Chapter ML:III

III. Decision Trees

- Decision Trees Basics
- Impurity Functions
- Decision Tree Algorithms
- Decision Tree Pruning
Characterization of the model (model world):  
- $X$ is a set of feature vectors, also called feature space.
- $C$ is a set of classes.
- $c : X \to C$ is the ideal classifier for $X$.
- $D = \{ (x_1, c(x_1)), \ldots, (x_n, c(x_n)) \} \subseteq X \times C$ is a set of examples.

Task: Based on $D$, construction of a decision tree $T$ to approximate $c$. 
Characterization of the model (model world) :

- $X$ is a set of feature vectors, also called feature space.
- $C$ is a set of classes.
- $c : X \rightarrow C$ is the ideal classifier for $X$.
- $D = \{(x_1, c(x_1)), \ldots, (x_n, c(x_n))\} \subseteq X \times C$ is a set of examples.

Task: Based on $D$, construction of a decision tree $T$ to approximate $c$.

Characteristics of the ID3 algorithm:

1. Each splitting is based on one nominal feature and considers its complete domain. Splitting based on feature $A$ with domain $\{a_1, \ldots, a_k\}$:

   $$X = \{x \in X : x|_A = a_1\} \cup \ldots \cup \{x \in X : x|_A = a_k\}$$

2. Splitting criterion is the information gain.
Decision Tree Algorithms

ID3 Algorithm  [Mitchell 1997]  [algorithm template]

ID3(D, Attributes, Target)

1. Create a node t for the tree.

2. Label t with the most common value of Target in D.

3. If all examples in D are positive, return the single-node tree t, with label “+”.
   If all examples in D are negative, return the single-node tree t, with label “–”.

4. If Attributes is empty, return the single-node tree t.

   Otherwise:

5. Let A* be the attribute from Attributes that best classifies examples in D.
   Assign t the decision attribute A*.

6. For each possible value “a” in A* do:

   Add a new tree branch below t, corresponding to the test A* = “a”.

   Let D_a be the subset of D that has value “a” for A*.

   If D_a is empty:
       Then add a leaf node with label of the most common value of Target in D.
   Else add the subtree ID3(D_a, Attributes \ {A*}, Target).

7. Return t.
ID3 Algorithm (pseudo code)

```plaintext
ID3(D, Attributes, Target)

1. \( t = createNode() \)
2. \( label(t) = mostCommonClass(D, Target) \)
3. IF \( \forall \langle x, c(x) \rangle \in D : c(x) = c \) THEN return\( (t) \) ENDIF
4. IF \( Attributes = \emptyset \) THEN return\( (t) \) ENDIF
5. 
6. 
7. 
```
ID3 Algorithm (pseudo code)  [algorithm template]

ID3($D$, $Attributes$, $Target$)

1. $t = createNode()$
2. $label(t) = mostCommonClass(D, Target)$
3. IF $\forall \langle x, c(x) \rangle \in D : c(x) = c$ THEN $return(t)$ ENDIF
4. IF $Attributes = \emptyset$ THEN $return(t)$ ENDIF
5. $A^* = \arg\max_{A \in Attributes}(informationGain(D, A))$
6. 
7. 

ID3 Algorithm (pseudo code) [algorithm template]

**ID3( D, Attributes, Target)**

1. \( t = \text{createNode}(\) 
2. \( \text{label}(t) = \text{mostCommonClass}(D, \text{Target}) \)
3. \( \text{IF} \ \forall(x, c(x)) \in D : c(x) = c \ \text{THEN} \ \text{return}(t) \ \text{ENDIF} \)
4. \( \text{IF} \ \text{Attributes} = \emptyset \ \text{THEN} \ \text{return}(t) \ \text{ENDIF} \)
5. \( A^* = \text{argmax}_{A \in \text{Attributes}}(\text{informationGain}(D, A)) \)
6. \( \text{FOREACH} \ a \in A^* \ \text{DO} \)
   
   \[ D_a = \{(x, c(x)) \in D : x|_{A^*} = a\} \]
   
   \( \text{IF} \ D_a = \emptyset \ \text{THEN} \)

   \[
   \text{ELSE} \\
   \text{createEdge}(t, a, \text{ID3}(D_a, \text{Attributes} \setminus \{A^*\}, \text{Target})) \\
   \text{ENDIF} \\
   \text{ENDDO} \)
7. \( \text{return}(t) \)
Decision Tree Algorithms

ID3 Algorithm (pseudo code)  [algorithm template]

\[ ID3(D, Attributes, Target) \]

1. \( t = createNode() \)
2. \( \text{label}(t) = \text{mostCommonClass}(D, \text{Target}) \)
3. \( \text{IF } \forall \langle x, c(x) \rangle \in D : c(x) = c \text{ THEN } \text{return}(t) \text{ ENDIF} \)
4. \( \text{IF } \text{Attributes} = \emptyset \text{ THEN } \text{return}(t) \text{ ENDIF} \)
5. \( A^* = \text{argmax}_{A \in \text{Attributes}}(\text{informationGain}(D, A)) \)
6. \( \text{FOREACH } a \in A^* \text{ DO} \)
   \( D_a = \{(x, c(x)) \in D : x|_{A^*} = a\} \)
   \( \text{IF } D_a = \emptyset \text{ THEN} \)
   \( t' = createNode() \)
   \( \text{label}(t') = \text{mostCommonClass}(D, \text{Target}) \)
   \( \text{createEdge}(t, a, t') \)
   \( \text{ELSE} \)
   \( \text{createEdge}(t, a, ID3(D_a, \text{Attributes} \setminus \{A^*\}, \text{Target})) \)
   \( \text{ENDIF} \)

7. \( \text{return}(t) \)
Remarks:

- “Target” designates the feature (= attribute) that is comprised of the labels according to which an example can be classified. Within Mitchell’s algorithm the respective class labels are ‘+’ and ‘−’, modeling the binary classification situation. In the pseudo code version, Target may contain multiple (more than two) classes.

- Step 3 of the ID3 algorithm checks the purity of $D$ and, given this case, assigns the unique class $c$, $c \in \text{dom}(\text{Target})$, as label to the respective node.
Decision Tree Algorithms

ID3 Algorithm: Example

Example set $D$ for mushrooms, implicitly defining a feature space $X$ over the three dimensions color, size, and points:

<table>
<thead>
<tr>
<th>Color</th>
<th>Size</th>
<th>Points</th>
<th>Eatability</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>red</td>
<td>small</td>
<td>yes</td>
</tr>
<tr>
<td>2</td>
<td>brown</td>
<td>small</td>
<td>no</td>
</tr>
<tr>
<td>3</td>
<td>brown</td>
<td>large</td>
<td>yes</td>
</tr>
<tr>
<td>4</td>
<td>green</td>
<td>small</td>
<td>no</td>
</tr>
<tr>
<td>5</td>
<td>red</td>
<td>large</td>
<td>no</td>
</tr>
</tbody>
</table>
Top-level call of ID3. Analyze a splitting with regard to the feature “color”:

\[
D_{\text{color}} = \begin{array}{c|cc}
\text{toxic} & \text{eatable} \\
\hline
\text{red} & 1 & 1 \\
\text{brown} & 0 & 2 \\
\text{green} & 0 & 1 \\
\end{array}
\]

\[\Rightarrow |D_{\text{red}}| = 2, |D_{\text{brown}}| = 2, |D_{\text{green}}| = 1\]

Estimated a-priori probabilities:

\[
p_{\text{red}} = \frac{2}{5} = 0.4, \quad p_{\text{brown}} = \frac{2}{5} = 0.4, \quad p_{\text{green}} = \frac{1}{5} = 0.2
\]
Top-level call of ID3. Analyze a splitting with regard to the feature “color”:

$$D|_{\text{color}} = \begin{array}{c|cc}
\text{toxic} & \text{eatable} \\
\hline
\text{red} & 1 & 1 \\
\text{brown} & 0 & 2 \\
\text{green} & 0 & 1 \\
\end{array}$$

$$\Rightarrow \quad |D_{\text{red}}| = 2, \ |D_{\text{brown}}| = 2, \ |D_{\text{green}}| = 1$$

Estimated a-priori probabilities:

$$p_{\text{red}} = \frac{2}{5} = 0.4, \quad p_{\text{brown}} = \frac{2}{5} = 0.4, \quad p_{\text{green}} = \frac{1}{5} = 0.2$$

**Conditional entropy** values for all attributes:

$$H(C \mid \text{color}) = -(0.4 \cdot (\frac{1}{2} \log_2 \frac{1}{2} + \frac{1}{2} \log_2 \frac{1}{2}) +$$
$$0.4 \cdot (\frac{0}{2} \log_2 \frac{0}{2} + \frac{2}{2} \log_2 \frac{2}{2}) +$$
$$0.2 \cdot (\frac{0}{1} \log_2 \frac{0}{1} + \frac{1}{1} \log_2 \frac{1}{1})) = 0.4$$

$$H(C \mid \text{size}) \approx 0.55$$

$$H(C \mid \text{points}) = 0.4$$
Remarks:

- The smaller $H(C | \text{feature})$ is, the larger becomes the information gain. Hence, the difference $H(C) - H(C | \text{feature})$ needs not to be computed since $H(C)$ is constant within each recursion step.

- In the example, the information gain in the first recursion step is maximum for the two features “color” and “points”.
Decision Tree Algorithms
ID3 Algorithm: Example (continued)

Decision tree before the first recursion step:

The feature “points” was chosen in Step 5 of the ID3 algorithm.
Decision tree before the second recursion step:

The feature “color” was chosen in Step 5 of the ID3 algorithm.
Final decision tree after second recursion step:

Break of a tie: choosing the class “toxic” for $D_{green}$ in Step 6 of the ID3 algorithm.
Decision Tree Algorithms

ID3 Algorithm: Hypothesis Space

Inductive bias is the rigidity in applying the (little bit of) knowledge learned from a training set for the classification of unseen feature vectors.

Observations:

- Decision tree search happens in the space of all hypotheses.

- To generate a decision tree, the ID3 algorithm needs per branch at most as many decisions as features are given.
Decision Tree Algorithms

ID3 Algorithm: Inductive Bias

Inductive bias is the rigidity in applying the (little bit of) knowledge learned from a training set for the classification of unseen feature vectors.

Observations:

- Decision tree search happens in the space of all hypotheses.
  - The target concept is a member of the hypothesis space.

- To generate a decision tree, the ID3 algorithm needs per branch at most as many decisions as features are given.
  - No backtracking takes place
  - Local optimization of decision trees
Decision Tree Algorithms

ID3 Algorithm: Inductive Bias

Inductive bias is the rigidity in applying the (little bit of) knowledge learned from a training set for the classification of unseen feature vectors.

Observations:

- Decision tree search happens in the space of all hypotheses.
  - The target concept is a member of the hypothesis space.

- To generate a decision tree, the ID3 algorithm needs per branch at most as many decisions as features are given.
  - no backtracking takes place
  - local optimization of decision trees

Where the inductive bias of the ID3 algorithm becomes manifest:

- Small decision trees are preferred.
- Highly discriminative features tend to be closer to the root.

Is this justified?


Remarks:

- Let $A_j$ be the finite domain (the possible values) of feature $A_j$, $j = 1, \ldots, p$, and let $C$ be a set of classes. Then, a hypothesis space $H$ that is comprised of all decision trees corresponds to the set of all functions $h : A_1 \times \ldots \times A_p \rightarrow C$. Typically, $C = \{0, 1\}$.

- The inductive bias of the ID3 algorithm is of a different kind than the inductive bias of the candidate elimination algorithm (version space algorithm):
  1. The underlying hypothesis space $H$ of the candidate elimination algorithm is incomplete. $H$ corresponds to a coarsened view onto the space of all hypotheses since $H$ contains only conjunctions of attribute-value pairs as hypotheses. However, this restricted hypothesis space is searched completely by the candidate elimination algorithm. Keyword: restriction bias
  2. The underlying hypothesis space $H$ of the ID3 algorithm is complete. $H$ corresponds to the set of all discrete functions (from the Cartesian product of the feature domains onto the set of classes) that can be represented in the form of a decision tree. However, this complete hypothesis space is searched incompletely (following a preference). Keyword: preference bias or search bias

- The inductive bias of the ID3 algorithm renders the algorithm robust with respect to noise.
Decision Tree Algorithms

CART Algorithm [Breiman 1984] [ID3 Algorithm]

Characterization of the model (model world) [ML Introduction]:

- $X$ is a set of feature vectors, also called feature space. No restrictions are presumed for the features’ measurement scales.
- $C$ is a set of classes.
- $c : X \rightarrow C$ is the ideal classifier for $X$.
- $D = \{(x_1, c(x_1)), \ldots, (x_n, c(x_n))\} \subseteq X \times C$ is a set of examples.

Task: Based on $D$, construction of a decision tree $T$ to approximate $c$. 
Characterization of the model (model world):

- $X$ is a set of feature vectors, also called feature space. No restrictions are presumed for the features’ measurement scales.
- $C$ is a set of classes.
- $c : X \rightarrow C$ is the ideal classifier for $X$.
- $D = \{(x_1, c(x_1)), \ldots, (x_n, c(x_n))\} \subseteq X \times C$ is a set of examples.

Task: Based on $D$, construction of a decision tree $T$ to approximate $c$.

Characteristics of the CART algorithm:

1. Each splitting is binary and considers one feature at a time.
2. Splitting criterion is the information gain or the Gini index.
1. Let $A$ be a feature with domain $A$. Ensure a finite number of binary splittings for $X$ by applying the following domain partitioning rules:

- If $A$ is nominal, choose $A' \subset A$ such that $0 < |A'| \leq |A \setminus A'|$.
- If $A$ is ordinal, choose $a \in A$ such that $x_{\min} < a < x_{\max}$, where $x_{\min}, x_{\max}$ are the minimum and maximum values of feature $A$ in $D$.
- If $A$ is numeric, choose $a \in A$ such that $a = (x_k + x_l)/2$, where $x_k, x_l$ are consecutive elements in the ordered value list of feature $A$ in $D$. 
Decision Tree Algorithms

CART Algorithm (continued)

1. Let $A$ be a feature with domain $A$. Ensure a finite number of binary splittings for $X$ by applying the following domain partitioning rules:

   - If $A$ is nominal, choose $A' \subset A$ such that $0 < |A'| \leq |A \setminus A'|$.
   - If $A$ is ordinal, choose $a \in A$ such that $x_{\text{min}} < a < x_{\text{max}}$, where $x_{\text{min}}$, $x_{\text{max}}$ are the minimum and maximum values of feature $A$ in $D$.
   - If $A$ is numeric, choose $a \in A$ such that $a = (x_k + x_l)/2$, where $x_k$, $x_l$ are consecutive elements in the ordered value list of feature $A$ in $D$.

2. For node $t$ of a decision tree generate all splittings of the above type.

3. Choose a splitting from the set of splittings that maximizes the impurity reduction $\Delta \iota$:

   $$\Delta \iota(D(t), \{D(t_L), D(t_R)\}) = \iota(t) - \frac{|D_L|}{|D|} \cdot \iota(t_L) - \frac{|D_R|}{|D|} \cdot \iota(t_R),$$

   where $t_L$ and $t_R$ denote the left and right successor of $t$. 

Decision Tree Algorithms
CART Algorithm (continued)

Illustration for two numeric features, i.e., the feature space $X$ corresponds to a two-dimensional plane:

By a sequence of splittings the feature space $X$ is partitioned into rectangles that are parallel to the two axes.