13기 정규세션

ToBig's 12기 배유나

Ensemble

Contents

Unit 01 | Ensemble

앙상블 정의

‘ 조화 또는 통일 ‘

< 모델 예측에 입각? >
어떤 데이터의 값을 예측한다고 할 때, 하나의 모델을 활용합니다.
하지만 여러 개의 모델을 조화롭게 학습시켜 그 모델들의 예측 결과들을 이용한다면
더 정확한 예측값을 구할 수 있을 겁니다.

Unit 01 | Ensemble

앙상블 학습은 여러 개의 모델을 결합하여
하나의 모델보다 더 좋은 성능을 내는 머신러닝 기법입니다.
앙상블 학습의 핵심은 여러 개의 약 분류기 (Weak Classifier)를 결합하여
강 분류기(Strong Classifier)를 만드는 것입니다.
그리하여 모델의 정확성이 향상됩니다.

앙상블의 기대효과
1. 단일 모형을 사용할때 보다 성능 분산이 감소하고, 즉 과적합 방지
2. 개별 모형이 성능이 안좋을 경우에는 결합 모형의 성능이 더 향상

Unit 01 | Ensemble

모형 결합 방법은 크게 취합(aggregation) 방법론과 부스팅(boosting) 방법론으로 나뉜다.

-취합 방법론
1. 다수결 (Majority Voting)
2. 배깅 (Bagging)
3. 랜덤 포레스트 (Random Forests)

-부스팅 방법론
1. 에이다 부스트 (AdaBoost)
2. 그레디언트 부스트(Gradient Boost)

Unit 02 | 필요 선행 지식

필요 선행 지식 !

1. 부트스트랩
2. ~~결정트리~~

(-> 갓기오 띵강이 있었으니 더 이상 설명은 필요 없음)

Unit 02 | 부트스트랩

모수 분포 추정 시 복원 추출하여 각 표본에 대한 통계량을 다시 계산

예, 1억 개의 모집단에서 뽑은 200개의 표본이 있다. 200개로만 통계량을 구하는 것이 아니라 **200개를 기준으로 복원 추출**하여 새로운 통계량을 구하는 것

1. 200개의 표본 중 하나를 뽑아 기록하고 다시 제자리에 둡니다.
2. 이를 n번 반복합니다.
3. n번 재표본추출한 값의 평균을 구합니다.
4. 1~3 단계를 R번 반복합니다. (R: 부트스트랩 반복 횟수)
5. R개의 평균에 대한 결과를 사용하여 신뢰구간을 구합니다.

표본이 200개밖에 없을지라도 부트스트랩을 통해 200개 보다 더 많은 통계량을 가질 수 있다.

Unit 02 | 부트스트랩

Data 관점에서 설명

"부트스트랩 = 뻥튀기"

10000개의 레코드로 된 데이터세트가 있다고 가정합니다.
10000개의 레코드를 10000번 복원추출(resampling)을 합니다. 그러면 갯수는 똑같이 10000개가 됩니다. 다시 이 과정을 반복해서 100번을 해서 10000개 짜리 데이터세트를 100개를 만들고 이 것으로 각 모델들을 만듭니다. 그러면 100개의 조금씩 다른 모델을 만들 수 있습니다.

(* 추가적 부트스트랩의 효과 설명 된 자료 : https://stats.stackexchange.com/questions/26088/explaining-to-laypeople-why-bootstrapping-works)

Unit 03 | Ensemble

모형 결합 방법은 크게 취합(aggregation) 방법론과 부스팅(boosting) 방법론으로 나뉜다.

-취합 방법론
1. 다수결 (Majority Voting)
2. 배깅 (Bagging)
3. 랜덤 포레스트 (Random Forests)

-부스팅 방법론
1. 에이다 부스트 (AdaBoost)
2. 그레디언트 부스트(Gradient Boost)

취합 방법론 – 1. 다수결 방법 (Majority Voting)

다수결 방법은 가장 단순한 모형 결합 방법으로 전혀 다른 모형도 결합할 수 있습니다.
다수결 방법은 두 가지로 나뉜다.

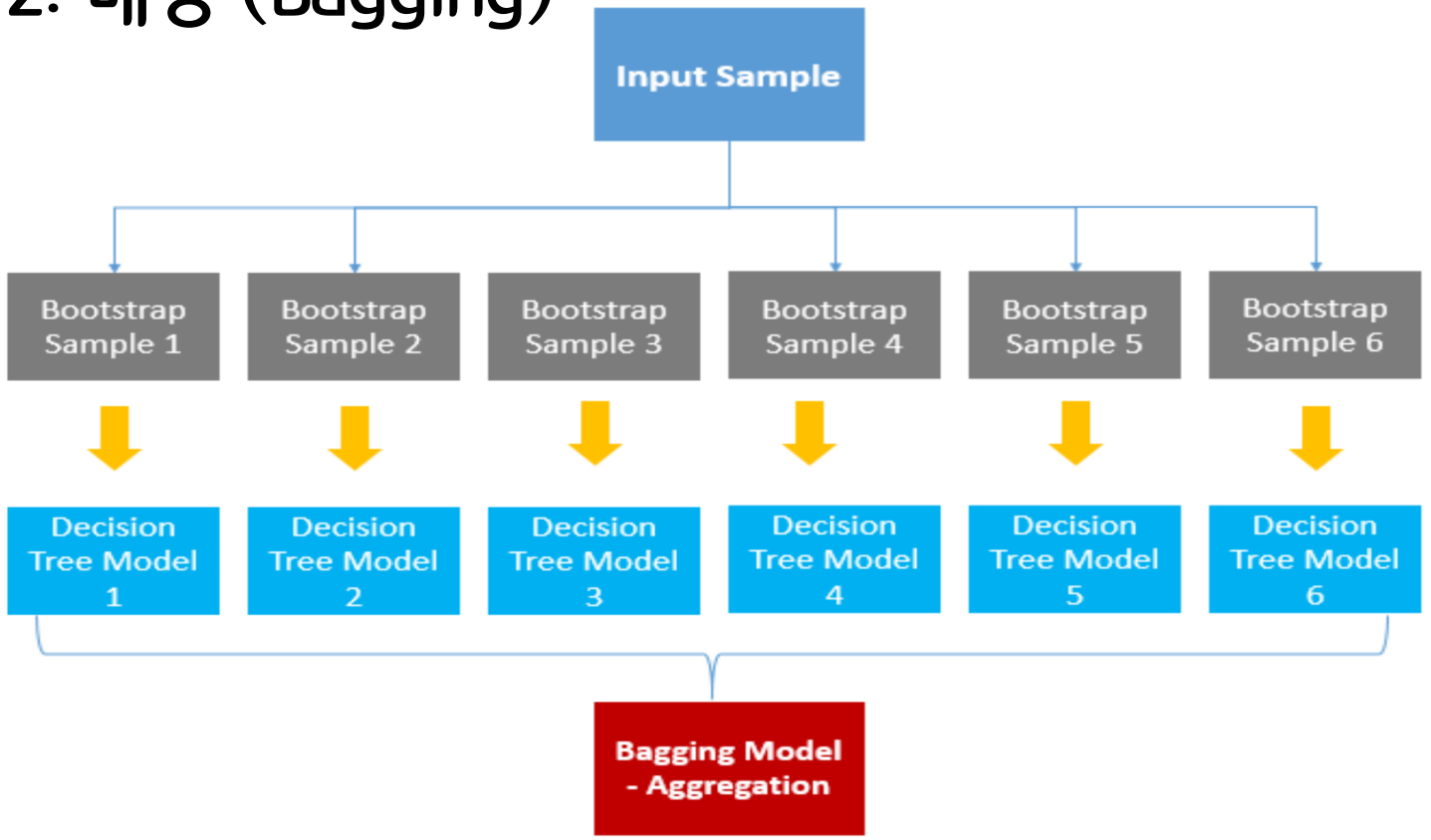
hard voting : 단순 투표. 개별 모형의 결과 기준
soft voting : 가중치 투표. 각 분류기의 예측을 평균 내어 확률이 가장 높은 클래스

취합 방법론 – 2. 배깅 (Bagging)

Bootstrap Aggregation의 약자.
배깅은 샘플을 여러 번 뽑아(Bootstrap) 각 모델을 학습시켜 결과물을 집계(Aggregration)하는 방법
동일한 모델을 사용하여 데이터만 분할하여 여러개 모델을 학습

배깅은 부트스트래핑과 유사하게 트레이닝 데이터를 랜덤하게 선택해서
다수결 모형을 적용

취합 방법론 – 2. 배깅 (Bagging)



출처: swallow.github.io

1) 데이터로부터 부트스트랩을 합니다. (복원 랜덤 샘플링)

2) Voting
- Categorical Data는 투표 방식(다수결)으로 결과를 집계
- Continuous Data는 평균으로 집계

취합 방법론 – 2. 배깅 (Bagging)

OOB(Out-of-Bag) 평가

배깅은 중복을 허용하는 리샘플링(resampling) 즉, 부트스트래핑(bootstraping)
전체 학습 데이터셋에서 어떠한 데이터 샘플은 여러번 샘플링 되고, 또 어떠한 샘플은 전혀 샘플링 되지 않을 수가 있다. 평균적으로 학습 단계에서 전체 학습 데이터셋 중 63% 정도만 샘플링 되며, 샘플링 되지 않은 나머지 37% 데이터 샘플들을 oob(out-of-bag) 샘플이라고 한다.

앙상블(배깅) 모델의 학습 단계에서는 oob 샘플이 사용되지 않기 때문에, 이러한 oob 샘플을 검증셋(validation set)이나 교차검증(cross validation)에 사용할 수 있다.

취합 방법론 – 2. 배깅 (Bagging)

배깅은 간단하면서도 효과적인 방법으로, 배깅을 활용한 모델이 '랜덤 포레스트'입니다

취합 방법론 – 3. 랜덤 포레스트 (Random Forests)

랜덤포레스트 직역하면,

나무가 모여 숲을 이룬다
= 결정트리(Decision Tree)가 모여 랜덤 포레스트(Random Forest)를 구성한다.

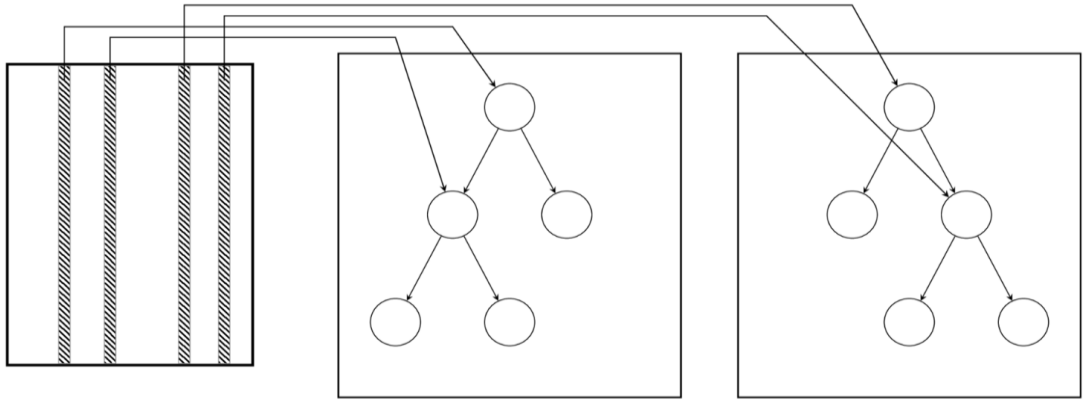
취합 방법론 – 3. 랜덤 포레스트 (Random Forests)

결정트리 하나로도 머신러닝을 할 수 있지만, 결정트리는 오버피팅이 된다는 단점이 있다.
그러나 여러 개의 결정트리를 통해 랜덤포레스트를 만들면 오버피팅이 되는 단점 해결가능

예, 성적을 예측하기 위해서는 많은 요소를 고려해야 합니다. 공부시간 투자, 구글링, 연습, 스터디, 끈기 등등등 수많은 요소가 필요할 것입니다. 이렇게 많은 요소를 기반으로 성적을 예측한다면 분명 오버피팅이 일어날 것입니다. 가령 지금 피쳐가 30개라면, 30개의 피쳐를 기반으로 하나의 결정트리를 만든다면 트리의 가지가 많아질 거고, 오버피팅 초래…..

취합 방법론 – 3. 랜덤 포레스트 (Random Forests)

하지만, 30개의 Feature 중 랜덤으로 5개의 Feature만 선택해서 하나의 결정트리를 만들고, 또 30개 중 랜덤으로 5개의 Feature를 선택해서 또 다른 결정 트리를 만들는 과정을 계속해서 반복하여 여러개의 결정트리를 만들면 결정트리마나 하나씩의 예측값을 뱉게됩니다.



(랜덤포레스트는 데이터 특징차원의 일부만 선택하여 사용한다. 하지만 노드 분리시 모든 독립 변수들을 비교하여 최선의 독립 변수를 선택하는 것이 아니라 독립 변수 차원을 랜덤하게 감소시킨 다음 그 중에서 독립 변수를 선택한다. 이렇게 하면 개별 모형들 사이의 상관관계가 줄어들기 때문에 모형 성능의 변동이 감소하는 효과가 있다.)

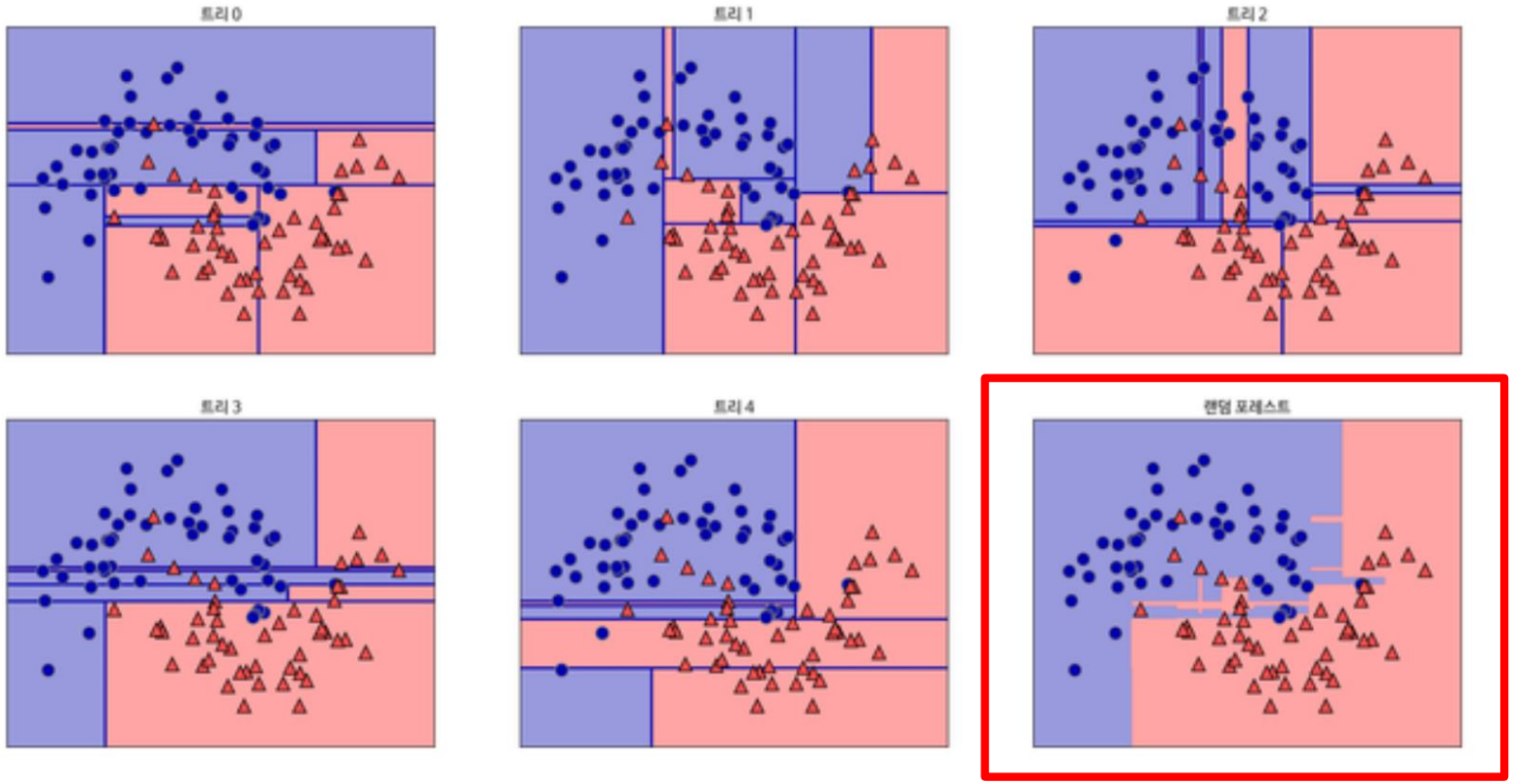
취합 방법론 – 3. 랜덤 포레스트 (Random Forests)

여러 결정 트리들이 뱉은 예측값 중 가장 많이 나온 값을 최종 예측값으로 선정! (다수결의 법칙)
이렇게 의견을 통합하거나 여러 가지 결과를 합치는 방식이 앙상블!!

즉, 하나의 거대한 (깊은) 결정 트리를 만드는 게 아니고 여러 개의 작은 결정 트리를 만드는 것.
여러 개의 작은 결정트리가 예측한 값들 중 가장 많은 값(classification) 혹은
평균값(regression)을 최종 예측값으로 선정

쉽게말해, 걍 집단지성?!? 혼자 문제 푸는 것보다 20명 머리 모아서 문제 푸는 게 더 잘 맞춘다 같은 원리임

취합 방법론 – 3. 랜덤 포레스트 (Random Forests)



출처: 텐서 플로우 블로그

우측 하단 그림 빼고 나머지는 결정 트리 각각의 결정경계인데 보면 결정트리 경계는 다소 모호하고 오버피팅된 게 보이죠.

5개의 결정트리 경계를 평균내어 만든 랜덤 포레스트의 경계는 더 깔끔한 것을 확인 할 수 있습니다

Unit 03 | Ensemble

모형 결합 방법은 크게 취합(aggregation) 방법론과 부스팅(boosting) 방법론으로 나뉜다.

-취합 방법론
1. 다수결 (Majority Voting)
2. 배깅 (Bagging)
3. 랜덤 포레스트 (Random Forests)

-부스팅 방법론
1. 에이다 부스트 (AdaBoost)
2. 그레디언트 부스트(Gradient Boost)

부스팅(Boosting)

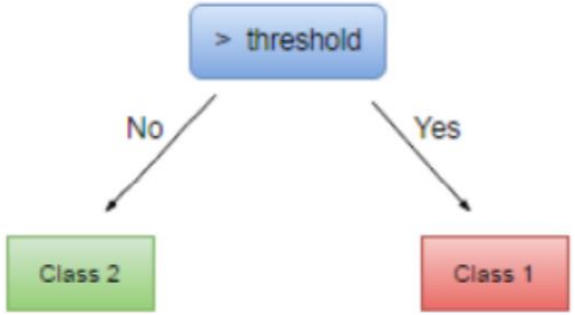
부스팅은 가중치를 활용하여 약 분류기를 강 분류기로 만드는 방법입니다.

배깅은 Deicison Tree1과 Decision Tree2가 서로 독립적으로 결과를 예측합니다. 여러 개의 독립적인 결정 트리가 각각 값을 예측한 뒤, 그 결과 값을 집계해 최종 결과 값을 예측

하지만 부스팅은 모델 간 팀워크가 이루어집니다. 처음 모델이 예측을 하면 그 예측 결과에 따라 데이터에 가중치가 부여되고, 부여된 가중치가 다음 모델에 영향을 줍니다. 잘못 분류된 데이터에 집중(가중치 부여)하여 새로운 분류 규칙을 만드는 단계를 반복합니다.

부스팅 방법론 – 4. 에이다 부스트 (AdaBoost)

" adaptive + boosting "
간단한 약분류기(feature 하나를 가지고 if문 하나정도의 depth를 가진 분류기)들이



상호보완하도록 순차적으로 학습, 이들을 조합하여 최종 강분류기의 성능을 증폭시킨다.

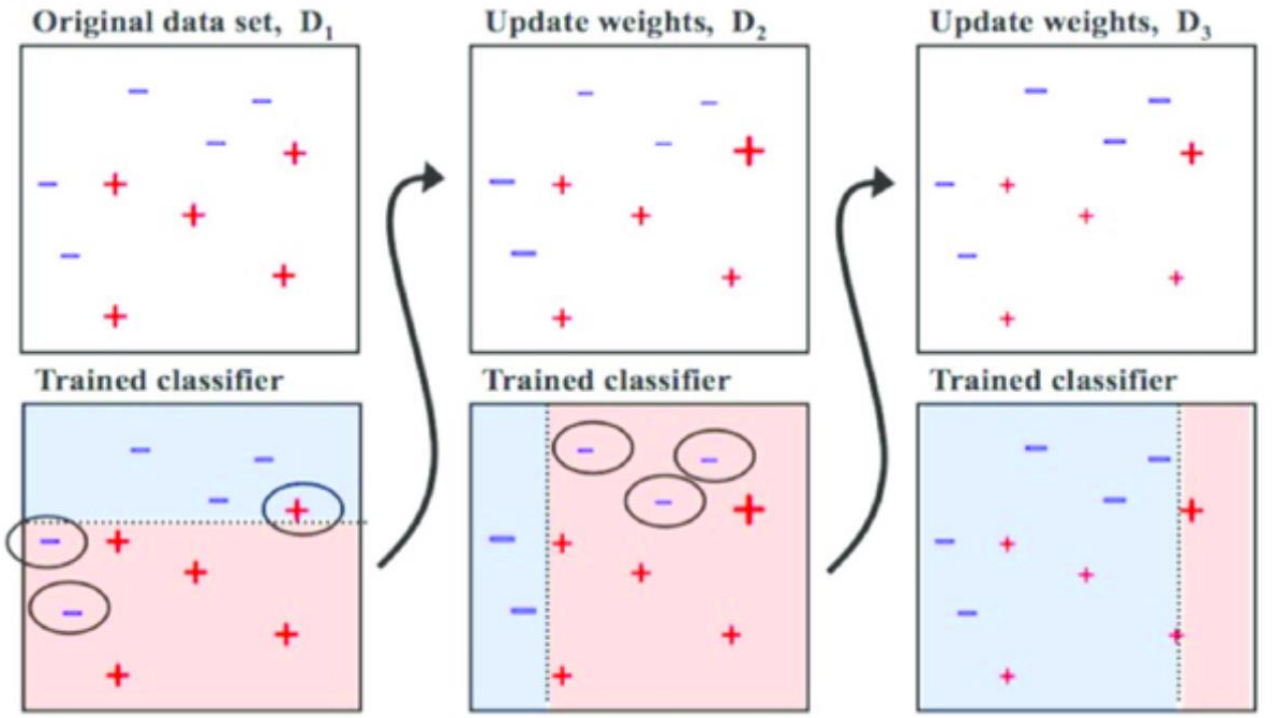
부스팅 방법론 – 4. 에이다 부스트 (AdaBoost)

아다부스트는 과소적합됐던 학습 데이터 샘플의 가중치를 높이면서(adaptive하게)
새로 학습된 모델이 학습하기 어려운 데이터에 더 잘 적합(더 집중)되도록 하는 방식이다

1. 먼저 전제 학습 데이터셋을 이용해 모델을 만든 후, 잘못 예측(분류)된 샘플의 가중치를 상대적으로 높여준다.
2. 그 다음 두 번째 모델을 학습 시킬때 이렇게 업데이트된 가중치를 반영하여 모델을 학습 시킨다.
3. 이와 같은 과정을 반복한다.

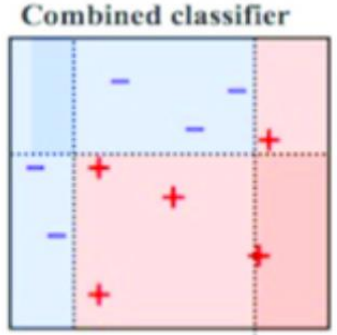
에이다 부스트 (AdaBoost)

잘못 분류가 된 데이터는 가중치를 높여주고,
잘 분류된 데이터는 가중치를 낮추어 줍니다. (크기)



위쪽의 +는 잘못 분류가 되었고, 아래쪽의 두 -도 잘못 분류

분류가 잘못된 데이터에 가중치를 부여해주는 이유는 다음 모델에서 더 집중해 분류하기 위함

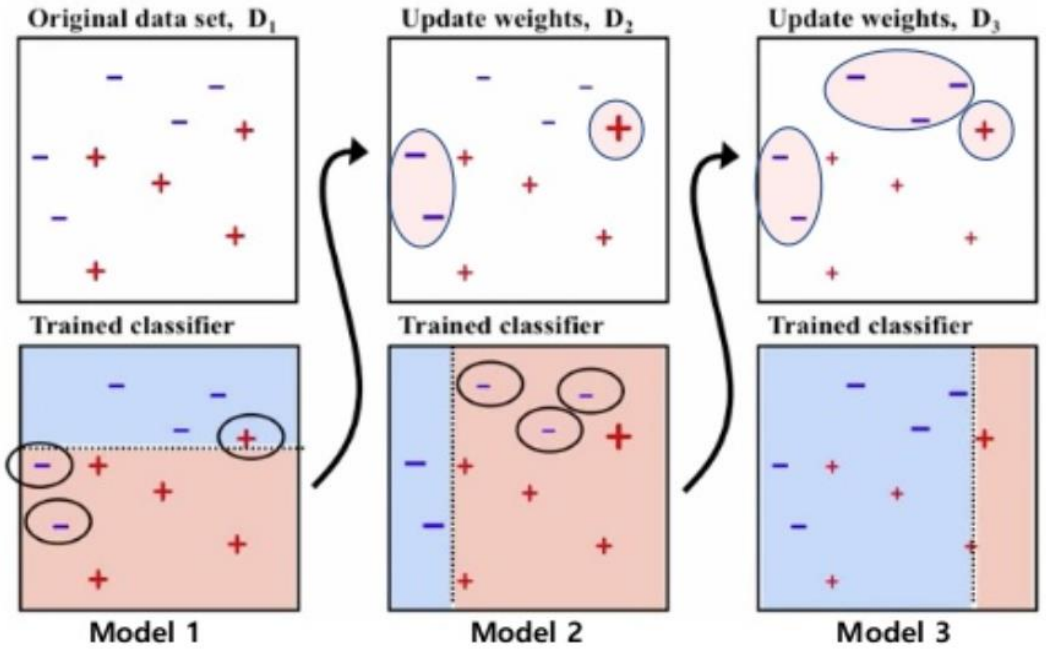


D1, D2, D3의 Classifier를 합쳐 최종 Classifier를 구할 수 있습니다. 최종 Classfier는 +와 -를 정확하게 구분해줍니다.

출처: Medium (Boosting and Bagging explained with examples)

부스팅 방법론 – 4. 에이다 부스트 (AdaBoost)

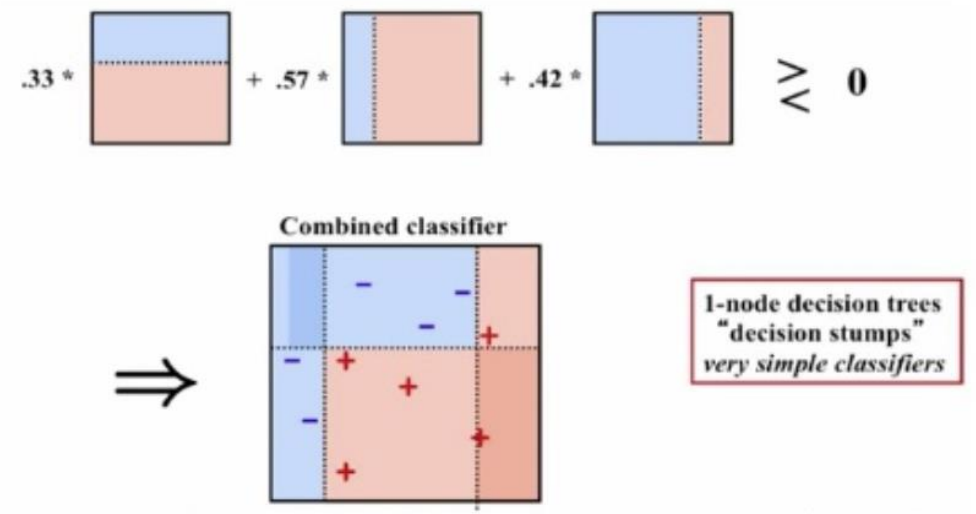
- Model1에서 잘못 예측한 데이터에 가중치를 부여
- Model2는 잘못 예측한 데이터를 분류하는데 더 집중
- Model3는 Model1, 2가 잘못 예측한 데이터를 분류하는데 집중



- Cost Function : 가중치(W)를 반영하여 계산

$$J(\theta) = \sum_i w_i J_i(\theta, x^{(i)})$$

- 3개의 모델별로 계산된 가중치를 합산하여 최종 모델을 생성



https://www.youtube.com/watch?v=ix6IvwbVpw0&list=PL4zv4UkoVTPflyFDJdJtz248-8Wdz_nct&index=3

부스팅 방법론 – 4. 에이다 부스트 (AdaBoost)

Algorithm 10.1 *AdaBoost.M1.*

1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.

   값 샘플의 weight를 1/N로 초기화

   N은 데이터의 갯수

2. For $m = 1$ to M: M은 모델의 갯수

   Weight값을 기준으로 분류기 생성(resampling)

   (a) Fit a classifier $G_m(x)$ to the training data using weights w_i.

   (b) Compute 해당 분류기의 에러 계산

   $$I(y_i, G_m(x_i)) = \begin{cases} 0 & \text{if } y_i = G_m(x_i) \\ 1 & \text{if } y_i \neq G_m(x_i) \end{cases}$$

   $$\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$

   (c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$. 해당 분류기의 분류기 가중치 생성

   (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))],\ i = 1, 2, \ldots, N$.

   Instance의 weight 업데이트

3. Output $G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.

부스팅 방법론 – 4. 에이다 부스트 (AdaBoost)

값 샘플의 weight를 1/N로 초기화

1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N.$

N은 데이터의 갯수

모든 Instance의 초기값은 다 동일하게 시작을 함.

Ex) Instance가 100개 이면 모든 W_i = 1/100

부스팅 방법론 – 4. 에이다 부스트 (AdaBoost)

2. For $m = 1$ to M: M은 모델의 갯수

Weight값을 기준으로 분류기 생성(resampling)

(a) Fit a classifier $G_m(x)$ to the training data using weights w_i.

Instance의 Weight값을 기준으로 Sampling을 진행

최초 샘플링 : 모두 뽑힐 확률이 동일함.

n round 샘플링 : 틀린 값에 가중치가 부여되므로 뽑힐 확률이 높아짐

!지금부터 for문이 돌아간다고 생각하면 됩니다!!!

부스팅 방법론 – 4. 에이다 부스트 (AdaBoost)

(b) Compute 해당 분류기의 에러 계산 $I(y_i, G_m(x_i)) = \begin{cases} 0 & \text{if } y_i = G_m(x_i) \\ 1 & \text{if } y_i \neq G_m(x_i) \end{cases}$

$$\mathrm{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$

예측값과 실제값이 같으면 0이 되고, 다르면 1이 되게 함.

분자 : 틀린 예측 값의 instance의 weigh만 남게 되어 모두 더함

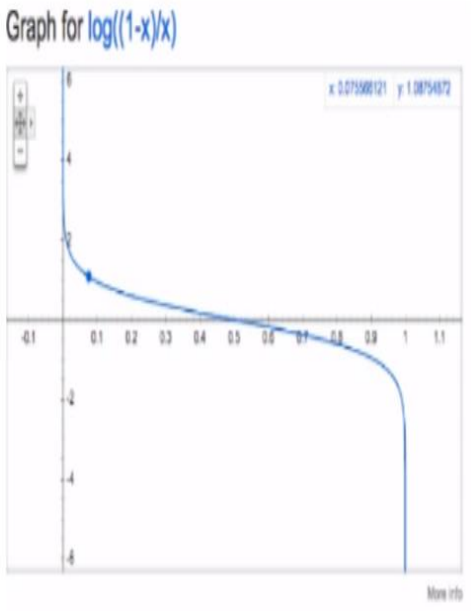
분모 : 모든 가중치를 더 함(Normalizer 역할)

따라서 err의 범위는 0<err<1이다.

부스팅 방법론 – 4. 에이다 부스트 (AdaBoost)

(c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$. 해당 분류기의 분류기 가중치 생성

- 에러가 높으면 m번째 모델은 좋지 않는 모델
- 에러가 낮으면 m번째 모델은 좋은 모델
- 따라서 반비례 관계이므로 (1-err)/err를 통해 반대방향으로 바꿔 모델의 가중치를 계산한다.
- 전체 앙상블에 얼마만큼의 역할을 반영할지 계산하는거임!!



부스팅 방법론 – 4. 에이다 부스트 (AdaBoost)

$$I(y_i, G_m(x_i)) = \begin{cases} 0 & \text{if } y_i = G_m(x_i) \\ 1 & \text{if } y_i \neq G_m(x_i) \end{cases}$$

(d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))]$, $i = 1, 2, \ldots, N$.

Instance의 weight 업데이트

모델의 가중치의 exp를 하여 틀린 놈들만 가중치를 업데이트 시켜버림.

이 짓을 for문이 Classifier 개수만큼 돌아가게 합니다.

부스팅 방법론 – 4. 에이다 부스트 (AdaBoost)

3. Output $G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.

그러면 Classifier M개를 돌렸겠죠? 그리고 각각의 모델의 가중치가 있을 겁니다. 그 가중치를 곱하여 각각의 역할 비중을 줘서 Sumation을 통해 예측을 하는겁니다.

부스팅 방법론 – 4. 에이다 부스트 (AdaBoost)

AdaBoost는 weight가 낮은 데이터 주위에 높은 weight를 가진 데이터가 있으면
의도치 않게 잘못 분류가 되면서 성능이 크게 떨어지는 단점이 있다….

그럼 어떻게???

Gradient Descent 알고리즘을 사용하여서 오류값을 최소화 하는 방식으로 학습
예로 Mean Squared Error를 쓰는 경우라면,
현재 모델의 MSE의 미분값인 residual을 target으로 두고 다음 모델을 학습함으로써
이전 모델에서 오류를 다음 모델에서 줄일 수 있도록 한다.

그게 그래디언트 부스팅!!!!!!!!!!!!!~~~~~~~~~~

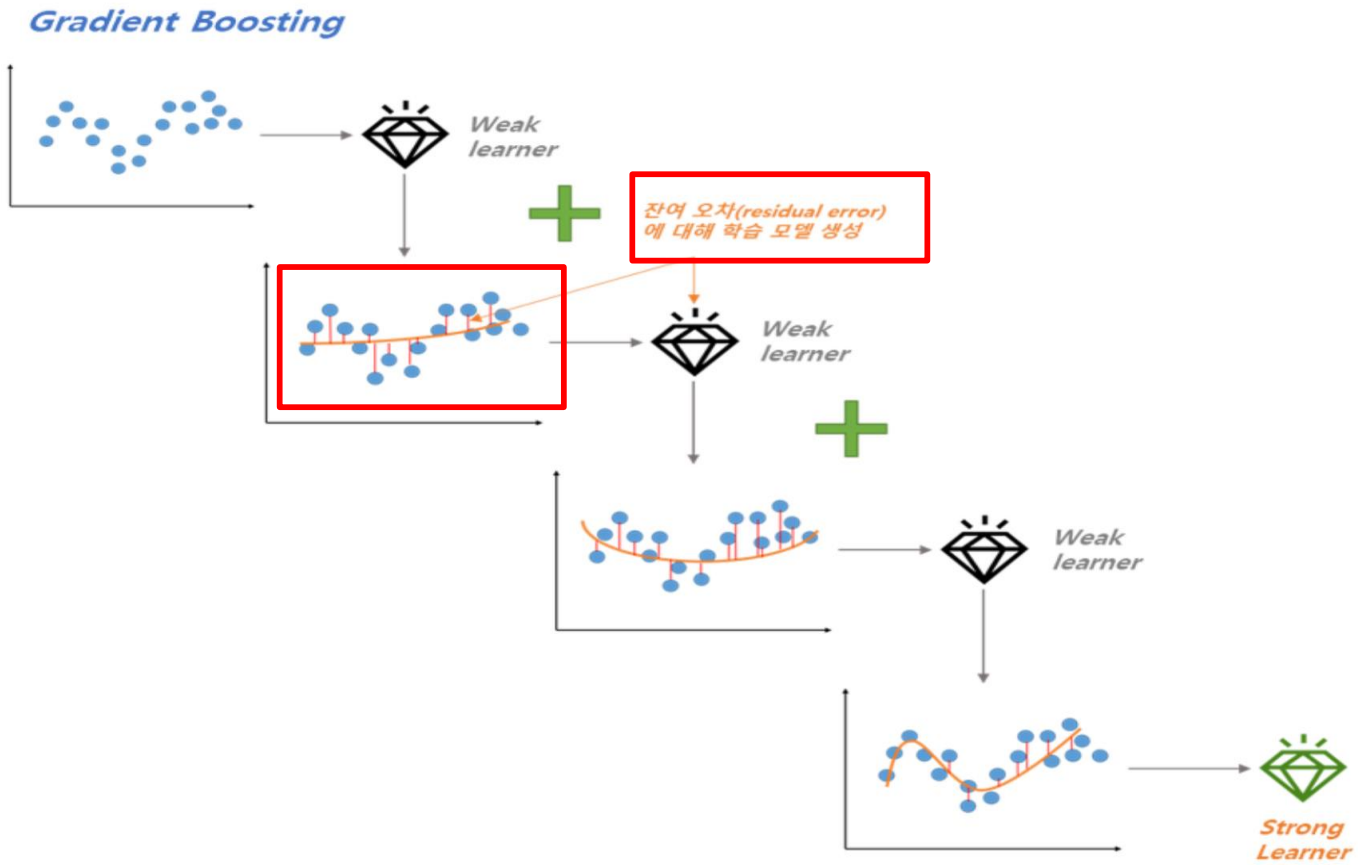
부스팅 방법론 – 5. 그레디언트 부스트(Gradient Boost)

"Sequential + Additive Model"

전에 학습된 모델의 오차를 보완하는 방향으로 모델을 추가해주는 방법은 동일하다.
하지만, 그래디언트 부스팅은 아다부스트 처럼 학습단계 마다 데이터 샘플의 가중치를
업데이트 해주는 것이 아니라
학습 전단계 모델에서의 잔여 오차(residual error)에 대해 새로운 모델을 학습시키는 방법

-> residual 을 예측하여 발전하는 weak learner!!

부스팅 방법론 – 5. 그레디언트 부스트(Gradient Boost)



부스팅 발전 모델 – XGBoost / Light GBM

둘다 기반은 gradient boosting 임
Gradient Boost 는 굉장히 잘 만들어진 알고리즘이지만, 속도가 느리고 비효율적

이를 개선하기 위해서 Microsoft 에서 만든 알고리즘으로 완전한 패키지 형태로 제공되어 사용자들이 손쉽게 사용할 수 있다.
대표적인 차이는 Regularization 을 사용한다는 점과 각종 분산처리 지원을 통한 속도 향상 및 Package 화를 통해 자동으로 Hyper Parameter Search 를 지원하는 등 손쉽게 알고리즘을 사용할 수 있는 편의성 정도가 될 것 같다.

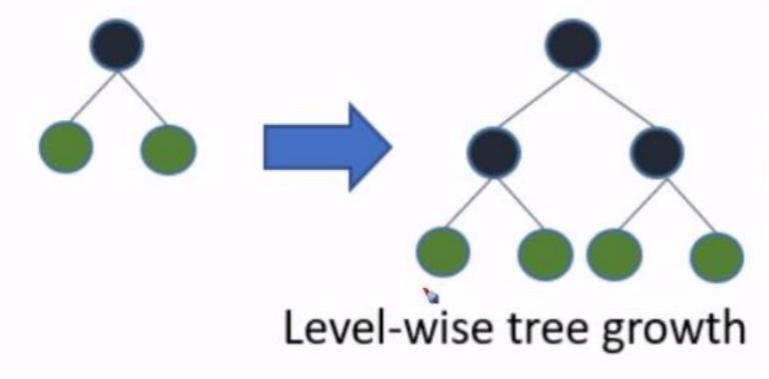
부스팅 발전 모델 – 6. XGBoost

GBM vs Xgboost

1. Extreme Gradient Boosting - XGBoost
2. Both xgboost and gbm follows the principle of gradient boosting.
3. There are however, the difference in modelling & performance details.
4. xgboost used a more regularized model formalization to control over-fitting, which gives it better performance.
5. Improved convergence techniques, vector and matrix type data structures for faster results
6. Unlike GBM XGBoost package is available in C++, Python, R, Java, Scala, Julia with same parameters for tuning

부스팅 발전 모델 – 6. XGBoost

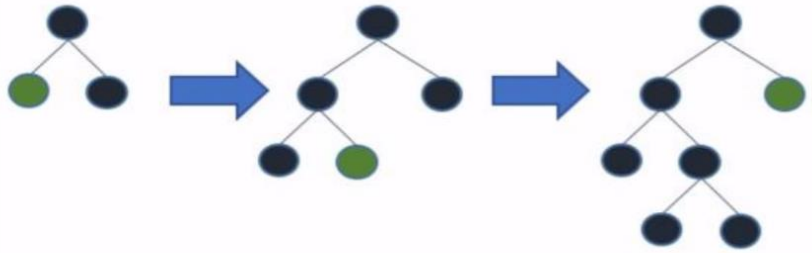
균형적으로 성장



XGBoost는 매우 뛰어난 부스팅 알고리즘이지만, 여전히 학습시간이 오래걸립니다.
XGBoost에서 GridSearchCV로 하이퍼 파라미터 튜닝을 수행하다보면,
수행시간이 너무 오래걸려서 많은 파라미터를 튜닝하기에 어려움을 겪을 수 밖에 없습니다.
물론 GBM보다 빠르겠지만, 대용량 데이터의 경우 만족할 만한 학습성능을 기대하려면
많은 CPU코어를 가진 시스템에서 높은 병렬도로 학습을 진행해야합니다.;;;;; money needs..;;

부스팅 발전 모델 – 7. Light GBM

한쪽으로만 성장



Leaf-wise tree growth

LightGBM의 가장 큰 장점은 XGBoost보다 학습에 걸리는 시간이 훨씬 적다는 점입니다.
또한 메모리 사용량도 상대적으로 적습니다.

사실 LightGBM과 XGBoost의 예측성능은 별다른 차이가 없습니다.
그러나 기능상의 다양성은 LightGBM이 더 많습니다. 더 최신것이기 때문이죠.
단점으로 알려진것중 하나는 데이터의 수가 적으면 과적합이 발생하기 쉽다는 것입니다.
(공식문서에서 10000개 이하.)

부스팅 발전 모델 – 7. Light GBM

기존의 대부분 트리 기반 알고리즘은 트리의 깊이를 효과적으로 줄이기 위한 균형 트리 분할 방식을 사용합니다.
즉, 최대한 균형 잡힌 트리를 유지하면서 분할하기 때문에 트리의 깊이가 최소화 될 수 밖에 없습니다. 이렇게 균형잡힌 트리를 생성하는 이유는 오버피팅에 보다 더 강한 구조를 가질 수 있다고 알려져있기 때문입니다.
반대로, 균형을 맞추기위한 시간이 필요하다는 상대적인 단점이 있습니다.

부스팅 발전 모델 – 7. Light GBM

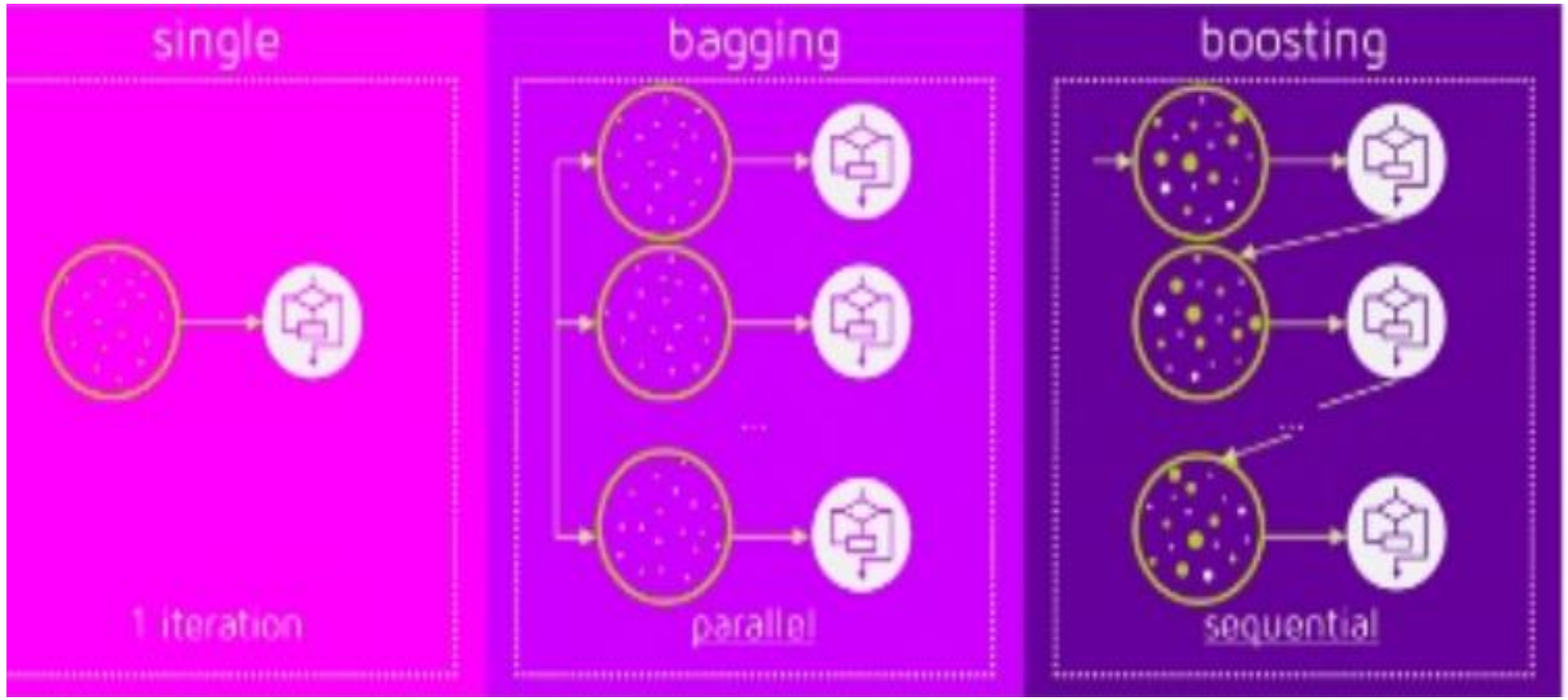
LightGBM의 리프 중심 트리분할 방식은 트리의 균형을 맞추지 않고,
최대손실 값을 가지는 리프노드를 지속적으로 분할 (한쪽만!)하면서,
트리의 깊이가 깊어지고, 비대칭적인 규칙트리가 생성됩니다.

하지만 이렇게 최대 손실값을 가지는 리프노드를 지속적으로 분할해 생성된 규칙트리는
학습을 반복할 수록, 결국은 "균형트리분할방식보다 예측오류 손실을 최소화할 수 있다"
는 것이 LightGBM의 구현 사상입니다.

Boosting 정리

알고리즘	특징	비고
AdaBoost	• 다수결을 통한 정답 분류 및 오답에 가중치 부여	
GBM	• Loss Function의 gradient를 통해 오답에 가중치 부여	gradient_boosting.pdf
Xgboost	• GBM 대비 성능향상 • 시스템 자원 효율적 활용 (CPU, Mem) • Kaggle을 통한 성능 검증 (많은 상위 랭커가 사용)	2014년 공개 boosting-algorithm-xgboost
Light GBM	• Xgboost 대비 성능향상 및 자원소모 최소화 • Xgboost가 처리하지 못하는 대용량 데이터 학습 가능 • Approximates the split (근사치의 분할)을 통한 성능 향상	2016년 공개 light-gbm-vs-xgboost

배깅과 부스팅의 차이가 뭘까?



배깅과 부스팅의 차이가 뭘까?

비교	Bagging	Boosting
특징	병렬 앙상블 모델 (각 모델은 서로 독립적)	연속 앙상블 (이전 모델의 오류를 고려)
목적	Variance 감소	Bias 감소
적합한 상황	복잡한 모델 (High variance, Low bias)	Low variance, High bias 모델
대표 알고리즘	Random Forest	Gradient Boosting, AdaBoost
Sampling	Random Sampling	Random Sampling with weight on error

한번 학습이 끝난 후 결과에 따라 가중치를 부여합니다. 그렇게 부여된 가중치가 다음 모델의 결과 예측에 영향을 줍니다.

배깅과 부스팅의 차이가 뭘까?

부스팅은 배깅에 비해 error가 적습니다. 즉, 성능이 좋습니다.
하지만 속도가 느리고 오버 피팅이 될 가능성이 있습니다.

그렇다면 실제 사용할 때는 배깅과 부스팅 중 어떤 것을 선택해야 할까요?

-> 상황에 따라 다르다고 할 수 있습니다.
개별 결정 트리의 낮은 성능이 문제라면 부스팅이 적합하고,
오버 피팅이 문제라면 배깅이 적합합니다.

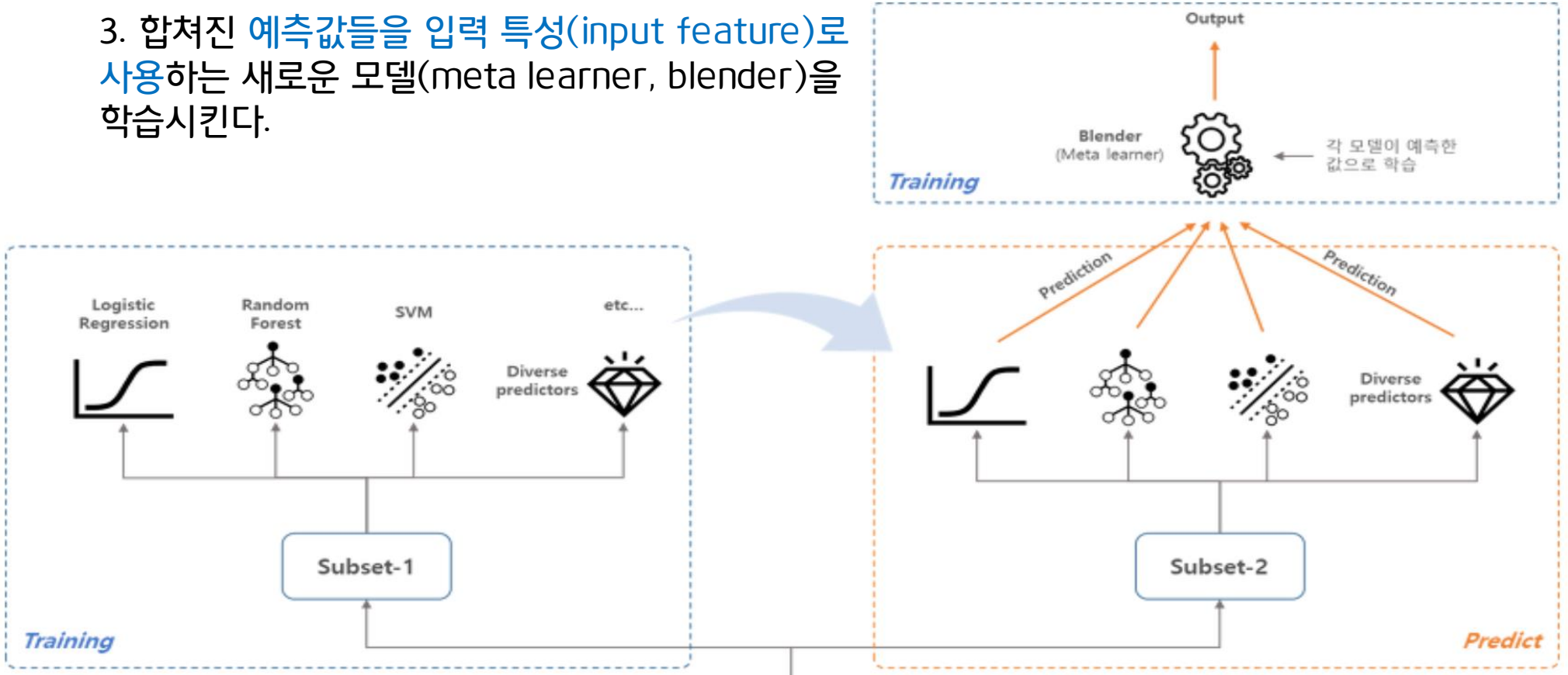
Unit 06 | 스태킹 (stacking)

Meta Modeling 이라고 불리기도 하는 이 방법은 이전 2가지 방식과는 조금 다릅니다.
“Two heads are better than one” 이라는 아이디어에서 출발합니다.

Stacking은 서로 다른 모델들을 조합해서 최고의 성능을 내는 모델을 생성합니다.
여기에서 사용되는 모델은 SVM, RandomForest, KNN 등 다양한 알고리즘을 사용할 수 있습니다. 이러한 조합을 통해 서로의 장점은 취하고 약점을 보완할 수 있게 되는 것 입니다.
(복잡성으로 인해 완벽한 파이썬 구현체는 아직 없다고함…)

Unit 06 | 스태킹 (stacking)

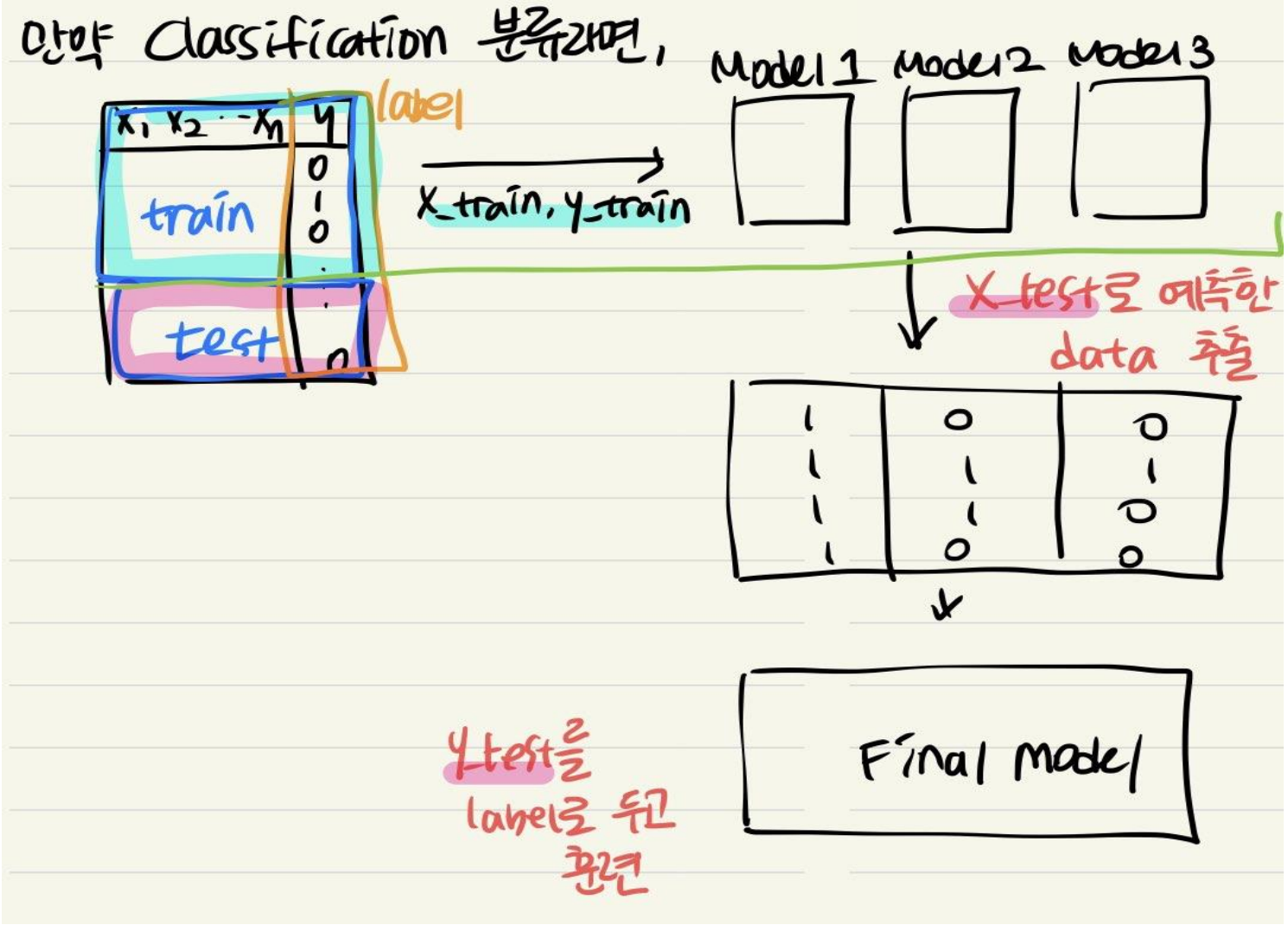
3. 합쳐진 예측값들을 입력 특성(input feature)로 사용하는 새로운 모델(meta learner, blender)을 학습시킨다.



1. 학습 데이터셋에서 샘플링을 통해 서브셋1을 만들고, 이 서브셋을 이용해 각 모델을 학습시킨다.

2. 학습 시킨 모델을 이용해 각 모델의 서브셋 2(subset-2) 예측값을 출력하고 예측값들을 합친다.

Unit 07 | 스태킹 (stacking)



Unit | 과제 (assignment)

Assignment 1. 캐글 Guide to Ensembling methods 정독
https://www.kaggle.com/amrmahmoud123/1-guide-to-ensembling-methods#

Assignment 2. 캐글 경진대회 참여 예정
https://www.kaggle.com/c/tobigs1213-assignment/data

Unit | references

Reference1 :투빅스 10기 이준걸 선배님 (앙상블 강의자료)
Reference2 : https://datascienceschool.net/view-notebook/766fe73c5c46424ca65329a9557d0918/
Reference3: Medium (Ensemble Learning - Bagging and Boosting)
Reference4: Bagging과 Boosting 그리고 Stacking
Reference5: 텐서 플로우 블로그 (결정 트리의 앙상블)
Reference6: 미디엄 (군중은 똑똑하다 - Random Forest)
Reference7: [파이썬][머신러닝][결정트리앙상블] 랜덤 포레스트

들어주셔서 감사합니다.