zero-base /

Life Changing Education

Chapter 20. GBM, XGBoost, LightGBM

GBM - Gradient Boosting Machine

GBM

- 부스팅 알고리즘은 여러 개의 약한 학습기(week learner)를 순차적으로 학습-예측하면서 잘못 예측한 데이터에 가중치를 부여해서 오류를 개선해가는 방식
- GBM은 가중치를 업데이트할 때 경사 하강법(Gradient Descent)을 이용하는 것이 큰 차이

zero-base /

HAR 데이터 읽기

```
import pandas as pd
url = 'https://raw.githubusercontent.com/PinkWink/ML_tutorial/'+\
                                    'master/dataset/HAR_dataset/features.txt'
feature_name_df = pd.read_csv(url, sep='\s+', header=None,
                              names=['column_index','column_name'])
feature_name = feature_name_df.iloc[:, 1].values.tolist()
X_train = pd.read_csv('./HAR_dataset/train/X_train.txt', sep='\s+', header=None)
X_test = pd.read_csv('./HAR_dataset/test/X_test.txt', sep='\s+', header=None)
X_train.columns = feature_name
X_test.columns = feature_name
```

- 지난시간에 각자 시도한 방법으로 해주세요.

GBM - Gradient Boosting Machine


```
y_train_url = 'https://raw.githubusercontent.com/PinkWink/ML_tutorial/' +\
                                'master/dataset/HAR_dataset/train/y_train.txt'
y_test_url = 'https://raw.githubusercontent.com/PinkWink/ML_tutorial/' +\
                                'master/dataset/HAR_dataset/test/y_test.txt'


y_train = pd.read_csv(y_train_url, sep='\s+', header=None, names=['action'])
y_test = pd.read_csv(y_test_url, sep='\s+', header=None, names=['action'])
```

zero-base /

필요 모듈을 import하고

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score
import time
import warnings

warnings.filterwarnings('ignore')
```

zero-base /

시작~

```
start_time = time.time()
gb_clf = GradientBoostingClassifier(random_state=13)
gb_clf.fit(X_train, y_train)
gb_pred = gb_clf.predict(X_test)

print('ACC : ', accuracy_score(y_test, gb_pred))
print('Fit time : ', time.time() - start_time)
```

```
ACC :  0.9385816084153377
Fit time :  522.2403657436371
```

- ACC가 93.9%, 계산시간 522초.. 와우~
- 일반적으로 GBM이 성능자체는 랜덤 포레스트보다는 좋다고 알려져 있음
- sckit-learn의 GBM은 속도가 아주 느린 것으로 알려져 있음

zero-base /

GridSearch로 조금 더 찾아보자~

```
from sklearn.model_selection import GridSearchCV

params = {
    'n_estimators' : [100, 500],
    'learning_rate' : [0.05, 0.1]
}

start_time = time.time()
grid = GridSearchCV(gb_clf, param_grid=params, cv=2, verbose=1, n_jobs=-1)
grid.fit(X_train, y_train)
print('Fit time : ', time.time() - start_time)
```

```
Fitting 2 folds for each of 4 candidates, totalling 8 fits


[Parallel(n_jobs=-1)]: Using backend LokyBackend with 16 concurrent workers.
```

zero-base /

시간이 오래 걸림

```
Fitting 2 folds for each of 4 candidates, totalling 8 fits


[Parallel(n_jobs=-1)]: Using backend LokyBackend with 16 concurrent workers.
[Parallel(n_jobs=-1)]: Done   4 out of   8 | elapsed:  5.8min remaining:  5.8
min
[Parallel(n_jobs=-1)]: Done   8 out of   8 | elapsed: 14.7min finished


Fit time :  2760.5976428985596
```

zero-base /

best 파라미터가 이렇다면~

```
grid.best_score_
```

```
0.9011153427638738
```

```
grid.best_params_
```

```
{'learning_rate': 0.05, 'n_estimators': 500}
```

zero-base /

Test 데이터에서의 성능은

```
accuracy_score(y_test, grid.best_estimator_.predict(X_test))
```

```
0.9392602646759416
```

XGBoost

zero-base /

개요

- XGBoost는 트리 기반의 앙상블 학습에서 가장 각광받는 알고리즘 중 하나
- GBM 기반의 알고리즘인데, GBM의 느린 속도를 다양한 규제를 통해 해결
- 특히 병렬 학습이 가능하도록 설계됨
- XGBoost는 반복 수행 시마다 내부적으로 학습데이터와 검증데이터를 교차검증을 수행
- 교차검증을 통해 최적화되면 반복을 중단하는 조기 중단 기능을 가지고 있음

zero-base /

install

```
~ source activate fc
(fc) ~ pip install xgboost
Collecting xgboost
  Downloading xgboost-1.0.2.tar.gz (821 kB)
     |████████████████████████████████| 821 kB 211 kB/s
Requirement already satisfied: numpy in ./opt/anaconda3/envs/fc/lib/python3.7/site-packages (from xgboost) (1.18.1)
Requirement already satisfied: scipy in ./opt/anaconda3/envs/fc/lib/python3.7/site-packages (from xgboost) (1.4.1)
Building wheels for collected packages: xgboost
  Building wheel for xgboost (setup.py) ... done
  Created wheel for xgboost: filename=xgboost-1.0.2-cp37-cp37m-macosx_10_15_x86_64.whl size=3417960 sha256=2c99751038a
fd374e324e75d7cdd2f674128725acb7505a67c4c08a7924a7c3e
  Stored in directory: /Users/pw/Library/Caches/pip/wheels/c9/1c/c9/db0188e82127a4f3c66d031147312a03e8a0676d907ecd7fbb
Successfully built xgboost
```

- pip install xgboost
  - 에러가 날 경우, conda install py-xgboost
- xgboost는 설치해야 함

zero-base /

주요 파라미터

- nthread : CPU의 실행 스레드 개수를 조정. 디폴트는 CPU의 전체 스레드를 사용하는 것
- eta : GBM 학습률
- num_boost_rounds : n_estimators와 같은 파라미터
- max_depth

zero-base /

이 정도 성능이 나온다

```
from xgboost import XGBClassifier

start_time = time.time()
xgb = XGBClassifier(n_estimators=400, learning_rate=0.1, max_depth=3)
xgb.fit(X_train.values, y_train)
print('Fit time : ', time.time() - start_time)
```

```
 Fit time :  289.5869493484497
```

```
accuracy_score(y_test, grid.best_estimator_.predict(X_test.values))
```

```
   0.9392602646759416
```

zero-base /

조기 종료 조건과 검증데이터를 지정할 수 있다

```
from xgboost import XGBClassifier

evals = [(X_test.values, y_test)]

start_time = time.time()
xgb = XGBClassifier(n_estimators=400, learning_rate=0.1, max_depth=3)
xgb.fit(X_train.values, y_train, early_stopping_rounds=10, eval_set=evals)
print('Fit time : ', time.time() - start_time)
```

```
[0]     validation_0-merror:0.17916
Will train until validation_0-merror hasn't improved in 10 rounds.
[1]     validation_0-merror:0.16288
[2]     validation_0-merror:0.15100
[3]     validation_0-merror:0.14388
[4]     validation_0-merror:0.14252
```

zero-base /

이렇게 종료된다

```
[112]   validation_0-merror:0.05972
[113]   validation_0-merror:0.05972
[114]   validation_0-merror:0.06006
[115]   validation_0-merror:0.05972
[116]   validation_0-merror:0.06006
[117]   validation_0-merror:0.06006
[118]   validation_0-merror:0.06006
[119]   validation_0-merror:0.06006
Stopping. Best iteration:
[109]   validation_0-merror:0.05735

Fit time :  127.58869504928589
```

zero-base /

성능은 비슷하다

```
accuracy_score(y_test, grid.best_estimator_.predict(X_test.values))
```

```
0.9392602646759416
```

LightGBM

zero-base /

LightGBM

- LightGBM은 XGBoost와 함께 부스팅 계열에서 가장 각광받는 알고리즘
- LGBM의 큰 장점은 속도
- 단, 적은 수의 데이터에는 어울리지 않음 (일반적으로 10000건 이상의 데이터가 필요하다고 함)
- GPU 버전도 존재함

zero-base /

install guide documents

- https://lightgbm.readthedocs.io/en/latest/Installation-Guide.html

zero-base /

install for Mac User

```
Last login: Tue Mar 31 13:23:27 on ttys001
~ brew install lightgbm
Updating Homebrew...
==> Auto-updated Homebrew!
Updated 1 tap (homebrew/core).
==> New Formulae
azcopy                      container-structure-test    llvm@9                      publish
bnfc                        dhall-yaml                  mtoc                        swift-format
ccheck                      gobo                        nef                         vpn-slice
cdk8s                       hsd                         oil
==> Updated Formulae
cmake ✔                     euler-py                    kubeseal                    powerman
```

- brew install lightgbm

```
(fc) ~ pip install lightgbm
Collecting lightgbm
  Downloading lightgbm-2.3.1-py2.py3-none-macosx_10_9_x86_64.macosx_10_10_x86_64.macosx_10_11_x86_64.macosx_10_12_x86_
64.macosx_10_13_x86_64.macosx_10_14_x86_64.macosx_10_15_x86_64.whl (679 kB)
     |████████████████████████████████| 679 kB 223 kB/s
Requirement already satisfied: scikit-learn in ./opt/anaconda3/envs/fc/lib/python3.7/site-packages (from lightgbm) (0.
22.1)
Requirement already satisfied: scipy in ./opt/anaconda3/envs/fc/lib/python3.7/site-packages (from lightgbm) (1.4.1)
Requirement already satisfied: numpy in ./opt/anaconda3/envs/fc/lib/python3.7/site-packages (from lightgbm) (1.18.1)
Requirement already satisfied: joblib>=0.11 in ./opt/anaconda3/envs/fc/lib/python3.7/site-packages (from scikit-learn-
>lightgbm) (0.14.1)
Installing collected packages: lightgbm
```

- pip install lightgbm

zero-base /

무서운 속도

```
from lightgbm import LGBMClassifier

start_time = time.time()
lgbm = LGBMClassifier(n_estimators=400)
lgbm.fit(X_train.values, y_train, early_stopping_rounds=100, eval_set=evals)
print('Fit time : ', time.time() - start_time)
```

```
[153]   valid_0's multi_logloss: 0.270435
[154]   valid_0's multi_logloss: 0.270811
[155]   valid_0's multi_logloss: 0.270844
[156]   valid_0's multi_logloss: 0.271312
[157]   valid_0's multi_logloss: 0.270996
[158]   valid_0's multi_logloss: 0.271048
[159]   valid_0's multi_logloss: 0.271304
[160]   valid_0's multi_logloss: 0.272792
[161]   valid_0's multi_logloss: 0.272478
[162]   valid_0's multi_logloss: 0.273176
[163]   valid_0's multi_logloss: 0.272896
[164]   valid_0's multi_logloss: 0.272415
Early stopping, best iteration is:
[64]    valid_0's multi_logloss: 0.229643
Fit time :  16.549540042877197
```

zero-base /

와우~

```
accuracy_score(y_test, grid.best_estimator_.predict(X_test.values))
```

```
0.9392602646759416
```