CSC 411 Lecture 3: Decision Trees

Roger Grosse, Amir-massoud Farahmand, and Juan Carrasquilla

University of Toronto
Decision Trees

- Simple but powerful learning algorithm
- One of the most widely used learning algorithms in Kaggle competitions

- Lets us introduce ensembles (Lectures 4–5), a key idea in ML more broadly
- Useful information theoretic concepts (entropy, mutual information, etc.)
Decision Trees

- width > 6.5cm?
  - Yes
  - height > 9.5cm?
    - Yes
      - Yes
    - No
      - No
  - No
    - height > 6.0cm?
      - Yes
        - Yes
      - No
        - No
Decision trees make predictions by recursively splitting on different attributes according to a tree structure.
Example with Discrete Inputs

What if the attributes are discrete?

<table>
<thead>
<tr>
<th>Example</th>
<th>Input Attributes</th>
<th>Goal</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>Yes, No, Yes, Some, $$, No, Yes, French, 0–10</td>
<td>WillWait</td>
</tr>
<tr>
<td>x2</td>
<td>Yes, No, Yes, Full, $, No, No, Thai, 30–60</td>
<td>y1 = Yes</td>
</tr>
<tr>
<td>x3</td>
<td>No, Yes, No, No, Some, $, No, No, Burger, 0–10</td>
<td>y2 = No</td>
</tr>
<tr>
<td>x4</td>
<td>Yes, No, Yes, Yes, Full, $, Yes, No, Thai, 10–30</td>
<td>y3 = Yes</td>
</tr>
<tr>
<td>x5</td>
<td>Yes, No, Yes, No, Full, $$, No, Yes, French, &gt;60</td>
<td>y4 = Yes</td>
</tr>
<tr>
<td>x6</td>
<td>No, Yes, No, Yes, Some, $$, Yes, Yes, Italian, 0–10</td>
<td>y5 = No</td>
</tr>
<tr>
<td>x7</td>
<td>No, Yes, No, No, None, $, Yes, No, Burger, 0–10</td>
<td>y6 = Yes</td>
</tr>
<tr>
<td>x8</td>
<td>No, No, No, Yes, Some, $$, Yes, Yes, Thai, 0–10</td>
<td>y7 = No</td>
</tr>
<tr>
<td>x9</td>
<td>No, Yes, Yes, No, Full, $, Yes, No, Burger, &gt;60</td>
<td>y8 = Yes</td>
</tr>
<tr>
<td>x10</td>
<td>Yes, Yes, Yes, Yes, Full, $$, No, Yes, Italian, 10–30</td>
<td>y9 = No</td>
</tr>
<tr>
<td>x11</td>
<td>No, No, No, No, None, $, No, No, Thai, 0–10</td>
<td>y10 = No</td>
</tr>
<tr>
<td>x12</td>
<td>Yes, Yes, Yes, Yes, Full, $, No, No, Burger, 30–60</td>
<td>y11 = No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>y12 = Yes</td>
</tr>
</tbody>
</table>

Attributes:

1. Alternate: whether there is a suitable alternative restaurant nearby.
2. Bar: whether the restaurant has a comfortable bar area to wait in.
3. Fri/Sat: true on Fridays and Saturdays.
4. Hungry: whether we are hungry.
5. Patrons: how many people are in the restaurant (values are None, Some, and Full).
6. Price: the restaurant’s price range ($, $\$, $$$).
7. Raining: whether it is raining outside.
8. Reservation: whether we made a reservation.
9. Type: the kind of restaurant (French, Italian, Thai or Burger).
10. WaitEstimate: the wait estimated by the host (0-10 minutes, 10-30, 30-60, >60).
Decision Tree: Example with Discrete Inputs

- The tree to decide whether to wait (T) or not (F)
Internal nodes test attributes

Branching is determined by attribute value

Leaf nodes are outputs (predictions)
Each path from root to a leaf defines a region $R_m$ of input space

Let $\{(x^{(m_1)}, t^{(m_1)}), \ldots, (x^{(m_k)}, t^{(m_k)})\}$ be the training examples that fall into $R_m$

**Classification tree:**
- discrete output
- leaf value $y^m$ typically set to the most common value in $\{t^{(m_1)}, \ldots, t^{(m_k)}\}$

**Regression tree:**
- continuous output
- leaf value $y^m$ typically set to the mean value in $\{t^{(m_1)}, \ldots, t^{(m_k)}\}$

Note: We will focus on classification

[Slide credit: S. Russell]
Expressiveness

- **Discrete-input, discrete-output case:**
  - Decision trees can express any function of the input attributes
  - E.g., for Boolean functions, truth table row $\rightarrow$ path to leaf:

```
A   B   A xor B
F   F   F
F   T   T
T   F   T
T   T   F
```

- **Continuous-input, continuous-output case:**
  - Can approximate any function arbitrarily closely
  - Trivially, there is a consistent decision tree for any training set w/ one path to leaf for each example (unless $f$ nondeterministic in $x$) but it probably won’t generalize to new examples

[Slide credit: S. Russell]
How do we Learn a Decision Tree?

- How do we construct a useful decision tree?
Learning Decision Trees

Learning the simplest (smallest) decision tree is an NP complete problem [if you are interested, check: Hyafil & Rivest’76]

- Resort to a **greedy heuristic**:
  - Start from an empty decision tree
  - Split on the “best” attribute
  - Recurse

- Which attribute is the “best”?
  - Choose based on accuracy?
Choosing a Good Split

- Why isn’t accuracy a good measure?

Is this split good? Zero accuracy gain.

Instead, we will use techniques from information theory

**Idea:** Use counts at leaves to define probability distributions, so we can measure uncertainty
Choosing a Good Split

- Which attribute is better to split on, $X_1$ or $X_2$?
  - Deterministic: good (all are true or false; just one class in the leaf)
  - Uniform distribution: bad (all classes in leaf equally probable)
  - What about distributions in between?

Note: Let’s take a slight detour and remember concepts from information theory

[Slide credit: D. Sontag]
We Flip Two Different Coins

Sequence 1:
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 ... ?

Sequence 2:
0 1 0 1 0 1 1 1 0 1 0 0 1 1 0 1 0 1 ... ?

versus

16

0 1 2

8 10

0 1
Quantifying Uncertainty

**Entropy** is a measure of expected “surprise”:

\[
H(X) = - \sum_{x \in X} p(x) \log_2 p(x)
\]

Measures the information content of each observation

- Unit = **bits**
- A fair coin flip has 1 bit of entropy
Quantifying Uncertainty

\[ H(X) = - \sum_{x \in X} p(x) \log_2 p(x) \]
**Entropy**

- **“High Entropy”**:  
  - Variable has a uniform like distribution  
  - Flat histogram  
  - Values sampled from it are less predictable

- **“Low Entropy”**  
  - Distribution of variable has many peaks and valleys  
  - Histogram has many lows and highs  
  - Values sampled from it are more predictable

[Slide credit: Vibhav Gogate]
Entropy of a Joint Distribution

- Example: \( X = \{ \text{Raining, Not raining} \} \), \( Y = \{ \text{Cloudy, Not cloudy} \} \)

<table>
<thead>
<tr>
<th></th>
<th>Cloudy</th>
<th>Not Cloudy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raining</td>
<td>24/100</td>
<td>1/100</td>
</tr>
<tr>
<td>Not Raining</td>
<td>25/100</td>
<td>50/100</td>
</tr>
</tbody>
</table>

\[
H(X, Y) = - \sum_{x \in X} \sum_{y \in Y} p(x, y) \log_2 p(x, y)
\]

\[
= - \frac{24}{100} \log_2 \frac{24}{100} - \frac{1}{100} \log_2 \frac{1}{100} - \frac{25}{100} \log_2 \frac{25}{100} - \frac{50}{100} \log_2 \frac{50}{100}
\]

\[
\approx 1.56 \text{bits}
\]
Specific Conditional Entropy

- Example: \( X = \{ \text{Raining}, \text{Not raining} \} \), \( Y = \{ \text{Cloudy}, \text{Not cloudy} \} \)

<table>
<thead>
<tr>
<th></th>
<th>Cloudy</th>
<th>Not Cloudy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raining</td>
<td>24/100</td>
<td>1/100</td>
</tr>
<tr>
<td>Not Raining</td>
<td>25/100</td>
<td>50/100</td>
</tr>
</tbody>
</table>

- What is the entropy of cloudiness \( Y \), \textit{given that it is raining}?

\[
H(Y|X = x) = - \sum_{y \in Y} p(y|x) \log_2 p(y|x)
\]

\[
= - \frac{24}{25} \log_2 \frac{24}{25} - \frac{1}{25} \log_2 \frac{1}{25}
\]

\[
\approx 0.24 \text{bits}
\]

- We used: \( p(y|x) = \frac{p(x,y)}{p(x)} \), and \( p(x) = \sum_{y} p(x,y) \) (sum in a row)
### Conditional Entropy

<table>
<thead>
<tr>
<th></th>
<th>Cloudy</th>
<th>Not Cloudy</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Raining</strong></td>
<td>24/100</td>
<td>1/100</td>
</tr>
<tr>
<td><strong>Not Raining</strong></td>
<td>25/100</td>
<td>50/100</td>
</tr>
</tbody>
</table>

- The expected conditional entropy:

$$H(Y|X) = \sum_{x \in X} p(x) H(Y|X = x)$$

$$= -\sum_{x \in X} \sum_{y \in Y} p(x, y) \log_2 p(y|x)$$
Conditional Entropy

Example: $X = \{\text{Raining, Not raining}\}$, $Y = \{\text{Cloudy, Not cloudy}\}$

<table>
<thead>
<tr>
<th></th>
<th>Cloudy</th>
<th>Not Cloudy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raining</td>
<td>24/100</td>
<td>1/100</td>
</tr>
<tr>
<td>Not Raining</td>
<td>25/100</td>
<td>50/100</td>
</tr>
</tbody>
</table>

What is the entropy of cloudiness, given the knowledge of whether or not it is raining?

$$H(Y|X) = \sum_{x \in X} p(x)H(Y|X = x)$$

$$= \frac{1}{4}H(\text{cloudy|is raining}) + \frac{3}{4}H(\text{cloudy|not raining})$$

$$\approx 0.75 \text{ bits}$$
Conditional Entropy

Some useful properties:

- $H$ is always non-negative
- Chain rule: $H(X, Y) = H(X|Y) + H(Y) = H(Y|X) + H(X)$
- If $X$ and $Y$ independent, then $X$ doesn’t tell us anything about $Y$: $H(Y|X) = H(Y)$
- But $Y$ tells us everything about $Y$: $H(Y|Y) = 0$
- By knowing $X$, we can only decrease uncertainty about $Y$: $H(Y|X) \leq H(Y)$
Information Gain

<table>
<thead>
<tr>
<th></th>
<th>Cloudy</th>
<th>Not Cloudy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raining</td>
<td>24/100</td>
<td>1/100</td>
</tr>
<tr>
<td>Not Raining</td>
<td>25/100</td>
<td>50/100</td>
</tr>
</tbody>
</table>

How much information about cloudiness do we get by discovering whether it is raining?

\[ IG(Y|X) = H(Y) - H(Y|X) \approx 0.25 \text{ bits} \]

This is called the **information gain** in \( Y \) due to \( X \), or the **mutual information** of \( Y \) and \( X \).

- If \( X \) is completely uninformative about \( Y \): \( IG(Y|X) = 0 \)
- If \( X \) is completely informative about \( Y \): \( IG(Y|X) = H(Y) \)
Information gain measures the informativeness of a variable, which is exactly what we desire in a decision tree attribute!

What is the information gain of this split?

Root entropy: \( H(Y) = -\frac{49}{149} \log_2 \left( \frac{49}{149} \right) - \frac{100}{149} \log_2 \left( \frac{100}{149} \right) \approx 0.91 \)

Leafs entropy: \( H(Y|\text{left}) = 0, \ H(Y|\text{right}) \approx 1. \)

\( IG(split) \approx 0.91 - \left( \frac{1}{3} \cdot 0 + \frac{2}{3} \cdot 1 \right) \approx 0.24 > 0 \)
At each level, one must choose:

1. Which variable to split.
2. Possibly where to split it.

Choose them based on how much information we would gain from the decision! (choose attribute that gives the highest gain)
Decision Tree Construction Algorithm

- Simple, greedy, recursive approach, builds up tree node-by-node

1. pick an attribute to split at a non-terminal node
2. split examples into groups based on attribute value
3. for each group:
   - if no examples – return majority from parent
   - else if all examples in same class – return class
   - else loop to step 1
### Attributes:

1. **Alternate**: whether there is a suitable alternative restaurant nearby.
2. **Bar**: whether the restaurant has a comfortable bar area to wait in.
3. **Fri/Sat**: true on Fridays and Saturdays.
4. **Hungry**: whether we are hungry.
5. **Patrons**: how many people are in the restaurant (values are None, Some, and Full).
6. **Price**: the restaurant's price range ($, $$, $$$).
7. **Raining**: whether it is raining outside.
8. **Reservation**: whether we made a reservation.
9. **Type**: the kind of restaurant (French, Italian, Thai or Burger).
10. **WaitEstimate**: the wait estimated by the host (0-10 minutes, 10-30, 30-60, >60).

[from: Russell & Norvig]
Attribute Selection

\[ IG(Y) = H(Y) - H(Y|X) \]

\[ IG(\text{type}) = 1 - \left[ \frac{2}{12} H(Y|\text{Fr.}) + \frac{2}{12} H(Y|\text{It.}) + \frac{4}{12} H(Y|\text{Thai}) + \frac{4}{12} H(Y|\text{Bur.}) \right] = 0 \]

\[ IG(\text{Patrons}) = 1 - \left[ \frac{2}{12} H(0, 1) + \frac{4}{12} H(1, 0) + \frac{6}{12} H(\frac{2}{6}, \frac{4}{6}) \right] \approx 0.541 \]
What Makes a Good Tree?

- Not too small: need to handle important but possibly subtle distinctions in data
- Not too big:
  - Computational efficiency (avoid redundant, spurious attributes)
  - Avoid over-fitting training examples
  - Human interpretability
- “Occam’s Razor”: find the simplest hypothesis that fits the observations
  - Useful principle, but hard to formalize (how to define simplicity?)
  - See Domingos, 1999, “The role of Occam’s razor in knowledge discovery”
- We desire small trees with informative nodes near the root
Decision Tree Miscellany

Problems:
- You have exponentially less data at lower levels
- Too big of a tree can overfit the data
- Greedy algorithms don’t necessarily yield the global optimum

Handling continuous attributes
- Split based on a threshold, chosen to maximize information gain

Decision trees can also be used for regression on real-valued outputs. Choose splits to minimize squared error, rather than maximize information gain.
Comparison to k-NN

Advantages of decision trees over KNN

- Good when there are lots of attributes, but only a few are important
- Good with discrete attributes
- Easily deals with missing values (just treat as another value)
- Robust to scale of inputs
- Fast at test time
- More interpretable

Advantages of KNN over decision trees

- Few hyperparameters
- Able to handle attributes/features that interact in complex ways (e.g. pixels)
- Can incorporate interesting distance measures (e.g. shape contexts)
- Typically make better predictions in practice
  - As we’ll see next lecture, ensembles of decision trees are much stronger. But they lose many of the advantages listed above.