Train In Data

Optuna

Main Functions

Optuna – Main setup

```
import optuna

def objective(trial):

    rf_n_estimators = trial.suggest_int("rf_n_estimators", 100, 1000)
    rf_max_depth = trial.suggest_int("rf_max_depth", 1, 4)

    model = RandomForestClassifier(
        max_depth=rf_max_depth, n_estimators=rf_n_estimators
    )

    score = cross_val_score(model, X_train, y_train, cv=3)
    accuracy = score.mean()
    return accuracy

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

Optuna – Main setup

```
import optuna

def objective(trial):

    rf_n_estimators = trial.suggest_int("rf_n_estimators", 100, 1000)
    rf_max_depth = trial.suggest_int("rf_max_depth", 1, 4)

    model = RandomForestClassifier(
        max_depth=rf_max_depth, n_estimators=rf_n_estimators
    )

    score = cross_val_score(model, X_train, y_train, cv=3)
    accuracy = score.mean()
    return accuracy

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

Optuna – create_study

optuna.create_study() - Parameters

- **storage**: url to a database
- **sampler**: the hyperparameter search algorithm (defo: tpe)
- **pruner**: the algorithm to prune unsuccessful trials (defo: MedianPruner)
- **direction**: ‘minimize’ or ‘maximize’
- **study_name**: name for the study, in case saved and retrieved later

Optuna – create_study

optuna.create_study() - Parameters

- **storage**: url to a database
- **sampler**: the hyperparameter search algorithm (defo: tpe)
- **pruner**: the algorithm to prune unsuccessful trials (defo: MedianPruner)
- **direction**: ‘minimize’ or ‘maximize’
- **study_name**: name for the study, in case saved and retrieved later

Optuna - sampler

optuna.samplers

The `samplers` module defines a base class for parameter sampling as described extensively in `BaseSampler`. The remaining classes in this module represent child classes, deriving from `BaseSampler`, which implement different sampling strategies.

`optuna.samplers.BaseSampler`	Base class for samplers.
`optuna.samplers.GridSampler`	Sampler using grid search.
`optuna.samplers.RandomSampler`	Sampler using random sampling.
`optuna.samplers.TPESampler`	Sampler using TPE (Tree-structured Parzen Estimator) algorithm.
`optuna.samplers.CmaEsSampler`	A sampler using cmaes as the backend.
`optuna.samplers.PartialFixedSampler`	Sampler with partially fixed parameters.
`optuna.samplers.NSGAIISampler`	Multi-objective sampler using the NSGA-II algorithm.
`optuna.samplers.MOTPESampler`	Multi-objective sampler using the MOTPE algorithm.
`optuna.samplers.IntersectionSearchSpace`	A class to calculate the intersection search space of a `BaseStudy`.
`optuna.samplers.intersection_search_space`	Return the intersection search space of the `BaseStudy`.

Optuna - pruner

optuna.pruners

The `pruners` module defines a `BasePruner` class characterized by an abstract `prune()` method, which, for a given trial and its associated study, returns a boolean value representing whether the trial should be pruned. This determination is made based on stored intermediate values of the objective function, as previously reported for the trial using `optuna.trial.Trial.report()`. The remaining classes in this module represent child classes, inheriting from `BasePruner`, which implement different pruning strategies.

`optuna.pruners.BasePruner`	Base class for pruners.
`optuna.pruners.MedianPruner`	Pruner using the median stopping rule.
`optuna.pruners.NopPruner`	Pruner which never prunes trials.
`optuna.pruners.PercentilePruner`	Pruner to keep the specified percentile of the trials.
`optuna.pruners.SuccessiveHalvingPruner`	Pruner using Asynchronous Successive Halving Algorithm.
`optuna.pruners.HyperbandPruner`	Pruner using Hyperband.
`optuna.pruners.ThresholdPruner`	Pruner to detect outlying metrics of the trials.

Optuna – trials

optuna.trial.Trials()

Used under the hood by optuna.study.Study.optimize()

We only need the below for the objective function

`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.

Train In Data

THANK YOU

www.trainindata.com