1. Auto-Sklearn

1 이론적 배경

관련 논문

Auto-Sklearn을 소개한 논문을 참고하여 Auto-Sklearn을 구성하는 이론적 배경에 대해 알아보겠습니다.

Auto-Sklearn 2.0: Hands-free AutoML via Meta-Learning

Matthias Feurer[1] FEURERM@CS.UNI-FREIBURG.DE
Katharina Eggensperger[1] EGGENSPK@CS.UNI-FREIBURG.DE
Stefan Falkner[2] STEFAN.FALKNER@DE.BOSCH.COM
Marius Lindauer[3] LINDAUER@TNT.UNI-HANNOVER.DE
Frank Hutter[1,2] FH@CS.UNI-FREIBURG.DE
[1] *Department of Computer Science, Albert-Ludwigs-Universität Freiburg*
[2] *Bosch Center for Artificial Intelligence, Renningen, Germany*
[3] *Institute of Information Processing, Leibniz University Hannover*

Abstract

Automated Machine Learning (AutoML) supports practitioners and researchers with the tedious task of designing machine learning pipelines and has recently achieved substantial success. In this paper we introduce new AutoML approaches motivated by our winning submission to the second ChaLearn AutoML challenge. We develop *PoSH Auto-sklearn*, which enables AutoML systems to work well on large datasets under rigid time limits using a new, simple and meta-feature-free meta-learning technique and employs a successful bandit strategy for budget allocation. However, *PoSH Auto-sklearn* introduces even more ways of running AutoML and might make it harder for users to set it up correctly. Therefore, we also go one step further and study the design space of AutoML itself, proposing a solution towards truly hands-free AutoML. Together, these changes give rise to the next generation of our AutoML system, *Auto-sklearn 2.0*. We verify the improvements by these additions in a large experimental study on 39 AutoML benchmark datasets and conclude the paper by comparing to other popular AutoML frameworks and *Auto-sklearn 1.0*, reducing the relative error by up to a factor of 4.5, and yielding a performance in 10 minutes that is substantially better than what *Auto-sklearn 1.0* achieves within an hour.

Keywords: Automated machine learning, hyperparameter optimization, meta-learning, automated AutoML, benchmark

Feurer, M., Eggensperger, K., Falkner, S., Lindauer, M., and Hutter, F. (2020). Auto-sklearn 2.0: Hands-free automl via meta-learning. arXiv preprint arXiv:2007.04074.

문제 정의

Auto-Sklearn에서 고려하는 문제는 모델 선택 및 하이퍼파라미터 튜닝 문제입니다.

모델 및 하이퍼 파라미터 선택 문제

$$\mathcal{M}^*, \boldsymbol{\lambda}^* = \underset{\mathcal{M} \in \Omega, \boldsymbol{\lambda} \in \Lambda}{\operatorname{argmax}} P(\mathcal{M}, \boldsymbol{\lambda}; D)$$

- $\mathcal{M}$: 모델
- λ: 하이퍼파라미터
- Ω: 모델 탐색 공간
- Λ: 하이퍼파라미터 탐색 공간

시간 제약 추가

$$\sum t_{\mathcal{M}, \boldsymbol{\lambda}; D} < T$$

- $t_{\mathcal{M},\lambda;D}$: 데이터 D로 하이퍼파라미터가 λ인 모델 $\mathcal{M}$을 평가하는 데 소요되는 시간
- T: 사용자가 설정한 최대 탐색 시간
- 즉, 이 제약식은 사용자가 설정한 시간 내에 탐색을 완료해야 한다는 의미임

문제 정의: 정책 개발

데이터에 따라 적절한 정책(policy)이 다를 수 있습니다. 여기서 정책이란 탐색 전략 및 탐색 공간을 의미합니다. 예를 들어, 어떤 데이터는 랜덤 서치를 이용해 랜덤 포레스트를 중심으로 탐색하는 것이 적절하며, 다른 데이터는 그리드 서치를 이용해 신경망과 회귀 모델을 중심으로 탐색하는 것이 적절할 수 있습니다.

메타 학습을 이용한 정책 개발

$$P(\pi|\mathbb{D}_{meta}) = \frac{1}{|\mathbb{D}_{meta}|} \sum_{D_i \in \mathbb{D}_{meta}} P(\pi|D_i)$$

- π: 정책
- $\mathbb{D}_{meta}$: 메타 데이터 집합
- $P(\pi, D_i)$: 정책 π를 데이터 D_i에 적용해서 찾은 모델 및 하이퍼파라미터의 성능

탐색 공간 정의

모델 선택과 하이퍼파라미터 튜닝 문제에서 가장 중요한 작업은 탐색 공간을 정의하는 것입니다. 좋은 성능의 모델이 학습될 가능성이 적은 모델과 하이퍼파라미터는 탐색 공간에서 배제하는 등 탐색 공간을 잘 정의하면 최적해를 효율적으로 찾을 수 있습니다.

분류 모델	에이다부스트	베르누이 나이브베이즈	결정 나무	• • •	신경망	16개 모델
회귀 모델	에이다부스트	결정 나무	가우시안 프로세스	• • •	서포트 벡터 회귀	12개 모델
데이터 전처리	주성분 분석	다항 특징 추가	특징 선택	• • •	스케일링	

활용 방법론

Auto-Sklearn은 대체 모델이 랜덤 포레스트인 베이지안 최적화를 사용해 해를 탐색함

- 대체 모델로 가우시안 프로세스 대신 랜덤 포레스트를 사용하는 이유는 가우시안 프로세스는 혼합형 변수로 구성된 고차원의 공간에 부적합하기 때문
- 랜덤 포레스트는 혼합형 변수에 적합하다는 장점이 있지만, 적확한 획득 함수를 구성하기 어렵다는 문제가 있음

메타 모델을 사용해 베이지안 최적화에서 사용할 초기해를 만듦

- 이 메타 모델은 209개의 분류 데이터와 111개의 회귀 데이터 로 학습한 k-최근접 이웃 모델임
- 메타 특징으로는 샘플 수에 로그를 취한 값, 클래스 개수, 특징 개수, 특징 개수에 로그를 취한 값, 결측이 있는 샘플 수, 범주형 특징 개수 등을 사용함

베이지안 최적화를 이용해 해를 탐색한 뒤, 앙상블 선택(ensemble selection)을 사용해 여러 모델을 종합

- 앙상블 선택은 지금까지 학습한 모든 모델에 대해 성능이 가장 우수한 모델부터 차례대로 앙상블에 투입하여 최고 성능의 앙상블을 찾음

1. Auto-Sklearn

2 설치 및 개발 환경

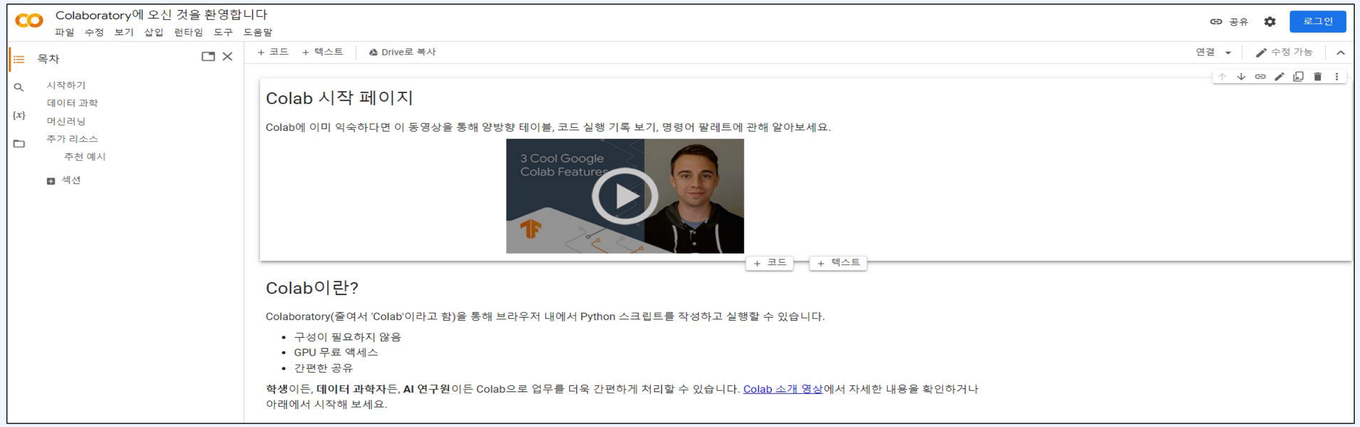
코랩이 필요한 이유

Auto-Sklearn은 리눅스에 설치해서 활용해야 합니다. 윈도우 머신은 Auto-Sklearn이 사용하는 resource 모듈을 쓸 수 없고, 맥 운영 체제에서는 의존성 문제가 있기 때문입니다. 그러나 Auto-Sklearn을 경험하는 것으로 충분한 실습에서 리눅스를 설치하긴 부담스럽습니다.

- 구글에서 제공하는 클라우드 기반의 주피터 노트북 개발 환경
- 별도의 파이썬 설치가 필요 없고 판다스, 사이킷런, 케라스(Keras), 텐서플로(TensorFlow) 등이 설치돼있음
- GPU를 무료로 사용할 수 있음

코랩 사용 방법

코랩 접속: https://colab.research.google.com/?hl=ko



파일 > 새 노트

Auto-Sklearn 설치

코랩에는 Auto-Sklearn이 설치돼 있지 않으므로, 다음과 같은 코드를 실행하여 Auto-Sklearn을 설치하겠습니다

Auto-Sklearn 설치

```
1  !sudo apt-get install build-essential swig
2  !curl https://raw.githubusercontent.com/automl/auto-sklearn/master/requirements.txt | xargs -n 1 -L 1 pip install
3  !pip install auto-sklearn
```

scipy 업데이트 (필요시)

```
1  !pip install --upgrade scipy
```

이처럼 주피터 노트북에서 !명령어 형태로 모듈 및 패키지를 설치할 수 있으며, 설치 뒤에는 반드시 런타임 혹은 커널을 다시 시작해야 함

1. Auto-Sklearn

3 자동화 실습

분류 모델 학습 자동화
: 예제 데이터 불러오기

분류 모델 학습 자동화를 실습할 예제 데이터를 불러옵니다.

예제 데이터 불러오기

```
1  from sklearn.datasets import load_breast_cancer
2  from sklearn.model_selection import train_test_split
3  X, y = load_breast_cancer(return_X_y = True)
4  X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 2022)
```

- **라인 3:** 사이킷런에서 제공하는 breast_cancer 데이터를 load_breast_cancer 함수를 사용해 불러옵니다.

분류 모델 학습 자동화
: AutoSklearnClassifier

분류 모델을 자동으로 학습하는 데 사용하는 클래스는 classification 모듈의 AutoSklearnClassifier입니다.

주요 인자

인자	설명
time_left_for_this_task	모델과 하이퍼파라미터를 탐색하는 시간에 대한 제약으로 기본값은 한 시간(3600)임
ensemble_size	앙상블에 포함될 모델 수로, 0으로 설정하면 앙상블 모델을 학습하지 않음
include	탐색에 반드시 포함하는 모델 및 전처리 단계를 사전 형태로 정의함
scoring_functions	각 모델을 평가할 함수 목록

인스턴스 생성

```
1  from autosklearn.classification import AutoSklearnClassifier as ASC
2  from autosklearn.metrics import *
3  automl = ASC(time_left_for_this_task = 600,
4                  scoring_functions = [accuracy, f1])
```

- **라인 1 ~ 4:** 최대 10분간 특별한 제약 없이 정확도와 F1 점수를 기준으로 모델과 하이퍼파라미터를 탐색하는 인스턴스를 다음과 같이 생성하겠습니다.
- **라인 4:** scoring_functions 인자에 autosklearn.metrics의 accuracy와 f1 함수를 입력합니다.

분류 모델 학습 자동화

AutoSklearnClassifier 클래스의 인스턴스는 다음과 같이 fit 메서드로 학습하며, 학습 결과는 leaderboard 메서드를 이용해 확인할 수 있습니다.

분류 모델 학습 자동화 수행

```
1  automl.fit(X_train, y_train)
2  automl.leaderboard()
```

model_id	rank	ensemble_weight	type	cost	duration
3	1	0.02	mlp	0.028369	2.811096
102	2	0.04	mlp	0.028369	5.028662
16	3	0.02	gradient_boosting	0.028369	1.235561
65	4	0.02	mlp	0.028369	3.809955
119	5	0.02	mlp	0.035461	1.950374
6	6	0.02	mlp	0.035461	3.595475
7	7	0.02	extra_trees	0.035461	2.652223

- rank: cost를 기준으로 매긴 순위
- ensemble_weight: 각 모델이 앙상블에서 차지하는 비율
- type: 모델 유형
- cost: k-겹 교차 검증에서 검증용 데이터로 계산한 비용
- duration: 모델을 학습하는 데 소요된 시간(초)

앙상블 모델 평가

학습한 앙상블을 평가 데이터로 평가하겠습니다.

앙상블 모델 평가

```
1  from sklearn.metrics import f1_score
2  y_pred = automl.predict(X_test)
3  score = f1_score(y_test, y_pred)
4  print(score)
```

0.9662921348314606

- 단 몇 줄의 코드로 우수한 성능의 앙상블 모델이 학습됐음을 알 수 있음

회귀 모델 학습 자동화
: 예제 데이터 불러오기

회귀 모델 학습 자동화를 실습할 예제 데이터를 불러옵니다.

예제 데이터 불러오기

```
1 from sklearn.datasets import load_boston
2 X, y = load_boston(return_X_y = True)
3 X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 2022)
```

회귀모델 학습 자동화

autosklearn.regression의 AutoSklearnRegressor를 불러와 신경망 모델 학습을 자동화해보겠습니다.

분류 모델 학습 자동화 수행

```
1  from autosklearn.regression import AutoSklearnRegressor as ASR
2  automl = ASR(time_left_for_this_task = 600,
3                scoring_functions = [mean_squared_error],
4                include = {"regressor":["mlp"]})
5  automl.fit(X_train, y_train)
6  automl.show_models()
```

- **라인 4:** include 인자를 사용해 사용하는 회귀 모델의 종류를 신경망(mlp)으로 한정했습니다.
- **라인 6**: show_models 메서드를 사용해 학습된 모델의 구성을 살펴봅니다.

```
{38: {'cost': 0.15522729460733553,
  'data_preprocessor': <autosklearn.pipeline.components.data_preprocessing.DataPreprocessorChoice at 0x7f79fb47ee50>,
  'ensemble_weight': 0.06,
  'feature_preprocessor': <autosklearn.pipeline.components.feature_preprocessing.FeaturePreprocessorChoice at 0x7f79fb6ee250>,
  'model_id': 38,
  'rank': 4,
  'regressor': <autosklearn.pipeline.components.regression.RegressorChoice at 0x7f79fb6d5710>,
  'sklearn_regressor': MLPRegressor(activation='tanh', alpha=1.3903438078091925e-07, beta_1=0.999,
         beta_2=0.9, early_stopping=True, hidden_layer_sizes=(18, 18, 18),
         learning_rate_init=0.0035247546366901105, max_iter=128,
         n_iter_no_change=32, random_state=1, verbose=0, warm_start=True)},
… 중략
```

앙상블 모델 평가

학습한 앙상블을 평가 데이터로 평가하겠습니다.

앙상블 모델 평가

```
1  from sklearn.metrics import mean_squared_error as MSE
2  y_pred = automl.predict(X_test)
3  score = MSE(y_test, y_pred)
4  print(score)
```

```
10.195320401627843
```