Decision Trees
Predicting potential loan defaults
What makes a loan risky?

I want to buy a new house!

Credit History ★★★★★
Income ★★★
Term ★★★★★★
Personal Info ★★★
Credit history explained

Did I pay previous loans on time?

Example: excellent, good, or fair
Income

What’s my income?

Example: $80K per year
Loan terms

How soon do I need to pay the loan?

**Example:** 3 years, 5 years,...
Personal information

Age, reason for the loan, marital status, ...

**Example:** Home loan for a married couple
Classifier review

Input: $x_i$

Output: $\hat{y}$
Predicted class

$\hat{y}_i = +1$
Safe

$\hat{y}_i = -1$
Risky
This module ... decision trees
Scoring a loan application

\[ x_i = (\text{Credit} = \text{poor}, \text{Income} = \text{high}, \text{Term} = 5 \text{ years}) \]
Decision tree learning task
Decision tree learning problem

Training data: \( N \) observations \((x_i, y_i)\)

<table>
<thead>
<tr>
<th>Credit</th>
<th>Term</th>
<th>Income</th>
<th>( y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>excellent</td>
<td>3 yrs</td>
<td>high</td>
<td>safe</td>
</tr>
<tr>
<td>fair</td>
<td>5 yrs</td>
<td>low</td>
<td>risky</td>
</tr>
<tr>
<td>fair</td>
<td>3 yrs</td>
<td>high</td>
<td>safe</td>
</tr>
<tr>
<td>poor</td>
<td>5 yrs</td>
<td>high</td>
<td>risky</td>
</tr>
<tr>
<td>excellent</td>
<td>3 yrs</td>
<td>low</td>
<td>risky</td>
</tr>
<tr>
<td>fair</td>
<td>5 yrs</td>
<td>low</td>
<td>safe</td>
</tr>
<tr>
<td>poor</td>
<td>3 yrs</td>
<td>high</td>
<td>risky</td>
</tr>
<tr>
<td>poor</td>
<td>5 yrs</td>
<td>low</td>
<td>safe</td>
</tr>
<tr>
<td>fair</td>
<td>3 yrs</td>
<td>high</td>
<td>safe</td>
</tr>
</tbody>
</table>
Quality metric: Classification error

• Error measures fraction of mistakes

\[
\text{Error} = \frac{\# \text{incorrect predictions}}{\# \text{examples}}
\]

– Best possible value : 0.0
– Worst possible value: 1.0
How do we find the best tree?

Exponentially large number of possible trees makes decision tree learning hard!

Learning the smallest decision tree is an NP-hard problem [Hyafil & Rivest ’76]
Greedy decision tree learning
Our training data table

Assume $N = 40$, 3 features

<table>
<thead>
<tr>
<th>Credit</th>
<th>Term</th>
<th>Income</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>excellent</td>
<td>3 yrs</td>
<td>high</td>
<td>safe</td>
</tr>
<tr>
<td>fair</td>
<td>5 yrs</td>
<td>low</td>
<td>risky</td>
</tr>
<tr>
<td>fair</td>
<td>3 yrs</td>
<td>high</td>
<td>safe</td>
</tr>
<tr>
<td>poor</td>
<td>5 yrs</td>
<td>high</td>
<td>risky</td>
</tr>
<tr>
<td>excellent</td>
<td>3 yrs</td>
<td>low</td>
<td>risky</td>
</tr>
<tr>
<td>fair</td>
<td>5 yrs</td>
<td>low</td>
<td>safe</td>
</tr>
<tr>
<td>poor</td>
<td>3 yrs</td>
<td>high</td>
<td>risky</td>
</tr>
<tr>
<td>poor</td>
<td>5 yrs</td>
<td>low</td>
<td>safe</td>
</tr>
<tr>
<td>fair</td>
<td>3 yrs</td>
<td>high</td>
<td>safe</td>
</tr>
</tbody>
</table>
Start with all the data

Loan status:  Safe  Risky

N = 40 examples

# of Safe loans

# of Risky loans
Compact visual notation: Root node

Loan status:  Safe  Risky

N = 40 examples

# of Safe loans

Root
22  18

# of Risky loans
Decision stump: Single level tree

Loan status:
Safe Risky

Root
22 18

Split on Credit

Credit?

excellent
9 0
Subset of data with Credit = excellent

fair
9 4
Subset of data with Credit = fair

poor
4 14
Subset of data with Credit = poor
Visual notation: Intermediate nodes

Loan status:
Safe Risky

Root

Credit?

excellent

fair

poor

Intermediate nodes

©2021 Carlos Guestrin

CS229: Machine Learning
Making predictions with a decision stump

Loan status:
Safe  Risky

For each intermediate node, set $\hat{y} =$ majority value
Selecting best feature to split on
How do we learn a decision stump?

Loan status:
Safe Risky

Root
22 18

Find the “best” feature to split on!

Credit?

excellent
9 0

fair
9 4

poor
4 14
How do we select the best feature?

**Choice 1: Split on Credit**

- Loan status: Safe  Risky
- Root 22 18
- Credit?
  - excellent 9 0
  - fair 9 4
  - poor 4 14

**Choice 2: Split on Term**

- Loan status: Safe  Risky
- Root 22 18
- Term?
  - 3 years 16 4
  - 5 years 6 14
How do we measure effectiveness of a split?

**Loan status:**
- Safe
- Risky

**Root: 22 18**

**Credit?**
- **excellent:** 9 0
- **fair:** 9 4
- **poor:** 4 14

**Idea:** Calculate classification error of this decision stump

**Error:** \[ \frac{\text{# mistakes}}{\text{# data points}} \]
Calculating classification error

- **Step 1:** $\hat{y} =$ class of majority of data in node
- **Step 2:** Calculate classification error of predicting $\hat{y}$ for this data

\[
\text{Error} = \frac{\text{number of mistakes}}{\text{total number of samples}} = \frac{18}{22 + 18} = \frac{18}{40} = 0.45
\]
Choice 1: Split on \textbf{Credit} history?

\textbf{Choice 1: Split on Credit}

\textbf{Loan status: Safe Risky}

\textbf{Root: 22 18}

\textbf{Does a split on Credit reduce classification error below 0.45?}

- \textbf{Credit?}
  - \textbf{excellent: 9 0}
  - \textbf{fair: 9 4}
  - \textbf{poor: 4 14}
Split on **Credit**: Classification error

**Choice 1: Split on Credit**

- **Loan status:**
  - Safe
  - Risky

- **Root:**
  - 22
  - 18

- **Credit?**
  - Excellent: 9 0
    - Safe: 0 mistakes
  - Fair: 9 4
    - Safe: 4 mistakes
  - Poor: 4 14
    - Risky: 4 mistakes

<table>
<thead>
<tr>
<th>Tree</th>
<th>Classification error</th>
</tr>
</thead>
<tbody>
<tr>
<td>(root)</td>
<td>0.45</td>
</tr>
<tr>
<td>Split on credit</td>
<td>0.2</td>
</tr>
</tbody>
</table>
Choice 2: Split on Term

Loan status: Safe Risky

Choice 2: Split on Term

Root
22 18

Term?

3 years 16 4
Safe

5 years 6 14
Risky
Evaluating the split on Term

Choice 2: Split on Term

Loan status:
Safe  Risky

Root
22  18

Term?

3 years
16  4

4 mistakes
Safe

5 years
6  14

6 mistakes
Risky

Error =

<table>
<thead>
<tr>
<th>Tree</th>
<th>Classification error</th>
</tr>
</thead>
<tbody>
<tr>
<td>(root)</td>
<td>0.45</td>
</tr>
<tr>
<td>Split on credit</td>
<td>0.2</td>
</tr>
<tr>
<td>Split on term</td>
<td>0.25</td>
</tr>
</tbody>
</table>
Choice 1 vs Choice 2: Comparing split on Credit vs Term

<table>
<thead>
<tr>
<th>Tree</th>
<th>Classification error</th>
</tr>
</thead>
<tbody>
<tr>
<td>(root)</td>
<td>0.45</td>
</tr>
<tr>
<td>split on credit</td>
<td>0.2</td>
</tr>
<tr>
<td>split on loan term</td>
<td>0.25</td>
</tr>
</tbody>
</table>

### Choice 1: Split on Credit
- Loan status: Safe Risky
  - Root: 22 18
  - Credit?
    - excellent: 9 0
    - poor: 4 14

### Choice 2: Split on Term
- Loan status: Safe Risky
  - Root: 22 18
  - Term?
    - 3 years: 16 4
    - 5 years: 6 14

Winner: Choice 2: Split on Term
Feature split selection algorithm

• Given a subset of data $M$ (a node in a tree)

• For each feature $h_i(x)$:
  1. Split data of $M$ according to feature $h_i(x)$
  2. Compute classification error of split

• Chose feature $h^*(x)$ with lowest classification error
Recursion & Stopping conditions
We’ve learned a decision stump, what next?

Loan status:
Safe Risky

Root
22 18

Credit?

excellent
9 0

fair
9 4

poor
4 14

Safe

Leaf node

All data points are Safe ➔
nothing else to do with this subset of data
Tree learning = Recursive stump learning

Loan status: Safe Risky

Root

Credit?

excellent

fair

poor

Build decision stump with subset of data where
Credit = fair

Build decision stump with subset of data where
Credit = poor

Safe

22 18

9 0

9 4

4 14
Second level

Loan status:
Safe Risky

Credit?

excellent 9 0

fair 9 4

poor 4 14

Term?

3 years 0 4

5 years 9 0

Income?

high 4 5

Low 0 9

Build another stump these data points
Final decision tree

Loan status:
- Safe
- Risky

Root
- 22 18

Credit?
- excellent: Safe
- Fair
  - 9 4
    - Term?
      - 3 years
        - 0 4
          - Risky
      - 5 years
        - 9 0
          - Safe
    - 9 14
      - Risky

Income?
- poor
  - 4 14
    - Risky
    - 9 0
      - Safe
- high
  - 4 5
    - 3 years
      - 0 2
      - Risky
    - 5 years
      - 4 3
      - Safe
- low
  - 0 9
    - Safe

©2021 Carlos Guestrin
CS299: Machine Learning
Simple greedy decision tree learning

1. Pick best feature to split on
2. Learn decision stump with this split
3. For each leaf of decision stump, recurse

When do we stop???
Stopping condition 1: All data agrees on y

All data in these nodes have same y value ➔ Nothing to do

Credit?
- excellent 9 0
  - Safe
- Fair 9 4
  - Term?
    - 3 years 0 4
      - Risky
    - 5 years 9 0
      - Safe
- Risky

Income?
- poor 4 14
  - low 0 9
    - Term?
      - 3 years 0 2
        - Risky
      - 5 years 4 3
        - Safe
Stopping condition 2: Already split on all features

Already split on all possible features → Nothing to do

Root

Credit?

excellent

Fair

Term?

3 years

5 years

Income?

poor

high

low

Term?

3 years

5 years

Term?

3 years

5 years

©2021 Carlos Guestrin
Greedy decision tree learning

• **Step 1:** Start with an empty tree

• **Step 2:** Select a feature to split data

• For each split of the tree:
  • **Step 3:** If nothing more to do, make predictions
  • **Step 4:** Otherwise, go to Step 2 & continue (recurse) on this split

Pick feature split leading to lowest classification error

Stopping conditions 1 & 2

Recursion
Is this a good idea?

**Proposed stopping condition 3:**
Stop if no split reduces the classification error
Stopping condition 3: 
Don’t stop if error doesn’t decrease???

\[ y = x[1] \text{ xor } x[2] \]

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>False</td>
</tr>
</tbody>
</table>

y values
True False

Root
2 2

\[
\text{Error} = _______.
\]

<table>
<thead>
<tr>
<th>Tree</th>
<th>Classification error</th>
</tr>
</thead>
<tbody>
<tr>
<td>(root)</td>
<td>0.5</td>
</tr>
</tbody>
</table>
Consider split on $x[1]$

\[ y = x[1] \text{ xor } x[2] \]

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>False</td>
</tr>
</tbody>
</table>

**y values**
- True
- False

**Root**
- 2
- 2

**x[1]**

**True**
- 1
- 1

**False**
- 1
- 1

**Error** = ______.

<table>
<thead>
<tr>
<th>Tree</th>
<th>Classification error</th>
</tr>
</thead>
<tbody>
<tr>
<td>(root)</td>
<td>0.5</td>
</tr>
<tr>
<td>Split on $x[1]$</td>
<td>0.5</td>
</tr>
</tbody>
</table>

\[ y = x[1] \text{ xor } x[2] \]
Consider split on \( x[2] \)

\[
y = x[1] \text{ xor } x[2]
\]

<table>
<thead>
<tr>
<th>( x[1] )</th>
<th>( x[2] )</th>
<th>( y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>False</td>
</tr>
</tbody>
</table>

**y values**

- True
- False

**Root**

\[
\begin{align*}
\text{Error} &= \frac{1+1}{2+2} \\
&= 0.5
\end{align*}
\]

Neither features improve training error... Stop now???
Final tree with stopping condition 3

$$y = x[1] \text{ xor } x[2]$$

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>False</td>
</tr>
</tbody>
</table>

y values
- True
- False

Tree | Classification error
---|---
with stopping condition 3 | 0.5

©2021 Carlos Guestrin
CS229: Machine Learning
Without stopping condition 3

Condition 3 (stopping when training error doesn’t improve) is not recommended!

\[ y = x[1] \text{ xor } x[2] \]

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>False</td>
</tr>
</tbody>
</table>

Tree | Classification error
---|-------------------
with stopping condition 3 | 0.5
without stopping condition 3 |
Decision tree learning: 
*Real valued features*
How do we use real values inputs?

<table>
<thead>
<tr>
<th>Income</th>
<th>Credit</th>
<th>Term</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>$105 K</td>
<td>excellent</td>
<td>3 yrs</td>
<td>Safe</td>
</tr>
<tr>
<td>$112 K</td>
<td>good</td>
<td>5 yrs</td>
<td>Risky</td>
</tr>
<tr>
<td>$73 K</td>
<td>fair</td>
<td>3 yrs</td>
<td>Safe</td>
</tr>
<tr>
<td>$69 K</td>
<td>excellent</td>
<td>5 yrs</td>
<td>Safe</td>
</tr>
<tr>
<td>$217 K</td>
<td>excellent</td>
<td>3 yrs</td>
<td>Risky</td>
</tr>
<tr>
<td>$120 K</td>
<td>good</td>
<td>5 yrs</td>
<td>Safe</td>
</tr>
<tr>
<td>$64 K</td>
<td>fair</td>
<td>3 yrs</td>
<td>Risky</td>
</tr>
<tr>
<td>$340 K</td>
<td>excellent</td>
<td>5 yrs</td>
<td>Safe</td>
</tr>
<tr>
<td>$60 K</td>
<td>good</td>
<td>3 yrs</td>
<td>Risky</td>
</tr>
</tbody>
</table>
Threshold split

Loan status:
Safe  Risky

Root
22  18

Split on the feature Income

Income?

< $60K
8  13

>= $60K
14  5

Subset of data with Income >= $60K
Finding the best threshold split

Infinite possible values of $t$

Income $< t^*$

Income $\geq t^*$

Income $= t^*$

$10K$

$120K$

Safe

Risky
Consider a threshold between points

Same _classification error_ for any threshold split between $v_A$ and $v_B$
Only need to consider mid-points

Finite number of splits to consider

Income
$10K

Safe
Risky

$120K
Threshold split selection algorithm

- **Step 1:** Sort the values of a feature $h_j(x)$:
  
  Let $\{v_1, v_2, v_3, \ldots v_N\}$ denote sorted values

- **Step 2:**
  - For $i = 1 \ldots N-1$
    
    - Consider split $t_i = (v_i + v_{i+1}) / 2$
    
    - Compute classification error for threshold split $h_j(x) \geq t_i$
  - Chose the $t^*$ with the lowest classification error
Visualizing the threshold split

Threshold split is the line $Age = 38$
Split on Age $\geq$ 38
Depth 2: Split on Income $\geq \$60K$

Threshold split is the line $\text{Income} = 60K$
Each split partitions the 2-D space
Decision trees vs logistic regression: 

\textit{Example}
## Logistic regression

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
<th>Weight Learned</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h_0(x)$</td>
<td>1</td>
<td>0.22</td>
</tr>
<tr>
<td>$h_1(x)$</td>
<td>$x[1]$</td>
<td>1.12</td>
</tr>
<tr>
<td>$h_2(x)$</td>
<td>$x[2]$</td>
<td>-1.07</td>
</tr>
</tbody>
</table>

The table above shows the features and their respective values along with the learned weights. The diagrams illustrate the decision boundary formed by the logistic regression model based on the learned weights.
Depth 1: Split on $x[1]$
Depth 2

![Decision Tree Diagram](image)

- **y values**
  - +
  - -

- **Root**
  - 18 13

- **x[1]**
  - x[1] < -0.07
    - 13 3
  - x[1] ≥ -0.07
    - 4 11

- **x[2]**
  - x[2] < -1.66
    - 7 0
  - x[2] ≥ -1.66
    - 6 3
  - x[2] < 1.55
    - 1 11
  - x[2] ≥ 1.55
    - 3 0
Threshold split caveat

For threshold splits, same feature can be used multiple times.
Decision boundaries
Comparing decision boundaries

### Decision Tree
- **Depth 1**
- **Depth 3**
- **Depth 10**

### Logistic Regression
- **Degree 1 features**
- **Degree 2 features**
- **Degree 6 features**
Summary of decision trees
What you can do now

• Define a decision tree classifier
• Interpret the output of a decision trees
• Learn a decision tree classifier using greedy algorithm
• Traverse a decision tree to make predictions
  – Majority class predictions
• Tackle continuous and discrete features