

## 앙상블

주어진 데이터를 ① 여러개의 서로 다른 예측 모델을 생성한 후,  
이러한 예측 모델 ② 결과를 종합하여 ③ 하나의 최종 예측 결과를 도출

### 앙상블



데이터관점에서는, 아래와 같습니다.

빨강이나 파랑이를 분류하는 이진분류 상황을 가정하겠습니다.

각 모델들의 예측값들이 있을 겁니다.

예측값들을 하나로 모은 다음에,

회귀가 목적이라 연속형 변수라면 평균을, 분류 문제면 다수결로 결과를 도출하는 것이 앙상블 입니다!

SVM	KNN	의사결정 나무	앙상블
			평균/다수결
1	1	1	1
0	1	0	0
1	0	1	1
0	1	0	0
1	0	0	0

## 배깅

배깅의 뜻은 Bootstrap aggregation입니다.

즉, bootstrap(=복원추출 사용한 표본추출 방법)을 aggregate(통합)하는 것이죠.

그냥 붓스트랩을 여러번 해서, 여러 훈련 세트를 만들고

② 각각의 훈련세트에 모델을 적용해서 그 결과 값을 평균 냄으로써, 분산을 줄이는 기법입니다.

각 훈련세트는 독립적입니다.

단일 훈련자료로부터 반복하여 표본들을 샘플링해 B개의 다른 붓스트랩 된 훈련 자료를 생성합니다.

B개의 훈련 자료를 사용하여 예측모델을 B개 만들고 그 예측 결과들을 평균을 냅니다.

이는 통계학적으로 분산을 낮춰주는 효과가 있습니다.

“예측량에 대한”

의사결정 나무는 높은 분산이 문제가 됩니다. (트리 분할이 데이터에 크게 의존하여 훈련 데이터의 작은 변화가 결정 로직에 큰 변화를 가져온다 = 불안정하다)

따라서, 의사결정나무에 배깅을 적용하면, ① B개의 붓스트랩 표본이 생성이 되고, ② 각 훈련세트에 대해 의사결정 나무를 만들고 ③ 그 결과 값을 평균을 내게 됩니다.

결국, 의사결정나무의 단점인 높은 분산이 줄어들게 되어 성능이 향상됩니다.

샘플이 한 번도 선택되지 않은 원이 여러가  
발생할 수 있는데, 전체 샘플의 약 36.8%가  
이해해라.

## 부스팅

배깅과 유사하지만, 붓스트랩 표본을 구성하는 재표본 과정에서  
분류가 잘못된 데이터에 더 큰 가중치/확률을 부여하여 표본을 추출

즉, 부스팅은 <sup>①</sup>붓스트랩 샘플링을 통해 훈련 세트를 한 번 뽑고 트리를 만듭니다.

<sup>②</sup>그 트리에서!!!!잘못 분류된 데이터를 봅니다.(더 정확히는 잔차를 봅니다.)

<sup>③</sup>그리고 그 잔차를 줄이는 방향으로 2번째 뽑히는 데이터에 가중치/확률을 부여합니다.  
이런식으로 계~~~~속 반복하는게 부스팅입니다.

한 스텝에서 계속 개선되어지니까, 학습속도가 당연히 느립니다.

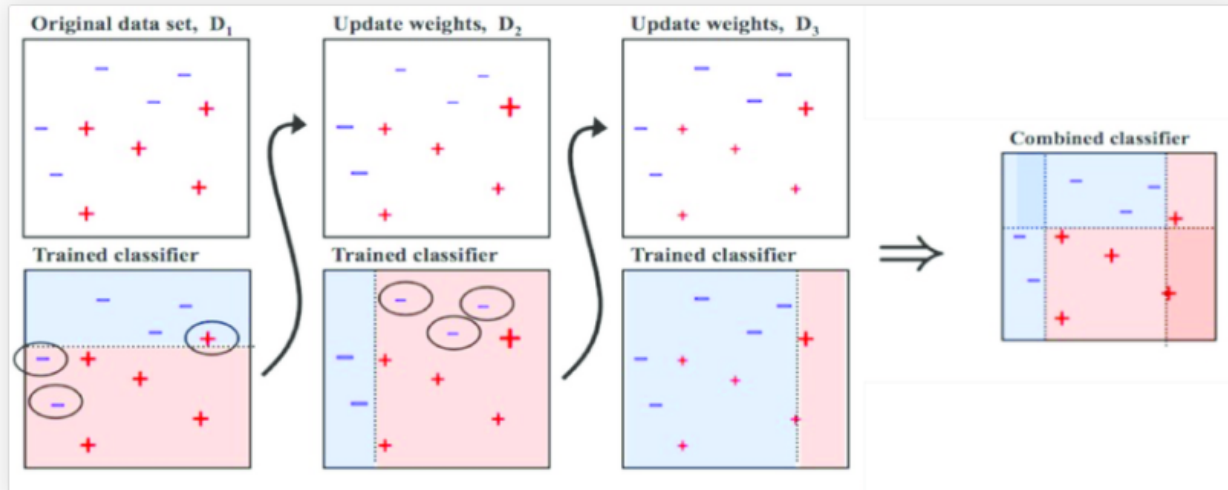
부스팅의 핵심 키워드 2개는

잘못된 데이터(잔차), 더 큰 가중치/확률 입니다!

## ensemble boosting - Adaboost

Adaboost는 앙상블 부스팅에서 가장 대표적인 방법 중 하나입니다. 어떻게 보면 제일 단순하면서도 강력한 알고리즘이죠.

Adaboost는 Adaptive Boosting으로서 이전 분류기가 틀린 부분을 adaptive하게 바꾸어가며 잘못 분류되는 데이터에 집중하도록 하는 것입니다.



출처: Medium (Boosting and Bagging explained with examples)

+와 -로 구성된 데이터셋을 분류하는 문제입니다.

D1에서는 2/5 지점을 횡단하는 구분선으로 데이터를 나누어주었습니다. 하지만 위쪽의 +는 잘못 분류가 되었고, 아래쪽의 두 -도 잘못 분류되었습니다. 잘못 분류가 된 데이터는 가중치를 높여주고, 잘 분류된 데이터는 가중치를 낮추어 줍니다.

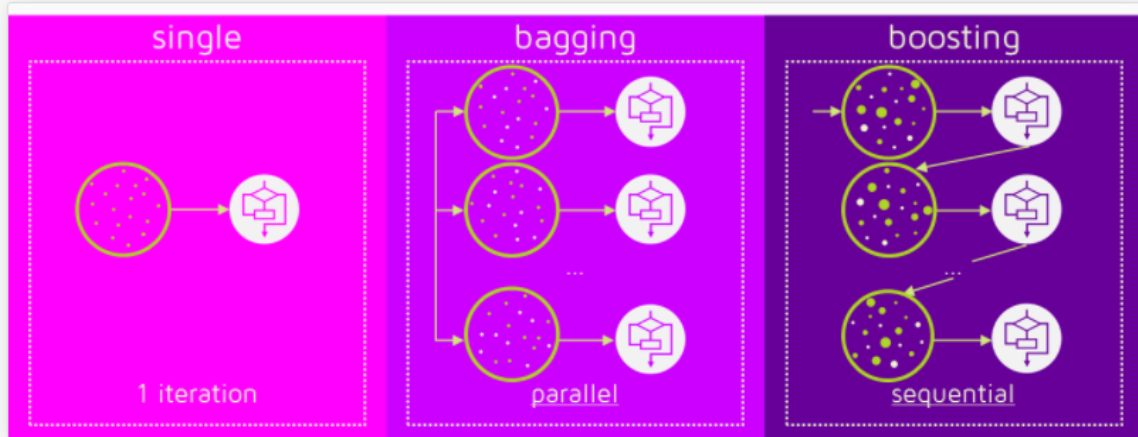
D2를 보면 D1에서 잘 분류된 데이터는 크기가 작아졌고(가중치가 낮아졌고) 잘못 분류된 데이터는 크기가 커졌습니다.(가중치가 커졌습니다.) 분류가 잘못된 데이터에 가중치를 부여해주는 이유는 다음 모델에서 더 집중해 분류하기 위함입니다.

D2에서는 오른쪽 세 개의 -가 잘못 분류되었습니다.

따라서 D3에서는 세 개의 -의 가중치가 커졌습니다. 맨 처음 모델에서 가중치를 부여한 +와 -는 D2에서는 잘 분류가 되었기 때문에 D3에서는 가중치가 다시 작아졌습니다.

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

## 배깅과 부스팅 차이



출처: [swallow.github.io](https://swallow.github.io)

위 그림에서 나타내는 바와 같이 배깅은 병렬로 학습하는 반면, 부스팅은 순차적으로 학습합니다. 한번 학습이 끝난 후 결과에 따라 가중치를 부여합니다. 그렇게 부여된 가중치가 다음 모델의 결과 예측에 영향을 줍니다.

오답에 대해서는 높은 가중치를 부여하고, 정답에 대해서는 낮은 가중치를 부여합니다. 따라서 오답을 정답으로 맞추기 위해 오답에 더 집중할 수 있게 되는 것입니다.

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.

## References

Reference1: [Medium \(Ensemble Learning - Bagging and Boosting\)](#)

## 랜덤포레스트

여러개의 의사결정나무를 결합하기 때문에, 해석·선택이 어렵다.

배깅의 일종입니다. 배깅과 다른 점은, '설명변수'도 무작위로 선택한다는 것입니다.

즉, 설명변수를 무작위로 선택함으로써, 트리의 다양성을 확보하여 모형간의 상관관계를 줄이고자 하는 것입니다. (decorrelate)

예를 들어, 1번째 샘플링에서는 변수가 a~e까지 5개가 선택되고 이를 기반으로 의사결정나무를 만듭니다.

2번째 샘플링에서는 변수가 c,w,z 3개가 선택되고 이를 기반으로 의사결정나무를 만듭니다.

3번째 샘플링에서는 변수가 a, u, v, b 4개가 선택되고 이를 기반으로 의사결정나무를 만듭니다.

이런식으로 여러개의, 서로 상관관계가 없는 트리를 막~~~~~만들어 내서

결과를 평균(회귀분석, 예측)내거나 다수결(분류)의 원칙으로 하나의 최종 결과를 보여주게 됩니다.

