Decision trees

Lecture 2b

Decision trees

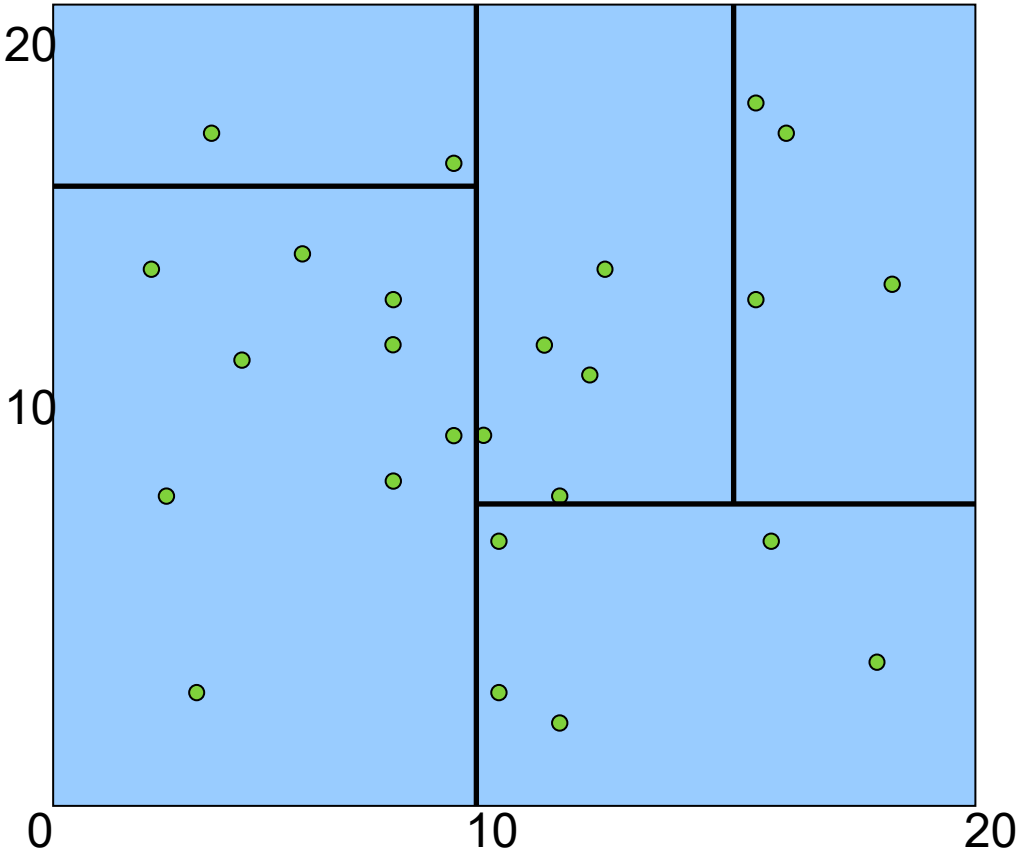
Idea

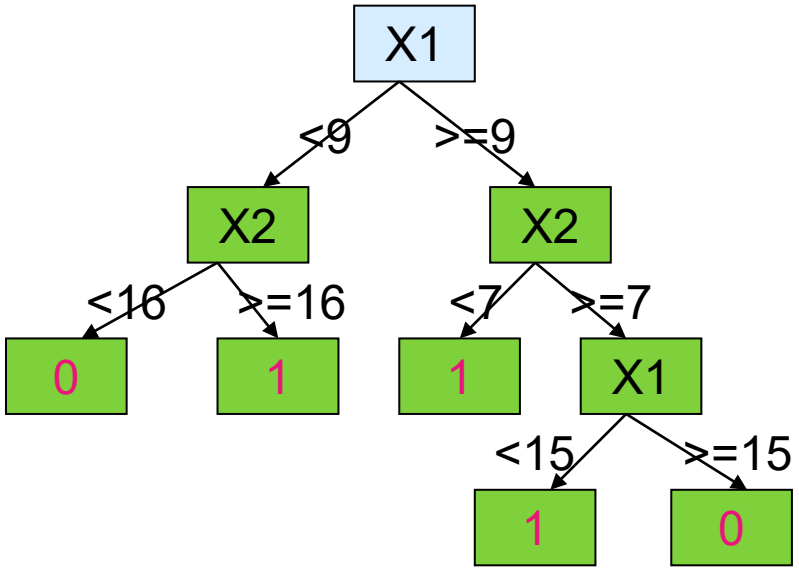
Split the domain of feature set into the set of hypercubes (rectangles, cubes) and define the target value to be constant within each hypercube

$$\widehat{y}(\mathbf{x}_\star) = \sum_{\ell=1}^{L} \widehat{y}_\ell \mathbb{I}\{\mathbf{x}_\star \in R_\ell\}$$

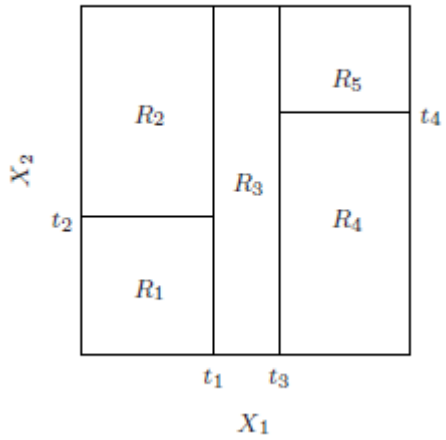
- Regression trees:
  - Target is a continuous variable
- Classification trees
  - Target is a class (qualitative) variable

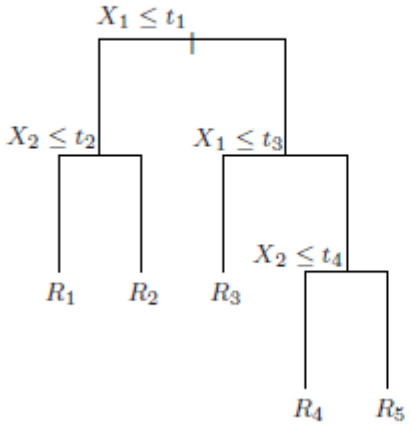
Classification tree toy example

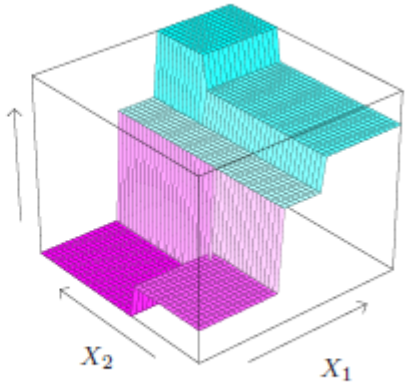




Regression tree toy example



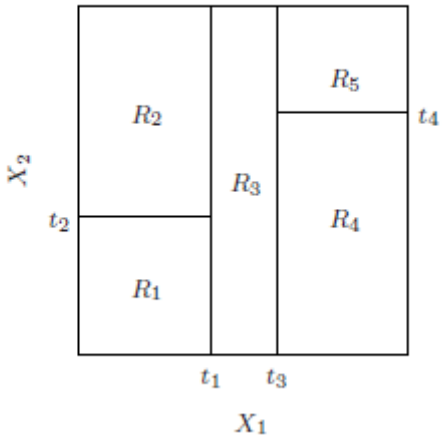




Decision trees

- A tree $T =< r_i, s_{r_i}, R_j, i = 1 \dots S, j = 1 \dots L >$
  - $x_{r_i} \leq s_{r_i}$ splitting rules (conditions), S- their amount
  - R_j-terminal nodes, L- their amount
  - labels μ_j in each terminal node

- Learning by MLE:
  - Step 1: Finding optimal tree
  - Step 2: Finding optimal labels in terminal nodes
  - Problem: NP-hard task!



Decision trees

- **Normal model** leads to regression trees
  - Objective: MSE
- **Multinomial model** leads to classification trees
  - Objective: cross-entropy (deviance)

Classification trees

- Target is categorical

- Classification probability $\pi_{\mathrm{lm}} = p(y = m | x \in R_l)$ is estimated for every class in a node
- How to estimate π_{lm} for class m and node R_l?

Class proportions

$$\hat{\pi}_{lm} = \frac{1}{n_l} \sum_{i: x_i \in R_l} I(y_i = m)$$

- For any node (or leaf), a label can be assigned

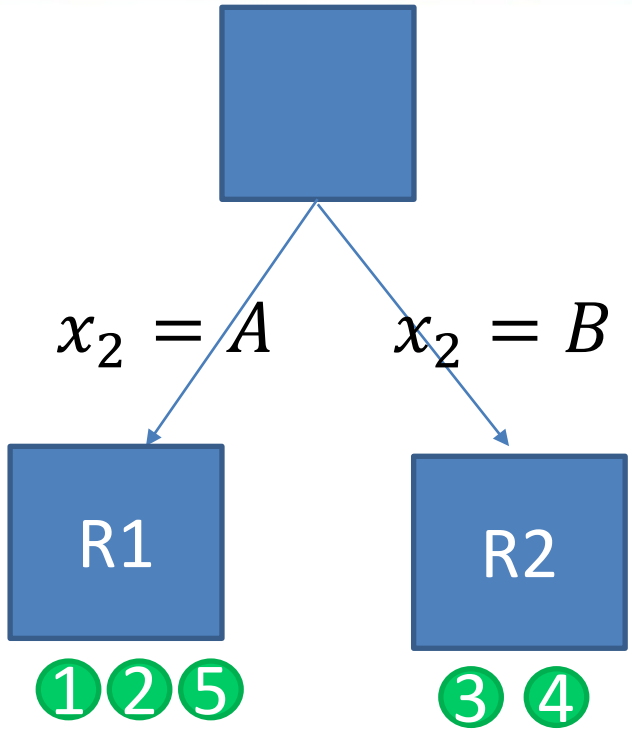
$$\hat{y}_l = \arg\max_m \pi_{lm}$$

Example

ID	x_1	x_2	y
1	2	A	C1
2	1	A	C1
3	3	B	C2
4	2	B	C2
5	3	A	C2

Example

- Splitting on x_2
  - $n_1 = 3$
  - $n_2 = 2$
  - $\hat{\pi}_{11} = \frac{2}{3}$
  - $\hat{\pi}_{12} = \frac{1}{3}$
  - $\hat{\pi}_{21} = 0$
  - $\hat{\pi}_{22} = 1$
  - $\hat{y}_1 = c1$
  - $\hat{y}_2 = c2$



Classification trees

- Impurity measure $\mathrm{Q}(R_l)$
  - R_l is a tree node (region)
  - Node can be split unless it is pure

- **Misclassification rate** $Q(R_l) = 1 - \max_m \hat{\pi}_{lm}$
- **Gini index** $Q(R_l) = \sum_{m=1}^{M} \hat{\pi}_{lm}(1 - \hat{\pi}_{lm})$
- **Cross-entropy** $Q(R_l) = -\sum_{m=1}^{M} \hat{\pi}_{lm} \ln \hat{\pi}_{lm}$

- Note: In many sources, deviance is $\mathrm{Q}(R_l)\, n_l$

Example

- Misclassification rate
  - $Q(R1) = 1/3$

- Gini index
  - $Q(R1) = 4/9$

Learning classification trees: CART

Step 1: Finding optimal tree: grow the tree in order to minimize global objective

1. Let C_0 be a hypercube containing all observations
2. Let queue C={C_0}
3. Pick up some C_i from C and find a variable x_j and value s that split C_j into two hypercubes

$$R_1 = \{x | x_j < s\} \ and \ R_2 = \{x | x_j \geq s\}$$

and minimizes

$$\min_{j,s}[n_1 Q(R_1) + n_2 Q(R_2)]$$

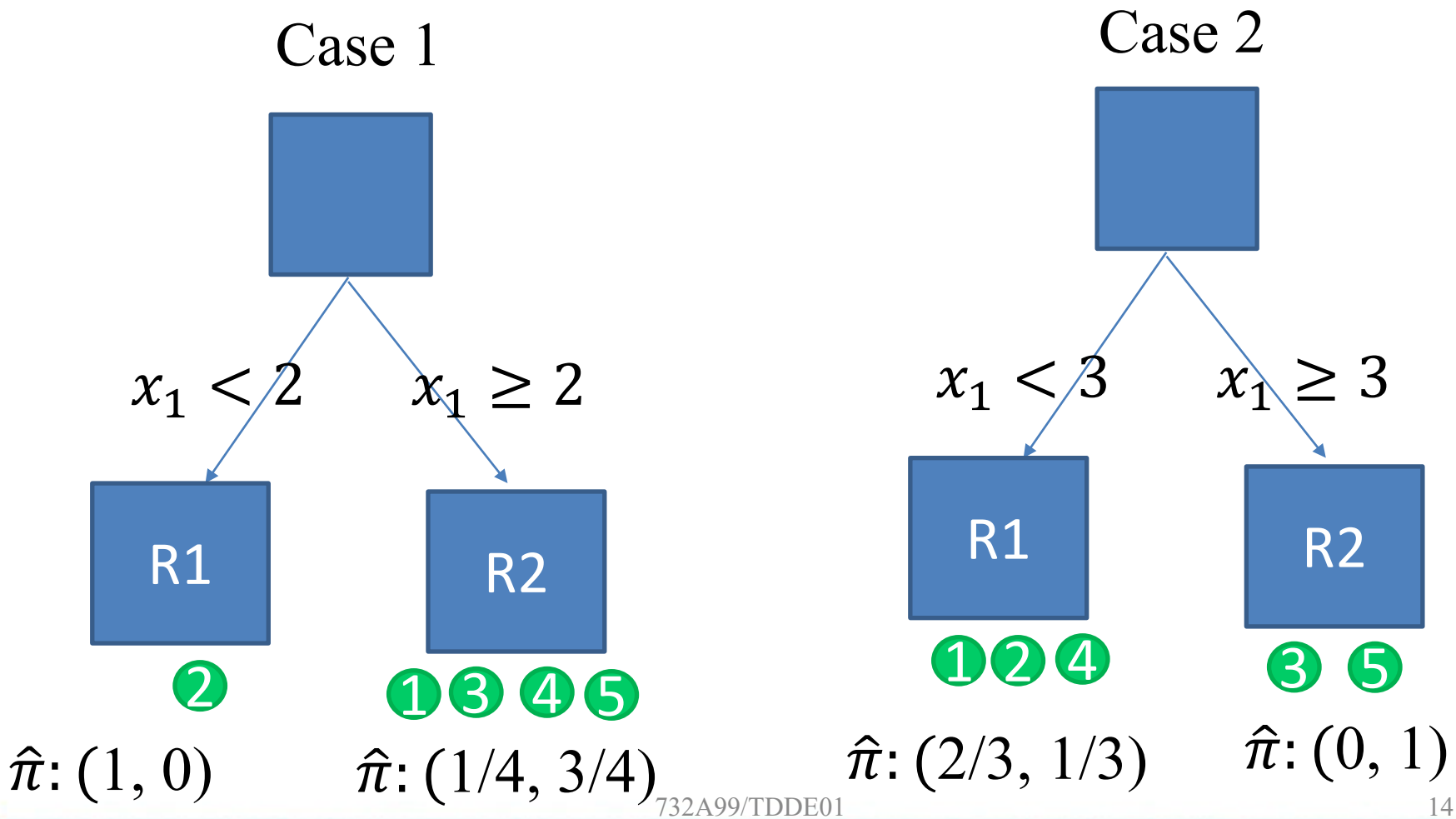
4. Remove C_j from C and add R_1 and R_2
5. Repeat 3-4 as many times as needed (until some stopping criterion is fulfilled or until each cube has only 1 observation)

Greedy algorithm (optimal tree is not found)

Example

ID	x_1	x_2	y
1	2	A	C1
2	1	A	C1
3	3	B	C2
4	2	B	C2
5	3	A	C2

Example


Case 1
$x_1 < 2$
$x_1 \geq 2$
R1
R2
2
1 3 4 5
$\hat{\pi}$: (1, 0)
$\hat{\pi}$: (1/4, 3/4)
Case 2
$x_1 < 3$
$x_1 \geq 3$
R1
R2
1 2 4
3 5
$\hat{\pi}$: (2/3, 1/3)
$\hat{\pi}$: (0, 1)

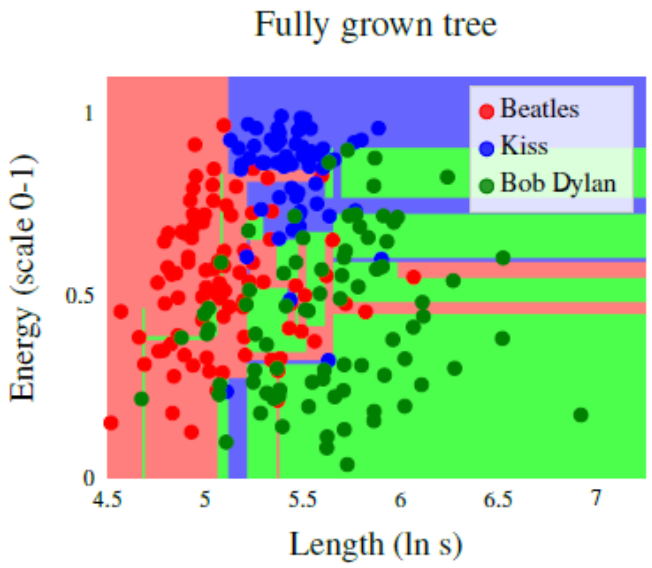
Example

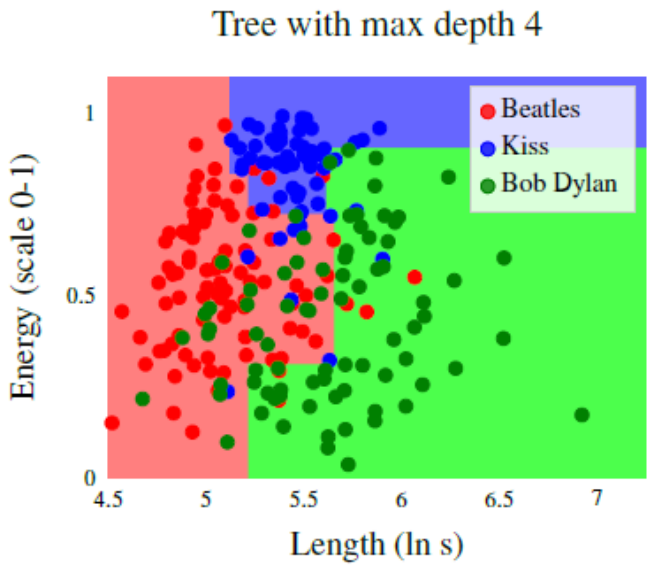
Assuming Gini index

- Case 1
  - $Q(R_1) = 0$
  - $Q(R_2) = 3/8$
  - $n_1 Q(R_1) + n_2 Q(R_2) = 1.5$
- Case 2
  - $Q(R_1) = 4/9$
  - $Q(R_2) = 0$
  - $n_1 Q(R_1) + n_2 Q(R_2) = 1.33$

CART: comments

- When to stop tree growing?





CART: comments

- The largest tree will interpolate the data → large trees = overfitting the data
- Too small trees=underfitting (important structure may not be captured)
- Optimal tree size?

Optimal trees

- **Postpruning**

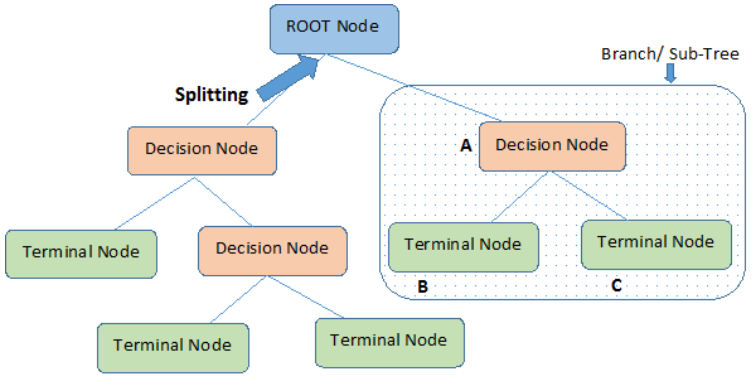
Weakest link pruning:

1. Merge two leaves that have smallest *n(parent)*Q(parent)-n(leave1)Q(leave1)-n(leave2)Q(leave2)*
2. For the current tree T, compute

$$\mathrm{I(T)} = \sum_{R_l \in leaves} n(R_l)Q(R_l) + \alpha|T|$$

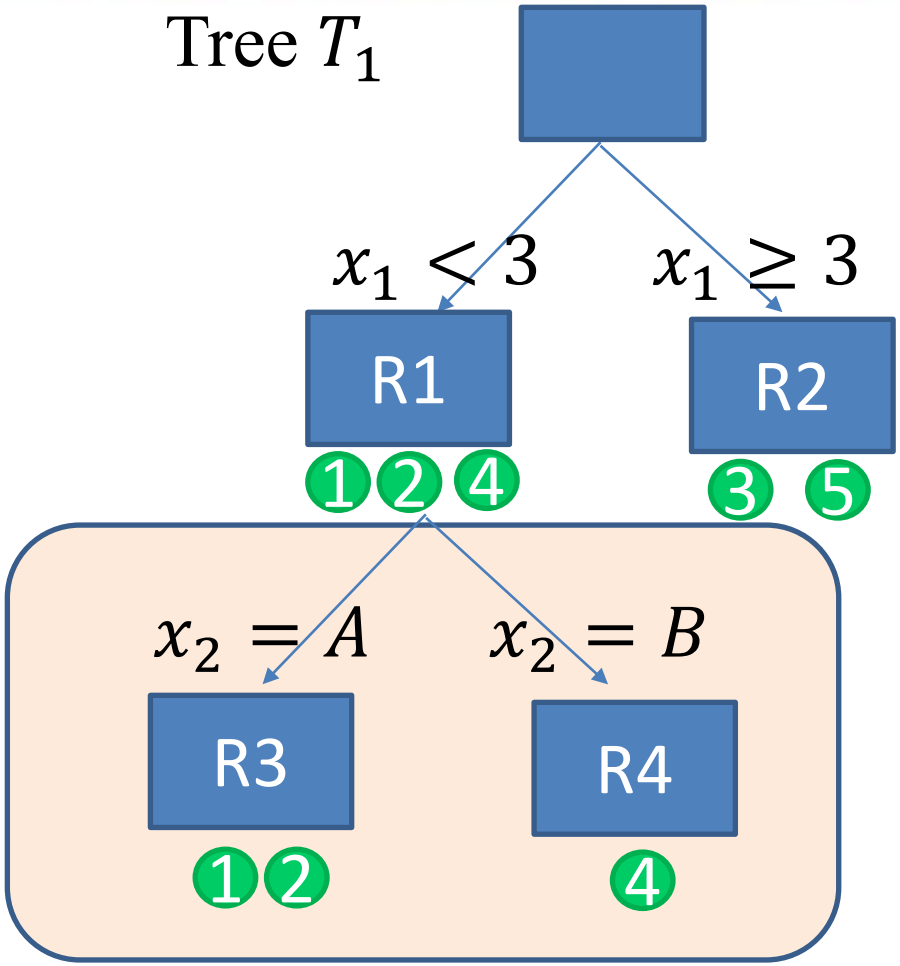
$|T|$ =#leaves

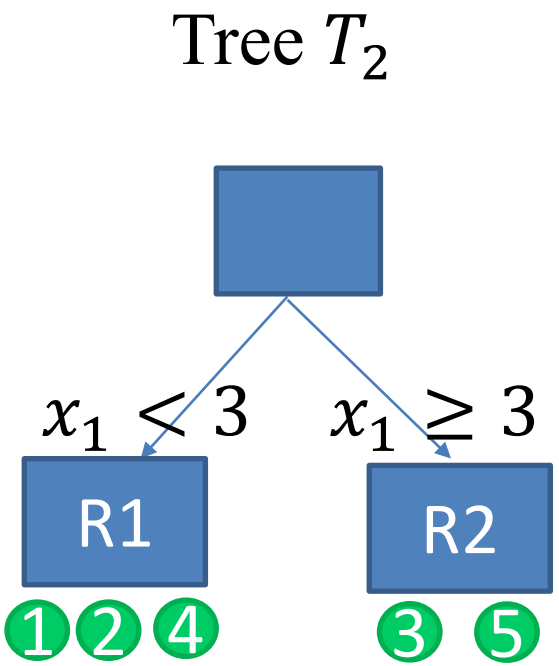
3. Repeat 1-2 until the tree with one leave is obtained
4. Select the tree with smallest I(T)



How to find the optimal α? Cross-validation.

Example


Tree T_1
$x_1 < 3$
$x_1 \geq 3$
R1
R2
1 2 4
3 5
$x_2 = A$
$x_2 = B$
R3
R4
1 2
4


Tree T_2
$x_1 < 3$
$x_1 \geq 3$
R1
R2
1 2 4
3 5

Example

- Parent: R_1, Children: R_3, R_4

- Assume
  - Gini index
  - $|T| = \#leaves$
  - $\alpha = 1$

  - $I(T_1) = 3$
  - $I(T_2) = 3.33$

Decision trees: comments

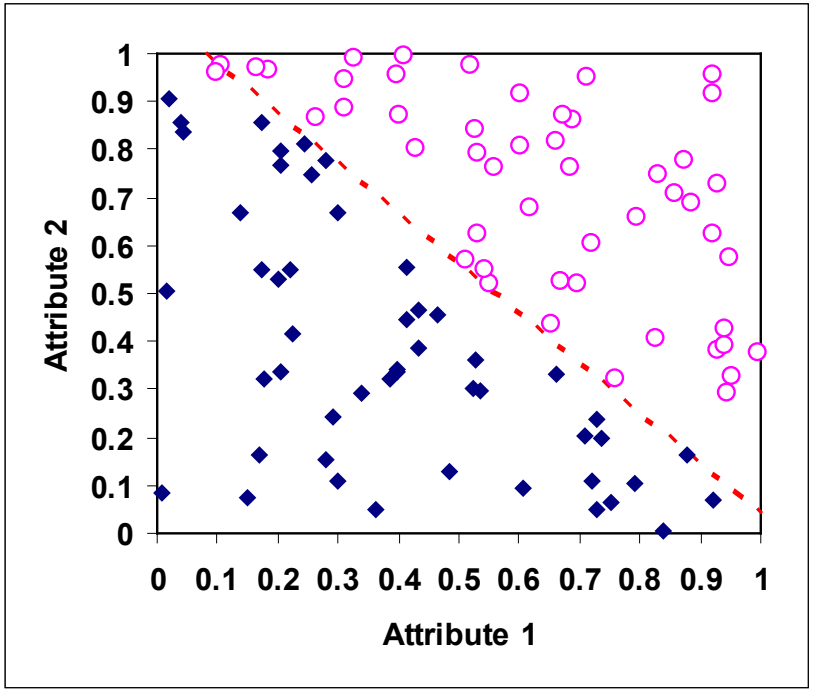
- Similar algorithms work for regression trees
  - Compute $\hat{y}_l = \frac{1}{n(R_l)} \sum_{i:x_i \in R_l} y_i$
  - replace $\mathrm{n} \cdot Q(R)$ by $SSE(R) = \sum_{i:x_i \in R_l} (y_i - \hat{y}_l)^2$

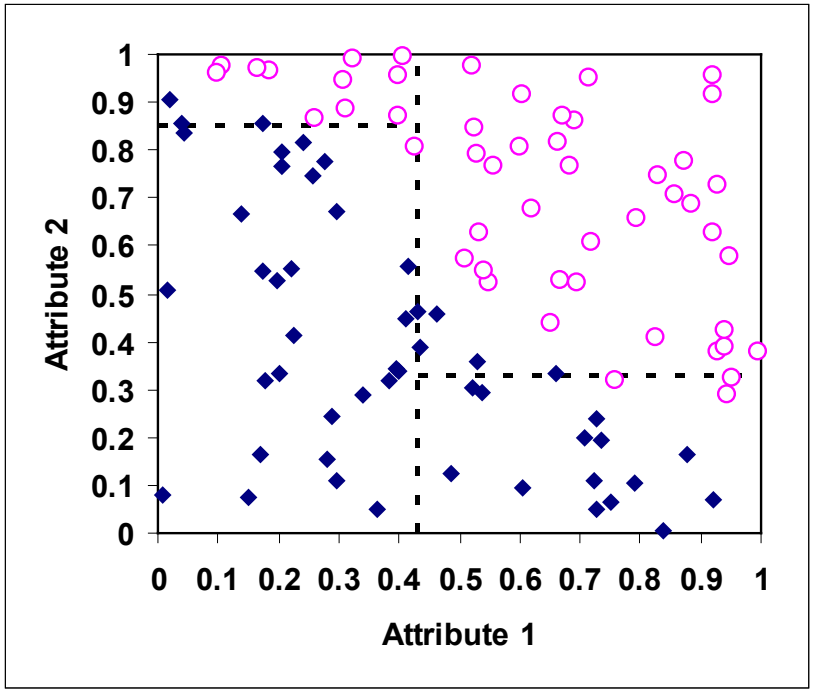
- Belongs to the class of **interpretable ML models**
- Easy to handle all types of features in one model
- **Automatic variable selection**
- Relatively robust to outliers
- Handle large datasets

- Trees have high variance: a small change in response→ totally different tree
- Greedy algorithms → fit may be not so good
- Lack of smoothness

Decision trees: issues

- Large trees may be needed to model an easy system:





Decision trees in R

- **tree** package
  - Alternative: **rpart**

tree(formula, data, weights, control, split = c("deviance", "gini"), …)

print(), summary(), plot(), text()

Example: breast cancer as a function av biological measurements

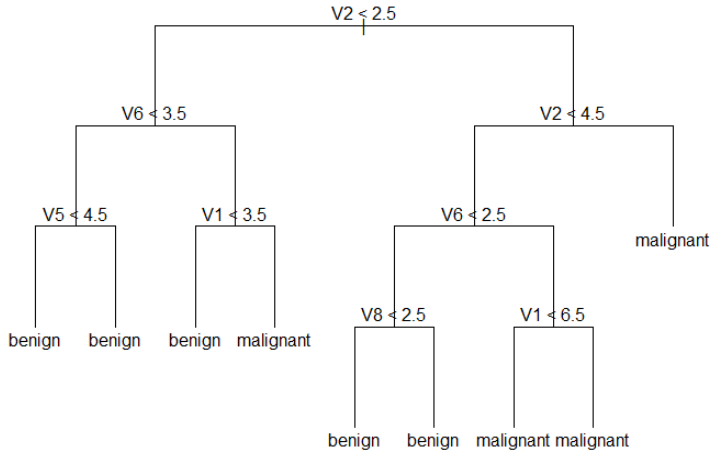
```
library(tree)
n=dim(biopsy)[1]
fit=tree(class~., data=biopsy)
plot(fit)
text(fit, pretty=0)
fit
summary(fit)
```

Decision trees in R

- Adjust the splitting in the tree with *control* parameter (leaf size for ex)

```
> fit
node), split, n, deviance, yval, (yprob)
      * denotes terminal node

 1) root 683 884.400 benign ( 0.650073 0.349927 )
   2) V2 < 2.5 418 108.900 benign ( 0.971292 0.028708 )
     4) V6 < 3.5 395  25.130 benign ( 0.994937 0.005063 )
       8) V5 < 4.5 389   0.000 benign ( 1.000000 0.000000 ) *
       9) V5 > 4.5 6   7.638 benign ( 0.666667 0.333333 ) *
     5) V6 > 3.5 23  31.490 benign ( 0.565217 0.434783 )
      10) V1 < 3.5 11   0.000 benign ( 1.000000 0.000000 ) *
      11) V1 > 3.5 12  10.810 malignant ( 0.166667 0.833333 ) *
   3) V2 > 2.5 265 217.900 malignant ( 0.143396 0.856604 )
     6) V2 < 4.5 90 120.300 malignant ( 0.388889 0.611111 )
      12) V6 < 2.5 30  27.030 benign ( 0.833333 0.166667 )
        24) V8 < 2.5 19   0.000 benign ( 1.000000 0.000000 ) *
        25) V8 > 2.5 11  15.160 benign ( 0.545455 0.454545 ) *
      13) V6 > 2.5 60  54.070 malignant ( 0.166667 0.833333 )
        26) V1 < 6.5 28  35.160 malignant ( 0.321429 0.678571 ) *
        27) V1 > 6.5 32   8.900 malignant ( 0.031250 0.968750 ) *
     7) V2 > 4.5 175  30.350 malignant ( 0.017143 0.982857 ) *
```




```
> summary(fit)

Classification tree:
tree(formula = class ~ ., data = biopsy)
Variables actually used in tree construction:
[1] "V2" "V6" "V5" "V1" "V8"
Number of terminal nodes:  9
Residual mean deviance:  0.1603 = 108 / 674
Misclassification error rate: 0.03221 = 22 / 683
```

Decision trees in R

- Misclassification results

```
Yfit=predict(fit, newdata=biopsy, type="class")
table(biopsy$class,Yfit)
```

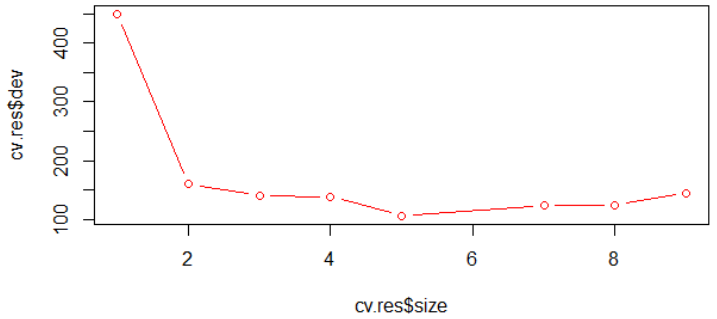
```
> table(biopsy$class,Yfit)
           Yfit
            benign malignant
  benign       440        18
  malignant      7       234
```

Decision trees in R

- Selecting optimal tree by penalizing
  - Cv.tree()

```
set.seed(12345)
ind=sample(1:n, floor(0.5*n))
train=biopsy[ind,]
valid=biopsy[-ind,]

fit=tree(class~., data=train)
set.seed(12345)
cv.res=cv.tree(fit)
plot(cv.res$size, cv.res$dev, type="b",
col="red")
```



What is optimal number of leaves?

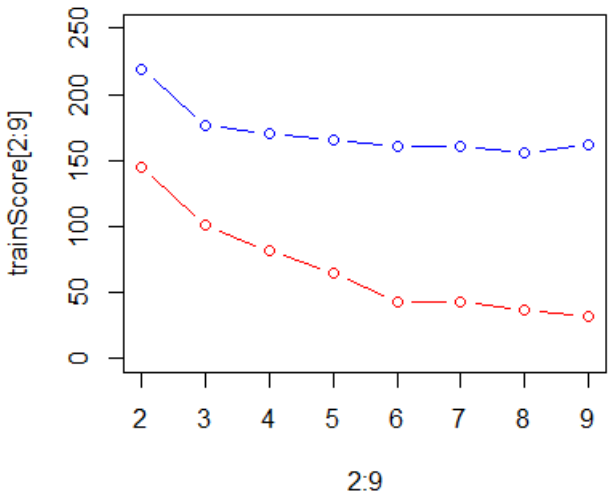
Decision trees in R

- Selecting optimal tree by train/validation

```
fit=tree(class~., data=train)

trainScore=rep(0,9)
testScore=rep(0,9)

for(i in 2:9) {
  prunedTree=prune.tree(fit,best=i)
  pred=predict(prunedTree, newdata=valid,
type="tree")
  trainScore[i]=deviance(prunedTree)
  testScore[i]=deviance(pred)
}
plot(2:9, trainScore[2:9], type="b", col="red",
ylim=c(0,250))
points(2:9, testScore[2:9], type="b", col="blue")
```



What is optimal number of leaves?

Decision trees in R

- Final tree: 5 leaves

```
finalTree=prune.tree(fit, best=5)
Yfit=predict(finalTree, newdata=valid,
type="class")
table(valid$class,Yfit)
```

```
> table(valid$class,Yfit)
           Yfit
            benign malignant
  benign       222         8
  malignant      6       114
```