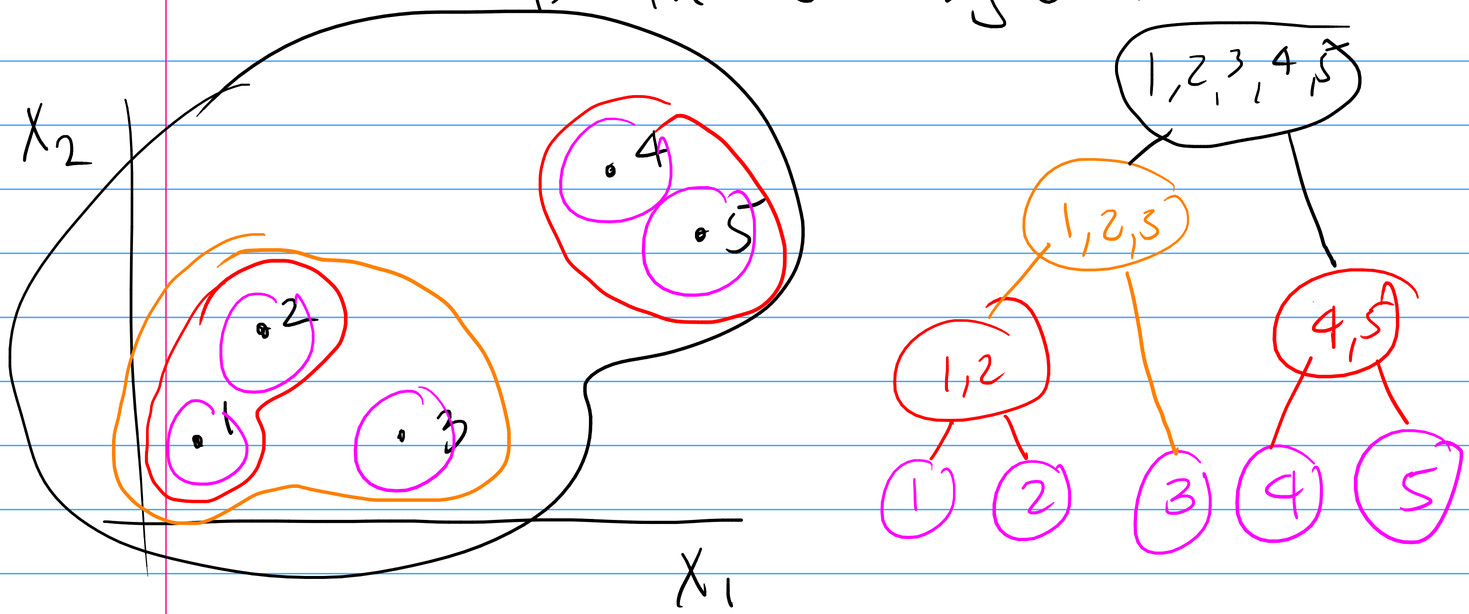
Lecture 19: Hierarchical Clustering

Build up a collection (hierarchy) of nested clusters.

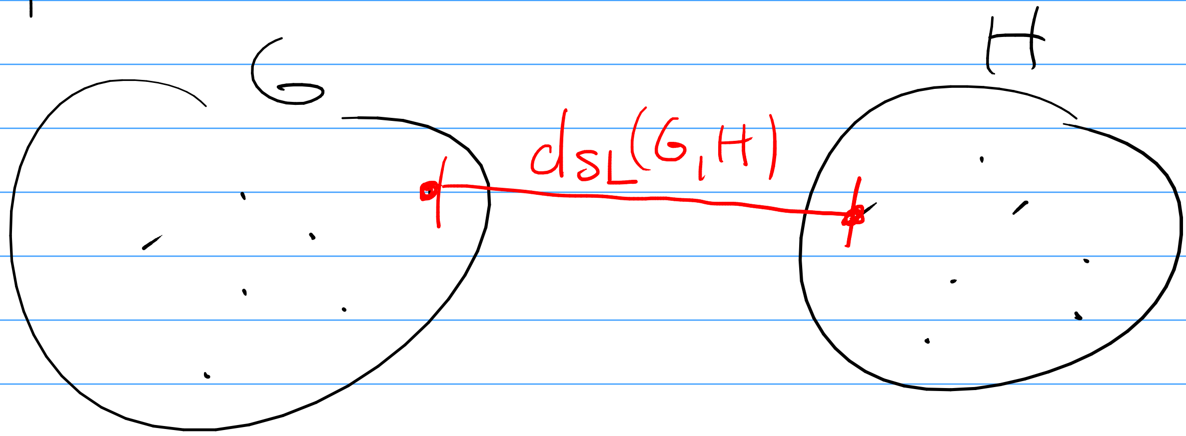
Agglomerative clustering: bottom-up procedure

1. Start w/ each pt as an individual cluster
2. merge clusters that are "close"
3. recursively do 2 until everything is in one big clusters



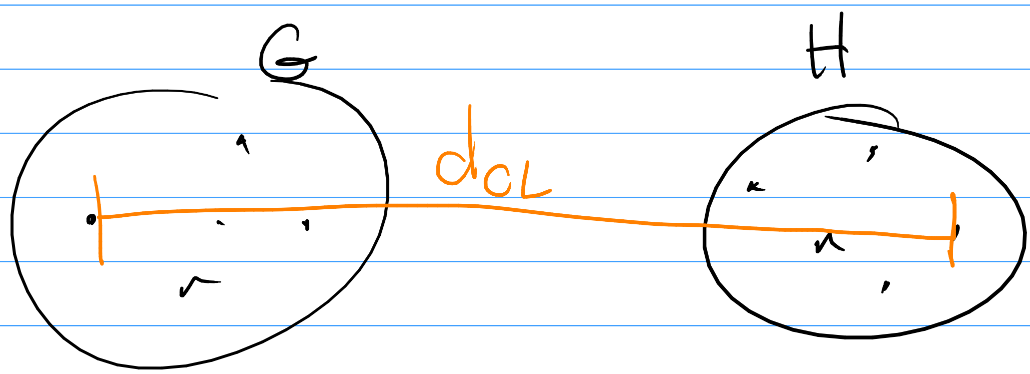
To do this, need some notion of "closeness" for clusters.

(1) Single-linkage: dist. btwn G and H is the min dist. btwn any two points in G and H



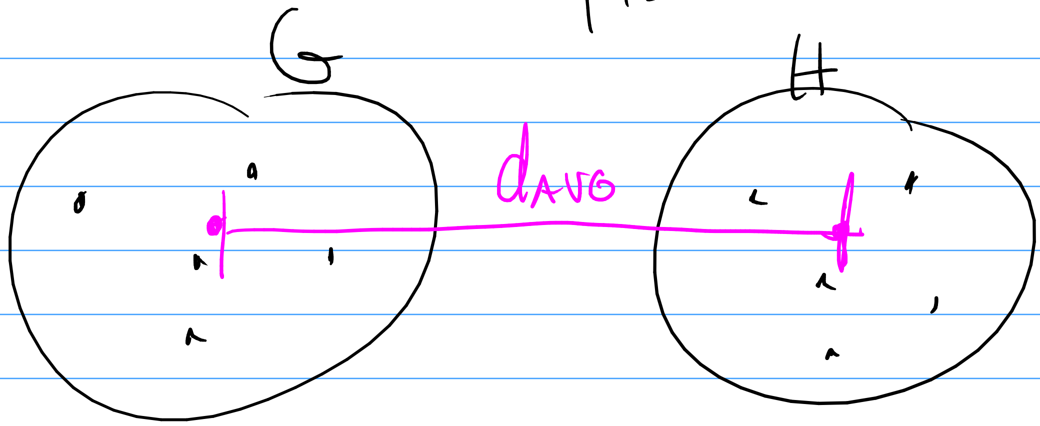
$$d_{SL}(G,H) = \min_{\substack{i \in G \\ i' \in H}} D_{ii'}$$

(2) Complete Linkage: G and H are close if max dissim/dist is small



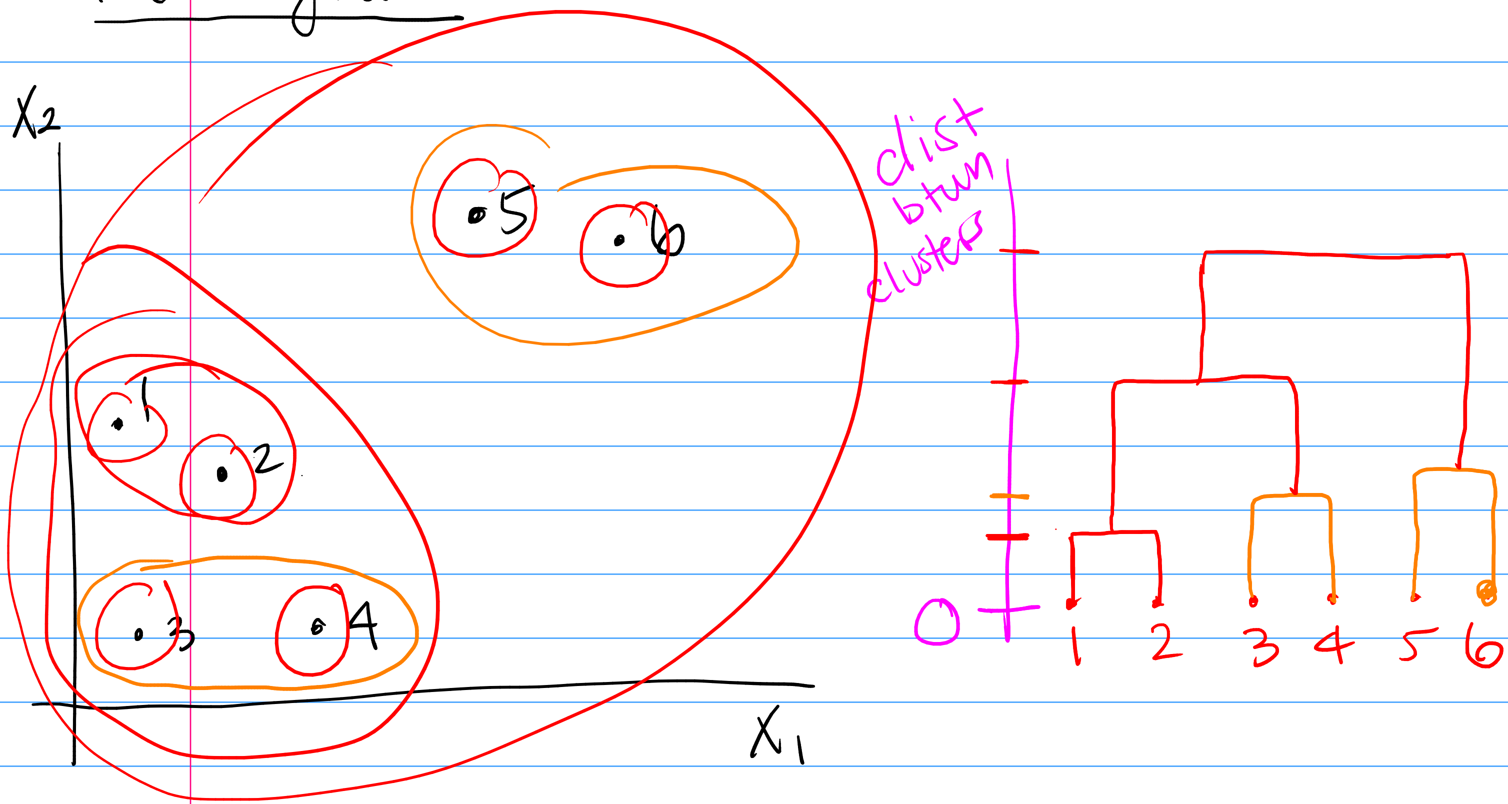
$$d_{CL}(G,H) = \max_{i \in G, i' \in H} D_{ii'}$$

(3) Average Linkage: avg. dissim btwn pts across clusters



$$d_{AVG}(G,H) = \frac{1}{N_G N_H} \sum_{i \in G} \sum_{i' \in H} D_{ii'}$$

Dendogram



Trees

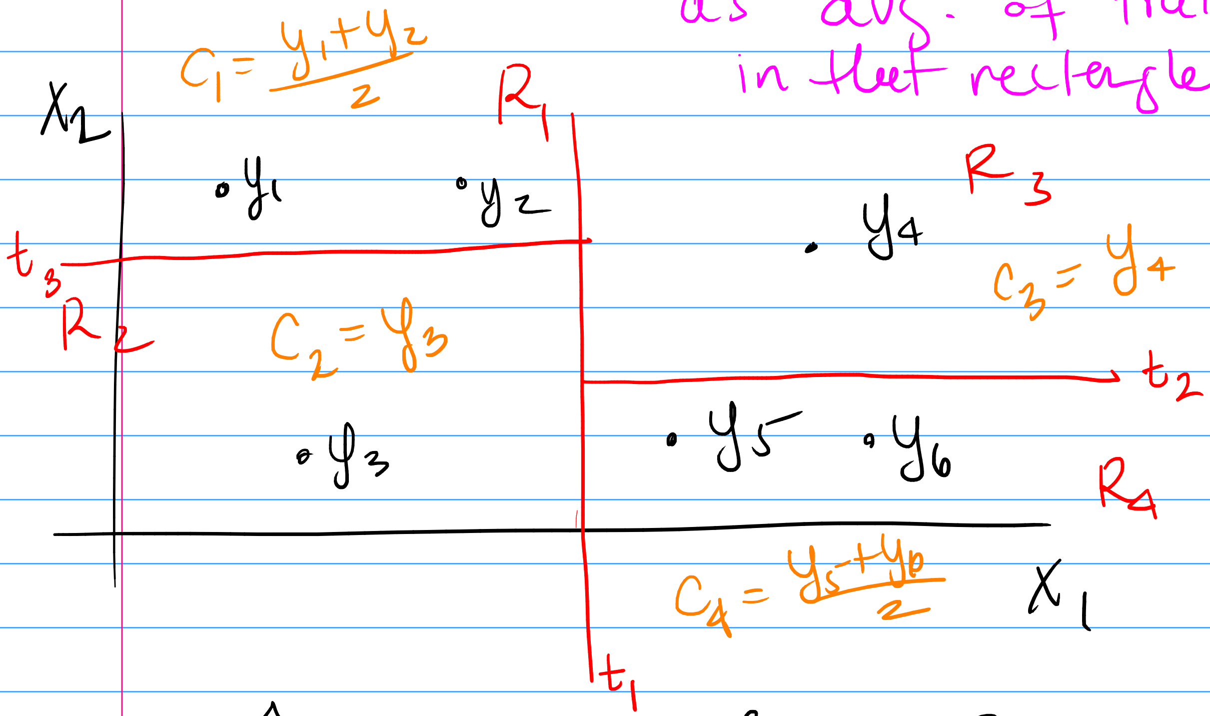
CARTs – Classification and regression trees

Regression Trees

Basic idea:

1. break up space of Xs into rectangles
2. on each rectangle – fit some really simple model

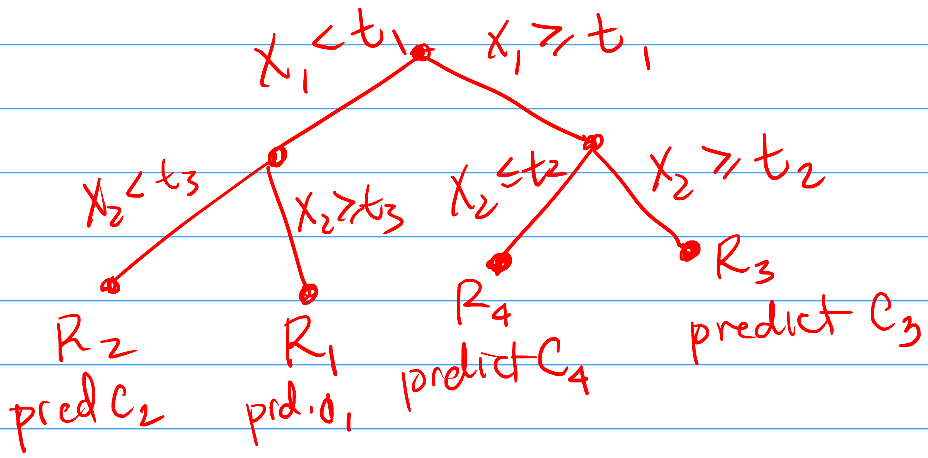
(predict y in each rectangle as avg. of training ys in that rectangle)



$$\hat{y} = \hat{f}(\underset{\sim}{x}) = c_i \quad \text{if } \underset{\sim}{x} \in R_i$$

Why called a tree?

Can represent as a decision tree



Goal: do this well

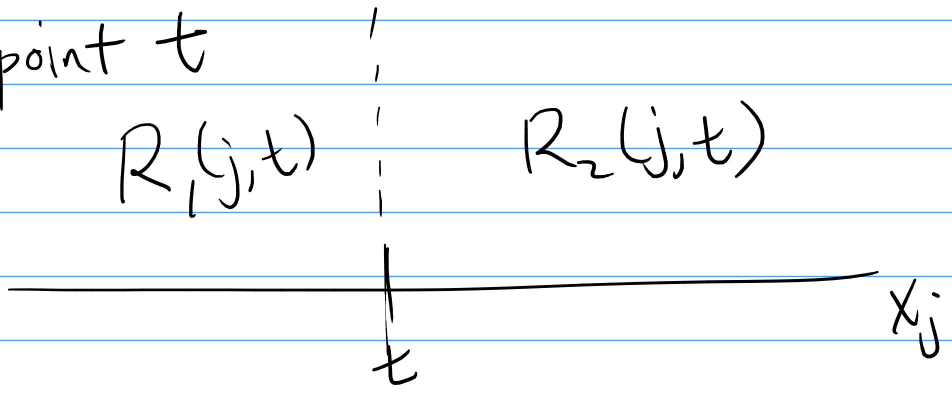
Need to decide

1. which variable to split on?
2. where I split that variable?
3. when do I stop?

Optimally, try all trees - computationally intractable

Soln: use a greedy approach

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



define

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

preds in R_1 and R_2 = mean of y_i's in R_1 and R_2

Algorithm: choose j and t

1. For each j search over possible t and calc $RSS(j,t)$
2. Choose j and t as vals that minimize this RSS
3. recursively do this for each half space

when do I stop?

→ too many splits, may overfit

→ too few, underfit

Could split until my RSS falls below some threshold.

Problem, bad split might lead to an even better split later

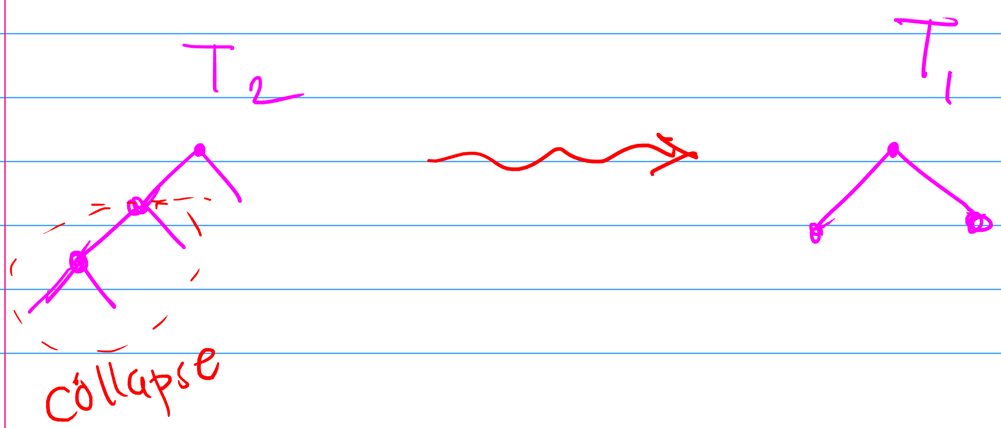
Better approach:

1. grow a really large tree (overfit)

2. reduce its size by pruning

Aside: a tree T_1 is called a sub-tree of T_2 if I can get T_1 by removing part of T_2

Ex.



Cost-complexity pruning

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

$\alpha \geq 0$

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

For some α, choose tree T that minimizes $C_\alpha(T)$

Way to do this:

1. grow a really large tree
2. search over sub-trees to find the one that minimizes $C_\alpha(T)$

If $\alpha = 0 \Rightarrow$ remove nothing, largest possible tree

$\alpha \to \infty \Rightarrow$ remove everything

Turns out that if $\alpha_1 \leq \alpha_2$ then the optimal tree T_{α_1} contains T_{α_2} as a sub-tree

As I increase α I get a sequence of nested sub-trees