Lecture 06

Decision Trees

STAT 451: Intro to Machine Learning, Fall 2020
Sebastian Raschka
http://stat.wisc.edu/~sraschka/teaching/stat451-fs2020/
Lecture 6: Decision Trees

Topics

1. Intro to decision trees

2. Recursive algorithms & Big-O

3. Types of decision trees

4. Splitting criteria

5. Gini & Entropy vs misclassification error

6. Improvements & dealing with overfitting

7. Code example
Decision Tree Terminology

- **Root node**: Work to do?
  - **Yes**: Stay in
  - **No**: Outlook?
    - Sunny: Go to beach
    - Rainy: Overcast
      - Yes: Go running
      - No: Friends busy?
        - Yes: Stay in
        - No: Go to movies

- **Internal node**: Decision Tree Terminology
- **Branch**: Decision Tree Terminology
- **Leaf node**: Decision Tree Terminology
Decision Trees as Rulesets

IF

_________________________    ___    ______________________
___   ______________________

THEN

________     ___     ___________
Sulfur <= 0.5  
gini = 0.478  
samples = 38  
value = [23, 15]  
class = non-active

12-Hydroxy <= 0.5  
gini = 0.219  
samples = 24  
value = [21, 3]  
class = non-active

Sulfate-Ester <= 0.5  
gini = 0.245  
samples = 14  
value = [2, 12]  
class = active

3-Keto <= 0.5  
gini = 0.48  
samples = 5  
value = [3, 2]  
class = non-active

gini = 0.0  
samples = 16  
value = [16, 0]  
class = non-active

gini = 0.444  
samples = 3  
value = [2, 1]  
class = non-active

gini = 0.5  
samples = 4  
value = [2, 2]  
class = non-active

gini = 0.0  
samples = 1  
value = [1, 0]  
class = non-active

gini = 0.48  
samples = 5  
value = [2, 4]  
class = active

gini = 0.0  
samples = 1  
value = [0, 1]  
class = active

“ENE4:” 90.5 %  
ZINC72400307: 90.4 %  
ZINC03876071: 3.2 %  
ZINC40576706: 0.0 %
ZINC 12

ZINC03876071

In ZINC since | Heavy atoms | Benign functionality
--- | --- | ---
October 5th, 2005 | 32 | No

Popular Name: DEXAMETHASONE SODIUM PHOSPHATE

Other Names:
21-Discodium phosphate dexamethasone; o-Fluoro-11beta,17,21-trihydroxy-16alpha-methylpregna-1,4-diene-3,20-dione 21-(dihydrogen phosphate)
disodium salt; C22H28F3O8P2Na; Corson; DECADRON; DECADRON W/ XYLOCAINE; DEXACEN-4; DEXACORT; DEXAIR; DEXAMETHASONE; D
2392-39-4: C08175; Dexamethasone sodium phosphate
2392-39-4: D00975; Dalalone (TN); Dexamethasone sodium phosphate (JAN/USP); Maxidex (TN)
Random forests, adaptive boosting, gradient boosting
Lecture 6: Decision Trees

Topics

1. Intro to decision trees

2. Recursive algorithms & Big-O

3. Types of decision trees

4. Splitting criteria

5. Gini & Entropy vs misclassification error

6. Improvements & dealing with overfitting

7. Code example
Recursion / Recursive Algorithms

```python
1 def some_fun(x):
2     if x == []:
3         return 0
4     else:
5         return 1 + some_fun(x[1:])
```

What does this function do?
Divide & Conquer Algorithms: Quicksort

```python
def quicksort(array):
    if len(array) < 2:
        return array
    else:
        pivot = array[0]
        smaller, bigger = [], []
        for ele in array[1:]:
            if ele <= pivot:
                smaller.append(ele)
            else:
                bigger.append(ele)
        return quicksort(smaller) + [pivot] + quicksort(bigger)
```

Divide & Conquer Algorithms: Quicksort

```python
def quicksort(array):
    if len(array) < 2:
        return array
    else:
        pivot = array[0]
        smaller, bigger = [], []
        for ele in array[1:]:
            if ele <= pivot:
                smaller.append(ele)
            else:
                bigger.append(ele)
        return quicksort(smaller) + [pivot] + quicksort(bigger)
```
Time complexity of quicksort:

\[ \mathcal{O}(\_\_\_\_\_) \]

```python
def quicksort(array):
    if len(array) < 2:
        return array
    else:
        pivot = array[0]
        smaller, bigger = [], []
        for ele in array[1:]:
            if ele <= pivot:
                smaller.append(ele)
            else:
                bigger.append(ele)
        return quicksort(smaller) + [pivot] + quicksort(bigger)
```
# Array Sorting Algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Time Complexity</th>
<th>Space Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Best</td>
<td>Average</td>
</tr>
<tr>
<td>Quicksort</td>
<td>$\Omega(n \log(n))$</td>
<td>$\Theta(n \log(n))$</td>
</tr>
<tr>
<td>Mergesort</td>
<td>$\Omega(n \log(n))$</td>
<td>$\Theta(n \log(n))$</td>
</tr>
<tr>
<td>Timsort</td>
<td>$\Omega(n)$</td>
<td>$\Theta(n \log(n))$</td>
</tr>
<tr>
<td>Heapsort</td>
<td>$\Omega(n \log(n))$</td>
<td>$\Theta(n \log(n))$</td>
</tr>
<tr>
<td>Bubble Sort</td>
<td>$\Omega(n)$</td>
<td>$\Theta(n^2)$</td>
</tr>
<tr>
<td>Insertion Sort</td>
<td>$\Omega(n)$</td>
<td>$\Theta(n^2)$</td>
</tr>
<tr>
<td>Selection Sort</td>
<td>$\Omega(n^2)$</td>
<td>$\Theta(n^2)$</td>
</tr>
<tr>
<td>Tree Sort</td>
<td>$\Omega(n \log(n))$</td>
<td>$\Theta(n \log(n))$</td>
</tr>
<tr>
<td>Shell Sort</td>
<td>$\Omega(n \log(n))$</td>
<td>$\Theta(n(\log(n))^2)$</td>
</tr>
<tr>
<td>Bucket Sort</td>
<td>$\Omega(n+k)$</td>
<td>$\Theta(n+k)$</td>
</tr>
<tr>
<td>Radix Sort</td>
<td>$\Omega(nk)$</td>
<td>$\Theta(nk)$</td>
</tr>
<tr>
<td>Counting Sort</td>
<td>$\Omega(n+k)$</td>
<td>$\Theta(n+k)$</td>
</tr>
<tr>
<td>Cubesort</td>
<td>$\Omega(n)$</td>
<td>$\Theta(n \log(n))$</td>
</tr>
</tbody>
</table>

* "worst" ~ inversely-sorted array

http://www.bigocheatsheet.com
Decision Tree in Pseudocode

GenerateTree($\mathcal{D}$):

- if $y = 1 \forall \langle x, y \rangle \in \mathcal{D}$ or $y = 0 \forall \langle x, y \rangle \in \mathcal{D}$:
  - return Tree

- else:
  - Pick best feature $x_j$:
    - $\mathcal{D}_0$ at Child$_0$ : $x_j = 0 \forall \langle x, y \rangle \in \mathcal{D}$
    - $\mathcal{D}_1$ at Child$_1$ : $x_j = 1 \forall \langle x, y \rangle \in \mathcal{D}$
  - return Node($x_j$, GenerateTree($\mathcal{D}_0$), GenerateTree($\mathcal{D}_1$))
Time Complexity ("Big-O")

Growing the tree: $\mathcal{O}(\ldots)$

Tip: It can be shown that optimal split is on boundary between adjacent examples (similar feature value) with different class labels.

Time Complexity ("Big-O")

Querying the tree: $\mathcal{O}(\ldots)$
Lecture 6: Decision Trees

Topics

1. Intro to decision trees

2. Recursive algorithms & Big-O

3. Types of decision trees

4. Splitting criteria

5. Gini & Entropy vs misclassification error

6. Improvements & dealing with overfitting

7. Code example
Decision Tree in Pseudocode

GenerateTree(\mathcal{D}):  

- if \( y = 1 \ \forall \langle x, y \rangle \in \mathcal{D} \) or \( y = 0 \ \forall \langle x, y \rangle \in \mathcal{D} \):
  - return Tree
- else:
  - Pick best feature \( x_j \):
    - \( \mathcal{D}_0 \) at Child\(_0\) : \( x_j = 0 \ \forall \langle x, y \rangle \in \mathcal{D} \)
    - \( \mathcal{D}_1 \) at Child\(_1\) : \( x_j = 1 \ \forall \langle x, y \rangle \in \mathcal{D} \)
  
  return Node(\( x_j \), GenerateTree(\( \mathcal{D}_0 \)), GenerateTree(\( \mathcal{D}_1 \)))
Generic Tree Growing Algorithm

1) Pick the feature that, when parent node is split, results in the largest information gain

2) Stop if child nodes are pure or information gain $\leq 0$

3) Go back to step 1 for each of the two child nodes
Generic Tree Growing Algorithm

1) Pick the feature that, when parent node is split, results in the largest information gain

2) Stop if child nodes are pure or information gain \( \leq 0 \)

3) Go back to step 1 for each of the two child nodes

• How make predictions of features in dataset not sufficient to make child nodes pure?
Design choices

- How to split
  - what measurement/criterion as measure of goodness
  - binary vs multi-category split

- When to stop
  - if leaf nodes contain only examples of the same class
  - feature values are all the same for all examples
  - statistical significance test
ID3 -- Iterative Dichotomizer 3

- one of the earlier/earliest decision tree algorithms
- cannot handle numeric features
- no pruning, prone to overfitting
- short and wide trees (compared to CART)
- maximizing information gain/minimizing entropy
- discrete features, binary and multi-category features
C4.5

• continuous and discrete features

• continuous is very expensive, because must consider all possible ranges
• handles missing attributes (ignores them in gain compute)
• post-pruning (bottom-up pruning)
• Gain Ratio
CART


• continuous and discrete features

• strictly binary splits (taller trees than ID3, C4.5)

• binary splits can generate better trees than C4.5, but tend to be larger and harder to interpret; k-attributes has a ways to create a binary partitioning

• variance reduction in regression trees

• Gini impurity, twoing criteria in classification trees

• cost complexity pruning
Others


- C5.0 (patented)

- ...
Lecture 6: Decision Trees

Topics

1. Intro to decision trees

2. Recursive algorithms & Big-O

3. Types of decision trees

4. Splitting criteria

5. Gini & Entropy vs misclassification error

6. Improvements & dealing with overfitting

7. Code example
## Finding a Decision Rule

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 cm</td>
<td>8 cm</td>
<td>9 cm</td>
<td>1</td>
</tr>
<tr>
<td>4 cm</td>
<td>11 cm</td>
<td>2 cm</td>
<td>0</td>
</tr>
<tr>
<td>6 cm</td>
<td>12 cm</td>
<td>4 cm</td>
<td>0</td>
</tr>
<tr>
<td>10 cm</td>
<td>9 cm</td>
<td>3 cm</td>
<td>1</td>
</tr>
<tr>
<td>5 cm</td>
<td>7 cm</td>
<td>8 cm</td>
<td>0</td>
</tr>
<tr>
<td>8 cm</td>
<td>9 cm</td>
<td>3 cm</td>
<td>1</td>
</tr>
<tr>
<td>3 cm</td>
<td>11 cm</td>
<td>5 cm</td>
<td>0</td>
</tr>
</tbody>
</table>
Drawing a Decision Boundary

<table>
<thead>
<tr>
<th>x₁</th>
<th>x₂</th>
<th>x₃</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 cm</td>
<td>8 cm</td>
<td>9 cm</td>
<td>1</td>
</tr>
<tr>
<td>4 cm</td>
<td>11 cm</td>
<td>2 cm</td>
<td>0</td>
</tr>
<tr>
<td>6 cm</td>
<td>12 cm</td>
<td>4 cm</td>
<td>0</td>
</tr>
<tr>
<td>10 cm</td>
<td>9 cm</td>
<td>3 cm</td>
<td>1</td>
</tr>
<tr>
<td>5 cm</td>
<td>7 cm</td>
<td>8 cm</td>
<td>0</td>
</tr>
<tr>
<td>8 cm</td>
<td>9 cm</td>
<td>3 cm</td>
<td>1</td>
</tr>
<tr>
<td>3 cm</td>
<td>11 cm</td>
<td>5 cm</td>
<td>0</td>
</tr>
</tbody>
</table>
Entropy: $H(X) = -\sum_{i=1}^{c} P(x_i) \log_2 P(x_i)$

Samples: $n$ (number of samples)

Value: $[a, b]$ (range of values)

Class: $0$ or $1$ (classification)

$x_1 \leq 5.5$
- Entropy: $0.985$
- Samples: 7
- Value: $[4, 3]$
- Class: $0$

$x_2 \leq 10.5$
- Entropy: $0.811$
- Samples: 4
- Value: $[1, 3]$
- Class: $1$

True
- $x_1 \leq 5.5$
- Entropy: $0.0$
- Samples: 3
- Value: $[3, 0]$
- Class: $0$

False
- $x_2 \leq 10.5$
- Entropy: $0.0$
- Samples: 1
- Value: $[1, 0]$
- Class: $0$
\[ x_2 \leq 10.0 \]
\[ \text{entropy} = 0.985 \]
\[ \text{samples} = 7 \]
\[ \text{value} = [4, 3] \]
\[ \text{class} = \text{Class 0} \]

\[ x_2 \leq 7.5 \]
\[ \text{entropy} = 0.811 \]
\[ \text{samples} = 4 \]
\[ \text{value} = [1, 3] \]
\[ \text{class} = \text{Class 1} \]

\[ \text{True} \]
\[ \text{entropy} = 0.0 \]
\[ \text{samples} = 3 \]
\[ \text{value} = [3, 0] \]
\[ \text{class} = \text{Class 0} \]

\[ \text{False} \]
\[ \text{entropy} = 0.0 \]
\[ \text{samples} = 1 \]
\[ \text{value} = [1, 0] \]
\[ \text{class} = \text{Class 0} \]
The Splitting Criterion
Information Gain

\[ GAIN(\mathcal{D}, x_j) = H(\mathcal{D}) - \sum_{v \in \text{Values}(x_j)} \frac{|\mathcal{D}_v|}{|\mathcal{D}|} H(\mathcal{D}_v) \]
Shannon Entropy

Refer to lecture notes
Entropy

\[ H = - \sum_i p(i \mid x_j) \log_2(p(i \mid x_j)) \]
Gini Impurity

\[ Gini = 1 - \sum_i (p(i | x_j)^2) \]
Misclassification Error

\[ ERR = \frac{1}{n} \sum_{i=1}^{n} L(\hat{y}^i, y^i), \]

\[ L(\hat{y}, y) = \begin{cases} 
0 & \text{if } \hat{y} = y, \\
1 & \text{otherwise.} 
\end{cases} \]
Misclassification Error

\[ ERR = 1 - \max_{i}(p(i \mid x_j)) \]
Lecture 6: Decision Trees

Topics

1. Intro to decision trees
2. Recursive algorithms & Big-O
3. Types of decision trees
4. Splitting criteria
5. Gini & Entropy vs misclassification error
6. Improvements & dealing with overfitting
7. Code example
Why Growing Decision Trees via Entropy instead of Misclassification Error?
Why Growing Decision Trees via Entropy instead of Misclassification Error?

\[ GAIN(\mathcal{D}, x_j) = I(\mathcal{D}) - \sum_{v \in \text{Values}(x_j)} \frac{|\mathcal{D}_v|}{|\mathcal{D}|} I(\mathcal{D}_v) \]
\[ y = 1 \quad y = 0 \]

\[ \begin{array}{c|c}
40 & 80 \\
\end{array} \]

\[ x_1 = 1 ? \]

\[ \begin{array}{c|c}
28 & 42 \\
\end{array} \]

No

\[ \begin{array}{c|c}
28 & 0 \\
\end{array} \]

No

\[ x_2 = 1 ? \]

\[ \begin{array}{c|c}
0 & 42 \\
\end{array} \]

Yes

\[ \begin{array}{c|c}
12 & 38 \\
\end{array} \]

Yes

\[ x_3 = 1 ? \]

\[ \begin{array}{c|c}
12 & 0 \\
\end{array} \]

No

\[ \begin{array}{c|c}
0 & 38 \\
\end{array} \]
Entropy = 0.918

| 40 | 80 |

$GAIN(D, x_j) = H(D) - \frac{|D_{x_j=1}|}{|D|} H(D_{x_j=1}) - \frac{|D_{x_j=0}|}{|D|} H(D_{x_j=0})$

$= 0.918 - \frac{70}{120} \times 0.971 - \frac{50}{120} \times 0.795$

$= 0.02$

Entropy = 0.971

| 28 | 42 |

Entropy = 0.0

| 28 | 0 |

Entropy = 0.0

| 0 | 42 |

Entropy = 0.0

| 12 | 38 |

Entropy = 0.795

| 12 | 0 |

Entropy = 0.0

| 0 | 38 |

Entropy = 0.0
GAIN(\mathcal{D}, x_j) = ERR(\mathcal{D}) - \frac{|\mathcal{D}_{x_j=1}|}{|\mathcal{D}|} ERR(\mathcal{D}_{x_j=1}) - \frac{|\mathcal{D}_{x_j=0}|}{|\mathcal{D}|} ERR(\mathcal{D}_{x_j=0})

= \frac{40}{120} - \frac{70}{120} \times \frac{28}{70} - \frac{50}{120} \times \frac{12}{50}

= 0
Lecture 6: Decision Trees

Topics

1. Intro to decision trees
2. Recursive algorithms & Big-O
3. Types of decision trees
4. Splitting criteria
5. Gini & Entropy vs misclassification error

6. Improvements & dealing with overfitting

7. Code example
Gain Ratio

\[ \text{GainRatio}(\mathcal{D}, x_j) = \frac{\text{Gain}(\mathcal{D}, x_j)}{\text{SplitInfo}(\mathcal{D}, x_j)} \]

where the split information measures the entropy of the feature:

\[ \text{SplitInfo}(\mathcal{D}, x_j) = - \sum_{v \in x_j} \frac{|D_v|}{|\mathcal{D}|} \log_2 \frac{|D_v|}{|\mathcal{D}|} \]

Penalizes splitting categorical attributes with many values (e.g., think date column, or really bad: row ID) via the split information

Quinlan 1986
Shortcomings

How would the decision tree split look like?
Overfitting

Q: Why (when) does the accuracy start at ~50%?
Pre-Pruning

- Set a depth cut-off (maximum tree depth) \textit{a priori}
- Cost-complexity pruning: \( C = \alpha I(\text{impurity measure}) + \beta \text{total number of nodes} \), where \( \alpha \) is a tuning parameter, and \( \beta \) is the total number of nodes.
- Stop growing if split is not statistically significant (e.g., \( \chi^2 \) test)
- Set a minimum number of data points for each node
Post-Pruning

• Grow full tree first, then remove nodes, in C4.5

• Reduced-error pruning, remove nodes via validation set eval. (problematic for limited data)

• Can also convert trees to rules first and then prune the rules
Post-Pruning

![Graph showing training and validation accuracy over tree depth](graph.png)

- **Training accuracy**
- **Validation accuracy**
- **Validation accuracy (post-pruning)**
Regression Trees
Decision Tree Summary: Pros and Cons

- (+) Easy to interpret and communicate
- (+) Can represent "complete" hypothesis space
- (-) Easy to overfit
- (-) Elaborate pruning required
- (-) Expensive to just fit a "diagonal line"
- (-) Output range is bounded (dep. on training examples) in regression trees
Decision Trees and ML Categories

- Supervised vs. unsupervised learning algorithm
- Classification vs. regression
- Optimization method: ______
- Eager vs. lazy learning algorithm
- Batch vs. online learning algorithm
- Parametric vs. nonparametric model
- Deterministic vs. stochastic
Lecture 6: Decision Trees

Topics

1. Intro to decision trees
2. Recursive algorithms & Big-O
3. Types of decision trees
4. Splitting criteria
5. Gini & Entropy vs misclassification error
6. Improvements & dealing with overfitting
7. Code example
Demo

06-trees_demo.ipynb