Decision Trees

COMP 135 Intro to Machine Learning
Liping Liu

slides modified from Roni Khardon's with permission

Decision tree

Let's look at a simple dataset for motivation:

<table>
<thead>
<tr>
<th>Outlook</th>
<th>Temp</th>
<th>Humidity</th>
<th>Windy</th>
<th>Play</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>False</td>
<td>No</td>
</tr>
<tr>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>True</td>
<td>No</td>
</tr>
<tr>
<td>Rainy</td>
<td>Mild</td>
<td>High</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>Rainy</td>
<td>Cool</td>
<td>Normal</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>True</td>
<td>Yes</td>
</tr>
<tr>
<td>Rainy</td>
<td>Mild</td>
<td>High</td>
<td>False</td>
<td>No</td>
</tr>
<tr>
<td>Rainy</td>
<td>Cool</td>
<td>Normal</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>Rainy</td>
<td>Mild</td>
<td>Normal</td>
<td>True</td>
<td>Yes</td>
</tr>
<tr>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>Rainy</td>
<td>Mild</td>
<td>High</td>
<td>True</td>
<td>No</td>
</tr>
</tbody>
</table>

- Class: Play tennis
- Attributes:
  - Outlook
  - Temp
  - Humidity
  - Windy

Decision Trees

- DTs identify regions in instance space:
  - Recursively split on values of features to define regions that have single label
- What does this look like with numerical features? (with threshold node tests, for example temp>23)

Decision Trees

- Decision trees can represent any discrete function of discrete attributes!
- Why?

Decision Trees

- Given training set, can we build a tree that agrees with the data? (yes; easy; why?)
- What is a good decision tree?
- Given training set, how can we build a good tree?
Decision Trees

- To develop the algorithm
- Let's focus on a simpler question first: which attribute should we choose for root of tree?
- Our choices are ...

Decision Trees

- Which attribute should we choose for root of tree?
- A numerical example:
  
  \[\begin{align*}
  \text{[Pos, Neg]} & \rightarrow [35, 15] + [15, 35] \\
  & \quad + [50, 0] + [0, 50] \\
  & \quad + [10, 30] + [30, 10] + [10, 10] \\
  & \quad + [25, 25] + [25, 25]
  \end{align*}\]

Decision Tree Learning Algorithm

- Assume for the moment that we have some good method of picking root attribute. What's next?

Continuing to Split

Assume we picked Outlook for the root. Then we must continue splitting each branch Until ... ?

- If data has a pure class
  - Make leaf node with that class
- Otherwise
  - Pick feature to split on
  - Divide data into sub-datasets according to the feature’s values
  - Recursively build a tree for each subset
Decision Trees

- Several selection criteria have been proposed and used.
- Information gain is commonly used (C4.5, J48)

- We need to learn about entropy ...

Entropy

- In theory, entropy characterizes the uncertainty of a distribution
  \[\text{Ent}(P) = -\sum_{x \in \mathcal{X}} P(x = x) \log(P(x = x)) = E_x[\log(P(x))]\]

  - Example: Bernoulli distribution \(P\) with parameter \(\mu\)
    \[\text{Ent}(P) = -\sum_{x = 0, 1} P(x = x) \log(P(x = x)) = -(1 - \mu) \log(1 - \mu) - \mu \log(\mu)\]

Decision Trees

- Uncertainty reduction of a split

  Now consider a split \(S \rightarrow S_1, \ldots, S_k\)
  where the \(S_i\) are subsets of \(S\)
  and may include examples from multiple classes

  \[\text{Gain(Split)} = \text{Ent}(S) - \sum_{j=1}^{k} \frac{|S_j|}{|S|} \text{Ent}(S_j)\]

  \(\text{Ent}(S)\) is the entropy of labels in the set \(S\). Same for \(\text{Ent}(S_i)\).
### Example: calculating Gain

- **Outlook = Sunny:**
  
  \[
  \text{entropy}(2/5, 3/5) = -2/5 \log_2(2/5) - 3/5 \log_2(3/5) = 0.971 \text{ bits}
  \]

- **Outlook = Overcast:**
  
  \[
  \text{entropy}(1, 0) = -1 \log_2(1) - 0 \log_2(0) = 0 \text{ bits}
  \]

- **Outlook = Rainy:**
  
  \[
  \text{entropy}(3/5, 2/5) = -3/5 \log_2(3/5) - 2/5 \log_2(2/5) = 0.971 \text{ bits}
  \]

- **Expected information for attribute:**
  
  \[
  \text{info}(3, 2, 4, 0, 3, 2) = (5/14) \times 0.971 + (4/14) \times 0 + (5/14) \times 0.693 = 0.693 \text{ bits}
  \]

---

### Decision Trees

\[
\text{gain}(\text{Outlook}) = \text{info}(9, 5) - \text{info}(2, 3, 4, 0, 3, 2) = 0.940 - 0.693 = 0.247 \text{ bits}
\]

\[
\text{gain}(\text{Temperature}) = 0.029 \text{ bits}
\]

\[
\text{gain}(\text{Humidity}) = 0.152 \text{ bits}
\]

\[
\text{gain}(\text{Windy}) = 0.048 \text{ bits}
\]

---

### Decision Tree Learning Algorithm

- **If data has a pure class**
  - Make leaf node with that class
- **Otherwise**
  - Pick feature that maximizes information gain to split on
  - Divide data into sub-datasets according to the feature’s values
  - Recursively build a tree for each subset

---

### Improved Heuristic for Wide Splits

- **Information Gain prefers wide splits**
- **Cancel the effect by split information**

\[
\text{Gain}(\text{Split}) = \text{Ent}(S) - \sum_{j} \frac{|S_j|}{|S|} \text{Ent}(S_j)
\]

\[
\text{SplitInfo} = \sum_{j} \frac{|S_j|}{|S|} \log \frac{|S|}{|S_j|}
\]

\[
\text{GainRatio} = \frac{\text{Gain}}{\text{SplitInfo}}
\]
Other Criteria

- Reduce other criteria other than entropy

The Gini Criterion = $2p(1 - p)$
Classification error = $\min(p, 1 - p)$

Mathematical expression for Gini Criterion:
$$G = \sum_{i=1}^{n} p_i(1 - p_i)$$

What does this look like?

Regression tree

- Reduce squared error

Squared error = $\frac{1}{n} \sum_{i=1}^{n} (y_i - \bar{y})^2$, with $\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i$

Note: $y_i$ is continuous here

Let $R$ be the set of instances in parent node, and $R_1$ and $R_2$ be the sets for two children, $R = R_1 \cup R_2$

Error reduction = $\text{sq_err}(R) - \frac{|R_1|}{|R|} \text{sq_err}(R_1) - \frac{|R_2|}{|R|} \text{sq_err}(R_2)$

Real Valued Attributes

- Naïve treatment makes a very wide split with possibly one example per branch.
- Is this good?

- Alternative picks threshold $t$ and tests
  (feature $\geq t$)
  to get a binary split.
- How can we pick $t$?

Missing Attribute Values

- Common in real data

- We can handle this in a way that works across algorithms (that is, also for kNN).
- How?

- But we can do better with a solution tailored for decision trees. How?

The Bad News

- So far great news
- Simple recursive algorithm
- And some improvements

- But …
The Bad News

• This is an example of overfitting

![Graph showing accuracy vs size of tree](image)

Overfitting in DT

• Why Does this happen?
  • Few examples at lower levels in tree
  • Quantities calculated “not reliable” in this case
  • Even worse with “noisy data”
  • And when features not sufficiently rich
  • Solutions?

Overfitting in DT

• Min # points at leaf for split to be legal
• Stop growing tree if “no information”
• Pruning: grow full tree and then test whether some parts should be removed.
• How? Note that full tree always looks better on training data! so just using accuracy on training data will not work

Pruning in C4.5 / J48

Standard Normal distribution

![Graph showing standard normal distribution](image)

Pruning in C4.5 / J48

• $p$ is true error and $f$ is observed error
• Algorithm uses
  \[ f \sim N \left( p, \frac{p(1-p)}{n} \right) \]
  and some reasoning to claim that (actual formula used by C4.5 is more complex):
  \[ p \leq f + \sqrt{\frac{1}{4n} Z^2} \]
  with confidence $1 - \alpha$.
• The error rate at each node is replaced with the upper bound
• Then the best pruning can be chosen
Overfitting in DT

- Solution 1 (C4.5, J48): uses a confidence interval based on class ratio at leaf and number of examples in the training set.
- This is not fully justified but works well in practice.

- Solution 2: use a validation set. Known as reduced error pruning (REP)

REP Example

<table>
<thead>
<tr>
<th>Decision</th>
<th>Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keep</td>
<td>3</td>
</tr>
<tr>
<td>Prune</td>
<td>5</td>
</tr>
</tbody>
</table>

REP Example

<table>
<thead>
<tr>
<th>Decision</th>
<th>Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keep</td>
<td>3</td>
</tr>
<tr>
<td>Prune</td>
<td>4</td>
</tr>
</tbody>
</table>

REP Example

 Decision  Errors
 Keep       7
 Prune      4

REP Example

 Decision  Errors
 Keep       3
 Prune      5

REP Example

 Decision  Errors
 Keep       3
 Prune      4
DT Recap

- DT divide the example space through recursive splits of feature values
- Recursive learning algorithm relies on good choice of root attribute
- IG and other criteria are used for choice
- Several variants, improvements, and generalizations
- Overfitting is a significant issue: solved by pruning or other methods