지능화 캡스톤 프로젝트

프로젝트 주제 발표

하이퍼파라미터 조정

2024. 6. 3.

충북대학교 산업인공지능학과

* * *

INDUSTRIAL AI RESEARCH CENTER

목차

본 과제는 하이퍼파라미터 조정에 대한 개념을 아래와 같은 순서로 정리하였습니다.

1. 모델 성능 개선 방법 설명
2. 모델 성능 개선 방법 중 하이퍼파라미터 개념 설명
3. 하이퍼파라미터 조정 방법 설명
4. 하이퍼파라미터 조정 방법 중 Optuna 소개
5. Optuna 사용 예시 소개
6. 시각화 도구로 Optuna-Dashboard Tool 소개
7. 결론

1. 모델 성능 개선 방법 설명

모델 성능 개선 방법

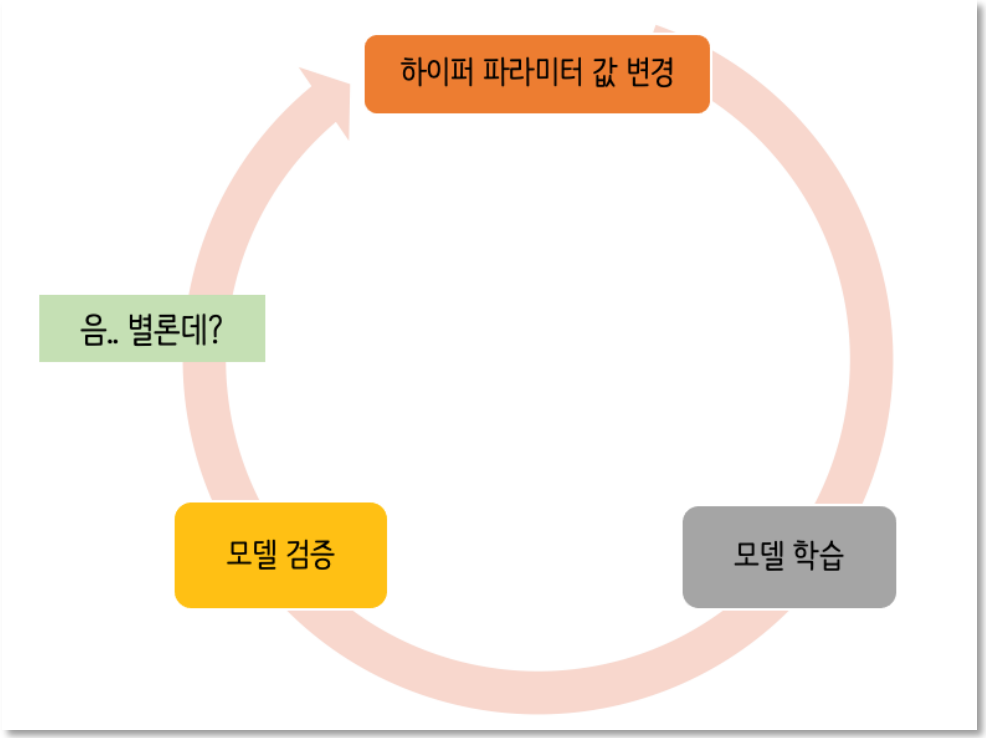
- 다양한 방법을 통해 머신러닝 모델의 성능을 개선할 수 있습니다.

모델 성능 개선 방법

- 데이터 전처리: 데이터 정제, 특성 선택 및 생성, 데이터증강

- 모델 선택: 적합한 알고리즘 선택

- 하이퍼파라미터 조정: 모델 성능 최적화



2. 모델 성능 개선 방법 중 하이퍼파라미터 개념 설명

하이퍼파라미터란?

- *모델 학습 전 설정해야 하는 값*
- *모델 학습 과정에서 직접 학습되지 않는 변수*
- *모델의 일반화 성능에 큰 영향*

- *예시: 학습률, 결정 트리의 깊이 등*

	Hyperparameter	Parameter
설명	초매개변수 모델 학습 과정에 반영되는 값 학습 시작 전에 미리 조정	매개변수 모델 내부에서 결정되는 변수 데이터로부터 학습 또는 예측되는 값
예시	학습률 손실 함수 배치 사이즈	정규분포의 평균, 표준편차 선형 회귀 계수 가중치, 편향
직접 조정 가능	O	X

파라미터와 하이퍼파라미터의 차이

- *파라미터:*
  - *학습 중 모델에 의해 조정되는 값 (예: 가중치)*
  - *모델이 데이터로부터 직접 학습하는 가중치*

- *하이퍼파라미터:*
  - *학습 전 설정되는 값 (예: 학습률)*
  - *학습률, 규제 강도 등 사전에 지정해야 하는 값*
  - *모델 설계 시 사용자가 직접 설정해야 하는 값 (예: 학습률, 배치 크기)*

3. 하이퍼파라미터 조정 방법 설명

하이퍼파라미터 조정의 중요성

- *성능 향상: 최적의 하이퍼파라미터를 찾아 모델 성능 극대화*
- *모델 일반화: 과적합 방지*

하이퍼파라미터 조정 방법

- *그리드 서치 (Grid Search)*
- *랜덤 서치 (Random Search)*
- *베이지안 최적화 (Bayesian Optimization)*
- *Optuna 등 자동화 도구*

특성/도구	GridSearchCV	RandomSearchCV	Optuna
검색 방식	격자 검색	랜덤 검색	트리 기반의 구조화된 파라미터 최적화
최적화 속도	느림	중간	빠름
자원 활용 효율	낮음	중간	높음
프루닝 기능	없음	없음	있음
사용자 정의 최적화	제한적	유연함	매우 유연함
시각화 도구	기본적인 시각화 지원	기본적인 시각화 지원	시각화 도구 제공

4. 하이퍼파라미터 조정 방법 중 Optuna 소개

Optuna 소개

- *정의 :*
  - *ㅇ 자동화된 하이퍼파라미터 최적화 소프트웨어*
  - *ㅇ 정의적 프로그래밍 방식의 하이퍼파라미터 최적화 프레임워크*
  - *** optuna는 2019년 처음 공개된, 확률적 최적화 알고리즘을 사용하여 하이퍼파라미터를 조정하는 도구입니다.*

- *특징 :*
  - *ㅇ 효율적인 샘플링, 동적 그래프 구조*
  - *ㅇ 다양한 최적화 알고리즘 제공 (TPE, CMA-ES 등)*

Optuna의 장점

- *사용 용이성: 간단한 코드로 설정 가능*
- *효율성: 빠른 최적화, 다양한 시각화 도구 제공*

*** 자동 프루닝 기능 : 비효율적인 트라이얼을 조기에 중단시킬 수 있는 자동 프루닝 기능을 제공하여, 불필요한 계산 자원의 낭비를 줄이고, 전체 최적화 과정의 속도를 높일 수 있다.*

*** 병렬화 용이 : 분산 환경에서의 병렬 실행을 쉽게 지원하여, 대규모 실험을 빠르게 처리할 수 있고, 큰 데이터 세트나 복잡한 모델에서 매우 유용하다.*

5. Optuna 사용 예시 소개

예시자료 : Project #1 조별 과제 자료 이용

- 과제시 작성된 코드

```python
import optuna
import torch
import torch.optim as optim
from torch.utils.data import DataLoader, Subset, random_split
from torchvision import transforms, datasets
from ClassificationModelGenerator import CNN, device
import matplotlib.pyplot as plt
import random

# 모델의 최종 성능을 기록할 변수
train_losses, val_losses, accuracies = [], [], []

def objective(trial):...

def run_hyperparameter_tuning():
    study = optuna.create_study(direction='minimize')
    study.optimize(objective, n_trials=10)  # Reduced number of trials for quicker testing

    print('Best hyperparameters:', study.best_params)
    print('Best validation loss:', study.best_value)

    plt.figure(figsize=(10, 5))
    plt.subplot(1, 2, 1)
    plt.plot(val_losses, label='Validation')
    plt.title('Loss Over Trials')
    plt.legend()
    plt.subplot(1, 2, 2)
    plt.plot(accuracies, label='Accuracy')
    plt.title('Accuracy Over Trials')
    plt.legend()

    plt.show()


run_hyperparameter_tuning()
```

Import optuna
Objectives 정의
Hyperparameter 범위 설정
Optimizer 정의
Sampler 정의
best hyperparmeters 출력

```python
# 하이퍼파라미터 범위 설정
batch_size = trial.suggest_int('batch_size', 32, 128, step=32)
learning_rate = trial.suggest_float('learning_rate', 1e-4, 1e-2, log=True)
num_epochs = trial.suggest_int('num_epochs', 10, 30, step=5)
```

```python
optimizer = optim.Adam(model.parameters(), lr=learning_rate)
```

```python
for label, samples in class_samples.items():
    random.shuffle(samples)

    n_samples = len(samples)
    n_train = int(0.65 * n_samples)
    n_val = int(0.2 * n_samples)
```

5. Optuna 사용 예시 소개 (계속)

Best hyperparameters 출력 예시

```
[I 2024-04-22 18:33:13,921] A new study created in memory with name: no-name-c95eed9b-9985-40e9-8bc9-27e9ad65af40
[I 2024-04-22 18:54:37,377] Trial 0 finished with value: 0.15635016674621105 and parameters: {'batch_size': 96,
'learning_rate': 0.007296685239288039, 'num_epochs': 10}. Best is trial 0 with value: 0.15635016674621105.
[I 2024-04-22 19:27:24,923] Trial 1 finished with value: 0.1789645908506924 and parameters: {'batch_size': 128,
'learning_rate': 0.006542512440915665, 'num_epochs': 15}. Best is trial 0 with value: 0.15635016674621105.
[I 2024-04-22 20:19:39,757] Trial 2 finished with value: 0.12607546104488623 and parameters: {'batch_size': 32,
'learning_rate': 0.0036450376204547243, 'num_epochs': 25}. Best is trial 2 with value: 0.12607546104488623.
[I 2024-04-22 20:50:50,200] Trial 3 finished with value: 0.09730753199009631 and parameters: {'batch_size': 64,
'learning_rate': 0.0004173542504720025, 'num_epochs': 15}. Best is trial 3 with value: 0.09730753199009631.
[I 2024-04-23 08:27:30,565] Trial 4 finished with value: 0.08642735462363306 and parameters: {'batch_size': 64,
'learning_rate': 0.00010933730778047905, 'num_epochs': 25}. Best is trial 4 with value: 0.08642735462363306.
[I 2024-04-23 08:59:01,916] Trial 5 finished with value: 0.10037415709934182 and parameters: {'batch_size': 64,
'learning_rate': 0.0012558721739461477, 'num_epochs': 15}. Best is trial 4 with value: 0.08642735462363306.
[I 2024-04-23 09:50:48,799] Trial 6 finished with value: 0.08576783714832636 and parameters: {'batch_size': 64,
'learning_rate': 0.0007716318302683345, 'num_epochs': 25}. Best is trial 6 with value: 0.08576783714832636.
[I 2024-04-23 10:12:27,244] Trial 7 finished with value: 0.854200650430912 and parameters: {'batch_size': 128,
'learning_rate': 0.00045501761820191365, 'num_epochs': 10}. Best is trial 6 with value: 0.08576783714832636.
[I 2024-04-23 10:53:57,250] Trial 8 finished with value: 0.1212177780364827 and parameters: {'batch_size': 96,
'learning_rate': 0.004541440111141002, 'num_epochs': 20}. Best is trial 6 with value: 0.08576783714832636.
[I 2024-04-23 11:25:51,464] Trial 9 finished with value: 0.14051162052862362 and parameters: {'batch_size': 64,
'learning_rate': 0.00445798104513055, 'num_epochs': 15}. Best is trial 6 with value: 0.08576783714832636.
Best hyperparameters: {'batch_size': 64, 'learning_rate': 0.0007716318302683345, 'num_epochs': 25}
Best validation loss: 0.08576783714832636
```

5. Optuna 사용 예시 소개 (계속)

하이퍼 파라미터 튜닝 결과

- 튜닝 전과 후를 비교해 보았을 때, 전반적으로 모델의 최종 성능이 향상됨

- 튜닝 전

```
모델의 최종 성능
Test Loss: 0.0753
Test Accuracy: 98.13%
Test Precision: 98.13%
Test Recall: 98.13%
Test F1 Score: 98.13%
```

- 튜닝 후

```
모델의 최종 성능
Test Loss: 0.0765
Test Accuracy: 98.16%
Test Precision: 98.15%
Test Recall: 98.16%
Test F1 Score: 98.15%
```

6. 시각화 도구로 Optuna-Dashboard Tool 소개

Optuna-Dashboard 소개

- 정의: 하이퍼파라미터 최적화 결과를 시각화하는 도구

```python
import optuna

if __name__ == "__main__":
    study_name = "quadratic-simple"
    study = optuna.create_study(
        storage=f"sqlite:///{study_name}.db",  # Specify the storage URL here.
        study_name=study_name
    )
    study.optimize(objective, n_trials=100)
    print(f"Best value: {study.best_value} (params: {study.best_params})")
```

```
pip install optuna-dashboard
optuna-dashboard sqlite:///quadratic-simple.db
```

6. 시각화 도구로 Optuna-Dashboard Tool 소개 (계속)

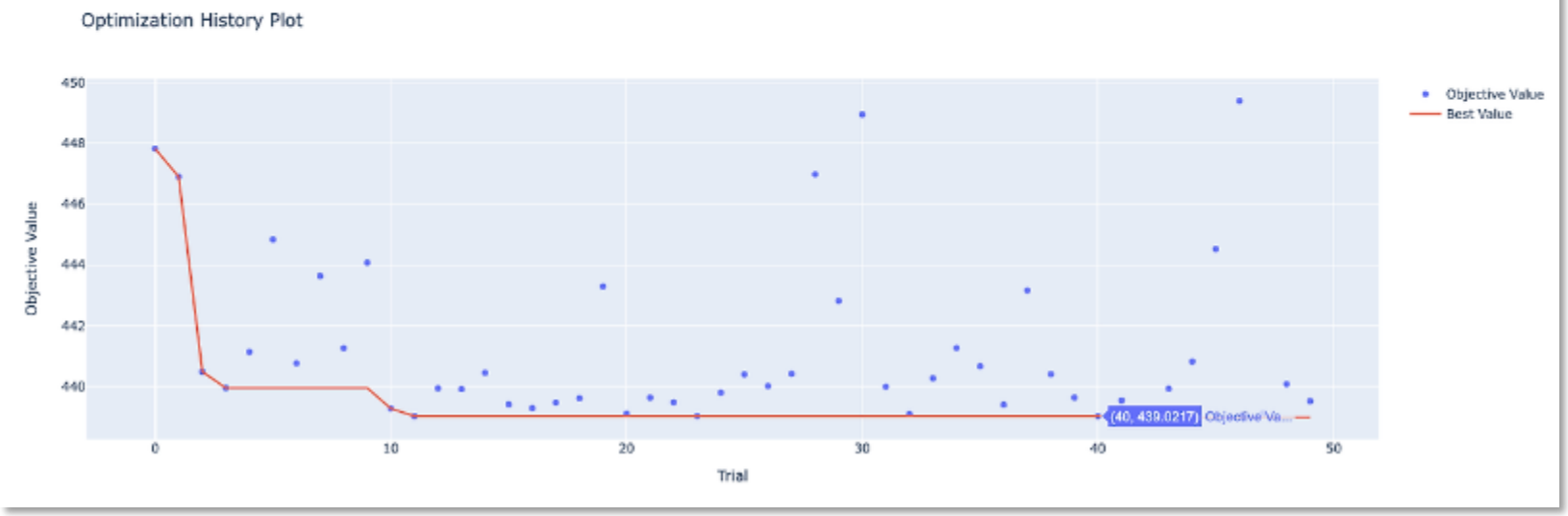
Optimization History Plot

- 몇 번째의 나의 하이퍼파라미터 값이 최저였는지, 최대였는지 확인할 수 있음.

- 여러 모델을 돌렸을 때 어떤 모델인지, 어떤 값인지 오름차순, 내림차순 확인할 수 있음.

```
# Optuna 시각화
from optuna.visualization import plot_optimization_history, plot_param_importances

# 최적화 과정 시각화
plot_optimization_history(study)
```



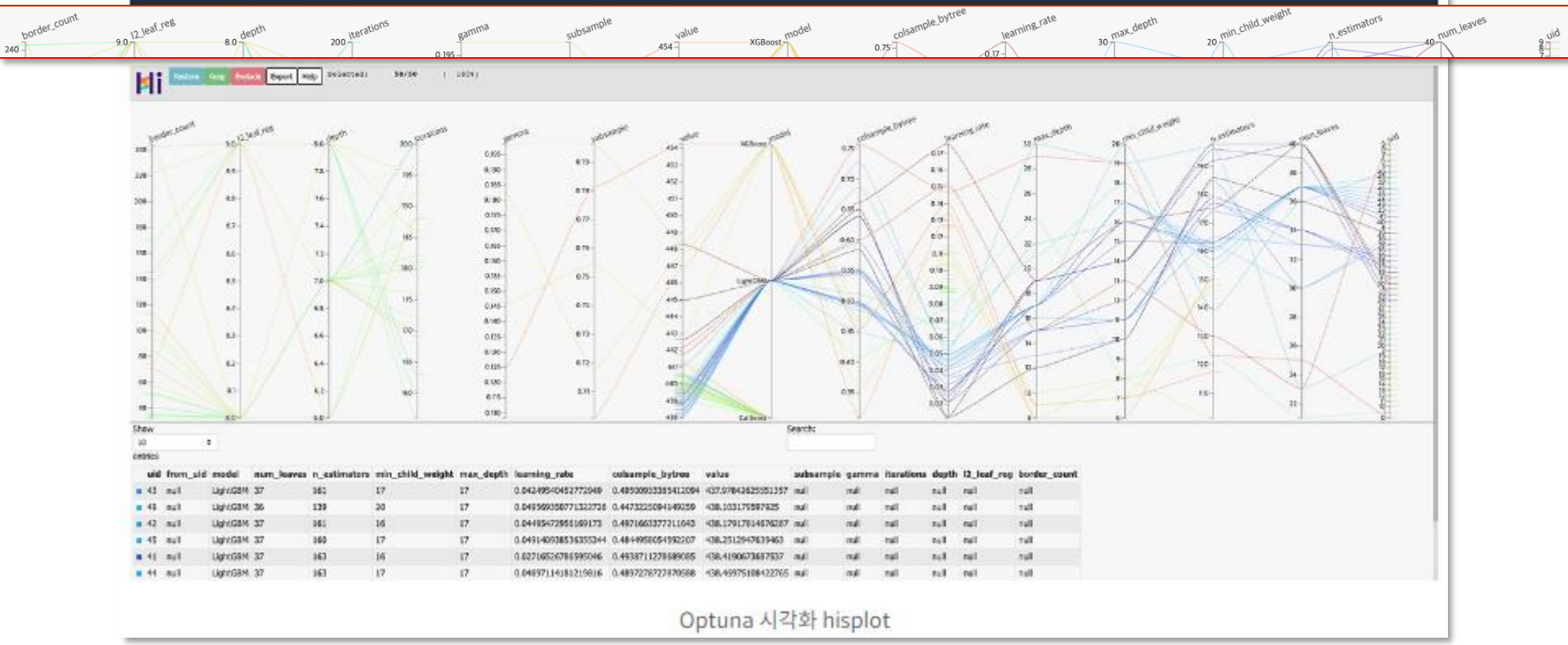
6. 시각화 도구로 Optuna-Dashboard Tool 소개 (계속)

Hiplot

- *여러 개의 모델을 돌렸을 때 어떤 값이 최저, 최대인지 엑셀처럼 내림차순, 오름차순으로 확인할 수 있음.*

```
import hiplot as hip

# Optuna 스터디 결과를 hiplot으로 변환
data = [dict(trial.params, value=trial.value) for trial in study.trials]
hip.Experiment.from_iterable(data).display()
```



Optuna 시각화 hisplot

7. 결론

결과 요약

- Optuna는
- 간편한 구현
- 다양한 알고리즘 지원
- 시각화 도구
- 분산 최적화

-> 최적의 하이퍼파라미터 조합을 자동화 하여 찾을 수 있음.

-> 최적의 조합을 통해 튜닝 후 학습과정이 안정적으로 변화하고 성능이 개선됨.

-> Optuna-dashboard tool 사용으로 시각화 자료 생성 실습을 진행해 보고 싶습니다.

감사합니다