Hyperparameter Tuning

Álvaro Román Gómez

5/29/23

Table of contents

```
import optuna
import pandas as pd
import numpy as np
import pickle

from xgboost import XGBClassifier
from sklearn.model_selection import cross_val_score
from sklearn.metrics import roc_auc_score
```

```
input_path = "../data/processed/"
output_path = "../data/processed/"

training_path = input_path + "train_data/"
test_path = input_path + "test_data/"


results_path = "../models/results/"
figures_path = "../../Memoria/figures/"
```

```
# FILES
results_file = "results_table.csv"
```

```
dataset_name = "maccs_keys"
```

```
# LOAD DATA
train = pd.read_csv(training_path + dataset_name + "_training.csv")
test = pd.read_csv(test_path + dataset_name + "_test.csv")

X_train = train.drop(columns=["activity"])
Y_train = train["activity"]

X_test = test.drop(columns=["activity"])
Y_test = test["activity"]
```

```
def objective(trial):
    n_estimators = trial.suggest_int("n_estimators", 100, 1000, step=50)
    max_depth = trial.suggest_int("max_depth", 3, 50, step=1)
    learning_rate = trial.suggest_float("learning_rate", 0.01, 0.1, step=0.01)
    subsample = trial.suggest_float("subsample", 0.5, 1.0, step=0.1)
    colsample_bytree = trial.suggest_float("colsample_bytree", 0.5, 1.0, step=0.1)
    gamma = trial.suggest_float("gamma", 0.0, 2.0, step=0.1)
    min_child_weight = trial.suggest_int("min_child_weight", 1, 20, step=1)
    reg_alpha = trial.suggest_float("reg_alpha", 0.0, 2.0, step=0.1)
    reg_lambda = trial.suggest_float("reg_lambda", 0.0, 2.0, step=0.1)

    model = XGBClassifier(
        n_estimators=n_estimators,
        max_depth=max_depth,
        learning_rate=learning_rate,
        subsample=subsample,
        colsample_bytree=colsample_bytree,
        gamma=gamma,
        min_child_weight=min_child_weight,
        reg_alpha=reg_alpha,
        reg_lambda=reg_lambda,
        random_state=42,
        n_jobs=-1,
    )

    score = cross_val_score(model, X_train, Y_train, scoring="roc_auc", cv=5, n_jobs=-1)
    auc = score.mean()

    return auc
```

```python
# OPTUNA STUDY
study = optuna.create_study(direction="maximize")
study.optimize(objective, n_trials=300)
```

```python
# BEST OPTUNA MODEL
best_model = study.best_trial.params
```

```python
# EVALUATE BEST MODEL ON TEST SET
model = XGBClassifier(
    n_estimators=best_model["n_estimators"],
    max_depth=best_model["max_depth"],
    learning_rate=best_model["learning_rate"],
    subsample=best_model["subsample"],
    colsample_bytree=best_model["colsample_bytree"],
    gamma=best_model["gamma"],
    min_child_weight=best_model["min_child_weight"],
    reg_alpha=best_model["reg_alpha"],
    reg_lambda=best_model["reg_lambda"],
    random_state=42,
    n_jobs=-1,
)

model.fit(X_train, Y_train)

Y_pred = model.predict(X_test)
Y_pred_proba = model.predict_proba(X_test)[:, 1]

auc = roc_auc_score(Y_test, Y_pred_proba)

print("AUC: ", auc)
```

```
AUC:  0.8380893300248138
```

```python
# EVALUTE BEST MODEL
score = cross_val_score(model, X_train, Y_train, scoring="roc_auc", cv=10, n_jobs=-1)
auc = score.mean()
```

```python
# SAVE MODEL FOR DEPLOYMENT
pickle.dump(model, open("../models/xgboost_model.pkl", "wb"))
```