Decision trees
Structure and Function of Decision Trees
Motivation

Why decision trees?
- Simpler and more versatile than other types of prediction functions
- Training methods are fairly elementary
input age, gender, genre, release year, title
1: if age ≥ 40 then
2:   if genre = war or genre = western then
3:     return 4.3
4:   else
5:     return 1.5
6:   end if
7: else
8:   if release year > 1998 then
9:     return 3.5
10:  else
11:    return 2.0
12:  end if
13: end if
A decision tree over \( d \) input features \( x_1, \ldots, x_d \) is a rooted binary tree in which:

- Each tree (non-leaf) node has two child nodes (left child and right child) and is associated with a predicate involving some input features (e.g., \( x_3 < 25 \)).
- Each leaf node (no children) is associated with a return value (e.g., \( 3.5 \)).
A decision tree $T$ defines a prediction function $f_T$

- Domain: possible values of $(x_1, \ldots, x_d)$
- Range: possible return values of leaf nodes

Value of $f_T(x_1, \ldots, x_d)$:

- $T$ routes input $(x_1, \ldots, x_d)$ to a leaf node
- Output the associated return value

(age, gender, genre, release year, title) = (25, F, action, 2002, "Spider-man")
Pseudocode for evaluating a decision tree

input values of input features $x_1, \ldots, x_d$
output value of $f_T(x_1, \ldots, x_d)$

1: $v := \text{root}(T)$
2: loop
3: if $v \in \text{leaves}(T)$ then
4:     return $\text{return-value}_T(v)$
5:     \{ $T$ routes $(x_1, \ldots, x_d)$ to $v$ \}
6: else
7:     $P := \text{predicate}_T(v)$
8:     if $P(x_1, \ldots, x_d) = \text{true}$ then
9:         $v := \text{left-child}_T(v)$
10:    else
11:         $v := \text{right-child}_T(v)$
12:    end if
13: end if
14: end loop


age ≥ 40

genre = war or
genre = western

release year > 1998

4.3 1.5 3.5 2.0


5 / 25
Decision trees used in practice typically restrict the types of predicates at tree nodes.
Practical restrictions

Decision trees used in practice typically restrict the types of predicates at tree nodes

- Each predicate only involves a single input feature
Decision trees used in practice typically restrict the types of predicates at tree nodes

- Each predicate only involves a single input feature
- For ordered (numerical) features (which take numerical values with an intrinsic ordering):
  - Predicates are ordinal comparisons with a fixed value
  - Example: “$x_3 \leq -3.14$”
Practical restrictions

Decision trees used in practice typically restrict the types of predicates at tree nodes

▶ Each predicate only involves a single input feature

▶ For **ordered (numerical) features** (which take numerical values with an intrinsic ordering):
  
  ▶ Predicates are ordinal comparisons with a fixed value
  
  ▶ Example: “$x_3 \leq -3.14$”

▶ For **categorical features** (which take values from an unordered set):
  
  ▶ Predicates are tests of membership in a fixed set
  
  ▶ Example: “$x_8 \in \{A, C, G\}$”
Greedy Heuristic for Learning Decision Trees
Learning classification trees over numerical features

Focus on classification trees over numerical features

Training data $S$: labeled examples, each of the form $(\mathbf{x}, y)$ with $\mathbf{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$ and $y \in \{1, 2, \ldots, K\} := \{1, 2, \ldots, K\}$

Only consider predicates of the form "$x_j \leq c$" for some feature index $j \in \{1, 2, \ldots, d\}$ and threshold value $c \in \mathbb{R}$

Return values are in $\{1, 2, \ldots, K\}$

Objective function: number of classification mistakes $\text{mistakes}(T; S) := \text{number of examples } (\mathbf{x}, y) \text{ in } S \text{ such that } f_T(\mathbf{x}) \neq y$
Focus on **classification trees** over numerical features

- Training data $S$: labeled examples, each of the form

  $$(\vec{x}, y)$$

  with $\vec{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$ and $y \in [K] := \{1, 2, \ldots, K\}$
Learning classification trees over numerical features

Focus on classification trees over numerical features

- Training data $S$: labeled examples, each of the form

  $$(\vec{x}, y)$$

  with $\vec{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$ and $y \in [K] := \{1, 2, \ldots, K\}$

- Only consider predicates of the form

  \[ "x_j \leq c" \]

  for some feature index $j \in [d] := \{1, 2, \ldots, d\}$ and threshold value $c \in \mathbb{R}$
Learning classification trees over numerical features

Focus on classification trees over numerical features

- Training data $S$: labeled examples, each of the form
  \[(\vec{x}, y)\]
  with $\vec{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$ and $y \in [K] := \{1, 2, \ldots, K\}$

- Only consider predicates of the form
  \["x_j \leq c"\]
  for some feature index $j \in [d] := \{1, 2, \ldots, d\}$ and threshold value $c \in \mathbb{R}$

- Return values are in $[K]$
Learning classification trees over numerical features

Focus on classification trees over numerical features

- Training data $S$: labeled examples, each of the form

$$(\vec{x}, y)$$

with $\vec{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$ and $y \in [K] := \{1, 2, \ldots, K\}$

- Only consider predicates of the form

"$x_j \leq c$"

for some feature index $j \in [d] := \{1, 2, \ldots, d\}$ and threshold value $c \in \mathbb{R}$

- Return values are in $[K]$

- Objective function: number of classification mistakes

$$\text{mistakes}(T; S) := \text{number of examples } (\vec{x}, y) \text{ in } S \text{ such that } f_T(\vec{x}) \neq y$$
Running example: Classifying irises

- Three classes of irises: Setosa (1), Versicolor (2), Virginica (3)
- Two numerical features:
  - $x_1$: ratio of sepal width to sepal length
  - $x_2$: ratio of petal width to petal length
- 120 training data (40 from each class)
Greedy heuristic for learning decision trees

**Greedy training heuristic:**
- Start with a decision tree $T_1$
- Repeatedly modify current tree $T_i$ to get a new decision tree $T_{i+1}$
  - Modification chosen greedily to improve (or not worsen) an objective function
    
    \[
    \text{objective}(T_1) \geq \text{objective}(T_2) \geq \text{objective}(T_3) \geq \cdots
    \]
- Stop at some point

Questions:
1. How to pick the first tree $T_1$?
2. How to modify $T_i$ to get $T_{i+1}$?
3. When to stop?
Greedy heuristic for learning decision trees

**Greedy training heuristic:**
- Start with a decision tree $T_1$
- Repeatedly modify current tree $T_i$ to get a new decision tree $T_{i+1}$
  - Modification chosen greedily to improve (or not worsen) an objective function
    \[
    \text{objective}(T_1) \geq \text{objective}(T_2) \geq \text{objective}(T_3) \geq \cdots
    \]
- Stop at some point

**Questions:**
1. How to pick the first tree $T_1$?
2. How to modify $T_i$ to get $T_{i+1}$?
3. When to stop?
The Initial Tree
Initial tree $T_1$ has a single leaf node $r$ as its root

- Implements a constant function $f_{T_1}(\vec{x}) = \text{return-value}_{T_1}(r)$

Question:

What should $\text{return-value}_{T_1}(r)$ be to minimize $\text{mistakes}(T_1; S)$?

Answer:

A plurality label among examples in $S$; i.e., a label $y^* \in [K]$ that appears at least as often in $S$ as any other label.
Initial tree $T_1$ has a single leaf node $r$ as its root

- Implements a constant function $f_{T_1}(\bar{x}) = \text{return-value}_{T_1}(r)$

**Question:** What should $\text{return-value}_{T_1}(r)$ be to minimize $\text{mistakes}(T_1; S)$?
Initial tree $T_1$ has a single leaf node $r$ as its root

- Implements a constant function $f_{T_1}(\vec{x}) = \text{return-value}_{T_1}(r)$

**Question:** What should $\text{return-value}_{T_1}(r)$ be to minimize $\text{mistakes}(T_1; S)$?

**Answer:** A **plurality label** among examples in $S$; i.e., a label $y^* \in [K]$ that appears at least as often in $S$ as any other label.
Define the **uncertainty** of a collection of labeled examples $S'$ by

$$\text{uncertainty}(S') := \min_{y^* \in [K]} \left[ \text{fraction of examples } (\vec{x}, y) \in S' \text{ with } y \neq y^* \right]$$

That is, the fraction of examples not equal to a particular plurality label

(For simplicity, if $S'$ is empty, take $\text{uncertainty}(S') = 0$)
Define the **uncertainty** of a collection of labeled examples $S'$ by

$$\text{uncertainty}(S') := \min_{y^* \in [K]} \left[ \text{fraction of examples } (\vec{x}, y) \in S' \text{ with } y \neq y^* \right]$$

That is, the fraction of examples not equal to a particular plurality label

(For simplicity, if $S'$ is empty, take $\text{uncertainty}(S') = 0$)

**Fact:** If $\text{return-value}_{T_1}(r)$ is a plurality label in $S$, then

$$\text{mistakes}(T_1; S) = |S| \cdot \text{uncertainty}(S)$$
Define the **uncertainty** of a collection of labeled examples $S'$ by

$$\text{uncertainty}(S') := \min_{y^* \in [K]} \left[ \text{fraction of examples } (\vec{x}, y) \in S' \text{ with } y \neq y^* \right]$$

That is, the fraction of examples not equal to a particular plurality label

(For simplicity, if $S'$ is empty, take $\text{uncertainty}(S') = 0$)

**Fact:** If $\text{return-value}_{T_1}(r)$ is a plurality label in $S$, then

$$\text{mistakes}(T_1; S) = |S| \cdot \text{uncertainty}(S)$$

(Why uncertainty? Many variants of this greedy heuristic just change the definition of uncertainty.)
Running example: Plurality label of iris data

▶ All three labels appear with equal frequency, so each is a plurality label
▶ Initial tree $T_1$: leaf node $r$ with $y_r = 1$ (an arbitrary choice)
  ▶ $f_T(x) = 1$ for all $x$
Improving the Initial Tree
Greedy heuristic for learning decision trees

**Greedy training heuristic:**

- Start with a decision tree $T_1$
- Repeatedly modify current tree $T_i$ to get a new decision tree $T_{i+1}$
  - Modification chosen greedily to improve (or not worsen) an objective function
    
    $\text{objective}(T_1) \geq \text{objective}(T_2) \geq \text{objective}(T_3) \geq \cdots$

- Stop at some point

**Questions:**

1. How to pick the first tree $T_1$?
2. How to modify $T_i$ to get $T_{i+1}$?
3. When to stop?
To reduce the objective, we **split** the leaf node $r$:

- Replace it with a tree node $r$ whose children are two (new) leaf nodes, $u$ and $v$
- 3-node decision tree is called a **decision stump**

\[ T_1 \xrightarrow{y_r} T_2 \]
\[ x_j \leq c \]
\[ y_u \quad y_v \]
Improving the initial tree

To reduce the objective, we split the leaf node $r$:

- Replace it with a tree node $r$ whose children are two (new) leaf nodes, $u$ and $v$
- 3-node decision tree is called a decision stump

Questions:

1. How to choose the predicate $P_r = "x_j \leq c"$?
2. How to choose the return values $y_u$ and $y_v$?
Improving the initial tree

To reduce the objective, we **split** the leaf node $r$:

- Replace it with a tree node $r$ whose children are two (new) leaf nodes, $u$ and $v$
- 3-node decision tree is called a **decision stump**

**Questions:**

1. How to choose the predicate $P_r = "x_j \leq c"$?
2. How to choose the return values $y_u$ and $y_v$?

Observe that $\text{mistakes}(T_2; S) = \text{mistakes}(T_2; S_u) + \text{mistakes}(T_2; S_v)$, where

$S_u := \left[ \text{examples } (\vec{x}, y) \in S \text{ s.t. } P_r(\vec{x}) = \text{true} \right]$ and $S_v := \left[ \text{examples } (\vec{x}, y) \in S \text{ s.t. } P_r(\vec{x}) = \text{false} \right]$
Improving the initial tree

To reduce the objective, we split the leaf node $r$:

- Replace it with a tree node $r$ whose children are two (new) leaf nodes, $u$ and $v$
- 3-node decision tree is called a decision stump

Questions:

1. How to choose the predicate $P_r = "x_j \leq c"$?
2. How to choose the return values $y_u$ and $y_v$?

Observe that $\text{mistakes}(T_2; S) = \text{mistakes}(T_2; S_u) + \text{mistakes}(T_2; S_v)$, where

$$S_u := \{\text{examples } (\vec{x}, y) \in S \text{ s.t. } P_r(\vec{x}) = \text{true}\} \quad \text{and} \quad S_v := \{\text{examples } (\vec{x}, y) \in S \text{ s.t. } P_r(\vec{x}) = \text{false}\}$$

Whatever $P_r$ is, best to let $y_u$ and $y_v$ be plurality labels for $S_u$ and $S_v$, respectively; upon which

$$\text{mistakes}(T_2; S) = |S_u| \cdot \text{uncertainty}(S_u) + |S_v| \cdot \text{uncertainty}(S_v) \quad (\star)$$
To reduce the objective, we **split** the leaf node $r$:

- Replace it with a tree node $r$ whose children are two (new) leaf nodes, $u$ and $v$
- 3-node decision tree is called a **decision stump**

**Questions:**

1. How to choose the predicate $P_r = "x_j \leq c"$?
2. How to choose the return values $y_u$ and $y_v$?

Observe that $\text{mistakes}(T_2; S) = \text{mistakes}(T_2; S_u) + \text{mistakes}(T_2; S_v)$, where

$\quad S_u := \left[\text{examples } (\vec{x}, y) \in S \text{ s.t. } P_r(\vec{x}) = \text{true}\right]$ \quad and \quad $S_v := \left[\text{examples } (\vec{x}, y) \in S \text{ s.t. } P_r(\vec{x}) = \text{false}\right]$ \quad

Whatever $P_r$ is, best to let $y_u$ and $y_v$ be plurality labels for $S_u$ and $S_v$, respectively; upon which

$\text{mistakes}(T_2; S) = |S_u| \cdot \text{uncertainty}(S_u) + |S_v| \cdot \text{uncertainty}(S_v)$  \quad (⋆)

Remains to choose predicate $P_r$ so resulting “split” of $S$ into $(S_u, S_v)$ minimizes (⋆)
Choosing the best split

Enumerate all possible predicates of the form \( x_j \leq c \):

\[ T_1 \quad y_r \quad \rightarrow \quad T_2 \quad x_j \leq c \]

\[ S_u := \{ \text{examples } (\vec{x}, y) \in S \text{ s.t. } x_j \leq c \} \]

\[ S_v := \{ \text{examples } (\vec{x}, y) \in S \text{ s.t. } x_j > c \} \]

Choose the split that minimizes

\[ |S_u| \cdot \text{uncertainty}(S_u) + |S_v| \cdot \text{uncertainty}(S_v) \]
Choosing the best split

Enumerate all possible predicates of the form “\(x_j \leq c\)”: 

- Only \(d\) possible feature indices \(j \in [d]\)

\[S_u := \text{examples } (\vec{x}, y) \in S \text{ s.t. } x_j \leq c,\]
\[S_v := \text{examples } (\vec{x}, y) \in S \text{ s.t. } x_j > c\]

Choose the split that minimizes

\[|S_u| \cdot \text{uncertainty}(S_u) + |S_v| \cdot \text{uncertainty}(S_v)\]
Choosing the best split

Enumerate all possible predicates of the form “$x_j \leq c$”:

- Only $d$ possible feature indices $j \in [d]$
- Infinitely many possible thresholds $c \in \mathbb{R}$, but:
  - Only $\leq n + 1$ different “splits” of $S$ into $(S_u, S_v)$

$$
S_u := \{ \text{examples } (\vec{x}, y) \in S \text{ s.t. } x_j \leq c \}, \\
S_v := \{ \text{examples } (\vec{x}, y) \in S \text{ s.t. } x_j > c \}
$$

- (And only $\leq n - 1$ of them are non-trivial)
Choosing the best split

Enumerate all possible predicates of the form “\(x_j \leq c\)”: 

- Only \(d\) possible feature indices \(j \in [d]\)
- Infinitely many possible thresholds \(c \in \mathbb{R}\), but:
  - Only \(\leq n + 1\) different “splits” of \(S\) into \((S_u, S_v)\)
    
    \[
    \begin{align*}
    S_u & := \{ \text{examples } (\vec{x}, y) \in S \text{ s.t. } x_j \leq c \}, \\
    S_v & := \{ \text{examples } (\vec{x}, y) \in S \text{ s.t. } x_j > c \}
    \end{align*}
    \]

- (And only \(\leq n - 1\) of them are non-trivial)
- Choose the split that minimizes

\[
|S_u| \cdot \text{uncertainty}(S_u) + |S_v| \cdot \text{uncertainty}(S_v)
\]
Running example: How to improve the initial tree?

```
<table>
<thead>
<tr>
<th>Ratio of Petal Width to Petal Length</th>
<th>Setosa (1)</th>
<th>Versicolor (2)</th>
<th>Virginica (3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.05</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.15</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.25</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.35</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.45</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.55</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.65</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.75</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.8</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>Ratio of Sepal Width to Sepal Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3 0.35 0.4 0.45 0.5 0.55 0.6 0.65 0.7 0.75 0.8</td>
</tr>
</tbody>
</table>
```

```
T_1 x \leq ?
```

```
T_2
```

```
1
```

```
? ?
```

```
x?
```
Running example: Choosing the best split

Decision stump with \( x_1 \leq 0.5484 \) has 41 mistakes

All decision stumps with feature \( x_2 \) have \( \geq 43 \) mistakes

\( T_1 \)

\( x_? \leq ? \)

\( T_2 \)
Running example: Best decision stump

![Decision stump diagram]

- **Setosa (1)**
- **Versicolor (2)**
- **Virginica (3)**

The decision stump is defined by the inequality:

\[ x_1 \leq 0.5484 \]
In general, for any tree $T_i$, if return values are set optimally,

$$\text{mistakes}(T_i; S) = \sum_{\ell \in \text{leaves}(T_i)} |S_{\ell}| \cdot \text{uncertainty}(S_{\ell})$$

where $S_{\ell}$ are training examples that are “routed” to leaf $\ell$. 
In general, for any tree $T_i$, if return values are set optimally,

$$\text{mistakes}(T_i; S) = \sum_{\ell \in \text{leaves}(T_i)} |S_\ell| \cdot \text{uncertainty}(S_\ell)$$

where $S_\ell$ are training examples that are “routed” to leaf $\ell$

For next tree $T_{i+1}$, enumerate all leaf nodes $\ell \in \text{leaves}(T_i)$, in addition to all possible splits of $\ell$
**General case**

In general, for any tree $T_i$, if return values are set optimally,

$$\text{mistakes}(T_i; S) = \sum_{\ell \in \text{leaves}(T_i)} |S_\ell| \cdot \text{uncertainty}(S_\ell)$$

where $S_\ell$ are training examples that are “routed” to leaf $\ell$

For next tree $T_{i+1}$, enumerate all leaf nodes $\ell \in \text{leaves}(T_i)$, in addition to all possible splits of $\ell$

- Choose leaf node $\ell$ and predicate “$x_j \leq c$” with highest reduction in objective value:

$$|S_\ell| \cdot \text{uncertainty}(S_\ell) - (|S_u| \cdot \text{uncertainty}(S_u) + |S_v| \cdot \text{uncertainty}(S_v))$$

where $S_u := \{ \text{examples } (\vec{x}, y) \in S_\ell \text{ s.t. } x_j \leq c \}$ and $S_v := \{ \text{examples } (\vec{x}, y) \in S_\ell \text{ s.t. } x_j > c \}$
General case

In general, for any tree $T_i$, if return values are set optimally,

$$\text{mistakes}(T_i; S) = \sum_{\ell \in \text{leaves}(T_i)} |S_\ell| \cdot \text{uncertainty}(S_\ell)$$

where $S_\ell$ are training examples that are “routed” to leaf $\ell$

For next tree $T_{i+1}$, enumerate all leaf nodes $\ell \in \text{leaves}(T_i)$, in addition to all possible splits of $\ell$

- Choose leaf node $\ell$ and predicate “$x_j \leq c$” with highest reduction in objective value:

$$|S_\ell| \cdot \text{uncertainty}(S_\ell) - (|S_u| \cdot \text{uncertainty}(S_u) + |S_v| \cdot \text{uncertainty}(S_v))$$

where $S_u := \{\text{examples } (\vec{x}, y) \in S_\ell \text{ s.t. } x_j \leq c\}$ and $S_v := \{\text{examples } (\vec{x}, y) \in S_\ell \text{ s.t. } x_j > c\}$

- Replace chosen leaf node with decision stump based on chosen predicate
Running example: Further improvement?

Current objective value: 41

\[|S_{\text{left}}| \cdot \text{uncertainty}(S_{\text{left}}) = 80 \cdot \frac{1}{2} = 40\]

\[|S_{\text{right}}| \cdot \text{uncertainty}(S_{\text{right}}) = 40 \cdot \frac{1}{40} = 1\]
Running example: Further improvement?

Reduction in objective from splitting right leaf: 1

\[ 40 \cdot \frac{1}{40} - (1 \cdot 0 + 39 \cdot 0) = 1 \]
Running example: Further improvement?

Reduction in objective from splitting left leaf: 24

\[ 80 \cdot \frac{1}{2} - (49 \cdot \frac{13}{49} + 31 \cdot \frac{3}{31}) = 24 \]
Stopping Criteria
Greedy heuristic for learning decision trees

**Greedy training heuristic:**
- Start with a decision tree $T_1$
- Repeatedly modify current tree $T_i$ to get a new decision tree $T_{i+1}$
  - Modification chosen greedily to improve (or not worsen) an objective function
    \[
    \text{objective}(T_1) \geq \text{objective}(T_2) \geq \text{objective}(T_3) \geq \cdots
    \]
- Stop at some point

**Questions:**
1. How to pick the first tree $T_1$?
2. How to modify $T_i$ to get $T_{i+1}$?
3. When to stop?
Many possible stopping criteria seem reasonable, including:

1. Stop when no split leads to reduction in objective value
   - Can be overly myopic

2. Stop when size of tree (i.e., number of leaf nodes) reaches predetermined "maximum size"
   - Motivated by desire for small trees
     - If "maximum size" is too small, tree may be under-fit (fail to capture relevant patterns needed for accurate prediction)

3. Stop when each leaf node is "pure" (i.e., all training examples routed to leaf node have same label or same feature vector)
   - Tends to lead to large trees
     - Tree may be over-fit to training data (adapt to idiosyncratic patterns in training data that are unhelpful)

Each has some potential drawbacks
Many possible stopping criteria seem reasonable, including:

1. Stop when no split leads to reduction in objective value
Many possible stopping criteria seem reasonable, including:

1. Stop when no split leads to reduction in objective value

2. Stop when size of tree (i.e., number of leaf nodes) reaches predetermined “maximum size”
Many possible stopping criteria seem reasonable, including:

1. Stop when no split leads to reduction in objective value

2. Stop when size of tree (i.e., number of leaf nodes) reaches predetermined “maximum size”

3. Stop when each leaf node is “pure” (i.e., all training examples routed to leaf node have same label or same feature vector)
Many possible stopping criteria seem reasonable, including:

1. Stop when no split leads to reduction in objective value

2. Stop when size of tree (i.e., number of leaf nodes) reaches predetermined “maximum size”

3. Stop when each leaf node is “pure” (i.e., all training examples routed to leaf node have same label or same feature vector)

Each has some potential drawbacks
Many possible stopping criteria seem reasonable, including:

1. Stop when no split leads to reduction in objective value
   ▶ Can be overly myopic

2. Stop when size of tree (i.e., number of leaf nodes) reaches predetermined “maximum size”

3. Stop when each leaf node is “pure” (i.e., all training examples routed to leaf node have same label or same feature vector)

Each has some potential drawbacks
Stopping criteria

Many possible stopping criteria seem reasonable, including:

1. Stop when no split leads to reduction in objective value
   ▶ Can be overly myopic

2. Stop when size of tree (i.e., number of leaf nodes) reaches predetermined “maximum size”
   ▶ Motivated by desire for small trees
   ▶ If “maximum size” is too small, tree may be **under-fit**
     (Fail to capture relevant patterns needed for accurate prediction)

3. Stop when each leaf node is “pure” (i.e., all training examples routed to leaf node have same label or same feature vector)

Each has some potential drawbacks
Many possible stopping criteria seem reasonable, including:

1. Stop when no split leads to reduction in objective value
   ▶ Can be overly myopic

2. Stop when size of tree (i.e., number of leaf nodes) reaches predetermined “maximum size”
   ▶ Motivated by desire for small trees
   ▶ If “maximum size” is too small, tree may be **under-fit**
     (Fail to capture relevant patterns needed for accurate prediction)

3. Stop when each leaf node is “pure” (i.e., all training examples routed to leaf node have same label or same feature vector)
   ▶ Tends to lead to large trees
   ▶ Tree may be **over-fit** to training data
     (Adapt to idiosyncratic patterns in training data that are unhelpful)

Each has some potential drawbacks
Stop when no split leads to reduction in objective value
Stop when no split leads to reduction in objective value

Some training data from $\mathbb{R}^2 \times \{\bigcirc, \times\}$:

<table>
<thead>
<tr>
<th>feature vector</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0, 0)</td>
<td>$\bigcirc$</td>
</tr>
<tr>
<td>(0, 1)</td>
<td>$\times$</td>
</tr>
<tr>
<td>(1, 0)</td>
<td>$\times$</td>
</tr>
<tr>
<td>(1, 1)</td>
<td>$\bigcirc$</td>
</tr>
</tbody>
</table>
**Myopia**

*Stop when no split leads to reduction in objective value*

Some training data from $\mathbb{R}^2 \times \{\circ, \times\}$:

<table>
<thead>
<tr>
<th>feature vector</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0, 0)</td>
<td>$\circ$</td>
</tr>
<tr>
<td>(0, 1)</td>
<td>$\times$</td>
</tr>
<tr>
<td>(1, 0)</td>
<td>$\times$</td>
</tr>
<tr>
<td>(1, 1)</td>
<td>$\circ$</td>
</tr>
</tbody>
</table>

- Single leaf node tree commits 2 mistakes

$\begin{align*}
x_2 & \\
\times & \quad \circ & \\
\circ & \quad \times & \\
x_1 & \\
\end{align*}$
Myopia

Stop when no split leads to reduction in objective value

Some training data from $\mathbb{R}^2 \times \{\circ, \times\}$:

<table>
<thead>
<tr>
<th>feature vector</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0, 0)</td>
<td>$\circ$</td>
</tr>
<tr>
<td>(0, 1)</td>
<td>$\times$</td>
</tr>
<tr>
<td>(1, 0)</td>
<td>$\times$</td>
</tr>
<tr>
<td>(1, 1)</td>
<td>$\circ$</td>
</tr>
</tbody>
</table>

- Single leaf node tree commits 2 mistakes
- Each decision stump commits 2 mistakes
  (Hence, no reduction in objective value!)
**Myopia**

*Stop when no split leads to reduction in objective value*

Some training data from $\mathbb{R}^2 \times \{\bigcirc, \times\}$:

<table>
<thead>
<tr>
<th>feature vector</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0, 0)</td>
<td>$\bigcirc$</td>
</tr>
<tr>
<td>(0, 1)</td>
<td>$\times$</td>
</tr>
<tr>
<td>(1, 0)</td>
<td>$\times$</td>
</tr>
<tr>
<td>(1, 1)</td>
<td>$\bigcirc$</td>
</tr>
</tbody>
</table>

- Single leaf node tree commits 2 mistakes
- Each decision stump commits 2 mistakes (Hence, no reduction in objective value!)
- The following tree commits 0 mistakes:
Motivation to seek “succinct” predictors (e.g., small trees):

1. Understandability/interpretability by humans
2. Occam’s razor
   ▶ “The simplest explanation is usually the right one” (as paraphrased on Wikipedia)

Caveats of this strategy include:

▶ Greedy training heuristic is not guaranteed to optimize objective within size bound
▶ Even if it did, need to choose the desired size
▶ Human aesthetics of “simplicity” need not align with what makes for a better predictor
Motivation to seek “succinct” predictors (e.g., small trees):

1. Understandability/interpretability by humans
Motivation to seek “succinct” predictors (e.g., small trees):

1. Understandability/interpretability by humans
2. Occam’s razor
   - “The simplest explanation is usually the right one”
     (as paraphrased on Wikipedia)
Motivation to seek “succinct” predictors (e.g., small trees):

1. Understandability/interpretability by humans
2. Occam’s razor
   - “The simplest explanation is usually the right one”
     (as paraphrased on Wikipedia)

Caveats of this strategy include:

William of Occam (c. 1287–1347)
Motivation to seek “succinct” predictors (e.g., small trees):

1. Understandability/interpretability by humans
2. Occam’s razor
   - “The simplest explanation is usually the right one” (as paraphrased on Wikipedia)

Caveats of this strategy include:
- Greedy training heuristic is not guaranteed to optimize objective within size bound
Succinctness

Motivation to seek “succinct” predictors (e.g., small trees):

1. Understandability/interpretability by humans
2. **Occam’s razor**
   - “The simplest explanation is usually the right one” (as paraphrased on Wikipedia)

Caveats of this strategy include:

- Greedy training heuristic is not guaranteed to optimize objective within size bound
- Even if it did, need to choose the desired size

William of Occam (c. 1287–1347)
Motivation to seek “succinct” predictors (e.g., small trees):

1. Understandability/interpretability by humans

2. Occam’s razor
   - “The simplest explanation is usually the right one”
     (as paraphrased on Wikipedia)

Caveats of this strategy include:

- Greedy training heuristic is not guaranteed to optimize objective within size bound
- Even if it did, need to choose the desired size
- Human aesthetics of “simplicity” need not align with what makes for a better predictor
Recap

- Decision trees are simple and versatile types of predictors
- Greedy training heuristic (and variants) repeatedly modifies a tree to improve an objective function
- Different stopping criteria may lead to trees of different qualities