Supervised Learning via Decision Trees

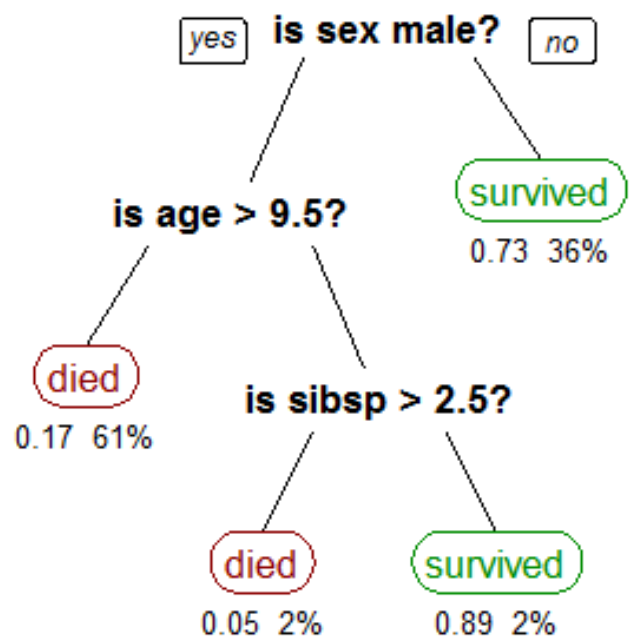
Lecture 4

Outline

1. Learning via feature splits
2. ID3
   - Information gain
3. Extensions
   - Continuous features
   - Gain ratio
   - Ensemble learning

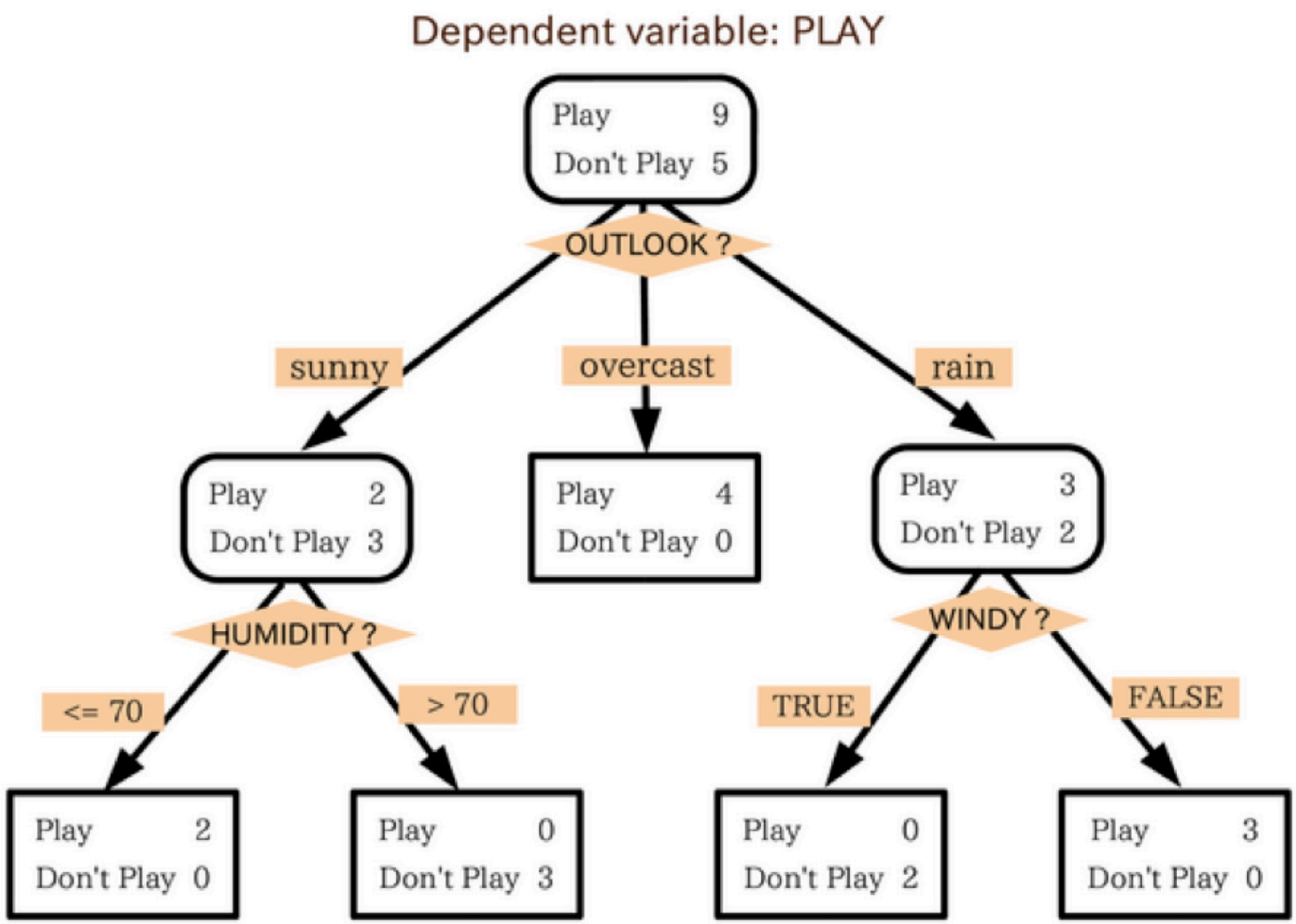
Decision Trees

- Sequence of decisions at *choice nodes* from root to a leaf node
  - Each choice node splits on a single feature
- Can be used for classification or regression
- Explicit, easy for humans to understand
- Typically very fast at testing/ prediction time

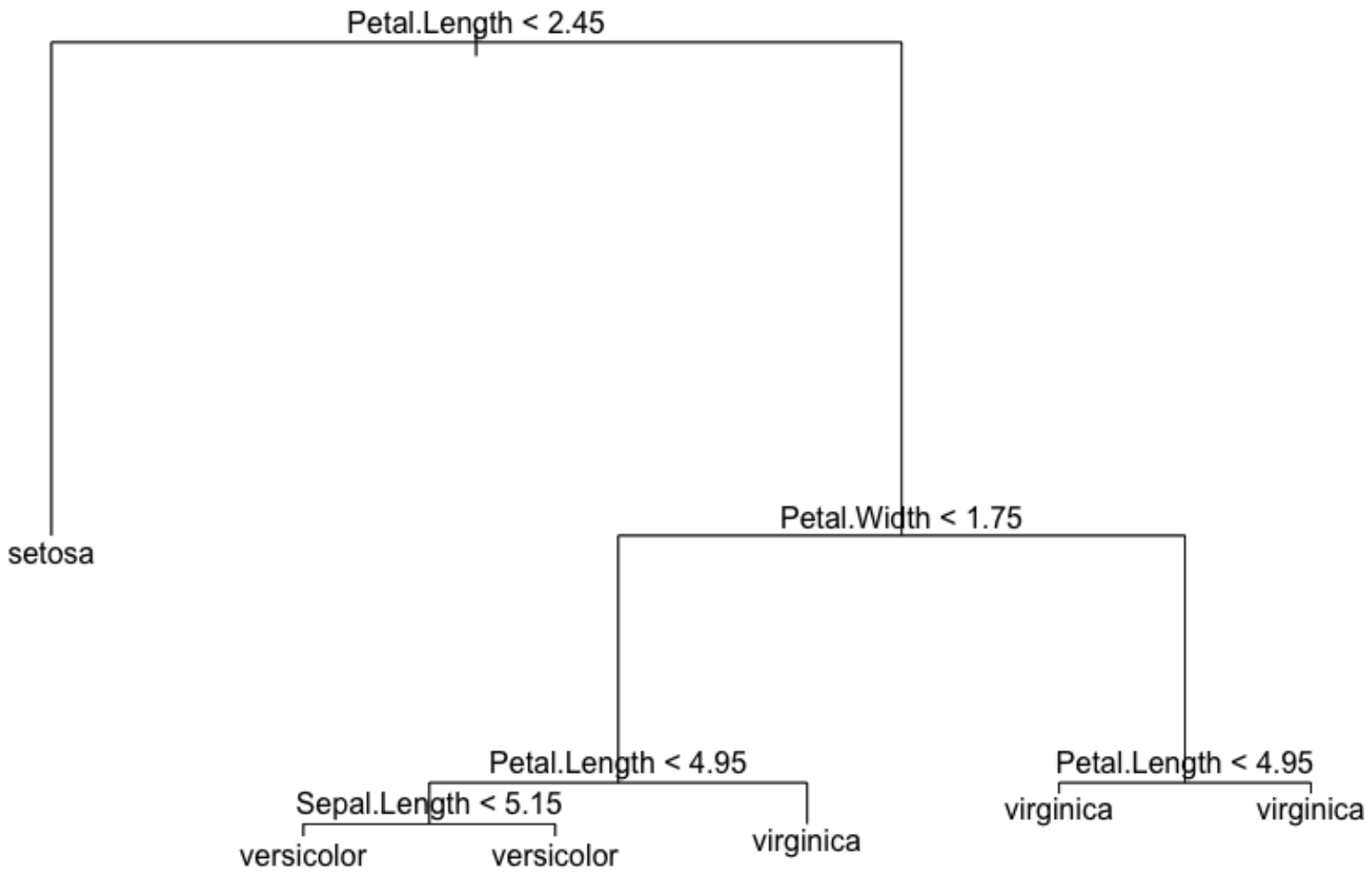


https://en.wikipedia.org/wiki/Decision_tree_learning

Weather Example



IRIS Example



Training Issues

- Approximation
  - Optimal tree-building is NP-complete
  - Typically greedy, top-down
- Bias vs. Variance
  - Occam's Razor vs. CC/SSN
    - Pruning, ensemble methods
- Splitting metric
  - **Information gain**, **gain ratio**, Gini impurity

Iterative Dichotomiser 3

- Invented by Ross Quinlan in 1986
  - Precursor to C4.5/5
- Categorical data only
- Greedily consumes features
  - Subtrees cannot consider previous feature(s) for further splits
  - *Typically* produces shallow trees

ID3: Algorithm Sketch

- If all examples "same", return *f*(examples)
- If no more features, return *f*(examples)
- A = "best" feature
  - For each distinct value of A
    - branch = ID3(attributes - {A})

Details

Classification

- "same" = same class
- *f*(examples) = majority

Regression

- "same" = std. dev. < ε
- *f*(examples) = average

Recursion

- A method of programming in which a function refers to itself in order to solve a problem
  - Example: ID3 calls itself for subtrees

- Never necessary
  - In some situations, results in simpler and/or easier-to-write code
  - Can often be more expensive in terms of memory + time

Example

Consider the **factorial** function

$$n! = \prod_{k=1}^{n} k = 1 * 2 * 3 * \ldots * n$$

Iterative Implementation

```
def factorial(n):
  result = 1
  for i in range(n):
    result *= (i+1)
  return result
```

Consider a Recursive Definition

Base Case

$$0! = 1$$

$$n! = n(n-1)! \quad \text{when } n \geq 1$$

Recursive Step

Recursive Implementation

```
def factorial_r(n):
    if n == 0:
        return 1
    else:
        return (n * factorial_r(n-1))
```

How the Code Executes

factorial_r
```
return 1
```

factorial_r
```
return 1 * factorial_r( 0 )
```

factorial_r
```
return 2 * factorial_r( 1 )
```

factorial_r
```
return 3 * factorial_r( 2 )
```

factorial_r
```
return 4 * factorial_r( 3 )
```

main
```
print factorial_r( 4 )
```

Stack Frame

Function Stack

How the Code Executes

factorial_r
```
return 1 * 1
```

factorial_r
```
return 2 * factorial_r( 1 )
```

factorial_r
```
return 3 * factorial_r( 2 )
```

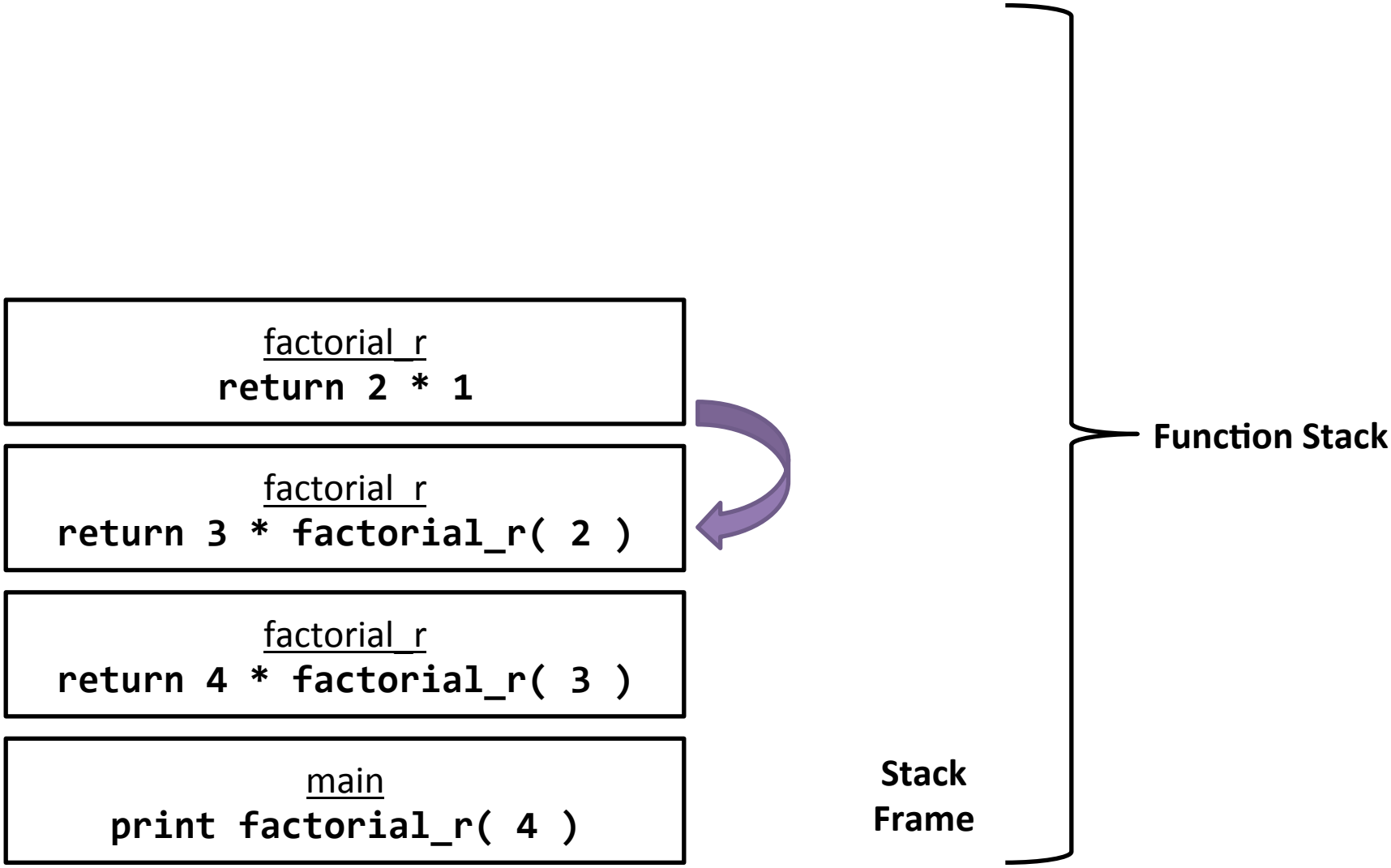
factorial_r
```
return 4 * factorial_r( 3 )
```

main
```
print factorial_r( 4 )
```

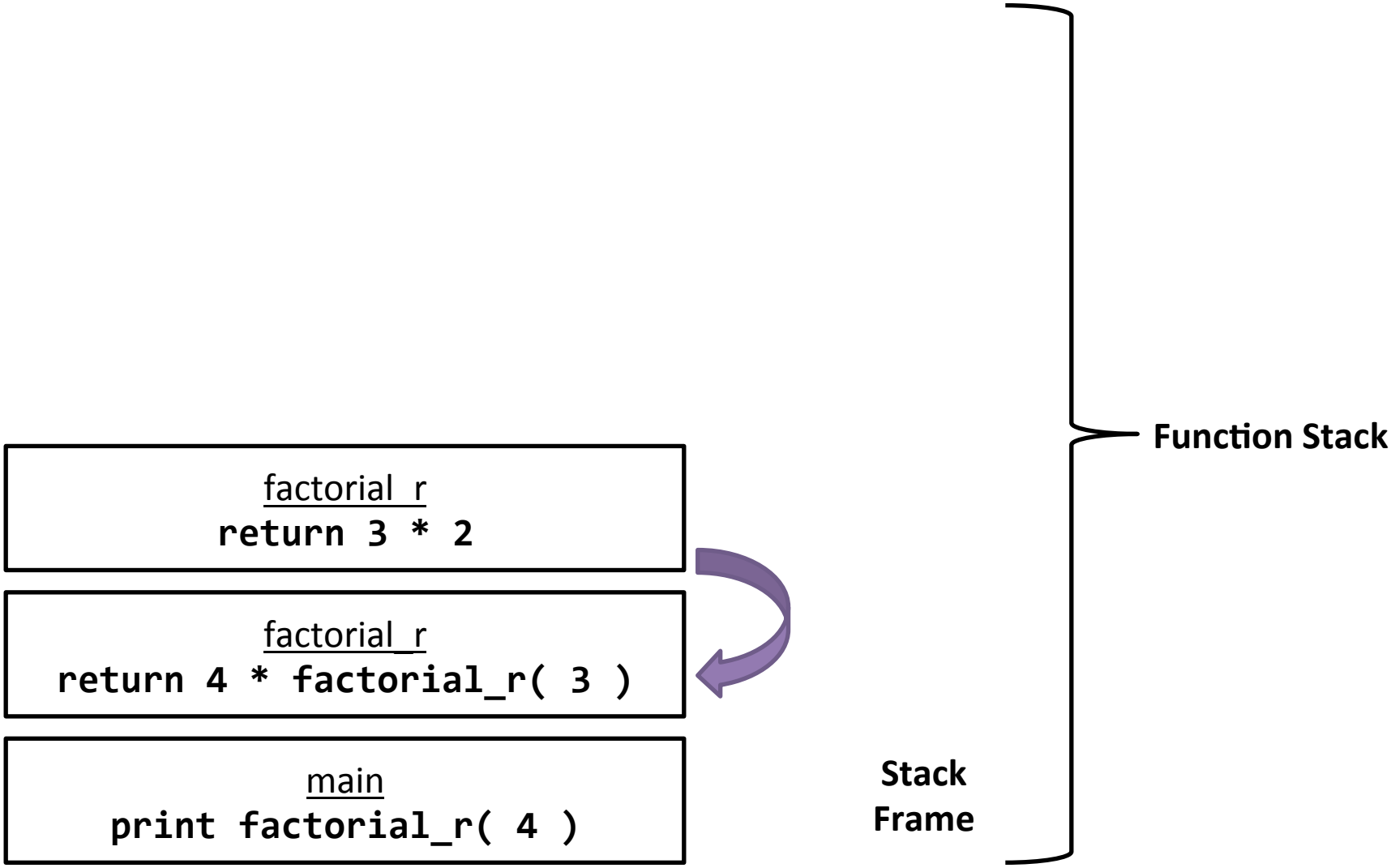
Stack Frame

Function Stack

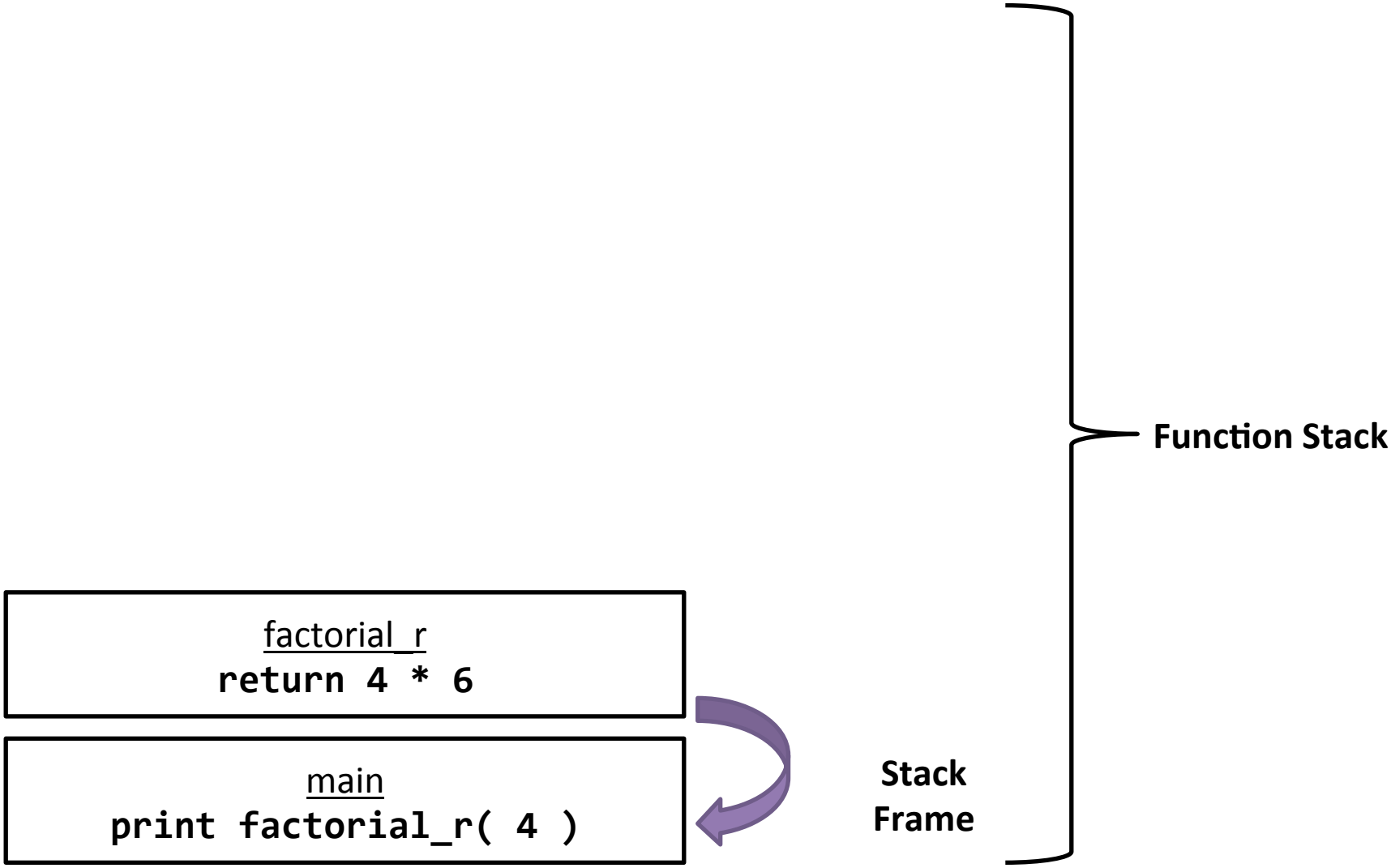
How the Code Executes



How the Code Executes



How the Code Executes



How the Code Executes

Function Stack

main
`print 24`

Stack Frame

ID3: Algorithm Sketch

- If all examples "same", return *f*(examples)
- If no more features, return *f*(examples)
- A = "best" feature
  - For each distinct value of A
    - branch = ID3(attributes - {A})

Base Cases

Recursive Step

Splitting Metric: The "best" Feature

Classification

- Information gain
  - Goal: choose splits that proceed from much->little uncertainty

Regression

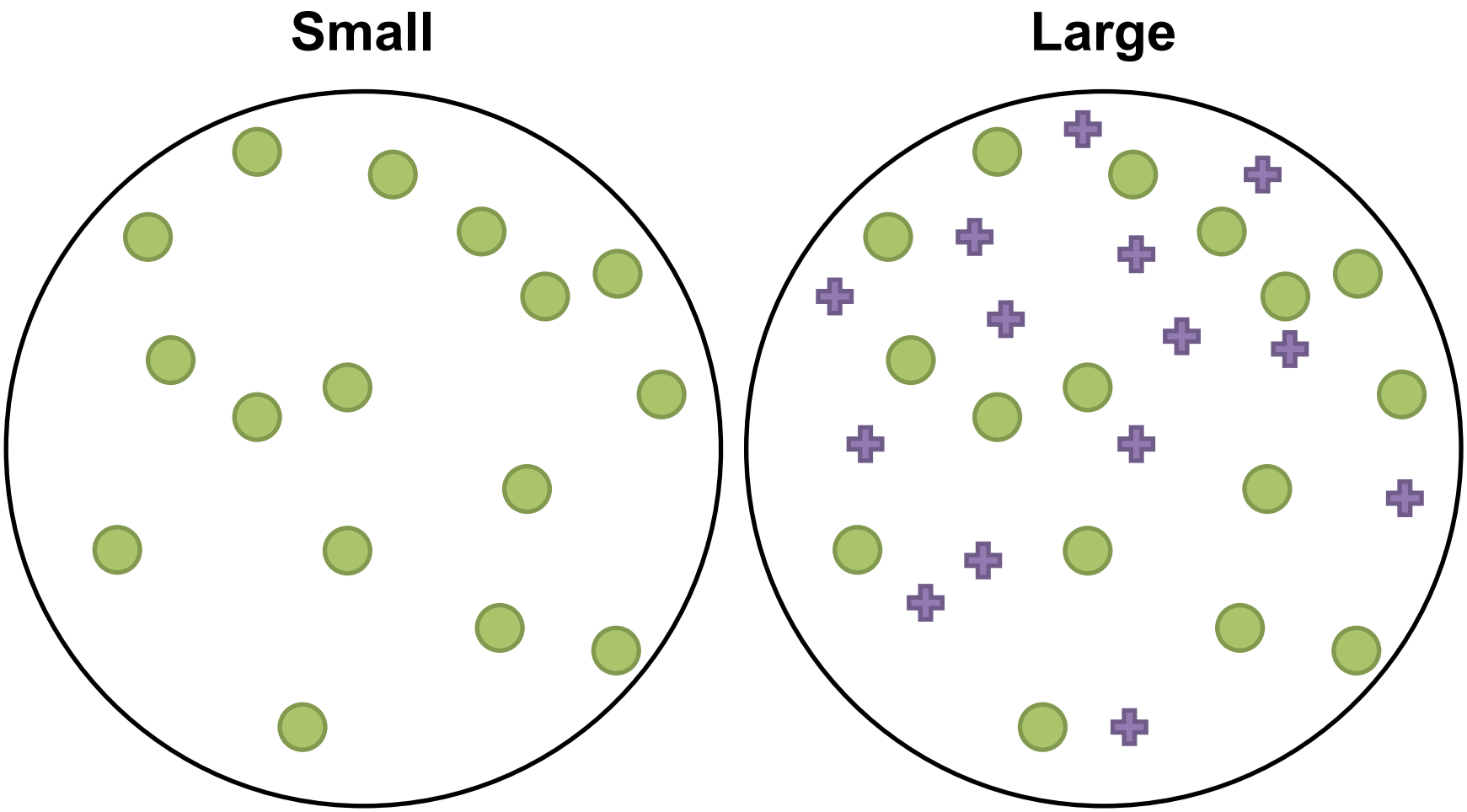
- Standard Deviation Reduction

http://www.saedsayad.com/decision_tree_reg.htm

Shannon Entropy

- Measure of “impurity” or uncertainty
- Intuition: the less likely the event, the more information is transmitted

Entropy Range



Quantifying Entropy

$$H(X) = E[I(X)]$$

Expected value of information

$$\sum_i P(x_i)I(x_i)$$

$$\int P(x)I(x)dx$$

Intuition for Information

$$I(X) = \ldots$$

$$I(X) \geq 0$$

- Shouldn't be negative

$$I(1) = 0$$

- Events that always occur communicate no information

$$I(X_1, X_2) = I(X_1) + I(X_2)$$

- Information from independent events are additive

Quantifying Information

$$I(X) = \log_b \frac{1}{P(X)} = -\log_b P(X)$$

Log Base = Units: 2=bit (**bi**nary digi**t**), 3=trit, e=nat

$$H(X) = -\sum_i P(x_i) \log_b P(x_i)$$

Log Base = Units: 2=shannon/bit

Example: Fair Coin Toss

$$\mathrm{I}(\text{heads}) = \log_2(\frac{1}{0.5}) = \log_2 2 = 1 \text{ bit}$$

$$\mathrm{I}(\text{tails}) = \log_2(\frac{1}{0.5}) = \log_2 2 = 1 \text{ bit}$$

$$\mathrm{H}(\text{fair toss}) = (0.5)(1) + (0.5)(1) =$$

$$= 1 \text{ shannon}$$

Example: Double Headed Coin

$$
\begin{aligned}
H(\text{double head}) &= (1) \cdot I(\text{head}) \\
&= (1) \cdot \log_2(\frac{1}{1}) \\
&= (1) \cdot (0) \\
&= 0 \text{ shannons}
\end{aligned}
$$

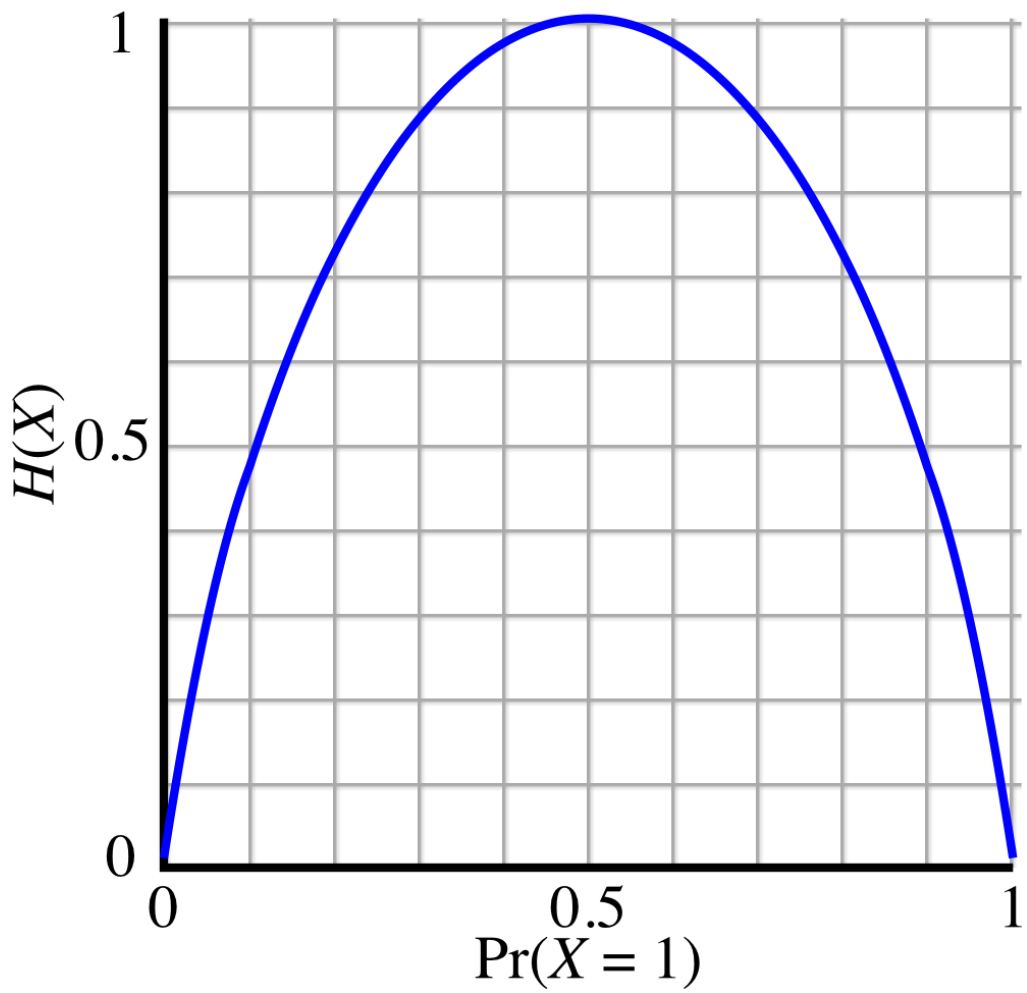
Exercise: Weighted Coin

Compute the entropy of a coin that will land on heads about 25% of the time, and tails the remaining 75%.

Answer

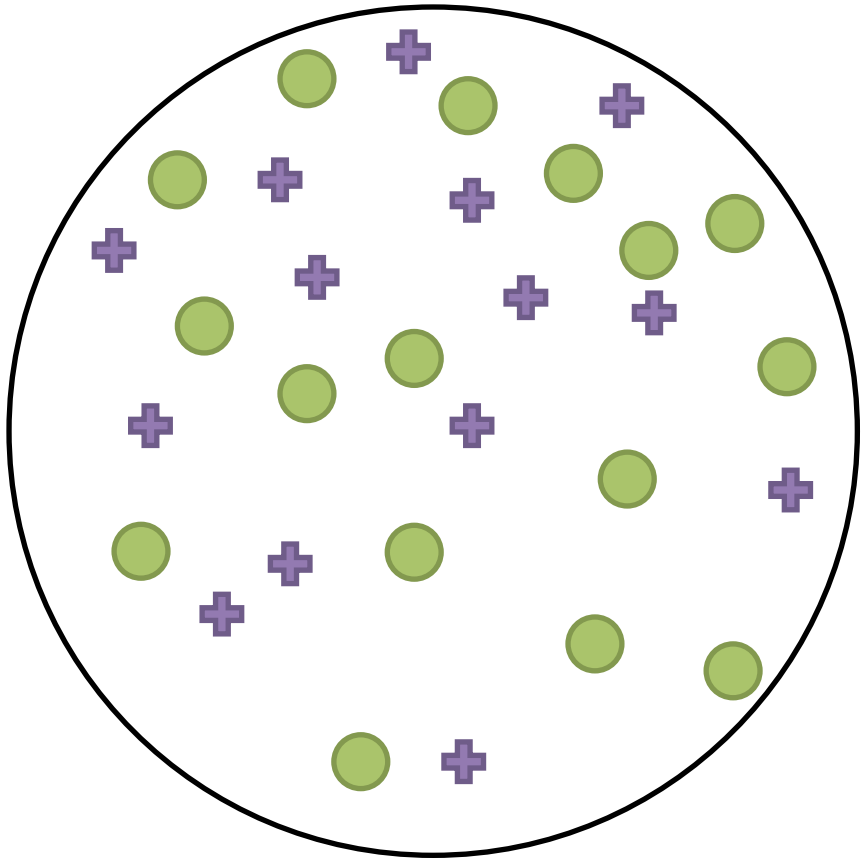
$$
\begin{aligned}
H(\text{weighted toss}) &= (0.25) \cdot I(\text{head}) + (0.75) \cdot I(\text{tails}) \\
&= (0.25) \cdot \log_2 \frac{1}{0.25} + (0.75) \cdot \log_2 \frac{1}{0.75} \\
&= 0.81 \text{ shannons}
\end{aligned}
$$

Entropy vs. P



Exercise

Calculate the entropy of the following data



Answer

$$
\begin{aligned}
H(\text{data}) &= \frac{16}{30} \cdot I(\text{green circle}) + \frac{14}{30} \cdot I(\text{purple cross}) \\
&= \frac{16}{30} \cdot \log_2 \frac{30}{16} + \frac{14}{30} \cdot \log_2 \frac{30}{14} \\
&= 0.99679 \text{ shannons}
\end{aligned}
$$

Bounds on Entropy

$$H(X) \geq 0$$

$$H(X) = 0 \iff \exists x \in X(P(x) = 1)$$

$$H_b(X) \leq \log_b(|\mathcal{X}|)$$

$|\mathcal{X}|$ denotes the number of elements in the range of X

$H_b(X) = \log_b(|\mathcal{X}|) \iff X$ has a uniform distribution over $|\mathcal{X}|$

Information Gain

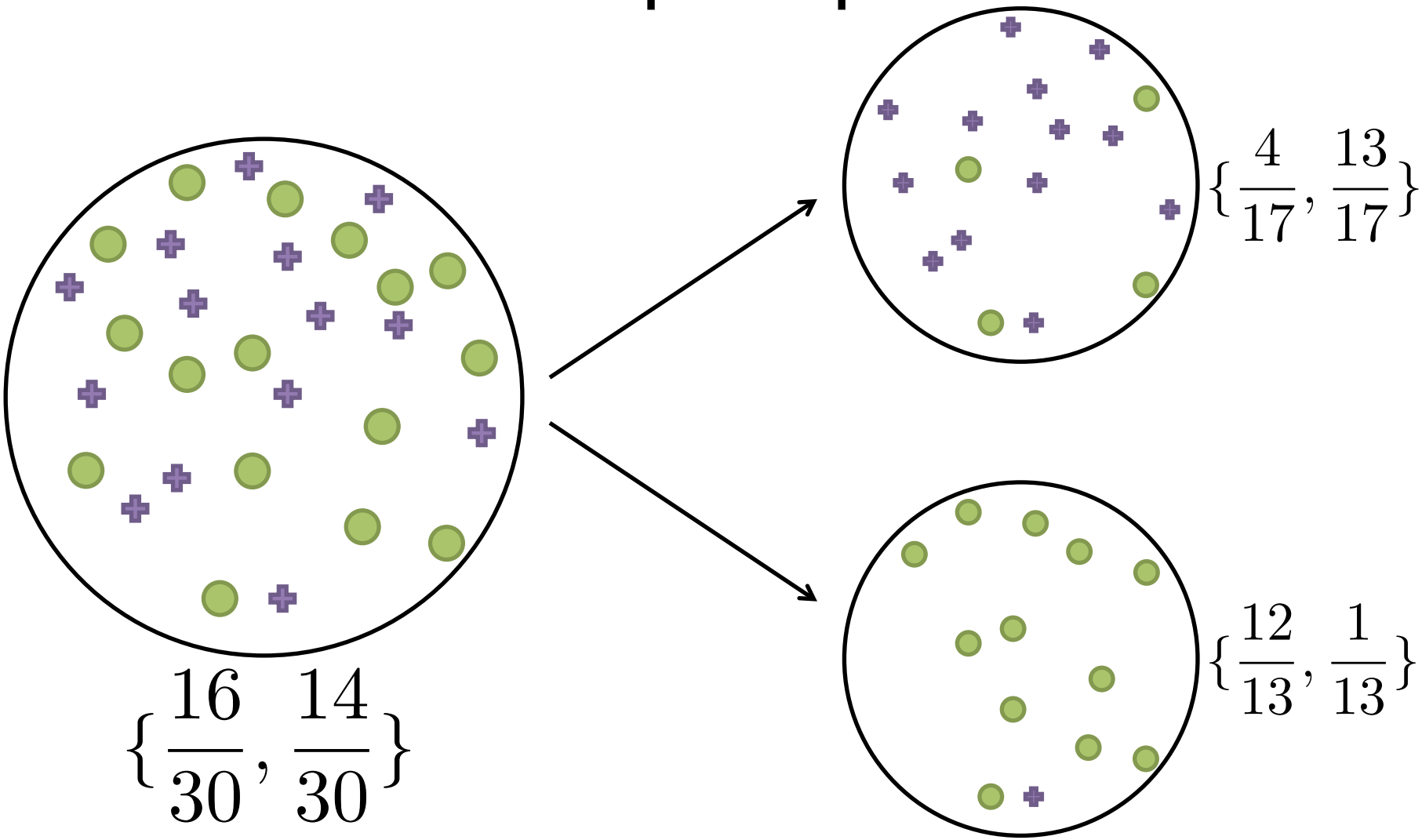
To use entropy for a splitting metric, we consider the **information gain** of an action as the resulting change in entropy

$$\mathrm{IG}(T, a) = \mathrm{H}(T) - \mathrm{H}(T|a)$$

$$= \mathrm{H}(T) - \sum_i \frac{|T_i|}{|T|} \mathrm{H}(T_i)$$

Average Entropy of the children

Example Split



Example Information Gain

$$\mathrm{H}_1 = \frac{4}{17}\log_2\frac{17}{4} + \frac{13}{17}\log_2\frac{17}{13} \sim 0.79$$

$$\mathrm{H}_2 = \frac{12}{13}\log_2\frac{13}{12} + \frac{1}{13}\log_2\frac{13}{1} \sim 0.39$$

$$\begin{aligned}\mathrm{IG} &= \mathrm{H}(T) - (\frac{17}{30}\mathrm{H}_1 + \frac{13}{30}\mathrm{H}_2) \\ &= 0.99679 - 0.62 \\ &= 0.38 \text{ shannons}\end{aligned}$$

Exercise

Consider the following dataset. Compute the information gain for each of the non-target attributes. Decide which attribute is the best to split on.

X	Y	Z	Class
1	1	1	A
1	1	0	A
0	0	1	B
1	0	0	B

H(C)

$$H(C) = -(0.5)\log_2 0.5 - (0.5)\log_2 0.5$$
$$= 1 \text{ shannon}$$

X	Y	Z	Class
1	1	1	A
1	1	0	A
0	0	1	B
1	0	0	B

IG(C,X)

$$\mathrm{H(C|X)} = \frac{3}{4}[\frac{2}{3}\log_2\frac{3}{2} + \frac{1}{3}\log_2\frac{3}{1}] + \frac{1}{4}[0]$$

$$= 0.689 \text{ shannons}$$

$$\mathrm{IG(C,X)} = 1 - 0.689 = 0.311 \text{ shannons}$$

X	Y	Z	Class
1	1	1	A
1	1	0	A
0	0	1	B
1	0	0	B

IG(C,Y)

$$H(C|Y) = \frac{1}{2}[0] + \frac{1}{2}[0]$$

$$= 0 \text{ shannons}$$

$$\mathrm{IG}(C,Y) = 1 - 0 = 1 \text{ shannon}$$

X	Y	Z	Class
1	1	1	A
1	1	0	A
0	0	1	B
1	0	0	B

IG(C,Z)

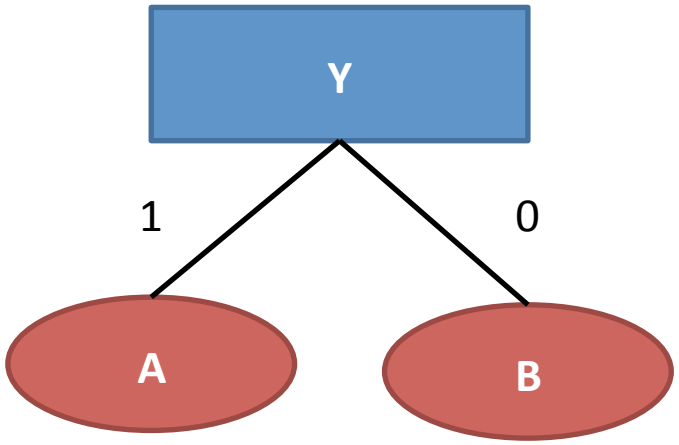
$$H(C|Y) = \frac{1}{2}[1] + \frac{1}{2}[1]$$

$$= 1 \text{ shannons}$$

$$\text{IG}(C, Z) = 1 - 1 = 0 \text{ shannons}$$

X	Y	Z	Class
1	1	1	A
1	1	0	A
0	0	1	B
1	0	0	B

Feature Split Choice



0.311	1.0	0.0	
X	**Y**	**Z**	**Class**
1	1	1	A
1	1	0	A
0	0	1	B
1	0	0	B

ID3: Algorithm Sketch

- If all examples “same”, return f(examples)
- If no more features, return f(examples)
- A = “best” feature
  - For each distinct value of A
    - branch = ID3(attributes - {A})

Example (MLiA)

No Surfacing	Flippers?	Fish?
Yes	Yes	Yes
Yes	Yes	Yes
Yes	No	No
No	Yes	No
No	Yes	No

0. Preliminaries

No Surfacing	Flippers?	Fish?
Yes	Yes	Yes
Yes	Yes	Yes
Yes	No	No
No	Yes	No
No	Yes	No

- Examples not the same class
- Features remain
- H(Fish?) = 0.971

1a: No Surfacing

No Surfacing	Flippers?	Fish?
Yes	Yes	Yes
Yes	Yes	Yes
Yes	No	No
No	Yes	No
No	Yes	No

- H(Fish? | No Surfacing) = 0.55
- IG(Fish?, No Surfacing) = 0.42

1b: Flippers?

No Surfacing	Flippers?	Fish?
Yes	Yes	Yes
Yes	Yes	Yes
Yes	No	No
No	Yes	No
No	Yes	No

- H(Fish? | Flippers?) = 0.8
- IG(Fish?, Flippers) = 0.17

2: Split on No Surfacing

Flippers?	Fish?
Yes	No
Yes	No

Flippers?	Fish?
Yes	Yes
Yes	Yes
No	No

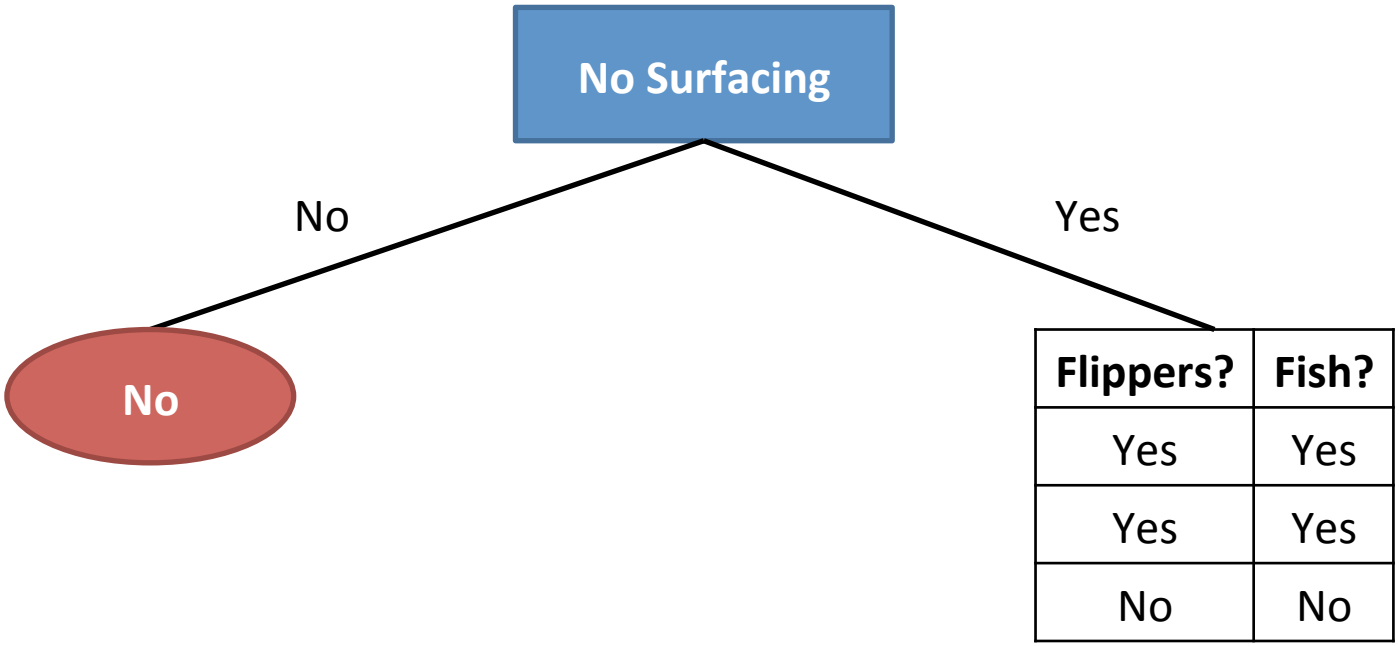
- Recurse(left)

2. Left

Flippers?	Fish?
Yes	No
Yes	No

- Examples the same class!
  - Return class leaf node

2: Split on No Surfacing



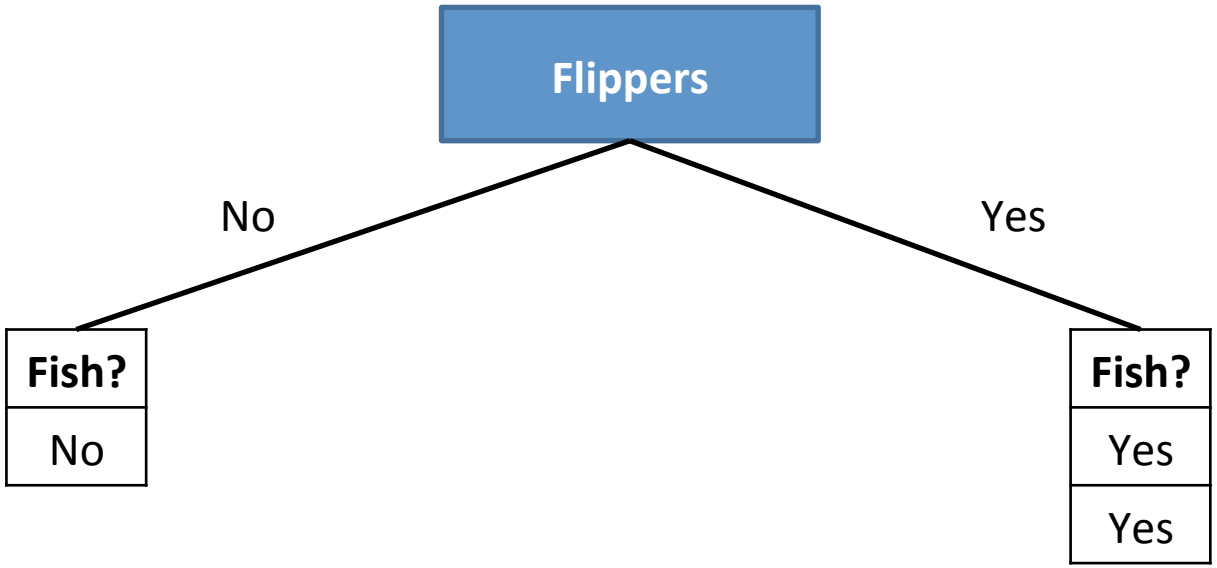
- Recurse(right)

2. Right

Flippers?	Fish?
Yes	Yes
Yes	Yes
No	No

- Examples not the same class
- One feature remaining
  - Split!

3. Split on Flippers



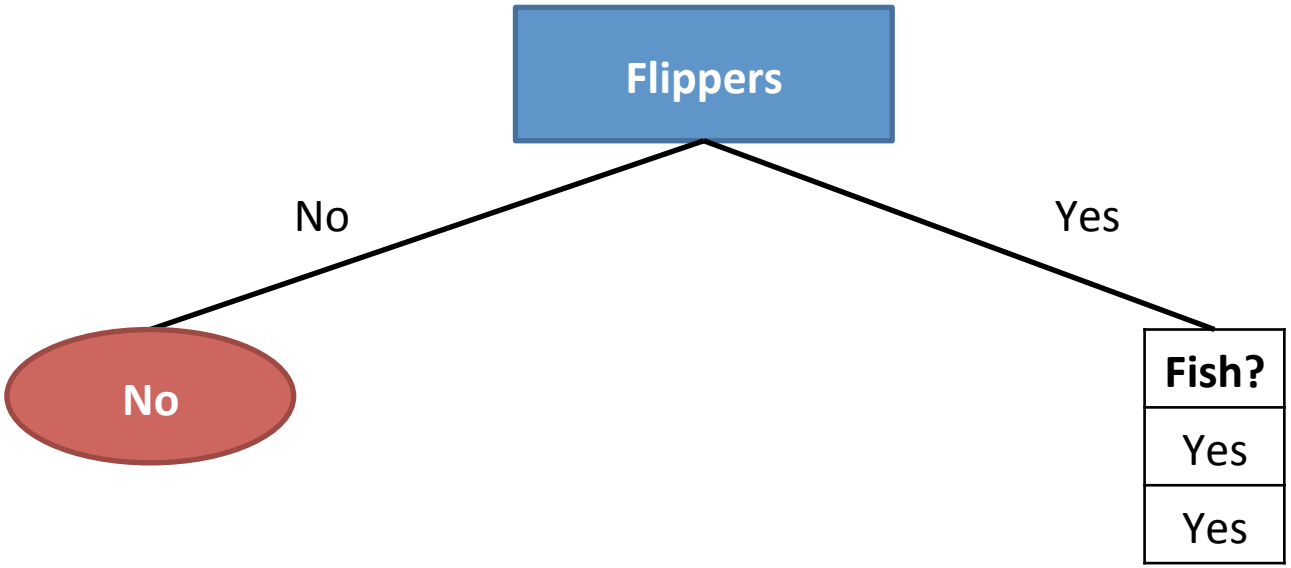
- Recurse(left)

3. Left

Fish?
No

- Examples the same class!
  - Return class leaf node

3. Split on Flippers



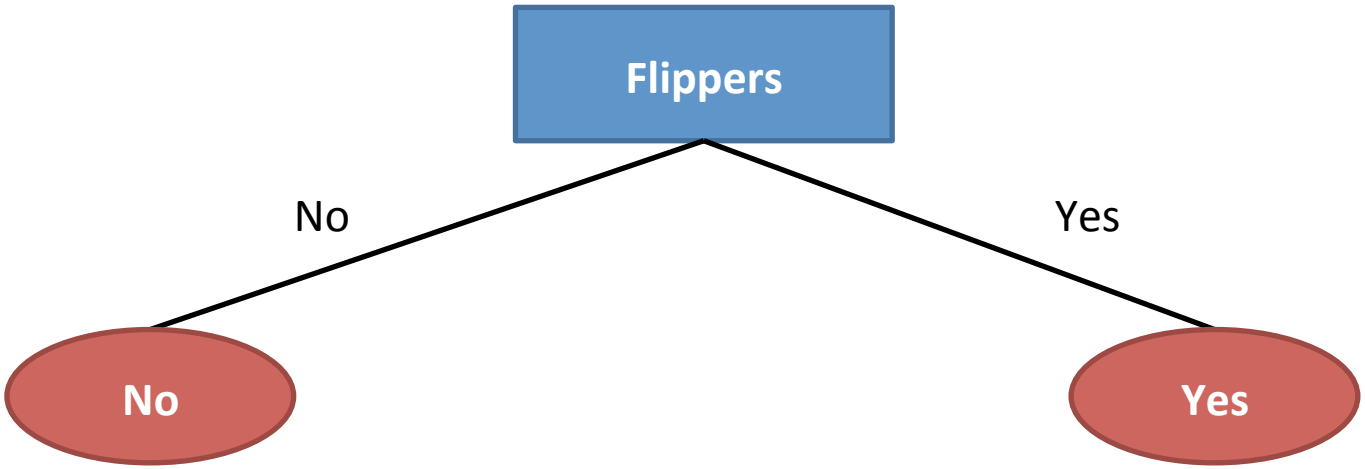
- Recurse(right)

3. Right

Fish?
Yes
Yes

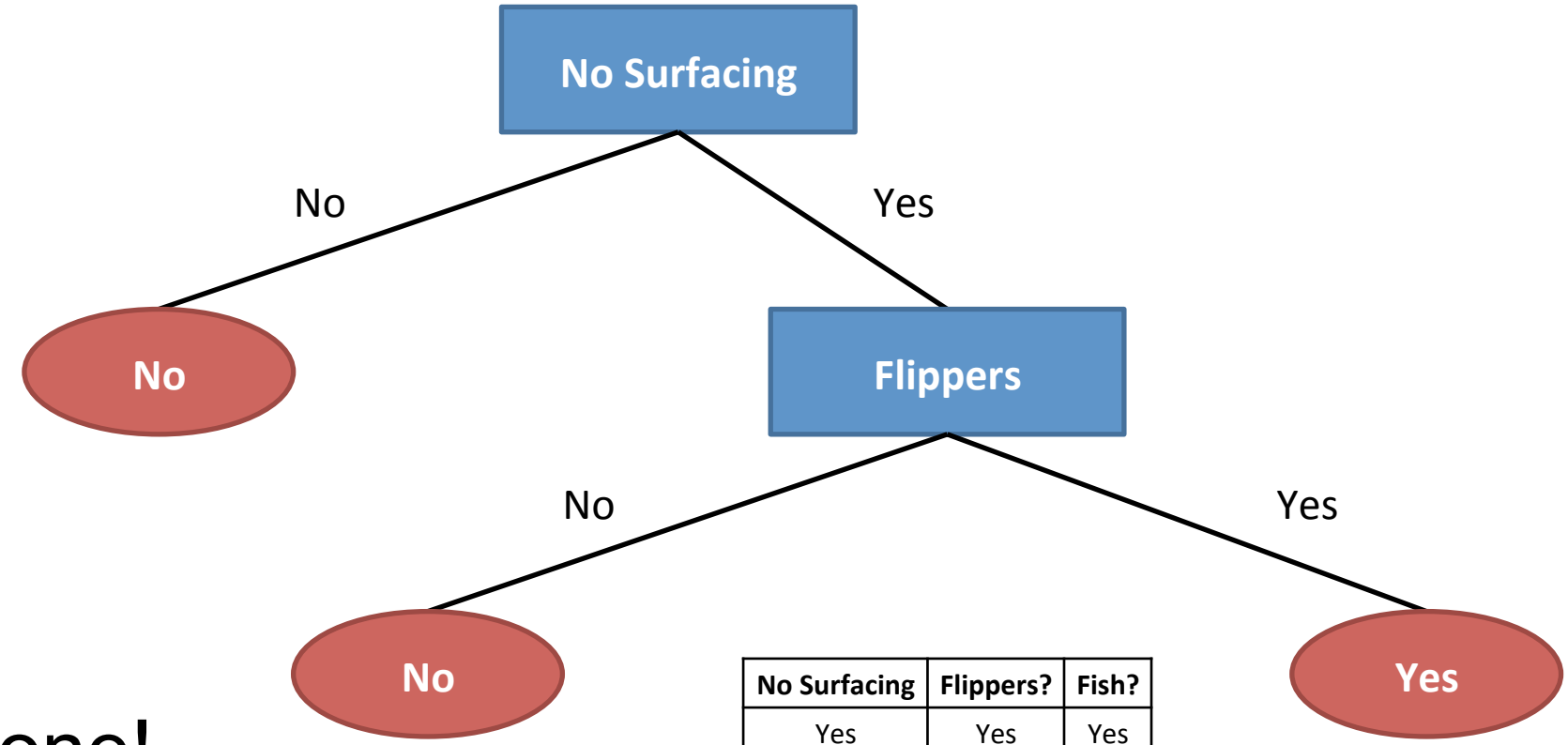
- Examples the same class!
  - Return class leaf node

3. Split on Flippers



- Return!

2: Split on No Surfacing



No Surfacing	Flippers?	Fish?
Yes	Yes	Yes
Yes	Yes	Yes
Yes	No	No
No	Yes	No
No	Yes	No

- Done!

Additional Base Case

- What to do given the following example input to ID3?
  - No additional features upon which to split

Fish?
Yes
Yes
No
No
No

- For categorical, majority vote

Extensions

- Generalization
- Continuous features
- Ensemble learning

Generalization

- Information gain biases towards features with many distinct values
  - Consider the value of CC/SSN

- Approaches to mediate
  - Gain ratio is a metric that divides each IG term by “SplitInfo”, which is large for features with many partitions (used in C4.5)
  - There are several pruning techniques that replace subtrees

Continuous Features

- You can always discretize/bin yourself
  - Run the risk of suboptimal depending on tree location
- Simple approach: binary splits, whereby left is ≤ threshold
- Consider each distinct value a threshold, calculate gain
  - Computationally expensive for large numbers of values
- C4.5 penalizes similar to large distinct sets

Ensemble Learning

- The **Random Forest** algorithm is an exemplar of using multiple trees
  - Each tree is trained via bootstrapped data (i.e. sampled with replacement)
  - Each choice node feature is selected from a random subset of overall
  - Decisions are bagged (i.e. aggregated over many trees)
  - Can use a validation set to weight via expected accuracy of each tree

Checkup

- ML task(s)?
  - Classification: binary/multi-class?
- Feature type(s)?
- Implicit/explicit?
- Parametric?
- Online?

Summary: ID3/Decision Trees

- Practicality
  - Easy, generally applicable
  - Need know nothing about the underlying process
  - Very popular, easy to understand
- Efficiency
  - Training: relatively fast, batch
  - Testing: typically very fast
- Performance
  - Possible to get stuck in suboptimal trees
    - Methods to help, hard in general