Overfitting in decision trees
What happens when we increase depth?

Training error reduces with depth

<table>
<thead>
<tr>
<th>Tree depth</th>
<th>depth = 1</th>
<th>depth = 2</th>
<th>depth = 3</th>
<th>depth = 5</th>
<th>depth = 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training error</td>
<td>0.22</td>
<td>0.13</td>
<td>0.10</td>
<td>0.03</td>
<td>0.00</td>
</tr>
<tr>
<td>Decision boundary</td>
<td><img src="image1" alt="Decision boundary at depth 1" /></td>
<td><img src="image2" alt="Decision boundary at depth 2" /></td>
<td><img src="image3" alt="Decision boundary at depth 3" /></td>
<td><img src="image4" alt="Decision boundary at depth 5" /></td>
<td><img src="image5" alt="Decision boundary at depth 10" /></td>
</tr>
</tbody>
</table>
Two approaches to picking simpler trees

1. **Early Stopping:**
   Stop the learning algorithm *before* tree becomes too complex

2. **Pruning:**
   Simplify the tree *after* the learning algorithm terminates
Technique 1: Early stopping

• **Stopping conditions (recap):**
  1. All examples have the same target value
  2. No more features to split on

• **Early stopping conditions:**
  1. Limit tree depth (choose `max_depth` using validation set)
  2. Do not consider splits that do not cause a sufficient decrease in classification error
  3. Do not split an intermediate node which contains too few data points
Challenge with early stopping condition 1

Hard to know exactly when to stop

Also, might want some branches of tree to go deeper while others remain shallow
Early stopping condition 2: Pros and Cons

• **Pros:**
  – A reasonable heuristic for early stopping to avoid useless splits

• **Cons:**
  – *Too short sighted:* We may miss out on “good” splits may occur right after “useless” splits
  – Saw this with “xor” example
Two approaches to picking simpler trees

1. **Early Stopping:**
   Stop the learning algorithm *before* tree becomes too complex

2. **Pruning:**
   Simplify the tree *after* the learning algorithm terminates

Complements early stopping
Pruning: *Intuition*
Train a complex tree, simplify later
Pruning motivation

- Classification Error
- True Error
- Training Error
- Tree depth

Simple tree
Complex tree

Simplify after tree is built
Don’t stop too early

©2021 Carlos Guestrin
Scoring trees: Desired total quality format

Want to balance:

i. How well tree fits data
ii. Complexity of tree

Total cost = measure of fit + measure of complexity
Simple measure of complexity of tree

\[ L(T) = \# \text{ of leaf nodes} \]
Balance simplicity & predictive power

Too complex, risk of overfitting

Start

excellent

Credit?

poor

fair

Term?

Safe

Risky

3 years

5 years

Income?

high

low

Term?

Safe

Risky

3 years

5 years

Too simple, high classification error

Start

Risky

©2021 Carlos Guestrin
Balancing fit and complexity

Total cost $C(T) = \text{Error}(T) + \lambda \ L(T)$

If $\lambda = 0$:

If $\lambda = \infty$:

If $\lambda$ in between:
Tree pruning algorithm
Step 1: Consider a split

Tree $T$

- Start
  - Credit?
    - fair
      - Term?
        - 3 years
          - Risky
        - 5 years
          - Safe
    - poor
      - Income?
        - high
          - Term?
            - 3 years
              - Risky
            - 5 years
              - Safe
        - low
          - Term?
            - 5 years
              - Risky

Candidate for pruning
Step 2: Compute total cost $C(T)$ of split

Tree $T$

Start

excellent

Credit?

fair

poor

Term?

3 years

5 years

Risky

Safe

Income?

high

low

Term?

3 years

5 years

Risky

Safe

Risky

Candidate for pruning

Tree

Error

#Leaves

Total

<table>
<thead>
<tr>
<th>Tree</th>
<th>Error</th>
<th>#Leaves</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T$</td>
<td>0.25</td>
<td>6</td>
<td>0.43</td>
</tr>
</tbody>
</table>

$C(T) = \text{Error}(T) + \lambda \text{L}(T)$

$\lambda = 0.3$
Step 2: “Undo” the splits on $T_{smaller}$

$$C(T) = \text{Error}(T) + \lambda \text{L}(T)$$

<table>
<thead>
<tr>
<th>Tree</th>
<th>Error</th>
<th>#Leaves</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T$</td>
<td>0.25</td>
<td>6</td>
<td>0.43</td>
</tr>
<tr>
<td>$T_{smaller}$</td>
<td>0.26</td>
<td>5</td>
<td>0.41</td>
</tr>
</tbody>
</table>

$\lambda = 0.3$
Prune if total cost is lower: $C(T_{\text{smaller}}) \leq C(T)$

Tree $T_{\text{smaller}}$

Worse training error but lower overall cost

$\lambda = 0.3$

<table>
<thead>
<tr>
<th>Tree</th>
<th>Error</th>
<th>#Leaves</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T$</td>
<td>0.25</td>
<td>6</td>
<td>0.43</td>
</tr>
<tr>
<td>$T_{\text{smaller}}$</td>
<td>0.26</td>
<td>5</td>
<td>0.41</td>
</tr>
</tbody>
</table>

$C(T) = \text{Error}(T) + \lambda \cdot L(T)$

Replace split by leaf node? YES!
Step 5: Repeat Steps 1-4 for every split

Decide if each split can be “pruned”
Summary of overfitting in decision trees
What you can do now…

• Identify when overfitting in decision trees
• Prevent overfitting with early stopping
  – Limit tree depth
  – Do not consider splits that do not reduce classification error
  – Do not split intermediate nodes with only few points
• Prevent overfitting by pruning complex trees
  – Use a total cost formula that balances classification error and tree complexity
  – Use total cost to merge potentially complex trees into simpler ones