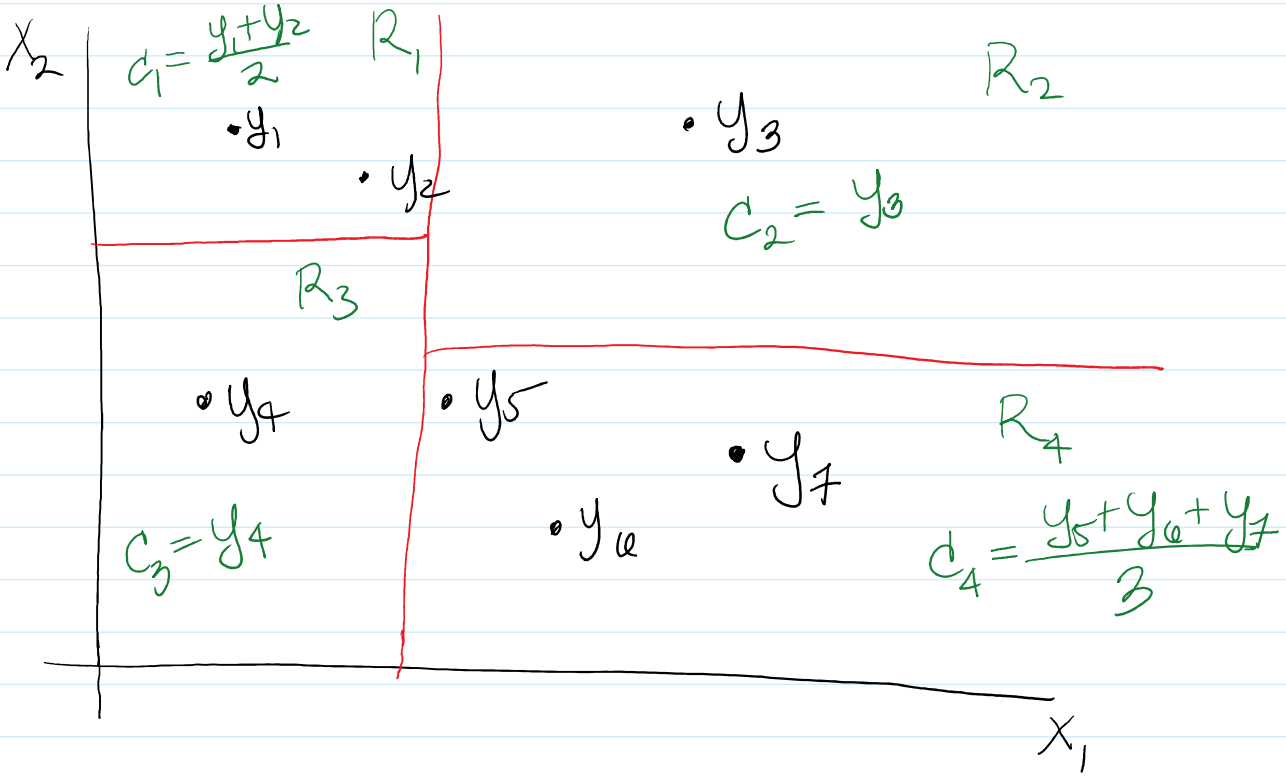
Back to Supervised Learning

Regression Trees

Basic idea

(1) break up feature space into a collection of rectangles – by successively splitting dimensions

(2) Fit a simple model to each rectangle (predict avg. value of train data in each rectangle)



$\hat{y} = \hat{f}(x) = c_i$ where $x \in R_i$

$\sim \sim M$

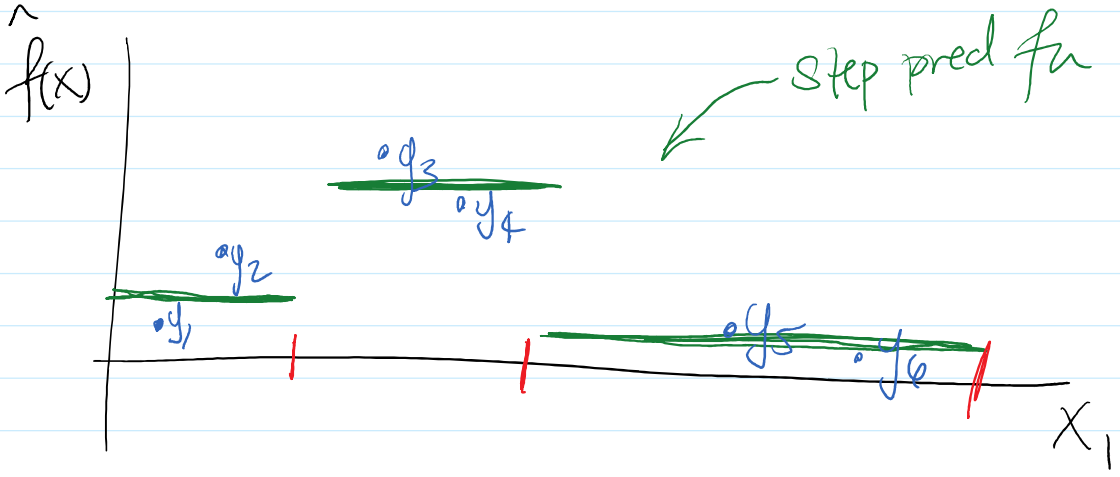
$\hat{y} = f(x) = c_i$ where $x \in R_i$

$= \sum_{i=1}^{M} c_i \, \mathbb{1}(x \in R_i)$

M: # rectangles

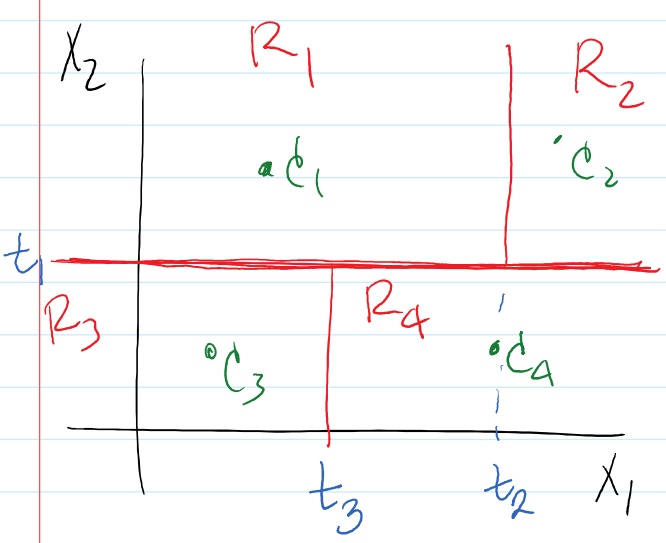
$\mathbb{1}(x \in R_i) = \begin{cases} 1 & x \in R_i \\ 0 & \text{else} \end{cases}$

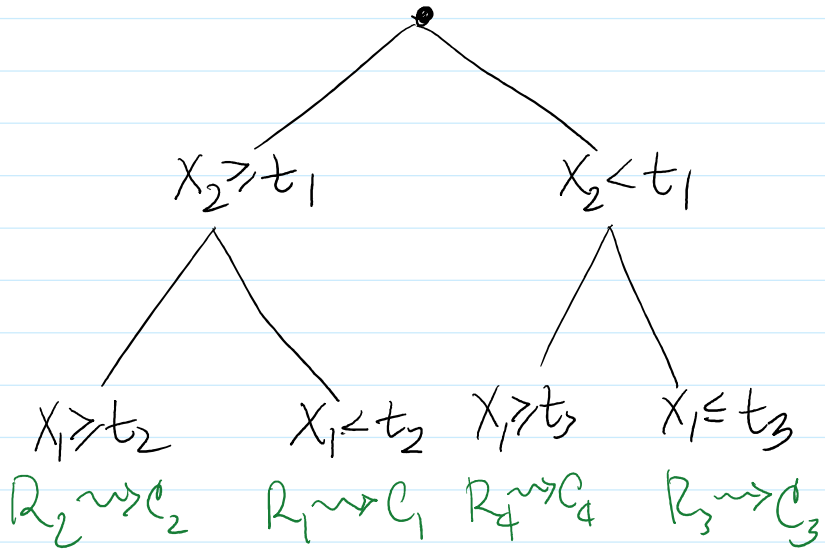
c_i: avg val. of my train ys in R_i



Why do we call this a tree?

Can represent as a binary decision tree.





Goal: Build a good tree.

Need to decide

→ (1) which variable to split?

→ (2) where do I split this variable?

(3) when do I stop (how many splits?)

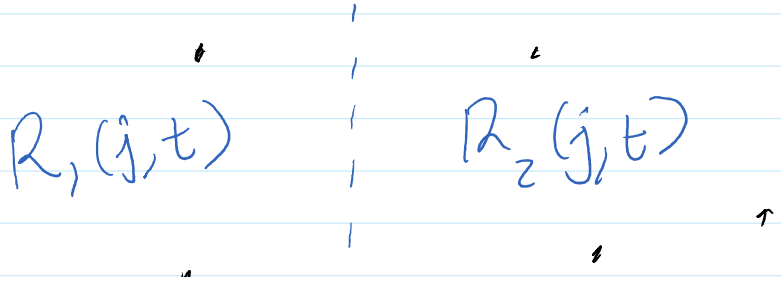
Optimal Approach:

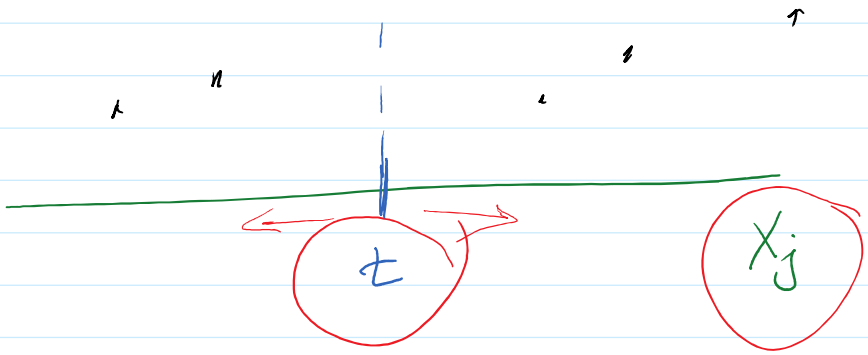
Consider all possible trees and choose the best.

Not computationally tractable.

Actually use is some (somewhat) greedy approach.

Define $R_1(j,t)$ and $R_2(j,t)$ to be the half-spaces formed by splitting on var X_j at point t.





furthermore let

$$\rightarrow \underline{RSS(j,t)} = RSS(R_1(j,t)) + RSS(R_2(j,t))$$

$$= \sum_{i: x_i \in R_1(j,t)} (y_i - c_1)^2 + \sum_{i: x_i \in R_2(j,t)} (y_i - c_2)^2$$

vals. predicted in Right and Left

turns at to minimize,

$$\begin{cases} c_1 = \text{avg of } y_s \text{ in } R_1 \\ c_2 = \text{avg of } y_s \text{ in } R_2 \end{cases}$$

Algo: Need to choose j, t.

(1) For each var. do a search over possible t and calc $RSS(j,t)$

(2) choose j, t to minimize $RSS(j,t)$.

ad make that split.

(3) Recursively do this for split data. (each side)

When do we stop?

→ too many splits (very flexible method)
Worry: overfitting

→ too few splits (inflexible method)
Worry: underfitting

Bad strategy very greedy approach.

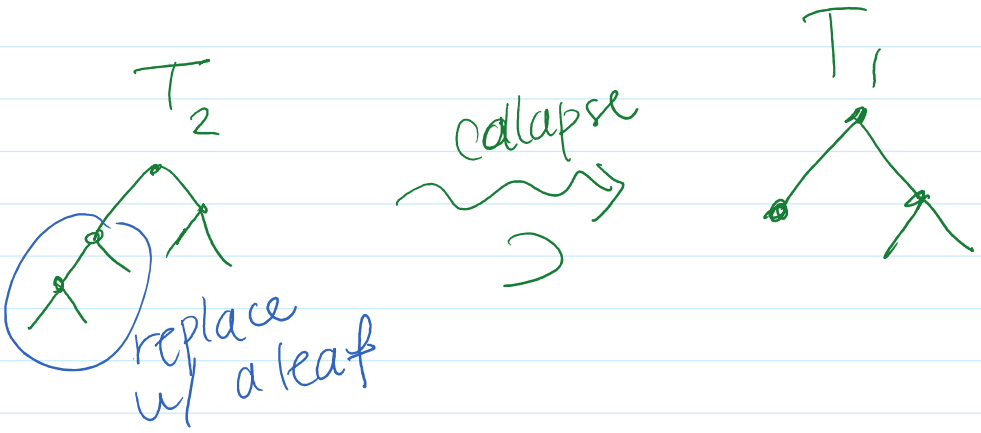
Set some threshold ad split until our RSS decreases by less than threshold.

problem: a bad split might lead to a better split down the road

Better: (1) grow a really large tree (overfit)

(2) reduce its size (pruning)

Aside: a tree t_1 is called a subtree of t_2 if I can get T_1 by "collapsing" part of T_2.



$T_1 \subset T_2$.

Do this strategy w/ cost-complexity pruning

$$C_\alpha(T) = RSS(T) + \alpha|T|$$

tree T

$|T|$: size of tree = # of leaf nodes

α: tuning parameter, $\alpha \geq 0$

Basically: given some $\alpha \geq 0$

(1) grow a large tree

(2) search over subtrees and choose subtree that minimizes $C_\alpha(T)$ (prune)

note: $\alpha = 0 \Rightarrow$ choose as large a tree as poss.

$\alpha = \infty \Rightarrow$ choose as small as possible

Fact: $\alpha_1 \leq \alpha_2$ then the tree that minimizes $C_{\alpha_1}(T)$ — called T_1 — and corresp T_2 minimizes $C_{\alpha_2}(T)$ then

$$T_2 \subset T_1.$$

So optimal trees over values of α are nested.

If I have a seq

$$\alpha_1 \leq \alpha_2 \leq \alpha_3 \leq \cdots \leq \alpha_N$$

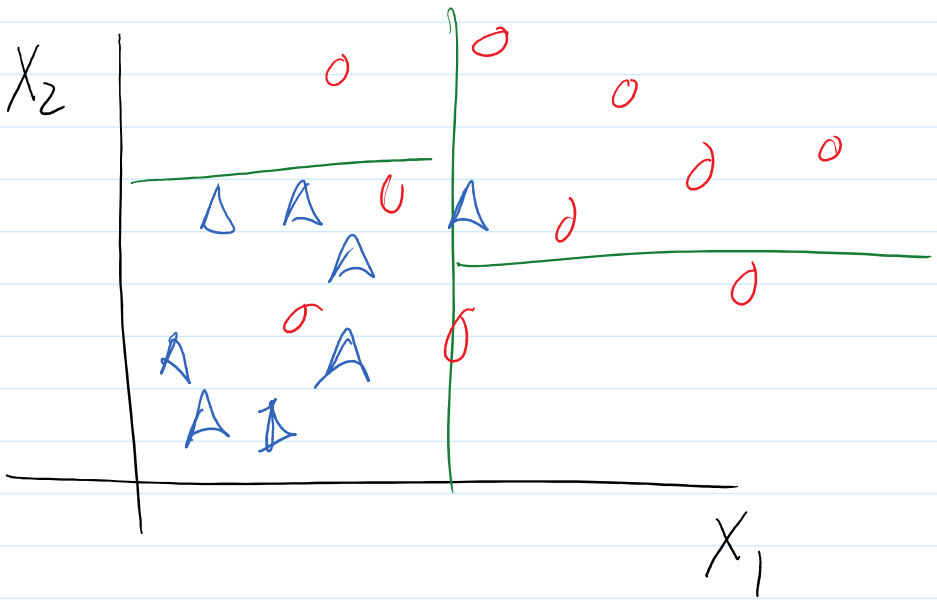
$$T_1 \supset T_2 \supset T_3 \supset \cdots \supset T_N$$

↑ nested seq of optimal trees over values of α

Can easily compute optimal CC trees over a seq of αs.

Classification Trees

Classification Tree



predict the majority class in each rectangle

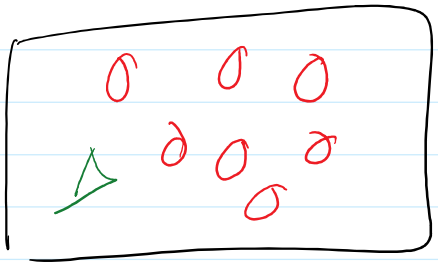
what makes a good split?

Regression tree: RSS (reduce RSS)

Classification Tree: node purity / impurity
(reduce node impurity)

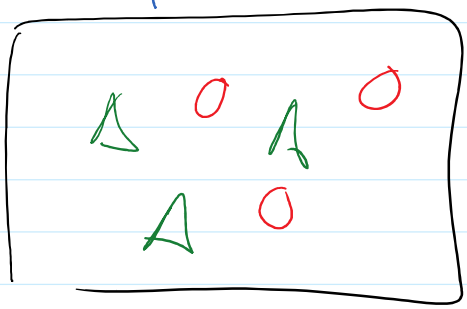
Ex.

pure node



almost all red ○

impure node



50/50 split

almost all red 0 — 50/50 split

Node Impurity Measures

pct. in node of class k

(1) mis. class. rate : $1 - \hat{p}_k$ where maj. class is class k

very pure ≈ 0

impure $\approx 1/K$

K = # classes

(2) Gini-Index

$$\sum_{k=1}^{K} \hat{p}_k(1-\hat{p}_k)$$

pure ≈ 0

impure $\approx 1/K$

(3) Entropy

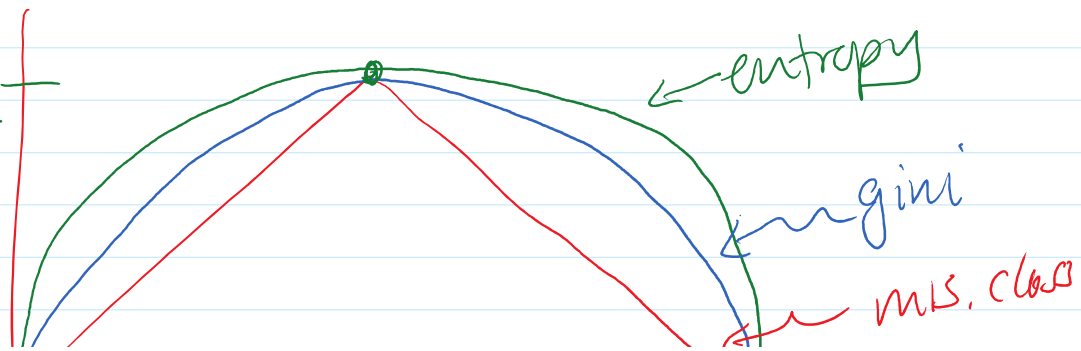
$$-\sum_{k=1}^{K} \hat{p}_k \log \hat{p}_k$$

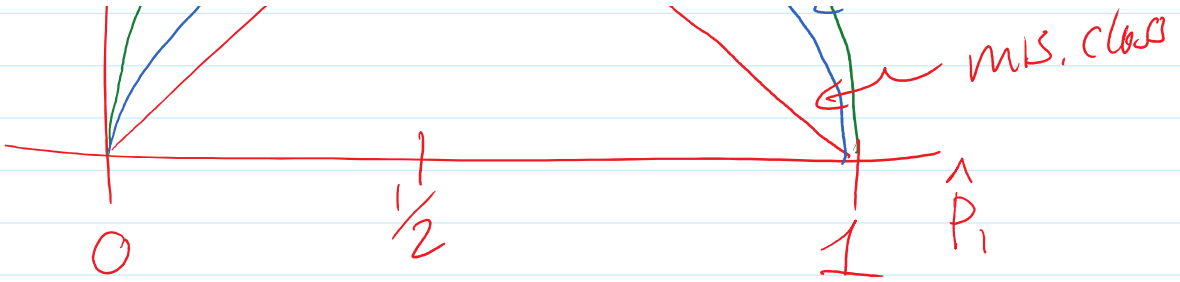
pure ≈ 0

impure $\approx 1/K$

two class

$K=2$




mls. class
0
1/2
1
p_1