Week3_예습과제_강정인

상태	시작 전
↗ 강의	📝 예습과제

Chapter 4 05 GBM

부스팅의 대표적인 구현은 AdaBoost와 그래디언트 부스트가 있음

에이다 부스트(AdaBoost)는 오류 데이터에 가중치 부여하면서 부스팅 수행

GBM은 에이다부스트와 유사하나 가중치 업데이트를 경사 하강법을 이용

- 경사 하강법 오류식 h(x) = y – f(x)를 최소화하는 방향성으로 반복적으로 가중치 값 업데이트

사이킷런은 GBM 기반의 분류를 위해 GradientBoostingClassifier 클래스 제공

일반적으로 GBM이 랜덤 포레스트보다 예측 성능이 뛰어남 하지만 수행 시간이 오래 걸리고 하이퍼 파라미터 튜닝 노력이 필요

GBM 하이퍼 파라미터 소개

loss: 경사 하강법에서 사용할 비용 함수를 지정

learning_rate: GBM이 학습을 진행할 때마다 적용하는 학습률

- Weak learner가 순차적으로 오류 값을 보정해 나가는 데 적용하는 계수
- 0~1 사이의 값을 지정할 수 있고 기본값은 1
- 작은 값을 적용하면 예측 성능이 높아질 가능성이 높지만 수행 시간이 오래걸림
- 큰 값을 적용하면 예측 성능이 떨어질 가능성이 높지만 빠른 수행이 가능

n_estimators: weak learn의 개수

- 개수가 많을수록 예측 성능이 좋음 하지만 수행 시간이 오래걸림
- 기본값은 100

Subsample: weak learner가 학습에 사용하는 데이터의 샘플링 비율

- 기본값은 1이며 전체 학습 데이터를 기반으로 학습한다는 의미

06 XGBoost

일반적으로 다른 머신러닝보다 뛰어난 예측 성능

XGBoost는 GBM에 기반하고 있지만 GBM의 느린 수행 시간 및 과적합 규제의 부재 등의 문제를 해결해서 매우 각광을 받고 있음

특히 XGBoost는 병렬 CPU 환경에서 병렬 학습이 가능함

일반 파라미터

booster: gbtree 또는 gblinear 선택, 디폴트는 gbtree

silent: 디폴트는 0이며 출력 메시지를 나타내고 싶지 않을 경우 1로 설정

nthread: CPU의 실행 스레드 개수를 조정하며 디폴트는 CPU의 전체 스레드를 다 사용

주요 부스터 파라미터

eta: GBM의 학습률과 같은 파라미터. 0에서 1 사이의 값을 지정하며 부스팅 스텝을 반복적으로 수행할 때 업데이트되는 학습률 값

파이썬 래퍼 기반의 xgboost를 사용할 경우 디폴트는 0.3

사이킷런 래퍼 클래스를 이용할 경우 디폴트는 0.1

num_boost_rounds: GBM의 n_estimators와 같은 파라미터

min_child_weight: 트리에서 추가적으로 가지를 나눌지 결정하기 위해 필요한 weight의 총합

gamma: 트리의 리프 노드를 추가적으로 나눌지를 결정할 최소 손실 감소 값

col_samples: GBM의 max_features와 유사

lambda: L2 Regularization 적용 값

scale_pos_weight : 특정 값으로 치우친 비대칭한 클래스로 구성된 데이터 세트의 균형을 유지하기 위한 파라미터

학습 태스크 파라미터

objective: 최솟값을 가져야 할 손실 함수 정의

binary:logistic: 이진 분류일 때 적용

multi:softmax: 다중 분류일 때 적용

multi:softprob: multi:softmax와 유사하나 개별 레이블 클래스의 해당되는 예측 확률을 반환

eval_metric: 검증에 사용되는 함수를 정의

XGBoost는 기본 GBM에서 부족한 여러 가지 성능 향상 기능

대표적인 기능으로 조기 중단(Early Stopping) 기능이 있음

07LightGBM

XGBoost보다 학습에 걸리는 시간이 훨씬 적음

또한 메모리 사용량도 상대적으로 적음

LightGBM은 일반 GBM 계열의 트리 분할 방법과 다르게 리프 중심 트리 분할 방식을 사용

리프 중심 트리 분할 방식은 트리의 균형을 맞추지 않고 최대 손실 값을 가지는 리프 노드를 지속적으로 분할하면서 트리의 깊이가 깊어지고 비대칭적인 규칙 트리가 생성

Light GBM의 하이퍼 파라미터

주요 파라미터

num_iterations : 반복수행하려는 트리의 개수 지정

learning_rate: 0과 1 사이의 값을 지정하며 부스팅 스텝을 반복적으로 수행할 때 업데이트되는 학습률 값

boosting[default=gbdt]: 부스팅의 트리를 생성하는 알고리즘을 기술

bagging_fraction : 트리가 커져서 과적합되는 것을 제어하기 위해서 데이터를 샘플리하는 비율 지정

feature_fraction: 개별 트리를 학습할 때마다 무작위로 선택하는 피처의 비율

lambda_l2 : L2 regulation 제어를 위한 값

lambda_l1 : L1 regulation 제어를 위한 값

Learning Task 파라미터

Objective: 최솟값을 가져야 할 손실함수를 정의

08 베이지안 최적화 기반의 HyperOpt를 이용한 하이퍼 파라미터 튜닝

베이지한 최적화 단계

1. 랜덤하게 하이퍼 파라미터들을 샘플링하고 성능 결과 관측
2. 관측된 값을 기반으로 대체 모델을 최적 함수를 추정
3. 추정된 최적 함수를 기반으로 획득 함수는 다음으로 관측할 하이퍼 파라미터 값을 계산
4. 획득 함수로부터 전달된 하이퍼 파라미터를 수행하여 관측된 값을 기반으로 대체 모델 갱신

HyperOpt의 주요 로직

첫째는 입력 변수명과 입력값의 검색 공간 설정

둘째는 목적 함수의 설정

마지막으로 목적 함수의 반환 최솟값을 가지는 최적 입력값 유추

11. **스태킹 앙상블**

스태킹은 개별 알고리즘으로 예측한 기반으로 다시 예측을 수행

스태킹은 두 종류의 모델 필요

첫 번째는 개별적인 기반 모델이고 두 번째 이 개별 기반 모델의 예측 데이터를 학습 데이터로 만드어서 학습하는 최종 메타 모델

스태킹 모델의 핵심은 여러 개별 모델의 예측 데이터를 가각 스태킹 형태로 결합해 최종 메타 모델의 학습용 피처 데이터 세트와 테스트용 피처 데이터 세트를 만드는 것임

기본 스태킹 모델

개별 모델은 KNN, 랜덤 포레스트, 결정 트리, 에이다부스트이며 이 모델의 예측 결과를 합한 데이터 세트로 학습하고 예측하는 최종 모델은 로지스틱 회귀

CV 세트 기반의 스태킹

1. 각 모델별로 원본 학습하고 테스트 데이터를 예측한 결과 값을 기반으로 메타 모델을 위한 학습용/테스트용 데이터를 생성
2. 만들어진 학습용 데이터를 모두 스태킹 형태로 합쳐서 학습용/테스트용 데이터 세트 생성
3. 메타 모델을 생성된 학습용 데이터 세트를 기반으로 학습한 뒤 최종적으로 생성된 테스트 데이터 세트를 예측하고 원본 테스트 데이터의 레이블 데이터를 기반으로 평가