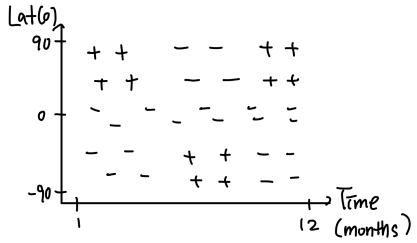
10강: Decision Trees and Ensemble Methods

Decision Trees

이전까지는 주로 linear model에 배웠고 이번 단원에는 non-linear model 중 첫번째로 decision tree에 대해 배워볼 것이다.

다음의 binary classification문제를 보자. 주어진 시간(월), 위도를 가지고 스키를 타기 좋은지 아닌지를 구분하는 분류기를 만든다고 생각해보자.

스키 타는 것이 가능한 지역과 시간에는 + 표기를 하고 불가능한 지역에 -를 표시하면 아래와 같다.



이 경우에는 linear한 decision bounary를 얻기 힘들 것이다. svm을 통해 decision boundary를 얻는다고 하더라도 베스트한 선택이라고 볼 수는 없다.

그렇다면 전체 공간을 개별 공간으로 partition하는 **decision tree**를 선택하는 것이 좋을 것이다.

그렇다면 decision tree는 어떻게 region을 나누는 것일까?

정답은

> **Greedy, Top down, Recursive partitioning**

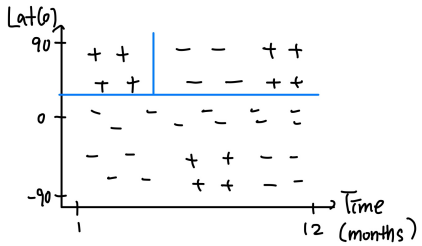
- Greedy: 모든 케이스를 측정하여 best partition을 선택하기 때문
- Top-down: 전체 데이터를 가지고 서서히 분류하기 때문
- Recursive: 데이터를 분류하고 나서 분류된 한 부분을 가지고 다시 분류하기 때문

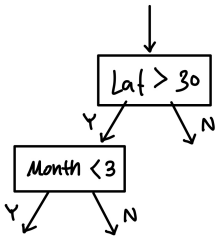
우리의 예시에 돌아와서 적용해보자

첫번째로는 위도가 30도 인지 아닌지로 분류할 수 있다. 그래서 위도가 30도 보다 큰지역과 작은 지역으로 나눠진다.

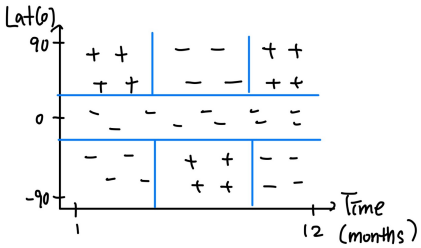
두번째로는 시간(월)을 가지고 분류할 수 있는데 3월보다 작은 달인지 아닌지로 구분할 수 있다.

이렇게 나눠진 그림은 다음과 같다.





이러반 방법으로 계속 분류하면 다음과 같은 결과를 얻는다.



이를 공식으로 나타내면 다음과 같이 나타낼 수도 있다.

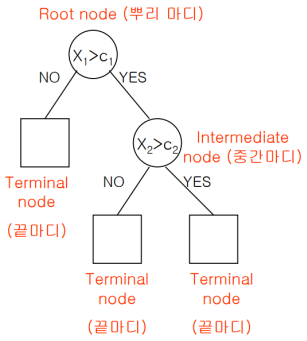
$$
\begin{gathered}
Region\ R_p \\
Looking\ for\ a\ split\ S_p \\
S_p(j,t) = (\{x|x_j < t,\ x \in R_p\}, \\
\{x|x_j \geq t,\ x \in R_p\})
\end{gathered}
$$

- 입력변수:
  j는 feature number, t는 임계값을 의미한다.
- R_p: region parent 상위/ 부모지역
- 해석:
  j에 해당하는
  x_j값이 임계값 t 보다 작을 때는 첫번째 지역 R_1에 할당한다.
  j에 해당하는
  x_j값이 임계값 t 보다 크거나 같은 경우에는 두번째 지역 R_2에 할당한다.
  여기에서
  R_1과 R_2는 children region이라고 한다.

(1) 추가적으로,, decision tree 모델 소개

의사결정나무는 데이터를 분석하여 이들 사이에 존재하는 패턴을 예측 가능한 규칙들의 조합으로 나타내며, 그 모양이 '나무'와 같다고 해서 의사결정나무라 불린다.

의사결정 나무를 일반화한 그림은 다음과 같다.



- root node
- terminal node
- X1, X2 : split variable
- c1, c2 : split point

초기지점은 root node이고 분기가 거듭될 수록 그에 해당하는 데이터의 개수는 줄어든다. 각 terminal node에 속하는 데이터의 개수를 합하면 root node의 데이터수와 일치한다. 바꿔 말하면 terminal node 간 교집합이 없다는 뜻이다.

한편 terminal node의 개수가 분리된 집합의 개수다. 예컨대 위 그림처럼 terminal node가 3개라면 전체 데이터가 3개의 부분집합으로 나눠진 셈이다.

How to choose splits?

그렇다면 우리는 어떤 기준으로 분류를 할 수 있을까? 이를 위해서는 분류가 얼마나 틀렸는지 계산하는 로스를 구해야 한다.

응 교수님 대신 들어온 어떤 분은 loss를 다음과 같이 정의했다.

> Define $L(R)$: loss on R
> Given C classes,
> define
> $\hat{P}_c$ to be the proportion of examples in R that are of class C .
>
> $$L_{misclass} = 1 - max_c \hat{P}_c$$

해석해보자면 $L(R)$은 지역 R의 loss이고, $\hat{P}_c$은 지역 R안에 있는 example 중에서 classC에 속하는 비율을 의미한다.

따라서 이를 위의 식에 대입하면 얼마나 잘못 분류했는지를 의미하는 $L(R)$ 즉, missclassification loss를 구할 수 있다. 우리는 이러한 loss를 줄이는 줄이는 방향으로 split을 선택해야 한다.

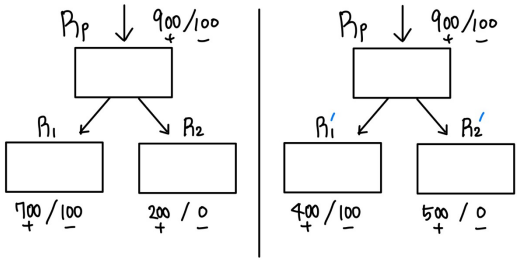
이를 일단 수식으로 표현하면 다음과 같다.

$$max_{j,t}\ L(R_p) - (L(R_1) + L(R_2))$$

이때 parent loss는 이미 정의된 것으로 고정되어 있다. 따라서 뒤의 부분인 children loss를 줄이는 방향으로 진행해야 한다. 이때 위의 식에서는 childern loss에 (-) 부호가 있기 때문에 전체 식을 최대화 하는 방향으로 split을 선택하게 된다.

Misclassification loss has Issues

그러나 우리가 구한 misclassification loss 계산에는 문제가 있다. 아래의 두 그림을 보면, 실제로 다르게 분류했음에도 불구하고 loss는 두 경우 모두 100으로 나왔다. 심지어 부모 노드의 loss도 100으로 같다.

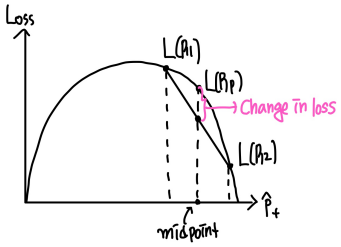


$$L(R_1) + L(R_2) = 100 + 0 = 100$$
$$L(R_1') + L(R_2') = 100 + 0 = 100$$
$$L(R_p) = 100$$

그래서 이러한 문제를 해결하기 위해서 단순 loss가 아닌 **cross entropy loss**를 사용하게 된다.

> Instead, define cross-entropy loss
>
> $$L_{cross} = -\sum_c \hat{P}_c log_2 \hat{P}_c$$

cross entropy loss는 $L(R_1)$과 $L(R_2)$의 평균에 해당하는 지점과 곡선과 직각이 되는 점 $L(R_p)$ 의 차이(Gain)를 계산하는 것이다.

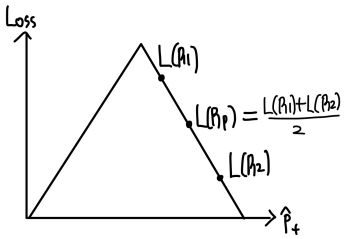


위의 그래프를 보면 부모 노드에서 발생한 loss 보다 자식 노드 2개에서 발생한 loss의 평균이 더 작은 경우인데 Change in loss로 적혀있는부분이 Gain이다.

이것은 Gini 값이라고 해서 아래와 같이 정의된 값으로도 많이 쓰인다.

> **Gini**
>
> $$gini = \sum_c \hat{P}_c(1 - \hat{P}_c)$$

추가적으로 misclassification loss도 그래프로 나타내보면 다음과 같다.

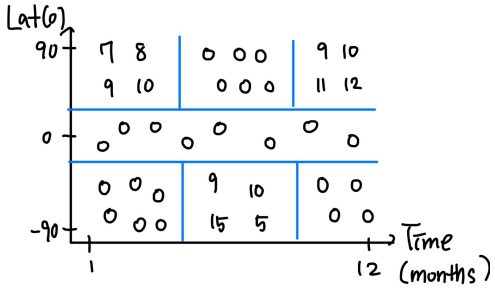


부모 노드의 loss 가 자식 노드 2개의 평균과 같아지기 때문에 자식 노드에서는 의미 없는 분류를 한 것과 같다.

(2) Regression Tree

Decision tree는 regression에서도 적용이 가능하다. 위에서 Decision trees의 예로 사용했던 상황을 또 다시 이용하여 예를 들고자 한다. 위에서는 스키를 타기에 적합한지 아닌지에 대해 구분하는 문제(classification)였는데 이번에는 적설량을 구분하는 문제(regression)로 바뀌었다.

decision tree에서와 같이 위도와 시간(월)에 따른 위치의 적설량을 숫자로 표시한다. 그리고 이것을 잘 분류하게 된다면 아래와 같이 구분선이 그려지게 될 것이다.



- prediction

이 경우에는 지역 R_m에 해당하는 예측값 $\hat{y_m}$을 수식으로 나타내면 다음과 같다.

$$\hat{y_m} = \frac{\sum_{i \in R_m} y_i}{|R_m|}$$

이것은 지역 R_m에 있는 모든 수의 합을 숫자의 개수로 나누어 계산하는 값이며, 결국 해당 지역의 평균값이 된다.

- loss

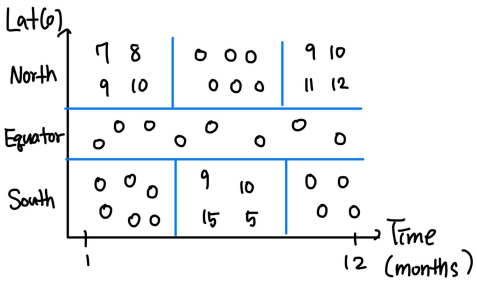
그 다음 loss는 MSE(Mean squred error)와 같이 각 예측값에서 평균값을 뺀 값의 제곱을 누적하여 더하여 전체 개수로 나누어 구한다.

$$L_{squared} = \frac{\sum_{i \in R_m} (y_i - \hat{y}_m)^2}{|R_m|}$$

(3) categorical variable

decision tree는 categorical variable을 가지고 모델을 만들 수 있다. 그러나 q개의 카테고리가 있다고 가정하면 2^q만큼의 possible split이 생기기 때문에 너무 복잡하고 비효율적이다.

그림으로 나타내면 다음과 같은데 왼쪽을 보면 North, Equator, South로 카테고리를 분류하였다.



참고로 이는 이진트리(binary tree) 의경우에도 적용할 수 있다.

(4) Regularization of Decision Trees

트리 모델을 모든 훈련데이터에 대해 훈련시키면 결국 모든 training case를 만족시키는 트리를 만들어야 한다. 이는 결국 최종 결과가 하나씩 연결과는 overfitting 문제를 발생시킨다.

이를 방지하기 위해 regularization이 필요한데 여기에 사용되는 다양한 휴리스틱 정규화 방법은 아래와 같다.

> Regularization of DTs
> 1) min leaf size : 최소 리프 사이즈를 제한
> 2) max depth : 트리 깊이를 제한
> 3) max number of nodes : 노드들의 최대 개수 제한
> 4) min decrease in loss : 최소로 줄어드는 로스 정함
> 5) pruning(misclassification with val set): 검증데이터에서 잘못 분류된 가지를 제거

특히, 4) min decrease in loss의 경우 부모노드보다 자식노드가 최소 어느정도는 더 분류를 잘해야 하는 지를 정하는 방법으로, data feature의 중요도 즉, 잘 분류하는 데이터의 feature에 따라서 위에서 아래로 잘 정리되게 만드는 방법이다.

(5) decision tree의 장점과 단점

- 장점

1. 설명하기 쉽다.
2. 모델을 해석하기 쉽다.
3. categorical variable를 다룰 수 있다.
4. 속도가 빠르다

- 단점

1. 분산이 크기 때문에 training data에 overfitting우려
2. 변수 사이에 상호작용이 작은 경우에 분석에 약하다. (반대로 상호작용이 큰 경우에는 강하다)
3. 예측정확도가 낮다

Ensemble Methods

위에서 단일 트리의 경우에는 예측정확도가 낮다는 단점이 있다고 언급했는데 이러한 단일 트리의 단점을 보완할 수 있는 방법이 앙상블 기법이다.

먼저 다음의 가정과 모집단의 평균에 대한 분산에 대해 살펴보자.

> Take x_i's which are random variables(RV)
>
> that are indepenent identically distributed(iid)
>
> $$Var(x_i) = \sigma^2 \qquad Var(\bar{x}) = Var(\frac{1}{n}\sum_i x_i) = \frac{\sigma^2}{n}$$

이를 보면 x의 개수가 늘어날 수록 모델의 variance가 줄어든다는 것을 알 수 있다.

여기서 독립 가정을 제거하면 전체 모집단의 평균에 대한 분산은 아래와 같이 계산할 수 있다.

> Drop independence assumption
> So, now
> x_i's are i.d
>
> x_i's correlated by ρ
>
> $$Var(\bar{x}) = \rho\sigma^2 + \frac{1-\rho}{n}\sigma^2$$

분산의 공식을 이해해보면

먼저 왼쪽 term은 correlated한 경우에 생겨나는데 ρ 가 작아지면 이 term도 줄어든다. 따라서 completely decorrelated해지면 분산은 iid 경우와 동일해진다.

오른쪽 term은 데이터 수가 증가하면 작아지는 term이다.

따라서 분산은 ρ에 의해 연관된 x sample이 많을수록 분산이 커지고 데이터 수가 클수록 분산이 작아진다.

이렇게 단일 트리모델이 분산이 크고 정확도가 낮다는 단점을 보완하기 위해 등장한 방법 중 하나가 앙상블이며 이에는 다양한 방법이 존재한다.

Ways to ensemble

1) different algorithms

2) different training sets

3) Bagging (대표적인 예 Random Forests)

4) Boosting (대표적인 예Adaboost, xgboost)

1), 2) 를 사용할 경우 새로운 trainin data set을 얻거나 새로운 algorithmn을 training해야 한다는 번거로움 때문에 이 대신 3),4)을 주로 사용한다.

Bagging

배깅의 풀네임은 **B**ootstrap **agg**regat**ing** 이다. 말 그대로 bootstrap을 통해서 다양한 데이터 셋을 만들고 이를 학습시킨 모델을 모으는(Aggregating)방법이다.

(1) bootstrap

have a true polulation P

training set $S \sim P$

Assume $P = S$

Bootstrap Samples $Z \sim S$

실제 모집단 P에서 샘플 S를 뽑아서 training set으로 사용한다. 이때 모집단과 샘플이 같다고 가정한다. 그리고 부트스트랩 통해 샘플에서 다시 샘플 Z를 추출하여 이것을 훈련에 사용하게 된다.

(2) Aggregating

Bootstrap Samples $z_1, \ldots, z_m$

train model G_m on z_m

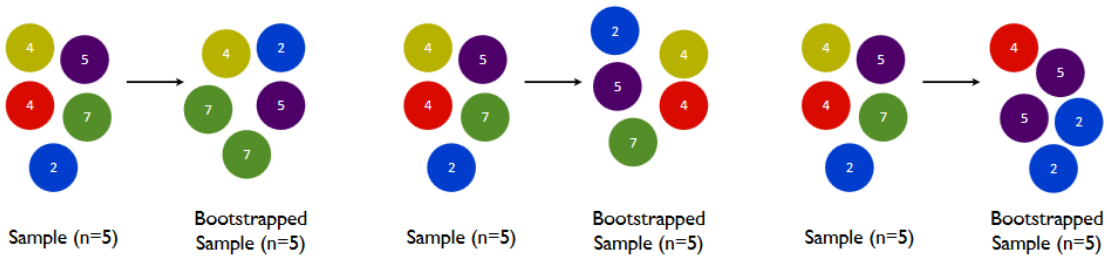
$G(m) = \frac{\sum_m^M G_m(x)}{M}$

이렇게 생성된 여러개의 샘플 z를 통해 모델을 여러번 훈련시킨 결과를 평균내어 최종 결과를 도출할 수 있다.

강의에는 없지만 부트스트랩부터 좀 더 자세하게 설명해보려고 한다.

- bootstrap

부트스트랩은 모집단으로부터 표본을 추출하는데 여기서 중요한 특징은 **복원추출(Sampling with replacement)**을 한다는 점이다.



이 예시는 모집단으로부터 3개의 부트스트랩 샘플을 얻는 결과이다. 각 x는 균일한 확률로 선택되기 때문에 중복되어 선택될수도 있고 선택되지 못할 수도 있다.

- OOB

각 부트스트랩에서 선택되지 못한 x를 oob(out of bag)라고 한다. 특정 x가 oob가 될 확률은 아래식을 통해 구할 수 있다.

$p = (1 - \frac{1}{N})^N$

N이 커질수록 아래의 식에 의해 이 값은 약 37% 정도에 가까워진다.

$$\begin{aligned} \lim_{N\to 0}\left(1-\frac{1}{N}\right)^N &= \lim_{N\to 0}\left[\left(1-\frac{1}{N}\right)^{-N}\right]^{-1} \\ &= e^{-1} \qquad \left(\because e = \lim_{n\to 0}\left(1+\frac{1}{n}\right)^n\right) \\ &= 0.3679 \end{aligned}$$

즉, N이 일정수준 이상으로 커지면 매 표본집단에서 모집단의 약 2/3은 포함되고, 나머지는 1/3은 OOB가 된다. OOB는 각 모델의 학습 데이터로 사용되지 않기 때문에 나중에 검증데이터로 사용하게 된다.

- effect

부트스트랩은 데이터의 분포를 변형하는 효과가 있다. 원래 데이터의 노이즈 ϵ가 특정 분포를 따르고 있다면 이를 통해 만드는 모델은 분포에 종속될 수 밖에 없다. 그러나 부트스트랩을 통해 분포를 다양하게 만들어 주면 특정 분포에 종속된 모델이 만들어지는 것을 방지함으로써 다양성을 확보할 수 있다.

게다가 OOB 데이터를 검증에 사용하면 모든 샘플을 학습과 검증에 활용하여 높은 검증력을 확보할 수 있다는 효과도 있다.

- learning

지도학습 알고리즘이라면 어떤 것도 배깅에 적용할 수 있다. 하지만 배깅의 목적이 데이터셋을 다양화하여 분산(Variance)을 줄이기 위함이기에 일반적으로는 **복잡도(Complexity)가 높은 모델**을 선택한다. 복잡도가 높은 모델은 분산에 의한 오차가 높기 때문에 배깅을 적용했을 때 의미있는 성능 향상 확률이 높기 때문이다. 복잡도가 높은 모델로는 의사 결정 나무(Decision Tree), 인공 신경망(ANN), 서포트 벡터 머신(SVM) 등이 있다.

- Result Aggregating

학습이 끝나면 결과를 취합해야 한다. 수많은 취합방법이 존재하는데 대표적인 방법으로는

1) majority voting (다수결) : N개의 모델 중 가장 많이 선택된 레이블을 결과로 도출

2) Weighted Voting (at Validation Accuracy) : 검증정확도에 가중치를 두어 투표하는 방법이다. 검증정확도가 높은 모델이 테스트 데이터에 대해 잘 판단할 것이라는 아이디어에서 시작되었다.

3) Weight voting (at Predicted probability) : 테스트의 인스턴스의 레이블을 판단하는 확률에 가중치를 두는 방식이다. 각 모델의 검정력은 동일하다고 보되 더 강하게 주장하는 모델의 결과에 가중치를 주자는 아이디어에서 시작되었다.

(3) bias-variance analysis

이렇게 부트스트랩을 적용하면 오차와 분산에 변화를 주게 된다. 앞서 분산을 정리한 것을 이에 적용하면 다음과 같다.

$$Var(\bar{x}) = \rho\sigma^2 + \frac{1-\rho}{M}\sigma^2$$

> Bootstrapping is driving down ρ
> Make M → less variance
> Bias slightly increased because of random subsampling

(4) decision trees+Bagging

위에서 언급한 것과 같이 decision tree는 높은 분산에 낮은 bias를 가지고 있기 때문에 분산을 낮추는 bagging과 잘 맞는다. 이때 bias는 약간 증가하게 되는데 variance가 훨씬 더 감소하기 때문에 무시해도 된다.

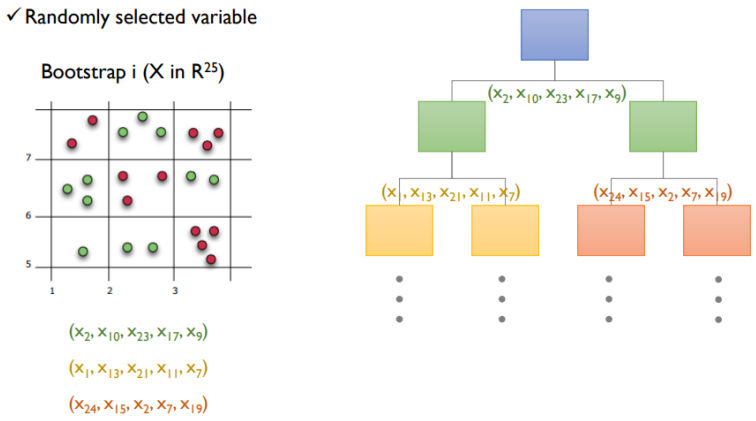
Random Forests

랜덤 포레스트는 배깅의 특수한 형태인데 기본모델로 decision tree를 사용한다. 다수의 의사 결정 나무를 결합 모델이기에 forest라는 이름이 붙었다.

- random

원래 의사 결정나무가 결정경계를 탐색할때 모든 변수를 고려하여 정보 획득량(information gain, IG)이 가장 높은 지점을 찾아낸다. 그러나 랜덤 포레스트는 random하게 일부 변수만들 선택하여 정보 획득량이 최고인 지점을 탐색한다.

아래는 25개의 변수 중에서 임의로 5개의 변수를 택할 때 변수가 어떻게 선택되는 지를 나타낸 그림이다..



- effect

decision tree+bagging을 사용할 경우 부트스트랩 샘플을 만들 때 복원추출을 하기 때문에 데이터가 겹치는 문제로 인해 각 tree모델마다 positive correlation이 존재한다.

random foreset는 매번 random하게 변수를 선택하기 때문에 더 다양한 모델을 만들어 낼 수 있고 각 모델마다 correlation이 작다.

- **Generalization Error**

랜덤 포레스트는 트리의 개수(population size)가 일정 수준 이상으로 커졌을 때 일반화 오차(generalization error)를 구할 수 있다. 일반화 오차의 상한선은 다음과 같이 결정된다.

$Generalization\ error \leq \frac{\bar{\rho}(1-s^2)}{s^2}$

$\bar{\rho}$: 각 트리 사이의 상관계수를 구한 뒤 평균을 구한 값

s^2: 마진값

개별 트리가 정확할수록

s^2이 커지기 때문에 일반화 오차는 감소한다. 그리고 트리 사이의 상관관계가 적을수록

$\bar{\rho}$ 가 줄어들어 일반화 오차가 감소하게 된다.

- **Permutation Importance**

랜덤포레스트를 통해 변수 중요도를 측정할수 있다.

방법은 다음과 같다.

1. 아무것도 건들지 않았을때는 oob error를 측정한다. 이 값을 e_i로 나타낸다.
2. 중요도를 측정하고자 하는 변수를 섞어준 뒤에 oob error를 측정한다. 이값은 p_i로 나타낸다.
3. 만약 결정경계를 형성할 때 해당 변수를 많이 사용했다면 p_i와 e_i값은 커질 것이다. 따라서 모든 트리에 대해 $p_i - e_i$값을 구해주어 중요도를 측정한다.

특히 m 번째 트리에서 두값의 차이를 $d_i^m = p_i^m - e_i^m$라고 하면 i번째 변수가 중요할수록 d_i의 평균이 증가하고 분산이 작아진다. 따라서 해단변수의 중요도 v_i를 아래와 같이 구할 수 있다.

$$\bar{d}_i = \frac{1}{m}\sum_{i=1}^{m} d_i^m, \quad s_i^2 = \frac{1}{m-1}\sum_{i=1}^{m}(d_i^m - \bar{d}_i)^2$$

$$\therefore v_i = \frac{\bar{d}_i}{s_i^2}$$

이렇게 구한 v_i는 상대적인 중요도를 나타낼뿐 절대적인 의미는 없다. 따라서 변수선택과 구분해서 사용해야 한다.

Boosting

마지막으로는 앙상블 방법중 하나인 boosting에 대해 알아볼 것이다.

- Weak model

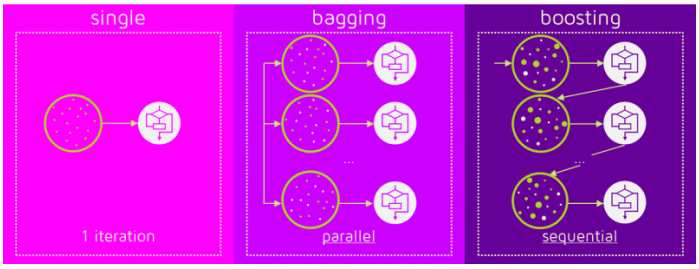
부스팅을 이해하기 위해 weak model에 대해 알아보자. weak model이란 말그대로 성능이 낮은 모델을 의미한다. 약한모델의 정확도는 60%정도라도 한다.

부스팅은 약한모델을 결합해 임의의 강한 모델의 성능 이상으로 끌어올릴(boosting)수 있을 것이라는 아이디어로부터 시작했다.

- weighted on wrong cases

틀린 문제를 중심으로 학습해야 점수가 오르는 것처럼 부스팅 알고리즘도 약한 모델이 학습결과를 보이면 이를 바탕으로 약한 부분을 공부하는 과정을 반복하면서 학습한다.

아래 그림과 같이 순차적으로 진행되기 때문에 병렬처리가 불가능하다. 그러나 stump tree(결정 경계를 하나만 형성하는 의사결정나무)와 같은 약한 모델을 기반으로 하므로 결정 경계가 없어도 (복잡도가 높은 모델을 사용하는 배깅)보다 빠르게 학습이 진행된다.



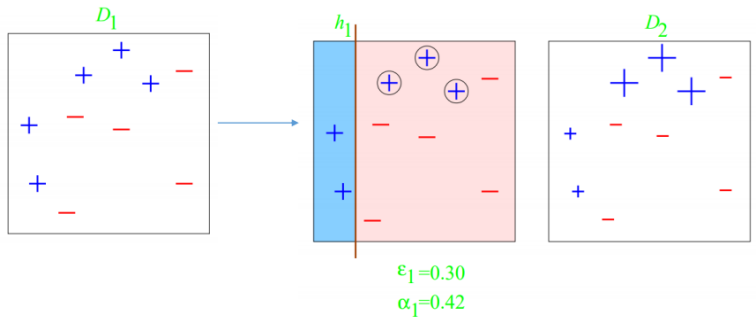
Adaboost

▼ 강의에는 없는데 작성중

틀리게 예측한 인스턴스에 가중치(Weight)를 부여한다는 아이디어를 기본으로 한다.

svm에서와 마찬가지로 이진분류문제에서 label이 -1,1 이다.

예시를 통해 작동원리를 살펴보자.



첫번째 데이터 셋 D_i에 대한 학습 결과 hi가 나왔다.

이를 바탕으로 ϵ_t와 α_t를 계산한다.

- ϵ_t: 훈련 데이터셋 중에서

h_t가 틀리게 판단한 데이터의 비율

- 가중치 α_t:

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon}{\epsilon}\right)$$

참고로 랜덤추측보다 성능이 떨어지는 경우 $\epsilon > 0.5$ 인 경우에는 break된다.

거의 모든 데이터가 제대로 판단되면 가중치값은 매우 커지고 랜덤 추측과 비슷한 상태면 가중치값은 0에 가까워진다.

이렇게 구해진 ϵ_t와 α_t로 $t+1$번째 학습에 사용될 데이터의 분포를 구하는데 사용된다.

$$D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

출처. 응교수님의 cs229 강의

Lecture 10 - Decision Trees and Ensemble Methods | Stanford CS229: Machine Learning (Autumn 2018)

https://youtu.be/wr9gUr-eWdA

이분 짱짱👍

부스팅(Boosting)과 Adaboost(Adaptive Boosting)

해당 게시물은 고려대학교 강필성 교수님의 강의 를 바탕으로 작성한 것입니다. 이번에는 앙상블 방법 중 하나인 부스팅(Boosting)에 대해서 알아보고 부스팅의 시초격인 Adaboost(Adaptive Boost ing)에 대해서 알아보도록 하겠습니다. 부스팅을 이해하기 위해서는 약한 모델(Weak Model)에 대해서 알아야 합니다. 약한 모델이란 말 그대로 성능이 낮은 모델을 가

https://yngie-c.github.io/machine%20learning/2021/03/20/adaboost/

$\exp(-\alpha_t y_i h_t(x_i))$	성능 좋은 분류기 $(\alpha_t \gg 0)$	그저 그런(?) 분류 $(\alpha_t \approx 0)$
$y_i = h_t(x_i)$ $[y_i h_t(x_i) = 1]$	≈ 0	< 1
$y_i \neq h_t(x_i)$ $[y_i h_t(x_i) = -1]$	$\gg 1$	> 1