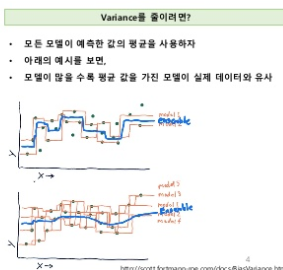
앙상블이란 : 여러 모델을 사용하여 데이터를 학습하고, 모든 모델의 예측결과를 평균하여 예측

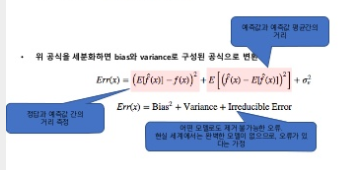
에러의 최소화

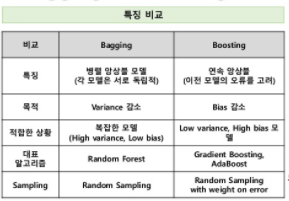
과적합 감소(Low bias, High variance 문제를 해결하자)

* Bias란 예측값과 실제값 간의 차이
* variance란 모델간의 예측 정확도 분산을 말함



에러는 Bias, Variance, 잔차(컨트롤할 수 없는 오류)로 이루어져 있다.



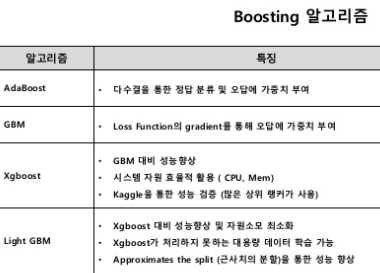


Bagging : 전체 데이터를 복원 샘플링 하여 여러개의 모델식을 생성, 정확도를 비교 후 값들을 평균내어 결과값 도출 -> low Variance가 됨

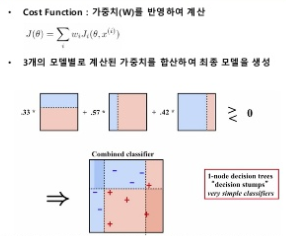
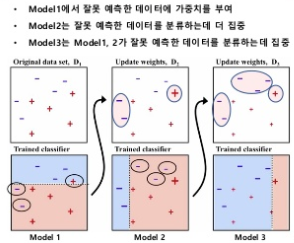
* 주의 - Bagging 시 특출 나게 유의한 변수 A가 있을 경우 아무리 복원추출로 여러 모델을 생성 하더라도 A로 치중된 모델만 나올 것이다.(모델간의 식별이 어려울 것이다)

\*랜덤 포레스트 : 데이터 샘플링 시 일부 feature들만 랜덤으로 선택 -> 모든 모델들은 다른 feature로 학습하게 되고, 이로 인해 tree correlation(트리 모델끼리의 상관성)이 줄어든다.

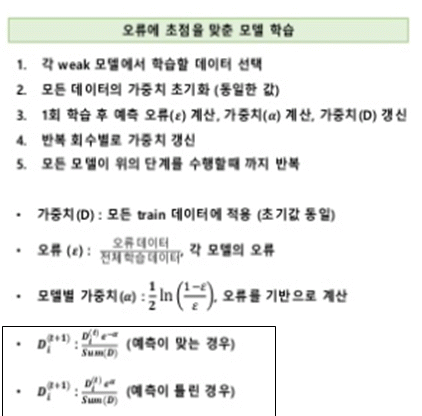
Boosting : 전체 데이터에서 샘플링 한 후 모델을 생성, 모델에서 오류가 생성된 데이터를 따로 추출하여 가중치를 부여(다음 모델에서 오류가 생긴 데이터의 식별을 강화하기 위해 - 이상치를 만드는 작업이라 생각하면 쉽다.), 다음 모델에 포함하여 새로운 모델식을 생성



Adaboost



model 1에서 분류하지 못한 세 개(-: 2개, +:1개)데이터에 가중치를 부여하여 model2 에서는 이 데이터를 식별하기 용이하게 모델이 세워진다.(그림을 보면 –그림과 +그림이 model1그림보다 커졌다)

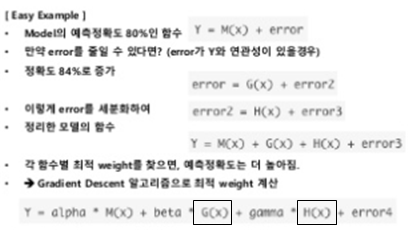


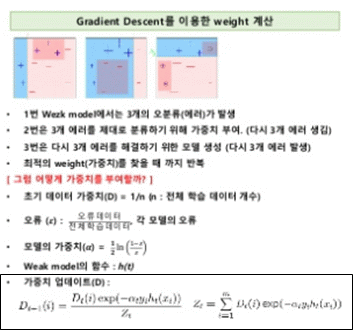
GBM(Gradient Boosting)

Adaboost와 기본 개념은 동일

가중치(D) 계산 방식에서 Gradient Descent를 이용하여 최적의 파라미터를 찾는다.

(이 부분을 XGboost에서도 이용한 것 같다 : 네모 친 부분을 기억하자)





Adaboost와 가중치 업데이트 부분만 다르다.

업데이트 과정은 알파(learning rate), 예측값 y, 예측모델 h(x)를 지수의 곱으로 이용하여 이전 모델의 가중치와 계산한다. 이렇게 수행함으로써 최적의 가중치 값을 찾아나가는 점이 descent gradient 성질과 유사함

XGboost (eXtreme Gradient Boost)

GBM + 분산/병렬 처리

지도학습으로 변수(x)를 학습하여 정답(y)를 예측

Xgboost를 지원하는 모델:

(예측, 분류, 랭크? 등 모든 모델을 지원한다 : 내생각엔 CART의 성질 때문으로 보임)

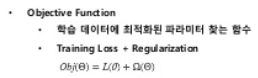
Binary classification

Multiclass classification

Regression

Learning to Rank

Objetive Function(목적함수)

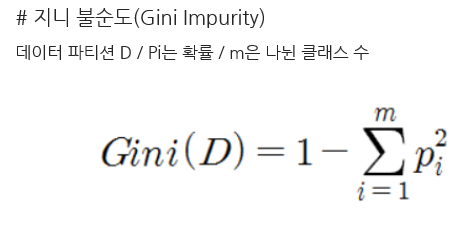


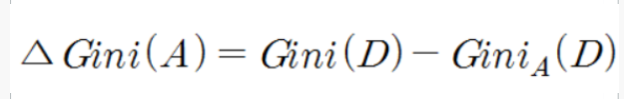
목적함수는 가중치를 최적화하기 위한 손실함수(Cost Function)와 Regularization으로 구성

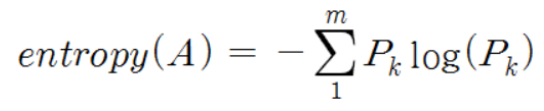
XGboost에서 사용하는 모델은 CART의 집합으로 이루어져 있다

CART(Classification And Regression Tree)란?

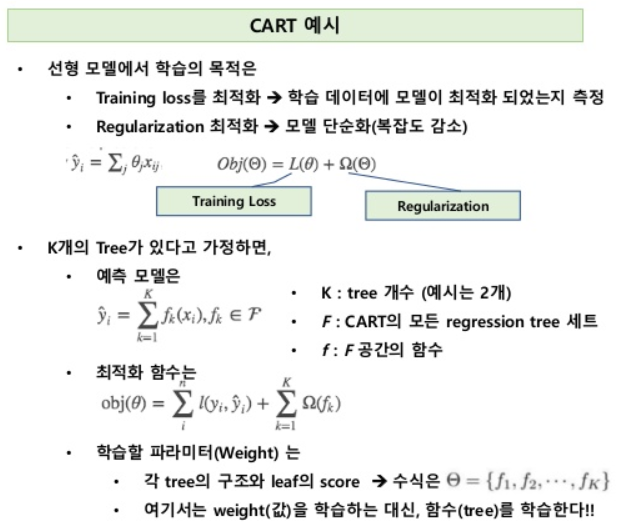
의사결정 나무 기법중 하나로 독립변수와 종속변수의 데이터 성질이 무엇이든 간에 이진분류 하여 예측 또는 분류가 가능하다.





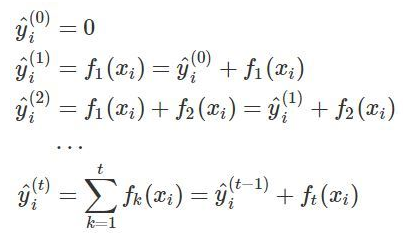


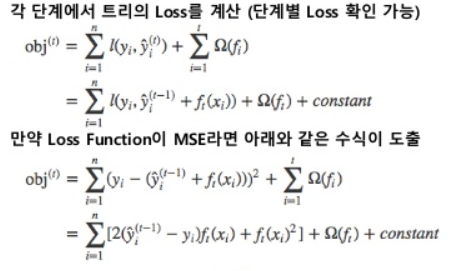
|  |  |  |  |
| --- | --- | --- | --- |
| CART | | 종속변수(y) | |
| 연속형 | 명목형 |
| 독립변수(x) | 연속형 | 가능 | 가능 |
| 명목형 | 가능 | 가능 |



기존 최적화 함수(SGD)는 한번의 여러개의 트리 최적화가 어려움 -> 그래서 한 단계씩

각 단계 t에서의 예측값 : Y^(t) -> 이전 단계의 트리에 새로운 트리를 더해 나감





2차항에 대해 손실 함수의 테일러 전개를 취함

테일러 전개? <http://hydepark.tistory.com/174> -> 복잡한 함수를 다항식으로 변경이 가능하다:

x=0 근처에서 f(x) 를 "이차식" a + bx + cx2 으로 근사시키려면?

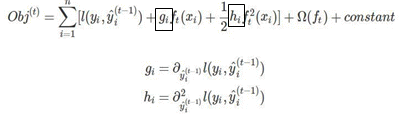
0 을 대입하여 같아야 하므로 a = f(0)

미분하여 0 을 대입하여 같아야 하므로 b = f'(0)

두번 미분하여 0 을 대입하여 같아야 하므로 2c = f''(0)

즉, a = f(0), b = f'(0) 는 아까와 같고 c = f''(0)/2 로 하면 됩니다. 이렇게 하면 일차식으로 근사 시켰을 때보다 넓은 범위에서도 상당히 그럴듯한 근사식이 됩니다.

==> 우리가 바꾸고 싶은 형태의 함수로 바꿀 수 있다고 생각함

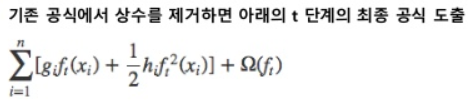
그리고 여기서는

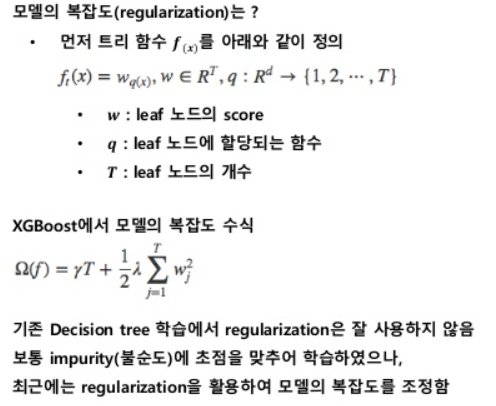
gi, hi의 함수를 가지는 형태의 함수로 바꾸고 싶어 한다.(이거 어디서 본거 같은데 하니 GBM에서

EMB00001b9053f3

아닌가?? ㅎㅎㅎㅎㅎㅎㅎ

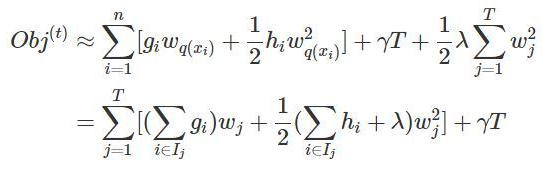
\* gi와 hi는 사용자 정의 손실 함수를 지원할 수 있는 방법이다. 이를 이용하여 로지스틱 회귀, 가중 로지스틱 회귀 등 복잡한 손실함수를 최적화 시킬 수 있다.





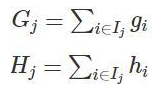
정규항을 보면 첫째항 감마티는 Lasso와 유사하고, 두 번째는 Ridge와 유사하다함

트리함수 f를 잎 노드의 점수에 관한 식으로 정리하면 w로 재정의하면

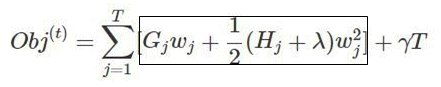


i와 n에 대한(트리모델)을 표현하면 모든 잎새노드의 합(j, T)으로 표현할 수 있다.

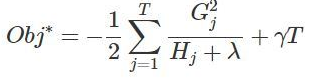
gi(1차 미분값), hi(2차 미분값)함수는 Gj, Hj로 표현이 가능하다.



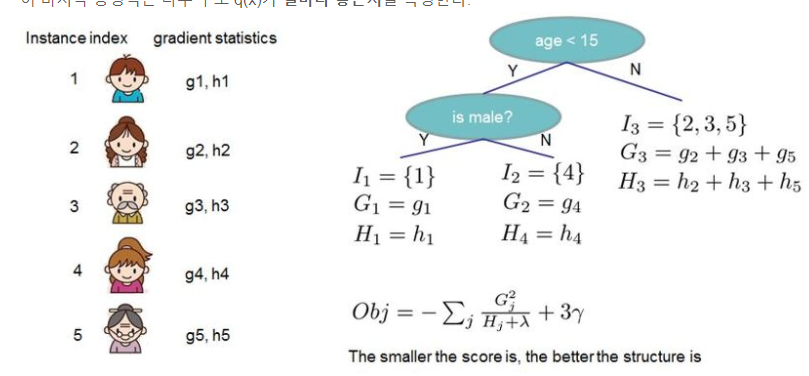
다시 식을 정리하면

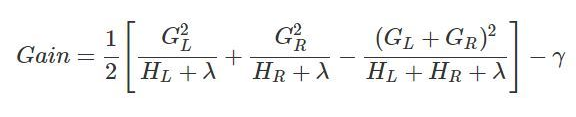


네모 부분을 wj에 대해 방정식을 세우면 wj = -Gj/(Hj+람다)가 되고 결국 최종 함수 q(x)는

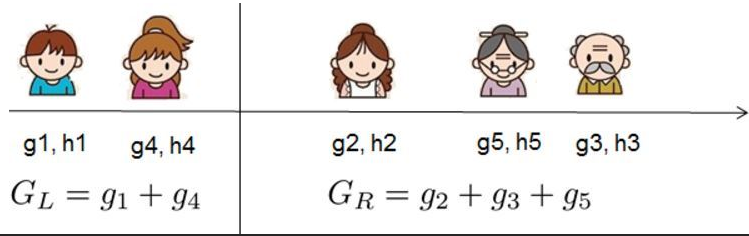


이 된다.





불순도 구하는 문제와 비슷하다. (왼쪽, 오른쪽 이진분류 하여 불순도 측정방식)



**태현님이 주신 자료를 바탕으로 작성하였습니다.**

**그리고** [**http://ishuca.tistory.com/388**](http://ishuca.tistory.com/388) **를 참조하였습니다**

