VLG OpenProject 2024

AutoML Automated Hyperparameter Optimisation

Bhawana Yadav | 21112036 IIT Roorkee
b_yadav@ch.iitr.ac.in

INTRODUCTION

In this project, I developed an automated hyperparameter optimization (HPO) system using AutoML techniques to enhance the performance of machine learning models on various datasets. The architecture integrates several machine learning models and employs efficient HPOtechniques, specifically Bayesian Optimization, to identify the best hyperparameter configurations. My approach addresses the problem of manual hyperparameter tuning, which is often time-consuming and suboptimal.

Problem Statement

The quality of performance of a Machine Learning model heavily depends on its hyperparameter settings. Given a dataset and a task, the choice of the machine learning (ML) model and its hyperparameters is typically performed manually.

My project aims to develop an automated hyperparameter optimization (HPO) system using AutoML techniques that can efficiently identify the best hyperparameter configuration for a given machine learning model and dataset.

Architecture

The system architecture is designed to handle different data types and seamlessly integrate with various machine learning models. The core components of my architecture include:

1. Data Preprocessing Module: This module is responsible for preparing datasets for training and evaluation. It handles tasks such as handling missing values, encoding categorical features, and scaling numerical features to ensure the data is suitable for model training.

2. Model Selection Module: This module incorporates various machine learning models including SVM, Random Forest, Gradient Boosting Classifier, and LightGBM. It provides a flexible interface to select and train different models based on the given dataset.

3. Bayesian Optimization Module: This module efficiently searches for optimal hyperparameter configurations using Bayesian Optimization. It defines the search space for hyperparameters, evaluates model performance for different configurations, and identifies the best set of hyperparameters.

4. Evaluation Module: This module assesses model performance using metrics like ROC AUCand cross-validation.

It provides a comprehensive evaluation of the model's performance, allowing for comparison between different models and hyperparameter configurations.

Specifications

● ModelsUsed:
○ Support Vector Machine (SVM)
○ RandomForest
○ Gradient Boosting Classifier
○ LightGBM

● Datasets:
○ Breast Cancer Dataset: A standard dataset from scikit-learn used for binary classification tasks.
○ Kaggle Dataset: A dataset from a Kaggle competition used for binary classification tasks related to loan default prediction.

● Evaluation Metrics:
○ ROCAUC:Receiver Operating Characteristic- Area Under Curve, a performance measurement for classification problems.
○ Learning Rate Distribution Comparison: A visual comparison of learning rates across different models to understand the effect of hyperparameter optimization.

METHODOLOGY

Data Preprocessing

```python
import numpy as np
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import cross_val_score, KFold
from hyperopt import fmin, tpe, hp, Trials, STATUS_OK
from scipy.stats import norm
from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.gaussian_process.kernels import RBF
from sklearn.model_selection import cross_val_score, train_test_split
import matplotlib.pyplot as plt

class BayesianOptimizer:
    def __init__(self, param_bounds, n_iter=50, init_samples=5):
        self.param_bounds = param_bounds
        self.n_iter = n_iter
        self.init_samples = init_samples
        self.kernel = RBF()
        self.gpr = GaussianProcessRegressor(kernel=self.kernel, alpha=1e-6, normalize_y=True, n_restarts_optimizer=10)
        self.samples = []
        self.scores = []
```

```python
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt

features = pd.read_csv('application_train.csv')

features = features.sample(n=16000, random_state=42)

features = features.select_dtypes('number')

labels = np.array(features['TARGET'].astype(np.int32))
features = features.drop(columns=['TARGET', 'SK_ID_CURR'])

train_features, test_features, train_labels, test_labels = train_test_split(features, labels, test_size=6000, random_state=42)

print('Train shape: ', train_features.shape)
print('Test shape: ', test_features.shape)

scaler = StandardScaler()
train_features = scaler.fit_transform(train_features)
test_features = scaler.transform(test_features)
```

IMPLEMENTATION

The core of my HPO system is the Bayesian Optimizer, designed to find the best hyperparameter configurations efficiently. Below is the detailed implementation and explanation of the Bayesian Optimizer

OPTIMIZATION

The BayesianOptimizer class implements a Bayesian optimization algorithm to find optimal hyperparameters for machine learning models efficiently.

OBJECTIVE FUNCTION

The objective function is designed to optimize the performance of a Random Forest Classifier on the Breast Cancer dataset. It takes hyperparameters as input and returns the negative mean ROCAUCscore obtained from cross-validation.

HYPERPARAMETER OPTIMIZATION

The Bayesian Optimizer is configured and executed to find the best hyperparameters for the Random Forest Classifier After the optimization process, the results are displayed, and a convergence plot is generated to visualize the improvement in the model's performance over iterations. The best hyperparameters found by the optimizer are highlighted.

CONCLUSION

The successful implementation of this automated HPO system validates the hypothesis that AutoML techniques can substantially enhance machine learning model performance. By achieving higher ROC AUC scores through efficient hyperparameter optimization, I demonstrated the system's value in improving predictive accuracy and overall model quality. This project not only addresses the problem of manual hyperparameter tuning but also sets a foundation for future research and development in automated machine learning. The results obtained affirm the project's success in meeting its objectives, paving the way for more advanced and automated approaches in machine learning workflows.

RESOURCES

Automated Model Tuning (kaggle.com)

AutoML | Hyperparameter Optimization

windmaple/awesome-AutoML: Curating a list of AutoML-related research, tools, projects and other resources (github.com)