CatBoost: unbiased boosting with categorical features

- Reading the paper in terms of Ensemble Learning -

1. Introduction

Gradient boosting은 weak learner를 loss function상에서 gradient descent라는 최적화 기법으로 기울기가 가장 큰 방향으로 sub tree들을 반복적으로 추가하여 결합하는 방법으로 성능을 향상시키는 boosting 기법 중 하나입니다. XGboost, Lightgbm 등 다양한 boosting 기법을 활용한 의사결정트리 알고리즘들은 기계학습 알고리즘 중 뛰어난 성능을 발휘하고 있습니다. Catboost는 이러한 boosting 기법을 활용한 의사결정트리 알고리즘 중 범주 분류 문제에 특화된 알고리즘으로서, 범주 분류 문제에 최적화된 다양한 방법론들을 적용하였습니다. 범주 분류 문제는 예측 문제에 비해 접근이 쉬우며, 일부 예측문제들은 종속변수를 범주화하여 범주 분류 문제로 전환하여 풀기 때문에 Catboost의 활용성은 굉장히 높다고 할 수 있습니다.

2 Background

2.1. Gradient Boosting

먼저 CatBoost를 이해하기 전에 Gradient Boosting에 대해서 짚고 넘어가야할 필요성이 있습니다. 아래와 같은 데이터가 주어졌다고 가정해보겠습니다.

Height (m)	Favorite Color	Gender	Weight (kg)
1.6	Blue	Male	88
1.6	Green	Female	76
1.5	Blue	Female	56
1.8	Red	Male	73
1.5	Green	Male	77
1.4	Blue	Female	57

Height (m)	Favorite Color	Gender	Residual
1.6	Blue	Male	16.8
1.6	Green	Female	4.8
1.5	Blue	Female	-15.2
1.8	Red	Male	1.8
1.5	Green	Male	5.8
1.4	Blue	Female	-14.2

이 데이터는 키(Height), 색깔(Favorite Color), 성별(Gender)를 활용하여 무게(Weight)를 예측하는 문제입니다. 예측하는 가장 간단한 방법은 먼저 무게의 평균으로 모든 데이터를 예측하는 것입니다. 71.2라는 평균값으로 예측을 진행하게 된다면 잔차(Residual)을 구할 수 있습니다.

이후 독립변수들을 사용하여 잔차를 예측하는 의사결정나무를 생성하게 됩니다. 잔차를 통해 의사결정 나무를 생성하게 되면, Leaf Node에는 분류된 값들이 여러 개가 생성되는 경우가 있는데, 이 값들은 평균을 통해 하나의 값으로 대체하게 됩니다.

잔차를 통해 생성된 의사결정 트리를 이용해 기존 예측값을 learning rate를 통해 수정하게 되면 잔차가 줄어드는 것을 확인할 수 있습니다. 첫 번째 행의 경우, 잔차가 16.8에서 15.1로 줄었습니다. Learning rate를 통해 수정할 오차를 결정하는 것은 overfitting을 방지하기 위한 목적입니다. 잔차가 업데이트 되었다면 이 잔차를 이용하여 또 다른 의사결정트리를 생성하게 됩니다.

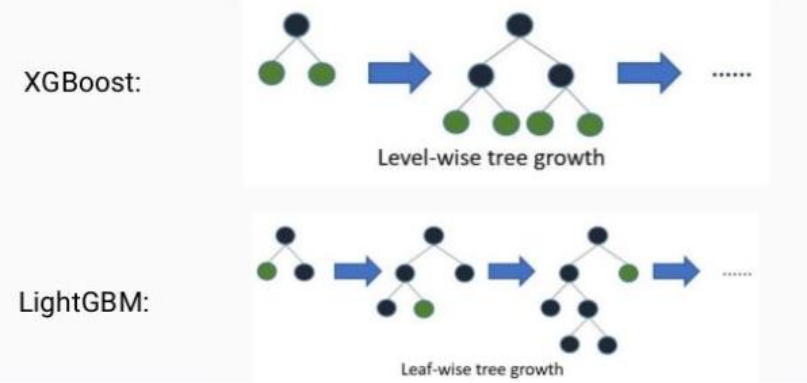
위의 그림처럼 잔차를 통해 의사결정트리를 생성하게 되면, 점점 실제값과 예측값의 오차가 감소하면서, 학습되는 것을 볼 수 있습니다.

이러한 gradient boosting 기법은 정확도를 높인다는 장점을 가지고 있지만, 여러 의사결정트리 모델들을 생성해내기 때문에 학습 속도가 굉장히 오래 걸린다는 단점을 가지고 있습니다. 또한 overfitting에 굉장히 취약하다는 단점을 가지고 있습니다. Boosting 기법 자체가 잔차를 줄이기 위해 학습하는 모델이기 때문에 태생적으로 가진 문제이기도 하지만, 다양한 기법들을 통해 이를 해결하고 있습니다.

3 Proposed Method

3.1. Level-wise Tree

의사결정트리 생성 시, 트리를 확장하는 방식은 대표적으로 Level-wise Tree와 Leaf-wise Tree방식이 존재합니다. Lightgbm은 Leaf-wise Tree 방식을 사용하며, XGboost / Catboost는 Level-wise Tree 방식을 사용하고 있습니다. 이 두 가지 방식 모두 overfitting을 최소화하고, 학습 속도를 개선하기 위해 제안된 방법들입니다.



Level-wise Tree는 트리의 깊이를 줄이기 위하여 대칭 방식을 채택하고 있습니다. 위의 그림처럼 균형 트리 방식은 깊이 증가 시, 트리의 대칭이 중요하기 때문에 깊이가 감소한다는 장점을 가지고 있습니다. 하지만 대칭을 유지하기 위한 추가 연산이 필요하다는 것이 단점입니다. Lightgbm에서 채택하고 있는 leaf-wise 방식은 트리의 대칭은 신경 쓰지 않으며, 비대칭적이고 복잡한 트리를 형성하게 됩니다.

3.2. Orderd Boosting

Catboost는 기존 2장에서 설명한 boosting 과정과 전체적인 과정은 비슷하지만 세부 내용이 다릅니다. 기존의 boosting 모델이 일괄적으로 모든 훈련 데이터를 대상으로 잔차 계산을 수행하였다면, Catboost는 일부만 가지고 잔차 계산을 진행하고, 이를 통해 모델을 생성합니다. 그 뒤의 데이터 잔차는 생성된 의사결정트리를 사용하여 계산하게 됩니다.

Height (m)	Favorite Color	Gender	Weight (kg)
1.6	Blue	Male	88
1.6	Green	Female	76
1.5	Blue	Female	56
1.8	Red	Male	73
1.5	Green	Male	77
1.4	Blue	Female	57

자세한 예제는 다음과 같습니다 위의 데이터에서 우선적으로 1행과 2행을 이용하여 잔차를 계산하고, 이를 기반으로 모델을 생성합니다. 그리고 3, 4행의 잔차를 모델로 예측합니다. 이 모델을 사용하여 1,2,3,4행의 잔차를 계산하고 이를 이용하여 모델을 생성합니다. 이후 5, 6행의 잔차를 계산합

니다. 이와 같은 방식을 데이터의 끝까지 반복하면서 의사결정 트리를 생성하게 됩니다. 만약 모델을 생성할 때, 데이터의 순서가 계속 같다면 매번 같은 순서대로 잔차를 예측하는 모델을 생성할 가능성이 존재하기 때문에 모델을 생성할 때마다 shuffling 혹은 sampling을 통해 데이터의 순서를 섞게 됩니다. 이러한 방식을 채택하여 boosting의 오버피팅을 극복하였습니다.

3.3. Ordered Target Encoding

Catboost는 범주형 변수를 인코딩 시킬 때, 기존과는 다른 방식으로 인코딩을 진행합니다. 간단한 예시를 통해 설명하면 아래와 같습니다

feature 1	class_labels
sunny	35
sunny	32
couldy	15
cloudy	14
mostly_cloudy	10
cloudy	20
cloudy	25

위의 데이터에서 feature1이 독립변수라고 한다면, feature1의 sunny는 인코딩 시, class_labels의 평균값인 33.5로 인코딩 됩니다. 마찬가지 방식으로 cloudy는 18.5로 인코딩 되게 됩니다. 이러한 방식은 기존에 범주형 변수를 인코딩하기 위해 사용하는 one-hot encoding보다 효율적으로 인코딩할 수 있으며, 마치 연속형 변수처럼 표현할 수 있다는 장점을 가지고 있지만, 종속변수의 정보가 반영되기 때문에 overfitting이 될 확률이 굉장히 높습니다.

따라서 Catboost는 이에 대한 해결책으로, 현재 데이터의 인코딩 값은 과거 데이터를 이용하여 산출하게 됩니다. 과거의 데이터에서 sunny와 cloudy의 class_labels의 평균이 각각 30, 28이라면 이를 현재 데이터에 적용하는 것입니다. 이러한 방식을 사용하게 되면 과거 데이터의 정보와 현재 데이터의 정보가 혼용된 모델을 생성해낼 수 있기 때문에 overfitting을 개선할 수 있으며, 수치값의 다양성을 확보할 수 있습니다.

3.4. Categorical Feature Combinations

country	hair color	class_label
India	black	1
India	black	1
India	black	1
india	black	1
russia	white	0
russia	white	0
russia	white	0
russia	white	0

예시를 통해 country와 hair_color는 범주와 class_label 간의 유사성이 굉장히 비슷한 것을 확인할 수 있습니다. Catboost는 이렇게 information gain의 유사성이 비슷한 feature를 하나의 feature로 통합시킵니다. 데이터 전처리에 있어 feature selction의 부담을 줄여주는 기능을 자체적으로 탑재하고 있습니다.

3.5. One-hot Encoding

Catboost는 모든 feature에 대하여 target encoding을 진행하지는 않습니다. 범주의 개수가 적은 feature일 경우, target encoding이 가진 장점인 수치값의 다양성을 확보한다는 의미가 퇴색되기 때문입니다. 따라서 파라미터를 통해 특정 feature의 범주의 개수가 파라미터보다 적은 경우, one-hot encoding을 사용하여 알고리즘 진행의 복잡성을 감소시킵니다.

4. Experiments(Code Review)

4.1. Large Movie Review Dataset

Large Movie Review Dataset

This is a dataset for binary sentiment classification containing substantially more data than previous benchmark datasets. We provide a set of 25,000 highly polar movie reviews for training, and 25,000 for testing. There is additional unlabeled data for use as well. Raw text and already processed bag of words formats are provided. See the README file contained in the release for more details.

Large Movie Review Dataset v1.0

When using this dataset, please cite our ACL 2011 paper [bib].

Contact

For comments or questions on the dataset please contact Andrew Maas. As you publish papers using the dataset please notify us so we can post a link on this page.

Publications Using the Dataset

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. (2011). Learning Word Vectors for Sentiment Analysis. *The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011).*

https://ai.stanford.edu/~amaas/data/sentiment/

<예시 데이터>

I always feel strange and guilty saying it (because I'm a fairly well-educated non-teenager), but I actually sort of lil
"When in Rome" was a traditional Mary-Kate and Ashley movie, complete with the foreign travel, accents, motorb
"When in Rome" and the other Olsen twin movies never pretend to be anything they're not; most of the time, the
My point is, people who watch this movie and expect it to be anything other than another Olsen twin movie will
This movie provides important historical and geographical information, just like many of their other movies (remer
As long as I still feel like I'm on my soapbox, and as long as I can make it relevant to the movie, let me take a mo
"When in Rome," while it does feature "high fashion" and globe-trotting and two girls from a valley in Cali, isn't re

본 실험에 사용한 데이터는 영화 리뷰 데이터입니다. Class label은 “긍정적” 또는 “부정적”으로 구성되어 있으며, 50000개의 데이터로 구성되어 있습니다. Train과 Test는 각각 25000개로 구성되어 있습니다.

```
os.chdir("C:/석사/2020학년도 2학기 수업/비즈니스 어낼리틱스/data/movie/")

df_train = pd.read_csv('train.csv')
df_test = pd.read_csv('test.csv')

#TFIDF 수행
tfidf = TfidfVectorizer(max_features=2000)

X_train = tfidf.fit_transform(df_train['text']).toarray()
X_test = tfidf.transform(df_test['text']).toarray()
```

데이터는 단어로 구성된 리뷰 데이터이기 때문에 텍스트 데이터에 대한 간단한 전처리를 수행합니다. 텍스트를 수치형태로 변환하기 위해 TFIDF 방식을 사용하여 2000개의 상위 단어만을 대상으로 전처리를 진행하였습니다.

문서1 : 사과를 먹고 싶은 날
문서2 : 바나나를 먹고 싶은 날
문서3 : 바나나를 받았는데, 길고 노란 바나나 입니다
문서4 : 저는 과일이 좋아요

-	과일이	길고	노란	먹고	바나나	사과	싶은	저는	좋아요
문서1	0	0	0	1	0	1	1	0	0
문서2	0	0	0	1	1	0	1	0	0
문서3	0	1	1	0	2	0	0	0	0
문서4	1	0	0	0	0	0	0	1	1

$$idf(d,t) = log(\frac{n}{1+df(t)})$$

단어	IDF(역 문서 빈도)
과일이	ln(4/(1+1)) = 0.693147
길고	ln(4/(1+1)) = 0.693147
노란	ln(4/(1+1)) = 0.693147
먹고	ln(4/(2+1)) = 0.287682
바나나	ln(4/(2+1)) = 0.287682

-	과일이	길고	노란	먹고	바나나	사과	싶은	저는	좋아요
문서1	0	0	0	0.287682	0	0.693147	0.287682	0	0
문서2	0	0	0	0.287682	0.287682	0	0.287682	0	0
문서3	0	0.693147	0.693147	0	0.575364	0	0	0	0
문서4	0.693147	0	0	0	0	0	0	0.693147	0.693147

TFIDF는 단어의 등장 빈도수를 기반으로 단어의 중요도를 수치화 시키는 기법 중 하나입니다. 계산 방식이 간단하며, 사용 혹은 추출하고 싶은 단어의 개수를 Top k 방식 혹은 수치 값의 threshold 등 다양한 방식을 통해 설정할 수 있습니다.

4.2. 모델 생성 방식

```
from sklearn.model_selection import KFold

kf = KFold(random_state=30,
           n_splits=10,
           shuffle=True,
          )

kf = kf.get_n_splits(X_train)

from sklearn.model_selection import GridSearchCV

param_grid = {

    'max_depth': list(range(1, 20))

}
```

```
estimator = DecisionTreeClassifier()

grid_search = GridSearchCV(estimator=estimator,
                           param_grid=param_grid,
                           cv= kf,
                           n_jobs=-1,
                           verbose=2
                          )

best = grid_search.best_params_

dt = DecisionTreeClassifier(max_depth=12)
```

모델의 하이퍼파라미터 선정에는 grid search 방식과 cross validation 방식을 사용하였습니다. 파라

미터 탐색 후, best 변수에서 가장 좋은 파라미터를 산출하여 모델을 생성하는 방식입니다. Lightgbm, Catboost는 다른 라이브러리에서 가져오지만 estimator를 설정하여 GridSearchCV에 적용시킬 수 있습니다.

4.3. 모델 구현

```
dt = DecisionTreeClassifier(max_depth=12)


rf = RandomForestClassifier(n_estimators=500,
                            max_features=0.06)

lgbm = LGBMClassifier(n_estimators=2000,
                      feature_fraction=0.06,
                      bagging_fraction=0.67,
                      bagging_freq=1)

cb = CatBoostClassifier(n_estimators=2000,
                        colsample_bylevel=0.06,
                        max_leaves=31,
                        subsample=0.67)
```

모델은 dt(단일 의사결정트리), rf(랜덤포레스트), lgbm(LightGBM), cb(CatBoost)로 4가지를 사용하였습니다. 각 모델의 파라미터는 4.2.절에서 언급한 바와 같은 프로세스를 통해 산출하였습니다. 주의할 점은 Lightgbm과 Catboost는 max_leves 설정 시에 $2^n - 1$ 개로 설정해야한다는 점입니다. max_leaves는 leaf node의 개수이기 때문에 2^n으로 설정할 경우 leaf-wise 혹은 level-wise를 수행하는 것이 의미가 사라지게 됩니다.

4.4. 정확도 비교

```
models = [dt, rf, lgbm, cb]
model_names = [i.__class__.__name__ for i in models]

for m, n in zip(models, model_names):

    if n in ["LGBMClassifier", "CatBoostClassifier"]:

        m.fit(X_train,
              y_train,
              early_stopping_rounds=15,
              verbose=0)
    else:
        m.fit(X_train, y_train)

        accuracy = np.mean(m.predict(X_test) == y_test)

        df_results.loc[n] = accuracy
```

모델의 하이퍼파라미터 선정을 완료 후, 전체 학습 데이터를 사용하여 모델을 생성하여 테스트 데이터를 통해 정확도를 계산합니다.

4.5. 결과

	단일트리	랜덤포레스트	Lightgbm	CatBoost
Accuracy	72.40%	83.08%	87.34%	**87.72%**

정확도 평가 결과는CatBoost가 Lightgbm보다 근소하게 앞서는 것을 확인할 수 있었습니다. Boosting 계열은 Bagging 계열의 ensemble 모델보다 더 좋은 성능을 발휘하는 것을 증명할 수 있습니다. 특히, 단일 의사결정트리의 정확도가 72.40%로 ensemble을 통한 의사결정트리 모델 생성이 굉장히 성능이 좋다는 점을 확인할 수 있었습니다.