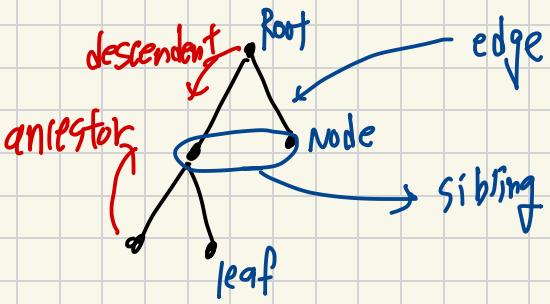
〈Tree〉



〈Regression Tree〉

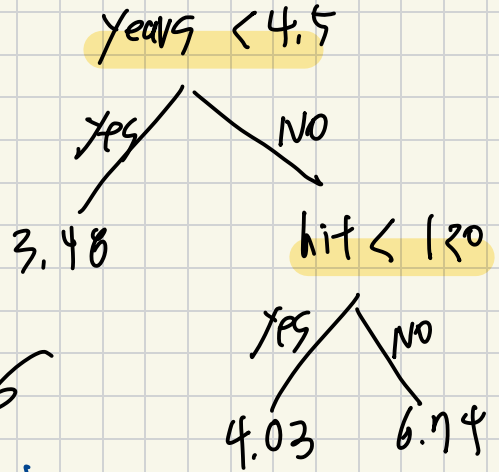
$X \in \mathbb{R}^d$인 vector이다.

우선 매 관찰에 $X \in$ {Hit, Years} 2개의 variable을 갖는 경우 예.

예를 들어 샘플 = {140, 20} 처럼 표현이 가능하다.

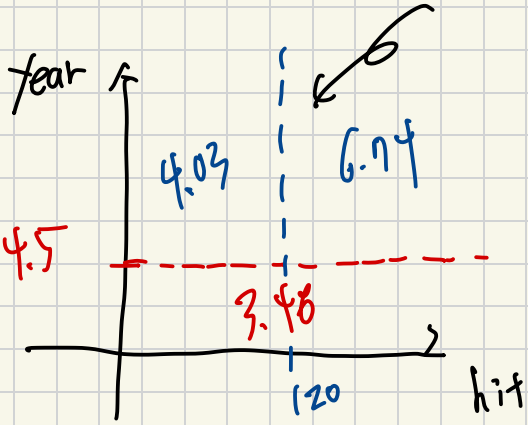
그리고 tree의 node의 경우 $x_j < t_k$ 등으로 표현이 된다.

example)



Internal node : 조건문

Leaves node : 결과.



Leaves node 수 만큼 split 된다.

→ 너무많은 경우의 수가 가능함을 제공한다.

〈Building Regression Tree〉

최종 목표: Non-overlapping하게 Area를 $R_1, R_2 \cdots R_N$으로 나눈다.

같은 영역 값들은 모두 Same 상수로 결정. → 너무 simple?

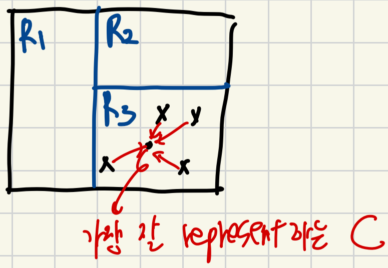
만일 더 detail한 Regression 필요시 leaves node를 늘리면 된다. → 해결.

Regression은 푸는 경우

$$\min \sum_{j=1}^{J} \sum_{i \in R} (y_i - \hat{y}_j)^2$$ 을 만족해야한다.

그래서 각 Region마다 그 차이를 minimize하는 대표값 C 설정.

$$\hat{y}_j = \underset{c}{\arg\min} \sum_{i \in R_j} (y_i - c)^2$$



↳ MLE 풀어보면 결국 R에 포함된 data들의 평균이 나오게 된다.

평균은 데이터들의 오차 제곱을 최소화 해주는 하나의 대푯값이다.

절대값의 오차를 최소화 해주는 대표값은 중앙값이다

〈Top-down Greedy Algorithm〉

root에서부터 하나씩 최선의 selection을 반복해 가는 방법으로 tree를 생성한다.

그러면 어느 variable X를 어느 기준 S와 비교해서 나눌지를 정해야한다.

⇒ 모든 $X \in \{X_1, X_2 \sim X_p\}$에 대해 모든 S를 고려해서

$$\sum_{x_i \in R_1} (y - \hat{y}_{R_1})^2 + \sum_{x_j \in R_2} (y - \hat{y}_{R_2})^2$$ 를 기준 이득이 가장 큰 X, S 설정.

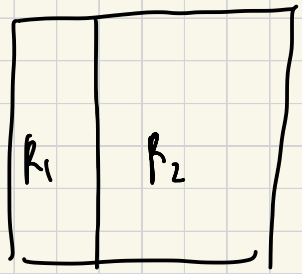
s 선별 (1) X_j를 오름차순 정렬

(2) X_{jt}와 $X_{j,t+1}$ 사이의 중간값을 s_t로 사용

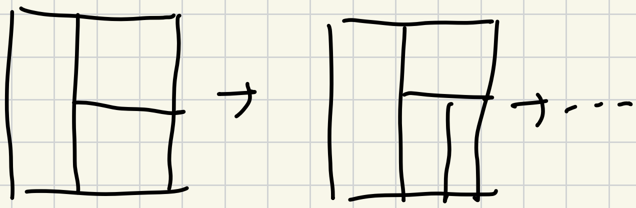
(3) 부등호는 $\leq s$ 아 $< s$ 사용 (분리의 기반만 하면 된다)

* 주로 1개의 Region에 k개의 data가 남을 때 까지 반복한다.

(1)



(2) R_1을 split할지 R_2를 split 할지도 결정 ⇒ 모든 경우를 다 따져서 greedy 하게 $n \to n+1 \to \cdots$ 하게 Region을 늘려 나간다.



⟨Tree pruning⟩

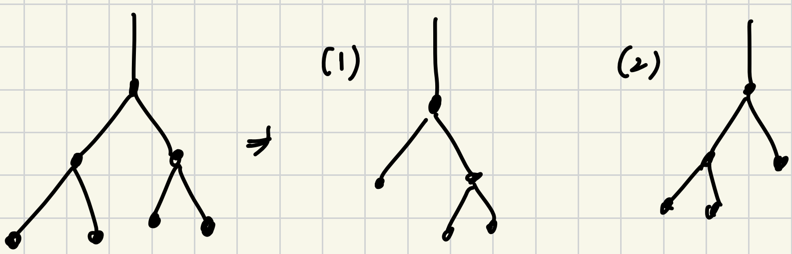
Region 내에 data point가 ↓ 이면 overfitting이 발생한다.

→ Boundary를 smooth하게 (딱 k번만 split) ⇒ Tree는 greedy 이기에 언제 Model이 성능이 좋아질지 모름 (무조건 다 해봐야 model 최적 파악 가능)

그래서 우선 split 한 후 & 다시 Region 을을 합쳐가는 방법이 pruning.

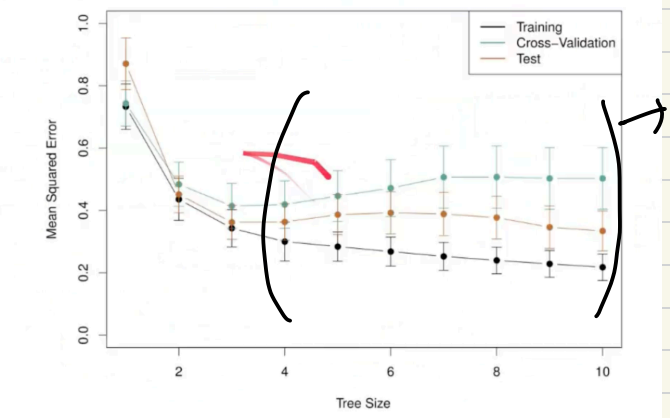
$$\sum_{m=1}^{|T|}\sum_{x_i \in R_m}(y_i-\hat{y}_{R_m})^2+\alpha|T|$$

$|T|$ → leaves node의 갯수.



(1), (2)를 compare.

각 대표값 1개 찾아서 (2 group 평균) 그 중의 RSS가 min인 leaves node를 merge한다.



→ over fitting 발생.

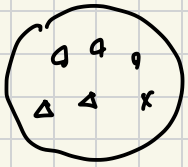
즉, leaves node = 3이 될때까지 pruning진행.

〈 Classification Tree 〉

y가 Categorical ⇒ Region을 다수결 select으로 변경.

(1) Region split하기

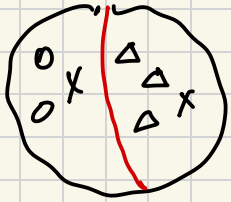
(2) Representation value구하기



o : $\frac{8}{8}$ △ : $\frac{2}{8}$ x : $\frac{1}{8}$ 을 구해서 o이 대표.

Split시 misclassification ratio를 minimize.

"Impurity" : R에 최대한 같은 class가 많이 되도록.



minimize)

$$\frac{1}{n_1}\text{Impurity}(R_1(V,s)) + \frac{1}{n_2}\text{Impurity}(R_2(V,s))$$

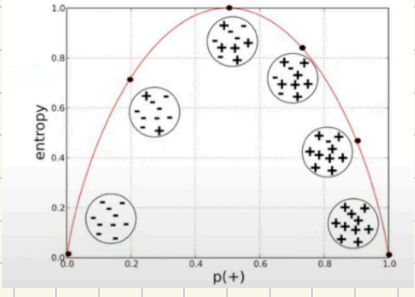
〈 Impurity 〉

$$\text{Impurity}(R) = -\sum_{k=1}^{n} P_{j,k} \log P_{j,k}$$

$$\text{Impurity}(R_j) = -\sum_{k=1}^{n} p_{j,k} \log p_{j,k}$$

- This is called entropy.
- Illustration by example:
  - [1, 0, 0, 0, 0] → -1 log 1 = 0 // 만장일치
  - [0.5, 0.5, 0, 0, 0] → -2 * 0.5 log 0.5 = log 2
  - [0.2, 0.2, 0.2, 0.2, 0.2] → -5 * 0.2 log 0.2 = log 5

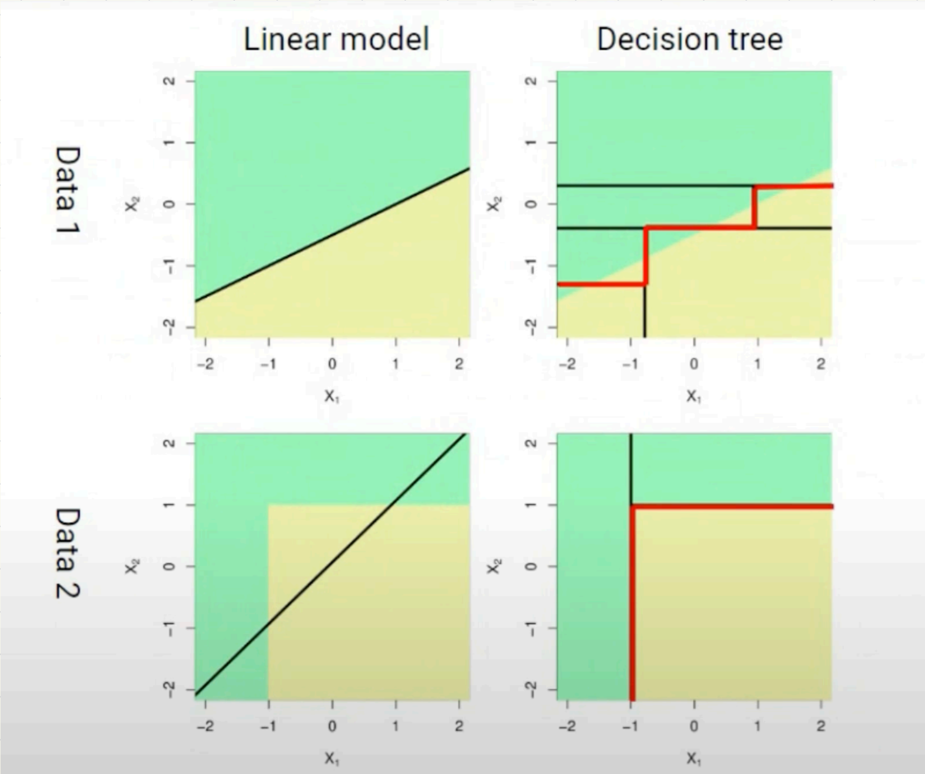
↓ 복잡도 증가.



〈 Gini-index 〉

$$G = \sum_{j=1}^{K} p_{j,k}(1 - p_{j,k})$$

- Illustration by example:
  - [1, 0, 0, 0, 0] → 0
  - [0.5, 0.5, 0, 0, 0] → 2 * 1/4 = 0.5
  - [0.2, 0.2, 0.2, 0.2, 0.2] → 5 * 0.2 * 0.8 = 0.8
- Gini index and the entropy are very similar numerically.



실제로는 Data의 원래 분포와 비슷하게 더 잘린다.
But Decision Tree가 더 비선형적 특징을 배운다

But Tree는 axis alline 되어 분류하는 단점이 존재한다.
(장단점이 존재)

Tree's pro : (1) 설명하기 쉽다
(2) 시각화가 가능하다.

Tree's cons : (1) 성능이 높지 않다.
(2) Robust 하지 않다 → Train에만 잘 작동....! (치명적)
→ Data가 추가되면 tree의 모양이 너무 많이 change 된다.