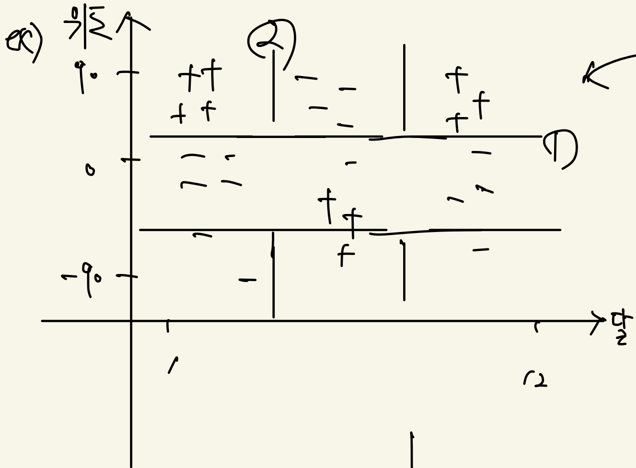
* Decision Trees

↳ 대표적인 비선형인 Model

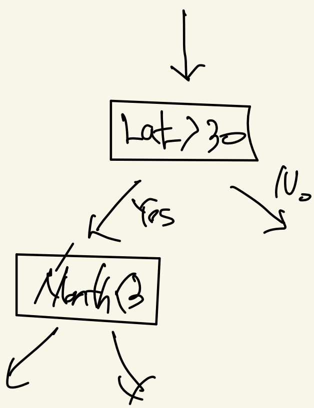
ex)



← 선형 Model로는 분류가 어려움.

Solve 방법

Greedy, Top-Down, Recursive Partitioning



→ Region R_p (분할 집합)

Looking for a split S_p

$$S_p(j, t) = \underbrace{(\{X \mid X_j < t, X \in R_p\}}_{R_1}, \underbrace{\{X \mid X_j \geq t, X \in R_p\})}_{R_2}$$

기능 (j), 임계값 (t)

How to choose splits?

→ Define $L(R)$: loss on Region

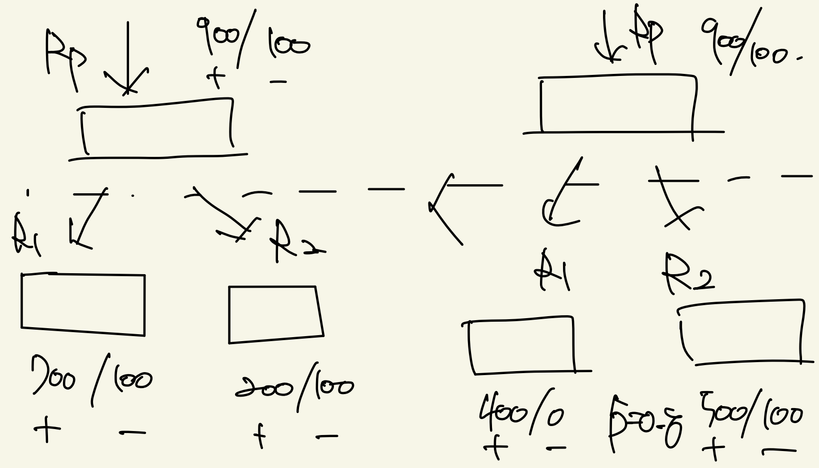
손실: $L_{misclass} = 1 - \max_c \hat{p}_c$

$$\max \underbrace{L(R_p)}_{\text{parent loss}} - \boxed{\underbrace{(L(R_1) + L(R_2))}_{\text{children loss}}}$$

← 이것만 고려하면 됨.

Misclassification 문제점

$\hat{p} = 0.9$



두 Model 모두 $L(R_p) = 100$으로

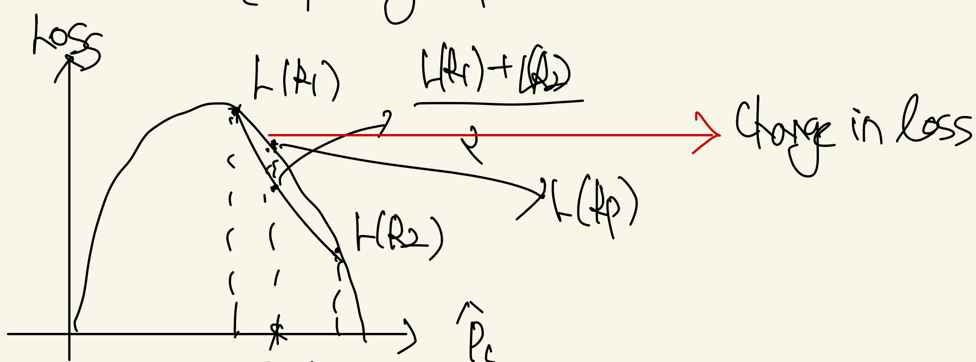
Misclassification

Non-sensitive

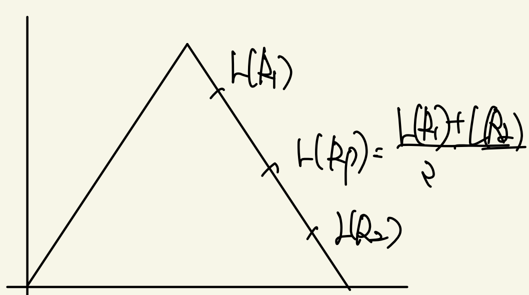
$\hat{p} = 1$

따라서, Cross-entropy loss 사용하자!

$$L_{cross} = -\sum_c \hat{p}_c \log_2 \hat{p}_c$$



[Cross-entropy] ↳ 부모는 아이들의 평균임 (midpoint)...

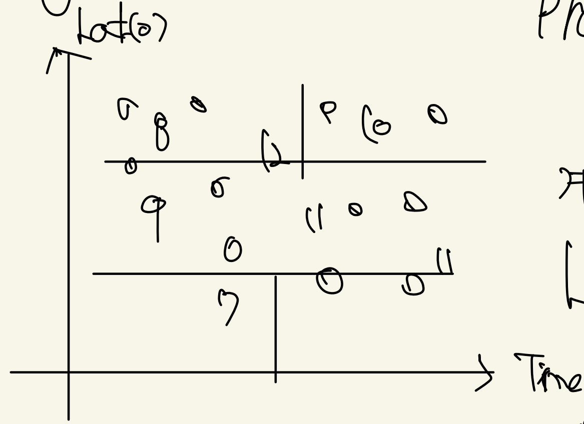


[Misclassification]

* 다른 손실함수

$$\text{Gini} \sum_c \hat{p}_c (1 - \hat{p}_c)$$

Regressions Trees



for R_m

Predict $\hat{y}_m = \dfrac{\sum_{i \in R_m} y_i}{|R_m|}$

계산

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

* 범주형 변수들을 쉽게 처리 가능

왜냐면 ..

Decision Tree는 과적합되기 쉬움, 따라서

Regularization (규제 전략)

1) min leaf size
2) max depth
3) max number of nodes
4) Min decrease in loss
   너무 일찍 stop 할 가능성 존재함.
5) Pruning (misclassification on val set)

Runtime

- n examples
- f features
- d depth

→ Test time: $O(d)$, $d < \log_2 n$

Train time: 각 지점은 part of $O(d)$ nodes

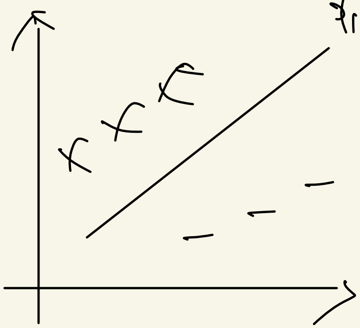
Cost of point at each node is $O(f)$

그러서 최종 cost $O(\underbrace{nf}_{\text{데이터 행렬 크기}}d)$ → 속도 빠르고 Runtime이 짧다

DT의 단점

No additive structure

$x_1 = x_2$ → 가능의 수 증가하고 성능 악화될시 성능 ↓



Recap

장점
1. fast
2. Interpretable
3. Categorical vars
4. fast

단점
1. High variance
   - overfitting
2. Bad at additive
3. low predictive acc

Ensembling

X_i가 iid 이면 $Var(X_i) = \sigma^2$, $Var(\bar{X}) = Var(\frac{1}{n}\sum X_i) = \frac{\sigma^2}{n}$

∵ 여러개의 독립적인 요소들의 고려하면 분산이 천천히 줄일수 있지 않을까..

독립이라는 가정 drop. X_i는 id, X_i's correlated by ρ 이면

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

↳ 가중합 ↓

Ways to ensemble
1) different algorithms.
2) different training sets
3) Bagging (Random Forests)
4) Boosting (Adaboost, xgboost)

Bagging - Bootstrap Aggregation

실제 모집단 P, Training set $S \sim P$ (Sampling). 가정 $P=S$

Bootstrap samples $Z \sim S$

∵ 여러 Sample 가지고 와서 별도의 모델을 훈련한 다음 평균을 낸다.

$Z_1, Z_2 \cdots Z_m$, Train Model G_m on Z_m $\quad G'(x) = \frac{\sum_{m=1}^{M} G_m(x)}{M}$

Bias-Variance 관점에서 보면, $Var(\bar{x}) = \rho\sigma^2 + \frac{1-\rho}{M}\sigma^2$

장점) driving down ρ / More $M \rightarrow$ less variance (과적합 상관 X)

단점) 편향을 증가시킨다 ($P=S$ 가정 때문에)

→ DTs + Bagging

↳ variance high
bias low) → 배깅에 적합한 이유

Random Forest ↳ (1) Decreasing P

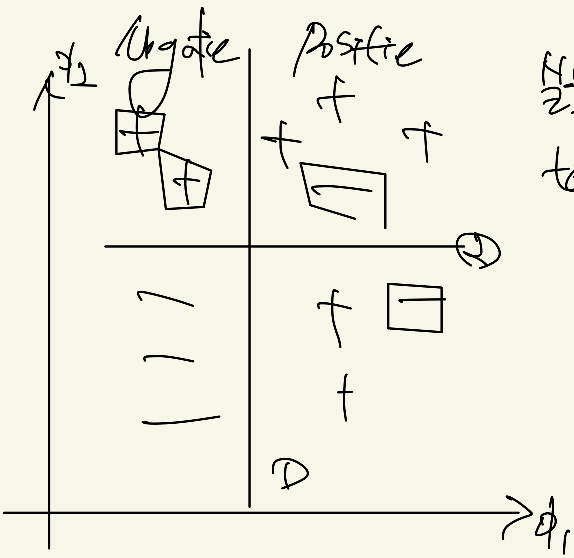
(2) Decorrelate Models

* consider only a fraction of your total features

Boosting

decrease bias, Additive

→ 하나의 Model 훈련하고 다음 예측 추가 ..



분류에 대한 가중치를 높이고 다음 test에 반영 ↳ α_m

ex) Adaboost $\log\left(\frac{1-err_m}{err_m}\right)$

$$G(x) = \sum_m \alpha_m G_m$$