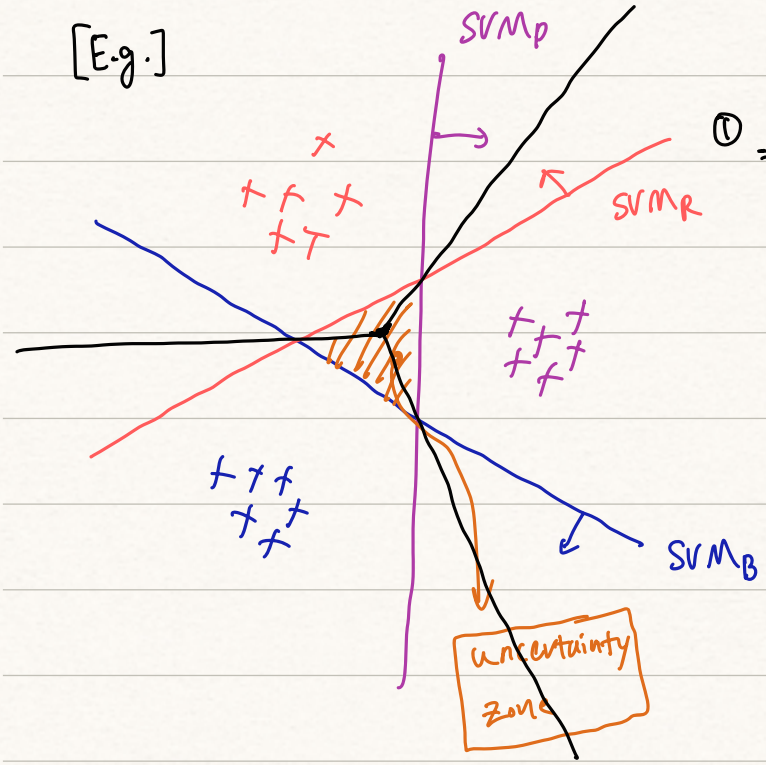
DSA5105 lec4

Recap:

SVM ⟶ Multi-class classification

[E.g.]



—: decision hyperplane

① 1 vs Rest

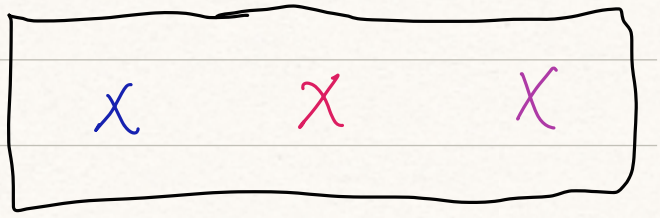
→ { SVM_R : Red vs Not Red
SVM_P : Pur vs Not Pur
SVM_B : Blue vs Not Blue

IDEA

{ $w_B^T x + b_B$
$w_P^T x + b_P$
$w_R^T x + b_R$
compare this 3 values

↓

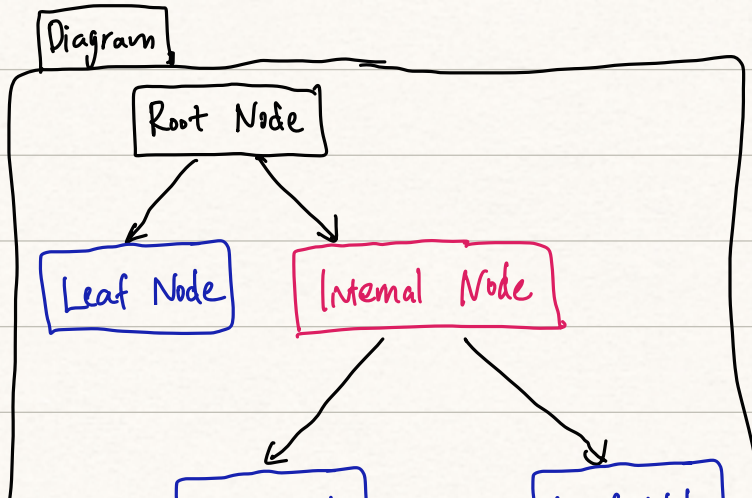
Cannot work in such kind of dataset



Hypothesis space $H = \{ f :$ some kind of function $f \}$

DTs ⟶ subsequent questions

Diagram



Leaf Node Leaf Node

Key Idea:

→ Stratify the input space into non-overlap regions!

[E.g.] → 1-dim example

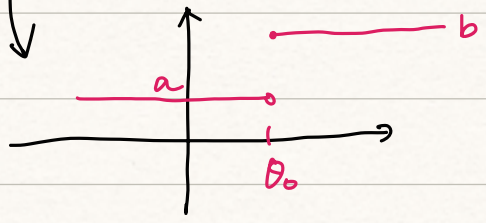
$f^*: [0,1] \to \mathbb{R}$ ⇝ Oracle

A depth-1 DT is a piece-wise function:

Hypothesis class

$H = \{ f_{a,b,\theta_0}(x) : f_{a,b,\theta_0}(x) = \begin{cases} a, & x < \theta_0 \\ b, & x \geq \theta_0 \end{cases} \}$

$f(x) = \begin{cases} a, & x < \theta_0 \\ b, & x \geq \theta_0 \end{cases}$



trainable parameter (a, b, θ_0)

suppose Oracle $f^*(x) = x$ ⇝ identical mapping suppose $x \sim U[0,1]$

then $R_{pop}[f_{a,b,\theta_0}(x)] = \mathbb{E}_{(x,y)}[\text{Loss}(f_{a,b,\theta_0}(x), y)]$

$$= \frac{1}{2}\int_0^{\theta_0} (a-x)^2 dx + \frac{1}{2}\int_{\theta_0}^1 (x-b)^2 dx$$

consider minimize R_{pop} to achieve $(\hat{a}, \hat{b}, \hat{\theta}_0) = (\frac{1}{2}, \frac{1}{4}, \frac{3}{4})$

① → $\frac{\partial R_{pop}}{\partial a} = \frac{(a-\theta_0)^2}{2} - \frac{a^2}{2}$

$$= \frac{\theta_0(\theta_0 - 2a)}{2}$$

② → $\frac{\partial R_{pop}}{\partial b} = \frac{(1-b)^2}{2} - \frac{(\theta_0 - b)^2}{2}$

$$= \frac{(1-\theta_0)(\theta_0 + 1 - 2b)}{2}$$

③ → first-order optimality condition $\Rightarrow \begin{cases} \frac{\partial R_{pop}}{\partial a} = 0 \\ \frac{\partial R_{pop}}{\partial b} = 0 \end{cases} \Rightarrow \begin{cases} \hat{a} = \frac{1}{2}\hat{\theta}_0 \\ \hat{b} = \frac{1}{2}(1+\hat{\theta}_0) \end{cases}$

④ → find $\hat{\theta}_0$, $\frac{\partial R_{pop}}{\partial \theta_0} = \frac{1}{2}(a-\theta_0)^2 - \frac{1}{2}(\theta_0 - b)^2$

first-order optimality condition $\Rightarrow \frac{\partial R_{pop}}{\partial \theta_0} = 0$

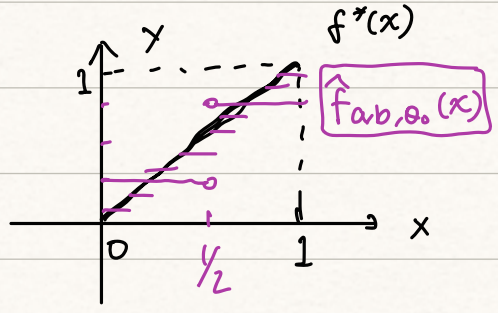
$\Leftrightarrow (\hat{a}-\hat{\theta}_0)^2 = (\hat{b}-\hat{\theta}_0)^2$

$\Leftrightarrow \frac{1}{4}\hat{\theta}_0^2 = (\frac{1}{2} - \frac{1}{2}\hat{\theta}_0)^2$

$\Leftrightarrow \boxed{\hat{\theta}_0 = 1/2}$

$\Rightarrow \begin{cases} \hat{a} = \frac{1}{4} \\ \hat{b} = \frac{3}{4} \end{cases}$

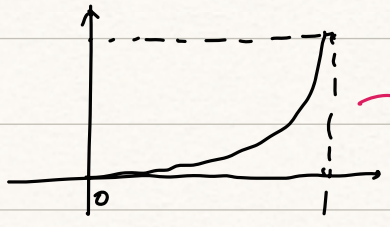
That is



$\hat{f}_{a,b,\theta_0}(x)$

use more trees to make the approximation more accurate

Another example:



→ $\hat{\theta}_0$ will be larger than 1/2

For DTs, if we have K split,

then we will have at most 2^{K-1} nodes!

↓

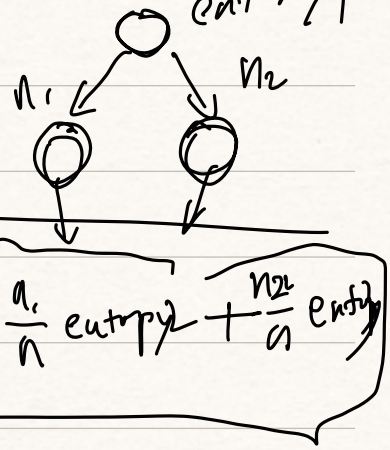
maximal number in the last level

---

n

DT → universal approximator like Neural Network

(universal approximation theorem)

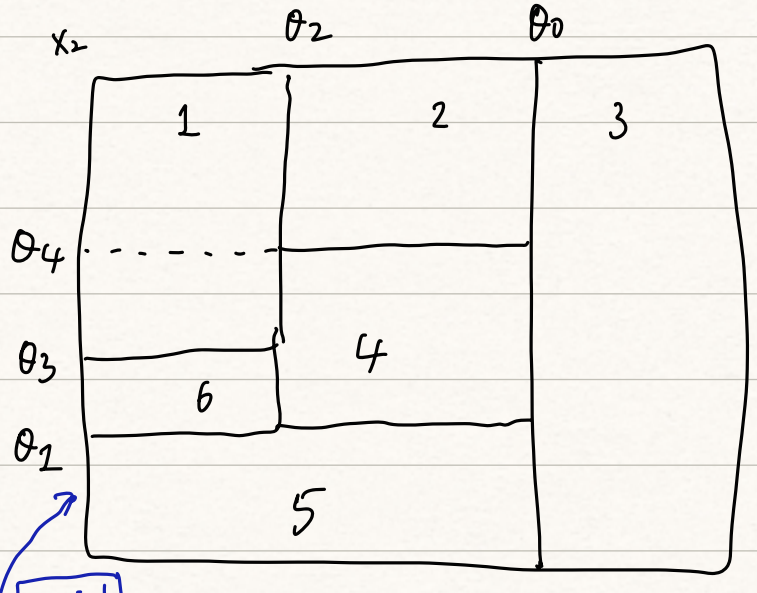


Formalize

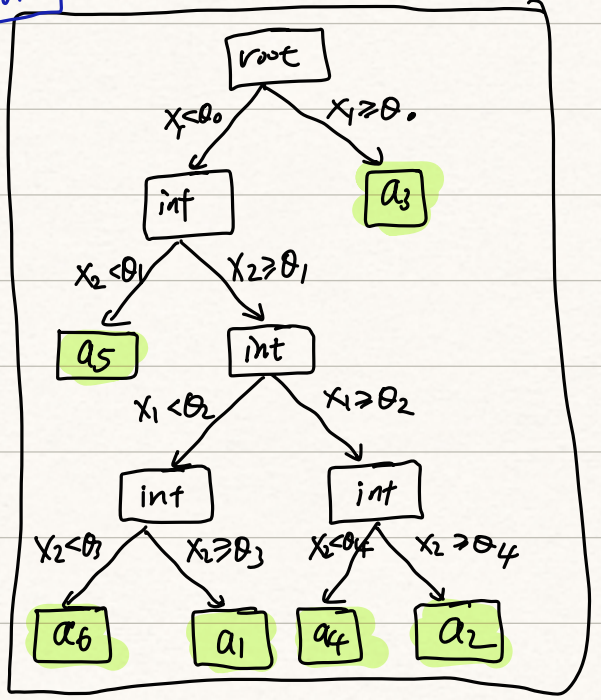
☆ a general decision tree, Hypothesis space is:

$$H = \{ f : f(x) = \sum_{j=1}^{J} a_j \mathbb{1}_{\{x \in R_j\}} ; \{R_j\}_{j=1}^{J} \text{ is a partition of } X, a_j \in Y \}$$

Target (Oracle) $f^* : x \in X \longmapsto y \in Y$



trivial



DT Diagram ↔ equivalent!

(from partition graph)

key trick: 找一条到底的线

Question: How to learn DT? (optimal parameter a_j & R_j)

Ans

optimal a_j → **Easy Part**

→ For **Regression Problem**, $\hat{a}_j$ = average target over region R_j

achieved by minimizing square loss

$$= \frac{\sum_i y_i \mathbb{1}\{x_i \in R_j\}}{\sum_i \mathbb{1}\{x_i \in R_j\}}$$

$$= \frac{\sum_{x_i \in R_j} y_i}{|R_j|}$$

= average target within region R_j

→ For **classification**, majority wins

$\Leftrightarrow$ $a_j = \text{mode}\{y_i : x_i \in R_j\}$

Optimal partition $\{R_j\}_{j=1}^{J}$ ← **Difficult Part**

→ **Regression** → **optimization problem** → **minimize empirical risk**

$$\frac{1}{2}\sum_{j=1}^{J}\sum_{x_i \in R_j}(y_i - a_j)^2$$

$$\min_{\{R_j\}\ \text{forms partition}} \frac{1}{2}\sum_{i=1}^{N}(f_{OT}(x_i) - y_i)^2$$

$$= \min_{\{R_j\}} \frac{1}{2}\sum_{i=1}^{N}\left(\sum_{j=1}^{J} a_j \mathbb{1}\{x_i \in R_j\} - y_i\right)^2$$

find optimal sol^n

Computationally infeasible since **partition number** (all possible partitions)

(Bell number $B(n)$)

$B(n) >> e^n$

NP-Hard

find good-enough solution

trade-off

greedy solution ⇒ **Recursive Binary Splitting**

always is not optimal

① pick one dimension (randomly or ...)

② find optimal split θ to split the input

w.r.t some optimization problem

dimension → $\begin{cases} x \geq \theta \\ x < \theta \end{cases}$

Binary split with θ

???

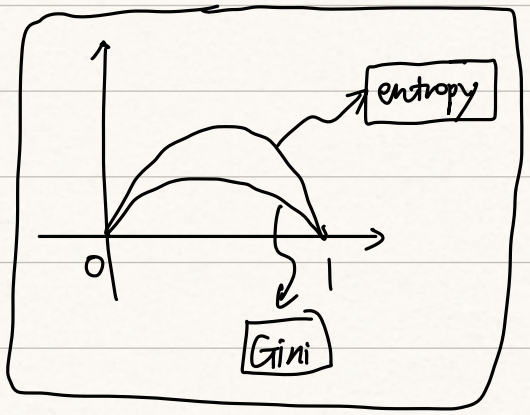
$-\sum_{j=1}^{J}\sum_{k=1}^{K} P(k,j) \log P(k|j)$ reasonable?

Classification → Modify the loss function

① Entropy: $-\sum_{j=1}^{J}\sum_{k=1}^{K} P(k|j) \cdot \log P(k|j)$

② Gini Impurity: $\sum_{j=1}^{J}\sum_{k=1}^{K} P(k|j)\,(1-P(k|j)) = 1 - \sum_{k=1}^{K} P(k|j)^2$

P(In region j, select 2 different classes)



a.) $P(k|j)$ → proportion that Region j data belongs to class k

b) $\sum_{k} P(k|j) = 1$

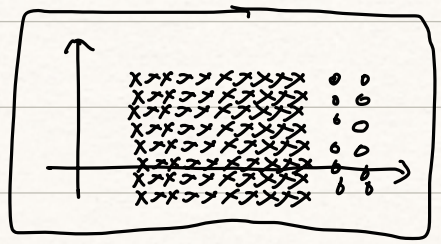
c) achieves minima when $P(k_j|j) = 1$ for $\forall j \in J$

→ the most pure case!

Disadvantage

→ greedy algo ⇒ sub-optima

→ sensitive to data variation & imbalanced data



→ over-fitting → Model Ensemble (Bagging)

Model Ensemble

- Bagging
- Boosting

(Bootstrap Aggregating)

Bootstrap → { Data (sample), features }

Aggregating →
- Reg: $\bar{f}(x) = \frac{1}{m}\sum_{j=1}^{m} f_m(x)$
- Classification: $\bar{f}(x) = \text{Mode}\{f_j(x) : j=1,\dots,m\}$

Bagging

$f_i(x) = f^*(x) + \underbrace{\varepsilon_i(x)}_{\text{noise}}$

Assume
- $\mathbb{E}[\varepsilon_i(x)] = 0$ → unrealistic
- $\text{Var}[\varepsilon_i(x)] = \sigma(x)^2$
- $\text{Cov}[\varepsilon_i(x), \varepsilon_j(x)] = 0$

Define Error

① $E(x) = \mathbb{E}[f_j(x) - f^*(x)]^2$

$\frac{1}{m}\sum_{j=1}^{m}[f_j(x) - f^*(x)]^2$

$= \mathbb{E}[\varepsilon_j(x)]^2$

$\frac{1}{m}\sum_{j=1}^{m}\mathbb{E}[\varepsilon_m(x)^2]$

$= \sigma(x)^2$

$\frac{1}{m}\sum_{j=1}^{m}\sigma^2(x) = \sigma^2(x)$

② $\bar{E}(x) = \mathbb{E}[\bar{f}(x) - f^*(x)]^2$

$= \mathbb{E}\left[\frac{1}{m}\sum_{i=1}^{m}(f_i(x) - f^*(x))\right]^2$

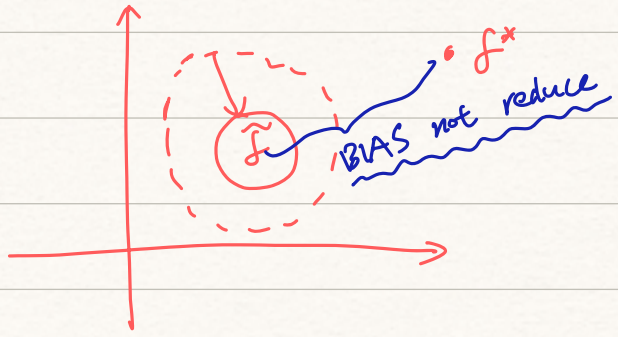
$= \frac{1}{m^2}\mathbb{E}\left[\sum_{i=1}^{m}(f_i(x) - f^*(x))^2 + \sum_{\substack{i,j \\ i\neq j}}(f_i(x) - f^*(x)) * (f_j(x) - f^*(x))\right]$

$= \frac{1}{m^2}\mathbb{E}\left[\sum_{i=1}^{m}\varepsilon_i(x)^2 + \sum_{i\neq j}\varepsilon_i * \varepsilon_j\right]$

$= \frac{1}{m}\sigma^2(x)$

$= \frac{1}{m}E(x)$ → shrink

Bagging → reduce variance



uniform weight (unweighted)

Boosting → Ada Boost → $R_{emp}(f) = \frac{1}{N}\sum_{i=1}^{N} L(f(x_i), y_i)$

reduce bias

model capacity ← train $f_1 \rightsquigarrow \alpha_1$ based on f_1 performance

update weight:
- f_1 correct → decrease weight
- f_1 wrong → increase weight

model capacity ← train $f_2 \rightsquigarrow \alpha_2$ based on f_2 performance

update weight:
- f_2 correct
- f_2 wrong

. . . . .

→ Aggregate Model $f_1 \ldots\ldots f_n$ & model capacity $\alpha_1 \ldots \alpha_n$

Cross-Validation [CV] → choose hyper-parameters

① use validation set { train / validate / test

② cannot use test data to select model ‖ solution (CV)

BAGGING IDEA

↓

cross-validation

a) split it into traing & test

b) split training set into K folds

c) train model for K times

(i-th validation, rest traing) $i=1,\dots,K$

↓

score i $i=1,2,\dots,K$ error on validation set

d) average score i $i=1,2,\dots,K$

↳ performance of one specific choice of hyper-param

⇓

score (hyper-param)

e) compare score over different choices of hyper-parameters

↓

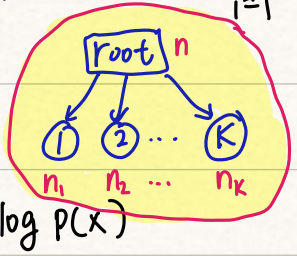
determine the optimal hyper-parameter!

My Personal Perspective

→ The Idea of GAIN-SPLIT := Entropy (root) $- \sum_{i=1}^{k} \frac{n_i}{n}$ Entropy (i)

↓

if split to K nodes



Recall: ① entropy $H(X) = -\sum P(x) \log P(x)$

② joint entropy $H(X,Y) = -\sum_{x,y} P(x,y) \log P(x,y)$ $\quad H(X,Y) \geq H(X)$

③ conditional entropy : $H(X|Y) = \sum_y P(y) H(X|Y=y)$

$H(X|Y)$ → $E_y[H(X|Y=y)] = -\sum_y P(x,y) \log P(X|y)$

A more reasonable objective

for classification DT: objective : $\min_{\text{partition}} H(X \mid \text{Partition})$

$\Leftrightarrow \min_{\{D_1 \dots D_J \text{ partition}\}} \sum_{j=1}^{J} \sum_{k=1}^{K} p(k,j) \log p(k \mid j)$

then, it is reasonable to consider gain_split for each split

$$\text{GAIN\_SPLIT} = \text{Entropy(root)} - \sum_{i=1}^{T} \frac{n_i}{n} \text{Entropy}(i)$$

Normalization

$$\text{Gain\_split\_ratio} = \frac{\text{Entropy(root)} - \text{Entropy(root|split)}}{\text{Entropy(split)}}$$