III. Decision Trees

- Decision Trees Basics
- Impurity Functions
- Decision Tree Algorithms
- Decision Tree Pruning
Decision Tree Algorithms

**ID3 Algorithm** [Quinlan 1986] [CART Algorithm]

Setting:

- $X$ is a set of feature vectors.
- $C$ 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.

Todo:

- Approximate $c(x)$, which is implicitly given via $D$, with a decision tree.
Decision Tree Algorithms

ID3 Algorithm [Quinlan 1986] [CART Algorithm]

Setting:

- $X$ is a set of feature vectors.
- $C$ 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.

Todo:

- Approximate $c(x)$, which is implicitly given via $D$, with a decision tree.

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 information gain.
Decision Tree Algorithms

ID3 Algorithm  [Mitchell 1997]  [algorithm template]

ID3(D, Features, 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 Features is empty, return the single-node tree t.
   - Otherwise:
      5. Let A* be the feature from Features that best classifies examples in D.
         Assign t the decision feature 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, Features \ {A*}, Target).
5. Return t.
Decision Tree Algorithms
ID3 Algorithm (pseudo code) [algorithm template]

ID3($D$, $Features$, $Target$)

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 $Features = \emptyset$ THEN $return(t)$ ENDIF
5.
6.
7.
Decision Tree Algorithms

ID3 Algorithm (pseudo code) [algorithm template]

\[\text{ID3}(D, \text{Features}, \text{Target})\]

1. \( t = \text{createNode}() \)
2. \( \text{label}(t) = \text{mostCommonClass}(D, \text{Target}) \)

3. \( \text{IF} \ \forall \langle x, c(x) \rangle \in D : c(x) = c \ \text{THEN} \ return(t) \ \text{ENDIF} \)
4. \( \text{IF} \ \text{Features} = \emptyset \ \text{THEN} \ return(t) \ \text{ENDIF} \)

5. \( A^* = \arg\max_{A \in \text{Features}} (\text{informationGain}(D, A)) \)

6. \( \)

7. \( \)
**Decision Tree Algorithms**

**ID3 Algorithm (pseudo code) [algorithm template]**

**ID3**(*D, Features, 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** $Features = \emptyset$ **THEN** return($t$) **ENDIF**
5. $A^* = \arg\max_{A \in Features}(informationGain(D, A))$
6. **FOREACH** $a \in A^*$ **DO**
   
   $D_a = \{(x, c(x)) \in D : x|_{A^*} = a\}$
   **IF** $D_a = \emptyset$ **THEN**

   **ELSE**
   
   $createEdge(t, a, ID3(D_a, Features \setminus \{A^\}\), Target))$
   **ENDIF**

   **ENDDO**
7. return($t$)
Decision Tree Algorithms

ID3 Algorithm (pseudo code) [algorithm template]

ID3\((D, \text{Features}, \text{Target})\)

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. \(\text{IF } \text{Features} = \emptyset \text{ THEN } \text{return}(t) \text{ ENDIF}\)
5. \(A^* = \arg\max_{A \in \text{Features}} (\text{informationGain}(D, A))\)
6. \(\text{FOREACH } a \in A^* \text{ DO}\)
   \(D_a = \{(x, c(x)) \in D : x|_{A^*} = a\}\)
   \(\text{IF } D_a = \emptyset \text{ THEN}\)
   \(t' = createNode()\)
   \(\text{label}(t') = \text{mostCommonClass}(D, \text{Target})\)
   \(\text{createEdge}(t, a, t')\)
   \(\text{ELSE}\)
   \(\text{createEdge}(t, a, ID3(D_a, \text{Features} \setminus \{A^*\}, \text{Target}))\)
   \(\text{ENDIF}\)
7. \(\text{return}(t)\)
Remarks:

- “Target” designates the class label 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) class labels.

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

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

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

Estimated a-priori probabilities:

\[p_{\text{red}} = \frac{2}{5} = 0.4, \quad p_{\text{brown}} = \frac{2}{5} = 0.4, \quad p_{\text{green}} = \frac{1}{5} = 0.2\]
Decision Tree Algorithms

ID3 Algorithm: Example (continued)

Top-level call of ID3. Analyze a splitting with regard to the feature “color”:

<table>
<thead>
<tr>
<th></th>
<th>toxic</th>
<th>edible</th>
</tr>
</thead>
<tbody>
<tr>
<td>red</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>brown</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>green</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

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

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

Estimated a-priori probabilities:

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

Conditional entropy values for all features:

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

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

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

- The smaller $H(C \mid \text{feature})$ is, the larger becomes the information gain. Hence, the difference $H(C) - H(C \mid \text{feature})$ 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 becomes maximum for the two features “color” and “points”.
Decision Tree Algorithms

ID3 Algorithm: Example (continued)

Decision tree before the first recursion step:

![Decision Tree Diagram]

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.
Decision Tree Algorithms

ID3 Algorithm: Example (continued)

Final decision tree after second recursion step:

```
feature: Points
  yes
  feature: Color
    red
    label: toxic
  green
    label: toxic
  brown
    label: edible
  no
    label: edible
```

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
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.

- 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
  - the decision tree is a result of \( local \) optimization
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
  - the decision tree is a result of local optimization

Where the inductive bias of the ID3 algorithm becomes manifest:

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

Is this justified?
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 $h$, $h : A_1 \times \ldots \times A_p \rightarrow C$. 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 feature-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 regarding noise.
Decision Tree Algorithms

CART Algorithm [Breiman 1984] [ID3 Algorithm]

Setting:

- $X$ is a set of feature vectors. 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.

Todo:

- Approximate $c(x)$, which is implicitly given via $D$, with a decision tree.
Decision Tree Algorithms

CART Algorithm  [Breiman 1984]  [ID3 Algorithm]

Setting:

- $X$ is a set of feature vectors. 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.

Todo:

- Approximate $c(x)$, which is implicitly given via $D$, with a decision tree.

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 $\mathcal{A}$. Ensure a finite number of binary splittings for $X$ by applying the following domain splitting rules:

- If $A$ is nominal, choose $A' \subseteq A$ such that $0 < |A'| \leq |\mathcal{A} \setminus A'|$.

- 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 $D$. 
1. Let $A$ be a feature with domain $A$. Ensure a finite number of **binary splittings** for $X$ by applying the following domain splitting 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)
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 split into rectangles that are parallel to the two axes.