[파머완 4.5장 ~ 4.7장, 4.10장 / HyperOpt]

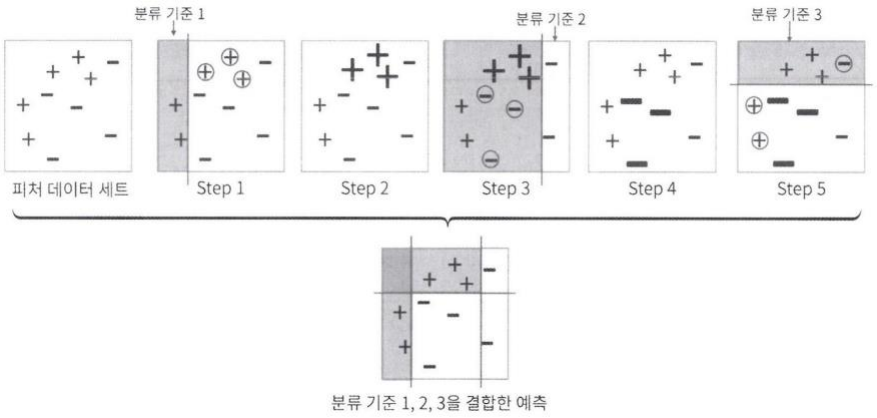
5. GBM(Gradient Boosting Machine)

1) GBM의 개요 및 실습

부스팅 알고리즘: 여러 개의 약한 학습기(weak learner)를 순차적으로 학습-예측하면서 잘못 예측한 데이터에 가중치 부여를 통해 오류를 개선해 나가며 학습하는 방식

ex) 에이다 부스트(오류 데이터에 가중치를 부여하면서 부스팅을 수행하는 대표적인 알고리즘), 그래디언트 부스트

(1) 에이다 부스트



+, -로 된 피처 데이터 세트

Step 1: 첫 번째 약한 학습기가 분류 기준 1로 +와 -를 분류한 것

동그라미로 표시된 ⊕ 데이터는 + 데이터가 잘못 분류된 오류 데이터

Step 2: 이 오류 데이터에 대해 가중치 값을 부여, 가중치가 부여된 오류 + 데이터는 다음 약한 학습기가 더 잘 분류할 수 있게 크기가 커짐

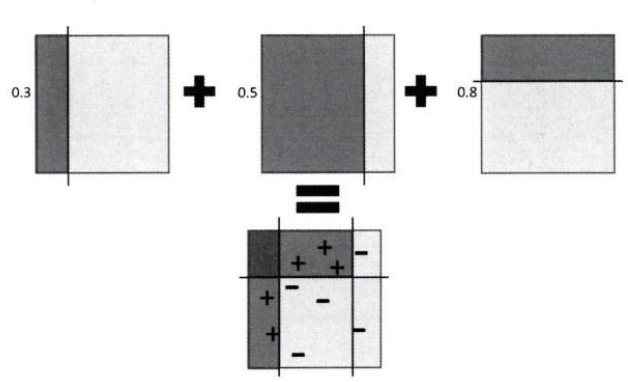
Step 3: 두 번째 약한 학습기가 분류 기준 2로 +와 -를 분류함, 마찬가지로 동그라미로 표시된 ⊖ 데이터는 잘못 분류된 오류 데이터

임

Step 4: 잘못 분류된 이 – 오류 데이터에 대해 다음 약한 학습기가 잘 분류할 수 있게 더 큰 가중치를 부여(오류 – 데이터의 크기가 커짐)

Step 5: 세 번째 약한 학습기가 분류 기준 3으로 +와 -를 분류하고 오류 데이터를 찾음, 에이다부스트는 이렇게 약한 학습기가 순차적으로 오류 값에 대해 가중치를 부여한 예측 결정 기준을 모두 결합해 예측 수행

마지막: 맨 아래에는 첫 번째, 두 번째, 세 번째 약한 학습기를 모두 결합한 결과 예측임, 개별 약한 학습기보다 훨씬 정확도가 높아졌음을 알 수 있음



개별 약한 학습기는 그림과 같이 각각 가중치를 부여해 결합

(2) 그래디언트 부스트

그래디언트 부스트도 에이다부스트와 유사 but 가중치 업데이트를 경사 하강법을 이용하는 것이 큰 차이

(오류 값) = (실제 값) – (예측 값)

분류의 실제 결괏값 = y

피처 = $x_1, x_2, x_3, \ldots, x_n$

이 피처에 기반한 예측 함수 = F(x) 함수

오류식: h(x) = y – F(x)

경사 하강법(Gradient Descent): 오류식을 최소화하는 방향성으로 반

복적으로 가중치 값을 업데이트하는 것
분류, 회귀 모두 가능
GBM 기반의 분류를 위해서 GradientBoostingClassifier 클래스 제공

➔ 주피터 노트북으로 사이킷런의 GBM을 이용해 사용자 행동 데이터 세트를 예측 분류

일반적으로 GBM이 랜덤 포레스트보다는 예측 성능이 조금 뛰어난 경우가 많음 but 수행 시간 오래 걸림 + 하이퍼 파라미터 튜닝 노력도 더 필요

GradientBoostingClassifier – 많은 시간 필요

랜덤 포레스트 – 상대적으로 빠른 수행 시간 보장

2) GBM 하이퍼 파라미터 및 튜닝

Loss	경사 하강법에서 사용할 비용 함수 지정 기본값 = deviance
learning_rate (n_estimators와 상호 보완적 조합&사용)	GBM이 학습을 진행할 때마다 적용하는 학습률 weak learner가 순차적으로 오류 값을 보정해 나가는 데 적용하는 계수 기본값 = 0.1
n_estimators	weak learner의 개수 기본값 = 100
subsample	weak learner가 학습에 사용하는 데이터의 샘플링 비율 기본값 = 1 (100%)

GBM은 과적합에도 뛰어난 예측 성능 가짐 but 시간 오래 걸림

➔ 그래디언트 부스팅 기반 ML 패키지 각광받음(XGBoost, LightGBM)

6. XGBoost(eXtra Gradient Boost)

1) XGBoost 개요

뛰어난 예측 성능	일반적으로 분류, 회귀 영역에서 뛰어난 예측 성능 발휘
GBM 대비 빠른 수행 시간	병렬 수행 및 다양한 기능으로 빠른 수행 성능을 보장
과적한 규제 (Regularization)	자체에 과적합 규제 기능으로 과적합에 강한 내구성 가짐
Tree pruning (나무 가지치기)	max_depth 파라미터로 분할 깊이를 조정하기도 하지만, tree pruning으로 더 이상 긍정 이득이 없는 분할을 가지치기하여 분할 수 더 줄이는 추가적 장점 소지
자체 내장된 교차 검증	반복 수행 시마다 내부적으로 교차 검증 수행 → 최적화된 반복 수행 횟수를 가질 수 있음 세트 평가 값이 최적화 되면 반복을 중간에 멈출 수 있는 조기 중단 기능이 있음
결손값 자체 처리	XGBoost는 결손값을 자체 처리할 수 있는 기능 가짐

핵심 라이브러리는 C/C++로 작성됨

"xgboost" 파이썬 패키지 지원 → 파이썬 래퍼 XGBoost

사이킷런과 연동 가능한 래퍼 클래스 제공 → 사이킷런 래퍼 XGBoost 모듈

사이킷런 래퍼 XGBoost 모듈은 사이킷런의 다른 Estimator과 사용법 동일 but 파이썬 네이티브 XGBoost는 고유의 API와 하이퍼 파라미터 이용

2) XGBoost 설치하기 → 좀이따가 와파 연결해서 해놓가

3) 파이썬 래퍼 XGBoost 하이퍼 파라미터

GBM과 유사한 하이퍼 파라미터 + 조기 중단 + 과적합 규제를 위한 하이퍼 파라미터

(1) 파이썬 래퍼 XGBoost 하이퍼 파라미터의 유형

일반 파라미터	일반적으로 실행 시 스레드의 개수나 silent 모드 등의 선택을 위한 파라미터, 디폴트 값의 거의 안바꿈
부스터 파라미터	트리 최적화, 부스팅, regularization 등과 관련 파라미터 등을 지칭
학습 태스크 파라미터	학습 수행 시의 객체 함수, 평가를 위한 지표 등을 설정하는 파라미터

대부분의 하이퍼 파라미터는 Booster 파라미터에 속함

주요 일반 파라미터	booster	gbtree(디폴트) / gblinear
	silent	디폴트 = 0 출력 메시지 x = 1
	nthread	CPU 실행 스레드 개수 조정 디폴트는 전체 스레드 모두 사용
	eta	디폴트 = 0.3 alias = learning_rate 0에서 1 사이의 값을 지정하며 부스팅 스텝을 반복적으로 수행할 때 업데이트되는 학습률 값
	num_boost_rounds	GBM의 n_estimators와 동일
	min_child_weight	디폴트 = 1 트리에서 추가적으로 가지를 나눌지를 결정하기 위해 필요한 데이터들의 weight 총합 과적합 조절
	gamma	디폴트 = 0 alias = min_split_loss 트리의 리프 노드를 추가적으로 분리할지 결정하는 최소 손실 감소 값
	max_depth	디폴트 = 6

주요 부스터 파라미터		트리 기반 알고리즘의 max_depth와 동일
	sub_sample	디폴트 = 1 GBM의 subsample과 동일 트리가 커져 과적합되는 것을 제어하기 위해 데이터 샘플링 비율 지정
	colsample_bytree	디폴트 = 1 GBM의 max_features와 유사 트리 생성에 필요한 피처(칼럼) 임의 샘플링
	lambda	디폴트 = 1 alias = reg_lambda
	alpha	디폴트 = 0 alias = reg_alpha
	scale_pos_weight	디폴트 = 1 비대칭한 클래스로 구성된 데이터 세트의 균형을 유지하기 위한 파라미터
학습 태스크 파라미터	objective	최소값을 가져야 할 손실 함수 정의
	binary:logistic	이진 분류에 적용
	multi:softmax	다중 분류일 때 적용
	multi:softprob	multi:softmax와 유사하나 개별 레이블 클래스의 해당되는 예측 확률 반환
	eval_metric	검증에 사용되는 함수 기본값 = rmse(회귀), error(분류) rmse, mae, logloss, error, merror, mlogloss, auc

(2) 과적합 문제가 심각할 경우 사용하는 방법

① eta 값 낮추기
② max_depth 값 낮추기
③ min_child_weight 값 높이기
④ gamma 값 높이기
⑤ subsample과 colsample_bytree를 조정하는 것도 트리가 너무 복잡하게 생성되는 것 예방 → 과적합 문제에 도움

4) 파이썬 래퍼 XGBoost 적용 – 위스콘신 유방암 예측

xgboost는 자체적으로 교차 검증, 성능 평가, 피처 중요도등의 시각화 기능을 가지고 있음
학습용과 테스트용 데이터 세트를 위해 별도의 객체인 DMatrix를 생성

params (dict)	부스터 파라미터
dtrain (DMatrix)	학습 데이터
num_boost_round (int)	부스팅 반복 횟수
nfold (int)	CV 폴드 개수
stratified (bool)	CV 수행 시 층화 표본 추출 수행 여부
metrics (string or list of strings)	CV 수행 시 모니터링할 성능 평가 지표
early_stopping_rounds	조기 중단을 활성화시킴, 반복 횟수 지정

5) 사이킷런 래퍼 XGBoost의 개요 및 적용
eta → learning_rate
sub_sample → subsample
lambda → reg_lambda
alpha → reg_alpha

XGBClassifier 클래스의 fit(), predict(), predict_proba()를 이용하여 학습과 예측 수행

7. LightGBM

1) LightGBM

장점: 더 빠른 학습과 예측 수행 시간, 더 작은 메모리 사용량, 카테고리형 피처의 자동 변환과 최적 분할(원-핫 인코딩 등을 사용하지 않고도 카테고리형 피처를 최적으로 변환하고 이에 따른 노드 분할 수행)

공통점: 대용량 데이터에 대한 뛰어난 예측 성능 및 병렬 컴퓨팅 기능을 제공하고 있으며, 추가로 최근에는 GPU까지 지원

2) LightGBM 하이퍼 파라미터

주요 파라미터	num_iterations	default = 100 반복 수행하려는 트리 개수 지정
	learning_rate	default =0.1 0에서 1사이의 값 지정하여 부스팅 스텝을 반복수행시 업데이트되는 학습률 값
	max_depth	default = -1 트리 기반 알고리즘의 max_depth와 동일
	min_data_in_leaf	default = 20 결정 트리의 min_samples_leaf와 동일
	num_leaves	default = 31 하나의 트리가 가질 수 있는 최대 리프 개수
	boosting	default = gbdt 부스팅의 트리 생성하는 알고리즘 기술
	bagging_fraction	default = 1.0 데이터 샘플링 비율 지정
	feature_fraction	default = 1.0 개별 트리를 학습할 때마다 무작위로 선택하는 피처의 비율

	lambda_l2	default = 0.0 regulation 제어를 위한 값
	lambda_l1	default = 0.0 L1 regulation 제어를 위한 값
Learning Task 파라미터	objective	최솟값을 가져야 할 손실함수

3) 하이퍼 파라미터 튜닝 방안

num_leaves의 개수를 중심으로 min_child_samples (min_data_in_leaf), max_depth를 함께 조정하면서 모델의 복잡도를 줄이는 것

4) 파이썬 래퍼 LightGBM과 사이킷런 래퍼 XGBoost, LightGBM 하이퍼 파라미터 비교

유형	파이썬 래퍼 LightGBM	사이킷런 래퍼 LightGBM	사이킷런 래퍼 XGBoost
파라미터명	num_iterations	n_estimators	n_estimators
	learning_rate	learning_rate	learning_rate
	max_depth	max_depth	max_depth
	min_data_in_leaf	min_child_samples	N/A
	bagging_fraction	subsample	subsample
	feature_fraction	colsample_bytree	colsample_bytree
	lambda_l2	reg_lambda	reg_lambda
	lambda_l1	reg_alpha	reg_alpha
	early_stopping_round	early_stopping_rounds	early_stopping_rounds
	num_leaves	num_leaves	N/A
	min_sum_hessian_in_leaf	min_child_weight	min_child_weight

5) LightGBM 적용 – 위스콘신 유방암 예측

10. 스태킹 앙상블

개별적인 여러 알고리즘을 서로 결합해 예측 결과를 도출→공통점

알고리즘으로 예측한 데이터를 기반으로 다시 예측 수행

1) 기본 스태킹 모델
   (1) 개별적인 기반 모델
   (2) 개별 기반 모델의 예측 데이터를 학습 데이터로 만들어서 학습하는 최종 메타 모델
2) CV 세트 기반의 스태킹

   학습 데이터가 아닌 테스트용 레이블 데이터 세트를 기반으로 학습했기에 과적합 문제가 생길 가능성 존재

   ➔ 개선 위해 개별 모델들이 각각 교차 검증으로 메타 모델을 위한 학습용 스태킹 데이터 생성과 예측 위한 테스트용 스태킹 데이터 생성 뒤 이를 기반으로 메타 모델이 학습과 예측 수행
      (1) 각 모델별 원본 학습/테스트 데이터 예측 결과 값 기반으로 메타 모델을 위한 학습용/테스트용 데이터 생성
      (2) 모두 스태킹 형태로 합쳐 메타 모델이 학습할 최종 학습용 데이터 세트 생성, 원본 테스트 데이터의 레이블 데이터 기반으로 평가