CHAPTER 9:

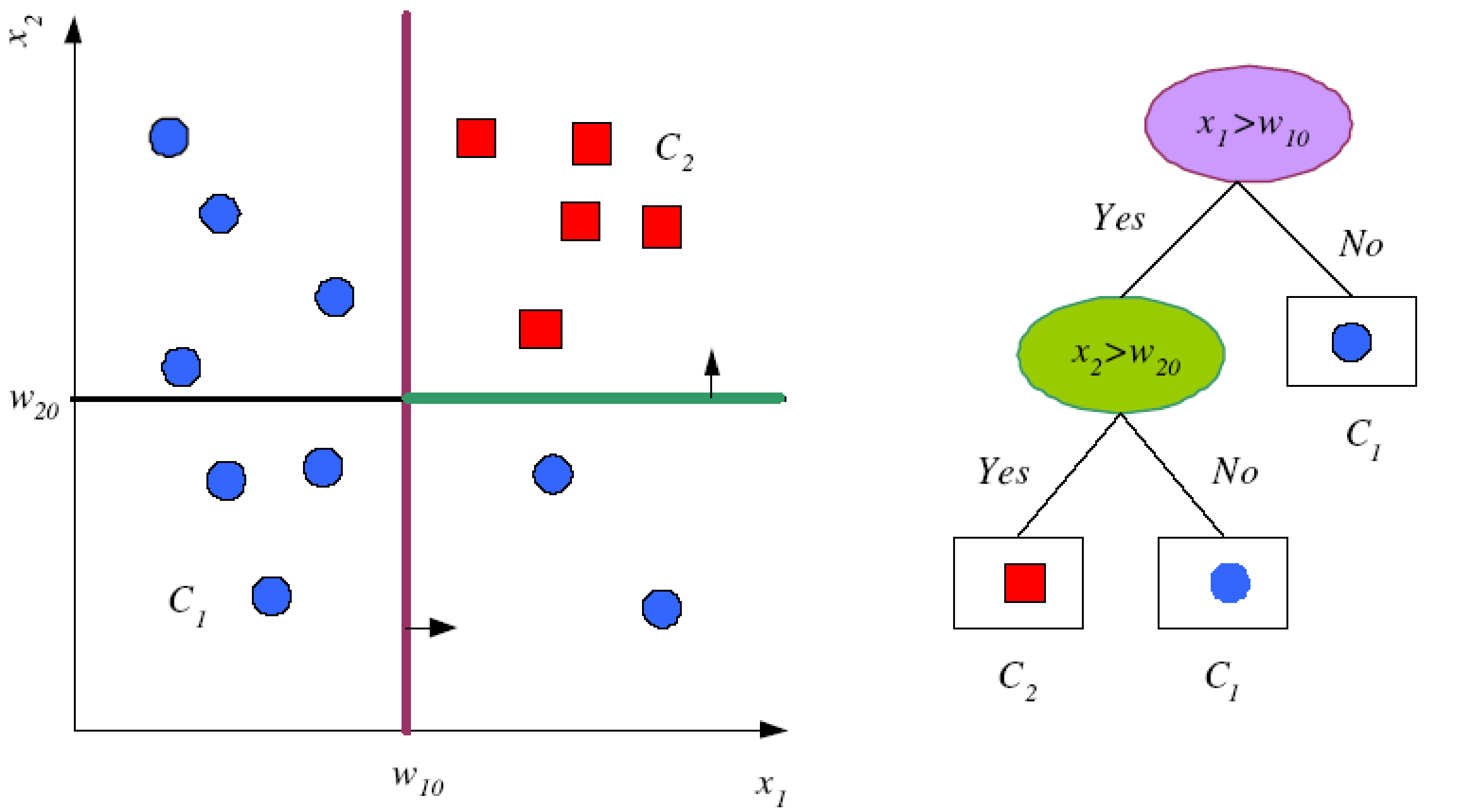
DECİSİON TREES

SECTIONS 9.1 ~ 9.2.1

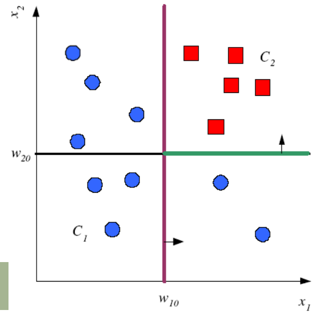
Decision tree

- Motivation
  - Explanable model with rules (if-then-else)
  - Usually not as accurate as other models
  - Classification, regression, etc
- How to learn a decision tree from data
  - Recursive
  - Greedily split the dataset into subgroups
  - Multiple hyperparameters and fine calibration

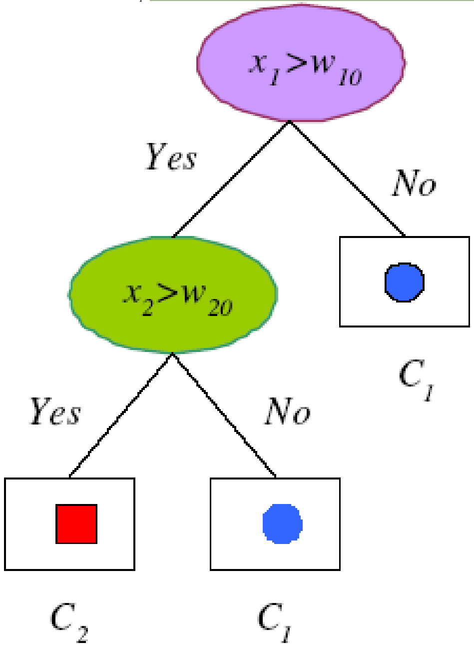
Tree Uses Nodes and Leaves

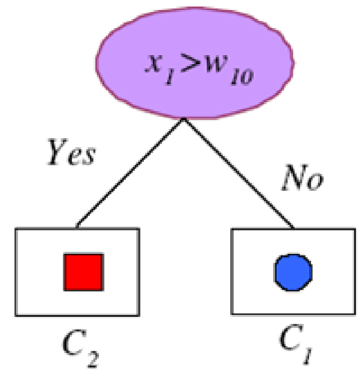


Two Types of Nodes

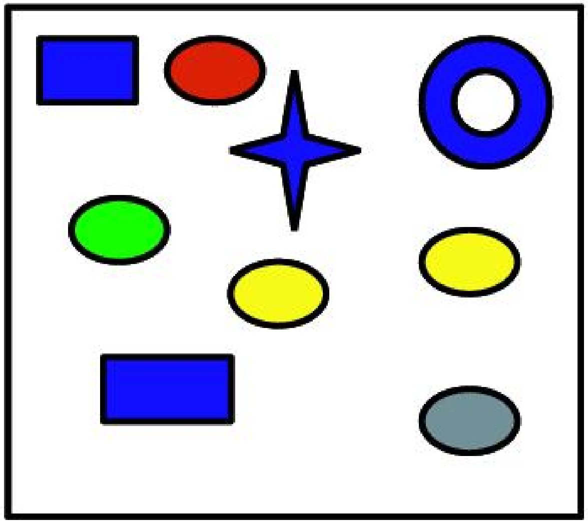


- **Terminal nodes (leafs)**
  - Provide the class (prediction)
- **Internal nodes**
  - Test the value of one or more features
  - Branch (split the data) based on the test
  - Root
  - Can have two or more children (not 1)
  - Variables can be numeric/discrete
- Terminal node does not have to be pure
- How to learn a decision tree from data
  - Choose (feature, branch) for internal nodes

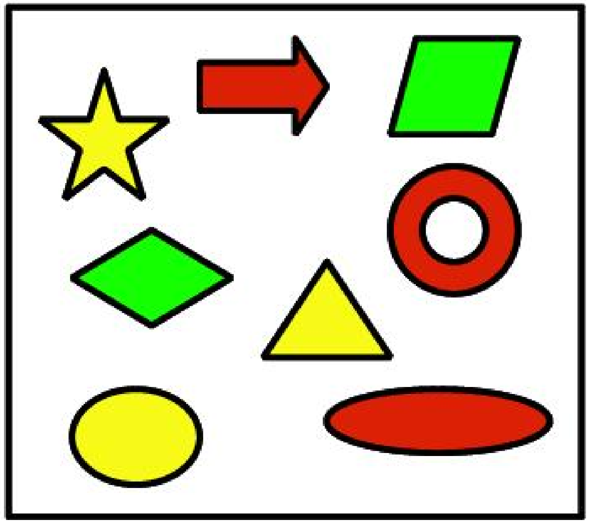




pos



neg

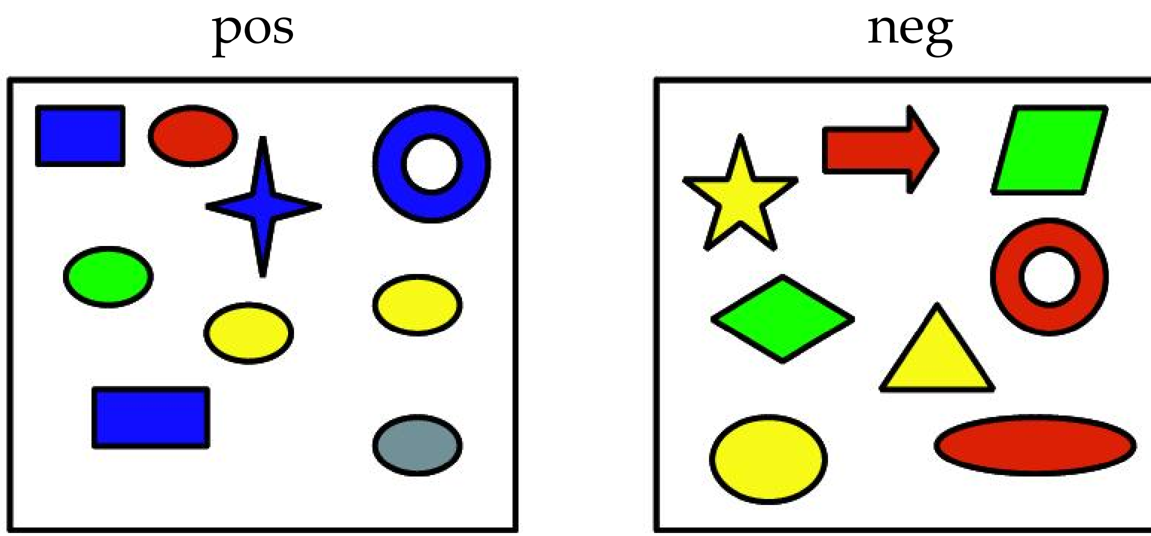


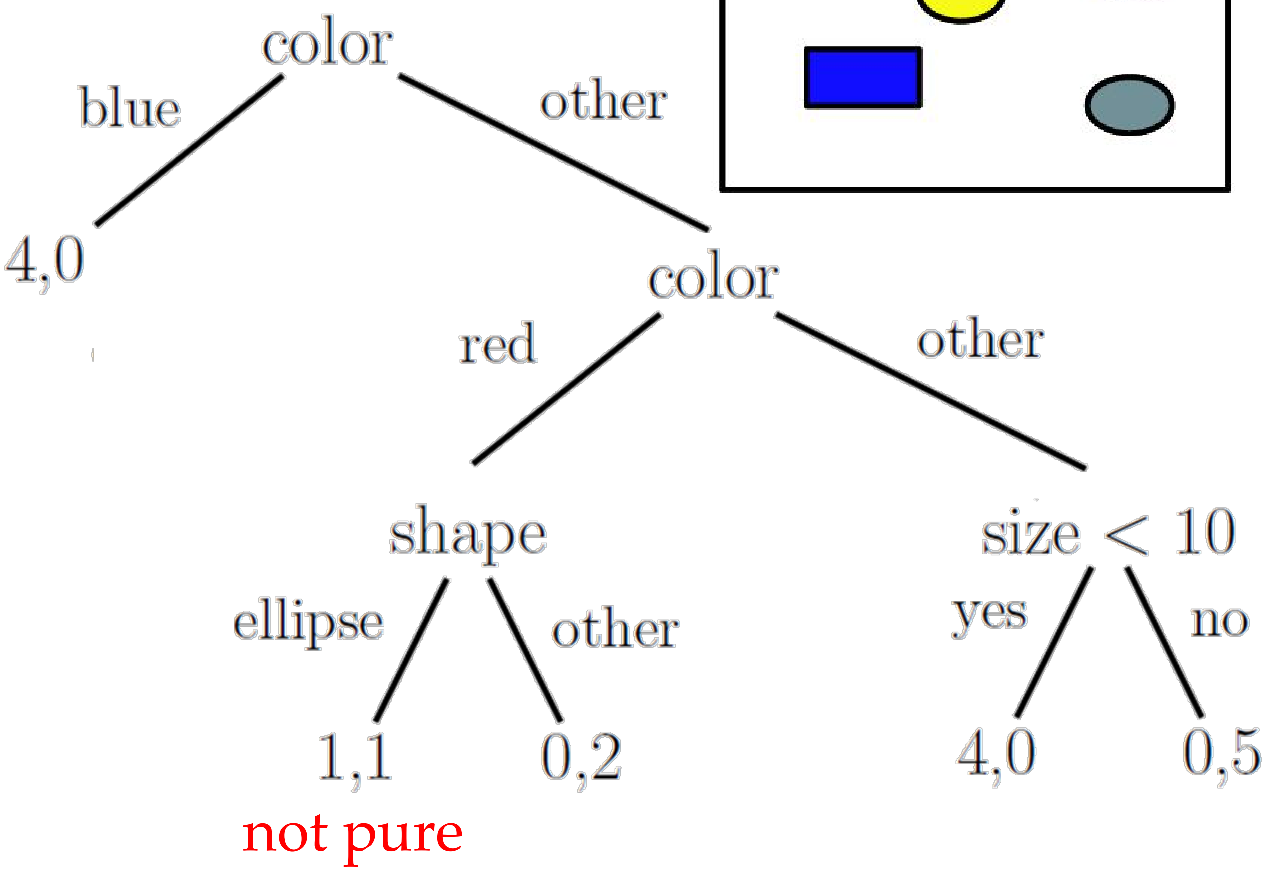
D features (attributes)

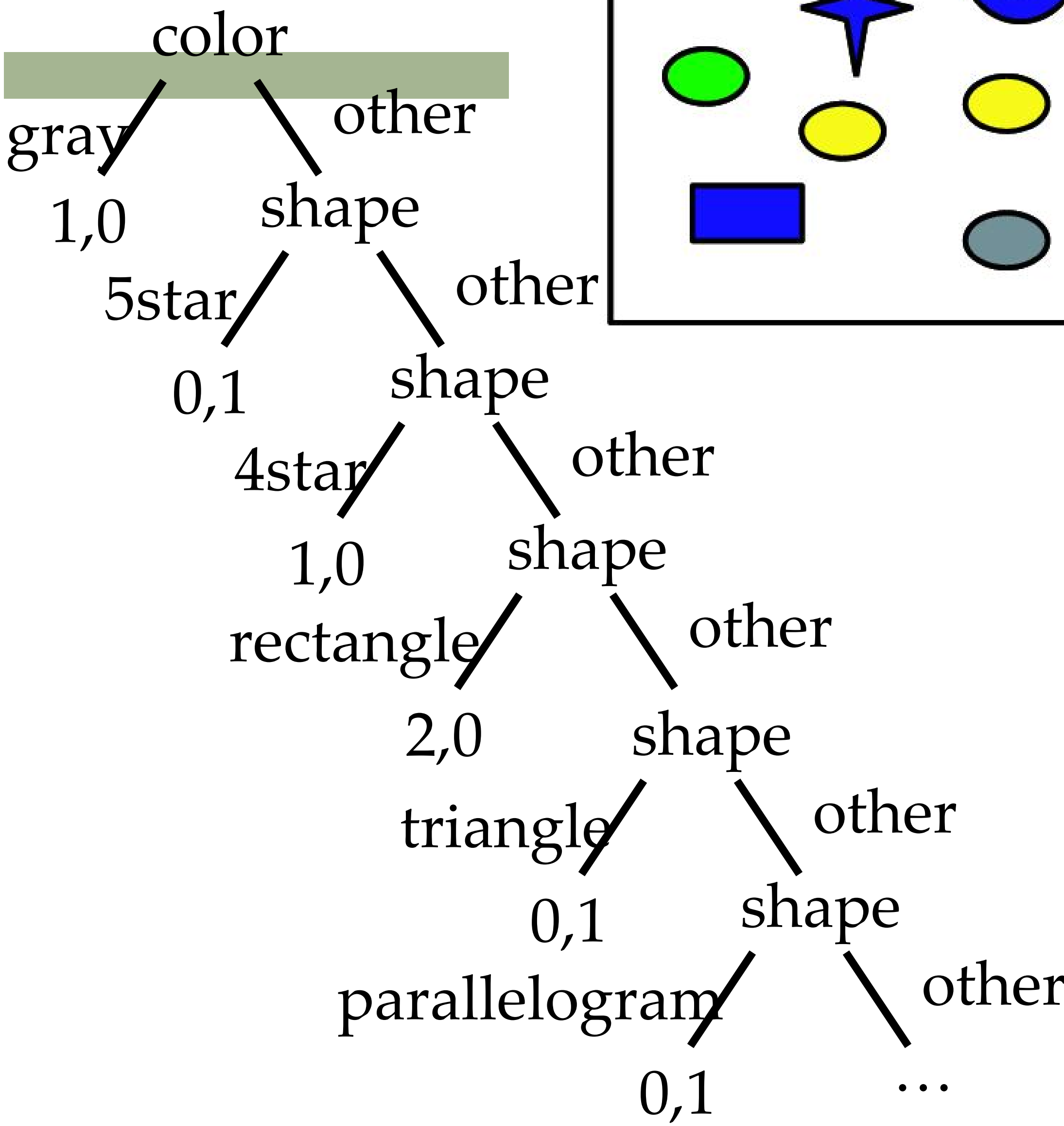
N cases

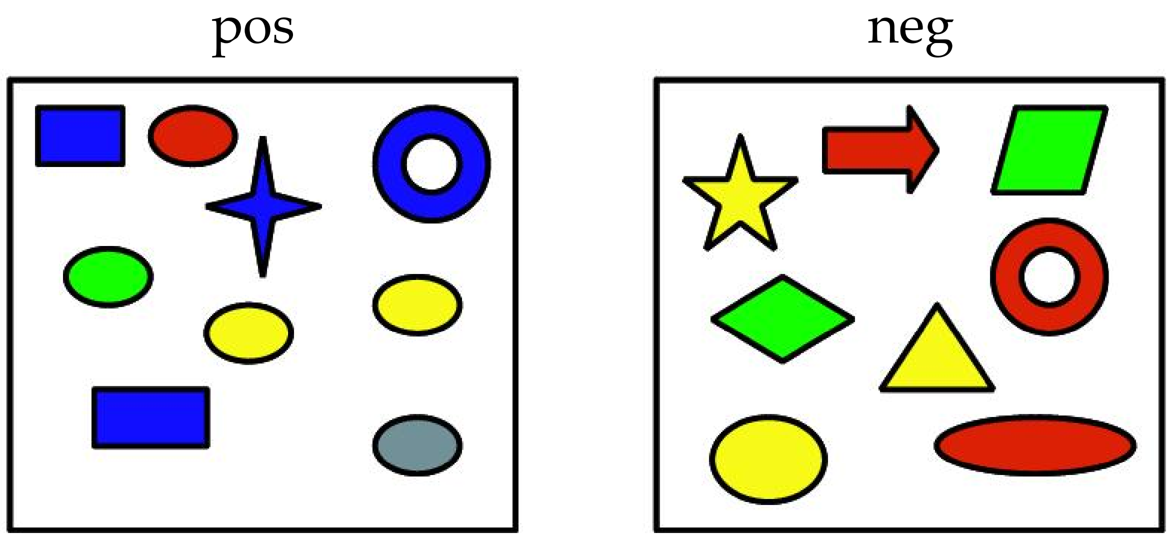
Color	Shape	Size (cm)
Blue	Square	10
Red	Ellipse	2.4
Red	Ellipse	20.7

Label
1
1
0

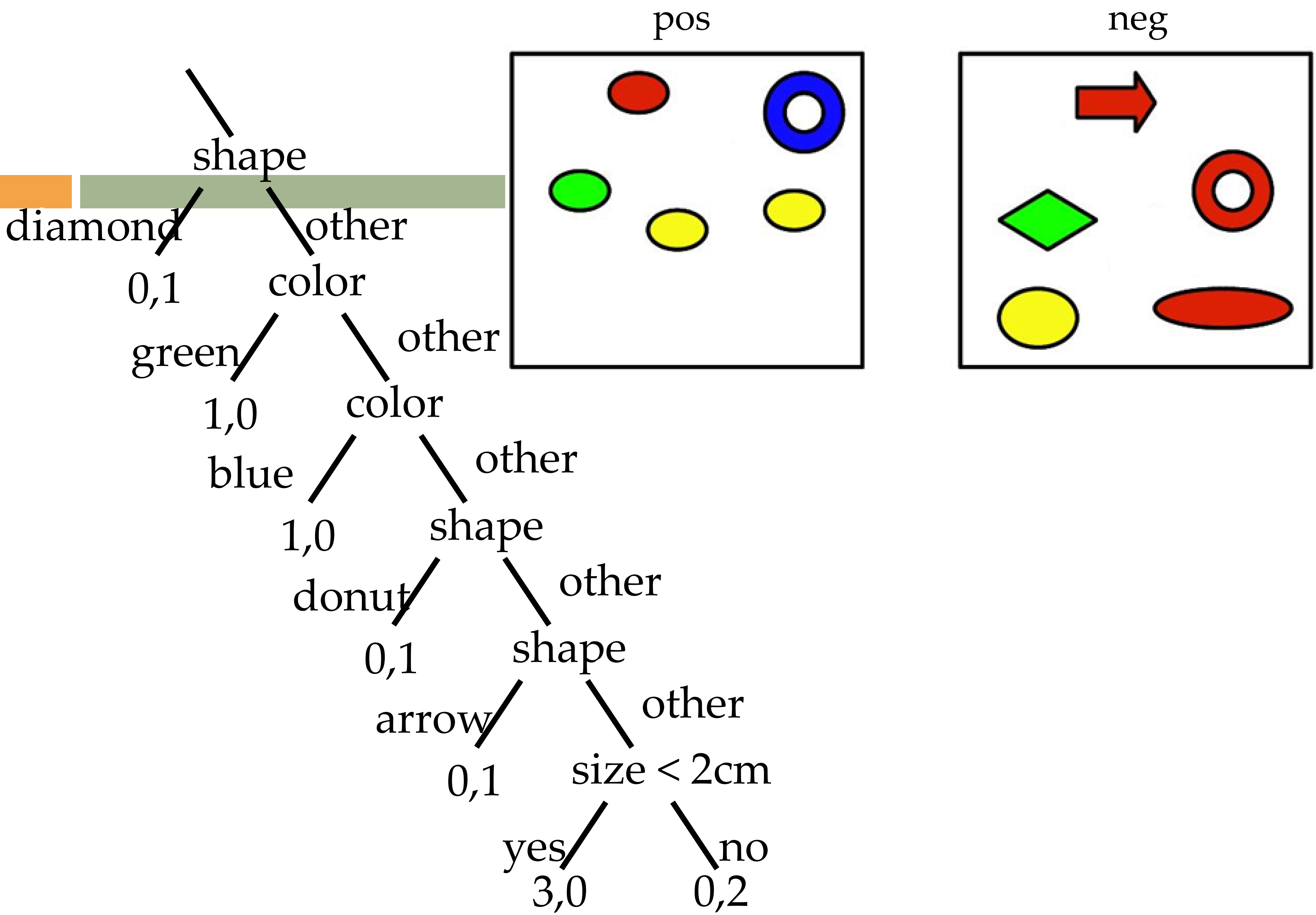

pos
neg

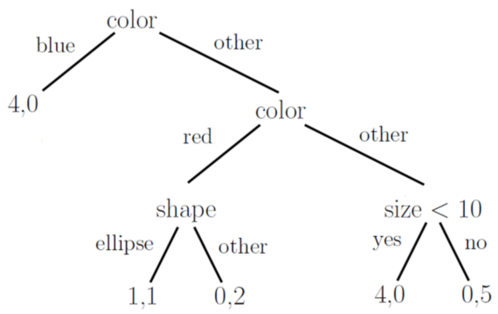

color
blue
other
4,0
color
red
other
shape
size < 10
ellipse
other
yes
no
1,1
0,2
4,0
0,5
not pure





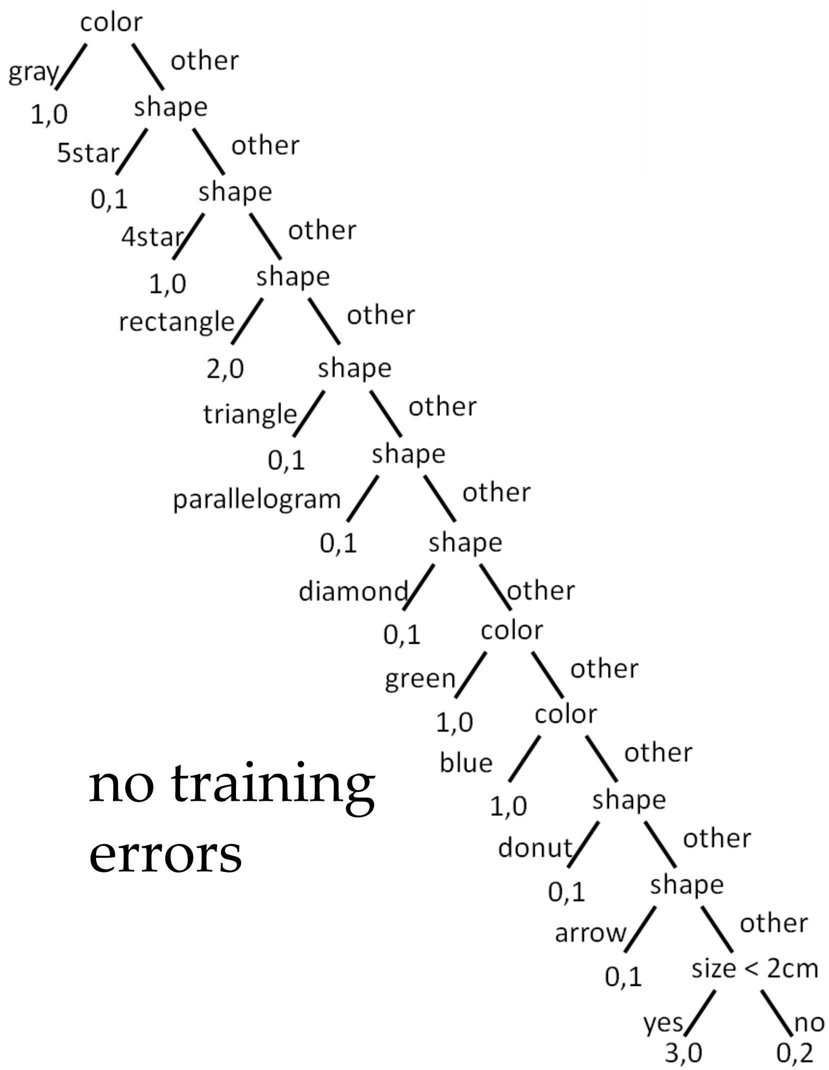
we can be more specific


pos
neg
shape
diamond
other
0,1
color
green
other
1,0
color
blue
other
1,0
shape
donut
other
0,1
shape
arrow
other
0,1
size < 2cm
yes
no
3,0
0,2



1 training error

or

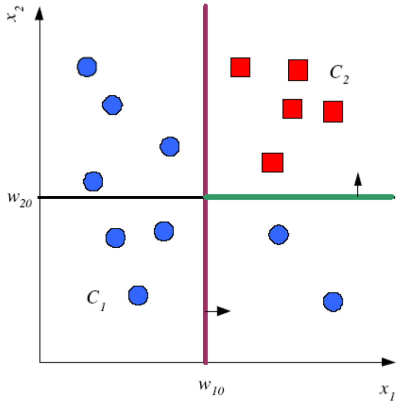


no training errors

Decision tree: divide and conquer

- Supervised learning: classification and regression
- Hierarchical, nonparametric (no a priori structure)
- Internal decision nodes implements a test function
  - Univariate: Uses a single attribute, x_i
    - Numeric x_i : Binary split : $x_i > w_m$
    - Discrete x_i : n-way split for n possible values or binary
  - Multivariate: Uses multiple/all attributes, $\boldsymbol{x}$
- Leaves
  - Classification: Class labels, or proportions
  - Regression: Numeric; r average, or local fit
- Highly interpretable (compare with Naïve Bayes)
- Learning is greedy; find the best split recursively (Breiman et al, 1984; Quinlan, 1986, 1993)

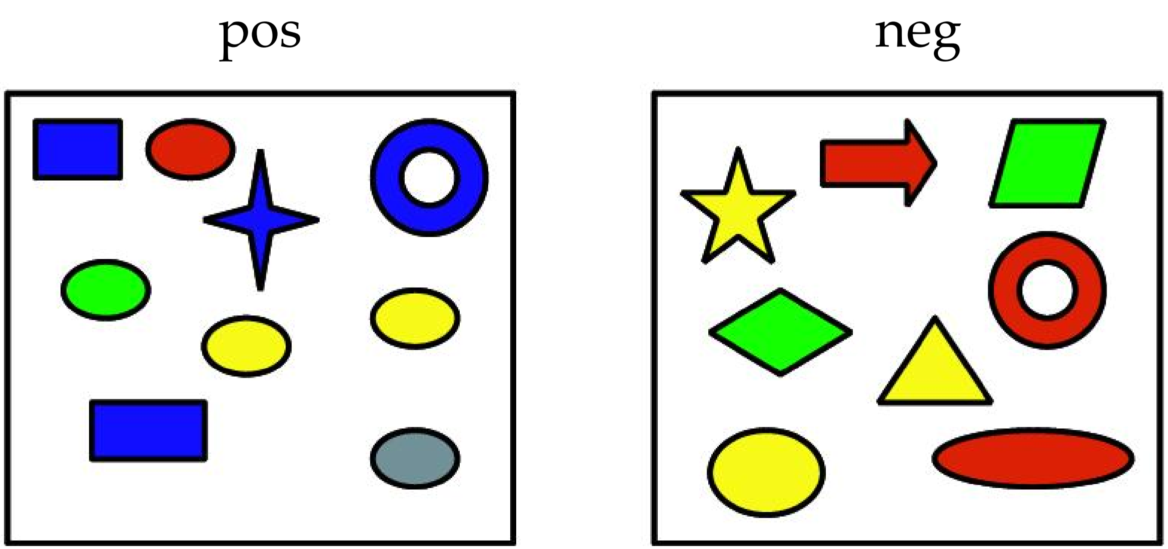
Classification Trees (ID3,CART,C4.5)

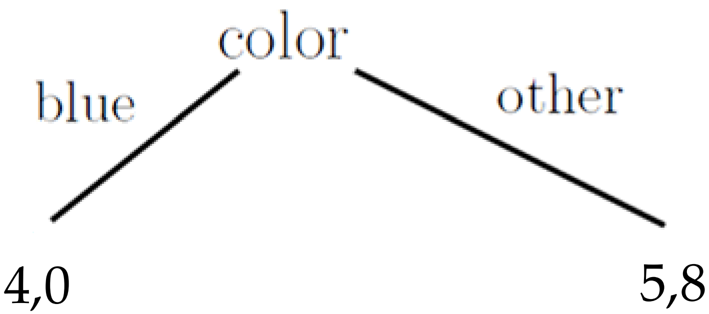


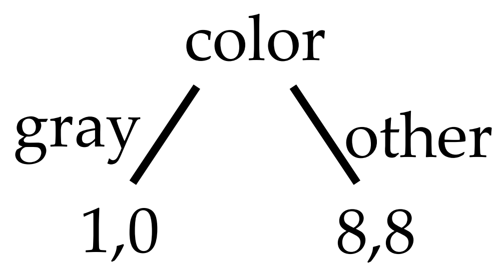
- Algorithms differ in branching models
  - Pick a feature x_j
  - Discrete case (with *n* values): split into *n* branches
  - Numeric case: discretize into **two** by thresholding

    $f_m(x) : x_j > w_{m0}$ (threshold)
- Goal: find the smallest tree that has low/zero training error
  - Small means depth, number node, breath, branching factor, etc
- NP-complete, forced to use local search based on heuristic
- Greedy: each step we look for the best split

  Score(D_1, D_2,… D_k) measuring "goodness" of splitting the data into *k* subsets
- Continue recursively until no more split (leaf node)





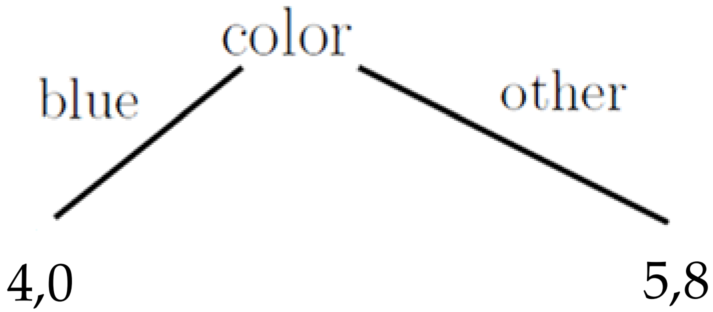


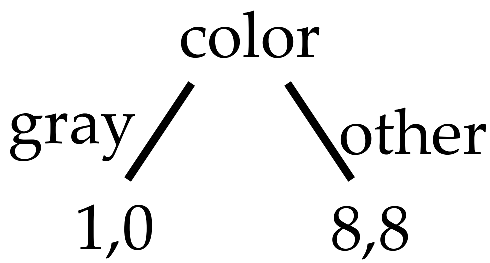
Error rate: 5/17

Error rate: 8/17

Use majority label at the leaf, then compute error rate
Also equal to the weighted average of error on both groups.

Selection of feature and splitting





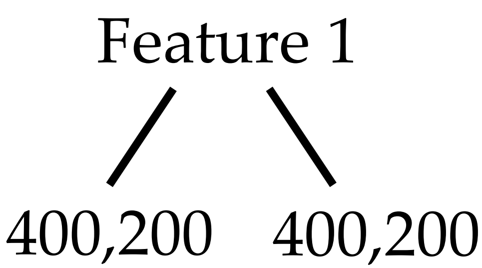
Error rate: 5/17

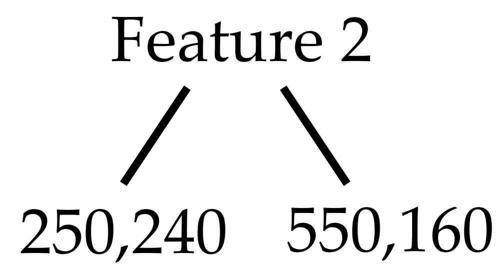
Error rate: 8/17

Select the feature and splitting value that "progresses most" towards a lower error.

best_loss = infinity
for feat in all possible features, i.e., {shape, color, size, …}
    for v in all possible values of feat (e.g., {red, blue, green yellow} for color)
        score = **error of splitting the current dataset along (feat, v)**
        if score < best_loss
            best_loss = score, and record feat and v

Accuracy score pitfall





Error rate: (200+200)/1200

Error rate: (240+160)/1200

Both have the same error rate!

Which one is “progressing more” towards a better solution?

best_loss = infinity
for feat in all possible features, i.e., {shape, color, size, …}
    for v in all possible values of feat (e.g., {red, blue, green yellow} for color)
        score = **some smart cost of splitting the current dataset along (feat, v)**
        if score < best_loss
            best_loss = score, and record feat and v

Best split in classification

- For node *m*, N_m instances reach *m*, N^i_m belong to C_i, then $\hat{P}(C_i \mid \mathbf{x}, m) \equiv p^i_m = \frac{N^i_m}{N_m}$
- Node *m* is pure if $p^i_m = 1$ for a certain *i*
- If node *m* is pure, generate a leaf and stop, otherwise split and continue recursively
- Measure of impurity is entropy:

$$\mathcal{I}_m = -\sum_{i=1}^{K} p^i_m \log_2 p^i_m$$

Entropy as an impurity measure

Measure of (degree of) uncertainty

The more clueless I am about the answer initially, the more information is contained in the answer

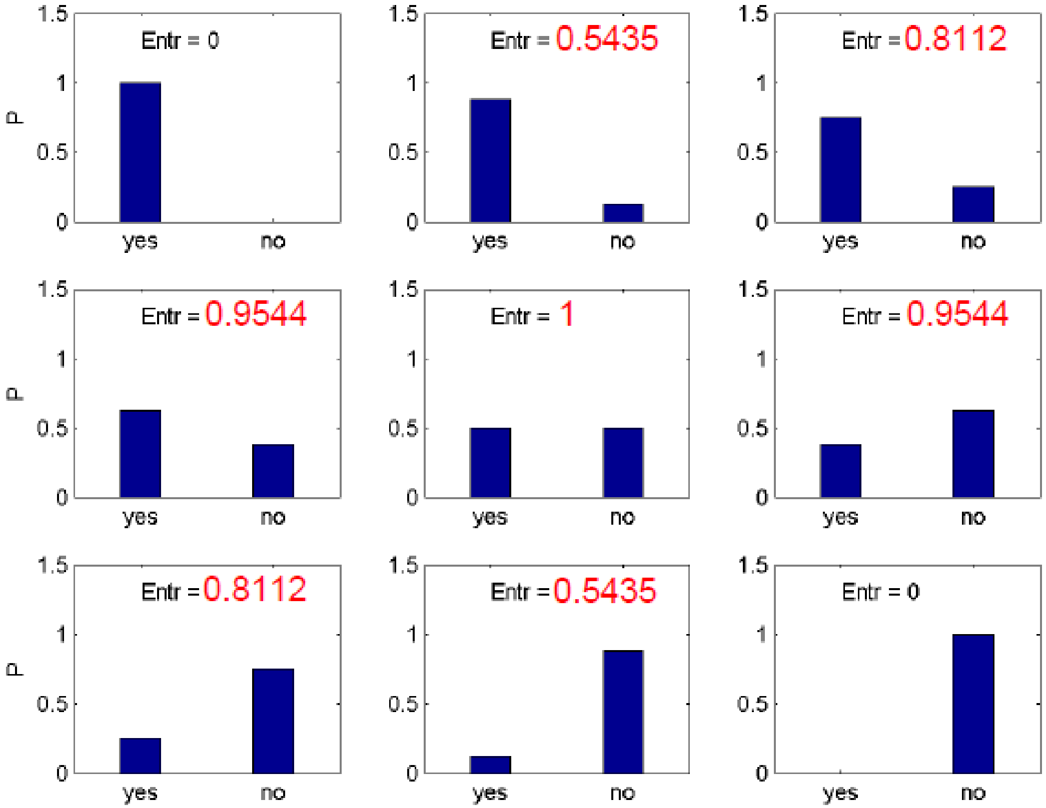
Information in an answer when prior is (P_1, …, P_n)

$$\Sigma_{i=1}^{n} - P_i \log_2 P_i$$

Scale: 1 bit = answer to Boolean question with equal prior

Roll of a 4-sided die has 2 bits of information.

Acquisition of information leads to reduction in entropy

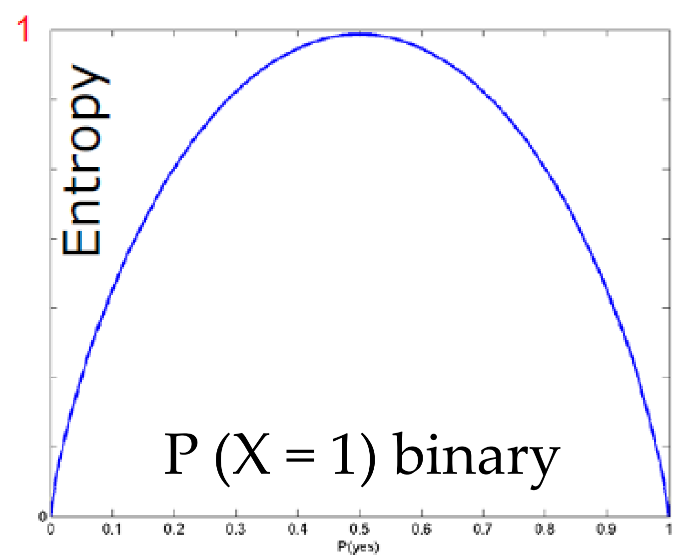


The entropy is maximal when all possibilities are equally likely.

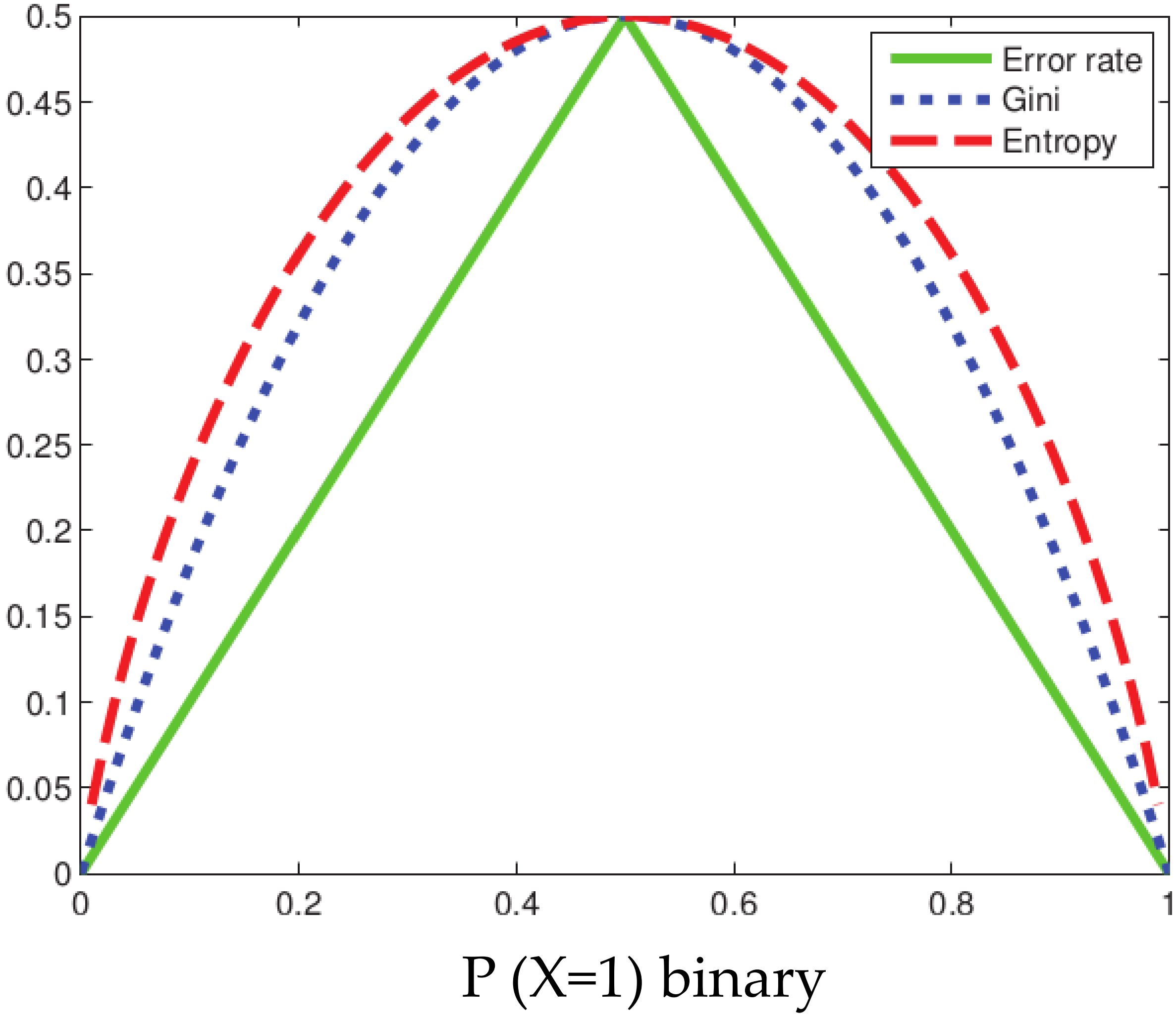
The goal of the decision tree is to decrease the entropy in each node.

Entropy is zero in a pure "yes" node (or pure "no" node).

Entropy is a measure of "uncertainty" of a random variable.



Impurity measures



Entropy:

$$\sum_{i=1}^{n} - P_i \log_2 P_i$$

= (in the binary case)

$$-p \log p \ - (1-p) \log p$$

The curve for entropy is rescaled (halved) on the left.

Gini index:

$$1 - \sum_{i=1}^{n} P_i^2$$

Error rate:

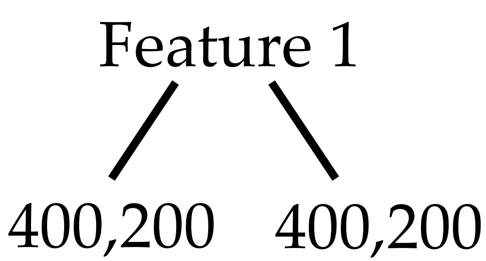
$$1 - \max_{i=1\ldots n} P_i$$

Best split in classification

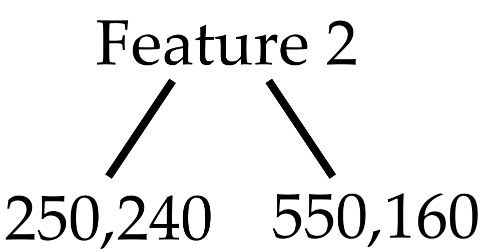
- If node m is pure, generate a leaf and stop, otherwise split and continue recursively
- Impurity after split: N_{mj} of N_m take branch j. N^i_{mj} belong to C_i

$$\hat{P}(C_i \mid \mathbf{x}, m, j) \equiv p^i_{mj} = \frac{N^i_{mj}}{N_{mj}} \qquad \mathcal{I}'_m = -\sum_{j=1}^{n} \frac{N_{mj}}{N_m} \sum_{i=1}^{K} p^i_{mj} \log_2 p^i_{mj}$$

- Find the variable and split that best reduces impurity (among all variables -- and split positions for numeric variables)



Error rate: 400/1200
Entropy: 0.92



Error rate: 400/1200
Entropy: 0.86

Best split in classification

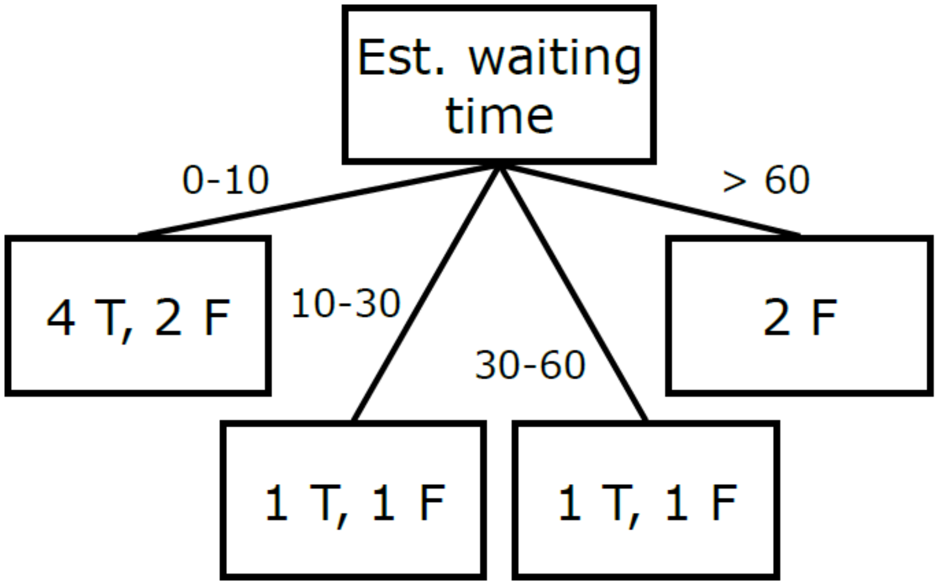
- If node m is pure, generate a leaf and stop, otherwise split and continue recursively
- Impurity after split: N_{mj} of N_m take branch j. N^i_{mj} belong to C_i

$$\hat{P}(C_i \mid \mathbf{x}, m, j) \equiv p^i_{mj} = \frac{N^i_{mj}}{N_{mj}} \qquad \mathcal{I}'_m = -\sum_{j=1}^{n} \frac{N_{mj}}{N_m} \sum_{i=1}^{K} p^i_{mj} \log_2 p^i_{mj}$$

- Find the variable and split that best reduces impurity (among all variables -- and split positions for numeric variables)

best_loss = infinity
for feat in all possible features, i.e., {shape, color, size, …}
    for v in all possible values of feat (e.g., {red, blue, green yellow} for color)
        score = **impurity** after splitting along (feat, v)
        if score < best_loss
            best_loss = score, and record feat and v

Information Gain Example



Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

More details at Decision_tree_example.pdf in the slides section

$$\boxed{Remainder(Wait)} = \frac{6}{12}\left[-\left(4/6\right)\log_2\left(4/6\right)-\left(2/6\right)\log_2\left(2/6\right)\right]$$

$$+\frac{2}{12}\left[-\left(1/2\right)\log_2\left(1/2\right)-\left(1/2\right)\log_2\left(1/2\right)\right]$$

$$+\frac{2}{12}\left[-\left(1/2\right)\log_2\left(1/2\right)-\left(1/2\right)\log_2\left(1/2\right)\right]+\frac{2}{12}\left[-\left(0/2\right)\log_2\left(0/2\right)-\left(2/2\right)\log_2\left(2/2\right)\right]=0.7925$$

$\boxed{\text{Gain(Wait)}}$ = 1 – 0.7925 = 0.2075

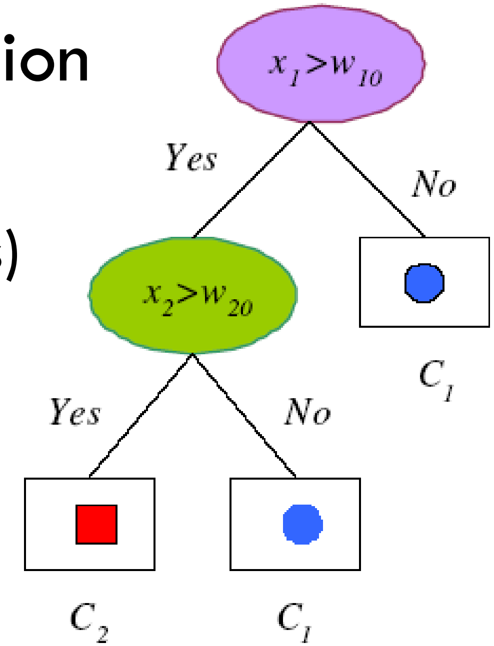
```
GenerateTree(𝒳)
    If NodeEntropy(𝒳)< θ_I  /* eq. 9.3
        Create leaf labelled by majority class in 𝒳
        Return
    i ← SplitAttribute(𝒳)
    For each branch of x_i
        Find 𝒳_i falling in branch
        GenerateTree(𝒳_i)
SplitAttribute(𝒳)
    MinEnt← MAX
    For all attributes i = 1, ..., d
        If x_i is discrete with n values
            Split 𝒳 into 𝒳_1, ..., 𝒳_n by x_i
            e ← SplitEntropy(𝒳_1, ..., 𝒳_n)
            If e<MinEnt MinEnt ← e; bestf ← i
        Else /* x_i is numeric */
            For all possible splits
                Split 𝒳 into 𝒳_1, 𝒳_2 on x_i
                e←SplitEntropy(𝒳_1, 𝒳_2)
                If e<MinEnt MinEnt ← e; bestf ← i
    Return bestf
```

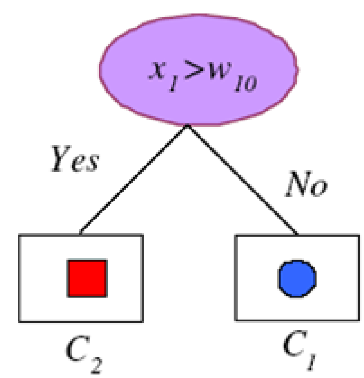
$$I'_m = -\sum_{j=1}^{n} \frac{N_{mj}}{N_m} \sum_{i=1}^{K} p^i_{mj} \log_2 p^i_{mj}$$

computational cost?

Pruning Trees

- Remove subtrees for better generalization (decrease variance)
  - Prepruning: Early stopping (e.g. < 5% points)
  - Postpruning: Grow the whole tree then prune subtrees that overfit on the pruning set
  - Set aside a subset of data for pruning
- Prepruning is faster, postpruning is more accurate (requires a separate pruning set)





Decision tree

- Motivation
  - Explanable model with rules (if-then-else)
  - Usually not as accurate as other models
  - Classification, regression, etc
- How to learn a decision tree from data
  - Recursive
  - Greedily split the dataset into subgroups
  - Various impurity measures
  - Pruning