Decision Tree Classification

Guest Lecturer: Joshua Ervin
Example: Predicting potential loan defaults

- Data: discrete for now (e.g. credit rating: excellent, fair, poor)
- Goal: Given a new loan application, predict whether or not the applicant will default on their loan:

<table>
<thead>
<tr>
<th>Credit</th>
<th>Term</th>
<th>Income</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>excellent</td>
<td>3 years</td>
<td>high</td>
<td>safe</td>
</tr>
<tr>
<td>fair</td>
<td>5 years</td>
<td>low</td>
<td>risky</td>
</tr>
<tr>
<td>fair</td>
<td>3 years</td>
<td>high</td>
<td>risky</td>
</tr>
<tr>
<td>poor</td>
<td>3 years</td>
<td>high</td>
<td>risky</td>
</tr>
</tbody>
</table>
Decision Tree

- **Internal Node**: A node that tests a feature
- **Branch**: Splits input data based on the value of a feature
- **Leaf**: Assigns a class to data (i.e. **SAFE**, **RISKY**)
Decision Stumps

Choice 1: Split on Credit

START
22:18

CREDIT

Excellent
SAFE 90

Fair
SAFE 9:4

Poor
RISKY 4:14

Choice 2: Split on Term Length

START
22:18

TERM

3 Years
SAFE 16:4

5 Years
RISKY 6:14

How do we decide which split to make?
○ Always pick the split which maximizes accuracy

accuracy = \frac{\text{#correct predictions}}{\text{#data points}}
Greedy Algorithm for Growing a Decision Tree

- Start with a single root node
- Repeat while the stopping rule is not met
  - Choose a feature \( x[i] \) to split that maximizes classification accuracy
- Stopping Rule:
  - 1) Do not branch if all data has the same label (pure)
  - 2) We have already split on that feature before
Greedy Algorithm

START 22:18

CREDIT

Excellent
SAFE 9:0

Fair
SAFE 9:4

Poor
RISKY 4:14

Classification Accuracy: 80%
Why is it ok to stop here?

Repeat the same decision tree stump-building process with this subset of data.
Greedy Algorithm

Classification Accuracy:
Greedy Algorithm

START 22:18

CREDIT

Excellent

SAFE 9:0

9:4

TERM

3 Years
3 Years

TERM

RISKY 0:4

SAFE 9:0

High

TERM

5 Years

TERM

RISKY 0:2

SAFE 4:3

INCOME

Poor

High

4:14

Low

Classification Accuracy: 92.5%
Early Stopping Rules

- Stopping Rules:
  1) Do not branch if all data has the same label (pure)
  2) We have already split on that feature before
  3*) If adding a branch does not increase accuracy, should we still branch?

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>-1</td>
<td>+1</td>
<td>+1</td>
</tr>
<tr>
<td>+1</td>
<td>-1</td>
<td>+1</td>
</tr>
<tr>
<td>+1</td>
<td>+1</td>
<td>-1</td>
</tr>
</tbody>
</table>
### XOR: Root

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>-1</td>
<td>+1</td>
<td>+1</td>
</tr>
<tr>
<td>+1</td>
<td>-1</td>
<td>+1</td>
</tr>
<tr>
<td>+1</td>
<td>+1</td>
<td>-1</td>
</tr>
</tbody>
</table>

### # Levels | Accuracy
--- | ---
0   | 50%  
1   | ?    
2   | ?    

**KEY**

-1:1

+1:2:2
XOR: 1 Split

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>-1</td>
<td>+1</td>
<td>+1</td>
</tr>
<tr>
<td>+1</td>
<td>-1</td>
<td>+1</td>
</tr>
<tr>
<td>+1</td>
<td>+1</td>
<td>-1</td>
</tr>
</tbody>
</table>

# Levels | Accuracy
--- | ----
0     | 50%  
1     | 50%  
2     | ?    

KEY
+1: -1

START
2:2

x[1]

+1
1:1

-1
1:1
XOR: 2 Splits

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>-1</td>
<td>+1</td>
<td>+1</td>
</tr>
<tr>
<td>+1</td>
<td>-1</td>
<td>+1</td>
</tr>
<tr>
<td>+1</td>
<td>+1</td>
<td>-1</td>
</tr>
</tbody>
</table>

# Levels | Accuracy
---|---
0 | 50%
1 | 50%
2 | 100%

KEY
+1: -1

START
2:2
Real Valued Data

- We’ve been making an assumption so far that our data takes on discrete values.
- How do we know here to split our data? There are an infinite number of possible splits we can make.

<table>
<thead>
<tr>
<th>Credit</th>
<th>Term</th>
<th>Income</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>excellent</td>
<td>3 years</td>
<td>$105,000.00</td>
<td>safe</td>
</tr>
<tr>
<td>fair</td>
<td>5 years</td>
<td>$63,000.00</td>
<td>risky</td>
</tr>
<tr>
<td>fair</td>
<td>3 years</td>
<td>$85,000.00</td>
<td>risky</td>
</tr>
<tr>
<td>poor</td>
<td>3 years</td>
<td>$99,000.00</td>
<td>risky</td>
</tr>
</tbody>
</table>
Real Valued Data

How do we know where to split our data?
Key Insight: Sort the data and split halfway between each pair of adjacent points. There will always be a finite number of splits. How many splits are there?
Which split is best? Pick the one that maximizes accuracy.
Real Valued Data

Age

Income

10k

20k

30k
Real Valued Data

**Graphical Representation:**
- A scatter plot with axes labeled 'Age' and 'Income'.
- Age categories: <= 21 vs. > 21.
- Income categories: 10k, 20k, 30k.
- Age <= 21 includes ages 10, 20, and 30 with corresponding income levels 10k, 20k, 30k.
- Age > 21

**Decision Tree:**
- **START:** 8:8
  - Age <= 21
    - RISKY: 3:4
  - Age > 21
    - SAFE: 4:5

Legend:
- Red markers represent RISKY
- Green markers represent SAFE
Real Valued Data

- Age: 10, 20, 30
- Income: 10k, 20k, 30k

Age <= 21: SAFE 3:0
Age > 21: RISKY 0:4

Income <= 20k: SAFE 3:0
Income > 20k: RISKY 0:4
Real Valued Data

- **Age:**
  - age <= 21
  - age > 21

- **Income:**
  - <=19k
  - >19k
  - <=20k
  - >20k

- **Decision Tree:**
  - START: 8:8
  - age <= 21: 3:4
  - age > 21: 5:4
  - income <=19k: SAFE 3:0
  - income >19k: RISKY 0:4
  - income <=20k: RISKY 0:4
  - income >20k: SAFE 5:0
Probabilistic Prediction

\[ P(y = -1) = \frac{4}{6} \]

\[ P(y = +1) = \frac{4}{5} \]

\[ P(y = +1) = \frac{4}{4} \]

\[ P(y = -1) = \frac{4}{4} \]
Overfitting

- Similar to regression, training error monotonically non-increases with model complexity.
- Model complexity with decision trees is commonly measured in the depth of the tree.
- Two methods for preventing overfitting:
  - 1) Early stopping
    - Stop the tree before it can get too complex
  - 2) Pruning
    - Create a complex tree and make it more simple
Overfitting
Overfitting: Early Stopping

- **Stopping Rules:**
  - 1) All data in the subset have the same label
  - 2) No more features left to split

- **Early Stopping Rule**
  - Only grow up to a max depth hyperparameter (choose via validation)
    - Can be difficult to know the depth.
    - Oftentimes the correct tree is one that is imbalanced
  - Don’t split if there is not a sufficient decrease in error
    - Problem: difficult to classify XOR problems
Exercise: Overfitting and cross validation

cross-validation(data d, folds k):
    fold_1, fold_k = split_data(d, k)
    for each model m:
        for i from 1 to k:
            model = train_model(m, fold_i)
            err = error(model, fold_i)
        avg_err = average err over folds
        keep track of m with smallest avg_err
    return m with smallest avg_err

<table>
<thead>
<tr>
<th>Max Height</th>
<th>Fold-1 Error</th>
<th>Fold-2 Error</th>
<th>Fold-3 Error</th>
<th>Test Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>10.3</td>
<td>14.2</td>
<td>12.5</td>
<td>14.5</td>
</tr>
<tr>
<td>10</td>
<td>5.6</td>
<td>4.3</td>
<td>7.3</td>
<td>8.7</td>
</tr>
<tr>
<td>15</td>
<td>3.1</td>
<td>10.4</td>
<td>8.8</td>
<td>6.9</td>
</tr>
</tbody>
</table>
Overfitting: Pruning

- Basic Idea: Train a tall, overfit model and then simplify it.
- Pruning is defined by a quality metric that balances classification error and model complexity.

\[ \text{Loss}(T) = \text{Error}(T) + \lambda r(T) \]

# Ions in model
Pruning Algorithm

1. Consider some arbitrary split
2. Compute the error if the split is taken away
3. Compute the penalty of keeping the split
4. Pick whichever one minimizes loss
5. Repeat 1-4 for all splits

\[ \text{Total}(t) = \text{Error}(t) + \lambda \times \text{# leaves}(t) \]

<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>4</td>
<td>0.43</td>
</tr>
<tr>
<td>T'</td>
<td>0.26</td>
<td>3</td>
<td>0.41</td>
</tr>
</tbody>
</table>

\( \lambda = 0.03 \)
Decision Trees for Regression

- Error measured by mean squared error
- Prediction is the mean value of all partitions in the sample