Decision and Regression Trees

Applied Machine Learning
Derek Hoiem

Dall-E: A dirt road splits around a large gnarly tree, fractal art
Recap of classification and regression

• Nearest neighbor is widely used
  – Super-powers: can instantly learn new classes and predict from one or many examples

• Naïve Bayes represents a common assumption as part of density estimation, more typical as part of an approach rather than the final predictor
  – Super-powers: Fast estimation from lots of data; not terrible estimation from limited data

• Logistic Regression is widely used
  – Super-powers: Effective prediction from high-dimensional features; good confidence estimates

• Linear Regression is widely used
  – Super-powers: Can extrapolate, explain relationships, and predict continuous values from many variables

• Almost all algorithms involve nearest neighbor, logistic regression, or linear regression
  – The main learning challenge is typically feature learning
• So far, we’ve seen two main choices for how to use features
  1. Nearest neighbor uses all the features jointly to find similar examples
  2. Linear models make predictions out of weighted sums of the features
• If you wanted to give someone a rule to split the ‘o’ from the ‘x’, what other idea might you try?
  If $x_2 < 0.6$ and $x_2 > 0.2$ and $x_2 < 0.7$, ‘o’
  Else ‘x’
  Can we learn these kinds of rules automatically?
Decision trees

- Training: Iteratively choose the attribute and split value that best separates the classes for the data in the current node
- Combines feature selection/modeling with prediction
Decision Tree Classification

Test example

- width > 6.5cm?
  - Yes
    - height > 9.5cm?
      - Yes
        - Orange
      - No
        - Lemon
  - No
    - height > 6.0cm?
      - Yes
        - Lemon
      - No
        - Lemon

Slide Credit: Zemel, Urtasun, Fidler
**Example with discrete inputs**

<table>
<thead>
<tr>
<th>Example</th>
<th>Input Attributes</th>
<th>Goal</th>
<th>WillWait</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>Yes No No Yes Some $$$ No Yes French 0–10</td>
<td>$y_1 = \text{Yes}$</td>
<td></td>
</tr>
<tr>
<td>$x_2$</td>
<td>Yes No No Yes Full $ No No No Thai 30–60</td>
<td>$y_2 = \text{No}$</td>
<td></td>
</tr>
<tr>
<td>$x_3$</td>
<td>No Yes No No Some $ No No No Burger 0–10</td>
<td>$y_3 = \text{Yes}$</td>
<td></td>
</tr>
<tr>
<td>$x_4$</td>
<td>Yes No Yes Yes Full $ Yes No Thai 10–30</td>
<td>$y_4 = \text{Yes}$</td>
<td></td>
</tr>
<tr>
<td>$x_5$</td>
<td>Yes No Yes No Full $$$ No Yes French &gt;60</td>
<td>$y_5 = \text{No}$</td>
<td></td>
</tr>
<tr>
<td>$x_6$</td>
<td>No Yes No No Some $$ Yes Yes Italian 0–10</td>
<td>$y_6 = \text{Yes}$</td>
<td></td>
</tr>
<tr>
<td>$x_7$</td>
<td>No Yes No No None $ Yes No No Burger 0–10</td>
<td>$y_7 = \text{No}$</td>
<td></td>
</tr>
<tr>
<td>$x_8$</td>
<td>No No No No None $ No No No Burger &gt;60</td>
<td>$y_8 = \text{Yes}$</td>
<td></td>
</tr>
<tr>
<td>$x_9$</td>
<td>No Yes Yes No Full $ Yes No No Burger &gt;60</td>
<td>$y_9 = \text{No}$</td>
<td></td>
</tr>
<tr>
<td>$x_{10}$</td>
<td>Yes Yes Yes Yes Full $$$ No Yes Italian 10–30</td>
<td>$y_{10} = \text{No}$</td>
<td></td>
</tr>
<tr>
<td>$x_{11}$</td>
<td>No No No No None $ No No No Thai 0–10</td>
<td>$y_{11} = \text{No}$</td>
<td></td>
</tr>
<tr>
<td>$x_{12}$</td>
<td>Yes Yes Yes Yes Full $ No No No Burger 30–60</td>
<td>$y_{12} = \text{Yes}$</td>
<td></td>
</tr>
</tbody>
</table>

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).

*Slide Credit: Zemel, Urtasun, Fidler*
Example with discrete inputs

- The tree to decide whether to wait (T) or not (F)

**Attributes:**
1. Alternate: whether there is a suitable alternative restaurant nearby.
2. Bar: whether the restaurant has a comfortable seating area.
3. En/Out: true on weekends and holidays.
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 (S, B, BB).
7. Reserve: whether it is a reserved table.
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+).

**Figure Source:** Zemel, Urtasun, Fidler
Decision Trees

- Internal nodes test attributes
- Branching is determined by attribute value
- Leaf nodes are outputs (class assignments)

Figure Source: Zemel, Urtasun, Fidler
Decision tree algorithm

Training

Recursively, for each node in tree:

1. If labels in the node are mixed:
   a. Choose attribute and split values based on data that reaches each node
   b. Branch and create 2 (or more) nodes
2. Return
Decision tree algorithm

Training

Recursively, for each node in tree:

1. If labels in the node are mixed:
   a. Choose attribute and split values based on data that reaches each node
   b. Branch and create 2 (or more) nodes

2. Return
Decision tree algorithm

Training
Recursively, for each node in tree:
1. If labels in the node are mixed:
   a. Choose attribute and split values based on data that reaches each node
   b. Branch and create 2 (or more) nodes
2. Return
Decision tree algorithm

Training

Recursively, for each node in tree:

1. If labels in the node are mixed:
   a. Choose attribute and split values based on data that reaches each node
   b. Branch and create 2 (or more) nodes

2. Return
Decision tree algorithm

Training

Recursively, for each node in tree:

1. If labels in the node are mixed:
   a. Choose attribute and split values based on data that reaches each node
   b. Branch and create 2 (or more) nodes

2. Return
Decision tree algorithm

**Prediction**
1. Check conditions to descend tree
2. Return label of leaf node
How do you choose what/where to split?

Which attribute is better to split on, $X_1$ or $X_2$?

<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_2$</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

Idea: Use counts at leaves to define probability distributions, so we can measure uncertainty

Slide Source: Zemel, Urtasun, Fidler
Quantifying Uncertainty: Coin Flip Example

Sequence 1:
0 0 0 1 0 0 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

2

8

10
Quantifying Uncertainty: Coin Flip Example

Entropy $H$:

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

- $8/9$  
- $1/9$

- $4/9$  
- $5/9$

$$-\frac{8}{9} \log_2 \frac{8}{9} - \frac{1}{9} \log_2 \frac{1}{9} \approx \frac{1}{2}$$

$$-\frac{4}{9} \log_2 \frac{4}{9} - \frac{5}{9} \log_2 \frac{5}{9} \approx 0.99$$

- How surprised are we by a new value in the sequence?
- How much information does it convey?

Slide Source: Zemel, Urtasun, Fidler
Quantifying Uncertainty: Coin Flip Example

Entropy: $H(X) = -\sum_{x \in X} p(x) \log_2 p(x)$
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}

Slide Source: Zemel, Urtasun, Fidler
Specific 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 $Y$, 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>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>

The expected conditional entropy:

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

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

Slide Source: Zemel, Urtasun, Fidler
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?

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

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

$$\approx 0.75 \text{ bits}$$
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?
  \[
  IG(Y|X) = H(Y) - H(Y|X)
  \approx 0.25 \text{ bits}
  \]

- Also called **information gain** in $Y$ due to $X$
- If $X$ is completely uninformative about $Y$: $IG(Y|X) = 0$
- If $X$ is completely informative about $Y$: $IG(Y|X) = H(Y)$
- How can we use this to construct our decision tree?
Constructing decision tree

Training
Recursively, for each node in tree:

1. If labels in the node are mixed:
   a. Choose attribute and split values based on data that reaches each node
   b. Branch and create 2 (or more) nodes
2. Return

1. Measure information gain
   • For each discrete attribute: compute information gain of split
   • For each continuous attribute: select most informative threshold and compute its information gain. Can be done efficiently based on sorted values.
2. Select attribute / threshold with highest information gain
Pause, stretch, and think: Is it better to split based on type or patrons?
\[ IG(Y) = H(Y) - H(Y|X) \]

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

\[ IG(Patrons) = 1 - \left[ \frac{2}{12} H(0,1) + \frac{4}{12} H(1,0) + \frac{6}{12} H\left(\frac{2}{6}, \frac{4}{6}\right) \right] \approx 0.541 \]
What if you need to predict a continuous value?

• Regression Tree
  – Same idea, but choose splits to minimize sum squared error
    \[ \sum_{n \in \text{node}} (f_{\text{node}}(x_n) - y_n)^2 \]
  – \(f_{\text{node}}(x_n)\) typically returns the mean prediction value of data points in the leaf node containing \(x_n\)
  – What are we minimizing?
Variants

• Different splitting criteria, e.g. Gini index: \( 1 - \sum_i p_i^2 \) (very similar result, a little faster to compute)

• Most commonly, split on one attribute at a time
  – In case of continuous vector data, can also split on linear projections of features

• Can stop early
  – when leaf node contains fewer than \( N_{\text{min}} \) points
  – when max tree depth is reached

• Can also predict multiple continuous values or multiple classes
Decision Tree vs. 1-NN

- Both have piecewise-linear decisions
- Decision tree is typically “axis-aligned”
- Decision tree has ability for early stopping to improve generalization
- True power of decision trees arrives with ensembles (lots of small or randomized trees)
Regression Tree for Temperature Prediction

- Min leaf size: 200
- RMSE = 3.42
- $R^2 = 0.88$

from sklearn import tree
from sklearn.tree import DecisionTreeRegressor
model = DecisionTreeRegressor(random_state=0, min_samples_leaf=200)
model.fit(x_train, y_train)
y_pred = model.predict(x_val)
tree_rmse = np.sqrt(np.mean((y_pred-y_val)**2))
tree_mae = np.sqrt(np.median(np.abs(y_pred-y_val)))
print('LR: RMSE={}, MAE={}'.format(tree_rmse, tree_mae))
print('R^2: {}'.format(1-tree_rmse**2/np.mean((y_pred-y_pred.mean())**2)))
plt.figure(figsize=(20,20))
tree.plot_tree(model)
plt.show()
Classification/Regression Trees Summary

• Key Assumptions
  – Samples with similar features have similar predictions

• Model Parameters
  – Tree structure with split criteria at each internal node and prediction at each leaf node

• Designs
  – Limits on tree growth
  – What kinds of splits are considered
  – Criterion for choosing attribute/split (e.g. gini impurity score is another common choice)

• When to Use
  – Want an explainable decision function (e.g. for medical diagnosis)
  – As part of an ensemble (as we’ll see Thursday)

• When Not to Use
  – One tree is not a great performer, but a forest is
Compare classifiers

\[ score(y) = w^T x + b \]

\[ score(y_n = 1) = w^T x_n + b \]
Things to remember

• Decision/regression trees learn to split up the feature space into partitions with similar values.

• Entropy is a measure of uncertainty.

• Information gain measures how much particular knowledge reduces prediction uncertainty.

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

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

• Ensembles: model averaging and forests