Lecture 9: Decision Tree

Hsuan-Tien Lin (林軒田)
htlin@csie.ntu.edu.tw

Department of Computer Science & Information Engineering
National Taiwan University (國立台灣大學資訊工程系)
Roadmap

1. Embedding Numerous Features: Kernel Models
2. Combining Predictive Features: Aggregation Models

Lecture 8: Adaptive Boosting

- optimal re-weighting for diverse hypotheses
- and adaptive linear aggregation to boost ‘weak’ algorithms

Lecture 9: Decision Tree

- Decision Tree Hypothesis
- Decision Tree Algorithm
- Decision Tree Heuristics in C&RT
- Decision Tree in Action

3. Distilling Implicit Features: Extraction Models
### What We Have Done

<table>
<thead>
<tr>
<th>aggregation type</th>
<th>blending</th>
<th>learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>uniform</td>
<td>voting/averaging</td>
<td>Bagging</td>
</tr>
<tr>
<td>non-uniform</td>
<td>linear</td>
<td>AdaBoost</td>
</tr>
<tr>
<td>conditional</td>
<td>stacking</td>
<td><strong>Decision Tree</strong></td>
</tr>
</tbody>
</table>

- **blending**: aggregate after getting $g_t$;
- **learning**: aggregate as well as getting $g_t$

**decision tree**: a traditional learning model that realizes **conditional aggregation**
Decision Tree for Watching MOOC Lectures

\[ G(x) = \sum_{t=1}^{T} q_t(x) \cdot g_t(x) \]

- **base hypothesis** \( g_t(x) \): leaf at end of path \( t \), a **constant** here
- **condition** \( q_t(x) \): \( \text{is } x \text{ on path } t? \)
- usually with **simple** internal nodes

**decision tree:** arguably one of the most **human-mimicking models**
Recursive View of Decision Tree

Path View: \( G(\mathbf{x}) = \sum_{t=1}^{T} [\mathbf{x} \text{ on path } t] \cdot \text{leaf}_t(\mathbf{x}) \)

Recursive View

\[
G(\mathbf{x}) = \sum_{c=1}^{C} [b(\mathbf{x}) = c] \cdot G_c(\mathbf{x})
\]

- \( G(\mathbf{x}) \): full-tree hypothesis
- \( b(\mathbf{x}) \): branching criteria
- \( G_c(\mathbf{x}) \): sub-tree hypothesis at the \( c \)-th branch

\( \text{tree} = (\text{root, sub-trees}), \) just like what your data structure instructor would say :-)

Hsuan-Tien Lin (NTU CSIE)
Disclaimers about Decision Tree

Usefulness
- human-explainable: widely used in business/medical data analysis
- simple: even freshmen can implement one :-)
- efficient in prediction and training

However......
- heuristic: mostly little theoretical explanations
- heuristics: ‘heuristics selection’ confusing to beginners
- arguably no single representative algorithm

decision tree: mostly heuristic but useful on its own
The following C-like code can be viewed as a decision tree of three leaves.

```c
if (income > 100000) return true;
else {
    if (debt > 50000) return false;
    else return true;
}
```

What is the output of the tree for \((\text{income}, \text{debt}) = (98765, 56789)\)?

1. true
2. false
3. 98765
4. 56789
The following C-like code can be viewed as a decision tree of three leaves.

```c
if (income > 100000) return true;
else {
    if (debt > 50000) return false;
    else return true;
}
```

What is the output of the tree for \((\text{income}, \text{debt}) = (98765, 56789)\)?

- 1. true
- 2. false
- 3. 98765
- 4. 56789

Reference Answer: 2

You can simply trace the code. The tree expresses a complicated boolean condition \([\text{income} > 100000 \text{ or } \text{debt} \leq 50000]\).
A Basic Decision Tree Algorithm

\[ G(\mathbf{x}) = \sum_{c=1}^{C} \left[ b(\mathbf{x}) = c \right] G_c(\mathbf{x}) \]

function DecisionTree(data \( \mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^{N} \))

if termination criteria met

return base hypothesis \( g_t(\mathbf{x}) \)

else

1. learn branching criteria \( b(\mathbf{x}) \)
2. split \( \mathcal{D} \) to \( C \) parts \( \mathcal{D}_c = \{(\mathbf{x}_n, y_n) : b(\mathbf{x}_n) = c\} \)
3. build sub-tree \( G_c \leftarrow \) DecisionTree(\( \mathcal{D}_c \))
4. return \( G(\mathbf{x}) = \sum_{c=1}^{C} \left[ b(\mathbf{x}) = c \right] G_c(\mathbf{x}) \)

four choices: number of branches, branching criteria, termination criteria, & base hypothesis
function DecisionTree(data $D = \{(x_n, y_n)\}_{n=1}^{N}$)

if termination criteria met
    return base hypothesis $g_t(x)$
else ...

2 split $D$ to $C$ parts $D_c = \{(x_n, y_n) : b(x_n) = c\}$

two simple choices

- $C = 2$ (binary tree)
- $g_t(x) = E_{in}$-optimal constant
  - binary/multiclass classification (0/1 error): majority of $\{y_n\}$
  - regression (squared error): average of $\{y_n\}$

disclaimer:
**C&RT** here is based on selected components of **CART** TM of California Statistical Software
function DecisionTree(data \( D = \{(x_n, y_n)\}_{n=1}^{N} \))

if termination criteria met
    return base hypothesis \( g_t(x) = E_{\text{in}}\)-optimal constant
else ...

1. learn branching criteria \( b(x) \)
2. split \( D \) to 2 parts \( D_c = \{(x_n, y_n) : b(x_n) = c\} \)

more simple choices

- simple internal node for \( C = 2 \): \( \{1, 2\}\)-output decision stump
- ‘easier’ sub-tree: branch by purifying

\[
  b(x) = \underset{\text{decision stumps } h(x)}{\text{argmin}} \sum_{c=1}^{2} |D_c \text{ with } h| \cdot \text{impurity}(D_c \text{ with } h)
\]

**C&RT:** bi-branching by purifying
by $E_{in}$ of optimal constant

- **regression error:**
  \[
  \text{impurity}(D) = \frac{1}{N} \sum_{n=1}^{N} (y_n - \bar{y})^2
  \]
  with $\bar{y} = \text{average of } \{y_n\}$

- **classification error:**
  \[
  \text{impurity}(D) = \frac{1}{N} \sum_{n=1}^{N} \mathbb{I}[y_n \neq y^*]
  \]
  with $y^* = \text{majority of } \{y_n\}$

for classification

- **Gini index:**
  \[
  1 - \sum_{k=1}^{K} \left( \frac{\sum_{n=1}^{N} \mathbb{I}[y_n = k]}{N} \right)^2
  \]
  —all $k$ considered together

- **classification error:**
  \[
  1 - \max_{1 \leq k \leq K} \frac{\sum_{n=1}^{N} \mathbb{I}[y_n = k]}{N}
  \]
  —optimal $k = y^*$ only

**popular choices:** Gini for classification, regression error for regression
Termination in C&RT

```
function DecisionTree(data D = {(x_n, y_n) \forall n = 1}^N_{n=1})
if termination criteria met
    return base hypothesis \( g_t(x) = E_{in}-optimal \) constant
else ...
    1) learn branching criteria

\[ b(x) = \arg\min_{\text{decision stumps } h(x)} \sum_{c=1}^{2} |D_c \text{ with } h| \cdot \text{impurity}(D_c \text{ with } h) \]

‘forced’ to terminate when
- all \( y_n \) the same: \( \text{impurity} = 0 \Rightarrow g_t(x) = y_n \)
- all \( x_n \) the same: no decision stumps

C&RT: fully-grown tree with constant leaves that come from bi-branching by purifying
```
For the Gini index, \(1 - \sum_{k=1}^{K} \left( \frac{\sum_{n=1}^{N} [y_n=k]}{N} \right)^2\). Consider \(K = 2\), and let \(\mu = \frac{N_1}{N}\), where \(N_1\) is the number of examples with \(y_n = 1\). Which of the following formula of \(\mu\) equals the Gini index in this case?

1. \(2\mu(1 - \mu)\)
2. \(2\mu^2(1 - \mu)\)
3. \(2\mu(1 - \mu)^2\)
4. \(2\mu^2(1 - \mu)^2\)
For the Gini index, \( 1 - \sum_{k=1}^{K} \left( \frac{\sum_{n=1}^{N} [y_n=k]}{N} \right)^2 \). Consider \( K = 2 \), and let \( \mu = \frac{N_1}{N} \), where \( N_1 \) is the number of examples with \( y_n = 1 \). Which of the following formula of \( \mu \) equals the Gini index in this case?

1. \( 2\mu(1 - \mu) \)
2. \( 2\mu^2(1 - \mu) \)
3. \( 2\mu(1 - \mu)^2 \)
4. \( 2\mu^2(1 - \mu)^2 \)

Reference Answer: 1

Simplify \( 1 - (\mu^2 + (1 - \mu)^2) \) and the answer should pop up.
Basic C&RT Algorithm

function DecisionTree(data $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^N$)
if cannot branch anymore
    return $g_t(x) = E_{in}$-optimal constant
else
    1. learn branching criteria

    $$b(x) = \arg\min_{\text{decision stumps } h(x)} \sum_{c=1}^{2} |\mathcal{D}_c \text{ with } h| \cdot \text{impurity}(\mathcal{D}_c \text{ with } h)$$

    2. split $\mathcal{D}$ to 2 parts $\mathcal{D}_c = \{(x_n, y_n) : b(x_n) = c\}$

    3. build sub-tree $G_c \leftarrow \text{DecisionTree}(\mathcal{D}_c)$

    4. return $G(x) = \sum_{c=1}^{2} \left[ b(x) = c \right] G_c(x)$

easily handle binary classification, regression, & multi-class classification
Regularization by Pruning

- fully-grown tree: \( E_{\text{in}}(G) = 0 \) if all \( x_n \) different but overfit (large \( E_{\text{out}} \)) because low-level trees built with small \( D_c \)

- need a regularizer, say, \( \Omega(G) = \text{NumberOfLeaves}(G) \)
- want regularized decision tree:

\[
\arg\min_{\text{all possible } G} \ E_{\text{in}}(G) + \lambda \Omega(G)
\]

—called pruned decision tree

- cannot enumerate all possible \( G \) computationally:
  —often consider only
  - \( G^{(0)} = \) fully-grown tree
  - \( G^{(i)} = \arg\min_G E_{\text{in}}(G) \) such that \( G \) is one-leaf removed from \( G^{(i-1)} \)

systematic choice of \( \lambda \)? validation
# Branching on Categorical Features

### Numerical Features

| Blood Pressure | 130, 98, 115, 147, 120 |

### Categorical Features

| Major Symptom | Fever, pain, tired, sweaty |

### Branching for Numerical Features

Decision Stump:

\[
 b(x) = \left\lfloor x_i \leq \theta \right\rfloor + 1
\]

with \( \theta \in \mathbb{R} \)

### Branching for Categorical Features

Decision Subset:

\[
 b(x) = \left\lfloor x_i \in S \right\rfloor + 1
\]

with \( S \subset \{1, 2, \ldots, K\} \)

### C&RT (and general decision trees)

Handles **categorical features easily**
Missing Features by Surrogate Branch

possible \( b(\mathbf{x}) = \left[ \text{weight} \leq 50\text{kg} \right] \)

if \textbf{weight} missing during prediction:

- what would human do?
  - go get \textbf{weight}
  - or, use threshold on height instead, because threshold on height \( \approx \) threshold on weight

- surrogate branch:
  - maintain surrogate branch \( b_1(\mathbf{x}), b_2(\mathbf{x}), \ldots \approx \) best branch \( b(\mathbf{x}) \) during training
  - allow \textbf{missing feature} for \( b(\mathbf{x}) \) during prediction by using \textbf{surrogate} instead

\textbf{C\&RT}: handles \textbf{missing features easily}
For a categorical branching criteria $b(x) = \left[ x_i \in S \right] + 1$ with $S = \{1, 6\}$. Which of the following is the explanation of the criteria?

1. if $i$-th feature is of type 1 or type 6, branch to first sub-tree; else branch to second sub-tree
2. if $i$-th feature is of type 1 or type 6, branch to second sub-tree; else branch to first sub-tree
3. if $i$-th feature is of type 1 and type 6, branch to second sub-tree; else branch to first sub-tree
4. if $i$-th feature is of type 1 and type 6, branch to first sub-tree; else branch to second sub-tree
For a categorical branching criteria $b(x) = \left\lfloor x_i \in S \right\rfloor + 1$ with $S = \{1, 6\}$. Which of the following is the explanation of the criteria?

1. if $i$-th feature is of type 1 or type 6, branch to first sub-tree; else branch to second sub-tree
2. if $i$-th feature is of type 1 or type 6, branch to second sub-tree; else branch to first sub-tree
3. if $i$-th feature is of type 1 and type 6, branch to second sub-tree; else branch to first sub-tree
4. if $i$-th feature is of type 1 and type 6, branch to first sub-tree; else branch to second sub-tree

Reference Answer: 2

Note that ‘$\in S$’ is an ‘or’-style condition on the elements of $S$ in human language.
A Simple Data Set
A Simple Data Set
A Simple Data Set
A Simple Data Set
A Simple Data Set
A Simple Data Set
A Simple Data Set
A Simple Data Set
A Simple Data Set
A Simple Data Set
A Simple Data Set

C&RT

AdaBoost-Stump

C&RT: ‘divide-and-conquer’
A Complicated Data Set

C&RT

AdaBoost-Stump

C&RT: even more efficient than AdaBoost-Stump
Practical Specialties of C&RT

- human-explainable
- multiclass easily
- categorical features easily
- missing features easily
- efficient non-linear training (and testing)

—almost no other learning model share all such specialties, except for other decision trees

another popular decision tree algorithm: C4.5, with different choices of heuristics
Which of the following is **not** a specialty of C&RT without pruning?

1. handles missing features easily
2. produces explainable hypotheses
3. achieves low $E_{in}$
4. achieves low $E_{out}$

Reference Answer:

The first two choices are easy; the third comes from the fact that fully grown C&RT greedy minimizes $E_{in}$ (almost always to 0). But as you may imagine, overfitting may happen and $E_{out}$ may not always be low.

Hsuan-Tien Lin (NTU CSIE)

Machine Learning Techniques
Which of the following is **not** a specialty of C&RT without pruning?

1. handles missing features easily
2. produces explainable hypotheses
3. achieves low $E_{in}$
4. achieves low $E_{out}$

Reference Answer: 4

The first two choices are easy; the third comes from the fact that fully grown C&RT greedy minimizes $E_{in}$ (almost always to 0). But as you may imagine, overfitting may happen and $E_{out}$ may not always be low.
Summary

1. Embedding Numerous Features: Kernel Models
2. Combining Predictive Features: Aggregation Models

Lecture 9: Decision Tree

- Decision Tree Hypothesis
  express path-conditional aggregation
- Decision Tree Algorithm
  recursive branching until termination to base
- Decision Tree Heuristics in C&RT
  pruning, categorical branching, surrogate
- Decision Tree in Action
  explainable and efficient

- next: aggregation of aggregation?!