)
```

Out:

```
Trial 0 finished with value: 1.5067618568079724 and parameters: {'x': 3.
↪2275022838300433}. Best is trial 0 with value: 1.5067618568079724.
Trial 1 finished with value: 24.96165877167141 and parameters: {'x': -2.
↪996164405988999}. Best is trial 0 with value: 1.5067618568079724.
Trial 2 finished with value: 0.45814470516718514 and parameters: {'x': 2.
↪676863874916652}. Best is trial 2 with value: 0.45814470516718514.
```

恢复 Study

为了恢复 study, 首先需要初始化一个*Study* 对象，并将该 study 的名字 `example-study` 和 DB URL 参数 `sqlite:///example.db` 传入其中。

```
study = optuna.create_study(study_name=study_name, storage=storage_name, load_if_
↪exists=True)
study.optimize(objective, n_trials=3)
```

Out:

```
Using an existing study with name 'example-study' instead of creating a new one.
Trial 3 finished with value: 54.55663920492726 and parameters: {'x': 9.38624662497315}
↪. Best is trial 2 with value: 0.45814470516718514.
Trial 4 finished with value: 14.274094669669244 and parameters: {'x': -1.
↪7781072866806262}. Best is trial 2 with value: 0.45814470516718514.
Trial 5 finished with value: 99.93271079015234 and parameters: {'x': -7.
↪996634973337395}. Best is trial 2 with value: 0.45814470516718514.
```

实验历史记录

我们可以通过*Study* 类来获得 study 和对应 trials 的历史记录。比如，下面的语句可以获取 `example-study` 的所有 trials.

```
study = optuna.create_study(study_name=study_name, storage=storage_name, load_if_
↪exists=True)
df = study.trials_dataframe(attrs=("number", "value", "params", "state"))
```

Out:

```
Using an existing study with name 'example-study' instead of creating a new one.
```

trials_dataframe() 方法会返回一个如下的 pandas dataframe：

```
print(df)
```

Out:

```
   number      value  params_x     state
0       0   1.506762  3.227502  COMPLETE
1       1  24.961659 -2.996164  COMPLETE
2       2   0.458145  2.676864  COMPLETE
3       3  54.556639  9.386247  COMPLETE
```

(下页继续)

(续上页)

```
4        4  14.274095 -1.778107  COMPLETE
5        5  99.932711 -7.996635  COMPLETE
```

Study 对象也有一些其他属性，比如*trials*, *best_value* 和*best_params* (见轻量级、多功能和跨平台架构).

```
print("Best params: ", study.best_params)
print("Best value: ", study.best_value)
print("Best Trial: ", study.best_trial)
print("Trials: ", study.trials)
```

Out:

```
Best params:  {'x': 2.676863874916652}
Best value:  0.45814470516718514
Best Trial:  FrozenTrial(number=2, values=[0.45814470516718514], datetime_
↪start=datetime.datetime(2022, 5, 26, 12, 5, 42, 892134), datetime_complete=datetime.
↪datetime(2022, 5, 26, 12, 5, 42, 908612), params={'x': 2.676863874916652},␣
↪distributions={'x': UniformDistribution(high=10.0, low=-10.0)}, user_attrs={},␣
↪system_attrs={}, intermediate_values={}, trial_id=3, state=TrialState.COMPLETE,␣
↪value=None)
Trials:  [FrozenTrial(number=0, values=[1.5067618568079724], datetime_start=datetime.
↪datetime(2022, 5, 26, 12, 5, 42, 796078), datetime_complete=datetime.datetime(2022,␣
↪5, 26, 12, 5, 42, 816506), params={'x': 3.2275022838300433}, distributions={'x':␣
↪UniformDistribution(high=10.0, low=-10.0)}, user_attrs={}, system_attrs={},␣
↪intermediate_values={}, trial_id=1, state=TrialState.COMPLETE, value=None),␣
↪FrozenTrial(number=1, values=[24.96165877167141], datetime_start=datetime.
↪datetime(2022, 5, 26, 12, 5, 42, 852903), datetime_complete=datetime.datetime(2022,␣
↪5, 26, 12, 5, 42, 867951), params={'x': -2.996164405988999}, distributions={'x':␣
↪UniformDistribution(high=10.0, low=-10.0)}, user_attrs={}, system_attrs={},␣
↪intermediate_values={}, trial_id=2, state=TrialState.COMPLETE, value=None),␣
↪FrozenTrial(number=2, values=[0.45814470516718514], datetime_start=datetime.
↪datetime(2022, 5, 26, 12, 5, 42, 892134), datetime_complete=datetime.datetime(2022,␣
↪5, 26, 12, 5, 42, 908612), params={'x': 2.676863874916652}, distributions={'x':␣
↪UniformDistribution(high=10.0, low=-10.0)}, user_attrs={}, system_attrs={},␣
↪intermediate_values={}, trial_id=3, state=TrialState.COMPLETE, value=None),␣
↪FrozenTrial(number=3, values=[54.55663920492726], datetime_start=datetime.
↪datetime(2022, 5, 26, 12, 5, 42, 976636), datetime_complete=datetime.datetime(2022,␣
↪5, 26, 12, 5, 42, 995327), params={'x': 9.38624662497315}, distributions={'x':␣
↪UniformDistribution(high=10.0, low=-10.0)}, user_attrs={}, system_attrs={},␣
↪intermediate_values={}, trial_id=4, state=TrialState.COMPLETE, value=None),␣
↪FrozenTrial(number=4, values=[14.274094669669244], datetime_start=datetime.
↪datetime(2022, 5, 26, 12, 5, 43, 26577), datetime_complete=datetime.datetime(2022,␣
↪5, 26, 12, 5, 43, 42780), params={'x': -1.7781072866806262}, distributions={'x':␣
↪UniformDistribution(high=10.0, low=-10.0)}, user_attrs={}, system_attrs={},␣
↪intermediate_values={}, trial_id=5, state=TrialState.COMPLETE, value=None),␣
↪FrozenTrial(number=5, values=[99.93271079015234], datetime_start=datetime.
↪datetime(2022, 5, 26, 12, 5, 43, 66882), datetime_complete=datetime.datetime(2022,␣
↪5, 26, 12, 5, 43, 81061), params={'x': -7.996634973337395}, distributions={'x':␣
↪UniformDistribution(high=10.0, low=-10.0)}, user_attrs={}, system_attrs={},␣
```

(下页继续)

(续上页)

Total running time of the script: (0 minutes 0.622 seconds)

用 Optuna 进行多目标优化

This tutorial showcases Optuna’s multi-objective optimization feature by optimizing the validation accuracy of Fashion MNIST dataset and the FLOPS of the model implemented in PyTorch.

我们采用 thop 来测量 FLOPS.

```
import thop
import torch
import torch.nn as nn
import torch.nn.functional as F
import torchvision

import optuna


DEVICE = torch.device("cuda") if torch.cuda.is_available() else torch.device("cpu")
DIR = ".."
BATCHSIZE = 128
N_TRAIN_EXAMPLES = BATCHSIZE * 30
N_VALID_EXAMPLES = BATCHSIZE * 10


def define_model(trial):
    n_layers = trial.suggest_int("n_layers", 1, 3)
    layers = []

    in_features = 28 * 28
    for i in range(n_layers):
        out_features = trial.suggest_int("n_units_l{}".format(i), 4, 128)
        layers.append(nn.Linear(in_features, out_features))
        layers.append(nn.ReLU())
        p = trial.suggest_float("dropout_{}".format(i), 0.2, 0.5)
        layers.append(nn.Dropout(p))

        in_features = out_features

    layers.append(nn.Linear(in_features, 10))
    layers.append(nn.LogSoftmax(dim=1))
```

(下页继续)

(续上页)

```
    return nn.Sequential(*layers)


# Defines training and evaluation.
def train_model(model, optimizer, train_loader):
    model.train()
    for batch_idx, (data, target) in enumerate(train_loader):
        data, target = data.view(-1, 28 * 28).to(DEVICE), target.to(DEVICE)
        optimizer.zero_grad()
        F.nll_loss(model(data), target).backward()
        optimizer.step()


def eval_model(model, valid_loader):
    model.eval()
    correct = 0
    with torch.no_grad():
        for batch_idx, (data, target) in enumerate(valid_loader):
            data, target = data.view(-1, 28 * 28).to(DEVICE), target.to(DEVICE)
            pred = model(data).argmax(dim=1, keepdim=True)
            correct += pred.eq(target.view_as(pred)).sum().item()

    accuracy = correct / N_VALID_EXAMPLES

    flops, _ = thop.profile(model, inputs=(torch.randn(1, 28 * 28).to(DEVICE),),␣
↪verbose=False)
    return flops, accuracy
```

Out:

```
/home/docs/checkouts/readthedocs.org/user_builds/optuna-zh-cn/envs/stable/lib/python3.
↪7/site-packages/torch/cuda/__init__.py:52: UserWarning:

CUDA initialization: Found no NVIDIA driver on your system. Please check that you␣
↪have an NVIDIA GPU and installed a driver from http://www.nvidia.com/Download/index.
↪aspx (Triggered internally at  /pytorch/c10/cuda/CUDAFunctions.cpp:100.)
```

定义多目标函数. 其目标为 FLOPS 和准确度.

```
def objective(trial):
    train_dataset = torchvision.datasets.FashionMNIST(
        DIR, train=True, download=True, transform=torchvision.transforms.ToTensor()
    )
    train_loader = torch.utils.data.DataLoader(
```

(下页继续)

(续上页)

```
        torch.utils.data.Subset(train_dataset, list(range(N_TRAIN_EXAMPLES))),
        batch_size=BATCHSIZE,
        shuffle=True,
    )

    val_dataset = torchvision.datasets.FashionMNIST(
        DIR, train=False, transform=torchvision.transforms.ToTensor()
    )
    val_loader = torch.utils.data.DataLoader(
        torch.utils.data.Subset(val_dataset, list(range(N_VALID_EXAMPLES))),
        batch_size=BATCHSIZE,
        shuffle=True,
    )
    model = define_model(trial).to(DEVICE)

    optimizer = torch.optim.Adam(
        model.parameters(), trial.suggest_float("lr", 1e-5, 1e-1, log=True)
    )

    for epoch in range(10):
        train_model(model, optimizer, train_loader)
    flops, accuracy = eval_model(model, val_loader)
    return flops, accuracy
```

运行多目标优化.

如果你的优化问题是多目标的, Optuna 预设你会为每一个目标指定优化方向. 具体在本例中, 我们希望最小化 FLOPS (我们想要更快的模型) 和最大化准确度. 所以我们将 `directions` 设置为 `["minimize", "maximize"]`.

```
study = optuna.create_study(directions=["minimize", "maximize"])
study.optimize(objective, n_trials=30, timeout=300)

print("Number of finished trials: ", len(study.trials))
```

Out:

```
Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/train-images-
↪idx3-ubyte.gz to ../FashionMNIST/raw/train-images-idx3-ubyte.gz

0it [00:00, ?it/s]
  0%|          | 0/26421880 [00:00<?, ?it/s]
```

(下页继续)

(续上页)

```
  0%|          | 49152/26421880 [00:00<01:47, 245800.96it/s]
  0%|          | 114688/26421880 [00:00<01:03, 415451.53it/s]
  1%|          | 212992/26421880 [00:00<00:41, 626641.77it/s]
  2%|1         | 442368/26421880 [00:00<00:21, 1205029.17it/s]
  3%|3         | 892928/26421880 [00:00<00:11, 2298475.10it/s]
  7%|6         | 1794048/26421880 [00:00<00:05, 4454796.77it/s]
 13%|#2        | 3342336/26421880 [00:01<00:02, 7919072.06it/s]
 18%|#8        | 4882432/26421880 [00:01<00:02, 10227247.85it/s]
 24%|##4       | 6471680/26421880 [00:01<00:01, 11960396.41it/s]
 30%|###       | 8052736/26421880 [00:01<00:01, 13096020.86it/s]
 37%|###6      | 9650176/26421880 [00:01<00:01, 13949535.11it/s]
 43%|####2     | 11239424/26421880 [00:01<00:01, 14530982.26it/s]
 49%|####8     | 12836864/26421880 [00:01<00:00, 14918256.39it/s]
 55%|#####4    | 14442496/26421880 [00:01<00:00, 15253596.43it/s]
 61%|######    | 16048128/26421880 [00:01<00:00, 15465740.17it/s]
 67%|######6   | 17678336/26421880 [00:01<00:00, 15707752.46it/s]
 73%|#######3  | 19316736/26421880 [00:02<00:00, 15890648.33it/s]
 79%|#######9  | 20963328/26421880 [00:02<00:00, 16058160.88it/s]
 86%|########5 | 22609920/26421880 [00:02<00:00, 16155043.42it/s]
 92%|#########1| 24264704/26421880 [00:02<00:00, 16263152.14it/s]
 98%|#########8| 25919488/26421880 [00:02<00:00, 16325161.67it/s]
26427392it [00:02, 10708587.67it/s]
Extracting ../FashionMNIST/raw/train-images-idx3-ubyte.gz to ../FashionMNIST/raw
Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/train-labels-
↪idx1-ubyte.gz to ../FashionMNIST/raw/train-labels-idx1-ubyte.gz

0it [00:00, ?it/s]
  0%|          | 0/29515 [00:00<?, ?it/s]
32768it [00:00, 99247.73it/s]
Extracting ../FashionMNIST/raw/train-labels-idx1-ubyte.gz to ../FashionMNIST/raw
Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/t10k-images-
↪idx3-ubyte.gz to ../FashionMNIST/raw/t10k-images-idx3-ubyte.gz

0it [00:00, ?it/s]
  0%|          | 0/4422102 [00:00<?, ?it/s]
  1%|          | 40960/4422102 [00:00<00:12, 361996.97it/s]
  2%|2         | 90112/4422102 [00:00<00:09, 434259.18it/s]
  4%|3         | 163840/4422102 [00:00<00:07, 568345.76it/s]
  8%|8         | 368640/4422102 [00:00<00:03, 1141556.84it/s]
 18%|#8        | 802816/4422102 [00:00<00:01, 2281822.61it/s]
 35%|###4      | 1531904/4422102 [00:00<00:00, 3971141.49it/s]
 67%|######6   | 2957312/4422102 [00:00<00:00, 7314825.62it/s]
4423680it [00:01, 4209306.93it/s]
```

(下页继续)

(续上页)

```
Extracting ../FashionMNIST/raw/t10k-images-idx3-ubyte.gz to ../FashionMNIST/raw
Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/t10k-labels-
↪idx1-ubyte.gz to ../FashionMNIST/raw/t10k-labels-idx1-ubyte.gz

0it [00:00, ?it/s]
  0%|          | 0/5148 [00:00<?, ?it/s]
8192it [00:00, 34965.80it/s]
Extracting ../FashionMNIST/raw/t10k-labels-idx1-ubyte.gz to ../FashionMNIST/raw
Processing...
/home/docs/checkouts/readthedocs.org/user_builds/optuna-zh-cn/envs/stable/lib/python3.
↪7/site-packages/torchvision/datasets/mnist.py:480: UserWarning:

The given NumPy array is not writeable, and PyTorch does not support non-writeable␣
↪tensors. This means you can write to the underlying (supposedly non-writeable)␣
↪NumPy array using the tensor. You may want to copy the array to protect its data or␣
↪make it writeable before converting it to a tensor. This type of warning will be␣
↪suppressed for the rest of this program. (Triggered internally at  /pytorch/torch/
↪csrc/utils/tensor_numpy.cpp:141.)

Done!
Number of finished trials:  30
```

可视化地检查位于帕累托前沿上的 trials

```
optuna.visualization.plot_pareto_front(study, target_names=["FLOPS", "accuracy"])
```

Out:

```
/home/docs/checkouts/readthedocs.org/user_builds/optuna-zh-cn/checkouts/stable/
↪tutorial/20_recipes/002_multi_objective.py:123: ExperimentalWarning:

plot_pareto_front is experimental (supported from v2.4.0). The interface can change␣
↪in the future.
```

Total running time of the script: (1 minutes 51.154 seconds)

用户定义属性

利用用户自定义属性，这个功能可以给实验做注解。

将用户定义属性添加到 Study

Study 对象提供了一个将键-值对设置为用户自定义属性的方法：*set_user_attr()*. 这里的键应该属于 str 类型，而值可以是任何能用 json.dumps 来序列化的对象。

```
import sklearn.datasets
import sklearn.model_selection
import sklearn.svm

import optuna


study = optuna.create_study(storage="sqlite:///example.db")
study.set_user_attr("contributors", ["Akiba", "Sano"])
study.set_user_attr("dataset", "MNIST")
```

我们可以利用 user_attr 属性来获取所有定义过的属性。

```
study.user_attrs  # {'contributors': ['Akiba', 'Sano'], 'dataset': 'MNIST'}
```

Out:

```
{'contributors': ['Akiba', 'Sano'], 'dataset': 'MNIST'}
```

StudySummary 对象中也包含了用户的自定义属性。我们可以从*get_all_study_summaries()* 中获取它。

```
study_summaries = optuna.get_all_study_summaries("sqlite:///example.db")
study_summaries[0].user_attrs  # {"contributors": ["Akiba", "Sano"], "dataset": "MNIST
↪"}
```

Out:

```
{'contributors': ['Akiba', 'Sano'], 'dataset': 'MNIST'}
```

参见:

在命令行界面里，optuna study set-user-attr 可用于设置用户定义属性。

将用户属性添加到 Trial 中

和*Study*类似，*Trial*对象也提供了一个设置属性的方法*set_user_attr()*方法。这些属性是在目标函数内部设置的。

```
def objective(trial):
    iris = sklearn.datasets.load_iris()
    x, y = iris.data, iris.target

    svc_c = trial.suggest_float("svc_c", 1e-10, 1e10, log=True)
    clf = sklearn.svm.SVC(C=svc_c)
    accuracy = sklearn.model_selection.cross_val_score(clf, x, y).mean()

    trial.set_user_attr("accuracy", accuracy)

    return 1.0 - accuracy  # return error for minimization


study.optimize(objective, n_trials=1)
```

可以用如下方式获取这些注解的属性：

```
study.trials[0].user_attrs
```

Out:

```
{'accuracy': 0.9266666666666667}
```

注意，在本例中，属性不是被注解到*Study*的，它属于一个单独的*Trial*.

Total running time of the script: (0 minutes 0.282 seconds)

命令行界面

Command	Description
create-study	创建一个新的 study.
delete-study	删除指定的 study.
dashboard	启动 web 面板 (beta).
storage upgrade	升级数据库 schema.
studies	输出 study 列表
study optimize	针对一个 study 开始优化过程。
study set-user-attr	设置 study 的用户属性。

Optuna 提供的命令行界面包括上表中的所有命令。

如果你不使用 IPython shell, 而是写 Python 脚本文件的话，编写像下面这样的脚本是完全可以的：

```
import optuna


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


if __name__ == "__main__":
    study = optuna.create_study()
    study.optimize(objective, n_trials=100)
    print("Best value: {} (params: {})\n".format(study.best_value, study.best_params))
```

Out:

```
Best value: 0.0010273417403227686 (params: {'x': 1.9679478278376836})
```

然而，通过 optuna 命令，我们可以少写一些上面这样的模版代码。假设有一个 foo.py 文件，它只包含如下代码：

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

即使在这种情况下，我们也可以利用如下命令启动优化过程（请暂时忽略 --storage sqlite:///example.db 的作用, 关于它的具体描述见*用 RDB 后端保存/恢复 Study*）。

```
$ cat foo.py
def objective(trial):
    x = trial.suggest_float('x', -10, 10)
    return (x - 2) ** 2

$ STUDY_NAME=`optuna create-study --storage sqlite:///example.db`
$ optuna study optimize foo.py objective --n-trials=100 --storage sqlite:///example.
↪db --study-name $STUDY_NAME
[I 2018-05-09 10:40:25,196] Finished a trial resulted in value: 54.353767789264026.␣
↪Current best value is 54.353767789264026 with parameters: {'x': -5.372500782588228}.
[I 2018-05-09 10:40:25,197] Finished a trial resulted in value: 15.784266965526376.␣
↪Current best value is 15.784266965526376 with parameters: {'x': 5.972941852774387}.
...
[I 2018-05-09 10:40:26,204] Finished a trial resulted in value: 14.704254135013741.␣
↪Current best value is 2.280758099793617e-06 with parameters: {'x': 1.
↪9984897821018828}.
```

请注意，`foo.py` 中只包含目标函数的定义。通过向 `optuna study optimize` 命令传递该文件名和对应目标函数的方法名，我们就可以启动优化过程。

Total running time of the script: (0 minutes 0.360 seconds)

用户定义的采样器 (Sampler)

你可以用用户定义的 sampler 来实现：

- 试验你自己的采样算法，
- 实现具体任务对应的算法来改进优化性能，或者
- 将其他的优化框架包装起来，整合进 Optuna 的流水线中 (比如*SkoptSampler*).

本节将介绍 sampler 类的内部行为，并展示一个实现用户自定义 sampler 的例子。

Sampler 概述

A sampler has the responsibility to determine the parameter values to be evaluated in a trial. When a *suggest* API (e.g., *suggest_float()*) is called inside an objective function, the corresponding distribution object (e.g., *UniformDistribution*) is created internally. A sampler samples a parameter value from the distribution. The sampled value is returned to the caller of the *suggest* API and evaluated in the objective function.

为了创建一个新的 sampler，你所定义的类需继承*BaseSampler*. 该基类提供三个抽象方法: *infer_relative_search_space()*, *sample_relative()* 和*sample_independent()*.

从这些方法名可以看出，Optuna 支持两种类型的采样过程：一种是 **relative sampling**, 它考虑了单个 trial 内参数之间的相关性，另一种是 **independent sampling**, 它对各个参数的采样是彼此独立的。

在一个 trial 刚开始时，*infer_relative_search_space()* 会被调用，它向该 trial 提供一个相对搜索空间。之后，*sample_relative()* 会被触发，它从该搜索空间中对相对参数进行采样。在目标函数的执行过程中，*sample_independent()* 用于对不属于该相对搜索空间的参数进行采样。

备注: 更多细节参见*BaseSampler* 的文档。

案例: 实现模拟退火 Sampler (SimulatedAnnealingSampler)

下面的代码根据 Simulated Annealing (SA) 定义类一个 sampler：

```
import numpy as np
import optuna
```

(下页继续)

(续上页)

```
class SimulatedAnnealingSampler(optuna.samplers.BaseSampler):
    def __init__(self, temperature=100):
        self._rng = np.random.RandomState()
        self._temperature = temperature  # Current temperature.
        self._current_trial = None  # Current state.

    def sample_relative(self, study, trial, search_space):
        if search_space == {}:
            return {}

        # Simulated Annealing algorithm.
        # 1. Calculate transition probability.
        prev_trial = study.trials[-2]
        if self._current_trial is None or prev_trial.value <= self._current_trial.
↪value:
            probability = 1.0
        else:
            probability = np.exp(
                (self._current_trial.value - prev_trial.value) / self._temperature
            )
        self._temperature *= 0.9  # Decrease temperature.

        # 2. Transit the current state if the previous result is accepted.
        if self._rng.uniform(0, 1) < probability:
            self._current_trial = prev_trial

        # 3. Sample parameters from the neighborhood of the current point.
        # The sampled parameters will be used during the next execution of
        # the objective function passed to the study.
        params = {}
        for param_name, param_distribution in search_space.items():
            if not isinstance(param_distribution, optuna.distributions.
↪UniformDistribution):
                raise NotImplementedError("Only suggest_float() is supported")

            current_value = self._current_trial.params[param_name]
            width = (param_distribution.high - param_distribution.low) * 0.1
            neighbor_low = max(current_value - width, param_distribution.low)
            neighbor_high = min(current_value + width, param_distribution.high)
            params[param_name] = self._rng.uniform(neighbor_low, neighbor_high)

        return params
```

(下页继续)

(续上页)

```
    # The rest are unrelated to SA algorithm: boilerplate
    def infer_relative_search_space(self, study, trial):
        return optuna.samplers.intersection_search_space(study)

    def sample_independent(self, study, trial, param_name, param_distribution):
        independent_sampler = optuna.samplers.RandomSampler()
        return independent_sampler.sample_independent(study, trial, param_name, param_
↪distribution)
```

备注: 为了代码的简洁性，上面的实现没有支持一些特性 (比如 maximization). 如果你对如何实现这些特性感兴趣，请看 examples/samplers/simulated_annealing.py.

你可以像使用内置的 sampler 一样使用 `SimulatedAnnealingSampler`:

```
def objective(trial):
    x = trial.suggest_float("x", -10, 10)
    y = trial.suggest_float("y", -5, 5)
    return x ** 2 + y


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

best_trial = study.best_trial
print("Best value: ", best_trial.value)
print("Parameters that achieve the best value: ", best_trial.params)
```

Out:

```
Best value:  -4.945941226368364
Parameters that achieve the best value:  {'x': 0.05100620365716724, 'y': -4.
↪948542859179881}
```

在上面这个优化过程中，参数 `x` 和 `y` 的值都是由 `SimulatedAnnealingSampler.sample_relative` 方法采样得出的。

备注: 严格意义上说，在第一个 trial 中，`SimulatedAnnealingSampler.sample_independent` 用于采样参数值。因为，如果没有已经完成的 trial 的话，`SimulatedAnnealingSampler.infer_relative_search_space` 中的*intersection_search_space()* 是无法对搜索空间进行推断的。

Total running time of the script: (0 minutes 0.376 seconds)

用户定义的 Pruner

在*optuna.pruners* 中, 我们描述了一个目标函数如何可选地包含一些 pruning 函数调用, 这些函数允许 Optuna 去终结中间结果无望的 tiral. 在本文档中, 我们描述了如何实现一个你自己的 pruner, 也即一个自定义 trial 终止条件的策略.

Pruning 界面概述

The *create_study()* constructor takes, as an optional argument, a pruner inheriting from *BasePruner*. The pruner should implement the abstract method *prune()*, which takes arguments for the associated *Study* and *Trial* and returns a boolean value: `True` if the trial should be pruned and `False` otherwise. Using the Study and Trial objects, you can access all other trials through the `get_trial()` method and, and from a trial, its reported intermediate values through the *intermediate_values()* (a dictionary which maps an integer `step` to a float value).

你可以参考 Optuna 内置的 pruner 的源代码作为构建你自己的 pruner 的模板. 在本文档中, 我们描述了一个简单 (但是却激进) 的 pruner 实现过程, 它会对那些处于同步骤 trial 中最后一名的 trial 进行剪枝.

备注: Please refer to the documentation of *BasePruner* or, for example, *ThresholdPruner* or *PercentilePruner* for more robust examples of pruner implementation, including error checking and complex pruner-internal logic.

例子: 实现 `LastPlacePruner`

We aim to optimize the `loss` and `alpha` hyperparameters for a stochastic gradient descent classifier (`SGDClassifier`) run on the sklearn iris dataset. We implement a pruner which terminates a trial at a certain step if it is in last place compared to completed trials at the same step. We begin considering pruning after a "warmup" of 1 training step and 5 completed trials. For demonstration purposes, we `print()` a diagnostic message from `prune` when it is about to return `True` (indicating pruning).

It may be important to note that the `SGDClassifier` score, as it is evaluated on a holdout set, decreases with enough training steps due to overfitting. This means that a trial could be pruned even if it had a favorable (high) value on a previous training set. After pruning, Optuna will take the intermediate value last reported as the value of the trial.

```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.linear_model import SGDClassifier

import optuna
```

(下页继续)

(续上页)

```
from optuna.pruners import BasePruner
from optuna.trial._state import TrialState


class LastPlacePruner(BasePruner):
    def __init__(self, warmup_steps, warmup_trials):
        self._warmup_steps = warmup_steps
        self._warmup_trials = warmup_trials

    def prune(self, study: "optuna.study.Study", trial: "optuna.trial.FrozenTrial") ->
↪ bool:
        # Get the latest score reported from this trial
        step = trial.last_step

        if step:  # trial.last_step == None when no scores have been reported yet
            this_score = trial.intermediate_values[step]

            # Get scores from other trials in the study reported at the same step
            completed_trials = study.get_trials(deepcopy=False, states=(TrialState.
↪COMPLETE,))
            other_scores = [
                t.intermediate_values[step]
                for t in completed_trials
                if step in t.intermediate_values
            ]
            other_scores = sorted(other_scores)

            # Prune if this trial at this step has a lower value than all completed␣
↪trials
            # at the same step. Note that steps will begin numbering at 0 in the␣
↪objective
            # function definition below.
            if step >= self._warmup_steps and len(other_scores) > self._warmup_trials:
                if this_score < other_scores[0]:
                    print(f"prune() True: Trial {trial.number}, Step {step}, Score
↪{this_score}")
                    return True

        return False
```

最后，让我们通过一个简单的超参数优化来确认该实现是正确的.

```
def objective(trial):
    iris = load_iris()
```

(下页继续)

(续上页)

```
    classes = np.unique(iris.target)
    X_train, X_valid, y_train, y_valid = train_test_split(
        iris.data, iris.target, train_size=100, test_size=50, random_state=0
    )

    loss = trial.suggest_categorical("loss", ["hinge", "log", "perceptron"])
    alpha = trial.suggest_float("alpha", 0.00001, 0.001, log=True)
    clf = SGDClassifier(loss=loss, alpha=alpha, random_state=0)
    score = 0

    for step in range(0, 5):
        clf.partial_fit(X_train, y_train, classes=classes)
        score = clf.score(X_valid, y_valid)

        trial.report(score, step)

        if trial.should_prune():
            raise optuna.TrialPruned()

    return score


pruner = LastPlacePruner(warmup_steps=1, warmup_trials=5)
study = optuna.create_study(direction="maximize", pruner=pruner)
study.optimize(objective, n_trials=50)
```

Out:

```
prune() True: Trial 6, Step 1, Score 0.84
prune() True: Trial 7, Step 1, Score 0.62
prune() True: Trial 10, Step 1, Score 0.62
prune() True: Trial 17, Step 1, Score 0.32
prune() True: Trial 26, Step 1, Score 0.62
prune() True: Trial 38, Step 4, Score 0.68
prune() True: Trial 42, Step 4, Score 0.68
prune() True: Trial 45, Step 4, Score 0.68
prune() True: Trial 49, Step 1, Score 0.62
```

Total running time of the script: (0 minutes 0.705 seconds)

Study.optimize 的回调

本教程展示了如何使用和实现一个用于*optimize()* 的 Optuna `Callback` .

`Callback` 会在 `objective` 每次求值以后被调用一次, 它接受*Study* 和*FrozenTrial* 作为参数并进行处理.

MLflowCallback 是个好例子.

在特定数量的 trial 被剪枝后终止优化

本例实现了一个有状态的回调函数. 如果特定数目的 trial 被剪枝了, 它将终止优化. 被剪枝的 trial 数是通过 `threshold` 来指定的.

```
import optuna


class StopWhenTrialKeepBeingPrunedCallback:
    def __init__(self, threshold: int):
        self.threshold = threshold
        self._consequtive_pruned_count = 0

    def __call__(self, study: optuna.study.Study, trial: optuna.trial.FrozenTrial) ->␣
↪None:
        if trial.state == optuna.trial.TrialState.PRUNED:
            self._consequtive_pruned_count += 1
        else:
            self._consequtive_pruned_count = 0

        if self._consequtive_pruned_count >= self.threshold:
            study.stop()
```

该目标函数会对除了前五个 trial 之外的所有 trial 进行剪枝 (`trial.number` 从 0 开始计数).

```
def objective(trial):
    if trial.number > 4:
        raise optuna.TrialPruned

    return trial.suggest_float("x", 0, 1)
```

在这里, 我们将阈值设置为 2: 优化过程会在一旦两个 trial 被剪枝后发生. 因此, 我们预期该 study 会在 7 个 trial 后停止.

```
import logging
import sys
```

(下页继续)

(续上页)

```
# Add stream handler of stdout to show the messages
optuna.logging.get_logger("optuna").addHandler(logging.StreamHandler(sys.stdout))

study_stop_cb = StopWhenTrialKeepBeingPrunedCallback(2)
study = optuna.create_study()
study.optimize(objective, n_trials=10, callbacks=[study_stop_cb])
```

Out:

```
A new study created in memory with name: no-name-df3417c5-834d-42a2-9978-20adf440a28d
Trial 0 finished with value: 0.7727512815488775 and parameters: {'x': 0.
↪7727512815488775}. Best is trial 0 with value: 0.7727512815488775.
Trial 1 finished with value: 0.5717267978885823 and parameters: {'x': 0.
↪5717267978885823}. Best is trial 1 with value: 0.5717267978885823.
Trial 2 finished with value: 0.8552077108941241 and parameters: {'x': 0.
↪8552077108941241}. Best is trial 1 with value: 0.5717267978885823.
Trial 3 finished with value: 0.9001926292198051 and parameters: {'x': 0.
↪9001926292198051}. Best is trial 1 with value: 0.5717267978885823.
Trial 4 finished with value: 0.7865904524969077 and parameters: {'x': 0.
↪7865904524969077}. Best is trial 1 with value: 0.5717267978885823.
Trial 5 pruned.
Trial 6 pruned.
```

从上面的日志中可以看出, study 如期在 7 个 trial 之后停止了.

Total running time of the script: (0 minutes 0.008 seconds)

手动指定超参数

你自然有一些特定的超参数集要先尝试, 比如初始学习率和叶子数量. 另外, 也有可能在让 Optuna 找到更好的超参数集之前，你已经尝试过一些集合.

Optuna 提供两个 API 以应对这种场景:

1. 将这些超参数集合传递过去并让 Optuna 对其求值 - *enqueue_trial()*
2. 将这些集合的结果标记为已完成的 `Trials` - *add_trial()*

第一个场景: 让 Optuna 对你的超参数求值

在这个场景中, 我们假设你已经有了一些开箱即用的超参数但是还没有对他们进行过求值, 你决定用 Optuna 来找到更好的超参数.

Optuna 有一个 API *optuna.study.Study.enqueue_trial()* , 它允许你将这些超参数传入 Optuna, Optuna 会对他们求值.

这部分将结合 LightGBM 向你介绍如何使用这些 API.

```
import lightgbm as lgb
import numpy as np
import sklearn.datasets
import sklearn.metrics
from sklearn.model_selection import train_test_split

import optuna
```

定义目标函数.

```
def objective(trial):
    data, target = sklearn.datasets.load_breast_cancer(return_X_y=True)
    train_x, valid_x, train_y, valid_y = train_test_split(data, target, test_size=0.
↪25)
    dtrain = lgb.Dataset(train_x, label=train_y)
    dvalid = lgb.Dataset(valid_x, label=valid_y)

    param = {
        "objective": "binary",
        "metric": "auc",
        "verbosity": -1,
        "boosting_type": "gbdt",
        "bagging_fraction": min(trial.suggest_float("bagging_fraction", 0.4, 1.0 + 1e-
↪12), 1),
        "bagging_freq": trial.suggest_int("bagging_freq", 0, 7),
        "min_child_samples": trial.suggest_int("min_child_samples", 5, 100),
    }

    # Add a callback for pruning.
    pruning_callback = optuna.integration.LightGBMPruningCallback(trial, "auc")
    gbm = lgb.train(
        param, dtrain, valid_sets=[dvalid], verbose_eval=False, callbacks=[pruning_
↪callback]
    )
```

(下页继续)

(续上页)

```
    preds = gbm.predict(valid_x)
    pred_labels = np.rint(preds)
    accuracy = sklearn.metrics.accuracy_score(valid_y, pred_labels)
    return accuracy
```

然后构建 Study 用于超参数优化.

```
study = optuna.create_study(direction="maximize", pruner=optuna.pruners.
↪MedianPruner())
```

在这里, 我们让 Optuna 对一些带有更大的 "bagging_fraq" 和默认值的集合求值.

```
study.enqueue_trial(
    {
        "bagging_fraction": 1.0,
        "bagging_freq": 0,
        "min_child_samples": 20,
    }
)

study.enqueue_trial(
    {
        "bagging_fraction": 0.75,
        "bagging_freq": 5,
        "min_child_samples": 20,
    }
)

import logging
import sys

# Add stream handler of stdout to show the messages to see Optuna works expectedly.
optuna.logging.get_logger("optuna").addHandler(logging.StreamHandler(sys.stdout))
study.optimize(objective, n_trials=100, timeout=600)
```

Out:

```
/home/docs/checkouts/readthedocs.org/user_builds/optuna-zh-cn/checkouts/stable/
↪tutorial/20_recipes/008_specify_params.py:81: ExperimentalWarning:

enqueue_trial is experimental (supported from v1.2.0). The interface can change in␣
↪the future.

/home/docs/checkouts/readthedocs.org/user_builds/optuna-zh-cn/envs/stable/lib/python3.
↪7/site-packages/optuna/study.py:857: ExperimentalWarning:
```

(下页继续)

(续上页)

```
create_trial is experimental (supported from v2.0.0). The interface can change in the␣
↪future.

/home/docs/checkouts/readthedocs.org/user_builds/optuna-zh-cn/envs/stable/lib/python3.
↪7/site-packages/optuna/study.py:857: ExperimentalWarning:

add_trial is experimental (supported from v2.0.0). The interface can change in the␣
↪future.

/home/docs/checkouts/readthedocs.org/user_builds/optuna-zh-cn/checkouts/stable/
↪tutorial/20_recipes/008_specify_params.py:89: ExperimentalWarning:

enqueue_trial is experimental (supported from v1.2.0). The interface can change in␣
↪the future.

/home/docs/checkouts/readthedocs.org/user_builds/optuna-zh-cn/envs/stable/lib/python3.
↪7/site-packages/lightgbm/engine.py:239: UserWarning:

'verbose_eval' argument is deprecated and will be removed in a future release of␣
↪LightGBM. Pass 'log_evaluation()' callback via 'callbacks' argument instead.

Trial 0 finished with value: 0.972027972027972 and parameters: {'bagging_fraction': 1.
↪0, 'bagging_freq': 0, 'min_child_samples': 20}. Best is trial 0 with value: 0.
↪972027972027972.
/home/docs/checkouts/readthedocs.org/user_builds/optuna-zh-cn/envs/stable/lib/python3.
↪7/site-packages/lightgbm/engine.py:239: UserWarning:

'verbose_eval' argument is deprecated and will be removed in a future release of␣
↪LightGBM. Pass 'log_evaluation()' callback via 'callbacks' argument instead.

Trial 1 finished with value: 0.965034965034965 and parameters: {'bagging_fraction': 0.
↪75, 'bagging_freq': 5, 'min_child_samples': 20}. Best is trial 0 with value: 0.
↪972027972027972.
/home/docs/checkouts/readthedocs.org/user_builds/optuna-zh-cn/envs/stable/lib/python3.
↪7/site-packages/lightgbm/engine.py:239: UserWarning:

'verbose_eval' argument is deprecated and will be removed in a future release of␣
↪LightGBM. Pass 'log_evaluation()' callback via 'callbacks' argument instead.

Trial 2 finished with value: 0.986013986013986 and parameters: {'bagging_fraction': 0.
↪4634984818580007, 'bagging_freq': 4, 'min_child_samples': 43}. Best is trial 2 with␣
↪value: 0.986013986013986.
```

(下页继续)

(续上页)

```
/home/docs/checkouts/readthedocs.org/user_builds/optuna-zh-cn/envs/stable/lib/python3.
↪7/site-packages/lightgbm/engine.py:239: UserWarning:

'verbose_eval' argument is deprecated and will be removed in a future release of␣
↪LightGBM. Pass 'log_evaluation()' callback via 'callbacks' argument instead.

Trial 3 finished with value: 0.986013986013986 and parameters: {'bagging_fraction': 0.
↪7498262874277628, 'bagging_freq': 7, 'min_child_samples': 48}. Best is trial 2 with␣
↪value: 0.986013986013986.
/home/docs/checkouts/readthedocs.org/user_builds/optuna-zh-cn/envs/stable/lib/python3.
↪7/site-packages/lightgbm/engine.py:239: UserWarning:

'verbose_eval' argument is deprecated and will be removed in a future release of␣
↪LightGBM. Pass 'log_evaluation()' callback via 'callbacks' argument instead.

Trial 4 finished with value: 0.9440559440559441 and parameters: {'bagging_fraction':␣
↪0.7947319219659837, 'bagging_freq': 4, 'min_child_samples': 93}. Best is trial 2␣
↪with value: 0.986013986013986.
/home/docs/checkouts/readthedocs.org/user_builds/optuna-zh-cn/envs/stable/lib/python3.
↪7/site-packages/lightgbm/engine.py:239: UserWarning:

'verbose_eval' argument is deprecated and will be removed in a future release of␣
↪LightGBM. Pass 'log_evaluation()' callback via 'callbacks' argument instead.

Trial 5 pruned. Trial was pruned at iteration 0.
Trial 6 pruned. Trial was pruned at iteration 0.
Trial 7 pruned. Trial was pruned at iteration 0.
Trial 8 pruned. Trial was pruned at iteration 0.
Trial 9 pruned. Trial was pruned at iteration 0.
Trial 10 pruned. Trial was pruned at iteration 0.
Trial 11 pruned. Trial was pruned at iteration 0.
Trial 12 pruned. Trial was pruned at iteration 0.
Trial 13 pruned. Trial was pruned at iteration 0.
Trial 14 pruned. Trial was pruned at iteration 0.
Trial 15 finished with value: 0.972027972027972 and parameters: {'bagging_fraction':␣
↪0.661493861887769, 'bagging_freq': 1, 'min_child_samples': 52}. Best is trial 2␣
↪with value: 0.986013986013986.
/home/docs/checkouts/readthedocs.org/user_builds/optuna-zh-cn/envs/stable/lib/python3.
↪7/site-packages/lightgbm/engine.py:239: UserWarning:

'verbose_eval' argument is deprecated and will be removed in a future release of␣
↪LightGBM. Pass 'log_evaluation()' callback via 'callbacks' argument instead.
```

(下页继续)

(续上页)

```
Trial 16 pruned. Trial was pruned at iteration 0.
Trial 17 pruned. Trial was pruned at iteration 0.
Trial 18 pruned. Trial was pruned at iteration 0.
Trial 19 pruned. Trial was pruned at iteration 0.
Trial 20 pruned. Trial was pruned at iteration 0.
Trial 21 pruned. Trial was pruned at iteration 0.
Trial 22 pruned. Trial was pruned at iteration 0.
Trial 23 pruned. Trial was pruned at iteration 0.
Trial 24 pruned. Trial was pruned at iteration 0.
Trial 25 pruned. Trial was pruned at iteration 0.
Trial 26 pruned. Trial was pruned at iteration 0.
Trial 27 pruned. Trial was pruned at iteration 0.
Trial 28 pruned. Trial was pruned at iteration 0.
Trial 29 pruned. Trial was pruned at iteration 0.
Trial 30 pruned. Trial was pruned at iteration 0.
Trial 31 pruned. Trial was pruned at iteration 0.
Trial 32 pruned. Trial was pruned at iteration 0.
Trial 33 pruned. Trial was pruned at iteration 0.
Trial 34 pruned. Trial was pruned at iteration 0.
Trial 35 pruned. Trial was pruned at iteration 0.
Trial 36 pruned. Trial was pruned at iteration 0.
Trial 37 finished with value: 0.972027972027972 and parameters: {'bagging_fraction':␣
↪0.7395765217168897, 'bagging_freq': 3, 'min_child_samples': 23}. Best is trial 2␣
↪with value: 0.986013986013986.
/home/docs/checkouts/readthedocs.org/user_builds/optuna-zh-cn/envs/stable/lib/python3.
↪7/site-packages/lightgbm/engine.py:239: UserWarning:

'verbose_eval' argument is deprecated and will be removed in a future release of␣
↪LightGBM. Pass 'log_evaluation()' callback via 'callbacks' argument instead.

Trial 38 pruned. Trial was pruned at iteration 0.
Trial 39 pruned. Trial was pruned at iteration 0.
Trial 40 pruned. Trial was pruned at iteration 0.
Trial 41 pruned. Trial was pruned at iteration 0.
Trial 42 pruned. Trial was pruned at iteration 0.
Trial 43 pruned. Trial was pruned at iteration 0.
Trial 44 pruned. Trial was pruned at iteration 0.
Trial 45 pruned. Trial was pruned at iteration 0.
Trial 46 pruned. Trial was pruned at iteration 0.
Trial 47 pruned. Trial was pruned at iteration 0.
Trial 48 pruned. Trial was pruned at iteration 0.
Trial 49 pruned. Trial was pruned at iteration 0.
Trial 50 pruned. Trial was pruned at iteration 0.
```

(下页继续)

(续上页)

```
Trial 51 pruned. Trial was pruned at iteration 0.
Trial 52 pruned. Trial was pruned at iteration 0.
Trial 53 pruned. Trial was pruned at iteration 0.
Trial 54 pruned. Trial was pruned at iteration 0.
Trial 55 finished with value: 0.986013986013986 and parameters: {'bagging_fraction':␣
↪0.8520330576045076, 'bagging_freq': 7, 'min_child_samples': 15}. Best is trial 2␣
↪with value: 0.986013986013986.
/home/docs/checkouts/readthedocs.org/user_builds/optuna-zh-cn/envs/stable/lib/python3.
↪7/site-packages/lightgbm/engine.py:239: UserWarning:

'verbose_eval' argument is deprecated and will be removed in a future release of␣
↪LightGBM. Pass 'log_evaluation()' callback via 'callbacks' argument instead.

Trial 56 pruned. Trial was pruned at iteration 0.
Trial 57 pruned. Trial was pruned at iteration 0.
Trial 58 pruned. Trial was pruned at iteration 0.
Trial 59 pruned. Trial was pruned at iteration 0.
Trial 60 pruned. Trial was pruned at iteration 0.
Trial 61 pruned. Trial was pruned at iteration 0.
Trial 62 pruned. Trial was pruned at iteration 0.
Trial 63 finished with value: 0.972027972027972 and parameters: {'bagging_fraction':␣
↪0.8745866104487076, 'bagging_freq': 5, 'min_child_samples': 20}. Best is trial 2␣
↪with value: 0.986013986013986.
/home/docs/checkouts/readthedocs.org/user_builds/optuna-zh-cn/envs/stable/lib/python3.
↪7/site-packages/lightgbm/engine.py:239: UserWarning:

'verbose_eval' argument is deprecated and will be removed in a future release of␣
↪LightGBM. Pass 'log_evaluation()' callback via 'callbacks' argument instead.

Trial 64 finished with value: 0.965034965034965 and parameters: {'bagging_fraction':␣
↪0.8344316889970204, 'bagging_freq': 5, 'min_child_samples': 9}. Best is trial 2␣
↪with value: 0.986013986013986.
/home/docs/checkouts/readthedocs.org/user_builds/optuna-zh-cn/envs/stable/lib/python3.
↪7/site-packages/lightgbm/engine.py:239: UserWarning:

'verbose_eval' argument is deprecated and will be removed in a future release of␣
↪LightGBM. Pass 'log_evaluation()' callback via 'callbacks' argument instead.

Trial 65 pruned. Trial was pruned at iteration 0.
Trial 66 pruned. Trial was pruned at iteration 0.
Trial 67 pruned. Trial was pruned at iteration 0.
Trial 68 pruned. Trial was pruned at iteration 0.
Trial 69 pruned. Trial was pruned at iteration 0.
```

(下页继续)

(续上页)

```
Trial 70 pruned. Trial was pruned at iteration 0.
Trial 71 pruned. Trial was pruned at iteration 0.
Trial 72 pruned. Trial was pruned at iteration 0.
Trial 73 pruned. Trial was pruned at iteration 0.
Trial 74 pruned. Trial was pruned at iteration 0.
Trial 75 pruned. Trial was pruned at iteration 0.
Trial 76 pruned. Trial was pruned at iteration 0.
Trial 77 pruned. Trial was pruned at iteration 0.
Trial 78 pruned. Trial was pruned at iteration 0.
Trial 79 pruned. Trial was pruned at iteration 0.
Trial 80 pruned. Trial was pruned at iteration 0.
Trial 81 pruned. Trial was pruned at iteration 0.
Trial 82 pruned. Trial was pruned at iteration 0.
Trial 83 pruned. Trial was pruned at iteration 0.
Trial 84 pruned. Trial was pruned at iteration 0.
Trial 85 pruned. Trial was pruned at iteration 0.
Trial 86 pruned. Trial was pruned at iteration 0.
Trial 87 pruned. Trial was pruned at iteration 0.
Trial 88 pruned. Trial was pruned at iteration 0.
Trial 89 pruned. Trial was pruned at iteration 0.
Trial 90 pruned. Trial was pruned at iteration 0.
Trial 91 pruned. Trial was pruned at iteration 0.
Trial 92 pruned. Trial was pruned at iteration 0.
Trial 93 pruned. Trial was pruned at iteration 0.
Trial 94 pruned. Trial was pruned at iteration 0.
Trial 95 pruned. Trial was pruned at iteration 0.
Trial 96 pruned. Trial was pruned at iteration 0.
Trial 97 pruned. Trial was pruned at iteration 0.
Trial 98 pruned. Trial was pruned at iteration 11.
Trial 99 pruned. Trial was pruned at iteration 0.
```

第二个场景: 让 Optuna 利用已经求值过的超参数.

在这个场景中, 我们假设你已经有了一些开箱即用的超参数, 而且你已经对他们进行求值过, 但是其结果并不让人满意. 因此你考虑使用 Optuna 来找到更好的超参数.

Optuna 有一个 API *optuna.study.Study.add_trial()* , 它让你向 Optuna 注册这些结果, 之后 Optuna 会在进行超参数采样的时候将它们考虑进去.

在本部分中, `objective` 和第一个场景中一样.

```
study = optuna.create_study(direction="maximize", pruner=optuna.pruners.
↪MedianPruner())
```

(下页继续)

(续上页)

```
study.add_trial(
    optuna.trial.create_trial(
        params={
            "bagging_fraction": 1.0,
            "bagging_freq": 0,
            "min_child_samples": 20,
        },
        distributions={
            "bagging_fraction": optuna.distributions.UniformDistribution(0.4, 1.0 +␣
↪1e-12),
            "bagging_freq": optuna.distributions.IntUniformDistribution(0, 7),
            "min_child_samples": optuna.distributions.IntUniformDistribution(5, 100),
        },
        value=0.94,
    )
)
study.add_trial(
    optuna.trial.create_trial(
        params={
            "bagging_fraction": 0.75,
            "bagging_freq": 5,
            "min_child_samples": 20,
        },
        distributions={
            "bagging_fraction": optuna.distributions.UniformDistribution(0.4, 1.0 +␣
↪1e-12),
            "bagging_freq": optuna.distributions.IntUniformDistribution(0, 7),
            "min_child_samples": optuna.distributions.IntUniformDistribution(5, 100),
        },
        value=0.95,
    )
)
study.optimize(objective, n_trials=100, timeout=600)
```

Out:

```
A new study created in memory with name: no-name-ebcef9a6-14c7-49dc-8b5d-d516934a1c81
/home/docs/checkouts/readthedocs.org/user_builds/optuna-zh-cn/checkouts/stable/
↪tutorial/20_recipes/008_specify_params.py:126: ExperimentalWarning:

create_trial is experimental (supported from v2.0.0). The interface can change in the␣
↪future.

/home/docs/checkouts/readthedocs.org/user_builds/optuna-zh-cn/checkouts/stable/
↪tutorial/20_recipes/008_specify_params.py:126: ExperimentalWarning:
```

(下页继续)

(续上页)

```
add_trial is experimental (supported from v2.0.0). The interface can change in the␣
↪future.

/home/docs/checkouts/readthedocs.org/user_builds/optuna-zh-cn/checkouts/stable/
↪tutorial/20_recipes/008_specify_params.py:141: ExperimentalWarning:

create_trial is experimental (supported from v2.0.0). The interface can change in the␣
↪future.

/home/docs/checkouts/readthedocs.org/user_builds/optuna-zh-cn/checkouts/stable/
↪tutorial/20_recipes/008_specify_params.py:141: ExperimentalWarning:

add_trial is experimental (supported from v2.0.0). The interface can change in the␣
↪future.

/home/docs/checkouts/readthedocs.org/user_builds/optuna-zh-cn/envs/stable/lib/python3.
↪7/site-packages/lightgbm/engine.py:239: UserWarning:

'verbose_eval' argument is deprecated and will be removed in a future release of␣
↪LightGBM. Pass 'log_evaluation()' callback via 'callbacks' argument instead.

Trial 2 finished with value: 0.9440559440559441 and parameters: {'bagging_fraction':␣
↪0.5921681795227542, 'bagging_freq': 7, 'min_child_samples': 13}. Best is trial 1␣
↪with value: 0.95.
/home/docs/checkouts/readthedocs.org/user_builds/optuna-zh-cn/envs/stable/lib/python3.
↪7/site-packages/lightgbm/engine.py:239: UserWarning:

'verbose_eval' argument is deprecated and will be removed in a future release of␣
↪LightGBM. Pass 'log_evaluation()' callback via 'callbacks' argument instead.

Trial 3 finished with value: 0.986013986013986 and parameters: {'bagging_fraction': 0.
↪9694304841802756, 'bagging_freq': 6, 'min_child_samples': 99}. Best is trial 3 with␣
↪value: 0.986013986013986.
/home/docs/checkouts/readthedocs.org/user_builds/optuna-zh-cn/envs/stable/lib/python3.
↪7/site-packages/lightgbm/engine.py:239: UserWarning:

'verbose_eval' argument is deprecated and will be removed in a future release of␣
↪LightGBM. Pass 'log_evaluation()' callback via 'callbacks' argument instead.

Trial 4 finished with value: 0.951048951048951 and parameters: {'bagging_fraction': 0.
↪9482649706012053, 'bagging_freq': 6, 'min_child_samples': 5}. Best is trial 3 with␣
↪value: 0.986013986013986.
```

(下页继续)

(续上页)

```
/home/docs/checkouts/readthedocs.org/user_builds/optuna-zh-cn/envs/stable/lib/python3.
↪7/site-packages/lightgbm/engine.py:239: UserWarning:

'verbose_eval' argument is deprecated and will be removed in a future release of␣
↪LightGBM. Pass 'log_evaluation()' callback via 'callbacks' argument instead.

Trial 5 pruned. Trial was pruned at iteration 0.
Trial 6 pruned. Trial was pruned at iteration 19.
Trial 7 pruned. Trial was pruned at iteration 0.
Trial 8 pruned. Trial was pruned at iteration 0.
Trial 9 pruned. Trial was pruned at iteration 0.
Trial 10 pruned. Trial was pruned at iteration 0.
Trial 11 pruned. Trial was pruned at iteration 0.
Trial 12 pruned. Trial was pruned at iteration 0.
Trial 13 finished with value: 0.9790209790209791 and parameters: {'bagging_fraction':␣
↪0.9903914325997821, 'bagging_freq': 6, 'min_child_samples': 36}. Best is trial 3␣
↪with value: 0.986013986013986.
/home/docs/checkouts/readthedocs.org/user_builds/optuna-zh-cn/envs/stable/lib/python3.
↪7/site-packages/lightgbm/engine.py:239: UserWarning:

'verbose_eval' argument is deprecated and will be removed in a future release of␣
↪LightGBM. Pass 'log_evaluation()' callback via 'callbacks' argument instead.

Trial 14 pruned. Trial was pruned at iteration 0.
Trial 15 pruned. Trial was pruned at iteration 3.
Trial 16 pruned. Trial was pruned at iteration 0.
Trial 17 pruned. Trial was pruned at iteration 0.
Trial 18 pruned. Trial was pruned at iteration 0.
Trial 19 pruned. Trial was pruned at iteration 0.
Trial 20 pruned. Trial was pruned at iteration 0.
Trial 21 pruned. Trial was pruned at iteration 0.
Trial 22 pruned. Trial was pruned at iteration 0.
Trial 23 pruned. Trial was pruned at iteration 0.
Trial 24 pruned. Trial was pruned at iteration 0.
Trial 25 pruned. Trial was pruned at iteration 0.
Trial 26 finished with value: 0.9790209790209791 and parameters: {'bagging_fraction':␣
↪0.8050779265956571, 'bagging_freq': 7, 'min_child_samples': 13}. Best is trial 3␣
↪with value: 0.986013986013986.
/home/docs/checkouts/readthedocs.org/user_builds/optuna-zh-cn/envs/stable/lib/python3.
↪7/site-packages/lightgbm/engine.py:239: UserWarning:

'verbose_eval' argument is deprecated and will be removed in a future release of␣
↪LightGBM. Pass 'log_evaluation()' callback via 'callbacks' argument instead.
```

(下页继续)

(续上页)

```
Trial 27 pruned. Trial was pruned at iteration 0.
Trial 28 pruned. Trial was pruned at iteration 0.
Trial 29 pruned. Trial was pruned at iteration 0.
Trial 30 pruned. Trial was pruned at iteration 0.
Trial 31 pruned. Trial was pruned at iteration 0.
Trial 32 pruned. Trial was pruned at iteration 70.
Trial 33 pruned. Trial was pruned at iteration 0.
Trial 34 pruned. Trial was pruned at iteration 0.
Trial 35 pruned. Trial was pruned at iteration 0.
Trial 36 pruned. Trial was pruned at iteration 0.
Trial 37 pruned. Trial was pruned at iteration 1.
Trial 38 pruned. Trial was pruned at iteration 0.
Trial 39 pruned. Trial was pruned at iteration 0.
Trial 40 pruned. Trial was pruned at iteration 0.
Trial 41 pruned. Trial was pruned at iteration 41.
Trial 42 pruned. Trial was pruned at iteration 0.
Trial 43 pruned. Trial was pruned at iteration 0.
Trial 44 finished with value: 0.9790209790209791 and parameters: {'bagging_fraction':␣
↪0.7759592345407211, 'bagging_freq': 7, 'min_child_samples': 25}. Best is trial 3␣
↪with value: 0.986013986013986.
/home/docs/checkouts/readthedocs.org/user_builds/optuna-zh-cn/envs/stable/lib/python3.
↪7/site-packages/lightgbm/engine.py:239: UserWarning:

'verbose_eval' argument is deprecated and will be removed in a future release of␣
↪LightGBM. Pass 'log_evaluation()' callback via 'callbacks' argument instead.

Trial 45 pruned. Trial was pruned at iteration 1.
Trial 46 pruned. Trial was pruned at iteration 0.
Trial 47 pruned. Trial was pruned at iteration 0.
Trial 48 pruned. Trial was pruned at iteration 1.
Trial 49 pruned. Trial was pruned at iteration 0.
Trial 50 pruned. Trial was pruned at iteration 0.
Trial 51 pruned. Trial was pruned at iteration 0.
Trial 52 pruned. Trial was pruned at iteration 0.
Trial 53 pruned. Trial was pruned at iteration 0.
Trial 54 pruned. Trial was pruned at iteration 0.
Trial 55 pruned. Trial was pruned at iteration 0.
Trial 56 pruned. Trial was pruned at iteration 0.
Trial 57 pruned. Trial was pruned at iteration 1.
Trial 58 pruned. Trial was pruned at iteration 0.
Trial 59 pruned. Trial was pruned at iteration 0.
Trial 60 pruned. Trial was pruned at iteration 0.
```

(下页继续)

(续上页)

```
Trial 61 pruned. Trial was pruned at iteration 0.
Trial 62 pruned. Trial was pruned at iteration 0.
Trial 63 pruned. Trial was pruned at iteration 0.
Trial 64 pruned. Trial was pruned at iteration 0.
Trial 65 pruned. Trial was pruned at iteration 0.
Trial 66 pruned. Trial was pruned at iteration 3.
Trial 67 pruned. Trial was pruned at iteration 0.
Trial 68 pruned. Trial was pruned at iteration 0.
Trial 69 pruned. Trial was pruned at iteration 0.
Trial 70 pruned. Trial was pruned at iteration 0.
Trial 71 finished with value: 0.972027972027972 and parameters: {'bagging_fraction':␣
↪0.9970935811879088, 'bagging_freq': 2, 'min_child_samples': 12}. Best is trial 3␣
↪with value: 0.986013986013986.
/home/docs/checkouts/readthedocs.org/user_builds/optuna-zh-cn/envs/stable/lib/python3.
↪7/site-packages/lightgbm/engine.py:239: UserWarning:

'verbose_eval' argument is deprecated and will be removed in a future release of␣
↪LightGBM. Pass 'log_evaluation()' callback via 'callbacks' argument instead.

Trial 72 pruned. Trial was pruned at iteration 1.
Trial 73 pruned. Trial was pruned at iteration 0.
Trial 74 finished with value: 0.993006993006993 and parameters: {'bagging_fraction':␣
↪0.9979877428860009, 'bagging_freq': 1, 'min_child_samples': 65}. Best is trial 74␣
↪with value: 0.993006993006993.
/home/docs/checkouts/readthedocs.org/user_builds/optuna-zh-cn/envs/stable/lib/python3.
↪7/site-packages/lightgbm/engine.py:239: UserWarning:

'verbose_eval' argument is deprecated and will be removed in a future release of␣
↪LightGBM. Pass 'log_evaluation()' callback via 'callbacks' argument instead.

Trial 75 pruned. Trial was pruned at iteration 0.
Trial 76 pruned. Trial was pruned at iteration 0.
Trial 77 pruned. Trial was pruned at iteration 0.
Trial 78 pruned. Trial was pruned at iteration 0.
Trial 79 pruned. Trial was pruned at iteration 0.
Trial 80 pruned. Trial was pruned at iteration 3.
Trial 81 pruned. Trial was pruned at iteration 0.
Trial 82 pruned. Trial was pruned at iteration 0.
Trial 83 pruned. Trial was pruned at iteration 1.
Trial 84 pruned. Trial was pruned at iteration 0.
Trial 85 pruned. Trial was pruned at iteration 1.
Trial 86 pruned. Trial was pruned at iteration 0.
Trial 87 pruned. Trial was pruned at iteration 0.
```

(下页继续)

(续上页)

```
Trial 88 pruned. Trial was pruned at iteration 0.
Trial 89 pruned. Trial was pruned at iteration 0.
Trial 90 pruned. Trial was pruned at iteration 0.
Trial 91 pruned. Trial was pruned at iteration 0.
Trial 92 pruned. Trial was pruned at iteration 0.
Trial 93 pruned. Trial was pruned at iteration 2.
Trial 94 pruned. Trial was pruned at iteration 2.
Trial 95 pruned. Trial was pruned at iteration 0.
Trial 96 pruned. Trial was pruned at iteration 1.
Trial 97 pruned. Trial was pruned at iteration 0.
Trial 98 pruned. Trial was pruned at iteration 1.
Trial 99 pruned. Trial was pruned at iteration 0.
Trial 100 pruned. Trial was pruned at iteration 0.
Trial 101 pruned. Trial was pruned at iteration 1.
```

Total running time of the script: (0 minutes 6.512 seconds)

Ask-and-Tell 接口

Optuna 带有 *Ask-and-Tell* 接口, 它为超参数优化提供了一个更灵活的接口. 本教程将展示三种可以用到 ask-and-tell 接口的情况.

- 只进行最小改动, 就可将 *Optuna* 应用到一个现存的优化问题上
- 定义-运行 *(Define-and-Run)*
- 批优化

只进行最小改动, 就可将 Optuna 应用到一个现存的优化问题上

考虑一个传统的有监督分类问题: 你想最大化验证准确率. 所以, 你训练了一个 *LogisticRegression* 作为简单模型.

```
import numpy as np
from sklearn.datasets import make_classification
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

import optuna


X, y = make_classification(n_features=10)
X_train, X_test, y_train, y_test = train_test_split(X, y)
```

(下页继续)

(续上页)

```
C = 0.01
clf = LogisticRegression(C=C)
clf.fit(X_train, y_train)
val_accuracy = clf.score(X_test, y_test)  # the objective
```

然后你试图通过 Optuna 来优化超参数 `C` 和 `solver`. 当你简单地引入 Optuna 后, 你定义了一个 `objective` 函数, 它接受 `trial` 并且调用 `trial` 的 `suggest_*` 方法来采样超参数:

```
def objective(trial):
    X, y = make_classification(n_features=10)
    X_train, X_test, y_train, y_test = train_test_split(X, y)

    C = trial.suggest_loguniform("C", 1e-7, 10.0)
    solver = trial.suggest_categorical("solver", ("lbfgs", "saga"))

    clf = LogisticRegression(C=C, solver=solver)
    clf.fit(X_train, y_train)
    val_accuracy = clf.score(X_test, y_test)

    return val_accuracy


study = optuna.create_study(direction="maximize")
study.optimize(objective, n_trials=10)
```

这个接口并不灵活. 比如, 如果 `objective` 需要不同于 `trial` 的额外参数, 你就需要顶一个类, 就像 How to define objective functions that have own arguments? 里做的那样. 而 ask-and-tell 接口提供了更加灵活的语法来优化超参数. 下面的例子等同于上面的代码块.

```
study = optuna.create_study(direction="maximize")

n_trials = 10
for _ in range(n_trials):
    trial = study.ask()  # `trial` is a `Trial` and not a `FrozenTrial`.

    C = trial.suggest_loguniform("C", 1e-7, 10.0)
    solver = trial.suggest_categorical("solver", ("lbfgs", "saga"))

    clf = LogisticRegression(C=C, solver=solver)
    clf.fit(X_train, y_train)
    val_accuracy = clf.score(X_test, y_test)

    study.tell(trial, val_accuracy)  # tell the pair of trial and objective value
```

主要区别是这里用了两个方法*optuna.study.Study.ask()* 和*optuna.study.Study.tell()*. *optuna.study.Study.ask()* 创建了一个 trial, 它可以采样超参数, 而*optuna.study.Study.tell()* 通过传递 `trial` 和一个目标函数值完成这个 trial. 你可以在没有 `objective` 函数的情况下对你的原始代码应用 Optuna 的超参数优化.

如果你想用 pruner 让你的优化变得更快, 你需要显式地将 trial 的状态传入到*optuna.study.Study.tell()* 的参数中, 就像下面这样:

```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import train_test_split

import optuna


X, y = load_iris(return_X_y=True)
X_train, X_valid, y_train, y_valid = train_test_split(X, y)
classes = np.unique(y)
n_train_iter = 100

# define study with hyperband pruner.
study = optuna.create_study(
    direction="maximize",
    pruner=optuna.pruners.HyperbandPruner(
        min_resource=1, max_resource=n_train_iter, reduction_factor=3
    ),
)

for _ in range(20):
    trial = study.ask()

    alpha = trial.suggest_uniform("alpha", 0.0, 1.0)

    clf = SGDClassifier(alpha=alpha)
    pruned_trial = False

    for step in range(n_train_iter):
        clf.partial_fit(X_train, y_train, classes=classes)

        intermediate_value = clf.score(X_valid, y_valid)
        trial.report(intermediate_value, step)

        if trial.should_prune():
            pruned_trial = True
```

(下页继续)

(续上页)

```
            break

if pruned_trial:
    study.tell(trial, state=optuna.trial.TrialState.PRUNED)  # tell the pruned state
else:
    score = clf.score(X_valid, y_valid)
    study.tell(trial, score)  # tell objective value
```

备注: *optuna.study.Study.tell()* 可以接受一个 trial number 而不是 trial 对象本身. `study.tell(trial.number, y)` 等价于 `study.tell(trial, y)`.

定义-运行 (Define-and-Run)

ask-and-tell 接口同时支持 *define-by-run* 和 *define-and-run* API. 在上面 define-by-run 的例子之外, 本部分展示 *define-and-run* 的 API.

在调用*optuna.study.Study.ask()* 方法之前为 define-and-run API 定义超参数分布. 例如,

```
distributions = {
    "C": optuna.distributions.LogUniformDistribution(1e-7, 10.0),
    "solver": optuna.distributions.CategoricalDistribution(("lbfgs", "saga")),
}
```

在每次调用时, 将 `distributions` 传递给*optuna.study.Study.ask()* 方法. 返回的 `trial` 里将包含建议的超参数.

```
study = optuna.create_study(direction="maximize")
n_trials = 10
for _ in range(n_trials):
    trial = study.ask(distributions)  # pass the pre-defined distributions.

    # two hyperparameters are already sampled from the pre-defined distributions
    C = trial.params["C"]
    solver = trial.params["solver"]

    clf = LogisticRegression(C=C, solver=solver)
    clf.fit(X_train, y_train)
    val_accuracy = clf.score(X_test, y_test)

    study.tell(trial, val_accuracy)
```

批优化

为了更快地完成优化, ask-and-tell 接口使我们可以优化一个批目标函数. 比如, 并行求值, 向量操作等.

下面这个目标函数接受批超参数 `xs` 而不是一个单独的超参数, 并对整个向量计算目标函数.

```
def batched_objective(xs: np.ndarray):
    return xs ** 2 + 1
```

在下面的例子中, 一批中包含的超参数个数是 10, 而 `batched_objective` 进行了三次求值. 因此, trial 的个数是 30. 注意, 在批求值以后, 你需要存储 `trial_ids` 或者 `trial` 才能调用*optuna.study.Study.tell()* 方法

```
batch_size = 10
study = optuna.create_study()

for _ in range(3):

    # create batch
    trial_ids = []
    samples = []
    for _ in range(batch_size):
        trial = study.ask()
        trial_ids.append(trial.number)
        x = trial.suggest_int("x", -10, 10)
        samples.append(x)

    # evaluate batched objective
    samples = np.array(samples)
    objectives = batched_objective(samples)

    # finish all trials in the batch
    for trial_id, objective in zip(trial_ids, objectives):
        study.tell(trial_id, objective)
```

Total running time of the script: (0 minutes 0.169 seconds)

重新使用最佳值

有时候, 在完成超参数优化以后, 你可能要用最佳的超参数对目标函数重新求值.

比如,

- 你已经用 Optuna 发现了好的超参数, 并且想用已经发现的最佳超参数来运行一个类似的 *objective* 函数以进一步分析结果, 或者

- 为节省时间, 你已经用 Optuna 来优化了一个部分数据集. 在超参数调整以后, 你想在整个数据集上用你找到的最佳超参数来训练模型.

best_trial 提供了一个接口, 用当前的最佳超参数值对目标函数进行重新求值.

像上面的第一个例子一样, 本教程展示了利用当前最佳值来重新运行一个不同的 *objective* 函数的例子.

进一步研究最佳模型

考虑一个如下的采用了 Optuna 的经典有监督分类问题

```
from sklearn import metrics
from sklearn.datasets import make_classification
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split


import optuna


def objective(trial):
    X, y = make_classification(n_features=10, random_state=1)
    X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1)

    C = trial.suggest_loguniform("C", 1e-7, 10.0)

    clf = LogisticRegression(C=C)
    clf.fit(X_train, y_train)

    return clf.score(X_test, y_test)


study = optuna.create_study(direction="maximize")
study.optimize(objective, n_trials=10)

print(study.best_trial.value)  # Show the best value.
```

Out:

```
0.92
```

假设在超参数优化之后, 你想计算出同一个数据集上的其他的测度, 比如召回率, 精确度和 f1-score. 你可以定义另一个目标函数, 令其和 `objective` 高度相似, 以用最佳超参数来重现模型.

```
def detailed_objective(trial):
    # Use same code objective to reproduce the best model
    X, y = make_classification(n_features=10, random_state=1)
    X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1)

    C = trial.suggest_loguniform("C", 1e-7, 10.0)

    clf = LogisticRegression(C=C)
    clf.fit(X_train, y_train)

    # calculate more evaluation metrics
    pred = clf.predict(X_test)

    acc = metrics.accuracy_score(pred, y_test)
    recall = metrics.recall_score(pred, y_test)
    precision = metrics.precision_score(pred, y_test)
    f1 = metrics.f1_score(pred, y_test)

    return acc, f1, recall, precision
```

将 `study.best_trial` 传递给 `detailed_objective` 做参数.

```
detailed_objective(study.best_trial)  # calculate acc, f1, recall, and precision
```

Out:

```
(0.92, 0.9285714285714286, 0.9285714285714286, 0.9285714285714286)
```

`best_trial` 和普通的 trials 的不同之处

这里使用了*best_trial*, 它返回一个 *best_trial*, 属于*FrozenTrial* 类型. *FrozenTrial* 与活跃的 trial 不同, 而且在某些情况下它的行为和*Trial* 也不一样. 比如, 剪枝没法使用, 因为*should_prune* 总是返回 `False`.

Total running time of the script: (0 minutes 0.060 seconds)

6.3 API Reference

6.3.1 optuna

在 optuna 被用在其他模块中时，*optuna* 模块主要作为 Optuna 的基础功能的别名来使用。目前有两个模块有别名：(1) *optuna.study*，包含和 Study 生命周期有关的函数，(2) *optuna.exceptions*，是当 trial 被剪枝时抛出的 TrialPruned 异常。

optuna.create_study	创建新*Study*.
optuna.load_study	加载指定名字的*Study*。
optuna.delete_study	删除*Study* 对象。
optuna.get_all_study_summaries	获取指定存储内所有 study 的历史记录。
optuna.TrialPruned	被剪枝的 trial 异常。

optuna.create_study

optuna.**create_study**(*storage=None*, *sampler=None*, *pruner=None*, *study_name=None*, *direction=None*, *load_if_exists=False*, *, *directions=None*)

创建一个新的*Study*.

示例

```
import optuna


def objective(trial):
    x = trial.suggest_float("x", 0, 10)
    return x ** 2


study = optuna.create_study()
study.optimize(objective, n_trials=3)
```

参数

- **storage** (*Optional[Union[str, optuna.storages._base.BaseStorage]]*) –数据库 URL. 如果该参数被设置为 None, 我们将采用内存存储, 而*Study* 将不会被持久化.

  备注:

  当传入一个数据库 URL 后, Optuna 将在内部使用 SQLAlchemy 来连接数据库. 更多细节请参考 SQLAlchemy’s document. 如果想更改 SQLAlchemy Engine 的默认选项, 你可以用你想要的选项来实例化*RDBStorage* 并将其, 而不是一个 URL, 传入 storage 参数.

- **sampler**(*Optional[optuna.samplers._base.BaseSampler]*)—一个实现了值 suggestion 背景算法的 Sampler 对象. 如果设为 None 的话, *TPESampler* 将在单目标优化时被采用, 而*NSGAIISampler* 会在多目标优化时被采用. 参看*samplers*.

- **pruner** (*Optional[optuna.pruners._base.BasePruner]*) –一个用于提前终止无望 trial 的 pruner 对象. 如果设置为 None 的话, *MedianPruner* 将会被采用. 参见*pruners*.
- **study_name** (*Optional[str]*) –Study 的名字. 如果该参数被设置为 None 的话, Optuna 将会采用一个自动生成的名字.
- **direction** (*Optional[Union[str, optuna._study_direction.StudyDirection]]*) –优化的方向. 最小化时设置为 minimize ，最大化时设置为 maximize. 你也可以传入一个对应的*StudyDirection* 对象.

  备注: 如果 *direction* 和 *directions* 都没有设置的话, study 的方向将被设置为 “minimize” .

- **load_if_exists** (*bool*) –用于处理在 study 名字冲突情况下的行为选项. 在已经有一个名为 study_name 的 sudy 于 storage 中存在时, 如果 load_if_exists 被设置为 False, Optuna 将抛出*DuplicatedStudyError* 错误. 否则, study 的创建过程将被略过而返回已有的那个 study.
- **directions** (*Optional[Sequence[Union[str, optuna._study_direction.StudyDirection]]]*) –在多目标优化中的一系列的方向.

返回 A *Study* object.

引发 **ValueError** –如果 directions 的长度是 0. 或者, 如果 direction 既不是‘minimize’也不是 *maximize*. 或者, 如果 direction 和 directions 被同时设定的话.

返回类型 *optuna.study.Study*

参见:

optuna.create_study() is an alias of *optuna.study.create_study()*.

optuna.load_study

optuna.**load_study**(*study_name*, *storage*, *sampler=None*, *pruner=None*)

加载指定名字的现存*Study*.

示例

```
import optuna


def objective(trial):
    x = trial.suggest_float("x", 0, 10)
    return x ** 2


study = optuna.create_study(storage="sqlite:///example.db", study_name="my_study")
study.optimize(objective, n_trials=3)

loaded_study = optuna.load_study(study_name="my_study", storage="sqlite:///
↪example.db")
assert len(loaded_study.trials) == len(study.trials)
```

参数

- **study_name** (*str*) –Study 的名字. 每个 study 都有一个独特的名字作为标识符.
- **storage** (*Union[str, optuna.storages._base.BaseStorage]*) –数据库 URL, 比如 sqlite:///example.db. 更多细节参见文档*create_study()*.
- **sampler** (*Optional[optuna.samplers._base.BaseSampler]*)–一个实现了值 suggestion 背景算法的 Sampler 对象. 如果设为 None 的话, *TPESampler* 将在单目标优化时被采用. 参看*samplers*.
- **pruner** (*Optional[optuna.pruners._base.BasePruner]*) –一个用于提前终止无望 trial 的 pruner 对象. 如果设置为 None 的话, *MedianPruner* 将会被采用. 参见*pruners*.

返回类型 *optuna.study.Study*

参见:

optuna.load_study() 是*optuna.study.load_study()* 的别名.

optuna.delete_study

optuna.**delete_study**(*study_name*, *storage*)

Delete a *Study* object.

示例

```
import optuna


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


study = optuna.create_study(study_name="example-study", storage="sqlite:///
↪example.db")
study.optimize(objective, n_trials=3)

optuna.delete_study(study_name="example-study", storage="sqlite:///example.db")
```

参数

- **study_name** (*str*) –Study’s name.
- **storage** (*Union[str, optuna.storages._base.BaseStorage]*) –数据库 URL, 比如 sqlite:///example.db. 更多细节参见文档*create_study()*.

返回类型 None

参见:

optuna.delete_study() 是*optuna.study.delete_study()* 的别名.

optuna.get_all_study_summaries

optuna.**get_all_study_summaries**(*storage*)

返回指定存储中的所有 study 的历史记录.

示例

```
import optuna


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

(下页继续)

(续上页)

```
study = optuna.create_study(study_name="example-study", storage="sqlite:///
↪example.db")
study.optimize(objective, n_trials=3)

study_summaries = optuna.study.get_all_study_summaries(storage="sqlite:///example.
↪db")
assert len(study_summaries) == 1

study_summary = study_summaries[0]
assert study_summary.study_name == "example-study"
```

参数 **storage** (*Union[str, optuna.storages._base.BaseStorage]*) – 数据库 URL, 比如 sqlite:///example.db. 更多细节参见文档*create_study()*.

返回 由*StudySummary* 对象构成的 study 历史记录列表.

返回类型 *List*[*optuna._study_summary.StudySummary*]

参见:

optuna.get_all_study_summaries() 是*optuna.study.get_all_study_summaries()* 的别名.

optuna.TrialPruned

exception optuna.**TrialPruned**

被剪枝的 trial 异常.

该错误告诉用户当前*Trial* 已经被剪枝. 就像下面的例子里展示的那样, 它被设定在*optuna.trial.Trial.should_prune()* 之后被触发.

参见:

optuna.TrialPruned() 是*optuna.exceptions.TrialPruned()* 的别名.

示例

```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import train_test_split

import optuna
```

(下页继续)

(续上页)

```
X, y = load_iris(return_X_y=True)
X_train, X_valid, y_train, y_valid = train_test_split(X, y)
classes = np.unique(y)


def objective(trial):
    alpha = trial.suggest_float("alpha", 0.0, 1.0)
    clf = SGDClassifier(alpha=alpha)
    n_train_iter = 100

    for step in range(n_train_iter):
        clf.partial_fit(X_train, y_train, classes=classes)

        intermediate_value = clf.score(X_valid, y_valid)
        trial.report(intermediate_value, step)

        if trial.should_prune():
            raise optuna.TrialPruned()

    return clf.score(X_valid, y_valid)


study = optuna.create_study(direction="maximize")
study.optimize(objective, n_trials=20)
```

6.3.2 optuna.cli

cli 模块使用 cliff framework 框架实现了 Optuna 的命令行功能。

```
optuna
    [--version]
    [-v | -q]
    [--log-file LOG_FILE]
    [--debug]
    [--storage STORAGE]
```

--version

显示应用的版本号并退出

-v, --verbose

打印冗余内容。可重复。

-q, --quiet

禁止打印除了警告和错误之外的输出。

--log-file <LOG_FILE>

确定用于记录日志的文件名。该选项默认情况下是禁用的。

--debug

显示异常处理和堆栈跟踪信息。

--storage <STORAGE>

数据库 URL. (比如 `sqlite:///example.db`)

create-study

创建新的 study

```
optuna create-study
    [--study-name STUDY_NAME]
    [--direction {minimize,maximize}]
    [--skip-if-exists]
```

--study-name <STUDY_NAME>

设定具有可读性的 study 名，以方便和其他的 study 区分。

--direction <DIRECTION>

设定新 study 的优化方向。如果方向是最小化的话就设成 'minimize' , 如果是最大化就设置成 'maximize' .

--skip-if-exists

如果开启该选项，当存在一个同名的 study 时，optuna 在优化开始时将跳过创建 study 的过程，并且不报错。

该命令由 optuna 插件提供。

dashboard

启动 web dashboard (beta).

This feature is deprecated since version 2.7.0. Please use optuna-dashboard instead.

```
optuna dashboard
    [--study STUDY]
    [--study-name STUDY_NAME]
    [--out OUT]
    [--allow-websocket-origin BOKEH_ALLOW_WEBSOCKET_ORIGINS]
```

--study <STUDY>

该参数已弃用。作为替代，请使用–study-name.

--study-name <STUDY_NAME>

The name of the study to show on the dashboard.

--out <OUT>, **-o** <OUT>

HTML 文件的输出路径。如果该参数没有设定的话，Optuna 将会启动一个 HTTP server 并 host 一个 dashboard

--allow-websocket-origin <BOKEH_ALLOW_WEBSOCKET_ORIGINS>

允许从制定的 host 发起的的 websocket 请求。该选项用法和 bokeh 的 `–allow-websocket-origin` 一样。更多细节见 https://docs.bokeh.org/en/latest/docs/reference/command/subcommands/serve.html. 设定具有可读性的 study 名，以方便和其他的 study 区分。

该命令由 optuna 插件提供。

delete-study

删除特定的 study。

```
optuna delete-study [--study-name STUDY_NAME]
```

--study-name <STUDY_NAME>

待删除的 study 名。

该命令由 optuna 插件提供。

storage upgrade

升级数据库架构。

```
optuna storage upgrade
```

该命令由 optuna 插件提供。

studies

显示 study 列表。

```
optuna studies
    [-f {csv,json,table,value,yaml}]
    [-c COLUMN]
    [--quote {all,minimal,none,nonnumeric}]
```

(下页继续)

(续上页)

```
    [--noindent]
    [--max-width <integer>]
    [--fit-width]
    [--print-empty]
    [--sort-column SORT_COLUMN]
    [--sort-ascending | --sort-descending]
```

-f <FORMATTER>, **--format** <FORMATTER>

输出格式，默认情况下是表格。

-c COLUMN, **--column** COLUMN

用于设定要展示的（表格）列。可设置多个 column 参数以同时显示多个 column.

--quote <QUOTE_MODE>

when to include quotes, defaults to nonnumeric

--noindent

设定是否在 json 输出格式中禁用缩进。

--max-width <integer>

最大展示宽度。如果设置成小于 1 的数值，就代表禁用最大宽度限制。你也可以通过设置 CLIFF_MAX_TERM_WIDTH `` 环境变量来限制最大宽度，但是 ``--max-width 参数的优先级比环境变量高。

--fit-width

将输出表格设置成与显示屏同宽。如果 -max-width > 1 的话，该选项会自动启用。设定是否在 json 中禁用缩进。也可以通过设置 CLIFF_FIT_WIDTH 环境变量的值为 1 来启用该选项。

--print-empty

在无数据的情况下也输出空表格

--sort-column SORT_COLUMN

按照指定列来对表格的行进行排序。越靠前指定的列在排序中的优先级越高。没有被指定的列不参与排序。

--sort-ascending

sort the column(s) in ascending order

--sort-descending

sort the column(s) in descending order

该命令由 optuna 插件提供。

study optimize

针对特定 study 开始优化过程。自 2.0.0 版本后弃用。

```
optuna study optimize
    [--n-trials N_TRIALS]
    [--timeout TIMEOUT]
    [--n-jobs N_JOBS]
    [--study STUDY]
    [--study-name STUDY_NAME]
    file
    method
```

--n-trials <N_TRIALS>

优化过程将运行的总 trial 数目。如果不设定该数目的话，优化过程会一直持续下去。

--timeout <TIMEOUT>

到达指定时间（按秒计）以后终止 study. 如果不设定该数目的话，优化过程也会一直持续下去。

--n-jobs <N_JOBS>

并行运行的任务个数。如果该参数设置为 -1, 则任务的实际数目将等于 CPU 的核心数。

--study <STUDY>

该参数已弃用。作为替代，请使用–study-name.

--study-name <STUDY_NAME>

被优化的 study 名。

file

用于指定定义了目标函数的 Python 脚本文件名。

method

目标函数的方法名。

该命令由 optuna 插件提供。

study set-user-attr

针对特定 study 设定用户属性 (user attribute).

```
optuna study set-user-attr
    [--study STUDY]
    [--study-name STUDY_NAME]
    --key KEY
    --value VALUE
```

--study <STUDY>

该参数已弃用。作为替代，请使用–study-name.

--study-name <STUDY_NAME>

用于设置用户属性的 study 名。

--key <KEY>, **-k** <KEY>

设定用户属性中的键 (key).

--value <VALUE>, **-v** <VALUE>

设定用户属性中的值 (value).

该命令由 optuna 插件提供。

6.3.3 optuna.distributions

distributions 模块定义了代表各种概率分布的类, 主要用于优化 trial 时进行初始超参数建议. 这些分布类继承自库内部的 BaseDistribution, 并且以特定的参数被初始化, 比如对*UniformDistribution* 来说是 low 和 high 端点.

Optuna 用户不应该直接使用分布类, 而应该使用*Trial* 提供的实用函数, 如*suggest_int()*.

optuna.distributions.UniformDistribution	线性域的均匀分布.
optuna.distributions.LogUniformDistribution	Log 均匀分布.
optuna.distributions.DiscreteUniformDistribution	线性离散均匀分布.
optuna.distributions.IntUniformDistribution	整数上的均匀分布.
optuna.distributions.IntLogUniformDistribution	Log 操作后的整数均匀分布.
optuna.distributions.CategoricalDistribution	分类分布.
optuna.distributions.distribution_to_json	将分布序列化成 json 格式.
optuna.distributions.json_to_distribution	将 JSON 格式的分布反序列化.
optuna.distributions.check_distribution_compatibility	检查两分布的兼容性.

optuna.distributions.UniformDistribution

class optuna.distributions.**UniformDistribution**(*low*, *high*)

线性域的均匀分布.

一般情况下, 该对象由*suggest_uniform()* 进行实例化, 然后被传递到*samplers*.

low

该分布的取值范围下界. 其中 low 值是包含在取值范围内的.

high

该分布的取值范围上界。其中 high 值是包含在取值范围内的.

引发 **ValueError** –如果 low 的值比 high 大的话.

参数

- **low** (*float*) –
- **high** (*float*) –

返回类型 None

Methods

single()	测试该分布是否仅包含单一值.
to_external_repr(param_value_in_internal_repr)	将参数的内部表示转化为外部表示.
to_internal_repr(param_value_in_external_repr)	将参数的外部表示转化为内部表示.

single()

测试该分布是否仅包含单一值.

返回 如果该分布仅包含一个值的话为 True, 否则为 False.

返回类型 bool

to_external_repr(*param_value_in_internal_repr*)

将参数的内部表示转化为外部表示.

参数 **param_value_in_internal_repr** (*float*) –Optuna 对参数值的内部表示

返回 Optuna 对参数值的外部表示

返回类型 *Any*

to_internal_repr(*param_value_in_external_repr*)

将参数的外部表示转化为内部表示.

参数 **param_value_in_external_repr** (*Any*) –Optuna 对参数值的外部表示

返回 Optuna 对参数值的内部表示

返回类型 float

optuna.distributions.LogUniformDistribution

class optuna.distributions.**LogUniformDistribution**(*low*, *high*)

对数域的均匀分布.

一般情况下, 该对象在 log=True 和*suggest_loguniform()* 时由*suggest_float()* 进行实例化, 然后被传递到*samplers*.

low

该分布的取值范围下界. 其中 low 值是包含在取值范围内的.

high

该分布的取值范围上界. 其中 high 值是包含在取值范围内的.

引发 **ValueError** –如果 low 的值比 high 大, 或者 low 小于等于 0 的话.

参数

- **low** (*float*) –
- **high** (*float*) –

返回类型 None

Methods

single()	测试该分布是否仅包含单一值.
to_external_repr(param_value_in_internal_repr)	将参数的内部表示转化为外部表示.
to_internal_repr(param_value_in_external_repr)	将参数的外部表示转化为内部表示.

single()

测试该分布是否仅包含单一值.

返回 如果该分布仅包含一个值的话为 True, 否则为 False.

返回类型 bool

to_external_repr(*param_value_in_internal_repr*)

将参数的内部表示转化为外部表示.

参数 **param_value_in_internal_repr** (*float*) –Optuna 对参数值的内部表示

返回 Optuna 对参数值的外部表示

返回类型 *Any*

to_internal_repr(*param_value_in_external_repr*)

将参数的外部表示转化为内部表示.

参数 **param_value_in_external_repr** (*Any*) –Optuna 对参数值的外部表示

返回 Optuna 对参数值的内部表示

返回类型 float

optuna.distributions.DiscreteUniformDistribution

class optuna.distributions.**DiscreteUniformDistribution**(*low*, *high*, *q*)

线性域的离散均匀分布.

一般情况下, 该对象在有 step 参数和*suggest_discrete_uniform()* 时由*suggest_uniform()* 进行实例化, 然后被传递到*samplers*.

备注: 如果区间 [low, high] 不能被步数 q 整除的话，值 high 会被替代成 $kq + \text{low}$ 小于 high 的最大值，其中 k 是整数.

---

low

该分布的取值范围下界. 其中 low 值是包含在取值范围内的.

high

该分布的取值范围上界. 其中 high 值是包含在取值范围内的.

q

离散化步长

引发 **ValueError** –如果 low 的值比 high 大的话.

参数

- **low** (*float*) –
- **high** (*float*) –
- **q** (*float*) –

返回类型 None

Methods

single()	测试该分布是否仅包含单一值.
to_external_repr(param_value_in_internal_repr)	将参数的内部表示转化为外部表示.
to_internal_repr(param_value_in_external_repr)	将参数的外部表示转化为内部表示.

single()

测试该分布是否仅包含单一值.

返回 如果该分布仅包含一个值的话为 `True`, 否则为 `False`.

返回类型 bool

to_external_repr(*param_value_in_internal_repr*)

将参数的内部表示转化为外部表示.

参数 **param_value_in_internal_repr** (*float*) –Optuna 对参数值的内部表示

返回 Optuna 对参数值的外部表示

返回类型 *Any*

to_internal_repr(*param_value_in_external_repr*)

将参数的外部表示转化为内部表示.

参数 **param_value_in_external_repr** (*Any*) –Optuna 对参数值的外部表示

返回 Optuna 对参数值的内部表示

返回类型 float

optuna.distributions.IntUniformDistribution

class optuna.distributions.**IntUniformDistribution**(*low*, *high*, *step=1*)

整数均匀分布.

一般情况下, 该对象由*suggest_int()* 进行实例化, 然后被传递到*samplers*.

备注: 如果区间 [low, high] 不能被步数 q 整除的话，值 high 会被替代成 $k \times q + \text{low}$high 的最大值，其中 k 是整数。

---

low

该分布的取值范围下界. 其中 `low` 值是包含在取值范围内的.

high

该分布的取值范围上界。其中 `high` 值是包含在取值范围内的.

step

值之间的步长

引发 **ValueError** –如果 low 的值比 high 大, 或者 step 小于等于 0 的话.

参数

- **low** (*int*) –
- **high** (*int*) –
- **step** (*int*) –

返回类型 None

Methods

single()	测试该分布是否仅包含单一值.
to_external_repr(param_value_in_internal_repr)	将参数的内部表示转化为外部表示.
to_internal_repr(param_value_in_external_repr)	将参数的外部表示转化为内部表示.

single()

测试该分布是否仅包含单一值.

返回 如果该分布仅包含一个值的话为 True, 否则为 False.

返回类型 bool

to_external_repr(*param_value_in_internal_repr*)

将参数的内部表示转化为外部表示.

参数 **param_value_in_internal_repr** (*float*) –Optuna 对参数值的内部表示

返回 Optuna 对参数值的外部表示

返回类型 int

to_internal_repr(*param_value_in_external_repr*)

将参数的外部表示转化为内部表示.

参数 **param_value_in_external_repr** (*int*) –Optuna 对参数值的外部表示

返回 Optuna 对参数值的内部表示

返回类型 float

optuna.distributions.IntLogUniformDistribution

class optuna.distributions.**IntLogUniformDistribution**(*low*, *high*, *step=1*)

对数域上的均匀整数分布.

一般情况下, 该对象由*suggest_int()* 进行实例化, 然后被传递到*samplers*.

low

该分布的取值范围下界. 其中 low 值是包含在取值范围内的.

high

该分布的取值范围上界。其中 high 值是包含在取值范围内的.

step

值之间的步长

> **警告:** 在 v2.0.0 中被废弃. 在未来, step 将会被移除. 目前, 这个特性的移除计划在 v4.0.0 中发生, 但是该计划可能会有改变.
>
> Optuna 中的 sampler 和其他依赖这个分布的组件将会忽略这个值并假定 step 永远为 1. 在被废弃之前, 用户定义的 sampler 可使用 1 之外的其他值.

引发 **ValueError** –如果 low 的值比 high 大, 或者 low 小于 1 的话.

参数

- **low** (*int*) –
- **high** (*int*) –
- **step** (*int*) –

返回类型 None

Methods

single()	测试该分布是否仅包含单一值.
to_external_repr(param_value_in_internal_repr)	将参数的内部表示转化为外部表示.
to_internal_repr(param_value_in_external_repr)	将参数的外部表示转化为内部表示.

Attributes

step

single()

测试该分布是否仅包含单一值.

返回 True if the range of this distribution contains just a single value, otherwise False.

返回类型 bool

to_external_repr(*param_value_in_internal_repr*)

将参数的内部表示转化为外部表示.

参数 **param_value_in_internal_repr** (*float*) –Optuna 对参数值的内部表示

返回 Optuna 对参数值的外部表示

返回类型 int

to_internal_repr(*param_value_in_external_repr*)

将参数的外部表示转化为内部表示.

参数 **param_value_in_external_repr** (*int*) –Optuna 对参数值的外部表示

返回 Optuna 对参数值的内部表示

返回类型 float

optuna.distributions.CategoricalDistribution

class optuna.distributions.**CategoricalDistribution**(*choices*)

分类分布.

一般情况下, 该对象由*suggest_categorical()* 进行实例化之后会被传递到*samplers*

参数 **choices** (*Sequence[Union[None, bool, int, float, str]]*) –Parameter value candidates.

返回类型 None

备注: 不是所有类型的参数都一定能和 optuna 的存储引擎兼容，请尽量使用 None, bool, int, float 或者 str.

---

choices

Parameter value candidates.

引发 **ValueError** –If `choices` do not contain any elements.

参数 **choices** (*Sequence[Union[None, bool, int, float, str]]*) –

返回类型 None

Methods

single()	测试该分布是否仅包含单一值.
to_external_repr(param_value_in_internal_repr)	将参数的内部表示转化为外部表示.
to_internal_repr(param_value_in_external_repr)	将参数的外部表示转化为内部表示.

single()

测试该分布是否仅包含单一值.

返回 如果该分布仅包含一个值的话为 `True`, 否则为 `False`.

返回类型 bool

to_external_repr(*param_value_in_internal_repr*)

将参数的内部表示转化为外部表示.

参数 **param_value_in_internal_repr** (*float*) –Optuna 对参数值的内部表示

返回 Optuna 对参数值的外部表示

返回类型 *Union*[None, bool, int, float, str]

to_internal_repr(*param_value_in_external_repr*)

将参数的外部表示转化为内部表示.

参数 **param_value_in_external_repr** (*Union[None, bool, int, float, str]*) –Optuna 对参数值的外部表示

返回 Optuna 对参数值的内部表示

返回类型 float

optuna.distributions.distribution_to_json

optuna.distributions.**distribution_to_json**(*dist*)

将一个分布序列化成 JSON 格式

参数 **dist** (*optuna.distributions.BaseDistribution*) –待序列化的分布.

返回 给定分布的 JSON 字符串.

返回类型 str

optuna.distributions.json_to_distribution

optuna.distributions.**json_to_distribution**(*json_str*)

对 JSON 格式的分布进行反序列化.

参数 **json_str** (*str*) – 已被 JSON 序列化的分布.

返回 反序列化后的分布.

引发 **ValueError** – 如果被指定为未知类的话.

返回类型 optuna.distributions.BaseDistribution

optuna.distributions.check_distribution_compatibility

optuna.distributions.**check_distribution_compatibility**(*dist_old*, *dist_new*)

检查两分布的兼容性的函数.

注意，该方法不应由 optuna 用户来调用

参数

- **dist_old** (*optuna.distributions.BaseDistribution*) – 从存储内容中恢复的分布
- **dist_new** (*optuna.distributions.BaseDistribution*) – 新添加到存储内容的分布

返回 如果是 True 的话，表明这些分布是兼容的，否则不兼容。

引发 **ValueError** – 如果 dist_old 和 dist_new 属于不同类的分布或者 dist_old.choices 不匹配*CategoricalDistribution* 的 dist_new.choices.

返回类型 None

6.3.4 optuna.exceptions

exceptions 模块定义了 Optuna 特有的异常，它派生自一个基本的*OptunaError* 类。对于库用户来说，特别重要的是*TrialPruned* 异常。此异常在*optuna.trial.Trial.should_prune()* 对一个应被剪枝的 trial 返回 True 时被抛出。

optuna.exceptions.OptunaError	Optuna 异常基类。
optuna.exceptions.TrialPruned	被剪枝 trial 异常。
optuna.exceptions.CLIUsageError	命令行界面异常。
optuna.exceptions.StorageInternalError	存储异常。
optuna.exceptions.DuplicatedStudyError	重复 study 名导致的异常。

optuna.exceptions.OptunaError

exception optuna.exceptions.**OptunaError**

Optuna 错误的基类.

optuna.exceptions.TrialPruned

exception optuna.exceptions.**TrialPruned**

被剪枝 trial 的异常

该错误表明当前*Trial*已被剪枝. 像下面例子中那样, 一般*optuna.trial.Trial.should_prune()*出现后会抛出该异常.

参见:

*optuna.TrialPruned*是*optuna.exceptions.TrialPruned*的一个别名.

示例

```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import train_test_split

import optuna

X, y = load_iris(return_X_y=True)
X_train, X_valid, y_train, y_valid = train_test_split(X, y)
classes = np.unique(y)


def objective(trial):
    alpha = trial.suggest_float("alpha", 0.0, 1.0)
    clf = SGDClassifier(alpha=alpha)
    n_train_iter = 100

    for step in range(n_train_iter):
        clf.partial_fit(X_train, y_train, classes=classes)

        intermediate_value = clf.score(X_valid, y_valid)
        trial.report(intermediate_value, step)

        if trial.should_prune():
            raise optuna.TrialPruned()
```

(下页继续)

(续上页)

```
    return clf.score(X_valid, y_valid)


study = optuna.create_study(direction="maximize")
study.optimize(objective, n_trials=20)
```

optuna.exceptions.CLIUsageError

exception optuna.exceptions.**CLIUsageError**

命令行界面异常.

命令行界面将在收到无效配置时抛出异常.

optuna.exceptions.StorageInternalError

exception optuna.exceptions.**StorageInternalError**

存储操作的异常.

当后端存储数据库操作失败时, 该错误将会被触发.

optuna.exceptions.DuplicatedStudyError

exception optuna.exceptions.**DuplicatedStudyError**

重复 study 名的异常

当存储中已有一个和指定 study 同名的 study 时, 该错误将会被触发.

6.3.5 optuna.importance

importance 模块提供了根据给定 study 中已完成的 trials 评估超参数重要性的功能。实用函数*get_param_importances()* 将一个 Study 和可选的评价器作为它的两个输入。评价器必须从 BaseImportanceEvaluator 派生，当没有传入时，默认初始化为*FanovaImportanceEvaluator*。实现自定义评估器的用户应该参考*FanovaImportanceEvaluator* 或*MeanDecreaseImpurityImportanceEvaluator* 作为指导，请格外注意 Evaluator 的 evaluate() 函数的返回值的格式。

optuna.importance.get_param_importances	在给定的 study 中根据已完成的 trial 来评估参数重要性。
optuna.importance.FanovaImportanceEvaluator	fANOVA 重要性求解器。
optuna.importance.MeanDecreaseImpurityImportanceEvaluator	Mean Decrease Impurity (MDI) 参数重要性求解器。

optuna.importance.get_param_importances

optuna.importance.**get_param_importances**(*study*, ***, *evaluator=None*, *params=None*, *target=None*)

Evaluate parameter importances based on completed trials in the given study.

The parameter importances are returned as a dictionary where the keys consist of parameter names and their values importances. The importances are represented by floating point numbers that sum to 1.0 over the entire dictionary. The higher the value, the more important. The returned dictionary is of type `collections.OrderedDict` and is ordered by its values in a descending order.

If `params` is `None`, all parameter that are present in all of the completed trials are assessed. This implies that conditional parameters will be excluded from the evaluation. To assess the importances of conditional parameters, a `list` of parameter names can be specified via `params`. If specified, only completed trials that contain all of the parameters will be considered. If no such trials are found, an error will be raised.

If the given study does not contain completed trials, an error will be raised.

备注： If `params` is specified as an empty list, an empty dictionary is returned.

参见：

See *plot_param_importances()* to plot importances.

参数

- **study** (`optuna.study.Study`) –An optimized study.
- **evaluator** (*Optional[optuna.importance._base.BaseImportanceEvaluator]*) –An importance evaluator object that specifies which algorithm to base the importance assessment on. Defaults to *FanovaImportanceEvaluator*.
- **params** (*Optional[List[str]]*) –A list of names of parameters to assess. If `None`, all parameters that are present in all of the completed trials are assessed.
- **target** (*Optional[Callable[[optuna.trial._frozen.FrozenTrial], float]]*) –A function to specify the value to evaluate importances. If it is `None` and `study` is being used for single-objective optimization, the objective values are used.

备注： Specify this argument if `study` is being used for multi-objective optimization.

返回 An `collections.OrderedDict` where the keys are parameter names and the values are assessed importances.

引发 **ValueError** –If `target` is `None` and `study` is being used for multi-objective optimization.

返回类型 *Dict*[str, float]

optuna.importance.FanovaImportanceEvaluator

class optuna.importance.**FanovaImportanceEvaluator**(*, *n_trees=64*, *max_depth=64*, *seed=None*)

fANOVA importance evaluator.

Implements the fANOVA hyperparameter importance evaluation algorithm in An Efficient Approach for Assessing Hyperparameter Importance.

Given a study, fANOVA fits a random forest regression model that predicts the objective value given a parameter configuration. The more accurate this model is, the more reliable the importances assessed by this class are.

备注： Requires the sklearn Python package.

备注： Pairwise and higher order importances are not supported through this class. They can be computed using `_Fanova` directly but is not recommended as interfaces may change without prior notice.

备注： The performance of fANOVA depends on the prediction performance of the underlying random forest model. In order to obtain high prediction performance, it is necessary to cover a wide range of the hyperparameter search space. It is recommended to use an exploration-oriented sampler such as *RandomSampler*.

备注： For how to cite the original work, please refer to https://automl.github.io/fanova/cite.html.

参数

- **n_trees** (*int*) –The number of trees in the forest.
- **max_depth** (*int*) –The maximum depth of the trees in the forest.

- **seed** (*Optional[int]*) –Controls the randomness of the forest. For deterministic behavior, specify a value other than None.

返回类型 None

Methods

evaluate(study[, params, target])	Evaluate parameter importances based on completed trials in the given study.

evaluate (*study*, *params=None*, ***, *target=None*)

Evaluate parameter importances based on completed trials in the given study.

备注： This method is not meant to be called by library users.

参见：

Please refer to *get_param_importances()* for how a concrete evaluator should implement this method.

参数

- **study** (optuna.study.Study) –An optimized study.
- **params** (*Optional[List[str]]*) –A list of names of parameters to assess. If None, all parameters that are present in all of the completed trials are assessed.
- **target** (*Optional[Callable[[optuna.trial._frozen.FrozenTrial], float]]*) –A function to specify the value to evaluate importances. If it is None and study is being used for single-objective optimization, the objective values are used.

  备注： Specify this argument if study is being used for multi-objective optimization.

返回 An collections.OrderedDict where the keys are parameter names and the values are assessed importances.

引发 **ValueError** –If target is None and study is being used for multi-objective optimization.

返回类型 *Dict*[str, float]

optuna.importance.MeanDecreaseImpurityImportanceEvaluator

class optuna.importance.**MeanDecreaseImpurityImportanceEvaluator**(*, *n_trees=64*, *max_depth=64*, *seed=None*)

Mean Decrease Impurity (MDI) parameter importance evaluator.

This evaluator fits a random forest that predicts objective values given hyperparameter configurations. Feature importances are then computed using MDI.

备 注: This evaluator requires the sklean Python package and is based on sklearn.ensemble.RandomForestClassifier.feature_importances_.

参数

- **n_trees** (*int*) –Number of trees in the random forest.
- **max_depth** (*int*) –The maximum depth of each tree in the random forest.
- **seed** (*Optional[int]*) –Seed for the random forest.

返回类型 None

Methods

evaluate(study[, params, target])	Evaluate parameter importances based on completed trials in the given study.

evaluate(*study*, *params=None*, *, *target=None*)

Evaluate parameter importances based on completed trials in the given study.

备注: This method is not meant to be called by library users.

参见:

Please refer to *get_param_importances()* for how a concrete evaluator should implement this method.

参数

- **study** (optuna.study.Study) –An optimized study.
- **params** (*Optional[List[str]]*) –A list of names of parameters to assess. If None, all parameters that are present in all of the completed trials are assessed.

- **target** (*Optional[Callable[[optuna.trial._frozen.FrozenTrial], float]]*) –A function to specify the value to evaluate importances. If it is `None` and `study` is being used for single-objective optimization, the objective values are used.

  备注： Specify this argument if `study` is being used for multi-objective optimization.

返回 An `collections.OrderedDict` where the keys are parameter names and the values are assessed importances.

引发 **ValueError** –If `target` is `None` and `study` is being used for multi-objective optimization.

返回类型 *Dict*[str, float]

6.3.6 optuna.integration

integration 模块包含用于将 Optuna 与外部机器学习框架集成的类。

对于 Optuna 支持的大多数 ML 框架来说，相应的 Optuna 集成类的作用只是实现一个回调对象和函数，符合框架特定的回调 API，在模型训练的每一个中间步骤中都会被调用。这些回调在不同 ML 框架中实现的功能包括：

(1) 使用*optuna.trial.Trial.should_prune()* 向 Optuna trial 报告模型中间分数，

(2) 根据*optuna.trial.Trial.should_prune()* 的结果，通过抛出*optuna.TrialPruned()*，对当前模型进行剪枝，并且

(3) 将当前的 trial number 等中间 Optuna 数据汇报给框架，功能类似*MLflowCallback* 。

针对 scikit-learn，我们提供了一个集成的*OptunaSearchCV* 估计器，它结合了 scikit-learn BaseEstimator 功能和对类级 `Study` 对象的访问。

AllenNLP

optuna.integration.AllenNLPExecutor	为了让 Jsonnet 配置文件在 Optuna 中可用的 AllenNLP 扩展。
optuna.integration.allennlp.dump_best_config	保存 study 中最佳 trial 的参数更新到 JSON 配置文件。
optuna.integration.AllenNLPPruningCallback	用于清除无望 trial 的 AllenNLP 回调函数。

optuna.integration.AllenNLPExecutor

class optuna.integration.**AllenNLPExecutor**(*trial*, *config_file*, *serialization_dir*, *metrics='best_validation_accuracy'*, ***, *include_package=None*, *force=False*, *file_friendly_logging=False*)

AllenNLP extension to use optuna with Jsonnet config file.

This feature is experimental since AllenNLP major release will come soon. The interface may change without prior notice to correspond to the update.

See the examples of objective function.

You can also see the tutorial of our AllenNLP integration on AllenNLP Guide.

备注： From Optuna v2.1.0, users have to cast their parameters by using methods in Jsonnet. Call `std.parseInt` for integer, or `std.parseJson` for floating point. Please see the example configuration.

备注： In *AllenNLPExecutor*, you can pass parameters to AllenNLP by either defining a search space using Optuna suggest methods or setting environment variables just like AllenNLP CLI. If a value is set in both a search space in Optuna and the environment variables, the executor will use the value specified in the search space in Optuna.

参数

- **trial** (optuna.trial._trial.Trial) – A *Trial* corresponding to the current evaluation of the objective function.
- **config_file** (*str*) – Config file for AllenNLP. Hyperparameters should be masked with `std.extVar`. Please refer to the config example.
- **serialization_dir** (*str*) – A path which model weights and logs are saved.
- **metrics** (*str*) – An evaluation metric for the result of `objective`.
- **force** (*bool*) – If `True`, an executor overwrites the output directory if it exists.
- **file_friendly_logging** (*bool*) – If `True`, tqdm status is printed on separate lines and slows tqdm refresh rate.
- **include_package** (*Optional[Union[str, List[str]]]*) – Additional packages to include. For more information, please see AllenNLP documentation.

备注： Added in v1.4.0 as an experimental feature. The interface may change in newer versions without prior

notice. See https://github.com/optuna/optuna/releases/tag/v1.4.0.

Methods

run()	Train a model using AllenNLP.

run()

Train a model using AllenNLP.

返回类型 float

optuna.integration.allennlp.dump_best_config

optuna.integration.allennlp.**dump_best_config**(*input_config_file*, *output_config_file*, *study*)

Save JSON config file after updating with parameters from the best trial in the study.

参数

- **input_config_file** (*str*) –Input Jsonnet config file used with *AllenNLPExecutor*.
- **output_config_file** (*str*) –Output JSON config file.
- **study** (optuna.study.Study) –Instance of *Study*. Note that *optimize()* must have been called.

返回类型 None

optuna.integration.AllenNLPPruningCallback

class optuna.integration.**AllenNLPPruningCallback**(*trial=None*, *monitor=None*)

AllenNLP callback to prune unpromising trials.

See the example if you want to add a pruning callback which observes a metric.

You can also see the tutorial of our AllenNLP integration on AllenNLP Guide.

备注: When *AllenNLPPruningCallback* is instantiated in Python script, trial and monitor are mandatory.

On the other hand, when *AllenNLPPruningCallback* is used with *AllenNLPExecutor*, trial and monitor would be None. *AllenNLPExecutor* sets environment variables for a study name, trial id, monitor, and storage. Then *AllenNLPPruningCallback* loads them to restore trial and monitor.

参数

- **trial** (*Optional[optuna.trial._trial.Trial]*) –A *Trial* corresponding to the current evaluation of the objective function.
- **monitor** (*Optional[str]*) –An evaluation metric for pruning, e.g. `validation_loss` or `validation_accuracy`.

备注： Added in v2.0.0 as an experimental feature. The interface may change in newer versions without prior notice. See https://github.com/optuna/optuna/releases/tag/v2.0.0.

Methods

on_epoch(trainer, metrics, epoch[, is_primary])	Check if a training reaches saturation.
register(*args, **kwargs)	Stub method for *TrainerCallback.register*.

on_epoch (*trainer*, *metrics*, *epoch*, *is_primary=True*, ***kwargs*)

Check if a training reaches saturation.

参数

- **trainer** (*GradientDescentTrainer*) –AllenNLP’s trainer
- **metrics** (*Dict[str, Any]*) –Dictionary of metrics.
- **epoch** (*int*) –Number of current epoch.
- **is_primary** (*bool*) –A flag for AllenNLP internal.
- **kwargs** (*Any*) –

返回类型 None

classmethod register (**args*, ***kwargs*)

Stub method for *TrainerCallback.register*.

This method has the same signature as Registrable.register in AllenNLP.

参数

- **args** (*Any*) –
- **kwargs** (*Any*) –

返回类型 *Callable*

BoTorch

optuna.integration.BoTorchSampler	采用 BoTorch 的采样器。BoTorch 是一个建立在 PyTorch 上的贝叶斯优化库。
optuna.integration.botorch.qei_candidates_func	Quasi MC-based batch Expected Improvement (qEI).
optuna.integration.botorch.qehvi_candidates_func	Quasi MC-based batch Expected Hypervolume Improvement (qEHVI).
optuna.integration.botorch.qparego_candidates_func	Quasi MC-based extended ParEGO (qParEGO) for constrained multi-objective optimization.

optuna.integration.BoTorchSampler

class optuna.integration.**BoTorchSampler**(*, *candidates_func=None*, *constraints_func=None*, *n_startup_trials=10*, *independent_sampler=None*)

A sampler that uses BoTorch, a Bayesian optimization library built on top of PyTorch.

This sampler allows using BoTorch's optimization algorithms from Optuna to suggest parameter configurations. Parameters are transformed to continuous space and passed to BoTorch, and then transformed back to Optuna's representations. Categorical parameters are one-hot encoded.

参见:

See an example how to use the sampler.

参见:

See the BoTorch homepage for details and for how to implement your own `candidates_func`.

备注: An instance of this sampler *should be not used with different studies* when used with constraints. Instead, a new instance should be created for each new study. The reason for this is that the sampler is stateful keeping all the computed constraints.

参数

- **candidates_func** (*Callable[[torch.Tensor, torch.Tensor, Optional[torch.Tensor], torch.Tensor], torch.Tensor]*) – An optional function that suggests the next candidates. It must take the training data, the objectives, the constraints, the search space bounds and return the next candidates. The arguments are of type `torch.Tensor`. The return value must be a `torch.Tensor`. However, if `constraints_func` is omitted, constraints will be `None`. For any constraints that failed to compute, the tensor will contain NaN.

If omitted, is determined automatically based on the number of objectives. If the number of objectives is one, Quasi MC-based batch Expected Improvement (qEI) is used. If the number of objectives is larger than one but smaller than four, Quasi MC-based batch Expected Hypervolume Improvement (qEHVI) is used. Otherwise, for larger number of objectives, the faster Quasi MC-based extended ParEGO (qParEGO) is used.

The function should assume *maximization* of the objective.

参见:

See *optuna.integration.botorch.qei_candidates_func()* for an example.

- **constraints_func** (*Optional[Callable[[optuna.trial._frozen.FrozenTrial], Sequence[float]]]*) –An optional function that computes the objective constraints. It must take a *FrozenTrial* and return the constraints. The return value must be a sequence of `float` s. A value strictly larger than 0 means that a constraints is violated. A value equal to or smaller than 0 is considered feasible.

  If omitted, no constraints will be passed to `candidates_func` nor taken into account during suggestion if `candidates_func` is omitted.

- **n_startup_trials** (*int*) –Number of initial trials, that is the number of trials to resort to independent sampling.

- **independent_sampler** (*Optional[optuna.samplers._base.BaseSampler]*) –An independent sampler to use for the initial trials and for parameters that are conditional.

备注: Added in v2.4.0 as an experimental feature. The interface may change in newer versions without prior notice. See https://github.com/optuna/optuna/releases/tag/v2.4.0.

Methods

after_trial(study, trial, state, values)	Trial post-processing.
infer_relative_search_space(study, trial)	Infer the search space that will be used by relative sampling in the target trial.
reseed_rng()	Reseed sampler's random number generator.
sample_independent(study, trial, param_name, ...)	Sample a parameter for a given distribution.
sample_relative(study, trial, search_space)	Sample parameters in a given search space.

after_trial (*study*, *trial*, *state*, *values*)

Trial post-processing.

This method is called after the objective function returns and right before the trials is finished and its state is stored.

备注： Added in v2.4.0 as an experimental feature. The interface may change in newer versions without prior notice. See https://github.com/optuna/optuna/releases/tag/v2.4.0.

参数

- **study** (optuna.study.Study) –Target study object.
- **trial** (optuna.trial._frozen.FrozenTrial) –Target trial object. Take a copy before modifying this object.
- **state** (optuna.trial._state.TrialState) –Resulting trial state.
- **values** (*Optional[Sequence[float]]*) –Resulting trial values. Guaranteed to not be None if trial succeeded.

返回类型 None

infer_relative_search_space(*study*, *trial*)

Infer the search space that will be used by relative sampling in the target trial.

This method is called right before *sample_relative()* method, and the search space returned by this method is passed to it. The parameters not contained in the search space will be sampled by using *sample_independent()* method.

参数

- **study** (optuna.study.Study) –Target study object.
- **trial** (optuna.trial._frozen.FrozenTrial) –Target trial object. Take a copy before modifying this object.

返回 A dictionary containing the parameter names and parameter's distributions.

返回类型 *Dict*[str, optuna.distributions.BaseDistribution]

参见:

Please refer to *intersection_search_space()* as an implementation of *infer_relative_search_space()*.

reseed_rng()

Reseed sampler's random number generator.

This method is called by the *Study* instance if trials are executed in parallel with the option n_jobs>1. In that case, the sampler instance will be replicated including the state of the random number generator, and they may suggest the same values. To prevent this issue, this method assigns a different seed to each random number generator.

返回类型 None

sample_independent(*study*, *trial*, *param_name*, *param_distribution*)

Sample a parameter for a given distribution.

This method is called only for the parameters not contained in the search space returned by *sample_relative()* method. This method is suitable for sampling algorithms that do not use relationship between parameters such as random sampling and TPE.

备注： The failed trials are ignored by any build-in samplers when they sample new parameters. Thus, failed trials are regarded as deleted in the samplers' perspective.

参数

- **study** (optuna.study.Study) –Target study object.
- **trial** (optuna.trial._frozen.FrozenTrial) –Target trial object. Take a copy before modifying this object.
- **param_name** (*str*) –Name of the sampled parameter.
- **param_distribution** (*optuna.distributions.BaseDistribution*) – Distribution object that specifies a prior and/or scale of the sampling algorithm.

返回 A parameter value.

返回类型 *Any*

sample_relative(*study*, *trial*, *search_space*)

Sample parameters in a given search space.

This method is called once at the beginning of each trial, i.e., right before the evaluation of the objective function. This method is suitable for sampling algorithms that use relationship between parameters such as Gaussian Process and CMA-ES.

备注： The failed trials are ignored by any build-in samplers when they sample new parameters. Thus, failed trials are regarded as deleted in the samplers' perspective.

参数

- **study** (optuna.study.Study) –Target study object.
- **trial** (optuna.trial._frozen.FrozenTrial) –Target trial object. Take a copy before modifying this object.

- **search_space** (*Dict[str, optuna.distributions.BaseDistribution]*) –The search space returned by *infer_relative_search_space()*.

返回 A dictionary containing the parameter names and the values.

返回类型 *Dict*[str, *Any*]

optuna.integration.botorch.qei_candidates_func

optuna.integration.botorch.**qei_candidates_func**(*train_x*, *train_obj*, *train_con*, *bounds*)

Quasi MC-based batch Expected Improvement (qEI).

The default value of `candidates_func` in *BoTorchSampler* with single-objective optimization.

参数

- **train_x** (*torch.Tensor*) –Previous parameter configurations. A `torch.Tensor` of shape `(n_trials, n_params)`. `n_trials` is the number of already observed trials and `n_params` is the number of parameters. `n_params` may be larger than the actual number of parameters if categorical parameters are included in the search space, since these parameters are one-hot encoded. Values are not normalized.
- **train_obj** (*torch.Tensor*) –Previously observed objectives. A `torch.Tensor` of shape `(n_trials, n_objectives)`. `n_trials` is identical to that of `train_x`. `n_objectives` is the number of objectives. Observations are not normalized.
- **train_con** (*Optional[torch.Tensor]*) –Objective constraints. A `torch.Tensor` of shape `(n_trials, n_constraints)`. `n_trials` is identical to that of `train_x`. `n_constraints` is the number of constraints. A constraint is violated if strictly larger than 0. If no constraints are involved in the optimization, this argument will be `None`.
- **bounds** (*torch.Tensor*) –Search space bounds. A `torch.Tensor` of shape `(n_params, 2)`. `n_params` is identical to that of `train_x`. The first and the second column correspond to the lower and upper bounds for each parameter respectively.

返回 Next set of candidates. Usually the return value of BoTorch's `optimize_acqf`.

返回类型 torch.Tensor

备注： Added in v2.4.0 as an experimental feature. The interface may change in newer versions without prior notice. See https://github.com/optuna/optuna/releases/tag/v2.4.0.

---

optuna.integration.botorch.qehvi_candidates_func

optuna.integration.botorch.**qehvi_candidates_func**(*train_x*, *train_obj*, *train_con*, *bounds*)

Quasi MC-based batch Expected Hypervolume Improvement (qEHVI).

The default value of `candidates_func` in *BoTorchSampler* with multi-objective optimization when the number of objectives is three or less.

参见:

qei_candidates_func() for argument and return value descriptions.

备注: Added in v2.4.0 as an experimental feature. The interface may change in newer versions without prior notice. See https://github.com/optuna/optuna/releases/tag/v2.4.0.

参数

- **train_x** (*torch.Tensor*) –
- **train_obj** (*torch.Tensor*) –
- **train_con** (*Optional[torch.Tensor]*) –
- **bounds** (*torch.Tensor*) –

返回类型 torch.Tensor

optuna.integration.botorch.qparego_candidates_func

optuna.integration.botorch.**qparego_candidates_func**(*train_x*, *train_obj*, *train_con*, *bounds*)

Quasi MC-based extended ParEGO (qParEGO) for constrained multi-objective optimization.

The default value of `candidates_func` in *BoTorchSampler* with multi-objective optimization when the number of objectives is larger than three.

参见:

qei_candidates_func() for argument and return value descriptions.

备注: Added in v2.4.0 as an experimental feature. The interface may change in newer versions without prior notice. See https://github.com/optuna/optuna/releases/tag/v2.4.0.

参数

- **train_x** (*torch.Tensor*) –
- **train_obj** (*torch.Tensor*) –

- **train_con** (*Optional[torch.Tensor]*) –
- **bounds** (*torch.Tensor*) –

返回类型 torch.Tensor

Catalyst

optuna.integration.CatalystPruningCallback	用于清除无望 trial 的 Catalyst 回调。

optuna.integration.CatalystPruningCallback

class optuna.integration.**CatalystPruningCallback**(**args*, ***kwargs*)

Catalyst callback to prune unpromising trials.

This class is an alias to Catalyst's OptunaPruningCallback.

See the Catalyst's documentation for the detailed description.

> **警告:** Deprecated in v2.7.0. This feature will be removed in the future. The removal of this feature is currently scheduled for v4.0.0, but this schedule is subject to change. See https://github.com/optuna/optuna/releases/tag/v2.7.0.

Chainer

optuna.integration.ChainerPruningExtension	用于清除无望 trial 的 Chainer 扩展。
optuna.integration.ChainerMNStudy	一个将 Optuna 并入 ChainerMN 的*Study* wrapper.

optuna.integration.ChainerPruningExtension

class optuna.integration.**ChainerPruningExtension**(*trial*, *observation_key*, *pruner_trigger*)

Chainer extension to prune unpromising trials.

See the example if you want to add a pruning extension which observes validation accuracy of a Chainer Trainer.

参数

- **trial** (optuna.trial._trial.Trial) – A *Trial* corresponding to the current evaluation of the objective function.

- **observation_key** (*str*) – An evaluation metric for pruning, e.g., `main/loss` and `validation/main/accuracy`. Please refer to chainer.Reporter reference for further details.
- **pruner_trigger** (*Union[Tuple[int, str], IntervalTrigger, ManualScheduleTrigger]*) – A trigger to execute pruning. `pruner_trigger` is an instance of IntervalTrigger or ManualScheduleTrigger. IntervalTrigger can be specified by a tuple of the interval length and its unit like `(1, 'epoch')`.

返回类型 None

optuna.integration.ChainerMNStudy

class optuna.integration.**ChainerMNStudy**(*study*, *comm*)

A wrapper of *Study* to incorporate Optuna with ChainerMN.

参见:

ChainerMNStudy provides the same interface as *Study*. Please refer to *optuna.study.Study* for further details.

See the example if you want to optimize an objective function that trains neural network written with ChainerMN.

参数

- **study** (optuna.study.Study) – A *Study* object.
- **comm** (*CommunicatorBase*) – A ChainerMN communicator.

返回类型 None

Methods

optimize(func[, n_trials, timeout, catch])	Optimize an objective function.

optimize(*func*, *n_trials=None*, *timeout=None*, *catch=()*)

Optimize an objective function.

This method provides the same interface as *optuna.study.Study.optimize()* except the absence of `n_jobs` argument.

参数

- **func** (*Callable[[ChainerMNTrial, CommunicatorBase], float]*) –
- **n_trials** (*Optional[int]*) –
- **timeout** (*Optional[float]*) –

- **catch** (*Tuple[Type[Exception], ...]*) –

返回类型 None

fast.ai

optuna.integration.FastAIV1PruningCallback	用于清除 FastAI 中无望 trial 的回调函数。
optuna.integration.FastAIV2PruningCallback	用于清除 FastAI 中无望 trial 的回调函数。
optuna.integration.FastAIPruningCallback	:py:class:‘optuna.integration.fastaiv2.FastAIV2PruningCallback‘的别名

optuna.integration.FastAIV1PruningCallback

class optuna.integration.**FastAIV1PruningCallback**(*learn*, *trial*, *monitor*)

FastAI callback to prune unpromising trials for fastai.

备注： This callback is for fastai<2.0.

See the example if you want to add a pruning callback which monitors validation loss of a `Learner`.

示例

Register a pruning callback to `learn.fit` and `learn.fit_one_cycle`.

```
learn.fit(n_epochs, callbacks=[FastAIPruningCallback(learn, trial, "valid_loss")])
learn.fit_one_cycle(
    n_epochs,
    cyc_len,
    max_lr,
    callbacks=[FastAIPruningCallback(learn, trial, "valid_loss")],
)
```

参数

- **learn** (*Learner*) –fastai.basic_train.Learner.
- **trial** (optuna.trial._trial.Trial) –A *Trial* corresponding to the current evaluation of the objective function.

- **monitor** (*str*) –An evaluation metric for pruning, e.g. `valid_loss` and `Accuracy`. Please refer to fastai.callbacks.TrackerCallback reference for further details.

返回类型 None

警告： Deprecated in v2.4.0. This feature will be removed in the future. The removal of this feature is currently scheduled for v4.0.0, but this schedule is subject to change. See https://github.com/optuna/optuna/releases/tag/v2.4.0.

Methods

`on_epoch_end`(epoch, **kwargs)

optuna.integration.FastAIV2PruningCallback

class optuna.integration.**FastAIV2PruningCallback**(*trial*, *monitor='valid_loss'*)

FastAI callback to prune unpromising trials for fastai.

备注： This callback is for fastai>=2.0.

See the example if you want to add a pruning callback which monitors validation loss of a `Learner`.

示例

Register a pruning callback to `learn.fit` and `learn.fit_one_cycle`.

```
learn = cnn_learner(dls, resnet18, metrics=[error_rate])
learn.fit(n_epochs, cbs=[FastAIPruningCallback(trial)])  # Monitor "valid_loss"
learn.fit_one_cycle(
    n_epochs,
    lr_max,
    cbs=[FastAIPruningCallback(trial, monitor="error_rate")],  # Monitor "error_
↪rate"
)
```

参数

- **trial** (optuna.trial._trial.Trial) –A *Trial* corresponding to the current evaluation of the objective function.
- **monitor** (*str*) –An evaluation metric for pruning, e.g. `valid_loss` or `accuracy`. Please refer to fastai.callback.TrackerCallback reference for further details.

Methods

`after_epoch`()
`after_fit`()

optuna.integration.FastAIPruningCallback

optuna.integration.**FastAIPruningCallback**

:py:class:‘optuna.integration.fastaiv2.FastAIV2PruningCallback‘的别名

Keras

optuna.integration.KerasPruningCallback	用于清除无望 trial 的 Keras 回调函数。

optuna.integration.KerasPruningCallback

class optuna.integration.**KerasPruningCallback**(*trial*, *monitor*, *interval=1*)

Keras callback to prune unpromising trials.

See the example if you want to add a pruning callback which observes validation accuracy.

参数

- **trial** (optuna.trial._trial.Trial) –A *Trial* corresponding to the current evaluation of the objective function.
- **monitor** (*str*) –An evaluation metric for pruning, e.g., `val_loss` and `val_accuracy`. Please refer to keras.Callback reference for further details.
- **interval** (*int*) –Check if trial should be pruned every n-th epoch. By default `interval=1` and pruning is performed after every epoch. Increase `interval` to run several epochs faster before applying pruning.

返回类型 None

警告: Deprecated in v2.1.0. This feature will be removed in the future. The removal of this feature is currently scheduled for v4.0.0, but this schedule is subject to change. See https://github.com/optuna/optuna/releases/tag/v2.1.0.

Recent Keras release (2.4.0) simply redirects all APIs in the standalone keras package to point to tf.keras. There is now only one Keras: tf.keras. There may be some breaking changes for some workflows by upgrading to keras 2.4.0. Test before upgrading. REF:https://github.com/keras-team/keras/releases/tag/2.4.0

Methods

`on_epoch_end`(epoch[, logs])	

LightGBM

optuna.integration.LightGBMPruningCallback	用于清除无望 trial 的 LightGBM 回调函数。
optuna.integration.lightgbm.train	用于超参数调参的 LightGBM training API 的 wrapper.
optuna.integration.lightgbm.LightGBMTuner	用于 LightGBM 的超参数调参器。
optuna.integration.lightgbm.LightGBMTunerCV	带有交叉验证的、用于 LightGBM 的超参数调参器。

optuna.integration.LightGBMPruningCallback

class optuna.integration.**LightGBMPruningCallback**(*trial*, *metric*, *valid_name='valid_0'*)

Callback for LightGBM to prune unpromising trials.

See the example if you want to add a pruning callback which observes AUC of a LightGBM model.

参数

- **trial** (*optuna.trial._trial.Trial*) –A *Trial* corresponding to the current evaluation of the objective function.
- **metric** (*str*) –An evaluation metric for pruning, e.g., `binary_error` and `multi_error`. Please refer to LightGBM reference for further details.
- **valid_name** (*str*) –The name of the target validation. Validation names are specified by `valid_names` option of train method. If omitted, `valid_0` is used which is the default

name of the first validation. Note that this argument will be ignored if you are calling cv method instead of train method.

返回类型 None

optuna.integration.lightgbm.train

optuna.integration.lightgbm.**train**(**args*, ***kwargs*)

Wrapper of LightGBM Training API to tune hyperparameters.

It tunes important hyperparameters (e.g., `min_child_samples` and `feature_fraction`) in a stepwise manner. It is a drop-in replacement for lightgbm.train(). See a simple example of LightGBM Tuner which optimizes the validation log loss of cancer detection.

train() is a wrapper function of *LightGBMTuner*. To use feature in Optuna such as suspended/resumed optimization and/or parallelization, refer to *LightGBMTuner* instead of this function.

Arguments and keyword arguments for lightgbm.train() can be passed.

参数

- **args** (*Any*) –
- **kwargs** (*Any*) –

返回类型 *Any*

optuna.integration.lightgbm.LightGBMTuner

class optuna.integration.lightgbm.**LightGBMTuner**(*params*, *train_set*, *num_boost_round=1000*, *valid_sets=None*, *valid_names=None*, *fobj=None*, *feval=None*, *feature_name='auto'*, *categorical_feature='auto'*, *early_stopping_rounds=None*, *evals_result=None*, *verbose_eval=True*, *learning_rates=None*, *keep_training_booster=False*, *callbacks=None*, *time_budget=None*, *sample_size=None*, *study=None*, *optuna_callbacks=None*, *model_dir=None*, *verbosity=None*, *show_progress_bar=True*)

Hyperparameter tuner for LightGBM.

It optimizes the following hyperparameters in a stepwise manner: `lambda_l1`, `lambda_l2`, `num_leaves`, `feature_fraction`, `bagging_fraction`, `bagging_freq` and `min_child_samples`.

You can find the details of the algorithm and benchmark results in this blog article by Kohei Ozaki, a Kaggle Grandmaster.

Arguments and keyword arguments for lightgbm.train() can be passed. The arguments that only *LightGBMTuner* has are listed below:

参数

- **time_budget** (*Optional[int]*) –A time budget for parameter tuning in seconds.
- **study** (*Optional[optuna.study.Study]*) –A *Study* instance to store optimization results. The *Trial* instances in it has the following user attributes: `elapsed_secs` is the elapsed time since the optimization starts. `average_iteration_time` is the average time of iteration to train the booster model in the trial. `lgbm_params` is a JSON-serialized dictionary of LightGBM parameters used in the trial.
- **optuna_callbacks** (*Optional[List[Callable[[optuna.study.Study, optuna.trial._frozen.FrozenTrial], None]]]*) –List of Optuna callback functions that are invoked at the end of each trial. Each function must accept two parameters with the following types in this order: *Study* and `FrozenTrial`. Please note that this is not a `callbacks` argument of lightgbm.train() .
- **model_dir** (*Optional[str]*) –A directory to save boosters. By default, it is set to `None` and no boosters are saved. Please set shared directory (e.g., directories on NFS) if you want to access `get_best_booster()` in distributed environments. Otherwise, it may raise `ValueError`. If the directory does not exist, it will be created. The filenames of the boosters will be `{model_dir}/{trial_number}.pkl` (e.g., `./boosters/0.pkl`).
- **verbosity** (*Optional[int]*) –A verbosity level to change Optuna' s logging level. The level is aligned to LightGBM' s verbosity .

  警告: Deprecated in v2.0.0. `verbosity` argument will be removed in the future. The removal of this feature is currently scheduled for v4.0.0, but this schedule is subject to change.

  Please use *set_verbosity()* instead.

- **show_progress_bar** (*bool*) –Flag to show progress bars or not. To disable progress bar, set this `False`.

  备注: Progress bars will be fragmented by logging messages of LightGBM and Optuna. Please suppress such messages to show the progress bars properly.

- **params** (*Dict[str, Any]*) –

- **train_set** (*lgb.Dataset*) –
- **num_boost_round** (*int*) –
- **valid_sets** (*Optional[VALID_SET_TYPE]*) –
- **valid_names** (*Optional[Any]*) –
- **fobj** (*Optional[Callable[[...], Any]]*) –
- **feval** (*Optional[Callable[[...], Any]]*) –
- **feature_name** (*str*) –
- **categorical_feature** (*str*) –
- **early_stopping_rounds** (*Optional[int]*) –
- **evals_result** (*Optional[Dict[Any, Any]]*) –
- **verbose_eval** (*Optional[Union[bool, int]]*) –
- **learning_rates** (*Optional[List[float]]*) –
- **keep_training_booster** (*bool*) –
- **callbacks** (*Optional[List[Callable[[...], Any]]]*) –
- **sample_size** (*Optional[int]*) –

返回类型 None

Methods

compare_validation_metrics(val_score, best_score)	
get_best_booster()	Return the best booster.
higher_is_better()	
run()	Perform the hyperparameter-tuning with given parameters.
sample_train_set()	Make subset of *self.train_set* Dataset object.
tune_bagging([n_trials])	
tune_feature_fraction([n_trials])	
tune_feature_fraction_stage2([n_trials])	
tune_min_data_in_leaf()	
tune_num_leaves([n_trials])	
tune_regularization_factors([n_trials])	

Attributes

best_booster	Return the best booster.
best_params	Return parameters of the best booster.
best_score	Return the score of the best booster.

property best_booster: lightgbm.basic.Booster

Return the best booster.

> **警告:** Deprecated in v1.4.0. This feature will be removed in the future. The removal of this feature is currently scheduled for v3.0.0, but this schedule is subject to change. See https://github.com/optuna/optuna/releases/tag/v1.4.0.
>
> Please get the best booster via *get_best_booster* instead.

property best_params: Dict[str, Any]

Return parameters of the best booster.

property best_score: float

Return the score of the best booster.

get_best_booster()

Return the best booster.

If the best booster cannot be found, `ValueError` will be raised. To prevent the errors, please save boosters by specifying the `model_dir` argument of `__init__()`, when you resume tuning or you run tuning in parallel.

返回类型 lightgbm.basic.Booster

run()

Perform the hyperparameter-tuning with given parameters.

返回类型 None

sample_train_set()

Make subset of *self.train_set* Dataset object.

返回类型 None

optuna.integration.lightgbm.LightGBMTunerCV

class optuna.integration.lightgbm.**LightGBMTunerCV**(*params*, *train_set*, *num_boost_round=1000*, *folds=None*, *nfold=5*, *stratified=True*, *shuffle=True*, *fobj=None*, *feval=None*, *feature_name='auto'*, *categorical_feature='auto'*, *early_stopping_rounds=None*, *fpreproc=None*, *verbose_eval=True*, *show_stdv=True*, *seed=0*, *callbacks=None*, *time_budget=None*, *sample_size=None*, *study=None*, *optuna_callbacks=None*, *verbosity=None*, *show_progress_bar=True*, *model_dir=None*, *return_cvbooster=None*)

Hyperparameter tuner for LightGBM with cross-validation.

It employs the same stepwise approach as *LightGBMTuner*. *LightGBMTunerCV* invokes lightgbm.cv() to train and validate boosters while *LightGBMTuner* invokes lightgbm.train(). See a simple example which optimizes the validation log loss of cancer detection.

Arguments and keyword arguments for lightgbm.cv() can be passed except `metrics`, `init_model` and `eval_train_metric`. The arguments that only *LightGBMTunerCV* has are listed below:

参数

- **time_budget** (*Optional[int]*) –A time budget for parameter tuning in seconds.
- **study** (*Optional[optuna.study.Study]*) –A *Study* instance to store optimization results. The *Trial* instances in it has the following user attributes: `elapsed_secs` is the elapsed time since the optimization starts. `average_iteration_time` is the average time of iteration to train the booster model in the trial. `lgbm_params` is a JSON-serialized dictionary of LightGBM parameters used in the trial.
- **optuna_callbacks** (*Optional[List[Callable[[optuna.study.Study, optuna.trial._frozen.FrozenTrial], None]]]*) –List of Optuna callback functions that are invoked at the end of each trial. Each function must accept two parameters with the following types in this order: *Study* and `FrozenTrial`. Please note that this is not a `callbacks` argument of lightgbm.train() .
- **model_dir** (*Optional[str]*) –A directory to save boosters. By default, it is set to `None` and no boosters are saved. Please set shared directory (e.g., directories on NFS) if you want to access `get_best_booster()` in distributed environments. Otherwise, it may raise `ValueError`. If the directory does not exist, it will be created. The filenames of the boosters will be `{model_dir}/{trial_number}.pkl` (e.g., `./boosters/0.pkl`).
- **verbosity** (*Optional[int]*) –A verbosity level to change Optuna' s logging level. The level is aligned to LightGBM' s verbosity .

  警告: Deprecated in v2.0.0. `verbosity` argument will be removed in the future. The removal of this feature is currently scheduled for v4.0.0, but this schedule is subject to change.

  Please use *set_verbosity()* instead.

- **show_progress_bar** (*bool*) –Flag to show progress bars or not. To disable progress bar, set this `False`.

  备注: Progress bars will be fragmented by logging messages of LightGBM and Optuna. Please suppress such messages to show the progress bars properly.

- **return_cvbooster** (*Optional[bool]*) –Flag to enable `get_best_booster()`.
- **params** (*Dict[str, Any]*) –
- **train_set** (*lgb.Dataset*) –

- **num_boost_round** (*int*) –
- **folds** (*Optional[Union[Generator[Tuple[int, int], None, None], Iterator[Tuple[int, int]], BaseCrossValidator]]*) –
- **nfold** (*int*) –
- **stratified** (*bool*) –
- **shuffle** (*bool*) –
- **fobj** (*Optional[Callable[[...], Any]]*) –
- **feval** (*Optional[Callable[[...], Any]]*) –
- **feature_name** (*str*) –
- **categorical_feature** (*str*) –
- **early_stopping_rounds** (*Optional[int]*) –
- **fpreproc** (*Optional[Callable[[...], Any]]*) –
- **verbose_eval** (*Optional[Union[bool, int]]*) –
- **show_stdv** (*bool*) –
- **seed** (*int*) –
- **callbacks** (*Optional[List[Callable[[...], Any]]]*) –
- **sample_size** (*Optional[int]*) –

返回类型 None

Methods

compare_validation_metrics(val_score, best_score)	
get_best_booster()	Return the best cvbooster.
higher_is_better()	
run()	Perform the hyperparameter-tuning with given parameters.
sample_train_set()	Make subset of *self.train_set* Dataset object.
tune_bagging([n_trials])	
tune_feature_fraction([n_trials])	
tune_feature_fraction_stage2([n_trials])	
tune_min_data_in_leaf()	
tune_num_leaves([n_trials])	
tune_regularization_factors([n_trials])	

Attributes

best_params	Return parameters of the best booster.
best_score	Return the score of the best booster.

property best_params: Dict[str, Any]

Return parameters of the best booster.

property best_score: float

Return the score of the best booster.

get_best_booster()

Return the best cvbooster.

If the best booster cannot be found, ValueError will be raised. To prevent the errors, please save boosters by specifying both of the model_dir and the return_cvbooster arguments of __init__(), when you resume tuning or you run tuning in parallel.

返回类型 lightgbm.engine.CVBooster

run()

Perform the hyperparameter-tuning with given parameters.

返回类型 None

sample_train_set()

Make subset of *self.train_set* Dataset object.

返回类型 None

MLflow

optuna.integration.MLflowCallback	用 MLflow 来追踪 Optuna trial 的回调函数。

optuna.integration.MLflowCallback

class optuna.integration.**MLflowCallback**(*tracking_uri=None*, *metric_name='value'*, *nest_trials=False*, *tag_study_user_attrs=False*)

Callback to track Optuna trials with MLflow.

This callback adds relevant information that is tracked by Optuna to MLflow. The MLflow experiment will be named after the Optuna study name.

示例

Add MLflow callback to Optuna optimization.

```
import optuna
from optuna.integration.mlflow import MLflowCallback


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


mlflc = MLflowCallback(
    tracking_uri=YOUR_TRACKING_URI,
    metric_name="my metric score",
)

study = optuna.create_study(study_name="my_study")
study.optimize(objective, n_trials=10, callbacks=[mlflc])
```

参数

- **tracking_uri** (*Optional[str]*) –The URI of the MLflow tracking server.

  Please refer to mlflow.set_tracking_uri for more details.

- **metric_name** (*str*) –Name of the metric. Since the metric itself is just a number, *metric_name* can be used to give it a name. So you know later if it was roc-auc or accuracy.

- **nest_trials** (*bool*) –Flag indicating whether or not trials should be logged as nested runs. This is often helpful for aggregating trials to a particular study, under a given experiment.

- **tag_study_user_attrs** (*bool*) –Flag indicating whether or not to add the study's user attrs to the mlflow trial as tags. Please note that when this flag is set, key value pairs in study.user_attrs will supersede existing tags.

返回类型 None

备注: Added in v1.4.0 as an experimental feature. The interface may change in newer versions without prior notice. See https://github.com/optuna/optuna/releases/tag/v1.4.0.

MXNet

optuna.integration.MXNetPruningCallback	用于清除无望 trial 的 MXNet 回调函数。

optuna.integration.MXNetPruningCallback

class optuna.integration.**MXNetPruningCallback**(*trial*, *eval_metric*)

MXNet callback to prune unpromising trials.

See the example if you want to add a pruning callback which observes accuracy.

参数

- **trial** (optuna.trial._trial.Trial) –A *Trial* corresponding to the current evaluation of the objective function.

- **eval_metric** (*str*) –An evaluation metric name for pruning, e.g., `cross-entropy` and `accuracy`. If using default metrics like mxnet.metrics.Accuracy, use it's default metric name. For custom metrics, use the metric_name provided to constructor. Please refer to mxnet.metrics reference for further details.

返回类型 None

pycma

optuna.integration.PyCmaSampler	使用 cma 库作为后端的采样器。
optuna.integration.CmaEsSampler	用于向后兼容的 PyCmaSampler wrapper 类。

optuna.integration.PyCmaSampler

class optuna.integration.**PyCmaSampler**(*x0=None*, *sigma0=None*, *cma_stds=None*, *seed=None*, *cma_opts=None*, *n_startup_trials=1*, *independent_sampler=None*, *warn_independent_sampling=True*)

A Sampler using cma library as the backend.

示例

Optimize a simple quadratic function by using *PyCmaSampler*.

```
import optuna


def objective(trial):
    x = trial.suggest_float("x", -1, 1)
    y = trial.suggest_int("y", -1, 1)
    return x ** 2 + y


sampler = optuna.integration.PyCmaSampler()
study = optuna.create_study(sampler=sampler)
study.optimize(objective, n_trials=20)
```

Note that parallel execution of trials may affect the optimization performance of CMA-ES, especially if the number of trials running in parallel exceeds the population size.

备注： *CmaEsSampler* is deprecated and renamed to *PyCmaSampler* in v2.0.0. Please use *PyCmaSampler* instead of *CmaEsSampler*.

参数

- **x0** (*Optional[Dict[str, Any]]*) – A dictionary of an initial parameter values for CMA-ES. By default, the mean of `low` and `high` for each distribution is used. Please refer to cma.CMAEvolutionStrategy for further details of `x0`.

- **sigma0** (*Optional[float]*) –Initial standard deviation of CMA-ES. By default, `sigma0` is set to `min_range / 6`, where `min_range` denotes the minimum range of the distributions in the search space. If distribution is categorical, `min_range` is `len(choices) - 1`. Please refer to cma.CMAEvolutionStrategy for further details of `sigma0`.
- **cma_stds** (*Optional[Dict[str, float]]*) –A dictionary of multipliers of sigma0 for each parameters. The default value is 1.0. Please refer to cma.CMAEvolutionStrategy for further details of `cma_stds`.
- **seed** (*Optional[int]*) –A random seed for CMA-ES.
- **cma_opts** (*Optional[Dict[str, Any]]*) –Options passed to the constructor of cma.CMAEvolutionStrategy class.

  Note that `BoundaryHandler`, `bounds`, `CMA_stds` and `seed` arguments in `cma_opts` will be ignored because it is added by *PyCmaSampler* automatically.
- **n_startup_trials** (*int*) –The independent sampling is used instead of the CMA-ES algorithm until the given number of trials finish in the same study.
- **independent_sampler** (*Optional[optuna.samplers._base.BaseSampler]*) –A *BaseSampler* instance that is used for independent sampling. The parameters not contained in the relative search space are sampled by this sampler. The search space for *PyCmaSampler* is determined by *intersection_search_space()*.

  If `None` is specified, *RandomSampler* is used as the default.

  参见:

  optuna.samplers module provides built-in independent samplers such as *RandomSampler* and *TPESampler*.
- **warn_independent_sampling** (*bool*) –If this is `True`, a warning message is emitted when the value of a parameter is sampled by using an independent sampler.

  Note that the parameters of the first trial in a study are always sampled via an independent sampler, so no warning messages are emitted in this case.

返回类型 None

Methods

after_trial(study, trial, state, values)	Trial post-processing.
infer_relative_search_space(study, trial)	Infer the search space that will be used by relative sampling in the target trial.
reseed_rng()	Reseed sampler's random number generator.
sample_independent(study, trial, param_name, ...)	Sample a parameter for a given distribution.
sample_relative(study, trial, search_space)	Sample parameters in a given search space.

after_trial (*study*, *trial*, *state*, *values*)

Trial post-processing.

This method is called after the objective function returns and right before the trials is finished and its state is stored.

备注： Added in v2.4.0 as an experimental feature. The interface may change in newer versions without prior notice. See https://github.com/optuna/optuna/releases/tag/v2.4.0.

参数

- **study** (optuna.study.Study) –Target study object.
- **trial** (optuna.trial._frozen.FrozenTrial) –Target trial object. Take a copy before modifying this object.
- **state** (optuna.trial._state.TrialState) –Resulting trial state.
- **values** (*Optional[Sequence[float]]*) –Resulting trial values. Guaranteed to not be None if trial succeeded.

返回类型 None

infer_relative_search_space (*study*, *trial*)

Infer the search space that will be used by relative sampling in the target trial.

This method is called right before *sample_relative()* method, and the search space returned by this method is passed to it. The parameters not contained in the search space will be sampled by using *sample_independent()* method.

参数

- **study** (optuna.study.Study) –Target study object.
- **trial** (optuna.trial._frozen.FrozenTrial) –Target trial object. Take a copy before modifying this object.

返回 A dictionary containing the parameter names and parameter's distributions.

返回类型 *Dict*[str, optuna.distributions.BaseDistribution]

参见:

Please refer to *intersection_search_space()* as an implementation of *infer_relative_search_space()*.

reseed_rng()

Reseed sampler's random number generator.

This method is called by the *Study* instance if trials are executed in parallel with the option `n_jobs>1`. In that case, the sampler instance will be replicated including the state of the random number generator, and they may suggest the same values. To prevent this issue, this method assigns a different seed to each random number generator.

返回类型 None

sample_independent(*study*, *trial*, *param_name*, *param_distribution*)

Sample a parameter for a given distribution.

This method is called only for the parameters not contained in the search space returned by *sample_relative()* method. This method is suitable for sampling algorithms that do not use relationship between parameters such as random sampling and TPE.

备注: The failed trials are ignored by any build-in samplers when they sample new parameters. Thus, failed trials are regarded as deleted in the samplers' perspective.

参数

- **study** (`optuna.study.Study`) –Target study object.
- **trial** (`optuna.trial._frozen.FrozenTrial`) –Target trial object. Take a copy before modifying this object.
- **param_name** (*str*) –Name of the sampled parameter.
- **param_distribution** (*optuna.distributions.BaseDistribution*) – Distribution object that specifies a prior and/or scale of the sampling algorithm.

返回 A parameter value.

返回类型 float

sample_relative(*study*, *trial*, *search_space*)

Sample parameters in a given search space.

This method is called once at the beginning of each trial, i.e., right before the evaluation of the objective function. This method is suitable for sampling algorithms that use relationship between parameters such as Gaussian Process and CMA-ES.

备注： The failed trials are ignored by any build-in samplers when they sample new parameters. Thus, failed trials are regarded as deleted in the samplers' perspective.

参数

- **study** (optuna.study.Study) –Target study object.
- **trial** (optuna.trial._frozen.FrozenTrial) –Target trial object. Take a copy before modifying this object.
- **search_space** (*Dict[str, optuna.distributions.BaseDistribution]*) –The search space returned by *infer_relative_search_space()*.

返回 A dictionary containing the parameter names and the values.

返回类型 *Dict*[str, float]

optuna.integration.CmaEsSampler

class optuna.integration.**CmaEsSampler**(*x0=None*, *sigma0=None*, *cma_stds=None*, *seed=None*, *cma_opts=None*, *n_startup_trials=1*, *independent_sampler=None*, *warn_independent_sampling=True*)

Wrapper class of PyCmaSampler for backward compatibility.

警告： Deprecated in v2.0.0. This feature will be removed in the future. The removal of this feature is currently scheduled for v4.0.0, but this schedule is subject to change. See https://github.com/optuna/optuna/releases/tag/v2.0.0.

This class is renamed to *PyCmaSampler*.

Methods

after_trial(study, trial, state, values)	Trial post-processing.
infer_relative_search_space(study, trial)	Infer the search space that will be used by relative sampling in the target trial.
reseed_rng()	Reseed sampler's random number generator.
sample_independent(study, trial, param_name, ...)	Sample a parameter for a given distribution.
sample_relative(study, trial, search_space)	Sample parameters in a given search space.

参数

- **x0** (*Optional[Dict[str, Any]]*) –
- **sigma0** (*Optional[float]*) –
- **cma_stds** (*Optional[Dict[str, float]]*) –
- **seed** (*Optional[int]*) –
- **cma_opts** (*Optional[Dict[str, Any]]*) –
- **n_startup_trials** (*int*) –
- **independent_sampler** (*Optional[optuna.samplers._base.BaseSampler]*) –
- **warn_independent_sampling** (*bool*) –

返回类型 None

after_trial(*study*, *trial*, *state*, *values*)

Trial post-processing.

This method is called after the objective function returns and right before the trials is finished and its state is stored.

备注： Added in v2.4.0 as an experimental feature. The interface may change in newer versions without prior notice. See https://github.com/optuna/optuna/releases/tag/v2.4.0.

参数

- **study** (optuna.study.Study) –Target study object.
- **trial** (optuna.trial._frozen.FrozenTrial) –Target trial object. Take a copy before modifying this object.
- **state** (optuna.trial._state.TrialState) –Resulting trial state.

- **values** (*Optional[Sequence[float]]*) –Resulting trial values. Guaranteed to not be None if trial succeeded.

返回类型 None

infer_relative_search_space(*study*, *trial*)

Infer the search space that will be used by relative sampling in the target trial.

This method is called right before *sample_relative()* method, and the search space returned by this method is passed to it. The parameters not contained in the search space will be sampled by using *sample_independent()* method.

参数

- **study** (optuna.study.Study) –Target study object.
- **trial** (optuna.trial._frozen.FrozenTrial) –Target trial object. Take a copy before modifying this object.

返回 A dictionary containing the parameter names and parameter's distributions.

返回类型 *Dict*[str, optuna.distributions.BaseDistribution]

参见:

Please refer to *intersection_search_space()* as an implementation of *infer_relative_search_space()*.

reseed_rng()

Reseed sampler's random number generator.

This method is called by the *Study* instance if trials are executed in parallel with the option n_jobs>1. In that case, the sampler instance will be replicated including the state of the random number generator, and they may suggest the same values. To prevent this issue, this method assigns a different seed to each random number generator.

返回类型 None

sample_independent(*study*, *trial*, *param_name*, *param_distribution*)

Sample a parameter for a given distribution.

This method is called only for the parameters not contained in the search space returned by *sample_relative()* method. This method is suitable for sampling algorithms that do not use relationship between parameters such as random sampling and TPE.

备注: The failed trials are ignored by any build-in samplers when they sample new parameters. Thus, failed trials are regarded as deleted in the samplers' perspective.

参数

- **study** (optuna.study.Study) –Target study object.
- **trial** (optuna.trial._frozen.FrozenTrial) –Target trial object. Take a copy before modifying this object.
- **param_name** (*str*) –Name of the sampled parameter.
- **param_distribution** (*optuna.distributions.BaseDistribution*) – Distribution object that specifies a prior and/or scale of the sampling algorithm.

返回 A parameter value.

返回类型 float

sample_relative(*study*, *trial*, *search_space*)

Sample parameters in a given search space.

This method is called once at the beginning of each trial, i.e., right before the evaluation of the objective function. This method is suitable for sampling algorithms that use relationship between parameters such as Gaussian Process and CMA-ES.

备注： The failed trials are ignored by any build-in samplers when they sample new parameters. Thus, failed trials are regarded as deleted in the samplers' perspective.

参数

- **study** (optuna.study.Study) –Target study object.
- **trial** (optuna.trial._frozen.FrozenTrial) –Target trial object. Take a copy before modifying this object.
- **search_space** (*Dict[str, optuna.distributions.BaseDistribution]*) –The search space returned by *infer_relative_search_space()*.

返回 A dictionary containing the parameter names and the values.

返回类型 *Dict*[str, float]

PyTorch

optuna.integration.PyTorchIgnitePruningHandler	用于清除无望 trial 的 PyTorch Ignite handler。
optuna.integration.PyTorchLightningPruningCallback	用于清除无望 trial 的 PyTorch Lighting 回调函数。
optuna.integration.TorchDistributedTrial	A wrapper of *Trial* to incorporate Optuna with PyTorch distributed.

optuna.integration.PyTorchIgnitePruningHandler

class optuna.integration.**PyTorchIgnitePruningHandler**(*trial*, *metric*, *trainer*)

PyTorch Ignite handler to prune unpromising trials.

See the example if you want to add a pruning handler which observes validation accuracy.

参数

- **trial** (optuna.trial._trial.Trial) –A *Trial* corresponding to the current evaluation of the objective function.
- **metric** (*str*) –A name of metric for pruning, e.g., `accuracy` and `loss`.
- **trainer** (*Engine*) –A trainer engine of PyTorch Ignite. Please refer to ignite.engine.Engine reference for further details.

返回类型 None

optuna.integration.PyTorchLightningPruningCallback

class optuna.integration.**PyTorchLightningPruningCallback**(*trial*, *monitor*)

PyTorch Lightning callback to prune unpromising trials.

See the example if you want to add a pruning callback which observes accuracy.

参数

- **trial** (optuna.trial._trial.Trial) –A *Trial* corresponding to the current evaluation of the objective function.
- **monitor** (*str*) –An evaluation metric for pruning, e.g., `val_loss` or `val_acc`. The metrics are obtained from the returned dictionaries from e.g. `pytorch_lightning.LightningModule.training_step` or `pytorch_lightning.LightningModule.validation_epoch_end` and the names thus depend on how this dictionary is formatted.

返回类型 None

Methods

on_validation_end(trainer, pl_module)

optuna.integration.TorchDistributedTrial

class optuna.integration.**TorchDistributedTrial**(*trial*, *device=None*)

A wrapper of *Trial* to incorporate Optuna with PyTorch distributed.

参见:

TorchDistributedTrial provides the same interface as *Trial*. Please refer to *optuna.trial.Trial* for further details.

See the example if you want to optimize an objective function that trains neural network written with PyTorch distributed data parallel.

参数

- **trial** (*Optional[optuna.trial._trial.Trial]*) – A *Trial* object or None. Please set trial object in rank-0 node and set None in the other rank node.
- **device** (*Optional[torch.device]*) – A *torch.device* to communicate with the other nodes. Please set a CUDA device assigned to the current node if you use "nccl" as *torch.distributed* backend.

返回类型 None

备注: The methods of *TorchDistributedTrial* are expected to be called by all workers at once. They invoke synchronous data transmission to share processing results and synchronize timing.

备注: Added in v2.6.0 as an experimental feature. The interface may change in newer versions without prior notice. See https://github.com/optuna/optuna/releases/tag/v2.6.0.

Methods

report(value, step)
set_system_attr(key, value)
set_user_attr(key, value)
should_prune()
suggest_categorical(name, choices)
suggest_discrete_uniform(name, low, high, q)
suggest_float(name, low, high, *[, step, log])
suggest_int(name, low, high[, step, log])
suggest_loguniform(name, low, high)
suggest_uniform(name, low, high)

Attributes

datetime_start
distributions
number
params
system_attrs
user_attrs

scikit-learn

optuna.integration.OptunaSearchCV	带有交叉验证的超参数搜索。

optuna.integration.OptunaSearchCV

class optuna.integration.**OptunaSearchCV**(*estimator*, *param_distributions*, *cv=5*, *enable_pruning=False*, *error_score=nan*, *max_iter=1000*, *n_jobs=1*, *n_trials=10*, *random_state=None*, *refit=True*, *return_train_score=False*, *scoring=None*, *study=None*, *subsample=1.0*, *timeout=None*, *verbose=0*)

Hyperparameter search with cross-validation.

参数

- **estimator** (*BaseEstimator*) –Object to use to fit the data. This is assumed to implement the scikit-learn estimator interface. Either this needs to provide `score`, or `scoring` must be passed.
- **param_distributions** (*Mapping[str, optuna.distributions.BaseDistribution]*) –Dictionary where keys are parameters and values are distributions. Distributions are assumed to implement the optuna distribution interface.
- **cv** (*Optional[Union[BaseCrossValidator, int]]*) –Cross-validation strategy. Possible inputs for cv are:
  - integer to specify the number of folds in a CV splitter,
  - a CV splitter,
  - an iterable yielding (train, validation) splits as arrays of indices.

  For integer, if `estimator` is a classifier and `y` is either binary or multiclass, `sklearn.model_selection.StratifiedKFold` is used. otherwise, `sklearn.model_selection.KFold` is used.
- **enable_pruning** (*bool*) –If `True`, pruning is performed in the case where the underlying estimator supports `partial_fit`.
- **error_score** (*Union[numbers.Number, float, str]*) –Value to assign to the score if an error occurs in fitting. If ‘raise’, the error is raised. If numeric, `sklearn.exceptions.FitFailedWarning` is raised. This does not affect the refit step, which will always raise the error.
- **max_iter** (*int*) –Maximum number of epochs. This is only used if the underlying estimator supports `partial_fit`.

- **n_jobs** (*int*) –Number of `threading` based parallel jobs. `-1` means using the number is set to CPU count.

  备注: `n_jobs` allows parallelization using `threading` and may suffer from Python's GIL. It is recommended to use *process-based parallelization* if `func` is CPU bound.

  警告: Deprecated in v2.7.0. This feature will be removed in the future. It is recommended to use *process-based parallelization*. The removal of this feature is currently scheduled for v4.0.0, but this schedule is subject to change. See https://github.com/optuna/optuna/releases/tag/v2.7.0.

- **n_trials** (*int*) –Number of trials. If `None`, there is no limitation on the number of trials. If `timeout` is also set to `None`, the study continues to create trials until it receives a termination signal such as Ctrl+C or SIGTERM. This trades off runtime vs quality of the solution.
- **random_state** (*Optional[Union[int, numpy.random.mtrand.RandomState]]*) –Seed of the pseudo random number generator. If int, this is the seed used by the random number generator. If `numpy.random.RandomState` object, this is the random number generator. If `None`, the global random state from `numpy.random` is used.
- **refit** (*bool*) –If `True`, refit the estimator with the best found hyperparameters. The refitted estimator is made available at the `best_estimator_` attribute and permits using `predict` directly.
- **return_train_score** (*bool*) –If `True`, training scores will be included. Computing training scores is used to get insights on how different hyperparameter settings impact the overfitting/underfitting trade-off. However computing training scores can be computationally expensive and is not strictly required to select the hyperparameters that yield the best generalization performance.
- **scoring** (*Optional[Union[Callable[[...], float], str]]*) –String or callable to evaluate the predictions on the validation data. If `None`, `score` on the estimator is used.
- **study** (*Optional[optuna.study.Study]*) –Study corresponds to the optimization task. If `None`, a new study is created.
- **subsample** (*Union[float, int]*) –Proportion of samples that are used during hyperparameter search.
  - If int, then draw `subsample` samples.
  - If float, then draw `subsample` * `X.shape[0]` samples.

- **timeout** (*Optional[float]*) –Time limit in seconds for the search of appropriate models. If `None`, the study is executed without time limitation. If `n_trials` is also set to `None`, the study continues to create trials until it receives a termination signal such as Ctrl+C or SIGTERM. This trades off runtime vs quality of the solution.
- **verbose** (*int*) –Verbosity level. The higher, the more messages.

返回类型 None

best_estimator_

Estimator that was chosen by the search. This is present only if `refit` is set to `True`.

n_splits_

Number of cross-validation splits.

refit_time_

Time for refitting the best estimator. This is present only if `refit` is set to `True`.

sample_indices_

Indices of samples that are used during hyperparameter search.

scorer_

Scorer function.

study_

Actual study.

实际案例

```
import optuna
from sklearn.datasets import load_iris
from sklearn.svm import SVC

clf = SVC(gamma="auto")
param_distributions = {"C": optuna.distributions.LogUniformDistribution(1e-10,␣
↪1e10)}
optuna_search = optuna.integration.OptunaSearchCV(clf, param_distributions)
X, y = load_iris(return_X_y=True)
optuna_search.fit(X, y)
y_pred = optuna_search.predict(X)
```

备注： Added in v0.17.0 as an experimental feature. The interface may change in newer versions without prior notice. See https://github.com/optuna/optuna/releases/tag/v0.17.0.

Methods

fit(X[, y, groups])	Run fit with all sets of parameters.
get_params([deep])	Get parameters for this estimator.
score(X[, y])	Return the score on the given data.
set_params(**params)	Set the parameters of this estimator.

Attributes

*best_index_*	Index which corresponds to the best candidate parameter setting.
*best_params_*	Parameters of the best trial in the *Study*.
*best_score_*	Mean cross-validated score of the best estimator.
*best_trial_*	Best trial in the *Study*.
*classes_*	Class labels.
decision_function	Call `decision_function` on the best estimator.
inverse_transform	Call `inverse_transform` on the best estimator.
*n_trials_*	Actual number of trials.
predict	Call `predict` on the best estimator.
predict_log_proba	Call `predict_log_proba` on the best estimator.
predict_proba	Call `predict_proba` on the best estimator.
score_samples	Call `score_samples` on the best estimator.
set_user_attr	Call `set_user_attr` on the *Study*.
transform	Call `transform` on the best estimator.
*trials_*	All trials in the *Study*.
trials_dataframe	Call `trials_dataframe` on the *Study*.
*user_attrs_*	User attributes in the *Study*.

property best_index_: int

Index which corresponds to the best candidate parameter setting.

property best_params_: Dict[str, Any]

Parameters of the best trial in the *Study*.

property best_score_: float

Mean cross-validated score of the best estimator.

property best_trial_: *optuna.trial._frozen.FrozenTrial*

Best trial in the *Study*.

property classes_: Union[List[float], numpy.ndarray, pandas.core.series.Series]

Class labels.

property decision_function: Callable[[...], Union[List[float], numpy.ndarray, pandas.core.series.Series, List[List[float]], pandas.core.frame.DataFrame, scipy.sparse.base.spmatrix]]

Call `decision_function` on the best estimator.

This is available only if the underlying estimator supports `decision_function` and `refit` is set to `True`.

fit(*X*, *y=None*, *groups=None*, ***fit_params*)

Run fit with all sets of parameters.

参数

- **X** (*Union[List[List[float]], numpy.ndarray, pandas.core.frame.DataFrame, scipy.sparse.base.spmatrix]*) –Training data.
- **y** (*Optional[Union[List[float], numpy.ndarray, pandas.core.series.Series, List[List[float]], pandas.core.frame.DataFrame, scipy.sparse.base.spmatrix]]*) –Target variable.
- **groups** (*Optional[Union[List[float], numpy.ndarray, pandas.core.series.Series]]*) –Group labels for the samples used while splitting the dataset into train/validation set.
- ****fit_params** –Parameters passed to `fit` on the estimator.
- **fit_params** (*Any*) –

返回 Return self.

返回类型 self

get_params(*deep=True*)

Get parameters for this estimator.

参数 **deep** (*bool, default=True*) –If True, will return the parameters for this estimator and contained subobjects that are estimators.

返回 **params** –Parameter names mapped to their values.

返回类型 dict

property inverse_transform: Callable[[...], Union[List[List[float]], numpy.ndarray, pandas.core.frame.DataFrame, scipy.sparse.base.spmatrix]]

Call `inverse_transform` on the best estimator.

This is available only if the underlying estimator supports `inverse_transform` and `refit` is set to `True`.

property n_trials_: int

Actual number of trials.

property predict: Callable[[...], Union[List[float], numpy.ndarray, pandas.core.series.Series, List[List[float]], pandas.core.frame.DataFrame, scipy.sparse.base.spmatrix]]

Call `predict` on the best estimator.

This is available only if the underlying estimator supports `predict` and `refit` is set to `True`.

property predict_log_proba: Callable[[...], Union[List[List[float]], numpy.ndarray, pandas.core.frame.DataFrame, scipy.sparse.base.spmatrix]]

Call `predict_log_proba` on the best estimator.

This is available only if the underlying estimator supports `predict_log_proba` and `refit` is set to `True`.

property predict_proba: Callable[[...], Union[List[List[float]], numpy.ndarray, pandas.core.frame.DataFrame, scipy.sparse.base.spmatrix]]

Call `predict_proba` on the best estimator.

This is available only if the underlying estimator supports `predict_proba` and `refit` is set to `True`.

score(*X*, *y=None*)

Return the score on the given data.

参数

- **X** (*Union[List[List[float]], numpy.ndarray, pandas.core.frame.DataFrame, scipy.sparse.base.spmatrix]*) –Data.
- **y** (*Optional[Union[List[float], numpy.ndarray, pandas.core.series.Series, List[List[float]], pandas.core.frame.DataFrame, scipy.sparse.base.spmatrix]]*) –Target variable.

返回 Scaler score.

返回类型 score

property score_samples: Callable[[...], Union[List[float], numpy.ndarray, pandas.core.series.Series]]

Call `score_samples` on the best estimator.

This is available only if the underlying estimator supports `score_samples` and `refit` is set to `True`.

set_params(***params*)

Set the parameters of this estimator.

The method works on simple estimators as well as on nested objects (such as `Pipeline`). The latter have parameters of the form `<component>__<parameter>` so that it's possible to update each component of a nested object.

参数 ****params** (*dict*) –Estimator parameters.

返回 **self** –Estimator instance.

返回类型 estimator instance

property set_user_attr: Callable[[...], None]

Call `set_user_attr` on the *Study*.

property transform: Callable[[...], Union[List[List[float]], numpy.ndarray, pandas.core.frame.DataFrame, scipy.sparse.base.spmatrix]]

Call `transform` on the best estimator.

This is available only if the underlying estimator supports `transform` and `refit` is set to `True`.

property trials_: List[*optuna.trial._frozen.FrozenTrial*]

All trials in the *Study*.

property trials_dataframe: Callable[[...], pandas.core.frame.DataFrame]

Call `trials_dataframe` on the *Study*.

property user_attrs_: Dict[str, Any]

User attributes in the *Study*.

scikit-optimize

optuna.integration.SkoptSampler	以 Scikit-Optimize 为后端的 sampler。

optuna.integration.SkoptSampler

class optuna.integration.**SkoptSampler**(*independent_sampler=None*, *warn_independent_sampling=True*, *skopt_kwargs=None*, *n_startup_trials=1*, ***, *consider_pruned_trials=False*)

Sampler using Scikit-Optimize as the backend.

示例

Optimize a simple quadratic function by using *SkoptSampler*.

```
import optuna


def objective(trial):
    x = trial.suggest_float("x", -10, 10)
    y = trial.suggest_int("y", 0, 10)
    return x ** 2 + y


sampler = optuna.integration.SkoptSampler()
study = optuna.create_study(sampler=sampler)
study.optimize(objective, n_trials=10)
```

参数

- **independent_sampler** (*Optional[optuna.samplers._base.BaseSampler]*) –A *BaseSampler* instance that is used for independent sampling. The parameters not contained in the relative search space are sampled by this sampler. The search space for *SkoptSampler* is determined by *intersection_search_space()*.

  If `None` is specified, *RandomSampler* is used as the default.

  参见:

  optuna.samplers module provides built-in independent samplers such as *RandomSampler* and *TPESampler*.

- **warn_independent_sampling** (*bool*) –If this is `True`, a warning message is emitted when the value of a parameter is sampled by using an independent sampler.

  Note that the parameters of the first trial in a study are always sampled via an independent sampler, so no warning messages are emitted in this case.

- **skopt_kwargs** (*Optional[Dict[str, Any]]*) –Keyword arguments passed to the constructor of skopt.Optimizer class.

  Note that `dimensions` argument in `skopt_kwargs` will be ignored because it is added by *SkoptSampler* automatically.

- **n_startup_trials** (*int*) –The independent sampling is used until the given number of trials finish in the same study.

- **consider_pruned_trials** (*bool*) –If this is `True`, the PRUNED trials are considered for sampling.

备注： Added in v2.0.0 as an experimental feature. The interface may change in newer versions without prior notice. See https://github.com/optuna/optuna/releases/tag/v2.0.0.

备注： As the number of trials n increases, each sampling takes longer and longer on a scale of $O(n^3)$. And, if this is `True`, the number of trials will increase. So, it is suggested to set this flag `False` when each evaluation of the objective function is relatively faster than each sampling. On the other hand, it is suggested to set this flag `True` when each evaluation of the objective function is relatively slower than each sampling.

返回类型 None

Methods

after_trial(study, trial, state, values)	Trial post-processing.
infer_relative_search_space(study, trial)	Infer the search space that will be used by relative sampling in the target trial.
reseed_rng()	Reseed sampler's random number generator.
sample_independent(study, trial, param_name, ...)	Sample a parameter for a given distribution.
sample_relative(study, trial, search_space)	Sample parameters in a given search space.

after_trial (*study*, *trial*, *state*, *values*)

Trial post-processing.

This method is called after the objective function returns and right before the trials is finished and its state is stored.

备注： Added in v2.4.0 as an experimental feature. The interface may change in newer versions without prior notice. See https://github.com/optuna/optuna/releases/tag/v2.4.0.

参数

- **study** (optuna.study.Study) –Target study object.
- **trial** (optuna.trial._frozen.FrozenTrial) –Target trial object. Take a copy before modifying this object.
- **state** (optuna.trial._state.TrialState) –Resulting trial state.

- **values** (*Optional[Sequence[float]]*) –Resulting trial values. Guaranteed to not be None if trial succeeded.

返回类型 None

infer_relative_search_space(*study*, *trial*)

Infer the search space that will be used by relative sampling in the target trial.

This method is called right before *sample_relative()* method, and the search space returned by this method is passed to it. The parameters not contained in the search space will be sampled by using *sample_independent()* method.

参数

- **study** (optuna.study.Study) –Target study object.
- **trial** (optuna.trial._frozen.FrozenTrial) –Target trial object. Take a copy before modifying this object.

返回 A dictionary containing the parameter names and parameter's distributions.

返回类型 *Dict*[str, optuna.distributions.BaseDistribution]

参见:

Please refer to *intersection_search_space()* as an implementation of *infer_relative_search_space()*.

reseed_rng()

Reseed sampler's random number generator.

This method is called by the *Study* instance if trials are executed in parallel with the option n_jobs>1. In that case, the sampler instance will be replicated including the state of the random number generator, and they may suggest the same values. To prevent this issue, this method assigns a different seed to each random number generator.

返回类型 None

sample_independent(*study*, *trial*, *param_name*, *param_distribution*)

Sample a parameter for a given distribution.

This method is called only for the parameters not contained in the search space returned by *sample_relative()* method. This method is suitable for sampling algorithms that do not use relationship between parameters such as random sampling and TPE.

备注: The failed trials are ignored by any build-in samplers when they sample new parameters. Thus, failed trials are regarded as deleted in the samplers' perspective.

参数

- **study** (optuna.study.Study) –Target study object.
- **trial** (optuna.trial._frozen.FrozenTrial) –Target trial object. Take a copy before modifying this object.
- **param_name** (*str*) –Name of the sampled parameter.
- **param_distribution** (*optuna.distributions.BaseDistribution*) – Distribution object that specifies a prior and/or scale of the sampling algorithm.

返回 A parameter value.

返回类型 *Any*

sample_relative(*study*, *trial*, *search_space*)

Sample parameters in a given search space.

This method is called once at the beginning of each trial, i.e., right before the evaluation of the objective function. This method is suitable for sampling algorithms that use relationship between parameters such as Gaussian Process and CMA-ES.

备注： The failed trials are ignored by any build-in samplers when they sample new parameters. Thus, failed trials are regarded as deleted in the samplers' perspective.

参数

- **study** (optuna.study.Study) –Target study object.
- **trial** (optuna.trial._frozen.FrozenTrial) –Target trial object. Take a copy before modifying this object.
- **search_space** (*Dict[str, optuna.distributions.BaseDistribution]*) –The search space returned by *infer_relative_search_space()*.

返回 A dictionary containing the parameter names and the values.

返回类型 *Dict*[str, *Any*]

skorch

optuna.integration.SkorchPruningCallback	用于清除无望 trial 的 Skorch 回调函数。

optuna.integration.SkorchPruningCallback

class optuna.integration.**SkorchPruningCallback**(*trial*, *monitor*)

Skorch callback to prune unpromising trials.

2.1.0 新版功能.

参数

- **trial** (optuna.trial._trial.Trial) –A *Trial* corresponding to the current evaluation of the objective function.
- **monitor** (*str*) –An evaluation metric for pruning, e.g. `val_loss` or `val_acc`. The metrics are obtained from the returned dictionaries, i.e., `net.histroy`. The names thus depend on how this dictionary is formatted.

返回类型 None

Methods

on_epoch_end(net, **kwargs)

TensorFlow

optuna.integration.TensorBoardCallback	使用 TensorBoard 追踪 Optuna trial 的回调函数。
optuna.integration.TensorFlowPruningHook	用于清除无望 trial 的 TensorFlow SessionRunHook 钩子函数。
optuna.integration.TFKerasPruningCallback	用于清除无望 trial 的 tf.keras 回调函数。

optuna.integration.TensorBoardCallback

class optuna.integration.**TensorBoardCallback**(*dirname*, *metric_name*)

Callback to track Optuna trials with TensorBoard.

This callback adds relevant information that is tracked by Optuna to TensorBoard.

See the example.

参数

- **dirname** (*str*) –Directory to store TensorBoard logs.

- **metric_name** (*str*) –Name of the metric. Since the metric itself is just a number, *metric_name* can be used to give it a name. So you know later if it was roc-auc or accuracy.

返回类型 None

备注： Added in v2.0.0 as an experimental feature. The interface may change in newer versions without prior notice. See https://github.com/optuna/optuna/releases/tag/v2.0.0.

---

optuna.integration.TensorFlowPruningHook

class optuna.integration.**TensorFlowPruningHook**(*trial*, *estimator*, *metric*, *run_every_steps*)

TensorFlow SessionRunHook to prune unpromising trials.

See the example if you want to add a pruning hook to TensorFlow’s estimator.

参数

- **trial** (optuna.trial._trial.Trial) –A *Trial* corresponding to the current evaluation of the objective function.
- **estimator** (*tf.estimator.Estimator*) –An estimator which you will use.
- **metric** (*str*) –An evaluation metric for pruning, e.g., `accuracy` and `loss`.
- **run_every_steps** (*int*) –An interval to watch the summary file.

返回类型 None

Methods

after_run(run_context, run_values)
before_run(run_context)
begin()

optuna.integration.TFKerasPruningCallback

class optuna.integration.**TFKerasPruningCallback**(*trial*, *monitor*)

tf.keras callback to prune unpromising trials.

This callback is intend to be compatible for TensorFlow v1 and v2, but only tested with TensorFlow v2.

See the example if you want to add a pruning callback which observes the validation accuracy.

参数

- **trial** (optuna.trial._trial.Trial) –A *Trial* corresponding to the current evaluation of the objective function.
- **monitor** (*str*) –An evaluation metric for pruning, e.g., `val_loss` or `val_acc`.

返回类型 None

Methods

on_epoch_end(epoch[, logs])	

XGBoost

optuna.integration.XGBoostPruningCallback	用于清除无望 trial 的 XGBoost 回调函数。

optuna.integration.XGBoostPruningCallback

class optuna.integration.**XGBoostPruningCallback**(*trial*, *observation_key*)

Callback for XGBoost to prune unpromising trials.

See the example if you want to add a pruning callback which observes validation AUC of a XGBoost model.

参数

- **trial** (optuna.trial._trial.Trial) –A *Trial* corresponding to the current evaluation of the objective function.
- **observation_key** (*str*) –An evaluation metric for pruning, e.g., `validation-error` and `validation-merror`. When using the Scikit-Learn API, the index number of `eval_set` must be included in the `observation_key`, e.g., `validation_0-error` and `validation_0-merror`. Please refer to `eval_metric` in XGBoost reference for further details.

返回类型 None

6.3.7 optuna.logging

logging 模块使用 Python `logging` 包来实现日志记录。库用户可使用*set_verbosity()* 设置 verbosity 级别为 `optuna.logging.CRITICAL` (又名 `optuna.logging.FATAL`)、`optuna.logging.ERROR`、`optuna.logging.WARNING` (又名 `optuna.logging.WARN`)、`optuna.logging.INFO` 或 `optuna.logging.DEBUG` 。

optuna.logging.get_verbosity	返回目前 Optuna 的 root logger 的 logging 层级。
optuna.logging.set_verbosity	设置 Optuna 的 root logger 的 logging 层级。
optuna.logging.disable_default_handler	禁用 Optuna 的 root logger 的默认句柄。
optuna.logging.enable_default_handler	启用 Optuna 的 root logger 的默认句柄。
optuna.logging.disable_propagation	禁用库 log 输出的传递。
optuna.logging.enable_propagation	启用库 log 输出的传递。

optuna.logging.get_verbosity

optuna.logging.**get_verbosity**()

Return the current level for the Optuna' s root logger.

返回 Logging level, e.g., `optuna.logging.DEBUG` and `optuna.logging.INFO`.

返回类型 int

备注： Optuna has following logging levels:

- `optuna.logging.CRITICAL`, `optuna.logging.FATAL`
- `optuna.logging.ERROR`
- `optuna.logging.WARNING`, `optuna.logging.WARN`
- `optuna.logging.INFO`
- `optuna.logging.DEBUG`

---

optuna.logging.set_verbosity

optuna.logging.**set_verbosity**(*verbosity*)

Set the level for the Optuna' s root logger.

参数 **verbosity** (*int*) –Logging level, e.g., optuna.logging.DEBUG and optuna.logging.INFO.

返回类型 None

备注： Optuna has following logging levels:

- optuna.logging.CRITICAL, optuna.logging.FATAL
- optuna.logging.ERROR
- optuna.logging.WARNING, optuna.logging.WARN
- optuna.logging.INFO
- optuna.logging.DEBUG

---

optuna.logging.disable_default_handler

optuna.logging.**disable_default_handler**()

Disable the default handler of the Optuna' s root logger.

示例

Stop and then resume logging to sys.stderr.

```
import optuna

study = optuna.create_study()

# There are no logs in sys.stderr.
optuna.logging.disable_default_handler()
study.optimize(objective, n_trials=10)

# There are logs in sys.stderr.
optuna.logging.enable_default_handler()
study.optimize(objective, n_trials=10)
# [I 2020-02-23 17:00:54,314] Trial 10 finished with value: ...
# [I 2020-02-23 17:00:54,356] Trial 11 finished with value: ...
# ...
```

返回类型 None

optuna.logging.enable_default_handler

optuna.logging.**enable_default_handler**()

Enable the default handler of the Optuna’s root logger.

Please refer to the example shown in *disable_default_handler()*.

返回类型 None

optuna.logging.disable_propagation

optuna.logging.**disable_propagation**()

Disable propagation of the library log outputs.

Note that log propagation is disabled by default.

返回类型 None

optuna.logging.enable_propagation

optuna.logging.**enable_propagation**()

Enable propagation of the library log outputs.

Please disable the Optuna’s default handler to prevent double logging if the root logger has been configured.

示例

Propagate all log output to the root logger in order to save them to the file.

```
import optuna
import logging

logger = logging.getLogger()

logger.setLevel(logging.INFO)  # Setup the root logger.
logger.addHandler(logging.FileHandler("foo.log", mode="w"))

optuna.logging.enable_propagation()  # Propagate logs to the root logger.
optuna.logging.disable_default_handler()  # Stop showing logs in sys.stderr.

study = optuna.create_study()
```

(下页继续)

(续上页)

```
logger.info("Start optimization.")
study.optimize(objective, n_trials=10)

with open("foo.log") as f:
    assert f.readline().startswith("A new study created")
    assert f.readline() == "Start optimization.\n"
```

返回类型 None

6.3.8 optuna.multi_objective

该模块已被弃用，其功能已经被迁移到*optuna.samplers*, *optuna.study*, *optuna.trial* 和*optuna.visualization* 中。

multi-objective sampler 类

optuna.multi_objective.samplers.BaseMultiObjectiveSampler	multi-objective sampler 基类。
optuna.multi_objective.samplers.NSGAIIMultiObjectiveSampler	使用 NSGA-II 算法的 Multi-objective sampler。
optuna.multi_objective.samplers.RandomMultiObjectiveSampler	Multi-objective sampler using random sampling.
optuna.multi_objective.samplers.MOTPEMultiObjectiveSampler	采用 MOTPE 算法的多目标 sampler。

optuna.multi_objective.samplers.BaseMultiObjectiveSampler

class optuna.multi_objective.samplers.**BaseMultiObjectiveSampler**(**args*, ***kwargs*)

Base class for multi-objective samplers.

The abstract methods of this class are the same as ones defined by *BaseSampler* except for taking multi-objective versions of study and trial instances as the arguments.

警告： Deprecated in v2.4.0. This feature will be removed in the future. The removal of this feature is currently scheduled for v4.0.0, but this schedule is subject to change. See https://github.com/optuna/optuna/releases/tag/v2.4.0.

Methods

infer_relative_search_space(study, trial)	Infer the search space that will be used by relative sampling in the target trial.
reseed_rng()	Reseed sampler's random number generator.
sample_independent(study, trial, param_name, ...)	Sample a parameter for a given distribution.
sample_relative(study, trial, search_space)	Sample parameters in a given search space.

abstract infer_relative_search_space(*study*, *trial*)

Infer the search space that will be used by relative sampling in the target trial.

This method is called right before *sample_relative()* method, and the search space returned by this method is passed to it. The parameters not contained in the search space will be sampled by using *sample_independent()* method.

参数

- **study** (optuna.multi_objective.study.MultiObjectiveStudy) – Target study object.
- **trial** (optuna.multi_objective.trial.FrozenMultiObjectiveTrial) – Target trial object.

返回 A dictionary containing the parameter names and parameter's distributions.

返回类型 *Dict*[str, optuna.distributions.BaseDistribution]

参见:

Please refer to *intersection_search_space()* as an implementation of *infer_relative_search_space()*.

reseed_rng()

Reseed sampler's random number generator.

This method is called by the *MultiObjectiveStudy* instance if trials are executed in parallel with the option `n_jobs>1`. In that case, the sampler instance will be replicated including the state of the random number generator, and they may suggest the same values. To prevent this issue, this method assigns a different seed to each random number generator.

返回类型 None

abstract sample_independent(*study*, *trial*, *param_name*, *param_distribution*)

Sample a parameter for a given distribution.

This method is called only for the parameters not contained in the search space returned by `sample_relative()` method. This method is suitable for sampling algorithms that do not use the relationship between parameters such as random sampling.

参数

- **study** (optuna.multi_objective.study.MultiObjectiveStudy) –Target study object.
- **trial** (optuna.multi_objective.trial.FrozenMultiObjectiveTrial) –Target trial object.
- **param_name** (*str*) –Name of the sampled parameter.
- **param_distribution** (*optuna.distributions.BaseDistribution*) – Distribution object that specifies a prior and/or scale of the sampling algorithm.

返回 A parameter value.

返回类型 *Any*

abstract sample_relative(*study*, *trial*, *search_space*)

Sample parameters in a given search space.

This method is called once at the beginning of each trial, i.e., right before the evaluation of the objective function. This method is suitable for sampling algorithms that use the relationship between parameters.

参数

- **study** (optuna.multi_objective.study.MultiObjectiveStudy) –Target study object.
- **trial** (optuna.multi_objective.trial.FrozenMultiObjectiveTrial) –Target trial object.
- **search_space** (*Dict[str, optuna.distributions.BaseDistribution]*) –The search space returned by *infer_relative_search_space()*.

返回 A dictionary containing the parameter names and the values.

返回类型 *Dict*[str, *Any*]

optuna.multi_objective.samplers.NSGAIIMultiObjectiveSampler

class optuna.multi_objective.samplers.**NSGAIIMultiObjectiveSampler**(*population_size=50*, *mutation_prob=None*, *crossover_prob=0.9*, *swapping_prob=0.5*, *seed=None*)

Multi-objective sampler using the NSGA-II algorithm.

NSGA-II stands for “Nondominated Sorting Genetic Algorithm II”, which is a well known, fast and elitist multi-objective genetic algorithm.

For further information about NSGA-II, please refer to the following paper:

- A fast and elitist multiobjective genetic algorithm: NSGA-II

参数

- **population_size** (*int*) –Number of individuals (trials) in a generation.
- **mutation_prob** (*Optional[float]*) –Probability of mutating each parameter when creating a new individual. If `None` is specified, the value `1.0 / len(parent_trial.params)` is used where `parent_trial` is the parent trial of the target individual.
- **crossover_prob** (*float*) –Probability that a crossover (parameters swapping between parents) will occur when creating a new individual.
- **swapping_prob** (*float*) –Probability of swapping each parameter of the parents during crossover.
- **seed** (*Optional[int]*) –Seed for random number generator.

返回类型 None

警告: Deprecated in v2.4.0. This feature will be removed in the future. The removal of this feature is currently scheduled for v4.0.0, but this schedule is subject to change. See https://github.com/optuna/optuna/releases/tag/v2.4.0.

Methods

infer_relative_search_space(study, trial)	Infer the search space that will be used by relative sampling in the target trial.
reseed_rng()	Reseed sampler's random number generator.
sample_independent(study, trial, param_name, ...)	Sample a parameter for a given distribution.
sample_relative(study, trial, search_space)	Sample parameters in a given search space.

infer_relative_search_space (*study*, *trial*)

Infer the search space that will be used by relative sampling in the target trial.

This method is called right before *sample_relative()* method, and the search space returned by this method is passed to it. The parameters not contained in the search space will be sampled by using *sample_independent()* method.

参数

- **study** (optuna.multi_objective.study.MultiObjectiveStudy) –Target study object.
- **trial** (optuna.multi_objective.trial.FrozenMultiObjectiveTrial) –Target trial object.

返回 A dictionary containing the parameter names and parameter's distributions.

返回类型 *Dict*[str, optuna.distributions.BaseDistribution]

参见:

Please refer to *intersection_search_space()* as an implementation of *infer_relative_search_space()*.

reseed_rng()

Reseed sampler's random number generator.

This method is called by the *MultiObjectiveStudy* instance if trials are executed in parallel with the option `n_jobs>1`. In that case, the sampler instance will be replicated including the state of the random number generator, and they may suggest the same values. To prevent this issue, this method assigns a different seed to each random number generator.

返回类型 None

sample_independent(*study*, *trial*, *param_name*, *param_distribution*)

Sample a parameter for a given distribution.

This method is called only for the parameters not contained in the search space returned by `sample_relative()` method. This method is suitable for sampling algorithms that do not use the relationship between parameters such as random sampling.

参数

- **study** (optuna.multi_objective.study.MultiObjectiveStudy) –Target study object.
- **trial** (optuna.multi_objective.trial.FrozenMultiObjectiveTrial) –Target trial object.
- **param_name** (*str*) –Name of the sampled parameter.
- **param_distribution** (*optuna.distributions.BaseDistribution*) – Distribution object that specifies a prior and/or scale of the sampling algorithm.

返回 A parameter value.

返回类型 *Any*

sample_relative(*study*, *trial*, *search_space*)

Sample parameters in a given search space.

This method is called once at the beginning of each trial, i.e., right before the evaluation of the objective function. This method is suitable for sampling algorithms that use the relationship between parameters.

参数

- **study**(optuna.multi_objective.study.MultiObjectiveStudy)–Target study object.
- **trial**(optuna.multi_objective.trial.FrozenMultiObjectiveTrial)–Target trial object.
- **search_space**(*Dict[str, optuna.distributions.BaseDistribution]*)–The search space returned by *infer_relative_search_space()*.

返回 A dictionary containing the parameter names and the values.

返回类型 *Dict*[str, *Any*]

optuna.multi_objective.samplers.RandomMultiObjectiveSampler

class optuna.multi_objective.samplers.**RandomMultiObjectiveSampler**(*seed=None*)

Multi-objective sampler using random sampling.

This sampler is based on *independent sampling*. See also *BaseMultiObjectiveSampler* for more details of ‘independent sampling’.

示例

```
import optuna
from optuna.multi_objective.samplers import RandomMultiObjectiveSampler


def objective(trial):
    x = trial.suggest_float("x", -5, 5)
    y = trial.suggest_float("y", -5, 5)
    return x ** 2, y + 10


study = optuna.multi_objective.create_study(
    ["minimize", "minimize"], sampler=RandomMultiObjectiveSampler()
)
study.optimize(objective, n_trials=10)
```

Args: seed: Seed for random number generator.

> **警告:** Deprecated in v2.4.0. This feature will be removed in the future. The removal of this feature is currently scheduled for v4.0.0, but this schedule is subject to change. See https://github.com/optuna/optuna/releases/tag/v2.4.0.

Methods

infer_relative_search_space(study, trial)	Infer the search space that will be used by relative sampling in the target trial.
reseed_rng()	Reseed sampler's random number generator.
sample_independent(study, trial, param_name, ...)	Sample a parameter for a given distribution.
sample_relative(study, trial, search_space)	Sample parameters in a given search space.

参数 **seed** (*Optional[int]*) –

返回类型 None

infer_relative_search_space (*study*, *trial*)

Infer the search space that will be used by relative sampling in the target trial.

This method is called right before *sample_relative()* method, and the search space returned by this method is passed to it. The parameters not contained in the search space will be sampled by using *sample_independent()* method.

参数

- **study** (optuna.multi_objective.study.MultiObjectiveStudy) –Target study object.
- **trial** (optuna.multi_objective.trial.FrozenMultiObjectiveTrial) –Target trial object.

返回 A dictionary containing the parameter names and parameter's distributions.

返回类型 *Dict*[str, optuna.distributions.BaseDistribution]

参见:

Please refer to *intersection_search_space()* as an implementation of *infer_relative_search_space()*.

reseed_rng()

Reseed sampler’s random number generator.

This method is called by the *MultiObjectiveStudy* instance if trials are executed in parallel with the option `n_jobs>1`. In that case, the sampler instance will be replicated including the state of the random number generator, and they may suggest the same values. To prevent this issue, this method assigns a different seed to each random number generator.

返回类型 None

sample_independent(*study*, *trial*, *param_name*, *param_distribution*)

Sample a parameter for a given distribution.

This method is called only for the parameters not contained in the search space returned by `sample_relative()` method. This method is suitable for sampling algorithms that do not use the relationship between parameters such as random sampling.

参数

- **study**(optuna.multi_objective.study.MultiObjectiveStudy) –Target study object.
- **trial**(optuna.multi_objective.trial.FrozenMultiObjectiveTrial) –Target trial object.
- **param_name**(*str*) –Name of the sampled parameter.
- **param_distribution**(*optuna.distributions.BaseDistribution*) – Distribution object that specifies a prior and/or scale of the sampling algorithm.

返回 A parameter value.

返回类型 *Any*

sample_relative(*study*, *trial*, *search_space*)

Sample parameters in a given search space.

This method is called once at the beginning of each trial, i.e., right before the evaluation of the objective function. This method is suitable for sampling algorithms that use the relationship between parameters.

参数

- **study**(optuna.multi_objective.study.MultiObjectiveStudy) –Target study object.
- **trial**(optuna.multi_objective.trial.FrozenMultiObjectiveTrial) –Target trial object.
- **search_space**(*Dict[str, optuna.distributions.BaseDistribution]*) –The search space returned by *infer_relative_search_space()*.

返回 A dictionary containing the parameter names and the values.

返回类型 *Dict*[str, *Any*]

optuna.multi_objective.samplers.MOTPEMultiObjectiveSampler

class optuna.multi_objective.samplers.**MOTPEMultiObjectiveSampler**(*consider_prior=True*, *prior_weight=1.0*, *consider_magic_clip=True*, *consider_endpoints=True*, *n_startup_trials=10*, *n_ehvi_candidates=24*, *gamma=<function default_gamma>*, *weights_above=<function _default_weights_above>*, *seed=None*)

Multi-objective sampler using the MOTPE algorithm.

This sampler is a multiobjective version of *TPESampler*.

For further information about MOTPE algorithm, please refer to the following paper:

- Multiobjective tree-structured parzen estimator for computationally expensive optimization problems

参数

- **consider_prior** (*bool*) –Enhance the stability of Parzen estimator by imposing a Gaussian prior when True. The prior is only effective if the sampling distribution is either *UniformDistribution*, *DiscreteUniformDistribution*, *LogUniformDistribution*, *IntUniformDistribution*, or *IntLogUniformDistribution*.
- **prior_weight** (*float*) –The weight of the prior. This argument is used in *UniformDistribution*, *DiscreteUniformDistribution*, *LogUniformDistribution*, *IntUniformDistribution*, *IntLogUniformDistribution*, and *CategoricalDistribution*.
- **consider_magic_clip** (*bool*) –Enable a heuristic to limit the smallest variances of Gaussians used in the Parzen estimator.
- **consider_endpoints** (*bool*) –Take endpoints of domains into account when calculating variances of Gaussians in Parzen estimator. See the original paper for details on the

heuristics to calculate the variances.

- **n_startup_trials** (*int*) –The random sampling is used instead of the MOTPE algorithm until the given number of trials finish in the same study. 11 * number of variables - 1 is recommended in the original paper.
- **n_ehvi_candidates** (*int*) –Number of candidate samples used to calculate the expected hypervolume improvement.
- **gamma** (*Callable[[int], int]*) –A function that takes the number of finished trials and returns the number of trials to form a density function for samples with low grains. See the original paper for more details.
- **weights_above** (*Callable[[int], numpy.ndarray]*) –A function that takes the number of finished trials and returns a weight for them. As default, weights are automatically calculated by the MOTPE's default strategy.
- **seed** (*Optional[int]*) –Seed for random number generator.

返回类型 None

备注： Initialization with Latin hypercube sampling may improve optimization performance. However, the current implementation only supports initialization with random sampling.

示例

```
import optuna

seed = 128
num_variables = 9
n_startup_trials = 11 * num_variables - 1


def objective(trial):
    x = []
    for i in range(1, num_variables + 1):
        x.append(trial.suggest_float(f"x{i}", 0.0, 2.0 * i))
    return x


sampler = optuna.multi_objective.samplers.MOTPEMultiObjectiveSampler(
    n_startup_trials=n_startup_trials, n_ehvi_candidates=24, seed=seed
)
study = optuna.multi_objective.create_study(
```

(下页继续)

(续上页)

```
    ["minimize"] * num_variables, sampler=sampler
)
study.optimize(objective, n_trials=250)
```

警告： Deprecated in v2.4.0. This feature will be removed in the future. The removal of this feature is currently scheduled for v4.0.0, but this schedule is subject to change. See https://github.com/optuna/optuna/releases/tag/v2.4.0.

Methods

after_trial(study, trial, state, values)	Trial post-processing.
hyperopt_parameters()	Return the the default parameters of hyperopt (v0.1.2).
infer_relative_search_space(study, trial)	Infer the search space that will be used by relative sampling in the target trial.
reseed_rng()	Reseed sampler's random number generator.
sample_independent(study, trial, param_name, ...)	Sample a parameter for a given distribution.
sample_relative(study, trial, search_space)	Sample parameters in a given search space.

after_trial (*study*, *trial*, *state*, *values*)

Trial post-processing.

This method is called after the objective function returns and right before the trials is finished and its state is stored.

备注： Added in v2.4.0 as an experimental feature. The interface may change in newer versions without prior notice. See https://github.com/optuna/optuna/releases/tag/v2.4.0.

参数

- **study** (optuna.study.Study) –Target study object.
- **trial** (optuna.trial._frozen.FrozenTrial) –Target trial object. Take a copy before modifying this object.
- **state** (optuna.trial._state.TrialState) –Resulting trial state.
- **values** (*Optional[Sequence[float]]*) –Resulting trial values. Guaranteed to not be None if trial succeeded.

返回类型 None

static hyperopt_parameters()

Return the the default parameters of hyperopt (v0.1.2).

TPESampler can be instantiated with the parameters returned by this method.

示例

Create a *TPESampler* instance with the default parameters of hyperopt.

```
import optuna
from optuna.samplers import TPESampler


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


sampler = TPESampler(**TPESampler.hyperopt_parameters())
study = optuna.create_study(sampler=sampler)
study.optimize(objective, n_trials=10)
```

返回 A dictionary containing the default parameters of hyperopt.

返回类型 *Dict*[str, *Any*]

infer_relative_search_space(*study*, *trial*)

Infer the search space that will be used by relative sampling in the target trial.

This method is called right before *sample_relative()* method, and the search space returned by this method is passed to it. The parameters not contained in the search space will be sampled by using *sample_independent()* method.

参数

- **study** (*Union*[optuna.study.Study, optuna.multi_objective.study.MultiObjectiveStudy]) –Target study object.
- **trial** (*Union*[optuna.trial._frozen.FrozenTrial, optuna.multi_objective.trial.FrozenMultiObjectiveTrial]) –Target trial object. Take a copy before modifying this object.

返回 A dictionary containing the parameter names and parameter’s distributions.

返回类型 *Dict*[str, optuna.distributions.BaseDistribution]

参见:

Please refer to *intersection_search_space()* as an implementation of *infer_relative_search_space()*.

reseed_rng()

Reseed sampler’s random number generator.

This method is called by the *Study* instance if trials are executed in parallel with the option `n_jobs>1`. In that case, the sampler instance will be replicated including the state of the random number generator, and they may suggest the same values. To prevent this issue, this method assigns a different seed to each random number generator.

返回类型 None

sample_independent(*study*, *trial*, *param_name*, *param_distribution*)

Sample a parameter for a given distribution.

This method is called only for the parameters not contained in the search space returned by *sample_relative()* method. This method is suitable for sampling algorithms that do not use relationship between parameters such as random sampling and TPE.

备注: The failed trials are ignored by any build-in samplers when they sample new parameters. Thus, failed trials are regarded as deleted in the samplers’ perspective.

参数

- **study** (*Union[optuna.study.Study, optuna.multi_objective.study.MultiObjectiveStudy]*) –Target study object.
- **trial** (*Union[optuna.trial._frozen.FrozenTrial, optuna.multi_objective.trial.FrozenMultiObjectiveTrial]*) –Target trial object. Take a copy before modifying this object.
- **param_name** (*str*) –Name of the sampled parameter.
- **param_distribution** (*optuna.distributions.BaseDistribution*) –Distribution object that specifies a prior and/or scale of the sampling algorithm.

返回 A parameter value.

返回类型 *Any*

sample_relative(*study*, *trial*, *search_space*)

Sample parameters in a given search space.

This method is called once at the beginning of each trial, i.e., right before the evaluation of the objective function. This method is suitable for sampling algorithms that use relationship between parameters such as

Gaussian Process and CMA-ES.

备注： The failed trials are ignored by any build-in samplers when they sample new parameters. Thus, failed trials are regarded as deleted in the samplers' perspective.

参数

- **study** (*Union[optuna.study.Study, optuna.multi_objective.study.MultiObjectiveStudy]*) –Target study object.
- **trial** (*Union[optuna.trial._frozen.FrozenTrial, optuna.multi_objective.trial.FrozenMultiObjectiveTrial]*) –Target trial object. Take a copy before modifying this object.
- **search_space** (*Dict[str, optuna.distributions.BaseDistribution]*) –The search space returned by *infer_relative_search_space()*.

返回 A dictionary containing the parameter names and the values.

返回类型 *Dict*[str, *Any*]

optuna.multi_objective.study

optuna.multi_objective.study.MultiObjectiveStudy	A study corresponds to a multi-objective optimization task, i.e., a set of trials.
optuna.multi_objective.study.create_study	Create a new *MultiObjectiveStudy*.
optuna.multi_objective.study.load_study	Load the existing *MultiObjectiveStudy* that has the specified name.

optuna.multi_objective.study.MultiObjectiveStudy

class optuna.multi_objective.study.**MultiObjectiveStudy**(*study*)

A study corresponds to a multi-objective optimization task, i.e., a set of trials.

This object provides interfaces to run a new `Trial`, access trials' history, set/get user-defined attributes of the study itself.

Note that the direct use of this constructor is not recommended. To create and load a study, please refer to the documentation of *create_study()* and *load_study()* respectively.

警告: Deprecated in v2.4.0. This feature will be removed in the future. The removal of this feature is currently scheduled for v4.0.0, but this schedule is subject to change. See https://github.com/optuna/optuna/releases/tag/v2.4.0.

Methods

enqueue_trial(params)	Enqueue a trial with given parameter values.
get_pareto_front_trials()	Return trials located at the pareto front in the study.
get_trials([deepcopy, states])	Return all trials in the study.
optimize(objective[, timeout, n_trials, ...])	Optimize an objective function.
set_system_attr(key, value)	Set a system attribute to the study.
set_user_attr(key, value)	Set a user attribute to the study.

Attributes

directions	Return the optimization direction list.
n_objectives	Return the number of objectives.
sampler	Return the sampler.
system_attrs	Return system attributes.
trials	Return all trials in the study.
user_attrs	Return user attributes.

参数 **study** (optuna.study.Study) –

property directions: List[*optuna._study_direction.StudyDirection*]

Return the optimization direction list.

返回 A list that contains the optimization direction for each objective value.

enqueue_trial(*params*)

Enqueue a trial with given parameter values.

You can fix the next sampling parameters which will be evaluated in your objective function.

Please refer to the documentation of *optuna.study.Study.enqueue_trial()* for further details.

参数 **params** (*Dict[str, Any]*) –Parameter values to pass your objective function.

返回类型 None

get_pareto_front_trials()

Return trials located at the pareto front in the study.

A trial is located at the pareto front if there are no trials that dominate the trial. It's called that a trial `t0` dominates another trial `t1` if `all(v0 <= v1) for v0, v1 in zip(t0.values, t1.values)` and `any(v0 < v1) for v0, v1 in zip(t0.values, t1.values)` are held.

返回 A list of *FrozenMultiObjectiveTrial* objects.

返回类型 *List*[*optuna.multi_objective.trial.FrozenMultiObjectiveTrial*]

get_trials(*deepcopy=True*, *states=None*)

Return all trials in the study.

The returned trials are ordered by trial number.

参数

- **deepcopy** (*bool*) –Flag to control whether to apply `copy.deepcopy()` to the trials. Note that if you set the flag to `False`, you shouldn't mutate any fields of the returned trial. Otherwise the internal state of the study may corrupt and unexpected behavior may happen.
- **states** (*Optional[Tuple[optuna.trial._state.TrialState, ...]]*) –Trial states to filter on. If `None`, include all states.

返回 A list of *FrozenMultiObjectiveTrial* objects.

返回类型 *List*[*optuna.multi_objective.trial.FrozenMultiObjectiveTrial*]

property n_objectives: int

Return the number of objectives.

返回 Number of objectives.

optimize(*objective*, *timeout=None*, *n_trials=None*, *n_jobs=1*, *catch=()*, *callbacks=None*, *gc_after_trial=True*, *show_progress_bar=False*)

Optimize an objective function.

This method is the same as *optuna.study.Study.optimize()* except for taking an objective function that returns multi-objective values as the argument.

Please refer to the documentation of *optuna.study.Study.optimize()* for further details.

示例

```
import optuna


def objective(trial):
    # Binh and Korn function.
    x = trial.suggest_float("x", 0, 5)
    y = trial.suggest_float("y", 0, 3)

    v0 = 4 * x ** 2 + 4 * y ** 2
    v1 = (x - 5) ** 2 + (y - 5) ** 2
    return v0, v1


study = optuna.multi_objective.create_study(["minimize", "minimize"])
study.optimize(objective, n_trials=3)
```

参数

- **objective** (*Callable[[optuna.multi_objective.trial.MultiObjectiveTrial], Sequence[float]]*) –
- **timeout** (*Optional[int]*) –
- **n_trials** (*Optional[int]*) –
- **n_jobs** (*int*) –
- **catch** (*Tuple[Type[Exception], ...]*) –
- **callbacks** (*Optional[List[Callable[[optuna.multi_objective.study.MultiObjectiveStudy, optuna.multi_objective.trial.FrozenMultiObjectiveTrial], None]]]*) –
- **gc_after_trial** (*bool*) –
- **show_progress_bar** (*bool*) –

返回类型 None

property sampler:
optuna.multi_objective.samplers._base.BaseMultiObjectiveSampler

Return the sampler.

返回 A `BaseMultiObjectiveSampler` object.

set_system_attr (*key*, *value*)

Set a system attribute to the study.

Note that Optuna internally uses this method to save system messages. Please use *set_user_attr()* to set users' attributes.

参数

- **key** (*str*) –A key string of the attribute.
- **value** (*Any*) –A value of the attribute. The value should be JSON serializable.

返回类型 None

set_user_attr(*key*, *value*)

Set a user attribute to the study.

参数

- **key** (*str*) –A key string of the attribute.
- **value** (*Any*) –A value of the attribute. The value should be JSON serializable.

返回类型 None

property system_attrs: Dict[str, Any]

Return system attributes.

返回 A dictionary containing all system attributes.

property trials:
List[*optuna.multi_objective.trial.FrozenMultiObjectiveTrial*]

Return all trials in the study.

The returned trials are ordered by trial number.

This is a short form of `self.get_trials(deepcopy=True, states=None)`.

返回 A list of *FrozenMultiObjectiveTrial* objects.

property user_attrs: Dict[str, Any]

Return user attributes.

返回 A dictionary containing all user attributes.

optuna.multi_objective.study.create_study

optuna.multi_objective.study.**create_study**(*directions*, *study_name=None*, *storage=None*, *sampler=None*, *load_if_exists=False*)

Create a new *MultiObjectiveStudy*.

示例

```
import optuna


def objective(trial):
    # Binh and Korn function.
    x = trial.suggest_float("x", 0, 5)
    y = trial.suggest_float("y", 0, 3)

    v0 = 4 * x ** 2 + 4 * y ** 2
    v1 = (x - 5) ** 2 + (y - 5) ** 2
    return v0, v1


study = optuna.multi_objective.create_study(["minimize", "minimize"])
study.optimize(objective, n_trials=3)
```

参数

- **directions** (*List[str]*) –Optimization direction for each objective value. Set `minimize` for minimization and `maximize` for maximization.
- **study_name** (*Optional[str]*) –Study's name. If this argument is set to None, a unique name is generated automatically.
- **storage** (*Optional[Union[str, optuna.storages._base.BaseStorage]]*) –Database URL. If this argument is set to None, in-memory storage is used, and the *Study* will not be persistent.

  备注:

  When a database URL is passed, Optuna internally uses SQLAlchemy to handle the database. Please refer to SQLAlchemy's document for further details. If you want to specify non-default options to SQLAlchemy Engine, you can instantiate *RDBStorage* with your desired options and pass it to the `storage` argument instead of a URL.

- **sampler** (*Optional[optuna.multi_objective.samplers._base.BaseMultiObjectiveSampler]*) –A sampler object that implements background algorithm for value suggestion. If `None` is specified, *NSGAIIMultiObjectiveSampler* is used as the default. See also *samplers*.
- **load_if_exists** (*bool*) –Flag to control the behavior to handle a conflict of study names. In the case where a study named `study_name` already exists in the `storage`, a

DuplicatedStudyError is raised if `load_if_exists` is set to `False`. Otherwise, the creation of the study is skipped, and the existing one is returned.

返回 A *MultiObjectiveStudy* object.

返回类型 *optuna.multi_objective.study.MultiObjectiveStudy*

> **警告:** Deprecated in v2.4.0. This feature will be removed in the future. The removal of this feature is currently scheduled for v4.0.0, but this schedule is subject to change. See https://github.com/optuna/optuna/releases/tag/v2.4.0.

optuna.multi_objective.study.load_study

optuna.multi_objective.study.**load_study**(*study_name*, *storage*, *sampler=None*)

Load the existing *MultiObjectiveStudy* that has the specified name.

示例

```
import optuna


def objective(trial):
    # Binh and Korn function.
    x = trial.suggest_float("x", 0, 5)
    y = trial.suggest_float("y", 0, 3)

    v0 = 4 * x ** 2 + 4 * y ** 2
    v1 = (x - 5) ** 2 + (y - 5) ** 2
    return v0, v1


study = optuna.multi_objective.create_study(
    directions=["minimize", "minimize"],
    study_name="my_study",
    storage="sqlite:///example.db",
)
study.optimize(objective, n_trials=3)

loaded_study = optuna.multi_objective.study.load_study(
    study_name="my_study", storage="sqlite:///example.db"
)
assert len(loaded_study.trials) == len(study.trials)
```

参数

- **study_name** (*str*) –Study’ s name. Each study has a unique name as an identifier.
- **storage** (*Union[str, optuna.storages._base.BaseStorage]*) –Database URL such as `sqlite:///example.db`. Please see also the documentation of *create_study()* for further details.
- **sampler** (*Optional[optuna.multi_objective.samplers._base.BaseMultiObjectiveSampler]*) –A sampler object that implements background algorithm for value suggestion. If `None` is specified, *RandomMultiObjectiveSampler* is used as the default. See also *samplers*.

返回 A *MultiObjectiveStudy* object.

返回类型 *optuna.multi_objective.study.MultiObjectiveStudy*

> **警告:** Deprecated in v2.4.0. This feature will be removed in the future. The removal of this feature is currently scheduled for v4.0.0, but this schedule is subject to change. See https://github.com/optuna/optuna/releases/tag/v2.4.0.

optuna.multi_objective.trial

optuna.multi_objective.trial.MultiObjectiveTrial	A trial is a process of evaluating an objective function.
optuna.multi_objective.trial.FrozenMultiObjectiveTrial	Status and results of a *MultiObjectiveTrial*.

optuna.multi_objective.trial.MultiObjectiveTrial

class optuna.multi_objective.trial.**MultiObjectiveTrial**(*trial*)

A trial is a process of evaluating an objective function.

This object is passed to an objective function and provides interfaces to get parameter suggestion, manage the trial’ s state, and set/get user-defined attributes of the trial.

Note that the direct use of this constructor is not recommended. This object is seamlessly instantiated and passed to the objective function behind the *optuna.multi_objective.study.MultiObjectiveStudy.optimize()* method; hence library users do not care about instantiation of this object.

参数 **trial** (optuna.trial._trial.Trial) –A *Trial* object.

警告: Deprecated in v2.4.0. This feature will be removed in the future. The removal of this feature is currently scheduled for v4.0.0, but this schedule is subject to change. See https://github.com/optuna/optuna/releases/tag/v2.4.0.

Methods

report(values, step)	Report intermediate objective function values for a given step.
set_system_attr(key, value)	Set system attributes to the trial.
set_user_attr(key, value)	Set user attributes to the trial.
suggest_categorical(name, choices)	Suggest a value for the categorical parameter.
suggest_discrete_uniform(name, low, high, q)	Suggest a value for the discrete parameter.
suggest_float(name, low, high, *[, step, log])	Suggest a value for the floating point parameter.
suggest_int(name, low, high[, step, log])	Suggest a value for the integer parameter.
suggest_loguniform(name, low, high)	Suggest a value for the continuous parameter.
suggest_uniform(name, low, high)	Suggest a value for the continuous parameter.

Attributes

datetime_start	Return start datetime.
distributions	Return distributions of parameters to be optimized.
number	Return trial's number which is consecutive and unique in a study.
params	Return parameters to be optimized.
system_attrs	Return system attributes.
user_attrs	Return user attributes.

property datetime_start: Optional[datetime.datetime]

Return start datetime.

返回 Datetime where the *Trial* started.

property distributions: Dict[str, optuna.distributions.BaseDistribution]

Return distributions of parameters to be optimized.

返回 A dictionary containing all distributions.

property number: int

Return trial's number which is consecutive and unique in a study.

返回 A trial number.

property params: Dict[str, Any]

Return parameters to be optimized.

返回 A dictionary containing all parameters.

report(*values*, *step*)

Report intermediate objective function values for a given step.

The reported values are used by the pruners to determine whether this trial should be pruned.

参见:

Please refer to *BasePruner*.

备注: The reported values are converted to `float` type by applying `float()` function internally. Thus, it accepts all float-like types (e.g., `numpy.float32`). If the conversion fails, a `TypeError` is raised.

参数

- **values** (*Sequence[float]*) –Intermediate objective function values for a given step.
- **step** (*int*) –Step of the trial (e.g., Epoch of neural network training).

返回类型 None

set_system_attr(*key*, *value*)

Set system attributes to the trial.

Please refer to the documentation of *optuna.trial.Trial.set_system_attr()* for further details.

参数

- **key** (*str*) –
- **value** (*Any*) –

返回类型 None

set_user_attr(*key*, *value*)

Set user attributes to the trial.

Please refer to the documentation of *optuna.trial.Trial.set_user_attr()* for further details.

参数

- **key** (*str*) –
- **value** (*Any*) –

返回类型 None

suggest_categorical(*name*, *choices*)

Suggest a value for the categorical parameter.

Please refer to the documentation of *optuna.trial.Trial.suggest_categorical()* for further details.

参数

- **name** (*str*) –
- **choices** (*Sequence[Union[None, bool, int, float, str]]*) –

返回类型 *Union*[None, bool, int, float, str]

suggest_discrete_uniform(*name*, *low*, *high*, *q*)

Suggest a value for the discrete parameter.

Please refer to the documentation of *optuna.trial.Trial.suggest_discrete_uniform()* for further details.

参数

- **name** (*str*) –
- **low** (*float*) –
- **high** (*float*) –
- **q** (*float*) –

返回类型 float

suggest_float(*name*, *low*, *high*, ***, *step=None*, *log=False*)

Suggest a value for the floating point parameter.

Please refer to the documentation of *optuna.trial.Trial.suggest_float()* for further details.

参数

- **name** (*str*) –
- **low** (*float*) –
- **high** (*float*) –
- **step** (*Optional[float]*) –
- **log** (*bool*) –

返回类型 float

suggest_int(*name*, *low*, *high*, *step=1*, *log=False*)

Suggest a value for the integer parameter.

Please refer to the documentation of *optuna.trial.Trial.suggest_int()* for further details.

参数

- **name** (*str*) –
- **low** (*int*) –
- **high** (*int*) –
- **step** (*int*) –
- **log** (*bool*) –

返回类型 int

suggest_loguniform(*name*, *low*, *high*)

Suggest a value for the continuous parameter.

Please refer to the documentation of *optuna.trial.Trial.suggest_loguniform()* for further details.

参数

- **name** (*str*) –
- **low** (*float*) –
- **high** (*float*) –

返回类型 float

suggest_uniform(*name*, *low*, *high*)

Suggest a value for the continuous parameter.

Please refer to the documentation of *optuna.trial.Trial.suggest_uniform()* for further details.

参数

- **name** (*str*) –
- **low** (*float*) –
- **high** (*float*) –

返回类型 float

property system_attrs: Dict[str, Any]

Return system attributes.

返回 A dictionary containing all system attributes.

property user_attrs: Dict[str, Any]

Return user attributes.

返回 A dictionary containing all user attributes.

optuna.multi_objective.trial.FrozenMultiObjectiveTrial

class optuna.multi_objective.trial.**FrozenMultiObjectiveTrial**(*n_objectives*, *trial*)

Status and results of a *MultiObjectiveTrial*.

参数

- **n_objectives** (*int*) –
- **trial** (optuna.trial._frozen.FrozenTrial) –

number

Unique and consecutive number of *MultiObjectiveTrial* for each *MultiObjectiveStudy*. Note that this field uses zero-based numbering.

state

TrialState of the *MultiObjectiveTrial*.

values

Objective values of the *MultiObjectiveTrial*.

datetime_start

Datetime where the *MultiObjectiveTrial* started.

datetime_complete

Datetime where the *MultiObjectiveTrial* finished.

params

Dictionary that contains suggested parameters.

distributions

Dictionary that contains the distributions of *params*.

user_attrs

Dictionary that contains the attributes of the *MultiObjectiveTrial* set with *optuna.multi_objective.trial.MultiObjectiveTrial.set_user_attr()*.

intermediate_values

Intermediate objective values set with *optuna.multi_objective.trial.MultiObjectiveTrial.report()*.

警告： Deprecated in v2.4.0. This feature will be removed in the future. The removal of this feature is currently scheduled for v4.0.0, but this schedule is subject to change. See https://github.com/optuna/optuna/releases/tag/v2.4.0.

Attributes

datetime_complete
datetime_start
distributions
last_step
number
params
state
system_attrs
user_attrs

optuna.multi_objective.visualization

备注： *optuna.multi_objective.visualization* module uses plotly to create figures, but JupyterLab cannot render them by default. Please follow this installation guide to show figures in JupyterLab.

optuna.multi_objective.visualization.plot_pareto_front	Plot the pareto front of a study.

optuna.multi_objective.visualization.plot_pareto_front

optuna.multi_objective.visualization.**plot_pareto_front**(*study*, *names=None*, *include_dominated_trials=False*, *axis_order=None*)

Plot the pareto front of a study.

示例

The following code snippet shows how to plot the pareto front of a study.

```
import optuna


def objective(trial):
    x = trial.suggest_float("x", 0, 5)
    y = trial.suggest_float("y", 0, 3)

    v0 = 4 * x ** 2 + 4 * y ** 2
    v1 = (x - 5) ** 2 + (y - 5) ** 2
    return v0, v1


study = optuna.multi_objective.create_study(["minimize", "minimize"])
study.optimize(objective, n_trials=50)

fig = optuna.multi_objective.visualization.plot_pareto_front(study)
fig.show()
```

参数

- **study** (optuna.multi_objective.study.MultiObjectiveStudy) – A *MultiObjectiveStudy* object whose trials are plotted for their objective values.
- **names** (*Optional[List[str]]*) – Objective name list used as the axis titles. If None is specified, "Objective {objective_index}" is used instead.
- **include_dominated_trials** (*bool*) – A flag to include all dominated trial's objective values.
- **axis_order** (*Optional[List[int]]*) – A list of indices indicating the axis order. If None is specified, default order is used.

返回 A plotly.graph_objs.Figure object.

引发 **ValueError** – If the number of objectives of study isn't 2 or 3.

返回类型 plotly.graph_objs._figure.Figure

> **警告:** Deprecated in v2.4.0. This feature will be removed in the future. The removal of this feature is currently scheduled for v4.0.0, but this schedule is subject to change. See https://github.com/optuna/optuna/releases/tag/v2.4.0.

6.3.9 optuna.pruners

pruners 模块定义了一个*BasePruner* 类，其特点是有一个抽象的*prune()* 方法。该方法对一个给定的 trial 及其相关的 study，返回一个代表该 trial 是否应该被剪枝的布尔值。该值由存储的目标函数的中间值确定，这些中间值由前述*optuna.trial.Trial.report()* 传入。本模块中其余的类代表子类，继承自*BasePruner*，它们实现了不同的剪枝策略。

optuna.pruners.BasePruner	Pruner 基类
optuna.pruners.MedianPruner	使用中值停止规则的 pruner.
optuna.pruners.NopPruner	不剪枝 trial 的 pruner.
optuna.pruners.PercentilePruner	保留到指定百分位 trial 的 pruner.
optuna.pruners.SuccessiveHalvingPruner	使用异步连续减半算法的 pruner.
optuna.pruners.HyperbandPruner	使用 hyperband 的 pruner.
optuna.pruners.ThresholdPruner	用于检测 trial 的无关度量的 pruner.

optuna.pruners.BasePruner

class optuna.pruners.**BasePruner**

Base class for pruners.

Methods

prune(study, trial)	Judge whether the trial should be pruned based on the reported values.

abstract prune(*study*, *trial*)

Judge whether the trial should be pruned based on the reported values.

Note that this method is not supposed to be called by library users. Instead, *optuna.trial.Trial.report()* and *optuna.trial.Trial.should_prune()* provide user interfaces to implement pruning mechanism in an objective function.

参数

- **study** (optuna.study.Study) –Study object of the target study.
- **trial** (optuna.trial._frozen.FrozenTrial) –FrozenTrial object of the target trial. Take a copy before modifying this object.

返回 A boolean value representing whether the trial should be pruned.

返回类型 bool

optuna.pruners.MedianPruner

class optuna.pruners.**MedianPruner**(*n_startup_trials=5*, *n_warmup_steps=0*, *interval_steps=1*)

Pruner using the median stopping rule.

Prune if the trial’s best intermediate result is worse than median of intermediate results of previous trials at the same step.

示例

We minimize an objective function with the median stopping rule.

```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import train_test_split

import optuna

X, y = load_iris(return_X_y=True)
X_train, X_valid, y_train, y_valid = train_test_split(X, y)
classes = np.unique(y)


def objective(trial):
    alpha = trial.suggest_float("alpha", 0.0, 1.0)
    clf = SGDClassifier(alpha=alpha)
    n_train_iter = 100

    for step in range(n_train_iter):
        clf.partial_fit(X_train, y_train, classes=classes)

        intermediate_value = clf.score(X_valid, y_valid)
        trial.report(intermediate_value, step)

        if trial.should_prune():
```

(下页继续)

(续上页)

```
            raise optuna.TrialPruned()

    return clf.score(X_valid, y_valid)


study = optuna.create_study(
    direction="maximize",
    pruner=optuna.pruners.MedianPruner(
        n_startup_trials=5, n_warmup_steps=30, interval_steps=10
    ),
)
study.optimize(objective, n_trials=20)
```

参数

- **n_startup_trials** (*int*) –Pruning is disabled until the given number of trials finish in the same study.
- **n_warmup_steps** (*int*) –Pruning is disabled until the trial exceeds the given number of step. Note that this feature assumes that `step` starts at zero.
- **interval_steps** (*int*) –Interval in number of steps between the pruning checks, offset by the warmup steps. If no value has been reported at the time of a pruning check, that particular check will be postponed until a value is reported.

返回类型 None

Methods

prune(study, trial)	Judge whether the trial should be pruned based on the reported values.

prune (*study*, *trial*)

Judge whether the trial should be pruned based on the reported values.

Note that this method is not supposed to be called by library users. Instead, *optuna.trial.Trial.report()* and *optuna.trial.Trial.should_prune()* provide user interfaces to implement pruning mechanism in an objective function.

参数

- **study** (optuna.study.Study) –Study object of the target study.
- **trial** (optuna.trial._frozen.FrozenTrial) –FrozenTrial object of the target trial. Take a copy before modifying this object.

返回 A boolean value representing whether the trial should be pruned.

返回类型 bool

optuna.pruners.NopPruner

class optuna.pruners.**NopPruner**

Pruner which never prunes trials.

示例

```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import train_test_split

import optuna

X, y = load_iris(return_X_y=True)
X_train, X_valid, y_train, y_valid = train_test_split(X, y)
classes = np.unique(y)


def objective(trial):
    alpha = trial.suggest_float("alpha", 0.0, 1.0)
    clf = SGDClassifier(alpha=alpha)
    n_train_iter = 100

    for step in range(n_train_iter):
        clf.partial_fit(X_train, y_train, classes=classes)

        intermediate_value = clf.score(X_valid, y_valid)
        trial.report(intermediate_value, step)

        if trial.should_prune():
            assert False, "should_prune() should always return False with this␣
↪pruner."
            raise optuna.TrialPruned()

    return clf.score(X_valid, y_valid)


study = optuna.create_study(direction="maximize", pruner=optuna.pruners.
↪NopPruner())
```

(下页继续)

(续上页)

```
study.optimize(objective, n_trials=20)
```

Methods

prune(study, trial)	Judge whether the trial should be pruned based on the reported values.

prune (*study*, *trial*)

Judge whether the trial should be pruned based on the reported values.

Note that this method is not supposed to be called by library users. Instead, *optuna.trial.Trial.report()* and *optuna.trial.Trial.should_prune()* provide user interfaces to implement pruning mechanism in an objective function.

参数

- **study** (optuna.study.Study) –Study object of the target study.
- **trial** (optuna.trial._frozen.FrozenTrial) –FrozenTrial object of the target trial. Take a copy before modifying this object.

返回 A boolean value representing whether the trial should be pruned.

返回类型 bool

optuna.pruners.PercentilePruner

class optuna.pruners.**PercentilePruner** (*percentile*, *n_startup_trials=5*, *n_warmup_steps=0*, *interval_steps=1*)

Pruner to keep the specified percentile of the trials.

Prune if the best intermediate value is in the bottom percentile among trials at the same step.

示例

```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import train_test_split

import optuna
```

(下页继续)

(续上页)

```
X, y = load_iris(return_X_y=True)
X_train, X_valid, y_train, y_valid = train_test_split(X, y)
classes = np.unique(y)


def objective(trial):
    alpha = trial.suggest_float("alpha", 0.0, 1.0)
    clf = SGDClassifier(alpha=alpha)
    n_train_iter = 100

    for step in range(n_train_iter):
        clf.partial_fit(X_train, y_train, classes=classes)

        intermediate_value = clf.score(X_valid, y_valid)
        trial.report(intermediate_value, step)

        if trial.should_prune():
            raise optuna.TrialPruned()

    return clf.score(X_valid, y_valid)


study = optuna.create_study(
    direction="maximize",
    pruner=optuna.pruners.PercentilePruner(
        25.0, n_startup_trials=5, n_warmup_steps=30, interval_steps=10
    ),
)
study.optimize(objective, n_trials=20)
```

参数

- **percentile** (*float*) –Percentile which must be between 0 and 100 inclusive (e.g., When given 25.0, top of 25th percentile trials are kept).
- **n_startup_trials** (*int*) –Pruning is disabled until the given number of trials finish in the same study.
- **n_warmup_steps** (*int*) –Pruning is disabled until the trial exceeds the given number of step. Note that this feature assumes that `step` starts at zero.
- **interval_steps** (*int*) –Interval in number of steps between the pruning checks, offset by the warmup steps. If no value has been reported at the time of a pruning check, that particular check will be postponed until a value is reported. Value must be at least 1.

返回类型 None

Methods

prune(study, trial)	Judge whether the trial should be pruned based on the reported values.

prune (*study*, *trial*)

Judge whether the trial should be pruned based on the reported values.

Note that this method is not supposed to be called by library users. Instead, *optuna.trial.Trial.report()* and *optuna.trial.Trial.should_prune()* provide user interfaces to implement pruning mechanism in an objective function.

参数

- **study** (optuna.study.Study) –Study object of the target study.
- **trial** (optuna.trial._frozen.FrozenTrial) –FrozenTrial object of the target trial. Take a copy before modifying this object.

返回 A boolean value representing whether the trial should be pruned.

返回类型 bool

optuna.pruners.SuccessiveHalvingPruner

class optuna.pruners.**SuccessiveHalvingPruner**(*min_resource='auto'*, *reduction_factor=4*, *min_early_stopping_rate=0*, *bootstrap_count=0*)

Pruner using Asynchronous Successive Halving Algorithm.

Successive Halving is a bandit-based algorithm to identify the best one among multiple configurations. This class implements an asynchronous version of Successive Halving. Please refer to the paper of Asynchronous Successive Halving for detailed descriptions.

Note that, this class does not take care of the parameter for the maximum resource, referred to as R in the paper. The maximum resource allocated to a trial is typically limited inside the objective function (e.g., `step` number in simple_pruning.py, `EPOCH` number in chainer_integration.py).

参见:

Please refer to *report()*.

示例

We minimize an objective function with `SuccessiveHalvingPruner`.

```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import train_test_split

import optuna

X, y = load_iris(return_X_y=True)
X_train, X_valid, y_train, y_valid = train_test_split(X, y)
classes = np.unique(y)


def objective(trial):
    alpha = trial.suggest_float("alpha", 0.0, 1.0)
    clf = SGDClassifier(alpha=alpha)
    n_train_iter = 100

    for step in range(n_train_iter):
        clf.partial_fit(X_train, y_train, classes=classes)

        intermediate_value = clf.score(X_valid, y_valid)
        trial.report(intermediate_value, step)

        if trial.should_prune():
            raise optuna.TrialPruned()

    return clf.score(X_valid, y_valid)


study = optuna.create_study(
    direction="maximize", pruner=optuna.pruners.SuccessiveHalvingPruner()
)
study.optimize(objective, n_trials=20)
```

参数

- **min_resource** (*Union[str, int]*) –A parameter for specifying the minimum resource allocated to a trial (in the paper this parameter is referred to as r). This parameter defaults to 'auto' where the value is determined based on a heuristic that looks at the number of required steps for the first trial to complete.

  A trial is never pruned until it executes $\text{min\_resource} \times \text{reduction\_factor}^{\text{min\_early\_stopping\_rate}}$

steps (i.e., the completion point of the first rung). When the trial completes the first rung, it will be promoted to the next rung only if the value of the trial is placed in the top $\frac{1}{\text{reduction\_factor}}$ fraction of the all trials that already have reached the point (otherwise it will be pruned there). If the trial won the competition, it runs until the next completion point (i.e., $\text{min\_resource} \times \text{reduction\_factor}^{(\text{min\_early\_stopping\_rate}+\text{rung})}$ steps) and repeats the same procedure.

备注： If the step of the last intermediate value may change with each trial, please manually specify the minimum possible step to `min_resource`.

- **reduction_factor** (*int*) –A parameter for specifying reduction factor of promotable trials (in the paper this parameter is referred to as η). At the completion point of each rung, about $\frac{1}{\text{reduction\_factor}}$ trials will be promoted.
- **min_early_stopping_rate** (*int*) –A parameter for specifying the minimum early-stopping rate (in the paper this parameter is referred to as s).
- **bootstrap_count** (*int*) –Minimum number of trials that need to complete a rung before any trial is considered for promotion into the next rung.

返回类型 None

Methods

prune(study, trial)	Judge whether the trial should be pruned based on the reported values.

prune (*study*, *trial*)

Judge whether the trial should be pruned based on the reported values.

Note that this method is not supposed to be called by library users. Instead, *optuna.trial.Trial.report()* and *optuna.trial.Trial.should_prune()* provide user interfaces to implement pruning mechanism in an objective function.

参数

- **study** (optuna.study.Study) –Study object of the target study.
- **trial** (optuna.trial._frozen.FrozenTrial) –FrozenTrial object of the target trial. Take a copy before modifying this object.

返回 A boolean value representing whether the trial should be pruned.

返回类型 bool

optuna.pruners.HyperbandPruner

class optuna.pruners.**HyperbandPruner**(*min_resource=1*, *max_resource='auto'*, *reduction_factor=3*, *bootstrap_count=0*)

Pruner using Hyperband.

As SuccessiveHalving (SHA) requires the number of configurations n as its hyperparameter. For a given finite budget B, all the configurations have the resources of $\frac{B}{n}$ on average. As you can see, there will be a trade-off of B and $\frac{B}{n}$. Hyperband attacks this trade-off by trying different n values for a fixed budget.

备注:

- In the Hyperband paper, the counterpart of *RandomSampler* is used.
- Optuna uses *TPESampler* by default.
- The benchmark result shows that *optuna.pruners.HyperbandPruner* supports both samplers.

备注: If you use HyperbandPruner with *TPESampler*, it's recommended to consider to set larger n_trials or timeout to make full use of the characteristics of *TPESampler* because *TPESampler* uses some (by default, 10) *Trial*s for its startup.

As Hyperband runs multiple *SuccessiveHalvingPruner* and collect trials based on the current *Trial*'s bracket ID, each bracket needs to observe more than 10 *Trial*s for *TPESampler* to adapt its search space.

Thus, for example, if HyperbandPruner has 4 pruners in it, at least 4×10 trials are consumed for startup.

备注: Hyperband has several *SuccessiveHalvingPruner*. Each *SuccessiveHalvingPruner* is referred as "bracket" in the original paper. The number of brackets is an important factor to control the early stopping behavior of Hyperband and is automatically determined by min_resource, max_resource and reduction_factor as *The number of brackets = floor(log_{reduction_factor}(max_resource / min_resource)) + 1*. Please set reduction_factor so that the number of brackets is not too large (about 4 ~ 6 in most use cases). Please see Section 3.6 of the original paper for the detail.

参见:

Please refer to *report()*.

示例

We minimize an objective function with Hyperband pruning algorithm.

```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import train_test_split

import optuna

X, y = load_iris(return_X_y=True)
X_train, X_valid, y_train, y_valid = train_test_split(X, y)
classes = np.unique(y)
n_train_iter = 100


def objective(trial):
    alpha = trial.suggest_float("alpha", 0.0, 1.0)
    clf = SGDClassifier(alpha=alpha)

    for step in range(n_train_iter):
        clf.partial_fit(X_train, y_train, classes=classes)

        intermediate_value = clf.score(X_valid, y_valid)
        trial.report(intermediate_value, step)

        if trial.should_prune():
            raise optuna.TrialPruned()

    return clf.score(X_valid, y_valid)


study = optuna.create_study(
    direction="maximize",
    pruner=optuna.pruners.HyperbandPruner(
        min_resource=1, max_resource=n_train_iter, reduction_factor=3
    ),
)
study.optimize(objective, n_trials=20)
```

参数

- **min_resource** (*int*) –A parameter for specifying the minimum resource allocated to a trial noted as r in the paper. A smaller r will give a result faster, but a larger r will

give a better guarantee of successful judging between configurations. See the details for *SuccessiveHalvingPruner*.

- **max_resource** (*Union[str, int]*) –A parameter for specifying the maximum resource allocated to a trial. R in the paper corresponds to `max_resource / min_resource`. This value represents and should match the maximum iteration steps (e.g., the number of epochs for neural networks). When this argument is "auto", the maximum resource is estimated according to the completed trials. The default value of this argument is "auto".

  备注： With "auto", the maximum resource will be the largest step reported by *report()* in the first, or one of the first if trained in parallel, completed trial. No trials will be pruned until the maximum resource is determined.

  备注： If the step of the last intermediate value may change with each trial, please manually specify the maximum possible step to `max_resource`.

- **reduction_factor** (*int*) –A parameter for specifying reduction factor of promotable trials noted as η in the paper. See the details for *SuccessiveHalvingPruner*.
- **bootstrap_count** (*int*) –Parameter specifying the number of trials required in a rung before any trial can be promoted. Incompatible with `max_resouce` is `"auto"`. See the details for *SuccessiveHalvingPruner*.

返回类型 None

Methods

prune(study, trial)	Judge whether the trial should be pruned based on the reported values.

prune (*study*, *trial*)

Judge whether the trial should be pruned based on the reported values.

Note that this method is not supposed to be called by library users. Instead, *optuna.trial.Trial.report()* and *optuna.trial.Trial.should_prune()* provide user interfaces to implement pruning mechanism in an objective function.

参数

- **study** (optuna.study.Study) –Study object of the target study.

- **trial** (optuna.trial._frozen.FrozenTrial) –FrozenTrial object of the target trial. Take a copy before modifying this object.

返回 A boolean value representing whether the trial should be pruned.

返回类型 bool

optuna.pruners.ThresholdPruner

class optuna.pruners.**ThresholdPruner**(*lower=None*, *upper=None*, *n_warmup_steps=0*, *interval_steps=1*)

Pruner to detect outlying metrics of the trials.

Prune if a metric exceeds upper threshold, falls behind lower threshold or reaches `nan`.

示例

```
from optuna import create_study
from optuna.pruners import ThresholdPruner
from optuna import TrialPruned


def objective_for_upper(trial):
    for step, y in enumerate(ys_for_upper):
        trial.report(y, step)

        if trial.should_prune():
            raise TrialPruned()
    return ys_for_upper[-1]


def objective_for_lower(trial):
    for step, y in enumerate(ys_for_lower):
        trial.report(y, step)

        if trial.should_prune():
            raise TrialPruned()
    return ys_for_lower[-1]


ys_for_upper = [0.0, 0.1, 0.2, 0.5, 1.2]
ys_for_lower = [100.0, 90.0, 0.1, 0.0, -1]

study = create_study(pruner=ThresholdPruner(upper=1.0))
```

(下页继续)

(续上页)

```
study.optimize(objective_for_upper, n_trials=10)

study = create_study(pruner=ThresholdPruner(lower=0.0))
study.optimize(objective_for_lower, n_trials=10)
```

Args

lower: A minimum value which determines whether pruner prunes or not. If an intermediate value is smaller than lower, it prunes.

upper: A maximum value which determines whether pruner prunes or not. If an intermediate value is larger than upper, it prunes.

n_warmup_steps: Pruning is disabled if the step is less than the given number of warmup steps.

interval_steps: Interval in number of steps between the pruning checks, offset by the warmup steps. If no value has been reported at the time of a pruning check, that particular check will be postponed until a value is reported. Value must be at least 1.

Methods

prune(study, trial)	Judge whether the trial should be pruned based on the reported values.

参数

- **lower** (*Optional[float]*) –
- **upper** (*Optional[float]*) –
- **n_warmup_steps** (*int*) –
- **interval_steps** (*int*) –

返回类型 None

prune (*study*, *trial*)

Judge whether the trial should be pruned based on the reported values.

Note that this method is not supposed to be called by library users. Instead, *optuna.trial.Trial.report()* and *optuna.trial.Trial.should_prune()* provide user interfaces to implement pruning mechanism in an objective function.

参数

- **study** (optuna.study.Study) –Study object of the target study.

- **trial** (optuna.trial._frozen.FrozenTrial) –FrozenTrial object of the target trial. Take a copy before modifying this object.

返回 A boolean value representing whether the trial should be pruned.

返回类型 bool

6.3.10 optuna.samplers

samplers 模块定义了一个参数采样的基类，如*BaseSampler* 中广泛描述的那样。本模块中其余的类代表从*BaseSampler* 派生出来的子类，它们实现了不同的采样策略。

optuna.samplers.BaseSampler	Samplers 基类
optuna.samplers.GridSampler	使用网格搜索的 sampler.
optuna.samplers.RandomSampler	使用随机 sampling 的 sampler.
optuna.samplers.TPESampler	使用 TPE (Tree-structured Parzen Estimator) 算法的 sampler.
optuna.samplers.CmaEsSampler	A sampler using cmaes as the backend.
optuna.samplers.PartialFixedSampler	部分参数固定的 sampler.
optuna.samplers.NSGAIISampler	使用 NSGA-II 算法的多目标 sampler.
optuna.samplers.MOTPESampler	使用 MOTPE 算法的多目标 sampler.
optuna.samplers.IntersectionSearchSpace	用于计算 BaseStudy 的搜索空间的交集的类。
optuna.samplers.intersection_search_space	返回 BaseStudy 的搜索空间交集。

optuna.samplers.BaseSampler

class optuna.samplers.**BaseSampler**

Base class for samplers.

Optuna combines two types of sampling strategies, which are called *relative sampling* and *independent sampling*.

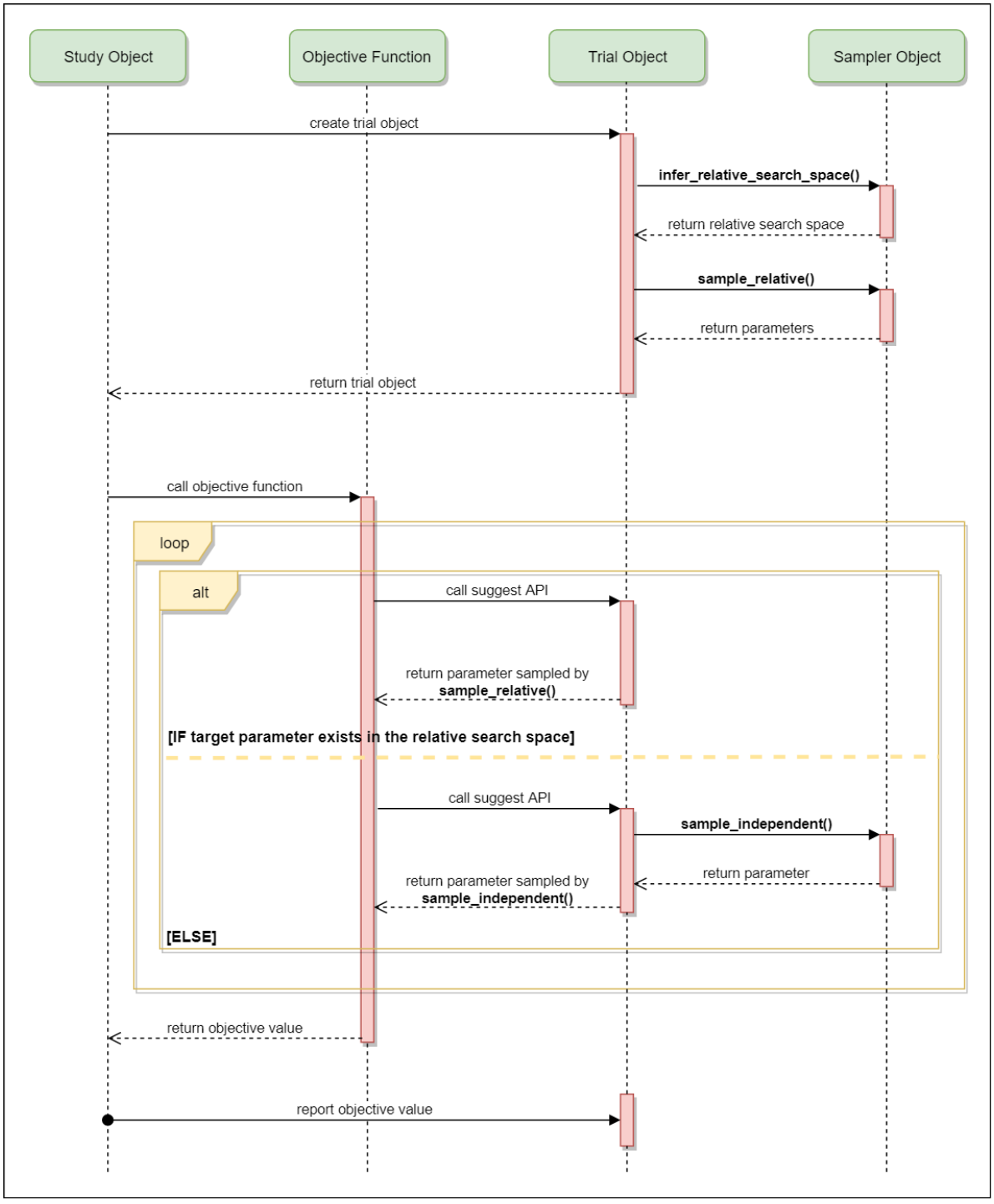
The relative sampling determines values of multiple parameters simultaneously so that sampling algorithms can use relationship between parameters (e.g., correlation). Target parameters of the relative sampling are described in a relative search space, which is determined by *infer_relative_search_space()*.

The independent sampling determines a value of a single parameter without considering any relationship between parameters. Target parameters of the independent sampling are the parameters not described in the relative search space.

More specifically, parameters are sampled by the following procedure. At the beginning of a trial, *infer_relative_search_space()* is called to determine the relative search space for the trial. Then, *sample_relative()* is invoked to sample parameters from the relative search space. During the execution

of the objective function, `sample_independent()` is used to sample parameters that don't belong to the relative search space.

The following figure depicts the lifetime of a trial and how the above three methods are called in the trial.



Methods

after_trial(study, trial, state, values)	Trial post-processing.
infer_relative_search_space(study, trial)	Infer the search space that will be used by relative sampling in the target trial.
reseed_rng()	Reseed sampler's random number generator.
sample_independent(study, trial, param_name, ...)	Sample a parameter for a given distribution.
sample_relative(study, trial, search_space)	Sample parameters in a given search space.

after_trial (*study*, *trial*, *state*, *values*)

Trial post-processing.

This method is called after the objective function returns and right before the trials is finished and its state is stored.

备注： Added in v2.4.0 as an experimental feature. The interface may change in newer versions without prior notice. See https://github.com/optuna/optuna/releases/tag/v2.4.0.

参数

- **study** (optuna.study.Study) –Target study object.
- **trial** (optuna.trial._frozen.FrozenTrial) –Target trial object. Take a copy before modifying this object.
- **state** (optuna.trial._state.TrialState) –Resulting trial state.
- **values** (*Optional[Sequence[float]]*) –Resulting trial values. Guaranteed to not be None if trial succeeded.

返回类型 None

abstract infer_relative_search_space (*study*, *trial*)

Infer the search space that will be used by relative sampling in the target trial.

This method is called right before *sample_relative()* method, and the search space returned by this method is passed to it. The parameters not contained in the search space will be sampled by using *sample_independent()* method.

参数

- **study** (optuna.study.Study) –Target study object.
- **trial** (optuna.trial._frozen.FrozenTrial) –Target trial object. Take a copy before modifying this object.

返回 A dictionary containing the parameter names and parameter's distributions.

返回类型 *Dict*[str, optuna.distributions.BaseDistribution]

参见:

Please refer to *intersection_search_space()* as an implementation of *infer_relative_search_space()*.

reseed_rng()

Reseed sampler's random number generator.

This method is called by the *Study* instance if trials are executed in parallel with the option `n_jobs>1`. In that case, the sampler instance will be replicated including the state of the random number generator, and they may suggest the same values. To prevent this issue, this method assigns a different seed to each random number generator.

返回类型 None

abstract sample_independent(*study*, *trial*, *param_name*, *param_distribution*)

Sample a parameter for a given distribution.

This method is called only for the parameters not contained in the search space returned by *sample_relative()* method. This method is suitable for sampling algorithms that do not use relationship between parameters such as random sampling and TPE.

备注: The failed trials are ignored by any build-in samplers when they sample new parameters. Thus, failed trials are regarded as deleted in the samplers' perspective.

参数

- **study** (optuna.study.Study) –Target study object.
- **trial** (optuna.trial._frozen.FrozenTrial) –Target trial object. Take a copy before modifying this object.
- **param_name** (*str*) –Name of the sampled parameter.
- **param_distribution** (*optuna.distributions.BaseDistribution*) – Distribution object that specifies a prior and/or scale of the sampling algorithm.

返回 A parameter value.

返回类型 *Any*

abstract sample_relative(*study*, *trial*, *search_space*)

Sample parameters in a given search space.

This method is called once at the beginning of each trial, i.e., right before the evaluation of the objective function. This method is suitable for sampling algorithms that use relationship between parameters such as Gaussian Process and CMA-ES.

备注： The failed trials are ignored by any build-in samplers when they sample new parameters. Thus, failed trials are regarded as deleted in the samplers' perspective.

参数

- **study** (optuna.study.Study) –Target study object.
- **trial** (optuna.trial._frozen.FrozenTrial) –Target trial object. Take a copy before modifying this object.
- **search_space** (*Dict[str, optuna.distributions.BaseDistribution]*) –The search space returned by *infer_relative_search_space()*.

返回 A dictionary containing the parameter names and the values.

返回类型 *Dict*[str, *Any*]

optuna.samplers.GridSampler

class optuna.samplers.**GridSampler**(*search_space*)

Sampler using grid search.

With *GridSampler*, the trials suggest all combinations of parameters in the given search space during the study.

示例

```
import optuna


def objective(trial):
    x = trial.suggest_float("x", -100, 100)
    y = trial.suggest_int("y", -100, 100)
    return x ** 2 + y ** 2


search_space = {"x": [-50, 0, 50], "y": [-99, 0, 99]}
study = optuna.create_study(sampler=optuna.samplers.GridSampler(search_space))
study.optimize(objective)
```

备注： *GridSampler* automatically stops the optimization if all combinations in the passed `search_space` have already been evaluated, internally invoking the *stop()* method.

备注： *GridSampler* does not take care of a parameter's quantization specified by discrete suggest methods but just samples one of values specified in the search space. E.g., in the following code snippet, either of `-0.5` or `0.5` is sampled as `x` instead of an integer point.

```
import optuna


def objective(trial):
    # The following suggest method specifies integer points between -5 and 5.
    x = trial.suggest_float("x", -5, 5, step=1)
    return x ** 2


# Non-int points are specified in the grid.
search_space = {"x": [-0.5, 0.5]}
study = optuna.create_study(sampler=optuna.samplers.GridSampler(search_space))
study.optimize(objective, n_trials=2)
```

备注： A parameter configuration in the grid is not considered finished until its trial is finished. Therefore, during distributed optimization where trials run concurrently, different workers will occasionally suggest the same parameter configuration. The total number of actual trials may therefore exceed the size of the grid.

备注： The grid is randomly shuffled and the order in which parameter configurations are suggested may vary. This is to reduce duplicate suggestions during distributed optimization.

参数 search_space (*Mapping[str, Sequence[Union[None, bool, int, float, str]]]*) – A dictionary whose key and value are a parameter name and the corresponding candidates of values, respectively.

返回类型 None

Methods

after_trial(study, trial, state, values)	Trial post-processing.
infer_relative_search_space(study, trial)	Infer the search space that will be used by relative sampling in the target trial.
reseed_rng()	Reseed sampler's random number generator.
sample_independent(study, trial, param_name, ...)	Sample a parameter for a given distribution.
sample_relative(study, trial, search_space)	Sample parameters in a given search space.

after_trial (*study*, *trial*, *state*, *values*)

Trial post-processing.

This method is called after the objective function returns and right before the trials is finished and its state is stored.

备注： Added in v2.4.0 as an experimental feature. The interface may change in newer versions without prior notice. See https://github.com/optuna/optuna/releases/tag/v2.4.0.

参数

- **study** (optuna.study.Study) –Target study object.
- **trial** (optuna.trial._frozen.FrozenTrial) –Target trial object. Take a copy before modifying this object.
- **state** (optuna.trial._state.TrialState) –Resulting trial state.
- **values** (*Optional[Sequence[float]]*) –Resulting trial values. Guaranteed to not be None if trial succeeded.

返回类型 None

infer_relative_search_space (*study*, *trial*)

Infer the search space that will be used by relative sampling in the target trial.

This method is called right before *sample_relative()* method, and the search space returned by this method is passed to it. The parameters not contained in the search space will be sampled by using *sample_independent()* method.

参数

- **study** (optuna.study.Study) –Target study object.
- **trial** (optuna.trial._frozen.FrozenTrial) –Target trial object. Take a copy before modifying this object.

返回 A dictionary containing the parameter names and parameter's distributions.

返回类型 *Dict*[str, optuna.distributions.BaseDistribution]

参见:

Please refer to *intersection_search_space()* as an implementation of *infer_relative_search_space()*.

reseed_rng()

Reseed sampler's random number generator.

This method is called by the *Study* instance if trials are executed in parallel with the option `n_jobs>1`. In that case, the sampler instance will be replicated including the state of the random number generator, and they may suggest the same values. To prevent this issue, this method assigns a different seed to each random number generator.

返回类型 None

sample_independent(*study*, *trial*, *param_name*, *param_distribution*)

Sample a parameter for a given distribution.

This method is called only for the parameters not contained in the search space returned by *sample_relative()* method. This method is suitable for sampling algorithms that do not use relationship between parameters such as random sampling and TPE.

备注: The failed trials are ignored by any build-in samplers when they sample new parameters. Thus, failed trials are regarded as deleted in the samplers' perspective.

参数

- **study** (optuna.study.Study) –Target study object.
- **trial** (optuna.trial._frozen.FrozenTrial) –Target trial object. Take a copy before modifying this object.
- **param_name** (*str*) –Name of the sampled parameter.
- **param_distribution** (*optuna.distributions.BaseDistribution*) – Distribution object that specifies a prior and/or scale of the sampling algorithm.

返回 A parameter value.

返回类型 *Any*

sample_relative(*study*, *trial*, *search_space*)

Sample parameters in a given search space.

This method is called once at the beginning of each trial, i.e., right before the evaluation of the objective function. This method is suitable for sampling algorithms that use relationship between parameters such as Gaussian Process and CMA-ES.

备注： The failed trials are ignored by any build-in samplers when they sample new parameters. Thus, failed trials are regarded as deleted in the samplers' perspective.

参数

- **study** (optuna.study.Study) –Target study object.
- **trial** (optuna.trial._frozen.FrozenTrial) –Target trial object. Take a copy before modifying this object.
- **search_space** (*Dict[str, optuna.distributions.BaseDistribution]*) –The search space returned by *infer_relative_search_space()*.

返回 A dictionary containing the parameter names and the values.

返回类型 *Dict*[str, *Any*]

optuna.samplers.RandomSampler

class optuna.samplers.**RandomSampler**(*seed=None*)

Sampler using random sampling.

This sampler is based on *independent sampling*. See also *BaseSampler* for more details of 'independent sampling'.

示例

```
import optuna
from optuna.samplers import RandomSampler


def objective(trial):
    x = trial.suggest_float("x", -5, 5)
    return x ** 2


study = optuna.create_study(sampler=RandomSampler())
study.optimize(objective, n_trials=10)
```

参数 **seed** (*Optional[int]*) –Seed for random number generator.

返回类型 None

Methods

after_trial(study, trial, state, values)	Trial post-processing.
infer_relative_search_space(study, trial)	Infer the search space that will be used by relative sampling in the target trial.
reseed_rng()	Reseed sampler's random number generator.
sample_independent(study, trial, param_name, ...)	Sample a parameter for a given distribution.
sample_relative(study, trial, search_space)	Sample parameters in a given search space.

after_trial (*study*, *trial*, *state*, *values*)

Trial post-processing.

This method is called after the objective function returns and right before the trials is finished and its state is stored.

备注： Added in v2.4.0 as an experimental feature. The interface may change in newer versions without prior notice. See https://github.com/optuna/optuna/releases/tag/v2.4.0.

参数

- **study** (optuna.study.Study) –Target study object.
- **trial** (optuna.trial._frozen.FrozenTrial) –Target trial object. Take a copy before modifying this object.
- **state** (optuna.trial._state.TrialState) –Resulting trial state.
- **values** (*Optional[Sequence[float]]*) –Resulting trial values. Guaranteed to not be None if trial succeeded.

返回类型 None

infer_relative_search_space (*study*, *trial*)

Infer the search space that will be used by relative sampling in the target trial.

This method is called right before *sample_relative()* method, and the search space returned by this method is passed to it. The parameters not contained in the search space will be sampled by using *sample_independent()* method.

参数

- **study** (optuna.study.Study) –Target study object.
- **trial** (optuna.trial._frozen.FrozenTrial) –Target trial object. Take a copy before modifying this object.

返回 A dictionary containing the parameter names and parameter's distributions.

返回类型 *Dict*[str, optuna.distributions.BaseDistribution]

参见:

Please refer to *intersection_search_space()* as an implementation of *infer_relative_search_space()*.

reseed_rng()

Reseed sampler's random number generator.

This method is called by the *Study* instance if trials are executed in parallel with the option `n_jobs>1`. In that case, the sampler instance will be replicated including the state of the random number generator, and they may suggest the same values. To prevent this issue, this method assigns a different seed to each random number generator.

返回类型 None

sample_independent(*study*, *trial*, *param_name*, *param_distribution*)

Sample a parameter for a given distribution.

This method is called only for the parameters not contained in the search space returned by *sample_relative()* method. This method is suitable for sampling algorithms that do not use relationship between parameters such as random sampling and TPE.

备注: The failed trials are ignored by any build-in samplers when they sample new parameters. Thus, failed trials are regarded as deleted in the samplers' perspective.

参数

- **study** (optuna.study.Study) –Target study object.
- **trial** (optuna.trial._frozen.FrozenTrial) –Target trial object. Take a copy before modifying this object.
- **param_name** (*str*) –Name of the sampled parameter.
- **param_distribution** (*optuna.distributions.BaseDistribution*) – Distribution object that specifies a prior and/or scale of the sampling algorithm.

返回 A parameter value.

返回类型 *Any*

sample_relative(*study*, *trial*, *search_space*)

Sample parameters in a given search space.

This method is called once at the beginning of each trial, i.e., right before the evaluation of the objective function. This method is suitable for sampling algorithms that use relationship between parameters such as Gaussian Process and CMA-ES.

备注： The failed trials are ignored by any build-in samplers when they sample new parameters. Thus, failed trials are regarded as deleted in the samplers' perspective.

参数

- **study** (optuna.study.Study) –Target study object.
- **trial** (optuna.trial._frozen.FrozenTrial) –Target trial object. Take a copy before modifying this object.
- **search_space** (*Dict[str, optuna.distributions.BaseDistribution]*) –The search space returned by *infer_relative_search_space()*.

返回 A dictionary containing the parameter names and the values.

返回类型 *Dict*[str, *Any*]

optuna.samplers.TPESampler

class optuna.samplers.**TPESampler**(*consider_prior=True*, *prior_weight=1.0*, *consider_magic_clip=True*, *consider_endpoints=False*, *n_startup_trials=10*, *n_ei_candidates=24*, *gamma=<function default_gamma>*, *weights=<function default_weights>*, *seed=None*, ***, *multivariate=False*, *warn_independent_sampling=True*)

Sampler using TPE (Tree-structured Parzen Estimator) algorithm.

This sampler is based on *independent sampling*. See also *BaseSampler* for more details of 'independent sampling'.

On each trial, for each parameter, TPE fits one Gaussian Mixture Model (GMM) `l(x)` to the set of parameter values associated with the best objective values, and another GMM `g(x)` to the remaining parameter values. It chooses the parameter value `x` that maximizes the ratio `l(x)/g(x)`.

For further information about TPE algorithm, please refer to the following papers:

- Algorithms for Hyper-Parameter Optimization
- Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures

示例

```
import optuna
from optuna.samplers import TPESampler


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


study = optuna.create_study(sampler=TPESampler())
study.optimize(objective, n_trials=10)
```

参数

- **consider_prior** (*bool*) –Enhance the stability of Parzen estimator by imposing a Gaussian prior when `True`. The prior is only effective if the sampling distribution is either *UniformDistribution*, *DiscreteUniformDistribution*, *LogUniformDistribution*, *IntUniformDistribution*, or *IntLogUniformDistribution*.
- **prior_weight** (*float*) –The weight of the prior. This argument is used in *UniformDistribution*, *DiscreteUniformDistribution*, *LogUniformDistribution*, *IntUniformDistribution*, *IntLogUniformDistribution*, and *CategoricalDistribution*.
- **consider_magic_clip** (*bool*) –Enable a heuristic to limit the smallest variances of Gaussians used in the Parzen estimator.
- **consider_endpoints** (*bool*) –Take endpoints of domains into account when calculating variances of Gaussians in Parzen estimator. See the original paper for details on the heuristics to calculate the variances.
- **n_startup_trials** (*int*) –The random sampling is used instead of the TPE algorithm until the given number of trials finish in the same study.
- **n_ei_candidates** (*int*) –Number of candidate samples used to calculate the expected improvement.
- **gamma** (*Callable[[int], int]*) –A function that takes the number of finished trials and returns the number of trials to form a density function for samples with low grains. See the original paper for more details.
- **weights** (*Callable[[int], numpy.ndarray]*) –A function that takes the number of finished trials and returns a weight for them. See Making a Science of Model Search:

Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures for more details.

- **seed** (*Optional[int]*) –Seed for random number generator.
- **multivariate** (*bool*) –If this is `True`, the multivariate TPE is used when suggesting parameters. The multivariate TPE is reported to outperform the independent TPE. See BOHB: Robust and Efficient Hyperparameter Optimization at Scale for more details.

备注： Added in v2.2.0 as an experimental feature. The interface may change in newer versions without prior notice. See https://github.com/optuna/optuna/releases/tag/v2.2.0.

- **warn_independent_sampling** (*bool*) –If this is `True` and `multivariate=True`, a warning message is emitted when the value of a parameter is sampled by using an independent sampler. If `multivariate=False`, this flag has no effect.

返回类型 None

Methods

after_trial(study, trial, state, values)	Trial post-processing.
hyperopt_parameters()	Return the the default parameters of hyperopt (v0.1.2).
infer_relative_search_space(study, trial)	Infer the search space that will be used by relative sampling in the target trial.
reseed_rng()	Reseed sampler's random number generator.
sample_independent(study, trial, param_name, ...)	Sample a parameter for a given distribution.
sample_relative(study, trial, search_space)	Sample parameters in a given search space.

after_trial(*study*, *trial*, *state*, *values*)

Trial post-processing.

This method is called after the objective function returns and right before the trials is finished and its state is stored.

备注： Added in v2.4.0 as an experimental feature. The interface may change in newer versions without prior notice. See https://github.com/optuna/optuna/releases/tag/v2.4.0.

参数

- **study** (optuna.study.Study) –Target study object.
- **trial** (optuna.trial._frozen.FrozenTrial) –Target trial object. Take a copy before modifying this object.
- **state** (optuna.trial._state.TrialState) –Resulting trial state.
- **values** (*Optional[Sequence[float]]*) –Resulting trial values. Guaranteed to not be None if trial succeeded.

返回类型 None

static hyperopt_parameters()

Return the the default parameters of hyperopt (v0.1.2).

TPESampler can be instantiated with the parameters returned by this method.

示例

Create a *TPESampler* instance with the default parameters of hyperopt.

```
import optuna
from optuna.samplers import TPESampler


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


sampler = TPESampler(**TPESampler.hyperopt_parameters())
study = optuna.create_study(sampler=sampler)
study.optimize(objective, n_trials=10)
```

返回 A dictionary containing the default parameters of hyperopt.

返回类型 *Dict*[str, *Any*]

infer_relative_search_space(*study*, *trial*)

Infer the search space that will be used by relative sampling in the target trial.

This method is called right before *sample_relative()* method, and the search space returned by this method is passed to it. The parameters not contained in the search space will be sampled by using *sample_independent()* method.

参数

- **study** (optuna.study.Study) –Target study object.

- **trial** (optuna.trial._frozen.FrozenTrial) –Target trial object. Take a copy before modifying this object.

返回 A dictionary containing the parameter names and parameter's distributions.

返回类型 *Dict*[str, optuna.distributions.BaseDistribution]

参见:

Please refer to *intersection_search_space()* as an implementation of *infer_relative_search_space()*.

reseed_rng()

Reseed sampler's random number generator.

This method is called by the *Study* instance if trials are executed in parallel with the option `n_jobs>1`. In that case, the sampler instance will be replicated including the state of the random number generator, and they may suggest the same values. To prevent this issue, this method assigns a different seed to each random number generator.

返回类型 None

sample_independent(*study*, *trial*, *param_name*, *param_distribution*)

Sample a parameter for a given distribution.

This method is called only for the parameters not contained in the search space returned by *sample_relative()* method. This method is suitable for sampling algorithms that do not use relationship between parameters such as random sampling and TPE.

备注: The failed trials are ignored by any build-in samplers when they sample new parameters. Thus, failed trials are regarded as deleted in the samplers' perspective.

参数

- **study** (optuna.study.Study) –Target study object.
- **trial** (optuna.trial._frozen.FrozenTrial) –Target trial object. Take a copy before modifying this object.
- **param_name** (*str*) –Name of the sampled parameter.
- **param_distribution** (*optuna.distributions.BaseDistribution*) – Distribution object that specifies a prior and/or scale of the sampling algorithm.

返回 A parameter value.

返回类型 *Any*

sample_relative(*study*, *trial*, *search_space*)

Sample parameters in a given search space.

This method is called once at the beginning of each trial, i.e., right before the evaluation of the objective function. This method is suitable for sampling algorithms that use relationship between parameters such as Gaussian Process and CMA-ES.

备注： The failed trials are ignored by any build-in samplers when they sample new parameters. Thus, failed trials are regarded as deleted in the samplers' perspective.

参数

- **study** (optuna.study.Study) –Target study object.
- **trial** (optuna.trial._frozen.FrozenTrial) –Target trial object. Take a copy before modifying this object.
- **search_space** (*Dict[str, optuna.distributions.BaseDistribution]*) –The search space returned by *infer_relative_search_space()*.

返回 A dictionary containing the parameter names and the values.

返回类型 *Dict*[str, *Any*]

optuna.samplers.CmaEsSampler

class optuna.samplers.**CmaEsSampler**(*x0=None*, *sigma0=None*, *n_startup_trials=1*, *independent_sampler=None*, *warn_independent_sampling=True*, *seed=None*, ***, *consider_pruned_trials=False*, *restart_strategy=None*, *inc_popsize=2*, *use_separable_cma=False*, *source_trials=None*)

A sampler using cmaes as the backend.

示例

Optimize a simple quadratic function by using *CmaEsSampler*.

```
import optuna


def objective(trial):
    x = trial.suggest_float("x", -1, 1)
```

(下页继续)

(续上页)

```
    y = trial.suggest_int("y", -1, 1)
    return x ** 2 + y


sampler = optuna.samplers.CmaEsSampler()
study = optuna.create_study(sampler=sampler)
study.optimize(objective, n_trials=20)
```

Please note that this sampler does not support CategoricalDistribution. If your search space contains categorical parameters, I recommend you to use *TPESampler* instead. Furthermore, there is room for performance improvements in parallel optimization settings. This sampler cannot use some trials for updating the parameters of multivariate normal distribution.

For further information about CMA-ES algorithm, please refer to the following papers:

- N. Hansen, The CMA Evolution Strategy: A Tutorial. arXiv:1604.00772, 2016.
- A. Auger and N. Hansen. A restart CMA evolution strategy with increasing population size. In Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2005), pages 1769–1776. IEEE Press, 2005.
- Raymond Ros, Nikolaus Hansen. A Simple Modification in CMA-ES Achieving Linear Time and Space Complexity. 10th International Conference on Parallel Problem Solving From Nature, Sep 2008, Dortmund, Germany. inria-00287367.
- Masahiro Nomura, Shuhei Watanabe, Youhei Akimoto, Yoshihiko Ozaki, Masaki Onishi. Warm Starting CMA-ES for Hyperparameter Optimization, AAAI. 2021.

参见:

You can also use *optuna.integration.PyCmaSampler* which is a sampler using cma library as the backend.

参数

- **x0** (*Optional[Dict[str, Any]]*) –A dictionary of an initial parameter values for CMA-ES. By default, the mean of `low` and `high` for each distribution is used. Note that `x0` is sampled uniformly within the search space domain for each restart if you specify `restart_strategy` argument.
- **sigma0** (*Optional[float]*) –Initial standard deviation of CMA-ES. By default, `sigma0` is set to `min_range / 6`, where `min_range` denotes the minimum range of the distributions in the search space.
- **seed** (*Optional[int]*) –A random seed for CMA-ES.
- **n_startup_trials** (*int*) –The independent sampling is used instead of the CMA-ES algorithm until the given number of trials finish in the same study.

- **independent_sampler** (*Optional[optuna.samplers._base.BaseSampler]*) –A *BaseSampler* instance that is used for independent sampling. The parameters not contained in the relative search space are sampled by this sampler. The search space for *CmaEsSampler* is determined by *intersection_search_space()*.

  If None is specified, *RandomSampler* is used as the default.

  参见:

  optuna.samplers module provides built-in independent samplers such as *RandomSampler* and *TPESampler*.

- **warn_independent_sampling** (*bool*) –If this is True, a warning message is emitted when the value of a parameter is sampled by using an independent sampler.

  Note that the parameters of the first trial in a study are always sampled via an independent sampler, so no warning messages are emitted in this case.

- **restart_strategy** (*Optional[str]*) –Strategy for restarting CMA-ES optimization when converges to a local minimum. If given None, CMA-ES will not restart (default). If given 'ipop', CMA-ES will restart with increasing population size. Please see also inc_popsize parameter.

  备注: Added in v2.1.0 as an experimental feature. The interface may change in newer versions without prior notice. See https://github.com/optuna/optuna/releases/tag/v2.1.0.

- **inc_popsize** (*int*) –Multiplier for increasing population size before each restart. This argument will be used when setting `restart_strategy = 'ipop'`.

- **consider_pruned_trials** (*bool*) –If this is True, the PRUNED trials are considered for sampling.

  备注: Added in v2.0.0 as an experimental feature. The interface may change in newer versions without prior notice. See https://github.com/optuna/optuna/releases/tag/v2.0.0.

  备注: It is suggested to set this flag False when the *MedianPruner* is used. On the other hand, it is suggested to set this flag True when the *HyperbandPruner* is used. Please see the benchmark result for the details.

- **use_separable_cma** (*bool*) –If this is True, the covariance matrix is constrained to be diagonal. Due to reduce the model complexity, the learning rate for the covariance matrix is increased. Consequently, this algorithm outperforms CMA-ES on separable functions.

备注：Added in v2.6.0 as an experimental feature. The interface may change in newer versions without prior notice. See https://github.com/optuna/optuna/releases/tag/v2.6.0.

- **source_trials** (*Optional[List[optuna.trial._frozen.FrozenTrial]]*) –This option is for Warm Starting CMA-ES, a method to transfer prior knowledge on similar HPO tasks through the initialization of CMA-ES. This method estimates a promising distribution from `source_trials` and generates the parameter of multivariate gaussian distribution. Please note that it is prohibited to use `x0`, `sigma0`, or `use_separable_cma` argument together.

备注：Added in v2.6.0 as an experimental feature. The interface may change in newer versions without prior notice. See https://github.com/optuna/optuna/releases/tag/v2.6.0.

引发 **ValueError** –If `restart_strategy` is not 'ipop' or `None`.

返回类型 None

Methods

after_trial(study, trial, state, values)	Trial post-processing.
infer_relative_search_space(study, trial)	Infer the search space that will be used by relative sampling in the target trial.
reseed_rng()	Reseed sampler's random number generator.
sample_independent(study, trial, param_name, ...)	Sample a parameter for a given distribution.
sample_relative(study, trial, search_space)	Sample parameters in a given search space.

after_trial(*study*, *trial*, *state*, *values*)

Trial post-processing.

This method is called after the objective function returns and right before the trials is finished and its state is stored.

备注：Added in v2.4.0 as an experimental feature. The interface may change in newer versions without prior notice. See https://github.com/optuna/optuna/releases/tag/v2.4.0.

参数

- **study** (`optuna.study.Study`) –Target study object.

- **trial** (optuna.trial._frozen.FrozenTrial) –Target trial object. Take a copy before modifying this object.
- **state** (optuna.trial._state.TrialState) –Resulting trial state.
- **values** (*Optional[Sequence[float]]*) –Resulting trial values. Guaranteed to not be None if trial succeeded.

返回类型 None

infer_relative_search_space(*study*, *trial*)

Infer the search space that will be used by relative sampling in the target trial.

This method is called right before *sample_relative()* method, and the search space returned by this method is passed to it. The parameters not contained in the search space will be sampled by using *sample_independent()* method.

参数

- **study** (optuna.study.Study) –Target study object.
- **trial** (optuna.trial._frozen.FrozenTrial) –Target trial object. Take a copy before modifying this object.

返回 A dictionary containing the parameter names and parameter's distributions.

返回类型 *Dict*[str, optuna.distributions.BaseDistribution]

参见:

Please refer to *intersection_search_space()* as an implementation of *infer_relative_search_space()*.

reseed_rng()

Reseed sampler's random number generator.

This method is called by the *Study* instance if trials are executed in parallel with the option n_jobs>1. In that case, the sampler instance will be replicated including the state of the random number generator, and they may suggest the same values. To prevent this issue, this method assigns a different seed to each random number generator.

返回类型 None

sample_independent(*study*, *trial*, *param_name*, *param_distribution*)

Sample a parameter for a given distribution.

This method is called only for the parameters not contained in the search space returned by *sample_relative()* method. This method is suitable for sampling algorithms that do not use relationship between parameters such as random sampling and TPE.

备注：The failed trials are ignored by any build-in samplers when they sample new parameters. Thus, failed trials are regarded as deleted in the samplers' perspective.

参数

- **study** (optuna.study.Study) –Target study object.
- **trial** (optuna.trial._frozen.FrozenTrial) –Target trial object. Take a copy before modifying this object.
- **param_name** (*str*) –Name of the sampled parameter.
- **param_distribution** (*optuna.distributions.BaseDistribution*) – Distribution object that specifies a prior and/or scale of the sampling algorithm.

返回 A parameter value.

返回类型 *Any*

sample_relative(*study*, *trial*, *search_space*)

Sample parameters in a given search space.

This method is called once at the beginning of each trial, i.e., right before the evaluation of the objective function. This method is suitable for sampling algorithms that use relationship between parameters such as Gaussian Process and CMA-ES.

备注：The failed trials are ignored by any build-in samplers when they sample new parameters. Thus, failed trials are regarded as deleted in the samplers' perspective.

参数

- **study** (optuna.study.Study) –Target study object.
- **trial** (optuna.trial._frozen.FrozenTrial) –Target trial object. Take a copy before modifying this object.
- **search_space** (*Dict[str, optuna.distributions.BaseDistribution]*) –The search space returned by *infer_relative_search_space()*.

返回 A dictionary containing the parameter names and the values.

返回类型 *Dict*[str, *Any*]

optuna.samplers.PartialFixedSampler

class optuna.samplers.**PartialFixedSampler**(*fixed_params*, *base_sampler*)

Sampler with partially fixed parameters.

2.4.0 新版功能.

示例

After several steps of optimization, you can fix the value of `y` and re-optimize it.

```
import optuna


def objective(trial):
    x = trial.suggest_float("x", -1, 1)
    y = trial.suggest_int("y", -1, 1)
    return x ** 2 + y


study = optuna.create_study()
study.optimize(objective, n_trials=10)

best_params = study.best_params
fixed_params = {"y": best_params["y"]}
partial_sampler = optuna.samplers.PartialFixedSampler(fixed_params, study.
↪sampler)

study.sampler = partial_sampler
study.optimize(objective, n_trials=10)
```

参数

- **fixed_params** (*Dict[str, Any]*) –A dictionary of parameters to be fixed.
- **base_sampler** (optuna.samplers._base.BaseSampler) –A sampler which samples unfixed parameters.

返回类型 None

备注： Added in v2.4.0 as an experimental feature. The interface may change in newer versions without prior notice. See https://github.com/optuna/optuna/releases/tag/v2.4.0.

Methods

after_trial(study, trial, state, values)	Trial post-processing.
infer_relative_search_space(study, trial)	Infer the search space that will be used by relative sampling in the target trial.
reseed_rng()	Reseed sampler's random number generator.
sample_independent(study, trial, param_name, ...)	Sample a parameter for a given distribution.
sample_relative(study, trial, search_space)	Sample parameters in a given search space.

after_trial (*study*, *trial*, *state*, *values*)

Trial post-processing.

This method is called after the objective function returns and right before the trials is finished and its state is stored.

备注： Added in v2.4.0 as an experimental feature. The interface may change in newer versions without prior notice. See https://github.com/optuna/optuna/releases/tag/v2.4.0.

参数

- **study** (optuna.study.Study) –Target study object.
- **trial** (optuna.trial._frozen.FrozenTrial) –Target trial object. Take a copy before modifying this object.
- **state** (optuna.trial._state.TrialState) –Resulting trial state.
- **values** (*Optional[Sequence[float]]*) –Resulting trial values. Guaranteed to not be None if trial succeeded.

返回类型 None

infer_relative_search_space (*study*, *trial*)

Infer the search space that will be used by relative sampling in the target trial.

This method is called right before *sample_relative()* method, and the search space returned by this method is passed to it. The parameters not contained in the search space will be sampled by using *sample_independent()* method.

参数

- **study** (optuna.study.Study) –Target study object.
- **trial** (optuna.trial._frozen.FrozenTrial) –Target trial object. Take a copy before modifying this object.

返回 A dictionary containing the parameter names and parameter's distributions.

返回类型 *Dict*[str, optuna.distributions.BaseDistribution]

参见:

Please refer to *intersection_search_space()* as an implementation of *infer_relative_search_space()*.

reseed_rng()

Reseed sampler's random number generator.

This method is called by the *Study* instance if trials are executed in parallel with the option `n_jobs>1`. In that case, the sampler instance will be replicated including the state of the random number generator, and they may suggest the same values. To prevent this issue, this method assigns a different seed to each random number generator.

返回类型 None

sample_independent(*study*, *trial*, *param_name*, *param_distribution*)

Sample a parameter for a given distribution.

This method is called only for the parameters not contained in the search space returned by *sample_relative()* method. This method is suitable for sampling algorithms that do not use relationship between parameters such as random sampling and TPE.

备注: The failed trials are ignored by any build-in samplers when they sample new parameters. Thus, failed trials are regarded as deleted in the samplers' perspective.

参数

- **study** (optuna.study.Study) –Target study object.
- **trial** (optuna.trial._frozen.FrozenTrial) –Target trial object. Take a copy before modifying this object.
- **param_name** (*str*) –Name of the sampled parameter.
- **param_distribution** (*optuna.distributions.BaseDistribution*) – Distribution object that specifies a prior and/or scale of the sampling algorithm.

返回 A parameter value.

返回类型 *Any*

sample_relative(*study*, *trial*, *search_space*)

Sample parameters in a given search space.

This method is called once at the beginning of each trial, i.e., right before the evaluation of the objective function. This method is suitable for sampling algorithms that use relationship between parameters such as Gaussian Process and CMA-ES.

备注： The failed trials are ignored by any build-in samplers when they sample new parameters. Thus, failed trials are regarded as deleted in the samplers' perspective.

参数

- **study** (*optuna.study.Study*) –Target study object.
- **trial** (*optuna.trial._frozen.FrozenTrial*) –Target trial object. Take a copy before modifying this object.
- **search_space** (*Dict[str, optuna.distributions.BaseDistribution]*) –The search space returned by *infer_relative_search_space()*.

返回 A dictionary containing the parameter names and the values.

返回类型 *Dict*[str, *Any*]

optuna.samplers.NSGAIISampler

class optuna.samplers.**NSGAIISampler**(*, *population_size=50*, *mutation_prob=None*, *crossover_prob=0.9*, *swapping_prob=0.5*, *seed=None*, *constraints_func=None*)

Multi-objective sampler using the NSGA-II algorithm.

NSGA-II stands for "Nondominated Sorting Genetic Algorithm II", which is a well known, fast and elitist multi-objective genetic algorithm.

For further information about NSGA-II, please refer to the following paper:

- A fast and elitist multiobjective genetic algorithm: NSGA-II

参数

- **population_size** (*int*) –Number of individuals (trials) in a generation.
- **mutation_prob** (*Optional[float]*) –Probability of mutating each parameter when creating a new individual. If `None` is specified, the value `1.0 / len(parent_trial.params)` is used where `parent_trial` is the parent trial of the target individual.
- **crossover_prob** (*float*) –Probability that a crossover (parameters swapping between parents) will occur when creating a new individual.

- **swapping_prob** (*float*) – Probability of swapping each parameter of the parents during crossover.
- **seed** (*Optional[int]*) – Seed for random number generator.
- **constraints_func** (*Optional[Callable[[optuna.trial._frozen.FrozenTrial], Sequence[float]]]*) – An optional function that computes the objective constraints. It must take a *FrozenTrial* and return the constraints. The return value must be a sequence of `float` s. A value strictly larger than 0 means that a constraints is violated. A value equal to or smaller than 0 is considered feasible. If constraints_func returns more than one value for a trial, that trial is considered feasible if and only if all values are equal to 0 or smaller.

  The constraints are handled by the constrained domination. A trial x is said to constrained-dominate a trial y, if any of the following conditions is true:

  1. Trial x is feasible and trial y is not.
  2. Trial x and y are both infeasible, but trial x has a smaller overall violation.
  3. Trial x and y are feasible and trial x dominates trial y.

备注： Added in v2.5.0 as an experimental feature. The interface may change in newer versions without prior notice. See https://github.com/optuna/optuna/releases/tag/v2.5.0.

返回类型 None

Methods

after_trial(study, trial, state, values)	Trial post-processing.
infer_relative_search_space(study, trial)	Infer the search space that will be used by relative sampling in the target trial.
reseed_rng()	Reseed sampler's random number generator.
sample_independent(study, trial, param_name, ...)	Sample a parameter for a given distribution.
sample_relative(study, trial, search_space)	Sample parameters in a given search space.

after_trial (*study*, *trial*, *state*, *values*)

Trial post-processing.

This method is called after the objective function returns and right before the trials is finished and its state is stored.

备注：Added in v2.4.0 as an experimental feature. The interface may change in newer versions without prior notice. See https://github.com/optuna/optuna/releases/tag/v2.4.0.

参数

- **study** (optuna.study.Study) –Target study object.
- **trial** (optuna.trial._frozen.FrozenTrial) –Target trial object. Take a copy before modifying this object.
- **state** (optuna.trial._state.TrialState) –Resulting trial state.
- **values** (*Optional[Sequence[float]]*) –Resulting trial values. Guaranteed to not be None if trial succeeded.

返回类型 None

infer_relative_search_space(*study*, *trial*)

Infer the search space that will be used by relative sampling in the target trial.

This method is called right before *sample_relative()* method, and the search space returned by this method is passed to it. The parameters not contained in the search space will be sampled by using *sample_independent()* method.

参数

- **study** (optuna.study.Study) –Target study object.
- **trial** (optuna.trial._frozen.FrozenTrial) –Target trial object. Take a copy before modifying this object.

返回 A dictionary containing the parameter names and parameter's distributions.

返回类型 *Dict*[str, optuna.distributions.BaseDistribution]

参见:

Please refer to *intersection_search_space()* as an implementation of *infer_relative_search_space()*.

reseed_rng()

Reseed sampler's random number generator.

This method is called by the *Study* instance if trials are executed in parallel with the option n_jobs>1. In that case, the sampler instance will be replicated including the state of the random number generator, and they may suggest the same values. To prevent this issue, this method assigns a different seed to each random number generator.

返回类型 None

sample_independent(*study*, *trial*, *param_name*, *param_distribution*)

Sample a parameter for a given distribution.

This method is called only for the parameters not contained in the search space returned by *sample_relative()* method. This method is suitable for sampling algorithms that do not use relationship between parameters such as random sampling and TPE.

备注： The failed trials are ignored by any build-in samplers when they sample new parameters. Thus, failed trials are regarded as deleted in the samplers' perspective.

参数

- **study** (optuna.study.Study) –Target study object.
- **trial** (optuna.trial._frozen.FrozenTrial) –Target trial object. Take a copy before modifying this object.
- **param_name** (*str*) –Name of the sampled parameter.
- **param_distribution** (*optuna.distributions.BaseDistribution*) – Distribution object that specifies a prior and/or scale of the sampling algorithm.

返回 A parameter value.

返回类型 *Any*

sample_relative(*study*, *trial*, *search_space*)

Sample parameters in a given search space.

This method is called once at the beginning of each trial, i.e., right before the evaluation of the objective function. This method is suitable for sampling algorithms that use relationship between parameters such as Gaussian Process and CMA-ES.

备注： The failed trials are ignored by any build-in samplers when they sample new parameters. Thus, failed trials are regarded as deleted in the samplers' perspective.

参数

- **study** (optuna.study.Study) –Target study object.
- **trial** (optuna.trial._frozen.FrozenTrial) –Target trial object. Take a copy before modifying this object.
- **search_space** (*Dict[str, optuna.distributions.BaseDistribution]*) –The search space returned by *infer_relative_search_space()*.

返回 A dictionary containing the parameter names and the values.

返回类型 *Dict*[str, *Any*]

optuna.samplers.MOTPESampler

class optuna.samplers.**MOTPESampler**(*, *consider_prior=True*, *prior_weight=1.0*, *consider_magic_clip=True*, *consider_endpoints=True*, *n_startup_trials=10*, *n_ehvi_candidates=24*, *gamma=<function default_gamma>*, *weights_above=<function _default_weights_above>*, *seed=None*)

Multi-objective sampler using the MOTPE algorithm.

This sampler is a multiobjective version of *TPESampler*.

For further information about MOTPE algorithm, please refer to the following paper:

- Multiobjective tree-structured parzen estimator for computationally expensive optimization problems

参数

- **consider_prior** (*bool*) –Enhance the stability of Parzen estimator by imposing a Gaussian prior when True. The prior is only effective if the sampling distribution is either *UniformDistribution*, *DiscreteUniformDistribution*, *LogUniformDistribution*, *IntUniformDistribution*, or *IntLogUniformDistribution*.
- **prior_weight** (*float*) –The weight of the prior. This argument is used in *UniformDistribution*, *DiscreteUniformDistribution*, *LogUniformDistribution*, *IntUniformDistribution*, *IntLogUniformDistribution*, and *CategoricalDistribution*.
- **consider_magic_clip** (*bool*) –Enable a heuristic to limit the smallest variances of Gaussians used in the Parzen estimator.
- **consider_endpoints** (*bool*) –Take endpoints of domains into account when calculating variances of Gaussians in Parzen estimator. See the original paper for details on the heuristics to calculate the variances.
- **n_startup_trials** (*int*) –The random sampling is used instead of the MOTPE algorithm until the given number of trials finish in the same study. 11 * number of variables - 1 is recommended in the original paper.
- **n_ehvi_candidates** (*int*) –Number of candidate samples used to calculate the expected hypervolume improvement.
- **gamma** (*Callable[[int], int]*) –A function that takes the number of finished trials and returns the number of trials to form a density function for samples with low grains. See

the original paper for more details.

- **weights_above** (*Callable[[int], numpy.ndarray]*) –A function that takes the number of finished trials and returns a weight for them. As default, weights are automatically calculated by the MOTPE's default strategy.
- **seed** (*Optional[int]*) –Seed for random number generator.

返回类型 None

备注： Initialization with Latin hypercube sampling may improve optimization performance. However, the current implementation only supports initialization with random sampling.

示例

```
import optuna

seed = 128
num_variables = 2
n_startup_trials = 11 * num_variables - 1


def objective(trial):
    x = []
    for i in range(1, num_variables + 1):
        x.append(trial.suggest_float(f"x{i}", 0.0, 2.0 * i))
    return x


sampler = optuna.samplers.MOTPESampler(
    n_startup_trials=n_startup_trials, n_ehvi_candidates=24, seed=seed
)
study = optuna.create_study(directions=["minimize"] * num_variables,␣
↪sampler=sampler)
study.optimize(objective, n_trials=n_startup_trials + 10)
```

备注： Added in v2.4.0 as an experimental feature. The interface may change in newer versions without prior notice. See https://github.com/optuna/optuna/releases/tag/v2.4.0.

Methods

after_trial(study, trial, state, values)	Trial post-processing.
hyperopt_parameters()	Return the the default parameters of hyperopt (v0.1.2).
infer_relative_search_space(study, trial)	Infer the search space that will be used by relative sampling in the target trial.
reseed_rng()	Reseed sampler's random number generator.
sample_independent(study, trial, param_name, ...)	Sample a parameter for a given distribution.
sample_relative(study, trial, search_space)	Sample parameters in a given search space.

after_trial (*study*, *trial*, *state*, *values*)

Trial post-processing.

This method is called after the objective function returns and right before the trials is finished and its state is stored.

备注： Added in v2.4.0 as an experimental feature. The interface may change in newer versions without prior notice. See https://github.com/optuna/optuna/releases/tag/v2.4.0.

参数

- **study** (optuna.study.Study) –Target study object.
- **trial** (optuna.trial._frozen.FrozenTrial) –Target trial object. Take a copy before modifying this object.
- **state** (optuna.trial._state.TrialState) –Resulting trial state.
- **values** (*Optional[Sequence[float]]*) –Resulting trial values. Guaranteed to not be None if trial succeeded.

返回类型 None

static hyperopt_parameters()

Return the the default parameters of hyperopt (v0.1.2).

TPESampler can be instantiated with the parameters returned by this method.

示例

Create a *TPESampler* instance with the default parameters of hyperopt.

```
import optuna
from optuna.samplers import TPESampler


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


sampler = TPESampler(**TPESampler.hyperopt_parameters())
study = optuna.create_study(sampler=sampler)
study.optimize(objective, n_trials=10)
```

返回 A dictionary containing the default parameters of hyperopt.

返回类型 *Dict*[str, *Any*]

infer_relative_search_space(*study*, *trial*)

Infer the search space that will be used by relative sampling in the target trial.

This method is called right before *sample_relative()* method, and the search space returned by this method is passed to it. The parameters not contained in the search space will be sampled by using *sample_independent()* method.

参数

- **study** (optuna.study.Study) –Target study object.
- **trial** (optuna.trial._frozen.FrozenTrial) –Target trial object. Take a copy before modifying this object.

返回 A dictionary containing the parameter names and parameter's distributions.

返回类型 *Dict*[str, optuna.distributions.BaseDistribution]

参见:

Please refer to *intersection_search_space()* as an implementation of *infer_relative_search_space()*.

reseed_rng()

Reseed sampler's random number generator.

This method is called by the *Study* instance if trials are executed in parallel with the option n_jobs>1. In that case, the sampler instance will be replicated including the state of the random number generator, and

they may suggest the same values. To prevent this issue, this method assigns a different seed to each random number generator.

返回类型 None

sample_independent(*study*, *trial*, *param_name*, *param_distribution*)

Sample a parameter for a given distribution.

This method is called only for the parameters not contained in the search space returned by *sample_relative()* method. This method is suitable for sampling algorithms that do not use relationship between parameters such as random sampling and TPE.

备注： The failed trials are ignored by any build-in samplers when they sample new parameters. Thus, failed trials are regarded as deleted in the samplers' perspective.

参数

- **study** (optuna.study.Study) –Target study object.
- **trial** (optuna.trial._frozen.FrozenTrial) –Target trial object. Take a copy before modifying this object.
- **param_name** (*str*) –Name of the sampled parameter.
- **param_distribution** (*optuna.distributions.BaseDistribution*) – Distribution object that specifies a prior and/or scale of the sampling algorithm.

返回 A parameter value.

返回类型 *Any*

sample_relative(*study*, *trial*, *search_space*)

Sample parameters in a given search space.

This method is called once at the beginning of each trial, i.e., right before the evaluation of the objective function. This method is suitable for sampling algorithms that use relationship between parameters such as Gaussian Process and CMA-ES.

备注： The failed trials are ignored by any build-in samplers when they sample new parameters. Thus, failed trials are regarded as deleted in the samplers' perspective.

参数

- **study** (optuna.study.Study) –Target study object.
- **trial** (optuna.trial._frozen.FrozenTrial) –Target trial object. Take a copy before modifying this object.

- **search_space** (*Dict[str, optuna.distributions.BaseDistribution]*) –The search space returned by *infer_relative_search_space()*.

返回 A dictionary containing the parameter names and the values.

返回类型 *Dict*[str, *Any*]

optuna.samplers.IntersectionSearchSpace

class optuna.samplers.**IntersectionSearchSpace**(*include_pruned=False*)

A class to calculate the intersection search space of a BaseStudy.

Intersection search space contains the intersection of parameter distributions that have been suggested in the completed trials of the study so far. If there are multiple parameters that have the same name but different distributions, neither is included in the resulting search space (i.e., the parameters with dynamic value ranges are excluded).

Note that an instance of this class is supposed to be used for only one study. If different studies are passed to *calculate()*, a ValueError is raised.

参数 **include_pruned** (*bool*) –Whether pruned trials should be included in the search space.

返回类型 None

Methods

calculate(study[, ordered_dict])	Returns the intersection search space of the BaseStudy.

calculate(*study*, *ordered_dict=False*)

Returns the intersection search space of the BaseStudy.

参数

- **study** (*optuna.study.BaseStudy*) –A study with completed trials.
- **ordered_dict** (*bool*) –A boolean flag determining the return type. If False, the returned object will be a dict. If True, the returned object will be an collections.OrderedDict sorted by keys, i.e. parameter names.

返回 A dictionary containing the parameter names and parameter’s distributions.

引发 **ValueError** –If different studies are passed into this method.

返回类型 *Dict*[str, optuna.distributions.BaseDistribution]

optuna.samplers.intersection_search_space

optuna.samplers.**intersection_search_space**(*study*, *ordered_dict=False*, *include_pruned=False*)

Return the intersection search space of the BaseStudy.

Intersection search space contains the intersection of parameter distributions that have been suggested in the completed trials of the study so far. If there are multiple parameters that have the same name but different distributions, neither is included in the resulting search space (i.e., the parameters with dynamic value ranges are excluded).

备注： *IntersectionSearchSpace* provides the same functionality with a much faster way. Please consider using it if you want to reduce execution time as much as possible.

参数

- **study** (*optuna.study.BaseStudy*) –A study with completed trials.
- **ordered_dict** (*bool*) –A boolean flag determining the return type. If False, the returned object will be a dict. If True, the returned object will be an collections.OrderedDict sorted by keys, i.e. parameter names.
- **include_pruned** (*bool*) –Whether pruned trials should be included in the search space.

返回 A dictionary containing the parameter names and parameter' s distributions.

返回类型 *Dict*[str, optuna.distributions.BaseDistribution]

6.3.11 optuna.storages

The *storages* module defines a BaseStorage class which abstracts a backend database and provides library-internal interfaces to the read/write histories of the studies and trials. Library users who wish to use storage solutions other than the default in-memory storage should use one of the child classes of BaseStorage documented below.

optuna.storages.RDBStorage	RDB 后端的存储类。
optuna.storages.RedisStorage	Redis 后端的存储类。

optuna.storages.RDBStorage

class optuna.storages.**RDBStorage**(*url*, *engine_kwargs=None*, *skip_compatibility_check=False*, ***, *heartbeat_interval=None*, *grace_period=None*, *failed_trial_callback=None*)

Storage class for RDB backend.

Note that library users can instantiate this class, but the attributes provided by this class are not supposed to be directly accessed by them.

示例

Create an *RDBStorage* instance with customized `pool_size` and `timeout` settings.

```
import optuna


def objective(trial):
    x = trial.suggest_float("x", -100, 100)
    return x ** 2


storage = optuna.storages.RDBStorage(
    url="sqlite:///:memory:",
    engine_kwargs={"pool_size": 20, "connect_args": {"timeout": 10}},
)

study = optuna.create_study(storage=storage)
study.optimize(objective, n_trials=10)
```

参数

- **url** (*str*) –URL of the storage.
- **engine_kwargs** (*Optional[Dict[str, Any]]*) –A dictionary of keyword arguments that is passed to sqlalchemy.engine.create_engine function.
- **skip_compatibility_check** (*bool*) –Flag to skip schema compatibility check if set to True.
- **heartbeat_interval** (*Optional[int]*) –Interval to record the heartbeat. It is recorded every `interval` seconds.
- **grace_period** (*Optional[int]*) –Grace period before a running trial is failed from the last heartbeat. If it is `None`, the grace period will be *2 * heartbeat_interval*.
- **failed_trial_callback** (*Optional[Callable[[optuna.Study, optuna.trial._frozen.FrozenTrial], None]]*) –A callback function that is invoked after failing each stale trial. The function must accept two parameters with the following types in this order: *Study* and `FrozenTrial`.

备注： The procedure to fail existing stale trials is called just before asking the study for a new

trial.

返回类型 None

备注： If you use MySQL, pool_pre_ping will be set to `True` by default to prevent connection timeout. You can turn it off with `engine_kwargs['pool_pre_ping']=False`, but it is recommended to keep the setting if execution time of your objective function is longer than the *wait_timeout* of your MySQL configuration.

引发

- **ValueError** –If the given *heartbeat_interval* or *grace_period* is not a positive integer.
- **RuntimeError** –If the a process that was failed by heartbeat but was actually running.

参数

- **url** (*str*) –
- **engine_kwargs** (*Optional[Dict[str, Any]]*) –
- **skip_compatibility_check** (*bool*) –
- **heartbeat_interval** (*Optional[int]*) –
- **grace_period** (*Optional[int]*) –
- **failed_trial_callback** (*Optional[Callable[[optuna.Study, optuna.trial._frozen.FrozenTrial], None]]*) –

返回类型 None

Methods

check_trial_is_updatable(trial_id, trial_state)	Check whether a trial state is updatable.
create_new_study([study_name])	Create a new study from a name.
create_new_trial(study_id[, template_trial])	Create and add a new trial to a study.
delete_study(study_id)	Delete a study.
fail_stale_trials(study_id)	Fail stale trials.
get_all_study_summaries()	Read a list of *StudySummary* objects.
get_all_trials(study_id[, deepcopy, states])	Read all trials in a study.
get_all_versions()	Return the schema version list.
get_best_trial(study_id)	Return the trial with the best value in a study.

下页继续

表 1 – 续上页

get_current_version()	Return the schema version currently used by this storage.
get_failed_trial_callback()	Get the failed trial callback function.
get_head_version()	Return the latest schema version.
get_heartbeat_interval()	Get the heartbeat interval if it is set.
get_n_trials(study_id[, state])	Count the number of trials in a study.
get_study_directions(study_id)	Read whether a study maximizes or minimizes an objective.
get_study_id_from_name(study_name)	Read the ID of a study.
get_study_id_from_trial_id(trial_id)	Read the ID of a study to which a trial belongs.
get_study_name_from_id(study_id)	Read the study name of a study.
get_study_system_attrs(study_id)	Read the optuna-internal attributes of a study.
get_study_user_attrs(study_id)	Read the user-defined attributes of a study.
get_trial(trial_id)	Read a trial.
get_trial_id_from_study_id_trial_number(...)	Read the trial id of a trial.
get_trial_number_from_id(trial_id)	Read the trial number of a trial.
get_trial_param(trial_id, param_name)	Read the parameter of a trial.
get_trial_params(trial_id)	Read the parameter dictionary of a trial.
get_trial_system_attrs(trial_id)	Read the optuna-internal attributes of a trial.
get_trial_user_attrs(trial_id)	Read the user-defined attributes of a trial.
is_heartbeat_enabled()	Check whether the storage enables the heartbeat.
read_trials_from_remote_storage(study_id)	Make an internal cache of trials up-to-date.
record_heartbeat(trial_id)	Record the heartbeat of the trial.
remove_session()	Removes the current session.
set_study_directions(study_id, directions)	Register optimization problem directions to a study.
set_study_system_attr(study_id, key, value)	Register an optuna-internal attribute to a study.
set_study_user_attr(study_id, key, value)	Register a user-defined attribute to a study.
set_trial_intermediate_value(trial_id, step, ...)	Report an intermediate value of an objective function.
set_trial_param(trial_id, param_name, ...)	Set a parameter to a trial.
set_trial_state(trial_id, state)	Update the state of a trial.
set_trial_system_attr(trial_id, key, value)	Set an optuna-internal attribute to a trial.
set_trial_user_attr(trial_id, key, value)	Set a user-defined attribute to a trial.
set_trial_values(trial_id, values)	Set return values of an objective function.
upgrade()	Upgrade the storage schema.

check_trial_is_updatable(*trial_id*, *trial_state*)

Check whether a trial state is updatable.

参数

- **trial_id** (*int*) –ID of the trial. Only used for an error message.
- **trial_state** (optuna.trial._state.TrialState) –Trial state to check.

引发 **RuntimeError** –If the trial is already finished.

返回类型 None

create_new_study(*study_name=None*)

Create a new study from a name.

If no name is specified, the storage class generates a name. The returned study ID is unique among all current and deleted studies.

参数 **study_name** (*Optional[str]*) –Name of the new study to create.

返回 ID of the created study.

引发 ***optuna.exceptions.DuplicatedStudyError*** –If a study with the same `study_name` already exists.

返回类型 int

create_new_trial(*study_id*, *template_trial=None*)

Create and add a new trial to a study.

The returned trial ID is unique among all current and deleted trials.

参数

- **study_id** (*int*) –ID of the study.
- **template_trial** (*Optional[optuna.trial._frozen.FrozenTrial]*) – Template `FronzenTrial` with default user-attributes, system-attributes, intermediate-values, and a state.

返回 ID of the created trial.

引发 **KeyError** –If no study with the matching `study_id` exists.

返回类型 int

delete_study(*study_id*)

Delete a study.

参数 **study_id** (*int*) –ID of the study.

引发 **KeyError** –If no study with the matching `study_id` exists.

返回类型 None

fail_stale_trials(*study_id*)

Fail stale trials.

The running trials whose heartbeat has not been updated for a long time will be failed, that is, those states will be changed to `FAIL`. The grace period is `2 * heartbeat_interval`.

参数 **study_id** (*int*) –ID of the related study.

返回 List of trial IDs of the failed trials.

返回类型 *List*[int]

get_all_study_summaries()

Read a list of `StudySummary` objects.

返回 A list of `StudySummary` objects.

返回类型 *List*[*optuna._study_summary.StudySummary*]

get_all_trials(*study_id*, *deepcopy=True*, *states=None*)

Read all trials in a study.

参数

- **study_id** (*int*) –ID of the study.
- **deepcopy** (*bool*) –Whether to copy the list of trials before returning. Set to `True` if you intend to update the list or elements of the list.
- **states** (*Optional[Tuple[optuna.trial._state.TrialState, ...]]*) –Trial states to filter on. If `None`, include all states.

返回 List of trials in the study.

引发 **KeyError** –If no study with the matching `study_id` exists.

返回类型 *List*[*optuna.trial._frozen.FrozenTrial*]

get_all_versions()

Return the schema version list.

返回类型 *List*[str]

get_best_trial(*study_id*)

Return the trial with the best value in a study.

This method is valid only during single-objective optimization.

参数 **study_id** (*int*) –ID of the study.

返回 The trial with the best objective value among all finished trials in the study.

引发

- **KeyError** –If no study with the matching `study_id` exists.
- **RuntimeError** –If the study has more than one direction.
- **ValueError** –If no trials have been completed.

返回类型 *optuna.trial._frozen.FrozenTrial*

get_current_version()

Return the schema version currently used by this storage.

返回类型 str

get_failed_trial_callback()

Get the failed trial callback function.

返回 The failed trial callback function if it is set, otherwise None.

返回类型 *Optional*[*Callable*[[*optuna.study.Study*, *optuna.trial._frozen.FrozenTrial*], None]]

get_head_version()

Return the latest schema version.

返回类型 str

get_heartbeat_interval()

Get the heartbeat interval if it is set.

返回 The heartbeat interval if it is set, otherwise None.

返回类型 *Optional*[int]

get_n_trials(*study_id*, *state=None*)

Count the number of trials in a study.

参数

- **study_id** (*int*) –ID of the study.
- **state** (*Optional[Union[Tuple[optuna.trial._state.TrialState, ...], optuna.trial._state.TrialState]]*) –Trial states to filter on. If None, include all states.

返回 Number of trials in the study.

引发 **KeyError** –If no study with the matching study_id exists.

返回类型 int

get_study_directions(*study_id*)

Read whether a study maximizes or minimizes an objective.

参数 **study_id** (*int*) –ID of a study.

返回 Optimization directions list of the study.

引发 **KeyError** –If no study with the matching study_id exists.

返回类型 *List*[*optuna._study_direction.StudyDirection*]

get_study_id_from_name(*study_name*)

Read the ID of a study.

参数 **study_name** (*str*) –Name of the study.

返回 ID of the study.

引发 **KeyError** –If no study with the matching `study_name` exists.

返回类型 int

get_study_id_from_trial_id(*trial_id*)

Read the ID of a study to which a trial belongs.

参数 **trial_id** (*int*) –ID of the trial.

返回 ID of the study.

引发 **KeyError** –If no trial with the matching `trial_id` exists.

返回类型 int

get_study_name_from_id(*study_id*)

Read the study name of a study.

参数 **study_id** (*int*) –ID of the study.

返回 Name of the study.

引发 **KeyError** –If no study with the matching `study_id` exists.

返回类型 str

get_study_system_attrs(*study_id*)

Read the optuna-internal attributes of a study.

参数 **study_id** (*int*) –ID of the study.

返回 Dictionary with the optuna-internal attributes of the study.

引发 **KeyError** –If no study with the matching `study_id` exists.

返回类型 *Dict*[str, *Any*]

get_study_user_attrs(*study_id*)

Read the user-defined attributes of a study.

参数 **study_id** (*int*) –ID of the study.

返回 Dictionary with the user attributes of the study.

引发 **KeyError** –If no study with the matching `study_id` exists.

返回类型 *Dict*[str, *Any*]

get_trial(*trial_id*)

Read a trial.

参数 **trial_id** (*int*) –ID of the trial.

返回 Trial with a matching trial ID.

引发 **KeyError** –If no trial with the matching `trial_id` exists.

返回类型 *optuna.trial._frozen.FrozenTrial*

get_trial_id_from_study_id_trial_number(*study_id*, *trial_number*)

Read the trial id of a trial.

参数

- **study_id** (*int*) –ID of the study.
- **trial_number** (*int*) –Number of the trial.

返回 ID of the trial.

引发 **KeyError** –If no trial with the matching `study_id` and `trial_number` exists.

返回类型 int

get_trial_number_from_id(*trial_id*)

Read the trial number of a trial.

备注: The trial number is only unique within a study, and is sequential.

参数 **trial_id** (*int*) –ID of the trial.

返回 Number of the trial.

引发 **KeyError** –If no trial with the matching `trial_id` exists.

返回类型 int

get_trial_param(*trial_id*, *param_name*)

Read the parameter of a trial.

参数

- **trial_id** (*int*) –ID of the trial.
- **param_name** (*str*) –Name of the parameter.

返回 Internal representation of the parameter.

引发 **KeyError** –If no trial with the matching `trial_id` exists. If no such parameter exists.

返回类型 float

get_trial_params(*trial_id*)

Read the parameter dictionary of a trial.

参数 **trial_id** (*int*) –ID of the trial.

返回 Dictionary of a parameters. Keys are parameter names and values are internal representations of the parameter values.

引发 **KeyError** –If no trial with the matching `trial_id` exists.

返回类型 *Dict*[str, *Any*]

get_trial_system_attrs(*trial_id*)

Read the optuna-internal attributes of a trial.

参数 **trial_id** (*int*) –ID of the trial.

返回 Dictionary with the optuna-internal attributes of the trial.

引发 **KeyError** –If no trial with the matching `trial_id` exists.

返回类型 *Dict*[str, *Any*]

get_trial_user_attrs(*trial_id*)

Read the user-defined attributes of a trial.

参数 **trial_id** (*int*) –ID of the trial.

返回 Dictionary with the user-defined attributes of the trial.

引发 **KeyError** –If no trial with the matching `trial_id` exists.

返回类型 *Dict*[str, *Any*]

is_heartbeat_enabled()

Check whether the storage enables the heartbeat.

返回 `True` if the storage supports the heartbeat and the return value of `get_heartbeat_interval()` is an integer, otherwise `False`.

返回类型 bool

read_trials_from_remote_storage(*study_id*)

Make an internal cache of trials up-to-date.

参数 **study_id** (*int*) –ID of the study.

引发 **KeyError** –If no study with the matching `study_id` exists.

返回类型 None

record_heartbeat(*trial_id*)

Record the heartbeat of the trial.

参数 **trial_id** (*int*) –ID of the trial.

返回类型 None

remove_session()

Removes the current session.

A session is stored in SQLAlchemy's ThreadLocalRegistry for each thread. This method closes and removes the session which is associated to the current thread. Particularly, under multi-thread use cases, it is important to call this method *from each thread*. Otherwise, all sessions and their associated DB connections are destructed by a thread that occasionally invoked the garbage collector. By default, it is not allowed to touch a SQLite connection from threads other than the thread that created the connection. Therefore, we need to explicitly close the connection from each thread.

返回类型 None

set_study_directions(*study_id*, *directions*)

Register optimization problem directions to a study.

参数

- **study_id** (*int*) –ID of the study.
- **directions** (*Sequence[optuna._study_direction.StudyDirection]*) –A sequence of direction whose element is either *MAXIMIZE* or *MINIMIZE*.

引发

- **KeyError** –If no study with the matching `study_id` exists.
- **ValueError** –If the directions are already set and the each coordinate of passed `directions` is the opposite direction or *NOT_SET*.

返回类型 None

set_study_system_attr(*study_id*, *key*, *value*)

Register an optuna-internal attribute to a study.

This method overwrites any existing attribute.

参数

- **study_id** (*int*) –ID of the study.
- **key** (*str*) –Attribute key.
- **value** (*Any*) –Attribute value. It should be JSON serializable.

引发 **KeyError** –If no study with the matching `study_id` exists.

返回类型 None

set_study_user_attr(*study_id*, *key*, *value*)

Register a user-defined attribute to a study.

This method overwrites any existing attribute.

参数

- **study_id** (*int*) –ID of the study.
- **key** (*str*) –Attribute key.
- **value** (*Any*) –Attribute value. It should be JSON serializable.

引发 **KeyError** –If no study with the matching `study_id` exists.

返回类型 None

set_trial_intermediate_value(*trial_id*, *step*, *intermediate_value*)

Report an intermediate value of an objective function.

This method overwrites any existing intermediate value associated with the given step.

参数

- **trial_id** (*int*) –ID of the trial.
- **step** (*int*) –Step of the trial (e.g., the epoch when training a neural network).
- **intermediate_value** (*float*) –Intermediate value corresponding to the step.

引发

- **KeyError** –If no trial with the matching `trial_id` exists.
- **RuntimeError** –If the trial is already finished.

返回类型 None

set_trial_param(*trial_id*, *param_name*, *param_value_internal*, *distribution*)

Set a parameter to a trial.

参数

- **trial_id** (*int*) –ID of the trial.
- **param_name** (*str*) –Name of the parameter.
- **param_value_internal** (*float*) –Internal representation of the parameter value.
- **distribution** (*optuna.distributions.BaseDistribution*) –Sampled distribution of the parameter.

引发

- **KeyError** –If no trial with the matching `trial_id` exists.
- **RuntimeError** –If the trial is already finished.

返回类型 None

set_trial_state(*trial_id*, *state*)

Update the state of a trial.

参数

- **trial_id** (*int*) –ID of the trial.
- **state** (optuna.trial._state.TrialState) –New state of the trial.

返回 True if the state is successfully updated. False if the state is kept the same. The latter happens when this method tries to update the state of *RUNNING* trial to *RUNNING*.

引发

- **KeyError** –If no trial with the matching trial_id exists.
- **RuntimeError** –If the trial is already finished.

返回类型 bool

set_trial_system_attr(*trial_id*, *key*, *value*)

Set an optuna-internal attribute to a trial.

This method overwrites any existing attribute.

参数

- **trial_id** (*int*) –ID of the trial.
- **key** (*str*) –Attribute key.
- **value** (*Any*) –Attribute value. It should be JSON serializable.

引发

- **KeyError** –If no trial with the matching trial_id exists.
- **RuntimeError** –If the trial is already finished.

返回类型 None

set_trial_user_attr(*trial_id*, *key*, *value*)

Set a user-defined attribute to a trial.

This method overwrites any existing attribute.

参数

- **trial_id** (*int*) –ID of the trial.
- **key** (*str*) –Attribute key.
- **value** (*Any*) –Attribute value. It should be JSON serializable.

引发

- **KeyError** –If no trial with the matching trial_id exists.

- **RuntimeError** –If the trial is already finished.

返回类型 None

set_trial_values(*trial_id*, *values*)

Set return values of an objective function.

This method overwrites any existing trial values.

参数

- **trial_id** (*int*) –ID of the trial.
- **values** (*Sequence[float]*) –Values of the objective function.

引发

- **KeyError** –If no trial with the matching `trial_id` exists.
- **RuntimeError** –If the trial is already finished.

返回类型 None

upgrade()

Upgrade the storage schema.

返回类型 None

optuna.storages.RedisStorage

class optuna.storages.**RedisStorage**(*url*)

Storage class for Redis backend.

Note that library users can instantiate this class, but the attributes provided by this class are not supposed to be directly accessed by them.

示例

We create an *RedisStorage* instance using the given redis database URL.

```
import optuna


def objective(trial):
    ...


storage = optuna.storages.RedisStorage(
    url="redis://passwd@localhost:port/db",
```

(下页继续)

(续上页)

```
)

study = optuna.create_study(storage=storage)
study.optimize(objective)
```

参数 **url** (*str*) –URL of the redis storage, password and db are optional. (ie: redis://localhost:6379)

返回类型 None

备注： If you use plan to use Redis as a storage mechanism for optuna, make sure Redis in installed and running. Please execute `$ pip install -U redis` to install redis python library.

备注： Added in v1.4.0 as an experimental feature. The interface may change in newer versions without prior notice. See https://github.com/optuna/optuna/releases/tag/v1.4.0.

Methods

check_trial_is_updatable(trial_id, trial_state)	Check whether a trial state is updatable.
create_new_study([study_name])	Create a new study from a name.
create_new_trial(study_id[, template_trial])	Create and add a new trial to a study.
delete_study(study_id)	Delete a study.
fail_stale_trials(study_id)	Fail stale trials.
get_all_study_summaries()	Read a list of *StudySummary* objects.
get_all_trials(study_id[, deepcopy, states])	Read all trials in a study.
get_best_trial(study_id)	Return the trial with the best value in a study.
get_failed_trial_callback()	Get the failed trial callback function.
get_heartbeat_interval()	Get the heartbeat interval if it is set.
get_n_trials(study_id[, state])	Count the number of trials in a study.
get_study_directions(study_id)	Read whether a study maximizes or minimizes an objective.
get_study_id_from_name(study_name)	Read the ID of a study.
get_study_id_from_trial_id(trial_id)	Read the ID of a study to which a trial belongs.
get_study_name_from_id(study_id)	Read the study name of a study.
get_study_system_attrs(study_id)	Read the optuna-internal attributes of a study.
get_study_user_attrs(study_id)	Read the user-defined attributes of a study.

下页继续

表 2 – 续上页

get_trial(trial_id)	Read a trial.
get_trial_id_from_study_id_trial_number()	Read the trial id of a trial.
get_trial_number_from_id(trial_id)	Read the trial number of a trial.
get_trial_param(trial_id, param_name)	Read the parameter of a trial.
get_trial_params(trial_id)	Read the parameter dictionary of a trial.
get_trial_system_attrs(trial_id)	Read the optuna-internal attributes of a trial.
get_trial_user_attrs(trial_id)	Read the user-defined attributes of a trial.
is_heartbeat_enabled()	Check whether the storage enables the heartbeat.
read_trials_from_remote_storage(study_id)	Make an internal cache of trials up-to-date.
record_heartbeat(trial_id)	Record the heartbeat of the trial.
remove_session()	Clean up all connections to a database.
set_study_directions(study_id, directions)	Register optimization problem directions to a study.
set_study_system_attr(study_id, key, value)	Register an optuna-internal attribute to a study.
set_study_user_attr(study_id, key, value)	Register a user-defined attribute to a study.
set_trial_intermediate_value(trial_id, step, ...)	Report an intermediate value of an objective function.
set_trial_param(trial_id, param_name, ...)	Set a parameter to a trial.
set_trial_state(trial_id, state)	Update the state of a trial.
set_trial_system_attr(trial_id, key, value)	Set an optuna-internal attribute to a trial.
set_trial_user_attr(trial_id, key, value)	Set a user-defined attribute to a trial.
set_trial_values(trial_id, values)	Set return values of an objective function.

check_trial_is_updatable(*trial_id*, *trial_state*)

Check whether a trial state is updatable.

参数

- **trial_id** (*int*) –ID of the trial. Only used for an error message.
- **trial_state** (optuna.trial._state.TrialState) –Trial state to check.

引发 **RuntimeError** –If the trial is already finished.

返回类型 None

create_new_study(*study_name=None*)

Create a new study from a name.

If no name is specified, the storage class generates a name. The returned study ID is unique among all current and deleted studies.

参数 **study_name** (*Optional[str]*) –Name of the new study to create.

返回 ID of the created study.

引发 *optuna.exceptions.DuplicatedStudyError* –If a study with the same `study_name` already exists.

返回类型 int

create_new_trial(*study_id*, *template_trial=None*)

Create and add a new trial to a study.

The returned trial ID is unique among all current and deleted trials.

参数

- **study_id** (*int*) –ID of the study.
- **template_trial** (*Optional[optuna.trial._frozen.FrozenTrial]*) – Template `FronzenTrial` with default user-attributes, system-attributes, intermediate-values, and a state.

返回 ID of the created trial.

引发 **KeyError** –If no study with the matching `study_id` exists.

返回类型 int

delete_study(*study_id*)

Delete a study.

参数 **study_id** (*int*) –ID of the study.

引发 **KeyError** –If no study with the matching `study_id` exists.

返回类型 None

fail_stale_trials(*study_id*)

Fail stale trials.

The running trials whose heartbeat has not been updated for a long time will be failed, that is, those states will be changed to *FAIL*. The grace period is `2 * heartbeat_interval`.

参数 **study_id** (*int*) –ID of the related study.

返回 List of trial IDs of the failed trials.

返回类型 *List*[int]

get_all_study_summaries()

Read a list of *StudySummary* objects.

返回 A list of *StudySummary* objects.

返回类型 *List*[*optuna._study_summary.StudySummary*]

get_all_trials(*study_id*, *deepcopy=True*, *states=None*)

Read all trials in a study.

参数

- **study_id** (*int*) –ID of the study.
- **deepcopy** (*bool*) –Whether to copy the list of trials before returning. Set to `True` if you intend to update the list or elements of the list.
- **states** (*Optional[Tuple[optuna.trial._state.TrialState, ...]]*) –Trial states to filter on. If `None`, include all states.

返回 List of trials in the study.

引发 **KeyError** –If no study with the matching `study_id` exists.

返回类型 *List*[*optuna.trial._frozen.FrozenTrial*]

get_best_trial(*study_id*)

Return the trial with the best value in a study.

This method is valid only during single-objective optimization.

参数 **study_id** (*int*) –ID of the study.

返回 The trial with the best objective value among all finished trials in the study.

引发

- **KeyError** –If no study with the matching `study_id` exists.
- **RuntimeError** –If the study has more than one direction.
- **ValueError** –If no trials have been completed.

返回类型 *optuna.trial._frozen.FrozenTrial*

get_failed_trial_callback()

Get the failed trial callback function.

返回 The failed trial callback function if it is set, otherwise `None`.

返回类型 *Optional*[*Callable*[[*optuna.study.Study*, *optuna.trial._frozen.FrozenTrial*], None]]

get_heartbeat_interval()

Get the heartbeat interval if it is set.

返回 The heartbeat interval if it is set, otherwise `None`.

返回类型 *Optional*[int]

get_n_trials(*study_id*, *state=None*)

Count the number of trials in a study.

参数

- **study_id** (*int*) –ID of the study.

- **state** (*Optional[Union[Tuple[optuna.trial._state.TrialState, ...], optuna.trial._state.TrialState]]*) –Trial states to filter on. If `None`, include all states.

返回 Number of trials in the study.

引发 **KeyError** –If no study with the matching `study_id` exists.

返回类型 int

get_study_directions(*study_id*)

Read whether a study maximizes or minimizes an objective.

参数 **study_id** (*int*) –ID of a study.

返回 Optimization directions list of the study.

引发 **KeyError** –If no study with the matching `study_id` exists.

返回类型 *List*[*optuna._study_direction.StudyDirection*]

get_study_id_from_name(*study_name*)

Read the ID of a study.

参数 **study_name** (*str*) –Name of the study.

返回 ID of the study.

引发 **KeyError** –If no study with the matching `study_name` exists.

返回类型 int

get_study_id_from_trial_id(*trial_id*)

Read the ID of a study to which a trial belongs.

参数 **trial_id** (*int*) –ID of the trial.

返回 ID of the study.

引发 **KeyError** –If no trial with the matching `trial_id` exists.

返回类型 int

get_study_name_from_id(*study_id*)

Read the study name of a study.

参数 **study_id** (*int*) –ID of the study.

返回 Name of the study.

引发 **KeyError** –If no study with the matching `study_id` exists.

返回类型 str

get_study_system_attrs(*study_id*)

Read the optuna-internal attributes of a study.

参数 **study_id** (*int*) –ID of the study.

返回 Dictionary with the optuna-internal attributes of the study.

引发 **KeyError** –If no study with the matching `study_id` exists.

返回类型 *Dict*[str, *Any*]

get_study_user_attrs(*study_id*)

Read the user-defined attributes of a study.

参数 **study_id** (*int*) –ID of the study.

返回 Dictionary with the user attributes of the study.

引发 **KeyError** –If no study with the matching `study_id` exists.

返回类型 *Dict*[str, *Any*]

get_trial(*trial_id*)

Read a trial.

参数 **trial_id** (*int*) –ID of the trial.

返回 Trial with a matching trial ID.

引发 **KeyError** –If no trial with the matching `trial_id` exists.

返回类型 *optuna.trial._frozen.FrozenTrial*

get_trial_id_from_study_id_trial_number(*study_id*, *trial_number*)

Read the trial id of a trial.

参数

- **study_id** (*int*) –ID of the study.
- **trial_number** (*int*) –Number of the trial.

返回 ID of the trial.

引发 **KeyError** –If no trial with the matching `study_id` and `trial_number` exists.

返回类型 int

get_trial_number_from_id(*trial_id*)

Read the trial number of a trial.

备注：The trial number is only unique within a study, and is sequential.

参数 **trial_id** (*int*) –ID of the trial.

返回 Number of the trial.

引发 **KeyError** –If no trial with the matching `trial_id` exists.

返回类型 int

get_trial_param(*trial_id*, *param_name*)

Read the parameter of a trial.

参数

- **trial_id** (*int*) –ID of the trial.
- **param_name** (*str*) –Name of the parameter.

返回 Internal representation of the parameter.

引发 **KeyError** –If no trial with the matching `trial_id` exists. If no such parameter exists.

返回类型 float

get_trial_params(*trial_id*)

Read the parameter dictionary of a trial.

参数 **trial_id** (*int*) –ID of the trial.

返回 Dictionary of a parameters. Keys are parameter names and values are internal representations of the parameter values.

引发 **KeyError** –If no trial with the matching `trial_id` exists.

返回类型 *Dict*[str, *Any*]

get_trial_system_attrs(*trial_id*)

Read the optuna-internal attributes of a trial.

参数 **trial_id** (*int*) –ID of the trial.

返回 Dictionary with the optuna-internal attributes of the trial.

引发 **KeyError** –If no trial with the matching `trial_id` exists.

返回类型 *Dict*[str, *Any*]

get_trial_user_attrs(*trial_id*)

Read the user-defined attributes of a trial.

参数 **trial_id** (*int*) –ID of the trial.

返回 Dictionary with the user-defined attributes of the trial.

引发 **KeyError** –If no trial with the matching `trial_id` exists.

返回类型 *Dict*[str, *Any*]

is_heartbeat_enabled()

Check whether the storage enables the heartbeat.

返回 `True` if the storage supports the heartbeat and the return value of `get_heartbeat_interval()` is an integer, otherwise `False`.

返回类型 bool

read_trials_from_remote_storage(*study_id*)

Make an internal cache of trials up-to-date.

参数 **study_id** (*int*) –ID of the study.

引发 **KeyError** –If no study with the matching `study_id` exists.

返回类型 None

record_heartbeat(*trial_id*)

Record the heartbeat of the trial.

参数 **trial_id** (*int*) –ID of the trial.

返回类型 None

remove_session()

Clean up all connections to a database.

返回类型 None

set_study_directions(*study_id*, *directions*)

Register optimization problem directions to a study.

参数

- **study_id** (*int*) –ID of the study.
- **directions** (*Sequence[optuna._study_direction.StudyDirection]*) –A sequence of direction whose element is either *MAXIMIZE* or *MINIMIZE*.

引发

- **KeyError** –If no study with the matching `study_id` exists.
- **ValueError** –If the directions are already set and the each coordinate of passed `directions` is the opposite direction or *NOT_SET*.

返回类型 None

set_study_system_attr(*study_id*, *key*, *value*)

Register an optuna-internal attribute to a study.

This method overwrites any existing attribute.

参数

- **study_id** (*int*) –ID of the study.
- **key** (*str*) –Attribute key.
- **value** (*Any*) –Attribute value. It should be JSON serializable.

引发 **KeyError** –If no study with the matching `study_id` exists.

返回类型 None

set_study_user_attr(*study_id*, *key*, *value*)

Register a user-defined attribute to a study.

This method overwrites any existing attribute.

参数

- **study_id** (*int*) –ID of the study.
- **key** (*str*) –Attribute key.
- **value** (*Any*) –Attribute value. It should be JSON serializable.

引发 **KeyError** –If no study with the matching `study_id` exists.

返回类型 None

set_trial_intermediate_value(*trial_id*, *step*, *intermediate_value*)

Report an intermediate value of an objective function.

This method overwrites any existing intermediate value associated with the given step.

参数

- **trial_id** (*int*) –ID of the trial.
- **step** (*int*) –Step of the trial (e.g., the epoch when training a neural network).
- **intermediate_value** (*float*) –Intermediate value corresponding to the step.

引发

- **KeyError** –If no trial with the matching `trial_id` exists.
- **RuntimeError** –If the trial is already finished.

返回类型 None

set_trial_param(*trial_id*, *param_name*, *param_value_internal*, *distribution*)

Set a parameter to a trial.

参数

- **trial_id** (*int*) –ID of the trial.
- **param_name** (*str*) –Name of the parameter.
- **param_value_internal** (*float*) –Internal representation of the parameter value.

- **distribution** (*optuna.distributions.BaseDistribution*) –Sampled distribution of the parameter.

引发

- **KeyError** –If no trial with the matching `trial_id` exists.
- **RuntimeError** –If the trial is already finished.

返回类型 None

set_trial_state(*trial_id*, *state*)

Update the state of a trial.

参数

- **trial_id** (*int*) –ID of the trial.
- **state** (`optuna.trial._state.TrialState`) –New state of the trial.

返回 `True` if the state is successfully updated. `False` if the state is kept the same. The latter happens when this method tries to update the state of *RUNNING* trial to *RUNNING*.

引发

- **KeyError** –If no trial with the matching `trial_id` exists.
- **RuntimeError** –If the trial is already finished.

返回类型 bool

set_trial_system_attr(*trial_id*, *key*, *value*)

Set an optuna-internal attribute to a trial.

This method overwrites any existing attribute.

参数

- **trial_id** (*int*) –ID of the trial.
- **key** (*str*) –Attribute key.
- **value** (*Any*) –Attribute value. It should be JSON serializable.

引发

- **KeyError** –If no trial with the matching `trial_id` exists.
- **RuntimeError** –If the trial is already finished.

返回类型 None

set_trial_user_attr(*trial_id*, *key*, *value*)

Set a user-defined attribute to a trial.

This method overwrites any existing attribute.

参数

- **trial_id** (*int*) –ID of the trial.
- **key** (*str*) –Attribute key.
- **value** (*Any*) –Attribute value. It should be JSON serializable.

引发

- **KeyError** –If no trial with the matching `trial_id` exists.
- **RuntimeError** –If the trial is already finished.

返回类型 None

set_trial_values(*trial_id*, *values*)

Set return values of an objective function.

This method overwrites any existing trial values.

参数

- **trial_id** (*int*) –ID of the trial.
- **values** (*Sequence[float]*) –Values of the objective function.

引发

- **KeyError** –If no trial with the matching `trial_id` exists.
- **RuntimeError** –If the trial is already finished.

返回类型 None

6.3.12 optuna.structs

该模块已被弃用，其以前的功能移至*optuna.trial* 和*optuna.study* 。

class optuna.structs.**TrialState**(*value*)

Trial 的状态

RUNNING

运行中的*Trial*.

COMPLETE

已完成且未触发错误的*Trial*.

PRUNED

已经被*TrialPruned* 剪枝的*Trial*.

FAIL

由于未捕获的错误，该*Trial* 已经失败。

1.4.0 版后已移除: 该类已废弃。请改用*TrialState*.

class optuna.structs.**StudyDirection**(*value*)

Study 的方向。

NOT_SET

方向未设置。

MINIMIZE

Study 将最小化目标函数

MAXIMIZE

Study 将最大化目标函数。

1.4.0 版后已移除: 该类已废弃，请改用*StudyDirection*.

class optuna.structs.**FrozenTrial**(*number*, *state*, *value*, *datetime_start*, *datetime_complete*, *params*, *distributions*, *user_attrs*, *system_attrs*, *intermediate_values*, *trial_id*, *, *values=None*)

警告: 在 v1.4.0 中被弃用。该特性将在未来被移除。目前我们计划在 v3.0.0 中移除它，但也可能会改变。参见 https://github.com/optuna/optuna/releases/tag/v1.4.0.

该类已经迁移至*trial*, 请改用*FrozenTrial*.

参数

- **number** (*int*) –
- **state** (optuna.trial._state.TrialState) –
- **value** (*Optional[float]*) –
- **datetime_start** (*Optional[datetime.datetime]*) –
- **datetime_complete** (*Optional[datetime.datetime]*) –
- **params** (*Dict[str, Any]*) –
- **distributions** (*Dict[str, optuna.distributions.BaseDistribution]*) –
- **user_attrs** (*Dict[str, Any]*) –
- **system_attrs** (*Dict[str, Any]*) –
- **intermediate_values** (*Dict[int, float]*) –

- **trial_id** (*int*) –
- **values** (*Optional[Sequence[float]]*) –

返回类型 None

property distributions: Dict[str, optuna.distributions.BaseDistribution]

包含了 params 分布的字典。

property duration: Optional[datetime.timedelta]

返回完成一个 trial 所消耗的时间。

返回 消耗的时间。

property last_step: Optional[int]

返回 trial 中 *intermediate_values* 的最大步数。

返回 *intermediate_values* 的最大步数。

report(*value*, *step*)

report 函数接口。

由于*FrozenTrial* 未被剪枝，该函数将不做任何处理。

参见:

Please refer to *should_prune()*.

参数

- **value** (*float*) –目标函数返回的值。
- **step** (*int*) –Trial 的步骤（比如，神经网络训练中的 epoch 数）。注意，pruners 假定 step 从零开始计算。比如*MedianPruner* 将仅检查 step 是否小于 n_warmup_steps 作为热身机制。

返回类型 None

should_prune()

建议该 trial 是否应该被剪枝。

无论剪枝算法是什么，建议值永远是 False.

备注: *FrozenTrial* 只对参数组合进行一次采样。

返回 False.

返回类型 bool

class optuna.structs.**StudySummary**(*study_name*, *direction*, *best_trial*, *user_attrs*, *system_attrs*, *n_trials*, *datetime_start*, *study_id*, *, *directions=None*)

> **警告:** 在 v1.4.0 中被弃用。该特性将在未来被移除。目前我们计划在 v3.0.0 中移除它，但也可能会改变。参见 https://github.com/optuna/optuna/releases/tag/v1.4.0.
>
> 该类已经迁移到*study*, 请改用*StudySummary*.

参数

- **study_name** (*str*) –
- **direction** (*Optional[optuna._study_direction.StudyDirection]*) –
- **best_trial** (*Optional[optuna.trial._frozen.FrozenTrial]*) –
- **user_attrs** (*Dict[str, Any]*) –
- **system_attrs** (*Dict[str, Any]*) –
- **n_trials** (*int*) –
- **datetime_start** (*Optional[datetime.datetime]*) –
- **study_id** (*int*) –
- **directions** (*Optional[Sequence[optuna._study_direction.StudyDirection]]*) –

6.3.13 optuna.study

study 模块实现了*Study* 对象和相关函数。*Study* 类有一个公共构造函数，但不建议直接使用这个构造函数。相反，要创建或加载一个*Study* 时, 库用户应该分别使用*create_study* 或*load_study()* 。

optuna.study.Study	一个 study 对应于一个优化任务，即一组 trials.
optuna.study.create_study	创建新的*Study*.
optuna.study.load_study	加载一个所给名字的已经存在的*Study*.
optuna.study.delete_study	删除一个*Study* 对象。
optuna.study.get_all_study_summaries	返回指定存储中所有 study 的历史记录。
optuna.study.StudyDirection	*Study* 的方向。
optuna.study.StudySummary	一个*Study* 的基本属性和总汇结果。

optuna.study.Study

class optuna.study.**Study**(*study_name*, *storage*, *sampler=None*, *pruner=None*)

A study corresponds to an optimization task, i.e., a set of trials.

This object provides interfaces to run a new *Trial*, access trials' history, set/get user-defined attributes of the study itself.

Note that the direct use of this constructor is not recommended. To create and load a study, please refer to the documentation of *create_study()* and *load_study()* respectively.

Methods

add_trial(trial)	Add trial to study.
add_trials(trials)	Add trials to study.
ask([fixed_distributions])	Create a new trial from which hyperparameters can be suggested.
enqueue_trial(params)	Enqueue a trial with given parameter values.
get_trials([deepcopy, states])	Return all trials in the study.
optimize(func[, n_trials, timeout, n_jobs, ...])	Optimize an objective function.
set_system_attr(key, value)	Set a system attribute to the study.
set_user_attr(key, value)	Set a user attribute to the study.
stop()	Exit from the current optimization loop after the running trials finish.
tell(trial[, values, state])	Finish a trial created with *ask()*.
trials_dataframe([attrs, multi_index])	Export trials as a pandas DataFrame.

Attributes

best_params	Return parameters of the best trial in the study.
best_trial	Return the best trial in the study.
best_trials	Return trials located at the Pareto front in the study.
best_value	Return the best objective value in the study.
direction	Return the direction of the study.
directions	Return the directions of the study.
system_attrs	Return system attributes.
trials	Return all trials in the study.
user_attrs	Return user attributes.

参数

- **study_name** (*str*) –
- **storage** (*Union[str, optuna.storages._base.BaseStorage]*) –
- **sampler** (*Optional[samplers.BaseSampler]*) –
- **pruner** (*Optional[optuna.pruners._base.BasePruner]*) –

返回类型 None

add_trial(*trial*)

Add trial to study.

The trial is validated before being added.

示例

```
import optuna
from optuna.distributions import UniformDistribution


def objective(trial):
    x = trial.suggest_float("x", 0, 10)
    return x ** 2


study = optuna.create_study()
assert len(study.trials) == 0

trial = optuna.trial.create_trial(
    params={"x": 2.0},
    distributions={"x": UniformDistribution(0, 10)},
    value=4.0,
)

study.add_trial(trial)
assert len(study.trials) == 1

study.optimize(objective, n_trials=3)
assert len(study.trials) == 4

other_study = optuna.create_study()

for trial in study.trials:
    other_study.add_trial(trial)
assert len(other_study.trials) == len(study.trials)
```

(下页继续)

(续上页)

```
other_study.optimize(objective, n_trials=2)
assert len(other_study.trials) == len(study.trials) + 2
```

参见:

This method should in general be used to add already evaluated trials (`trial.state.is_finished() == True`). To queue trials for evaluation, please refer to *enqueue_trial()*.

参见:

See *create_trial()* for how to create trials.

参数 **trial** (optuna.trial._frozen.FrozenTrial) –Trial to add.

引发 **ValueError** –If trial is an invalid state.

返回类型 None

备注: Added in v2.0.0 as an experimental feature. The interface may change in newer versions without prior notice. See https://github.com/optuna/optuna/releases/tag/v2.0.0.

---

add_trials(*trials*)

Add trials to study.

The trials are validated before being added.

示例

```
import optuna
from optuna.distributions import UniformDistribution


def objective(trial):
    x = trial.suggest_float("x", 0, 10)
    return x ** 2


study = optuna.create_study()
study.optimize(objective, n_trials=3)
assert len(study.trials) == 3

other_study = optuna.create_study()
```

(下页继续)

(续上页)

```
other_study.add_trials(study.trials)
assert len(other_study.trials) == len(study.trials)

other_study.optimize(objective, n_trials=2)
assert len(other_study.trials) == len(study.trials) + 2
```

参见:

See *add_trial()* for addition of each trial.

参数 **trials** (*Iterable[optuna.trial._frozen.FrozenTrial]*) – Trials to add.

引发 **ValueError** – If `trials` include invalid trial.

返回类型 None

备注: Added in v2.5.0 as an experimental feature. The interface may change in newer versions without prior notice. See https://github.com/optuna/optuna/releases/tag/v2.5.0.

---

ask (*fixed_distributions=None*)

Create a new trial from which hyperparameters can be suggested.

This method is part of an alternative to *optimize()* that allows controlling the lifetime of a trial outside the scope of `func`. Each call to this method should be followed by a call to *tell()* to finish the created trial.

参见:

The *Ask-and-Tell 接口* tutorial provides use-cases with examples.

示例

Getting the trial object with the *ask()* method.

```
import optuna


study = optuna.create_study()

trial = study.ask()

x = trial.suggest_float("x", -1, 1)

study.tell(trial, x ** 2)
```

示例

Passing previously defined distributions to the *ask()* method.

```
import optuna


study = optuna.create_study()

distributions = {
    "optimizer": optuna.distributions.CategoricalDistribution(["adam", "sgd
↪"]),
    "lr": optuna.distributions.LogUniformDistribution(0.0001, 0.1),
}

# You can pass the distributions previously defined.
trial = study.ask(fixed_distributions=distributions)

# `optimizer` and `lr` are already suggested and accessible with `trial.
↪params`.
assert "optimizer" in trial.params
assert "lr" in trial.params
```

参数 **fixed_distributions** (*Optional[Dict[str, optuna.distributions.BaseDistribution]]*) – A dictionary containing the parameter names and parameter's distributions. Each parameter in this dictionary is automatically suggested for the returned trial, even when the suggest method is not explicitly invoked by the user. If this argument is set to `None`, no parameter is automatically suggested.

返回 A *Trial*.

返回类型 *optuna.trial._trial.Trial*

property best_params: Dict[str, Any]

Return parameters of the best trial in the study.

返回 A dictionary containing parameters of the best trial.

引发 **RuntimeError** – If the study has more than one direction.

property best_trial: *optuna.trial._frozen.FrozenTrial*

Return the best trial in the study.

返回 A `FrozenTrial` object of the best trial.

引发 **RuntimeError** – If the study has more than one direction.

property best_trials: List[*optuna.trial._frozen.FrozenTrial*]

Return trials located at the Pareto front in the study.

A trial is located at the Pareto front if there are no trials that dominate the trial. It's called that a trial `t0` dominates another trial `t1` if `all(v0 <= v1) for v0, v1 in zip(t0.values, t1.values)` and `any(v0 < v1) for v0, v1 in zip(t0.values, t1.values)` are held.

返回 A list of *FrozenTrial* objects.

property best_value: float

Return the best objective value in the study.

返回 A float representing the best objective value.

引发 **RuntimeError** –If the study has more than one direction.

property direction: *optuna._study_direction.StudyDirection*

Return the direction of the study.

返回 A *StudyDirection* object.

引发 **RuntimeError** –If the study has more than one direction.

property directions: List[*optuna._study_direction.StudyDirection*]

Return the directions of the study.

返回 A list of *StudyDirection* objects.

enqueue_trial(*params*)

Enqueue a trial with given parameter values.

You can fix the next sampling parameters which will be evaluated in your objective function.

示例

```
import optuna


def objective(trial):
    x = trial.suggest_float("x", 0, 10)
    return x ** 2


study = optuna.create_study()
study.enqueue_trial({"x": 5})
study.enqueue_trial({"x": 0})
study.optimize(objective, n_trials=2)
```

(下页继续)

(续上页)

```
assert study.trials[0].params == {"x": 5}
assert study.trials[1].params == {"x": 0}
```

参数 **params** (*Dict[str, Any]*) –Parameter values to pass your objective function.

返回类型 None

备注： Added in v1.2.0 as an experimental feature. The interface may change in newer versions without prior notice. See https://github.com/optuna/optuna/releases/tag/v1.2.0.

---

get_trials(*deepcopy=True*, *states=None*)

Return all trials in the study.

The returned trials are ordered by trial number.

示例

```
import optuna


def objective(trial):
    x = trial.suggest_float("x", -1, 1)
    return x ** 2


study = optuna.create_study()
study.optimize(objective, n_trials=3)

trials = study.get_trials()
assert len(trials) == 3
```

参数

- **deepcopy** (*bool*) –Flag to control whether to apply `copy.deepcopy()` to the trials. Note that if you set the flag to `False`, you shouldn't mutate any fields of the returned trial. Otherwise the internal state of the study may corrupt and unexpected behavior may happen.
- **states** (*Optional[Tuple[optuna.trial._state.TrialState, ...]]*) –Trial states to filter on. If `None`, include all states.

返回 A list of `FrozenTrial` objects.

返回类型 *List*[*optuna.trial._frozen.FrozenTrial*]

optimize(*func*, *n_trials=None*, *timeout=None*, *n_jobs=1*, *catch=()*, *callbacks=None*, *gc_after_trial=False*, *show_progress_bar=False*)

Optimize an objective function.

Optimization is done by choosing a suitable set of hyperparameter values from a given range. Uses a sampler which implements the task of value suggestion based on a specified distribution. The sampler is specified in *create_study()* and the default choice for the sampler is TPE. See also *TPESampler* for more details on 'TPE'.

示例

```
import optuna


def objective(trial):
    x = trial.suggest_float("x", -1, 1)
    return x ** 2


study = optuna.create_study()
study.optimize(objective, n_trials=3)
```

参数

- **func** (*Callable[[optuna.trial._trial.Trial], Union[float, Sequence[float]]]*) –A callable that implements objective function.
- **n_trials** (*Optional[int]*) –The number of trials. If this argument is set to None, there is no limitation on the number of trials. If timeout is also set to None, the study continues to create trials until it receives a termination signal such as Ctrl+C or SIGTERM.
- **timeout** (*Optional[float]*) –Stop study after the given number of second(s). If this argument is set to None, the study is executed without time limitation. If n_trials is also set to None, the study continues to create trials until it receives a termination signal such as Ctrl+C or SIGTERM.
- **n_jobs** (*int*) –The number of parallel jobs. If this argument is set to -1, the number is set to CPU count.

备注: n_jobs allows parallelization using threading and may suffer from Python's GIL. It is recommended to use *process-based parallelization* if func is CPU bound.

> **警告:** Deprecated in v2.7.0. This feature will be removed in the future. It is recommended to use *process-based parallelization*. The removal of this feature is currently scheduled for v4.0.0, but this schedule is subject to change. See https://github.com/optuna/optuna/releases/tag/v2.7.0.

- **catch** (*Tuple[Type[Exception], ...]*) –A study continues to run even when a trial raises one of the exceptions specified in this argument. Default is an empty tuple, i.e. the study will stop for any exception except for *TrialPruned*.
- **callbacks** (*Optional[List[Callable[[optuna.study.Study, optuna.trial._frozen.FrozenTrial], None]]]*) –List of callback functions that are invoked at the end of each trial. Each function must accept two parameters with the following types in this order: *Study* and `FrozenTrial`.
- **gc_after_trial** (*bool*) –Flag to determine whether to automatically run garbage collection after each trial. Set to `True` to run the garbage collection, `False` otherwise. When it runs, it runs a full collection by internally calling `gc.collect()`. If you see an increase in memory consumption over several trials, try setting this flag to `True`.

  参见:

  在优化时, 我该如何避免耗尽内存 (running out of memory, OOM)?
- **show_progress_bar** (*bool*) –Flag to show progress bars or not. To disable progress bar, set this `False`. Currently, progress bar is experimental feature and disabled when `n_jobs` $\neq 1$.

引发 **RuntimeError** –If nested invocation of this method occurs.

返回类型 None

set_system_attr(*key*, *value*)

Set a system attribute to the study.

Note that Optuna internally uses this method to save system messages. Please use *set_user_attr()* to set users' attributes.

参数

- **key** (*str*) –A key string of the attribute.
- **value** (*Any*) –A value of the attribute. The value should be JSON serializable.

返回类型 None

set_user_attr(*key*, *value*)

Set a user attribute to the study.

参见:

See *user_attrs* for related attribute.

示例

```
import optuna


def objective(trial):
    x = trial.suggest_float("x", 0, 1)
    y = trial.suggest_float("y", 0, 1)
    return x ** 2 + y ** 2


study = optuna.create_study()

study.set_user_attr("objective function", "quadratic function")
study.set_user_attr("dimensions", 2)
study.set_user_attr("contributors", ["Akiba", "Sano"])

assert study.user_attrs == {
    "objective function": "quadratic function",
    "dimensions": 2,
    "contributors": ["Akiba", "Sano"],
}
```

参数

- **key** (*str*) –A key string of the attribute.
- **value** (*Any*) –A value of the attribute. The value should be JSON serializable.

返回类型 None

stop()

Exit from the current optimization loop after the running trials finish.

This method lets the running *optimize()* method return immediately after all trials which the *optimize()* method spawned finishes. This method does not affect any behaviors of parallel or successive study processes.

示例

```
import optuna


def objective(trial):
    if trial.number == 4:
        trial.study.stop()
    x = trial.suggest_float("x", 0, 10)
    return x ** 2


study = optuna.create_study()
study.optimize(objective, n_trials=10)
assert len(study.trials) == 5
```

引发 **RuntimeError** –If this method is called outside an objective function or callback.

返回类型 None

property system_attrs: Dict[str, Any]

Return system attributes.

返回 A dictionary containing all system attributes.

tell(*trial*, *values=None*, *state=TrialState.COMPLETE*)

Finish a trial created with *ask()*.

参见:

The *Ask-and-Tell 接口* tutorial provides use-cases with examples.

示例

```
import optuna
from optuna.trial import TrialState


def f(x):
    return (x - 2) ** 2


def df(x):
    return 2 * x - 4
```

(下页继续)

(续上页)

```
study = optuna.create_study()

n_trials = 30

for _ in range(n_trials):
    trial = study.ask()

    lr = trial.suggest_float("lr", 1e-5, 1e-1, log=True)

    # Iterative gradient descent objective function.
    x = 3  # Initial value.
    for step in range(128):
        y = f(x)

        trial.report(y, step=step)

        if trial.should_prune():
            # Finish the trial with the pruned state.
            study.tell(trial, state=TrialState.PRUNED)
            break

        gy = df(x)
        x -= gy * lr
    else:
        # Finish the trial with the final value after all iterations.
        study.tell(trial, y)
```

参数

- **trial** (*Union[optuna.trial._trial.Trial, int]*) –A *Trial* object or a trial number.
- **values** (*Optional[Union[float, Sequence[float]]]*) –Optional objective value or a sequence of such values in case the study is used for multi-objective optimization. Argument must be provided if `state` is *COMPLETE* and should be `None` if `state` is *FAIL* or *PRUNED*.
- **state** (`optuna.trial._state.TrialState`) –State to be reported. Must be *COMPLETE*, *FAIL* or *PRUNED*.

引发

- **TypeError** –If `trial` is not a *Trial* or an `int`.
- **ValueError** –If any of the following. `values` is a sequence but its length does not match

the number of objectives for its associated study. `state` is *COMPLETE* but `values` is `None`. `state` is *FAIL* or *PRUNED* but `values` is not `None`. `state` is not *COMPLETE*, *FAIL* or *PRUNED*. `trial` is a trial number but no trial exists with that number.

返回类型 None

property trials: List[*optuna.trial._frozen.FrozenTrial*]

Return all trials in the study.

The returned trials are ordered by trial number.

This is a short form of `self.get_trials(deepcopy=True, states=None)`.

返回 A list of `FrozenTrial` objects.

trials_dataframe(*attrs=('number', 'value', 'datetime_start', 'datetime_complete', 'duration', 'params', 'user_attrs', 'system_attrs', 'state')*, *multi_index=False*)

Export trials as a pandas DataFrame.

The DataFrame provides various features to analyze studies. It is also useful to draw a histogram of objective values and to export trials as a CSV file. If there are no trials, an empty DataFrame is returned.

示例

```
import optuna
import pandas


def objective(trial):
    x = trial.suggest_float("x", -1, 1)
    return x ** 2


study = optuna.create_study()
study.optimize(objective, n_trials=3)

# Create a dataframe from the study.
df = study.trials_dataframe()
assert isinstance(df, pandas.DataFrame)
assert df.shape[0] == 3  # n_trials.
```

参数

- **attrs** (*Tuple[str, ...]*) –Specifies field names of `FrozenTrial` to include them to a DataFrame of trials.

- **multi_index** (*bool*) –Specifies whether the returned DataFrame employs MultiIndex or not. Columns that are hierarchical by nature such as `(params, x)` will be flattened to `params_x` when set to `False`.

返回 A pandas DataFrame of trials in the *Study*.

返回类型 pandas.core.frame.DataFrame

备注： If `value` is in `attrs` during multi-objective optimization, it is implicitly replaced with `values`.

---

property user_attrs: Dict[str, Any]

Return user attributes.

参见：

See *set_user_attr()* for related method.

示例

```
import optuna


def objective(trial):
    x = trial.suggest_float("x", 0, 1)
    y = trial.suggest_float("y", 0, 1)
    return x ** 2 + y ** 2


study = optuna.create_study()

study.set_user_attr("objective function", "quadratic function")
study.set_user_attr("dimensions", 2)
study.set_user_attr("contributors", ["Akiba", "Sano"])

assert study.user_attrs == {
    "objective function": "quadratic function",
    "dimensions": 2,
    "contributors": ["Akiba", "Sano"],
}
```

返回 A dictionary containing all user attributes.

optuna.study.create_study

optuna.study.**create_study**(*storage=None*, *sampler=None*, *pruner=None*, *study_name=None*, *direction=None*, *load_if_exists=False*, ***, *directions=None*)

Create a new *Study*.

示例

```
import optuna


def objective(trial):
    x = trial.suggest_float("x", 0, 10)
    return x ** 2


study = optuna.create_study()
study.optimize(objective, n_trials=3)
```

参数

- **storage** (*Optional[Union[str, optuna.storages._base.BaseStorage]]*) –Database URL. If this argument is set to None, in-memory storage is used, and the *Study* will not be persistent.

  备注:

  When a database URL is passed, Optuna internally uses SQLAlchemy to handle the database. Please refer to SQLAlchemy's document for further details. If you want to specify non-default options to SQLAlchemy Engine, you can instantiate *RDBStorage* with your desired options and pass it to the `storage` argument instead of a URL.

- **sampler** (*Optional[optuna.samplers._base.BaseSampler]*) –A sampler object that implements background algorithm for value suggestion. If `None` is specified, *TPESampler* is used during single-objective optimization and *NSGAIISampler* during multi-objective optimization. See also *samplers*.
- **pruner** (*Optional[optuna.pruners._base.BasePruner]*) –A pruner object that decides early stopping of unpromising trials. If `None` is specified, *MedianPruner* is used as the default. See also *pruners*.
- **study_name** (*Optional[str]*) –Study's name. If this argument is set to None, a unique name is generated automatically.

- **direction** (*Optional[Union[str, optuna._study_direction.StudyDirection]]*) –Direction of optimization. Set `minimize` for minimization and `maximize` for maximization. You can also pass the corresponding *StudyDirection* object.

  备注： If none of *direction* and *directions* are specified, the direction of the study is set to "minimize" .

- **load_if_exists** (*bool*) –Flag to control the behavior to handle a conflict of study names. In the case where a study named `study_name` already exists in the `storage`, a *DuplicatedStudyError* is raised if `load_if_exists` is set to `False`. Otherwise, the creation of the study is skipped, and the existing one is returned.
- **directions** (*Optional[Sequence[Union[str, optuna._study_direction.StudyDirection]]]*) –A sequence of directions during multi-objective optimization.

返回 A *Study* object.

引发 **ValueError** –If the length of `directions` is zero. Or, if `direction` is neither 'minimize' nor 'maximize' when it is a string. Or, if the element of `directions` is neither *minimize* nor *maximize*. Or, if both `direction` and `directions` are specified.

返回类型 *optuna.study.Study*

参见:

optuna.create_study() is an alias of *optuna.study.create_study()*.

optuna.study.load_study

optuna.study.**load_study**(*study_name*, *storage*, *sampler=None*, *pruner=None*)

Load the existing *Study* that has the specified name.

示例

```
import optuna


def objective(trial):
    x = trial.suggest_float("x", 0, 10)
    return x ** 2
```

(下页继续)

(续上页)

```
study = optuna.create_study(storage="sqlite:///example.db", study_name="my_study")
study.optimize(objective, n_trials=3)

loaded_study = optuna.load_study(study_name="my_study", storage="sqlite:///
↪example.db")
assert len(loaded_study.trials) == len(study.trials)
```

参数

- **study_name** (*str*) –Study’ s name. Each study has a unique name as an identifier.
- **storage** (*Union[str, optuna.storages._base.BaseStorage]*) –Database URL such as `sqlite:///example.db`. Please see also the documentation of *create_study()* for further details.
- **sampler** (*Optional[optuna.samplers._base.BaseSampler]*) –A sampler object that implements background algorithm for value suggestion. If `None` is specified, *TPESampler* is used as the default. See also *samplers*.
- **pruner** (*Optional[optuna.pruners._base.BasePruner]*) –A pruner object that decides early stopping of unpromising trials. If `None` is specified, *MedianPruner* is used as the default. See also *pruners*.

返回类型 *optuna.study.Study*

参见:

optuna.load_study() is an alias of *optuna.study.load_study()*.

optuna.study.delete_study

optuna.study.**delete_study**(*study_name*, *storage*)

Delete a *Study* object.

示例

```
import optuna


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

(下页继续)

(续上页)

```
study = optuna.create_study(study_name="example-study", storage="sqlite:///
↪example.db")
study.optimize(objective, n_trials=3)

optuna.delete_study(study_name="example-study", storage="sqlite:///example.db")
```

参数

- **study_name** (*str*) –Study’ s name.
- **storage** (*Union[str, optuna.storages._base.BaseStorage]*) –Database URL such as `sqlite:///example.db`. Please see also the documentation of *create_study()* for further details.

返回类型 None

参见:

optuna.delete_study() is an alias of *optuna.study.delete_study()*.

optuna.study.get_all_study_summaries

optuna.study.**get_all_study_summaries**(*storage*)

Get all history of studies stored in a specified storage.

示例

```
import optuna


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


study = optuna.create_study(study_name="example-study", storage="sqlite:///
↪example.db")
study.optimize(objective, n_trials=3)

study_summaries = optuna.study.get_all_study_summaries(storage="sqlite:///example.
↪db")
assert len(study_summaries) == 1
```

(下页继续)

(续上页)

```
study_summary = study_summaries[0]
assert study_summary.study_name == "example-study"
```

参数 **storage** (*Union[str, optuna.storages._base.BaseStorage]*) – Database URL such as `sqlite:///example.db`. Please see also the documentation of *create_study()* for further details.

返回 List of study history summarized as *StudySummary* objects.

返回类型 *List*[*optuna._study_summary.StudySummary*]

参见:

optuna.get_all_study_summaries() is an alias of *optuna.study.get_all_study_summaries()*.

optuna.study.StudyDirection

class optuna.study.**StudyDirection**(*value*)

Direction of a *Study*.

NOT_SET

Direction has not been set.

MINIMIZE

Study minimizes the objective function.

MAXIMIZE

Study maximizes the objective function.

Attributes

NOT_SET
MINIMIZE
MAXIMIZE

optuna.study.StudySummary

class optuna.study.**StudySummary**(*study_name*, *direction*, *best_trial*, *user_attrs*, *system_attrs*, *n_trials*, *datetime_start*, *study_id*, ***, *directions=None*)

Basic attributes and aggregated results of a *Study*.

See also *optuna.study.get_all_study_summaries()*.

参数

- **study_name** (*str*) –
- **direction** (*Optional[optuna._study_direction.StudyDirection]*) –
- **best_trial** (*Optional[optuna.trial._frozen.FrozenTrial]*) –
- **user_attrs** (*Dict[str, Any]*) –
- **system_attrs** (*Dict[str, Any]*) –
- **n_trials** (*int*) –
- **datetime_start** (*Optional[datetime.datetime]*) –
- **study_id** (*int*) –
- **directions** (*Optional[Sequence[optuna._study_direction.StudyDirection]]*) –

study_name

Name of the *Study*.

direction

StudyDirection of the *Study*.

备注： This attribute is only available during single-objective optimization.

---

directions

A sequence of *StudyDirection* objects.

best_trial

`FrozenTrial` with best objective value in the *Study*.

user_attrs

Dictionary that contains the attributes of the *Study* set with *optuna.study.Study.set_user_attr()*.

system_attrs

Dictionary that contains the attributes of the *Study* internally set by Optuna.

n_trials

The number of trials ran in the *Study*.

datetime_start

Datetime where the *Study* started.

Attributes

direction
directions

6.3.14 optuna.trial

trial 模块包含与*Trial* 相关的类和功能。

一个*Trial* 实例代表了一个评估目标函数的过程。该实例被传递给目标函数来提供获取参数建议、管理 trial 状态、设置/获取用户定义的 trial 属性等接口，Optuna 用户可以通过接口定义自定义目标函数。基本上，Optuna 用户只在自己的自定义目标函数中使用它。

optuna.trial.Trial	一个 trial 是一个评估目标函数的过程。
optuna.trial.FixedTrial	一个始终为每个参数提供固定 suggestion 值的 trial 类。
optuna.trial.FrozenTrial	一个*Trial* 的状态和结果。
optuna.trial.TrialState	*Trial* 的状态。
optuna.trial.create_trial	创建一个新的*FrozenTrial* 。

optuna.trial.Trial

class optuna.trial.**Trial**(*study*, *trial_id*)

A trial is a process of evaluating an objective function.

This object is passed to an objective function and provides interfaces to get parameter suggestion, manage the trial's state, and set/get user-defined attributes of the trial.

Note that the direct use of this constructor is not recommended. This object is seamlessly instantiated and passed to the objective function behind the *optuna.study.Study.optimize()* method; hence library users do not care about instantiation of this object.

参数

- **study** (optuna.study.Study) –A *Study* object.
- **trial_id** (*int*) –A trial ID that is automatically generated.

返回类型 None

Methods

report(value, step)	Report an objective function value for a given step.
set_system_attr(key, value)	Set system attributes to the trial.
set_user_attr(key, value)	Set user attributes to the trial.
should_prune()	Suggest whether the trial should be pruned or not.
suggest_categorical(name, choices)	Suggest a value for the categorical parameter.
suggest_discrete_uniform(name, low, high, q)	Suggest a value for the discrete parameter.
suggest_float(name, low, high, *[, step, log])	Suggest a value for the floating point parameter.
suggest_int(name, low, high[, step, log])	Suggest a value for the integer parameter.
suggest_loguniform(name, low, high)	Suggest a value for the continuous parameter.
suggest_uniform(name, low, high)	Suggest a value for the continuous parameter.

Attributes

datetime_start	Return start datetime.
distributions	Return distributions of parameters to be optimized.
number	Return trial's number which is consecutive and unique in a study.
params	Return parameters to be optimized.
system_attrs	Return system attributes.
user_attrs	Return user attributes.

property datetime_start: Optional[datetime.datetime]

Return start datetime.

返回 Datetime where the *Trial* started.

property distributions: Dict[str, optuna.distributions.BaseDistribution]

Return distributions of parameters to be optimized.

返回 A dictionary containing all distributions.

property number: int

Return trial's number which is consecutive and unique in a study.

返回 A trial number.

property params: Dict[str, Any]

Return parameters to be optimized.

返回 A dictionary containing all parameters.

report(*value*, *step*)

Report an objective function value for a given step.

The reported values are used by the pruners to determine whether this trial should be pruned.

参见:

Please refer to *BasePruner*.

备注: The reported value is converted to `float` type by applying `float()` function internally. Thus, it accepts all float-like types (e.g., `numpy.float32`). If the conversion fails, a `TypeError` is raised.

示例

Report intermediate scores of SGDClassifier training.

```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import train_test_split

import optuna

X, y = load_iris(return_X_y=True)
X_train, X_valid, y_train, y_valid = train_test_split(X, y)


def objective(trial):
    clf = SGDClassifier(random_state=0)
    for step in range(100):
        clf.partial_fit(X_train, y_train, np.unique(y))
        intermediate_value = clf.score(X_valid, y_valid)
        trial.report(intermediate_value, step=step)
        if trial.should_prune():
            raise optuna.TrialPruned()

    return clf.score(X_valid, y_valid)
```

(下页继续)

(续上页)

```
study = optuna.create_study(direction="maximize")
study.optimize(objective, n_trials=3)
```

参数

- **value** (*float*) –A value returned from the objective function.
- **step** (*int*) –Step of the trial (e.g., Epoch of neural network training). Note that pruners assume that `step` starts at zero. For example, *MedianPruner* simply checks if `step` is less than `n_warmup_steps` as the warmup mechanism.

引发 **NotImplementedError** –If trial is being used for multi-objective optimization.

返回类型 None

set_system_attr(*key*, *value*)

Set system attributes to the trial.

Note that Optuna internally uses this method to save system messages such as failure reason of trials. Please use *set_user_attr()* to set users' attributes.

参数

- **key** (*str*) –A key string of the attribute.
- **value** (*Any*) –A value of the attribute. The value should be JSON serializable.

返回类型 None

set_user_attr(*key*, *value*)

Set user attributes to the trial.

The user attributes in the trial can be access via *optuna.trial.Trial.user_attrs()*.

示例

Save fixed hyperparameters of neural network training.

```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier

import optuna

X, y = load_iris(return_X_y=True)
```

(下页继续)

(续上页)

```
X_train, X_valid, y_train, y_valid = train_test_split(X, y, random_state=0)


def objective(trial):
    trial.set_user_attr("BATCHSIZE", 128)
    momentum = trial.suggest_uniform("momentum", 0, 1.0)
    clf = MLPClassifier(
        hidden_layer_sizes=(100, 50),
        batch_size=trial.user_attrs["BATCHSIZE"],
        momentum=momentum,
        solver="sgd",
        random_state=0,
    )
    clf.fit(X_train, y_train)

    return clf.score(X_valid, y_valid)


study = optuna.create_study(direction="maximize")
study.optimize(objective, n_trials=3)
assert "BATCHSIZE" in study.best_trial.user_attrs.keys()
assert study.best_trial.user_attrs["BATCHSIZE"] == 128
```

参数

- **key** (*str*) –A key string of the attribute.
- **value** (*Any*) –A value of the attribute. The value should be JSON serializable.

返回类型 None

should_prune()

Suggest whether the trial should be pruned or not.

The suggestion is made by a pruning algorithm associated with the trial and is based on previously reported values. The algorithm can be specified when constructing a *Study*.

备注： If no values have been reported, the algorithm cannot make meaningful suggestions. Similarly, if this method is called multiple times with the exact same set of reported values, the suggestions will be the same.

参见:

Please refer to the example code in *optuna.trial.Trial.report()*.

返回 A boolean value. If `True`, the trial should be pruned according to the configured pruning algorithm. Otherwise, the trial should continue.

引发 **NotImplementedError** –If trial is being used for multi-objective optimization.

返回类型 bool

suggest_categorical(*name*, *choices*)

Suggest a value for the categorical parameter.

The value is sampled from `choices`.

示例

Suggest a kernel function of SVC.

```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC

import optuna

X, y = load_iris(return_X_y=True)
X_train, X_valid, y_train, y_valid = train_test_split(X, y)


def objective(trial):
    kernel = trial.suggest_categorical("kernel", ["linear", "poly", "rbf"])
    clf = SVC(kernel=kernel, gamma="scale", random_state=0)
    clf.fit(X_train, y_train)
    return clf.score(X_valid, y_valid)


study = optuna.create_study(direction="maximize")
study.optimize(objective, n_trials=3)
```

参数

- **name** (*str*) –A parameter name.
- **choices** (*Sequence[Union[None, bool, int, float, str]]*) –Parameter value candidates.

返回类型 *Union*[None, bool, int, float, str]

参见:

CategoricalDistribution.

返回 A suggested value.

参数

- **name** (*str*) –
- **choices** (*Sequence[Union[None, bool, int, float, str]]*) –

返回类型 *Union*[None, bool, int, float, str]

suggest_discrete_uniform(*name*, *low*, *high*, *q*)

Suggest a value for the discrete parameter.

The value is sampled from the range $[\mathsf{low}, \mathsf{high}]$, and the step of discretization is q. More specifically, this method returns one of the values in the sequence $\mathsf{low}, \mathsf{low} + q, \mathsf{low} + 2q, \ldots, \mathsf{low} + kq \leq \mathsf{high}$, where k denotes an integer. Note that $high$ may be changed due to round-off errors if q is not an integer. Please check warning messages to find the changed values.

示例

Suggest a fraction of samples used for fitting the individual learners of GradientBoostingClassifier.

```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import train_test_split

import optuna

X, y = load_iris(return_X_y=True)
X_train, X_valid, y_train, y_valid = train_test_split(X, y)


def objective(trial):
    subsample = trial.suggest_discrete_uniform("subsample", 0.1, 1.0, 0.1)
    clf = GradientBoostingClassifier(subsample=subsample, random_state=0)
    clf.fit(X_train, y_train)
    return clf.score(X_valid, y_valid)


study = optuna.create_study(direction="maximize")
study.optimize(objective, n_trials=3)
```

参数

- **name** (*str*) –A parameter name.
- **low** (*float*) –Lower endpoint of the range of suggested values. `low` is included in the range.
- **high** (*float*) –Upper endpoint of the range of suggested values. `high` is included in the range.
- **q** (*float*) –A step of discretization.

返回 A suggested float value.

返回类型 float

suggest_float(*name*, *low*, *high*, *, *step=None*, *log=False*)

Suggest a value for the floating point parameter.

Note that this is a wrapper method for *suggest_uniform()*, *suggest_loguniform()* and *suggest_discrete_uniform()*.

1.3.0 新版功能.

参见:

Please see also *suggest_uniform()*, *suggest_loguniform()* and *suggest_discrete_uniform()*.

示例

Suggest a momentum, learning rate and scaling factor of learning rate for neural network training.

```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier

import optuna

X, y = load_iris(return_X_y=True)
X_train, X_valid, y_train, y_valid = train_test_split(X, y, random_state=0)


def objective(trial):
    momentum = trial.suggest_float("momentum", 0.0, 1.0)
    learning_rate_init = trial.suggest_float(
        "learning_rate_init", 1e-5, 1e-3, log=True
    )
```

(下页继续)

(续上页)

```
    power_t = trial.suggest_float("power_t", 0.2, 0.8, step=0.1)
    clf = MLPClassifier(
        hidden_layer_sizes=(100, 50),
        momentum=momentum,
        learning_rate_init=learning_rate_init,
        solver="sgd",
        random_state=0,
        power_t=power_t,
    )
    clf.fit(X_train, y_train)

    return clf.score(X_valid, y_valid)


study = optuna.create_study(direction="maximize")
study.optimize(objective, n_trials=3)
```

参数

- **name** (*str*) –A parameter name.
- **low** (*float*) –Lower endpoint of the range of suggested values. `low` is included in the range.
- **high** (*float*) –Upper endpoint of the range of suggested values. `high` is excluded from the range.

备注： If `step` is specified, `high` is included as well as `low` because this method falls back to *suggest_discrete_uniform()*.

- **step** (*Optional[float]*) –A step of discretization.

备注： The `step` and `log` arguments cannot be used at the same time. To set the `step` argument to a float number, set the `log` argument to `False`.

- **log** (*bool*) –A flag to sample the value from the log domain or not. If `log` is true, the value is sampled from the range in the log domain. Otherwise, the value is sampled from the range in the linear domain. See also *suggest_uniform()* and *suggest_loguniform()*.

备注： The `step` and `log` arguments cannot be used at the same time. To set the `log`

argument to `True`, set the `step` argument to `None`.

引发 **ValueError** –If `step is not None` and `log = True` are specified.

返回 A suggested float value.

返回类型 float

suggest_int(*name*, *low*, *high*, *step=1*, *log=False*)

Suggest a value for the integer parameter.

The value is sampled from the integers in [low, high].

示例

Suggest the number of trees in RandomForestClassifier.

```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split

import optuna

X, y = load_iris(return_X_y=True)
X_train, X_valid, y_train, y_valid = train_test_split(X, y)


def objective(trial):
    n_estimators = trial.suggest_int("n_estimators", 50, 400)
    clf = RandomForestClassifier(n_estimators=n_estimators, random_state=0)
    clf.fit(X_train, y_train)
    return clf.score(X_valid, y_valid)


study = optuna.create_study(direction="maximize")
study.optimize(objective, n_trials=3)
```

参数

- **name** (*str*) –A parameter name.
- **low** (*int*) –Lower endpoint of the range of suggested values. `low` is included in the range.
- **high** (*int*) –Upper endpoint of the range of suggested values. `high` is included in the range.

- **step** (*int*) –A step of discretization.

  备注： Note that high is modified if the range is not divisible by step. Please check the warning messages to find the changed values.

  备注： The method returns one of the values in the sequence $\mathsf{low}, \mathsf{low} + \mathsf{step}, \mathsf{low} + 2 * \mathsf{step}, \ldots, \mathsf{low} + k * \mathsf{step} \leq \mathsf{high}$, where k denotes an integer.

  备注： The `step != 1` and `log` arguments cannot be used at the same time. To set the `step` argument $\mathsf{step} \geq 2$, set the `log` argument to `False`.

- **log** (*bool*) –A flag to sample the value from the log domain or not.

  备注： If `log` is true, at first, the range of suggested values is divided into grid points of width 1. The range of suggested values is then converted to a log domain, from which a value is sampled. The uniformly sampled value is re-converted to the original domain and rounded to the nearest grid point that we just split, and the suggested value is determined. For example, if *low = 2* and *high = 8*, then the range of suggested values is *[2, 3, 4, 5, 6, 7, 8]* and lower values tend to be more sampled than higher values.

  备注： The `step != 1` and `log` arguments cannot be used at the same time. To set the `log` argument to `True`, set the `step` argument to 1.

引发 **ValueError** –If `step != 1` and `log = True` are specified.

返回类型 int

suggest_loguniform(*name*, *low*, *high*)

Suggest a value for the continuous parameter.

The value is sampled from the range $[\mathsf{low}, \mathsf{high})$ in the log domain. When $\mathsf{low} = \mathsf{high}$, the value of low will be returned.

示例

Suggest penalty parameter `C` of SVC.

```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC

import optuna

X, y = load_iris(return_X_y=True)
X_train, X_valid, y_train, y_valid = train_test_split(X, y)


def objective(trial):
    c = trial.suggest_loguniform("c", 1e-5, 1e2)
    clf = SVC(C=c, gamma="scale", random_state=0)
    clf.fit(X_train, y_train)
    return clf.score(X_valid, y_valid)


study = optuna.create_study(direction="maximize")
study.optimize(objective, n_trials=3)
```

参数

- **name** (*str*) –A parameter name.
- **low** (*float*) –Lower endpoint of the range of suggested values. `low` is included in the range.
- **high** (*float*) –Upper endpoint of the range of suggested values. `high` is excluded from the range.

返回 A suggested float value.

返回类型 float

suggest_uniform(*name*, *low*, *high*)

Suggest a value for the continuous parameter.

The value is sampled from the range $[\mathsf{low}, \mathsf{high})$ in the linear domain. When $\mathsf{low} = \mathsf{high}$, the value of low will be returned.

示例

Suggest a momentum for neural network training.

```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier

import optuna

X, y = load_iris(return_X_y=True)
X_train, X_valid, y_train, y_valid = train_test_split(X, y)


def objective(trial):
    momentum = trial.suggest_uniform("momentum", 0.0, 1.0)
    clf = MLPClassifier(
        hidden_layer_sizes=(100, 50),
        momentum=momentum,
        solver="sgd",
        random_state=0,
    )
    clf.fit(X_train, y_train)

    return clf.score(X_valid, y_valid)


study = optuna.create_study(direction="maximize")
study.optimize(objective, n_trials=3)
```

参数

- **name** (*str*) –A parameter name.
- **low** (*float*) –Lower endpoint of the range of suggested values. `low` is included in the range.
- **high** (*float*) –Upper endpoint of the range of suggested values. `high` is excluded from the range.

返回 A suggested float value.

返回类型 float

property system_attrs: Dict[str, Any]

Return system attributes.

返回 A dictionary containing all system attributes.

property user_attrs: Dict[str, Any]

Return user attributes.

返回 A dictionary containing all user attributes.

optuna.trial.FixedTrial

class optuna.trial.**FixedTrial**(*params*, *number=0*)

A trial class which suggests a fixed value for each parameter.

This object has the same methods as *Trial*, and it suggests pre-defined parameter values. The parameter values can be determined at the construction of the *FixedTrial* object. In contrast to *Trial*, *FixedTrial* does not depend on *Study*, and it is useful for deploying optimization results.

示例

Evaluate an objective function with parameter values given by a user.

```
import optuna


def objective(trial):
    x = trial.suggest_uniform("x", -100, 100)
    y = trial.suggest_categorical("y", [-1, 0, 1])
    return x ** 2 + y


assert objective(optuna.trial.FixedTrial({"x": 1, "y": 0})) == 1
```

备注: Please refer to *Trial* for details of methods and properties.

参数

- **params** (*Dict[str, Any]*) –A dictionary containing all parameters.
- **number** (*int*) –A trial number. Defaults to 0.

返回类型 None

Methods

`report`(value, step)
`set_system_attr`(key, value)
`set_user_attr`(key, value)
`should_prune`()
`suggest_categorical`(name, choices)
`suggest_discrete_uniform`(name, low, high, q)
`suggest_float`(name, low, high, *[, step, log])
`suggest_int`(name, low, high[, step, log])
`suggest_loguniform`(name, low, high)
`suggest_uniform`(name, low, high)

Attributes

`datetime_start`
`distributions`
`number`
`params`
`system_attrs`
`user_attrs`

optuna.trial.FrozenTrial

class optuna.trial.**FrozenTrial**(*number*, *state*, *value*, *datetime_start*, *datetime_complete*, *params*, *distributions*, *user_attrs*, *system_attrs*, *intermediate_values*, *trial_id*, *, *values=None*)

Status and results of a *Trial*.

This object has the same methods as *Trial*, and it suggests best parameter values among performed trials. In contrast to *Trial*, *FrozenTrial* does not depend on *Study*, and it is useful for deploying optimization results.

示例

Re-evaluate an objective function with parameter values optimized study.

```
import optuna


def objective(trial):
    x = trial.suggest_uniform("x", -1, 1)
    return x ** 2


study = optuna.create_study()
study.optimize(objective, n_trials=3)

assert objective(study.best_trial) == study.best_value
```

备注： Attributes are set in optuna.Study.optimize(), but several attributes can be updated after the optimization. That means such attributes are overwritten by the re-evaluation if your objective updates attributes of *Trial*.

Example:

Overwritten attributes.

```
import copy
import datetime

import optuna


def objective(trial):
    x = trial.suggest_uniform("x", -1, 1)
```

(下页继续)

(续上页)

```
    # this user attribute always differs
    trial.set_user_attr("evaluation time", datetime.datetime.now())

    return x ** 2


study = optuna.create_study()
study.optimize(objective, n_trials=3)

best_trial = study.best_trial
best_trial_copy = copy.deepcopy(best_trial)

# re-evaluate
objective(best_trial)

# the user attribute is overwritten by re-evaluation
assert best_trial.user_attrs != best_trial_copy.user_attrs
```

备注： Please refer to *Trial* for details of methods and properties.

number

Unique and consecutive number of *Trial* for each *Study*. Note that this field uses zero-based numbering.

state

TrialState of the *Trial*.

value

Objective value of the *Trial*.

values

Sequence of objective values of the *Trial*. The length is greater than 1 if the problem is multi-objective optimization.

datetime_start

Datetime where the *Trial* started.

datetime_complete

Datetime where the *Trial* finished.

params

Dictionary that contains suggested parameters.

user_attrs

Dictionary that contains the attributes of the *Trial* set with *optuna.trial.Trial.set_user_attr()*.

intermediate_values

Intermediate objective values set with *optuna.trial.Trial.report()*.

引发 **ValueError** –If both `value` and `values` are specified.

参数

- **number** (*int*) –
- **state** (optuna.trial._state.TrialState) –
- **value** (*Optional[float]*) –
- **datetime_start** (*Optional[datetime.datetime]*) –
- **datetime_complete** (*Optional[datetime.datetime]*) –
- **params** (*Dict[str, Any]*) –
- **distributions** (*Dict[str, optuna.distributions.BaseDistribution]*) –
- **user_attrs** (*Dict[str, Any]*) –
- **system_attrs** (*Dict[str, Any]*) –
- **intermediate_values** (*Dict[int, float]*) –
- **trial_id** (*int*) –
- **values** (*Optional[Sequence[float]]*) –

返回类型 None

Methods

report(value, step)	Interface of report function.
set_system_attr(key, value)	
set_user_attr(key, value)	
should_prune()	Suggest whether the trial should be pruned or not.
suggest_categorical(name, choices)	
suggest_discrete_uniform(name, low, high, q)	
suggest_float(name, low, high, *[, step, log])	
suggest_int(name, low, high[, step, log])	
suggest_loguniform(name, low, high)	
suggest_uniform(name, low, high)	

Attributes

datetime_start	
distributions	Dictionary that contains the distributions of *params*.
duration	Return the elapsed time taken to complete the trial.
last_step	Return the maximum step of *intermediate_values* in the trial.
number	
params	
system_attrs	
user_attrs	
value	
values	

property distributions: Dict[str, optuna.distributions.BaseDistribution]

Dictionary that contains the distributions of *params*.

property duration: Optional[datetime.timedelta]

Return the elapsed time taken to complete the trial.

返回 The duration.

property last_step: Optional[int]

Return the maximum step of *intermediate_values* in the trial.

返回 The maximum step of intermediates.

report(*value*, *step*)

Interface of report function.

Since *FrozenTrial* is not pruned, this report function does nothing.

参见:

Please refer to *should_prune()*.

参数

- **value** (*float*) –A value returned from the objective function.

- **step** (*int*) –Step of the trial (e.g., Epoch of neural network training). Note that pruners assume that `step` starts at zero. For example, *MedianPruner* simply checks if `step` is less than `n_warmup_steps` as the warmup mechanism.

返回类型 None

should_prune()

Suggest whether the trial should be pruned or not.

The suggestion is always `False` regardless of a pruning algorithm.

备注： *FrozenTrial* only samples one combination of parameters.

返回 `False`.

返回类型 bool

optuna.trial.TrialState

class optuna.trial.**TrialState**(*value*)

State of a *Trial*.

RUNNING

The *Trial* is running.

COMPLETE

The *Trial* has been finished without any error.

PRUNED

The *Trial* has been pruned with *TrialPruned*.

FAIL

The *Trial* has failed due to an uncaught error.

Methods

`is_finished()`

Attributes

RUNNING
COMPLETE
PRUNED
FAIL
WAITING

optuna.trial.create_trial

optuna.trial.**create_trial**(*, *state=TrialState.COMPLETE*, *value=None*, *values=None*, *params=None*, *distributions=None*, *user_attrs=None*, *system_attrs=None*, *intermediate_values=None*)

Create a new *FrozenTrial*.

示例

```
import optuna
from optuna.distributions import CategoricalDistribution
from optuna.distributions import UniformDistribution

trial = optuna.trial.create_trial(
    params={"x": 1.0, "y": 0},
    distributions={
        "x": UniformDistribution(0, 10),
        "y": CategoricalDistribution([-1, 0, 1]),
    },
    value=5.0,
)

assert isinstance(trial, optuna.trial.FrozenTrial)
assert trial.value == 5.0
assert trial.params == {"x": 1.0, "y": 0}
```

参见:

See *add_trial()* for how this function can be used to create a study from existing trials.

备注： Please note that this is a low-level API. In general, trials that are passed to objective functions are created inside *optimize()*.

备注： When `state` is `TrialState.COMPLETE`, the following parameters are required: * `params` * `distributions` * `value` or `values`

参数

- **state** (optuna.trial._state.TrialState) –Trial state.
- **value** (*Optional[float]*) –Trial objective value. Must be specified if `state` is `None` or *TrialState.COMPLETE*.
- **values** (*Optional[Sequence[float]]*) –Sequence of the trial objective values. The length is greater than 1 if the problem is multi-objective optimization. Must be specified if `state` is `None` or *TrialState.COMPLETE*.
- **params** (*Optional[Dict[str, Any]]*) –Dictionary with suggested parameters of the trial.
- **distributions** (*Optional[Dict[str, optuna.distributions.BaseDistribution]]*) –Dictionary with parameter distributions of the trial.
- **user_attrs** (*Optional[Dict[str, Any]]*) –Dictionary with user attributes.
- **system_attrs** (*Optional[Dict[str, Any]]*) –Dictionary with system attributes. Should not have to be used for most users.
- **intermediate_values** (*Optional[Dict[int, float]]*) –Dictionary with intermediate objective values of the trial.

返回 Created trial.

引发 **ValueError** –If both `value` and `values` are specified.

返回类型 *optuna.trial._frozen.FrozenTrial*

备注： Added in v2.0.0 as an experimental feature. The interface may change in newer versions without prior notice. See https://github.com/optuna/optuna/releases/tag/v2.0.0.

6.3.15 optuna.visualization

The *visualization* module provides utility functions for plotting the optimization process using plotly and matplotlib. Plotting functions generally take a *Study* object and optional parameters are passed as a list to the `params` argument.

备注： 在*optuna.visualization*模块中，以下函数使用 plotly 来绘制图像，但 JupyterLab 默认情况下不能渲染它们。请按照这个 installation guide 在 JupyterLab 中显示图像。

optuna.visualization.plot_contour	将 study 中的参数关系绘制成等高线图。
optuna.visualization.plot_edf	绘制 study 的目标值 EDF（经验分布函数）。
optuna.visualization.plot_intermediate_values	绘制一个 study 中所有 trial 的中间值。
optuna.visualization.plot_optimization_history	绘制一个 study 中所有 trial 的优化历史记录。
optuna.visualization.plot_parallel_coordinate	Plot the high-dimensional parameter relationships in a study.
optuna.visualization.plot_param_importances	绘制超参数重要性。
optuna.visualization.plot_pareto_front	绘制 study 的帕累托前沿面。
optuna.visualization.plot_slice	将 study 中的参数关系绘制成切片图。
optuna.visualization.is_available	返回是否可以使用 plotly 进行可视化。

optuna.visualization.plot_contour

optuna.visualization.**plot_contour**(*study*, *params=None*, *, *target=None*, *target_name='Objective Value'*)

Plot the parameter relationship as contour plot in a study.

Note that, If a parameter contains missing values, a trial with missing values is not plotted.

示例

The following code snippet shows how to plot the parameter relationship as contour plot.

```
import optuna


def objective(trial):
    x = trial.suggest_float("x", -100, 100)
```

(下页继续)

(续上页)

```
    y = trial.suggest_categorical("y", [-1, 0, 1])
    return x ** 2 + y


sampler = optuna.samplers.TPESampler(seed=10)
study = optuna.create_study(sampler=sampler)
study.optimize(objective, n_trials=30)

fig = optuna.visualization.plot_contour(study, params=["x", "y"])
fig.show()
```

参数

- **study** (optuna.study.Study) –A *Study* object whose trials are plotted for their target values.
- **params** (*Optional[List[str]]*) –Parameter list to visualize. The default is all parameters.
- **target** (*Optional[Callable[[optuna.trial._frozen.FrozenTrial], float]]*) –A function to specify the value to display. If it is None and study is being used for single-objective optimization, the objective values are plotted.

  备注： Specify this argument if study is being used for multi-objective optimization.

- **target_name** (*str*) –Target's name to display on the color bar.

返回 A plotly.graph_objs.Figure object.

引发 **ValueError** –If target is None and study is being used for multi-objective optimization.

返回类型 plotly.graph_objs._figure.Figure

optuna.visualization.plot_edf

optuna.visualization.**plot_edf**(*study*, ***, *target=None*, *target_name='Objective Value'*)

Plot the objective value EDF (empirical distribution function) of a study.

Note that only the complete trials are considered when plotting the EDF.

备注： EDF is useful to analyze and improve search spaces. For instance, you can see a practical use case of EDF in the paper Designing Network Design Spaces.

备注： The plotted EDF assumes that the value of the objective function is in accordance with the uniform distribution over the objective space.

示例

The following code snippet shows how to plot EDF.

```
import math

import optuna


def ackley(x, y):
    a = 20 * math.exp(-0.2 * math.sqrt(0.5 * (x ** 2 + y ** 2)))
    b = math.exp(0.5 * (math.cos(2 * math.pi * x) + math.cos(2 * math.pi * y)))
    return -a - b + math.e + 20


def objective(trial, low, high):
    x = trial.suggest_float("x", low, high)
    y = trial.suggest_float("y", low, high)
    return ackley(x, y)


sampler = optuna.samplers.RandomSampler(seed=10)

# Widest search space.
study0 = optuna.create_study(study_name="x=[0,5), y=[0,5)", sampler=sampler)
study0.optimize(lambda t: objective(t, 0, 5), n_trials=500)

# Narrower search space.
study1 = optuna.create_study(study_name="x=[0,4), y=[0,4)", sampler=sampler)
study1.optimize(lambda t: objective(t, 0, 4), n_trials=500)

# Narrowest search space but it doesn't include the global optimum point.
study2 = optuna.create_study(study_name="x=[1,3), y=[1,3)", sampler=sampler)
study2.optimize(lambda t: objective(t, 1, 3), n_trials=500)

fig = optuna.visualization.plot_edf([study0, study1, study2])
fig.show()
```

参数

- **study** (*Union[optuna.study.Study, Sequence[optuna.study.Study]]*) – A target *Study* object. You can pass multiple studies if you want to compare those EDFs.
- **target** (*Optional[Callable[[optuna.trial._frozen.FrozenTrial], float]]*) – A function to specify the value to display. If it is `None` and `study` is being used for single-objective optimization, the objective values are plotted.

  备注： Specify this argument if `study` is being used for multi-objective optimization.

- **target_name** (*str*) – Target's name to display on the axis label.

返回 A `plotly.graph_objs.Figure` object.

引发 **ValueError** – If `target` is `None` and `study` is being used for multi-objective optimization.

返回类型 plotly.graph_objs._figure.Figure

optuna.visualization.plot_intermediate_values

optuna.visualization.**plot_intermediate_values**(*study*)

Plot intermediate values of all trials in a study.

示例

The following code snippet shows how to plot intermediate values.

```
import optuna


def f(x):
    return (x - 2) ** 2


def df(x):
    return 2 * x - 4


def objective(trial):
    lr = trial.suggest_float("lr", 1e-5, 1e-1, log=True)

    x = 3
    for step in range(128):
        y = f(x)
```

(下页继续)

(续上页)

```
        trial.report(y, step=step)
        if trial.should_prune():
            raise optuna.TrialPruned()

        gy = df(x)
        x -= gy * lr

    return y


sampler = optuna.samplers.TPESampler(seed=10)
study = optuna.create_study(sampler=sampler)
study.optimize(objective, n_trials=16)

fig = optuna.visualization.plot_intermediate_values(study)
fig.show()
```

参数 **study** (optuna.study.Study) – A *Study* object whose trials are plotted for their intermediate values.

返回 A plotly.graph_objs.Figure object.

返回类型 plotly.graph_objs._figure.Figure

optuna.visualization.plot_optimization_history

optuna.visualization.**plot_optimization_history**(*study*, ***, *target=None*, *target_name='Objective Value'*)

Plot optimization history of all trials in a study.

示例

The following code snippet shows how to plot optimization history.

```
import optuna


def objective(trial):
    x = trial.suggest_float("x", -100, 100)
    y = trial.suggest_categorical("y", [-1, 0, 1])
    return x ** 2 + y
```

(下页继续)

(续上页)

```
sampler = optuna.samplers.TPESampler(seed=10)
study = optuna.create_study(sampler=sampler)
study.optimize(objective, n_trials=10)

fig = optuna.visualization.plot_optimization_history(study)
fig.show()
```

参数

- **study** (optuna.study.Study) –A *Study* object whose trials are plotted for their target values.
- **target** (*Optional[Callable[[optuna.trial._frozen.FrozenTrial], float]]*) –A function to specify the value to display. If it is None and study is being used for single-objective optimization, the objective values are plotted.

  备注： Specify this argument if study is being used for multi-objective optimization.

- **target_name** (*str*) –Target's name to display on the axis label and the legend.

返回 A plotly.graph_objs.Figure object.

引发 **ValueError** –If target is None and study is being used for multi-objective optimization.

返回类型 plotly.graph_objs._figure.Figure

optuna.visualization.plot_parallel_coordinate

optuna.visualization.**plot_parallel_coordinate**(*study*, *params=None*, ***, *target=None*, *target_name='Objective Value'*)

Plot the high-dimensional parameter relationships in a study.

Note that, If a parameter contains missing values, a trial with missing values is not plotted.

示例

The following code snippet shows how to plot the high-dimensional parameter relationships.

```
import optuna


def objective(trial):
```

(下页继续)

(续上页)

```
    x = trial.suggest_float("x", -100, 100)
    y = trial.suggest_categorical("y", [-1, 0, 1])
    return x ** 2 + y


sampler = optuna.samplers.TPESampler(seed=10)
study = optuna.create_study(sampler=sampler)
study.optimize(objective, n_trials=10)

fig = optuna.visualization.plot_parallel_coordinate(study, params=["x", "y"])
fig.show()
```

参数

- **study** (optuna.study.Study) –A *Study* object whose trials are plotted for their target values.
- **params** (*Optional[List[str]]*) –Parameter list to visualize. The default is all parameters.
- **target** (*Optional[Callable[[optuna.trial._frozen.FrozenTrial], float]]*) –A function to specify the value to display. If it is None and study is being used for single-objective optimization, the objective values are plotted.

  备注: Specify this argument if study is being used for multi-objective optimization.

- **target_name** (*str*) –Target's name to display on the axis label and the legend.

返回 A plotly.graph_objs.Figure object.

引发 **ValueError** –If target is None and study is being used for multi-objective optimization.

返回类型 plotly.graph_objs._figure.Figure

optuna.visualization.plot_param_importances

optuna.visualization.**plot_param_importances**(*study*, *evaluator=None*, *params=None*, ***, *target=None*, *target_name='Objective Value'*)

Plot hyperparameter importances.

示例

The following code snippet shows how to plot hyperparameter importances.

```
import optuna


def objective(trial):
    x = trial.suggest_int("x", 0, 2)
    y = trial.suggest_float("y", -1.0, 1.0)
    z = trial.suggest_float("z", 0.0, 1.5)
    return x ** 2 + y ** 3 - z ** 4


sampler = optuna.samplers.RandomSampler(seed=10)
study = optuna.create_study(sampler=sampler)
study.optimize(objective, n_trials=100)

fig = optuna.visualization.plot_param_importances(study)
fig.show()
```

参见:

This function visualizes the results of *optuna.importance.get_param_importances()*.

参数

- **study** (optuna.study.Study) –An optimized study.
- **evaluator** (*Optional[optuna.importance._base.BaseImportanceEvaluator]*) –An importance evaluator object that specifies which algorithm to base the importance assessment on. Defaults to *FanovaImportanceEvaluator*.
- **params** (*Optional[List[str]]*) –A list of names of parameters to assess. If None, all parameters that are present in all of the completed trials are assessed.
- **target** (*Optional[Callable[[optuna.trial._frozen.FrozenTrial], float]]*) –A function to specify the value to display. If it is None and study is being used for single-objective optimization, the objective values are plotted.

  备注: Specify this argument if study is being used for multi-objective optimization.

- **target_name** (*str*) –Target's name to display on the axis label.

返回 A plotly.graph_objs.Figure object.

引发 **ValueError** –If `target` is `None` and `study` is being used for multi-objective optimization.

返回类型 plotly.graph_objs._figure.Figure

optuna.visualization.plot_pareto_front

optuna.visualization.**plot_pareto_front**(*study*, ***, *target_names=None*, *include_dominated_trials=True*, *axis_order=None*)

Plot the Pareto front of a study.

示例

The following code snippet shows how to plot the Pareto front of a study.

```
import optuna


def objective(trial):
    x = trial.suggest_float("x", 0, 5)
    y = trial.suggest_float("y", 0, 3)

    v0 = 4 * x ** 2 + 4 * y ** 2
    v1 = (x - 5) ** 2 + (y - 5) ** 2
    return v0, v1


study = optuna.create_study(directions=["minimize", "minimize"])
study.optimize(objective, n_trials=50)

fig = optuna.visualization.plot_pareto_front(study)
fig.show()
```

参数

- **study** (optuna.study.Study) –A *Study* object whose trials are plotted for their objective values.
- **target_names** (*Optional[List[str]]*) –Objective name list used as the axis titles. If `None` is specified, "Objective {objective_index}" is used instead.
- **include_dominated_trials** (*bool*) –A flag to include all dominated trial's objective values.
- **axis_order** (*Optional[List[int]]*) –A list of indices indicating the axis order. If `None` is specified, default order is used.

返回 A `plotly.graph_objs.Figure` object.

引发 **ValueError** –If the number of objectives of `study` isn't 2 or 3.

返回类型 plotly.graph_objs._figure.Figure

备注: Added in v2.4.0 as an experimental feature. The interface may change in newer versions without prior notice. See https://github.com/optuna/optuna/releases/tag/v2.4.0.

---

optuna.visualization.plot_slice

`optuna.visualization.`**`plot_slice`**(*study*, *params=None*, *, *target=None*, *target_name='Objective Value'*)

Plot the parameter relationship as slice plot in a study.

Note that, If a parameter contains missing values, a trial with missing values is not plotted.

示例

The following code snippet shows how to plot the parameter relationship as slice plot.

```
import optuna


def objective(trial):
    x = trial.suggest_float("x", -100, 100)
    y = trial.suggest_categorical("y", [-1, 0, 1])
    return x ** 2 + y


sampler = optuna.samplers.TPESampler(seed=10)
study = optuna.create_study(sampler=sampler)
study.optimize(objective, n_trials=10)

fig = optuna.visualization.plot_slice(study, params=["x", "y"])
fig.show()
```

参数

- **study** (`optuna.study.Study`) –A *Study* object whose trials are plotted for their target values.
- **params** (*Optional[List[str]]*) –Parameter list to visualize. The default is all parameters.

- **target** (*Optional[Callable[[optuna.trial._frozen.FrozenTrial], float]]*) –A function to specify the value to display. If it is `None` and `study` is being used for single-objective optimization, the objective values are plotted.

  备注： Specify this argument if `study` is being used for multi-objective optimization.

- **target_name** (*str*) –Target's name to display on the axis label.

返回 A `plotly.graph_objs.Figure` object.

引发 **ValueError** –If `target` is `None` and `study` is being used for multi-objective optimization.

返回类型 plotly.graph_objs._figure.Figure

optuna.visualization.is_available

optuna.visualization.**is_available**()

Returns whether visualization with plotly is available or not.

备注： *visualization* module depends on plotly version 4.0.0 or higher. If a supported version of plotly isn't installed in your environment, this function will return `False`. In such case, please execute `$ pip install -U plotly>=4.0.0` to install plotly.

返回 `True` if visualization with plotly is available, `False` otherwise.

返回类型 bool

备注： *optuna.visualization.matplotlib* 模块的后端是 Matplotlib。

optuna.visualization.matplotlib

备注： The following functions use Matplotlib as a backend.

optuna.visualization.matplotlib.plot_contour	使用 Matplotlib 将 study 中参数关系画成等高线图。
optuna.visualization.matplotlib.plot_edf	使用 Matplotlib 绘制 study 中目标函数值的 EDF (经验分布函数)。
optuna.visualization.matplotlib.plot_intermediate_values	使用 Matplotlib 绘制 study 中所有 trial 的中间值。
optuna.visualization.matplotlib.plot_optimization_history	使用 Matplotlib 绘制 study 中所有 trial 的优化历史。
optuna.visualization.matplotlib.plot_parallel_coordinate	Plot the high-dimensional parameter relationships in a study with Matplotlib.
optuna.visualization.matplotlib.plot_param_importances	使用 Matplotlib 绘制超参数重要性。
optuna.visualization.matplotlib.plot_slice	使用 Matplotlib 将 study 中的参数关系绘制成切片图。
optuna.visualization.matplotlib.is_available	返回 Matplotlib 可视化是否可用。

optuna.visualization.matplotlib.plot_contour

optuna.visualization.matplotlib.**plot_contour**(*study*, *params=None*, *, *target=None*, *target_name='Objective Value'*)

Plot the parameter relationship as contour plot in a study with Matplotlib.

Note that, if a parameter contains missing values, a trial with missing values is not plotted.

参见:

Please refer to *optuna.visualization.plot_contour()* for an example.

> **警告:** Output figures of this Matplotlib-based *plot_contour()* function would be different from those of the Plotly-based *plot_contour()*.

示例

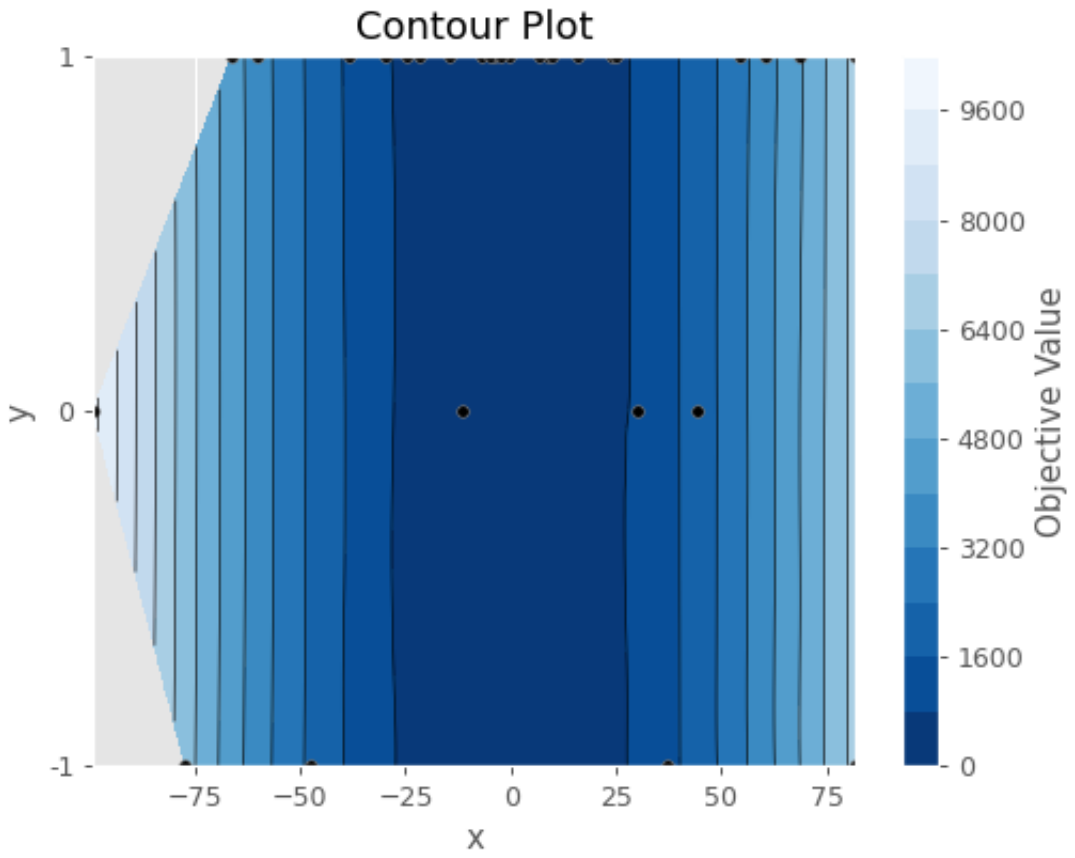
The following code snippet shows how to plot the parameter relationship as contour plot.

```
import optuna


def objective(trial):
```

(下页继续)

(续上页)

```
    x = trial.suggest_float("x", -100, 100)
    y = trial.suggest_categorical("y", [-1, 0, 1])
    return x ** 2 + y


sampler = optuna.samplers.TPESampler(seed=10)
study = optuna.create_study(sampler=sampler)
study.optimize(objective, n_trials=30)

optuna.visualization.matplotlib.plot_contour(study, params=["x", "y"])
```



参数

- **study** (optuna.study.Study) –A *Study* object whose trials are plotted for their target values.
- **params** (*Optional[List[str]]*) –Parameter list to visualize. The default is all parameters.
- **target** (*Optional[Callable[[optuna.trial._frozen.FrozenTrial],*

float]) –A function to specify the value to display. If it is `None` and `study` is being used for single-objective optimization, the objective values are plotted.

备注： Specify this argument if `study` is being used for multi-objective optimization.

- **target_name** (*str*) –Target's name to display on the color bar.

返回 A `matplotlib.axes.Axes` object.

引发 **ValueError** –If `target` is `None` and `study` is being used for multi-objective optimization.

返回类型 matplotlib.axes._axes.Axes

备注： Added in v2.2.0 as an experimental feature. The interface may change in newer versions without prior notice. See https://github.com/optuna/optuna/releases/tag/v2.2.0.

---

optuna.visualization.matplotlib.plot_edf

optuna.visualization.matplotlib.**plot_edf**(*study*, *, *target=None*, *target_name='Objective Value'*)

Plot the objective value EDF (empirical distribution function) of a study with Matplotlib.

参见:

Please refer to *optuna.visualization.plot_edf()* for an example, where this function can be replaced with it.

示例

The following code snippet shows how to plot EDF.

```
import math

import optuna


def ackley(x, y):
    a = 20 * math.exp(-0.2 * math.sqrt(0.5 * (x ** 2 + y ** 2)))
    b = math.exp(0.5 * (math.cos(2 * math.pi * x) + math.cos(2 * math.pi * y)))
    return -a - b + math.e + 20


def objective(trial, low, high):
```

(下页继续)

(续上页)

```
    x = trial.suggest_float("x", low, high)
    y = trial.suggest_float("y", low, high)
    return ackley(x, y)


sampler = optuna.samplers.RandomSampler(seed=10)

# Widest search space.
study0 = optuna.create_study(study_name="x=[0,5), y=[0,5)", sampler=sampler)
study0.optimize(lambda t: objective(t, 0, 5), n_trials=500)

# Narrower search space.
study1 = optuna.create_study(study_name="x=[0,4), y=[0,4)", sampler=sampler)
study1.optimize(lambda t: objective(t, 0, 4), n_trials=500)

# Narrowest search space but it doesn't include the global optimum point.
study2 = optuna.create_study(study_name="x=[1,3), y=[1,3)", sampler=sampler)
study2.optimize(lambda t: objective(t, 1, 3), n_trials=500)

optuna.visualization.matplotlib.plot_edf([study0, study1, study2])
```

参数

- **study** (*Union[optuna.study.Study, Sequence[optuna.study.Study]]*) –A target *Study* object. You can pass multiple studies if you want to compare those EDFs.
- **target** (*Optional[Callable[[optuna.trial._frozen.FrozenTrial], float]]*) –A function to specify the value to display. If it is `None` and `study` is being used for single-objective optimization, the objective values are plotted.

  备注: Specify this argument if `study` is being used for multi-objective optimization.

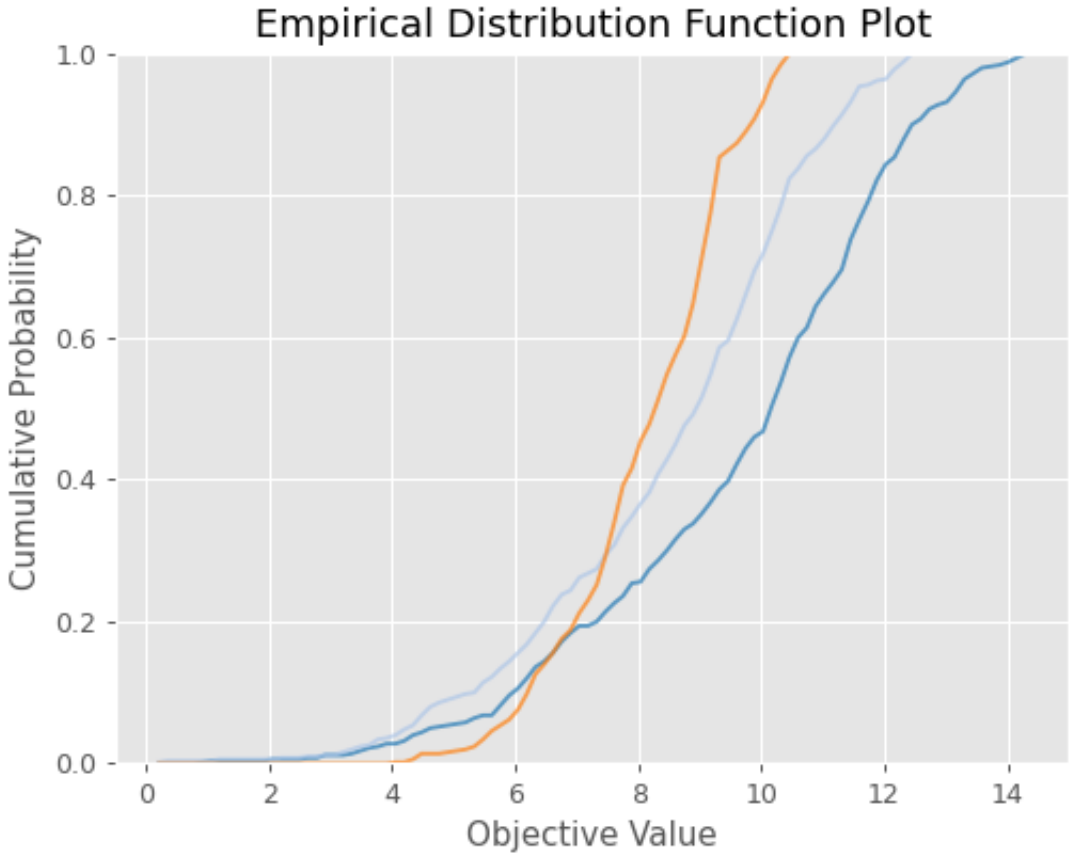
- **target_name** (*str*) –Target's name to display on the axis label.

返回 A `matplotlib.axes.Axes` object.

引发 **ValueError** –If `target` is `None` and `study` is being used for multi-objective optimization.

返回类型 matplotlib.axes._axes.Axes

备注: Added in v2.2.0 as an experimental feature. The interface may change in newer versions without prior


Empirical Distribution Function Plot
Cumulative Probability
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
10
12
14
Objective Value

notice. See https://github.com/optuna/optuna/releases/tag/v2.2.0.

optuna.visualization.matplotlib.plot_intermediate_values

optuna.visualization.matplotlib.**plot_intermediate_values**(*study*)

Plot intermediate values of all trials in a study with Matplotlib.

示例

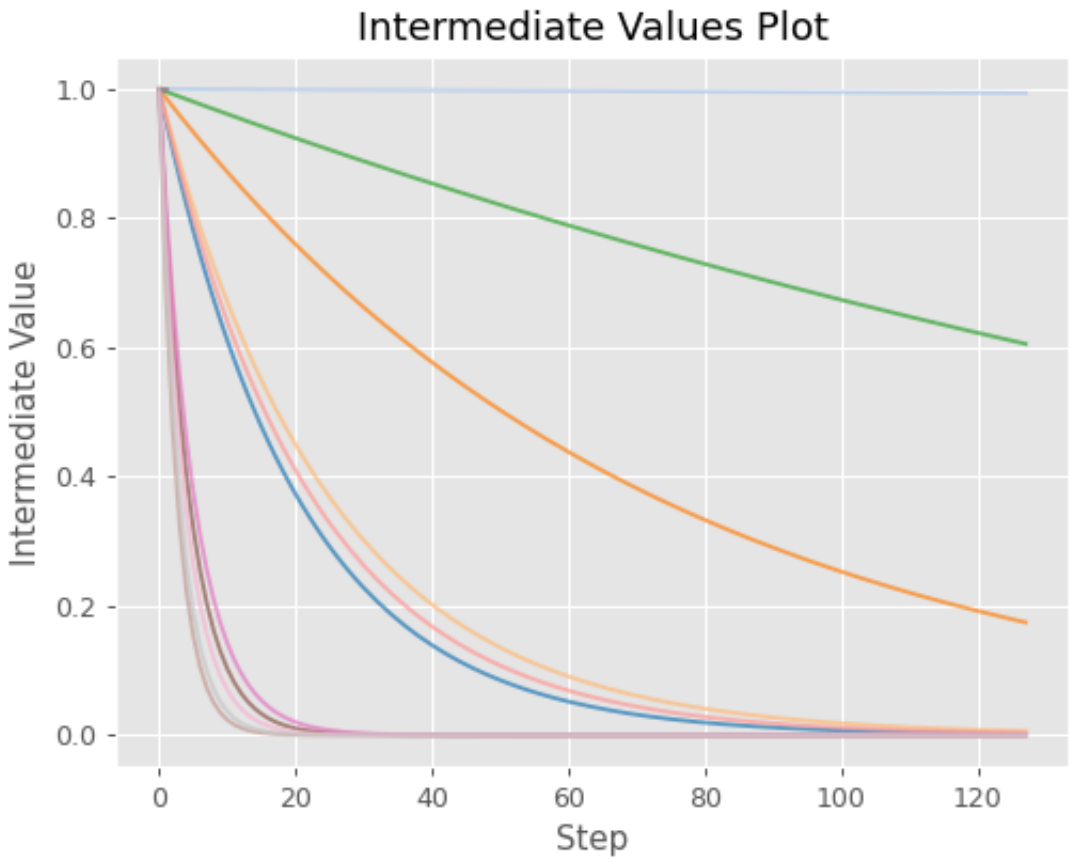
The following code snippet shows how to plot intermediate values.

```
import optuna


def f(x):
    return (x - 2) ** 2


def df(x):
    return 2 * x - 4


def objective(trial):
    lr = trial.suggest_float("lr", 1e-5, 1e-1, log=True)

    x = 3
    for step in range(128):
        y = f(x)

        trial.report(y, step=step)
        if trial.should_prune():
            raise optuna.TrialPruned()

        gy = df(x)
        x -= gy * lr

    return y


sampler = optuna.samplers.TPESampler(seed=10)
study = optuna.create_study(sampler=sampler)
study.optimize(objective, n_trials=16)
```

(下页继续)

(续上页)

```
optuna.visualization.matplotlib.plot_intermediate_values(study)
```



参见:

Please refer to *optuna.visualization.plot_intermediate_values()* for an example.

参数 **study** (optuna.study.Study) –A *Study* object whose trials are plotted for their intermediate values.

返回 A matplotlib.axes.Axes object.

返回类型 matplotlib.axes._axes.Axes

备注: Added in v2.2.0 as an experimental feature. The interface may change in newer versions without prior notice. See https://github.com/optuna/optuna/releases/tag/v2.2.0.

---

optuna.visualization.matplotlib.plot_optimization_history

optuna.visualization.matplotlib.**plot_optimization_history**(*study*, *, *target=None*, *target_name='Objective Value'*)

Plot optimization history of all trials in a study with Matplotlib.

参见:

Please refer to *optuna.visualization.plot_optimization_history()* for an example.

示例

The following code snippet shows how to plot optimization history.

```
import optuna


def objective(trial):
    x = trial.suggest_float("x", -100, 100)
    y = trial.suggest_categorical("y", [-1, 0, 1])
    return x ** 2 + y

sampler = optuna.samplers.TPESampler(seed=10)
study = optuna.create_study(sampler=sampler)
study.optimize(objective, n_trials=10)

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

参数

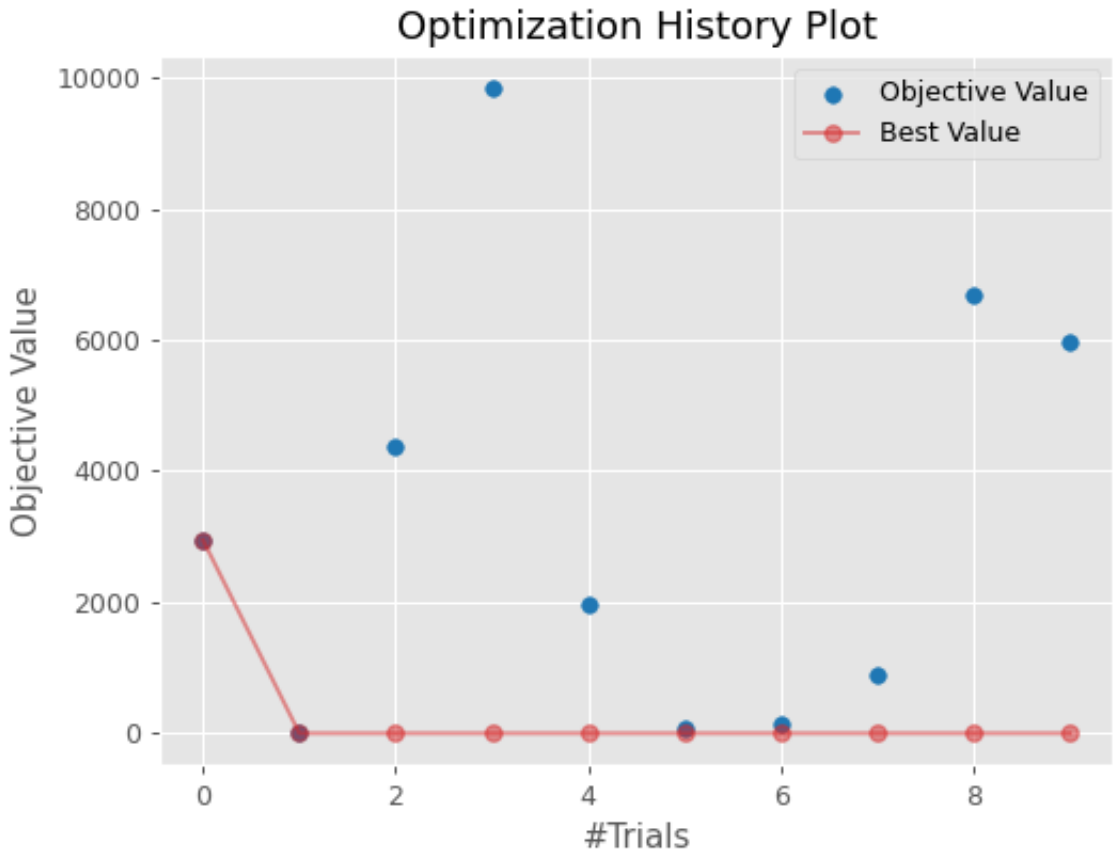
- **study** (optuna.study.Study) – A *Study* object whose trials are plotted for their target values.
- **target** (*Optional[Callable[[optuna.trial._frozen.FrozenTrial], float]]*) – A function to specify the value to display. If it is None and study is being used for single-objective optimization, the objective values are plotted.

  备注: Specify this argument if study is being used for multi-objective optimization.

- **target_name** (*str*) – Target's name to display on the axis label and the legend.

返回 A matplotlib.axes.Axes object.

引发 **ValueError** – If target is None and study is being used for multi-objective optimization.


Optimization History Plot
Objective Value
Best Value
10000
8000
6000
4000
2000
0
Objective Value
0
2
4
6
8
#Trials

返回类型 matplotlib.axes._axes.Axes

备注： Added in v2.2.0 as an experimental feature. The interface may change in newer versions without prior notice. See https://github.com/optuna/optuna/releases/tag/v2.2.0.

---

optuna.visualization.matplotlib.plot_parallel_coordinate

optuna.visualization.matplotlib.**plot_parallel_coordinate**(*study*, *params=None*, *, *target=None*, *target_name='Objective Value'*)

Plot the high-dimensional parameter relationships in a study with Matplotlib.

参见:

Please refer to *optuna.visualization.plot_parallel_coordinate()* for an example.

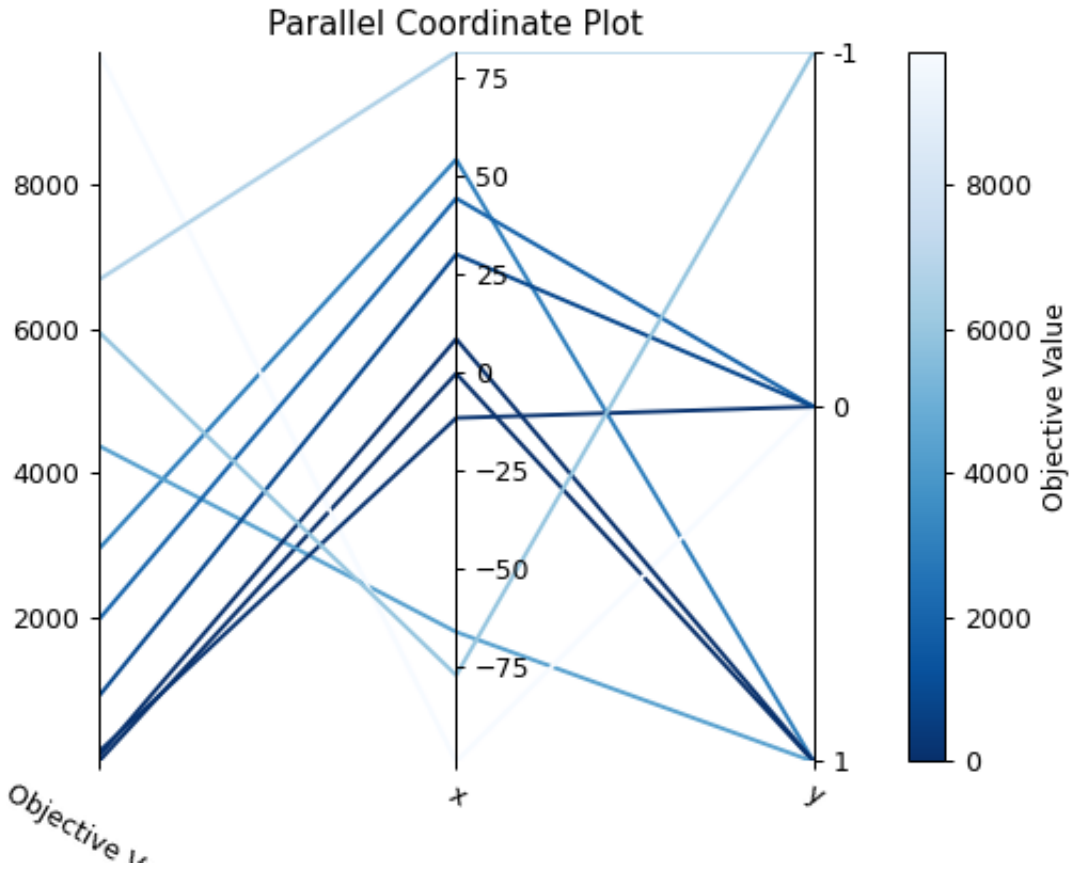
示例

The following code snippet shows how to plot the high-dimensional parameter relationships.

```
import optuna

def objective(trial):
    x = trial.suggest_float("x", -100, 100)
    y = trial.suggest_categorical("y", [-1, 0, 1])
    return x ** 2 + y


sampler = optuna.samplers.TPESampler(seed=10)
study = optuna.create_study(sampler=sampler)
study.optimize(objective, n_trials=10)

optuna.visualization.matplotlib.plot_parallel_coordinate(study, params=["x", "y"])
```

参数

- **study** (optuna.study.Study) –A *Study* object whose trials are plotted for their target values.
- **params** (*Optional[List[str]]*) –Parameter list to visualize. The default is all parameters.


Parallel Coordinate Plot
8000
6000
4000
2000
75
50
25
0
−25
−50
−75
-1
0
1
x
y
Objective Value
8000
6000
4000
2000
0

- **target** (*Optional[Callable[[optuna.trial._frozen.FrozenTrial], float]]*) –A function to specify the value to display. If it is `None` and `study` is being used for single-objective optimization, the objective values are plotted.

  备注： Specify this argument if `study` is being used for multi-objective optimization.

- **target_name** (*str*) –Target's name to display on the axis label and the legend.

返回 A `matplotlib.axes.Axes` object.

引发 **ValueError** –If `target` is `None` and `study` is being used for multi-objective optimization.

返回类型 matplotlib.axes._axes.Axes

备注： Added in v2.2.0 as an experimental feature. The interface may change in newer versions without prior notice. See https://github.com/optuna/optuna/releases/tag/v2.2.0.

optuna.visualization.matplotlib.plot_param_importances

optuna.visualization.matplotlib.**plot_param_importances**(*study*, *evaluator=None*, *params=None*, ***, *target=None*, *target_name='Objective Value'*)

Plot hyperparameter importances with Matplotlib.

参见:

Please refer to *optuna.visualization.plot_param_importances()* for an example.

示例

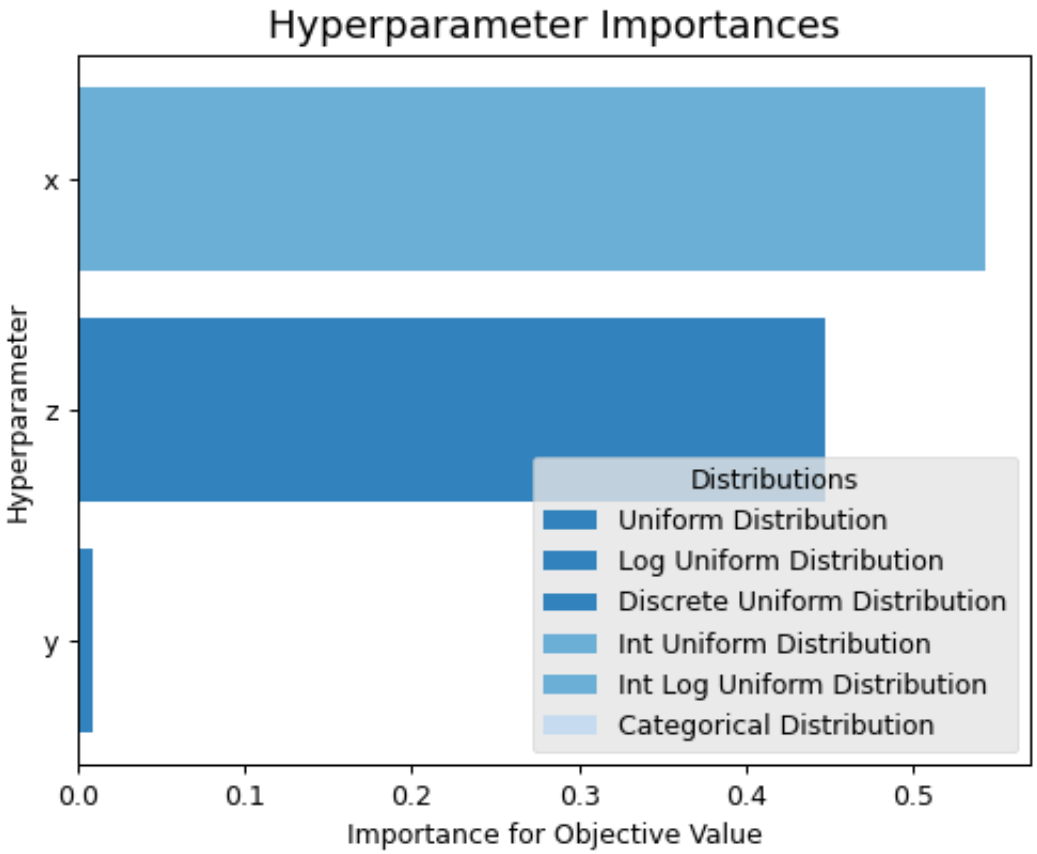
The following code snippet shows how to plot hyperparameter importances.

```
import optuna


def objective(trial):
    x = trial.suggest_int("x", 0, 2)
    y = trial.suggest_float("y", -1.0, 1.0)
    z = trial.suggest_float("z", 0.0, 1.5)
    return x ** 2 + y ** 3 - z ** 4
```

(下页继续)

(续上页)

```
sampler = optuna.samplers.RandomSampler(seed=10)
study = optuna.create_study(sampler=sampler)
study.optimize(objective, n_trials=100)

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



参数

- **study** (optuna.study.Study) –An optimized study.
- **evaluator** (*Optional[optuna.importance._base.BaseImportanceEvaluator]*) –An importance evaluator object that specifies which algorithm to base the importance assessment on. Defaults to *FanovaImportanceEvaluator*.
- **params** (*Optional[List[str]]*) –A list of names of parameters to assess. If None, all parameters that are present in all of the completed trials are assessed.
- **target** (*Optional[Callable[[optuna.trial._frozen.FrozenTrial], float]]*) –A function to specify the value to display. If it is None and study is being

used for single-objective optimization, the objective values are plotted.

备注： Specify this argument if `study` is being used for multi-objective optimization.

- **target_name** (*str*) –Target's name to display on the axis label.

返回 A `matplotlib.axes.Axes` object.

引发 **ValueError** –If `target` is `None` and `study` is being used for multi-objective optimization.

返回类型 matplotlib.axes._axes.Axes

备注： Added in v2.2.0 as an experimental feature. The interface may change in newer versions without prior notice. See https://github.com/optuna/optuna/releases/tag/v2.2.0.

---

optuna.visualization.matplotlib.plot_slice

optuna.visualization.matplotlib.**plot_slice**(*study*, *params=None*, *, *target=None*, *target_name='Objective Value'*)

Plot the parameter relationship as slice plot in a study with Matplotlib.

参见:

Please refer to *optuna.visualization.plot_slice()* for an example.

示例

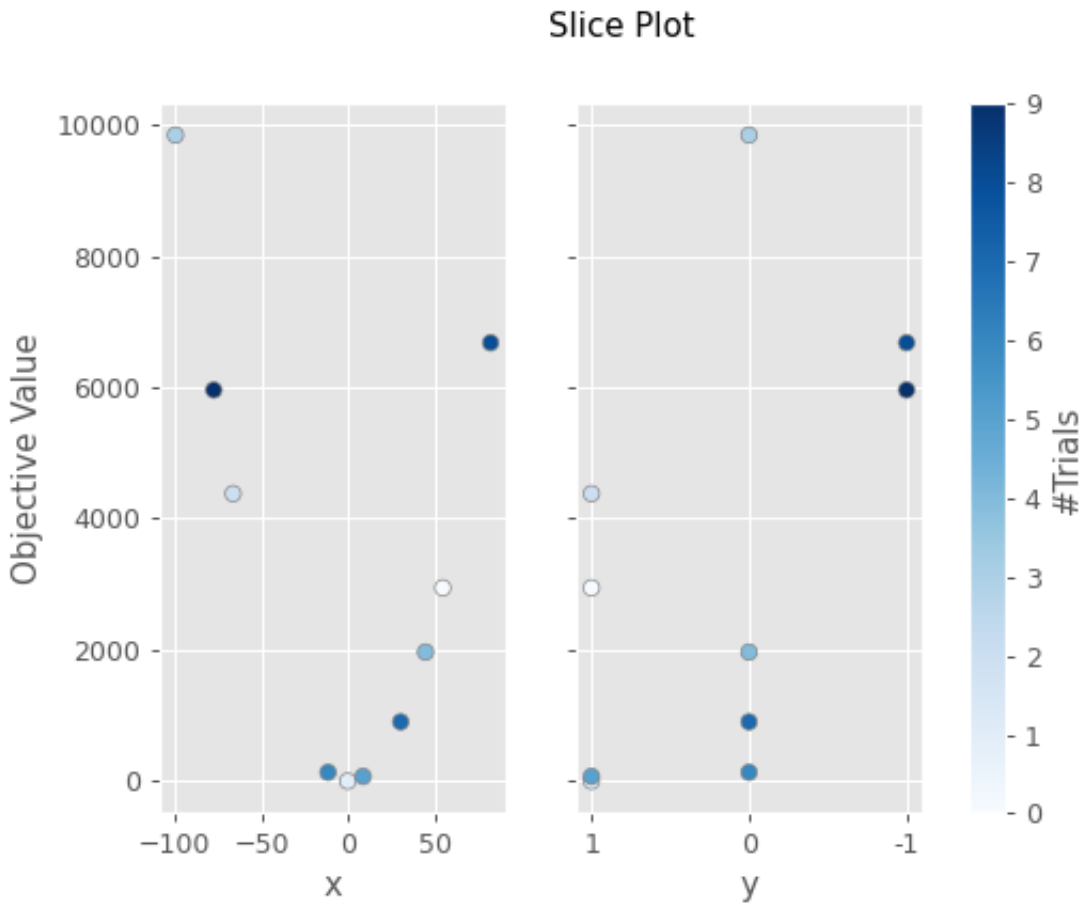
The following code snippet shows how to plot the parameter relationship as slice plot.

```
import optuna


def objective(trial):
    x = trial.suggest_float("x", -100, 100)
    y = trial.suggest_categorical("y", [-1, 0, 1])
    return x ** 2 + y


sampler = optuna.samplers.TPESampler(seed=10)
study = optuna.create_study(sampler=sampler)
study.optimize(objective, n_trials=10)
```

(下页继续)

(续上页)

```
optuna.visualization.matplotlib.plot_slice(study, params=["x", "y"])
```



参数

- **study** (optuna.study.Study) –A *Study* object whose trials are plotted for their target values.
- **params** (*Optional[List[str]]*) –Parameter list to visualize. The default is all parameters.
- **target** (*Optional[Callable[[optuna.trial._frozen.FrozenTrial], float]]*) –A function to specify the value to display. If it is None and study is being used for single-objective optimization, the objective values are plotted.

  备注: Specify this argument if study is being used for multi-objective optimization.

- **target_name** (*str*) –Target’ s name to display on the axis label.

返回 A `matplotlib.axes.Axes` object.

引发 **`ValueError`** –If `target` is `None` and `study` is being used for multi-objective optimization.

返回类型 matplotlib.axes._axes.Axes

备注： Added in v2.2.0 as an experimental feature. The interface may change in newer versions without prior notice. See https://github.com/optuna/optuna/releases/tag/v2.2.0.

---

optuna.visualization.matplotlib.is_available

`optuna.visualization.matplotlib.`**`is_available`**`()`

Returns whether visualization with Matplotlib is available or not.

备注： *`matplotlib`* module depends on Matplotlib version 3.0.0 or higher. If a supported version of Matplotlib isn't installed in your environment, this function will return `False`. In such a case, please execute `$ pip install -U matplotlib>=3.0.0` to install Matplotlib.

返回 `True` if visualization with Matplotlib is available, `False` otherwise.

返回类型 bool

备注： Added in v2.2.0 as an experimental feature. The interface may change in newer versions without prior notice. See https://github.com/optuna/optuna/releases/tag/v2.2.0.

6.4 常见问题

- *Trial* 是如何处理抛出异常的？
- *Trial* 返回的 *NaN* 是如何处理的？
- 动态地改变搜索空间会导致怎样的结果？
- 如何在两块 *GPU* 上同时跑两个 *trial?*
- 如何对目标函数进行测试？
- 在优化时，我该如何避免耗尽内存 *(running out of memory, OOM)?*

6.4.1 某某库可以和 Optuna 配合使用吗？（某某是你常用的机器学习库）

Optuna 和绝大多数机器学习库兼容，并且很容易同他们配合使用。参见 examples.

6.4.2 如何定义带有额外参数的目标函数？

有两种方法可以实现这类函数。

首先，如下例所示，可调用的 objective 类具有这个功能：

```
import optuna


class Objective(object):
    def __init__(self, min_x, max_x):
        # Hold this implementation specific arguments as the fields of the class.
        self.min_x = min_x
        self.max_x = max_x

    def __call__(self, trial):
        # Calculate an objective value by using the extra arguments.
        x = trial.suggest_float("x", self.min_x, self.max_x)
        return (x - 2) ** 2


# Execute an optimization by using an `Objective` instance.
study = optuna.create_study()
study.optimize(Objective(-100, 100), n_trials=100)
```

其次，你可以用 `lambda` 或者 `functools.partial` 来创建带有额外参数的函数（闭包）。下面是一个使用了 `lambda` 的例子：

```
import optuna

# Objective function that takes three arguments.
def objective(trial, min_x, max_x):
    x = trial.suggest_float("x", min_x, max_x)
    return (x - 2) ** 2


# Extra arguments.
min_x = -100
max_x = 100

# Execute an optimization by using the above objective function wrapped by `lambda`.
study = optuna.create_study()
study.optimize(lambda trial: objective(trial, min_x, max_x), n_trials=100)
```

Please also refer to sklearn_addtitional_args.py example, which reuses the dataset instead of loading it in each trial execution.

6.4.3 没有远程 RDB 的情况下可以使用 Optuna 吗？

可以。

在最简单的情况下，Optuna 使用内存 (in-memory) 存储：

```
study = optuna.create_study()
study.optimize(objective)
```

如果想保存和恢复 study 的话，你可以轻松地将 SQLite 用作本地存储。

```
study = optuna.create_study(study_name="foo_study", storage="sqlite:///example.db")
study.optimize(objective)  # The state of `study` will be persisted to the local␣
↪SQLite file.
```

更多细节请参考*用 RDB 后端保存/恢复 Study*。

6.4.4 如何保存和恢复 study？

有两种方法可以将 study 持久化。具体采用哪种取决于你是使用内存存储 (in-memory) 还是远程数据库存储 (RDB). 通过 `pickle` 或者 `joblib`, 采用了内存存储的 study 可以和普通的 Python 对象一样被存储和加载。比如用 `joblib` 的话：

```
study = optuna.create_study()
joblib.dump(study, "study.pkl")
```

恢复 study:

```
study = joblib.load("study.pkl")
print("Best trial until now:")
print(" Value: ", study.best_trial.value)
print(" Params: ")
for key, value in study.best_trial.params.items():
    print(f"    {key}: {value}")
```

如果你用的是 RDB, 具体细节请参考*用 RDB 后端保存/恢复 Study*.

6.4.5 如何禁用 Optuna 的日志信息？

默认情况下，Optuna 打印处于 `optuna.logging.INFO` 层级的日志信息。通过设置*optuna.logging.set_verbosity()*, 你可以改变这个层级。

比如，下面的代码可以终止打印每一个 trial 的结果：

```
optuna.logging.set_verbosity(optuna.logging.WARNING)

study = optuna.create_study()
study.optimize(objective)
# Logs like '[I 2020-07-21 13:41:45,627] Trial 0 finished with value:...' are␣
↪disabled.
```

更多的细节请参考*optuna.logging*.

6.4.6 如何在目标函数中保存训练好的机器学习模型？

Optuna 会保存超参数和对应的目标函数值，但是它不会存储诸如机器学习模型或者网络权重这样的中间数据。要保存模型或者权重的话，请利用你正在使用的机器学习库提供的对应功能。

在保存模型的时候，我们推荐将*optuna.trial.Trial.number*一同存储。这样易于之后确认对应的 trial. 比如，你可以用以下方式在目标函数中保存训练好的 SVM 模型：

```
def objective(trial):
    svc_c = trial.suggest_float("svc_c", 1e-10, 1e10, log=True)
    clf = sklearn.svm.SVC(C=svc_c)
    clf.fit(X_train, y_train)

    # Save a trained model to a file.
    with open("{}.pickle".format(trial.number), "wb") as fout:
        pickle.dump(clf, fout)
    return 1.0 - accuracy_score(y_valid, clf.predict(X_valid))
```

(下页继续)

(续上页)

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

# Load the best model.
with open("{}.pickle".format(study.best_trial.number), "rb") as fin:
    best_clf = pickle.load(fin)
print(accuracy_score(y_valid, best_clf.predict(X_valid)))
```

6.4.7 如何获得可复现的优化结果？

要让 Optuna 生成的参数可复现的话，你可以通过设置*RandomSampler* 或者*TPESampler* 中的参数 seed 来指定一个固定的随机数种子：

```
sampler = TPESampler(seed=10)  # Make the sampler behave in a deterministic way.
study = optuna.create_study(sampler=sampler)
study.optimize(objective)
```

但是这么做的需要注意以下两点。

首先，如果一个 study 的优化过程本身是分布式的或者并行的，那么这个过程中存在着固有的不确定性。因此，在这种情况下我们很难复现出同样的结果。如果你想复现结果的话，我们建议用顺序执行的方式来优化你的 study.

其次，如果你的目标函数的行为本身就是不确定的（也就是说，即使送入同样的参数，其返回值也不是唯一的），那么你就无法复现这个优化过程。要解决这个问题的话，请设置一个选项（比如随机数种子）来让你的优化目标的行为变成确定性的，前提是你用的机器学习库支持这一功能。

6.4.8 Trial 是如何处理抛出异常的？

那些抛出异常却没有对应的捕获机制的 trial 会被视作失败的 trial, 也就是处于*FAIL* 状态的 trial.

在默认情况下，除了目标函数中抛出的*TrialPruned*, 其他所有异常都会被传回给调用函数*optimize()*. 换句话说，当此类异常被抛出时，对应的 study 就会被终止。但有时候我们希望能用剩余的 trial 将该 study 继续下去。要这么做的话，你得通过*optimize()* 函数中的 catch 参数来指定要捕获的异常类型。这样，此类异常就会在 study 内部被捕获，而不会继续向外层传递。

你可以在日志信息里找到失败的 trial.

```
[W 2018-12-07 16:38:36,889] Setting status of trial#0 as TrialState.FAIL because of \
the following error: ValueError('A sample error in objective.')
```

你也可以通过查看 trial 的状态来找到它们：

```
study.trials_dataframe()
```

number	state	value	···	params	system_attrs
0	TrialState.FAIL		···	0	Setting status of trial#0 as TrialState.FAIL because of the following error: ValueError('A test error in objective.')
1	TrialState.COMPLETE	1269	···	1	

参见:

The `catch` argument in *`optimize()`*.

6.4.9 Trial 返回的 NaN 是如何处理的？

返回 `NaN` 的 trial 被视为失败的 trial, 但是它们并不会导致 study 被终止。

这些返回 `NaN` 的 trial 在日志里长这样：

```
[W 2018-12-07 16:41:59,000] Setting status of trial#2 as TrialState.FAIL because the \
objective function returned nan.
```

6.4.10 动态地改变搜索空间会导致怎样的结果？

Since parameters search spaces are specified in each call to the suggestion API, e.g. *`suggest_float()`* and *`suggest_int()`*, it is possible to, in a single study, alter the range by sampling parameters from different search spaces in different trials. The behavior when altered is defined by each sampler individually.

备注: 关于 TPE sampler 的讨论：https://github.com/optuna/optuna/issues/822

6.4.11 如何在两块 GPU 上同时跑两个 trial?

如果你的优化目标支持 GPU (CUDA) 加速，你又想指定优化所用的 GPU 的话，设置 `CUDA_VISIBLE_DEVICES` 环境变量可能是实现这一目标最轻松的方式了：

```
# On a terminal.
#
# Specify to use the first GPU, and run an optimization.
$ export CUDA_VISIBLE_DEVICES=0
```

(下页继续)

(续上页)

```
$ optuna study optimize foo.py objective --study-name foo --storage sqlite:///example.
↪db

# On another terminal.
#
# Specify to use the second GPU, and run another optimization.
$ export CUDA_VISIBLE_DEVICES=1
$ optuna study optimize bar.py objective --study-name bar --storage sqlite:///example.
↪db
```

更多细节见 CUDA C Programming Guide.

6.4.12 如何对目标函数进行测试？

在对目标函数的测试中，我们总倾向于使用固定的，而不是随机采样的参数。这时，你可以选择用*FixedTrial* 作为目标函数的输入参数。它会从一个给定的参数字典中送入固定的参数值。比如，针对函数 $x + y$, 你可以用如下方式送入两个任意的 x 和 y:

```
def objective(trial):
    x = trial.suggest_float("x", -1.0, 1.0)
    y = trial.suggest_int("y", -5, 5)
    return x + y


objective(FixedTrial({"x": 1.0, "y": -1}))  # 0.0
objective(FixedTrial({"x": -1.0, "y": -4}))  # -5.0
```

如果使用*FixedTrial* 的话，你也可以用如下方式写单元测试:

```
# A test function of pytest
def test_objective():
    assert 1.0 == objective(FixedTrial({"x": 1.0, "y": 0}))
    assert -1.0 == objective(FixedTrial({"x": 0.0, "y": -1}))
    assert 0.0 == objective(FixedTrial({"x": -1.0, "y": 1}))
```

6.4.13 在优化时, 我该如何避免耗尽内存 (running out of memory, OOM)?

如果内存使用量随着你运行更多的 trial 而增长，请尝试定期运行垃圾回收器。可在调用*optimize()* 时指定 gc_after_trial 为 True 或在回调函数中调用 gc.collect().

```
def objective(trial):
    x = trial.suggest_float("x", -1.0, 1.0)
    y = trial.suggest_int("y", -5, 5)
    return x + y


study = optuna.create_study()
study.optimize(objective, n_trials=10, gc_after_trial=True)

# `gc_after_trial=True` is more or less identical to the following.
study.optimize(objective, n_trials=10, callbacks=[lambda study, trial: gc.collect()])
```

运行垃圾回收器是有性能损失的，取决于你的目标函数运行的速度，这种损失可能是不能忽略的。因此，gc_after_trial 在默认情况下是 False. 注意，上面这个例子类似于在目标函数内运行垃圾回收器，不同之处在于 gc.collect() 哪怕在出现包括*TrialPruned* 这样的错误时也会被调用。

备注: *ChainerMNStudy* 目前并不提供 gc_after_trial 和用于*optimize()* 的回调接口。在使用该类时，你只能在目标函数内部调用垃圾回收器。

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python 模块索引

o

索引

符号

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

X

模块