349: Machine Learning

Fall 2024

Decision Trees
Part 2

But how do we build the decision tree using Information Gain?

About ID3

- A recursive, greedy algorithm to build a decision tree
- At each step it picks the best variable to split the data on, and then moves on
- It is "greedy" because it makes the optimal choice at the current step, without considering anything beyond the current step.
- This can lead to trouble, if one needs to consider things beyond a single variable (e.g. multiple variables) when making a choice.

About ID3

Characterization of the model

- X is a set of feature vectors, also called the feature space
- Y is a set of class labels
- $f: X \rightarrow Y$ is the ideal classifier for X
- $D := \{(x_1, f(x_1)), \ldots, (x_n, f(x_n))\}$

Task: Based on D, construct a decision tree T to approximate f.

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 $\{a_1, \ldots, a_k\}$:
2. Splitting criteria is **information gain**.

ID3 - Algorithm

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.

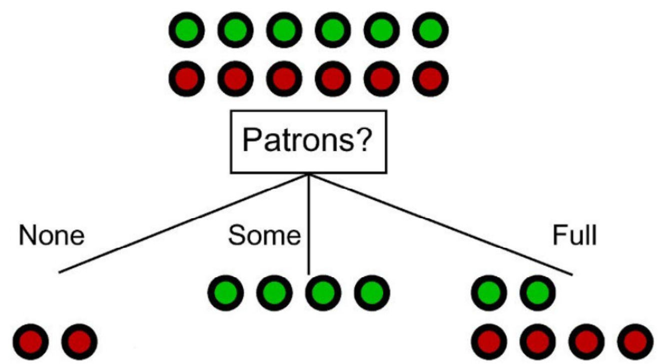
- Otherwise:
  5. Let A* be the attribute from Attributes that best classifies examples in D.

     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).

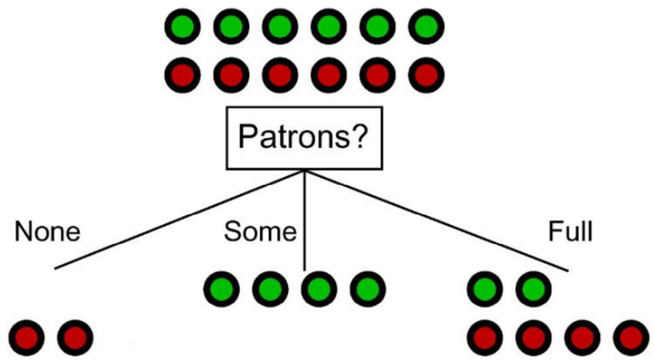
7. Return t.

D_a Example

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	*T*	*F*	*F*	*T*	*Some*	*$$$*	*F*	*T*	*French*	*0–10*	*T*
X_2	*T*	*F*	*F*	*T*	*Full*	*$*	*F*	*F*	*Thai*	*30–60*	*F*
X_3	*F*	*T*	*F*	*F*	*Some*	*$*	*F*	*F*	*Burger*	*0–10*	*T*
X_4	*T*	*F*	*T*	*T*	*Full*	*$*	*F*	*F*	*Thai*	*10–30*	*T*
X_5	*T*	*F*	*T*	*F*	*Full*	*$$$*	*F*	*T*	*French*	*>60*	*F*
X_6	*F*	*T*	*F*	*T*	*Some*	*$$*	*T*	*T*	*Italian*	*0–10*	*T*
X_7	*F*	*T*	*F*	*F*	*None*	*$*	*T*	*F*	*Burger*	*0–10*	*F*
X_8	*F*	*F*	*F*	*T*	*Some*	*$$*	*T*	*T*	*Thai*	*0–10*	*T*
X_9	*F*	*T*	*T*	*F*	*Full*	*$*	*T*	*F*	*Burger*	*>60*	*F*
X_{10}	*T*	*T*	*T*	*T*	*Full*	*$$$*	*F*	*T*	*Italian*	*10–30*	*F*
X_{11}	*F*	*F*	*F*	*F*	*None*	*$*	*F*	*F*	*Thai*	*0–10*	*F*
X_{12}	*T*	*T*	*T*	*T*	*Full*	*$*	*F*	*F*	*Burger*	*30–60*	*T*

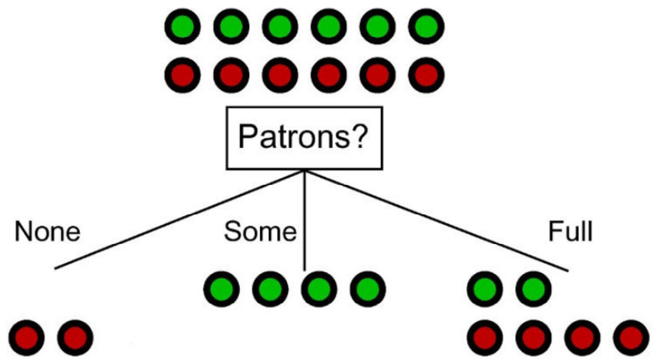


D_a Example



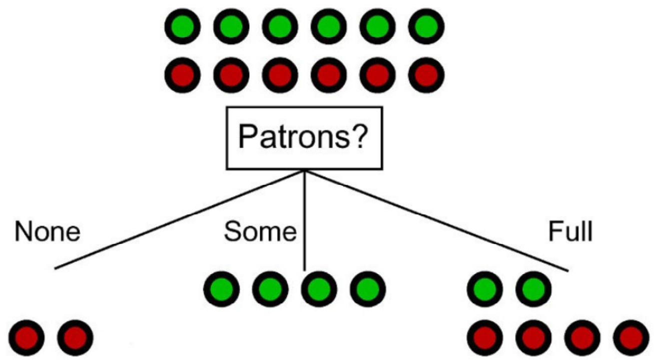
Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

D_a Example



Example	Alt	Bar	Fri	Hun	Pat	Price	Rain	Res	Type	Est	WillWait
	Attributes										Target
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

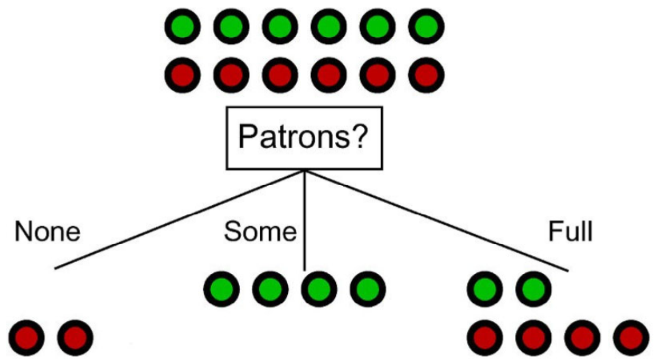
D_a Example



Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

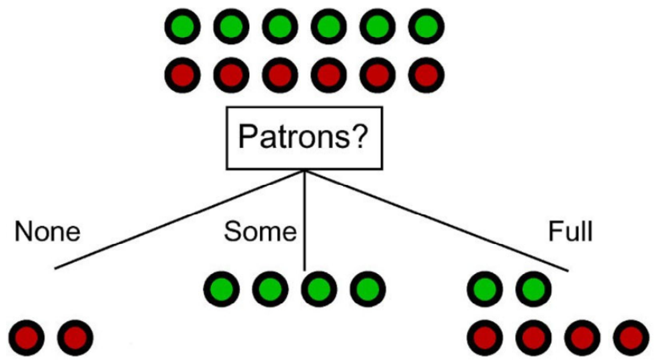
D_a Example



Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

D_a Example

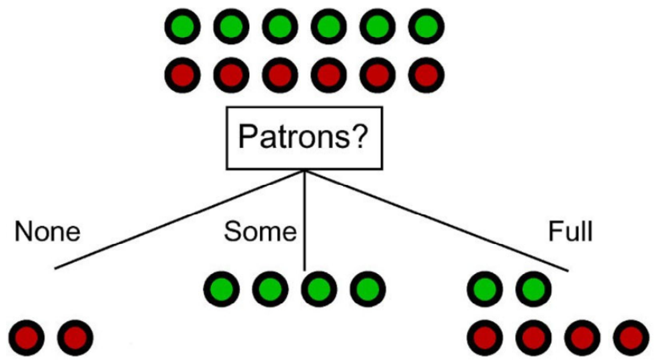


Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

D_a Example



Example	Alt	Bar	Fri	Hun	Pat	Price	Rain	Res	Type	Est	Target WillWait
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

Example	Alt	Bar	Fri	Hun	Pat	Price	Rain	Res	Type	Est	Target WillWait
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

Example	Alt	Bar	Fri	Hun	Pat	Price	Rain	Res	Type	Est	Target WillWait
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

ID3 – Pseudo Code

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

$t =$ *createNode*$()$

label$(t) =$ *mostCommonClass*$(D,$ *Target*$)$

IF $\forall \langle \mathbf{x}, c(\mathbf{x}) \rangle \in D : c(\mathbf{x}) = c$ **THEN** *return*(t) **ENDIF**

IF *Attributes* $= \emptyset$ **THEN** *return*(t) **ENDIF**

ID3 – Pseudo Code

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

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

ID3 – Pseudo Code

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

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

   $D_a = \{(\mathbf{x}, c(\mathbf{x})) \in D : \mathbf{x}|_{A^*} = a\}$

   IF $D_a = \emptyset$ **THEN**

   ELSE

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

   ENDIF

   ENDDO
7. *return*(t)

ID3 – Pseudo Code

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

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

   $D_a = \{(\mathbf{x}, c(\mathbf{x})) \in D : \mathbf{x}|_{A^*} = a\}$

   IF $D_a = \emptyset$ **THEN**

     $t' =$ *createNode*$()$

     label$(t') =$ *mostCommonClass*$(D,$ *Target*$)$

     createEdge(t, a, t')

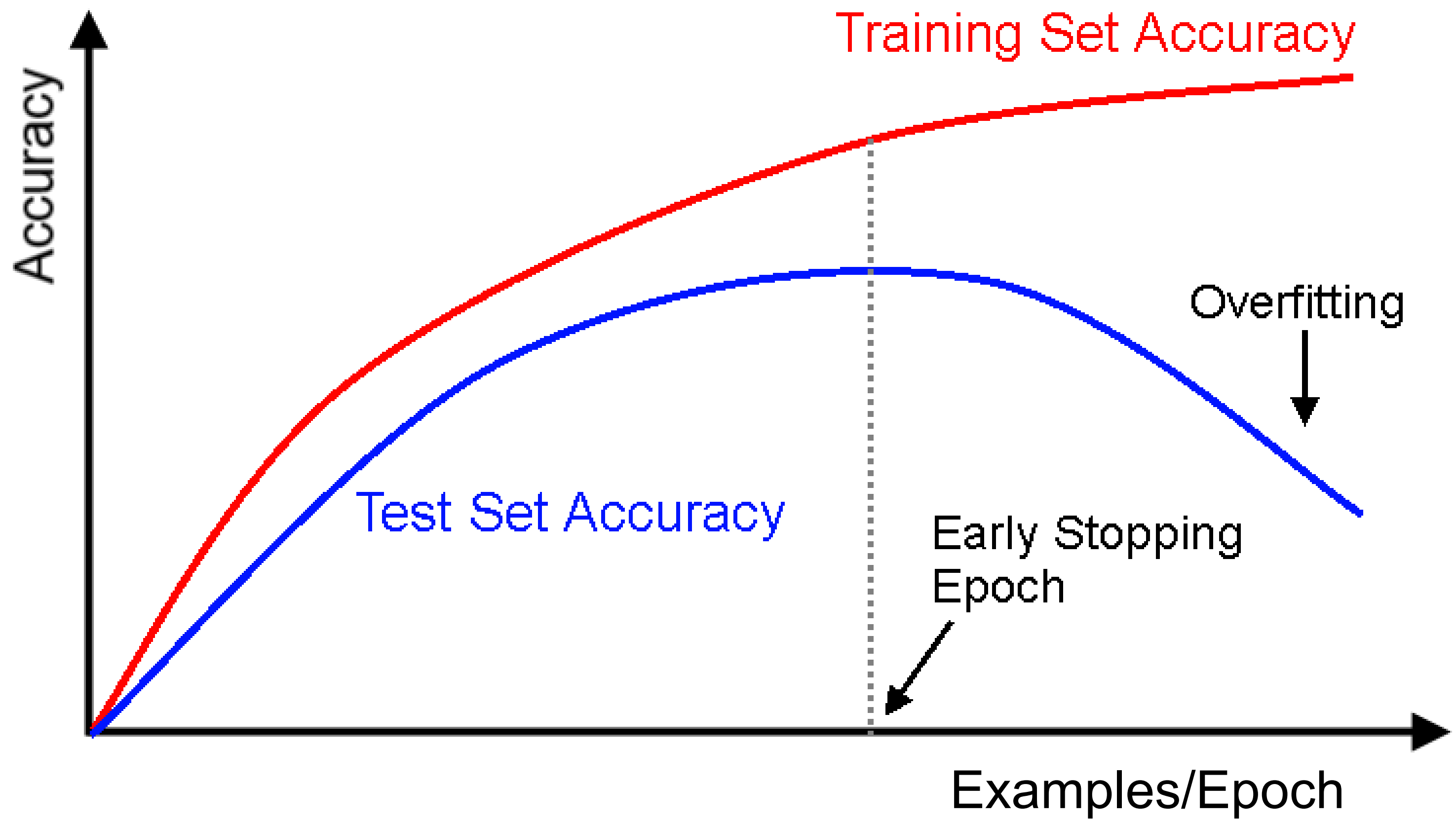
   ELSE

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

   ENDIF

   ENDDO
7. *return*(t)

What the learning curve tells us

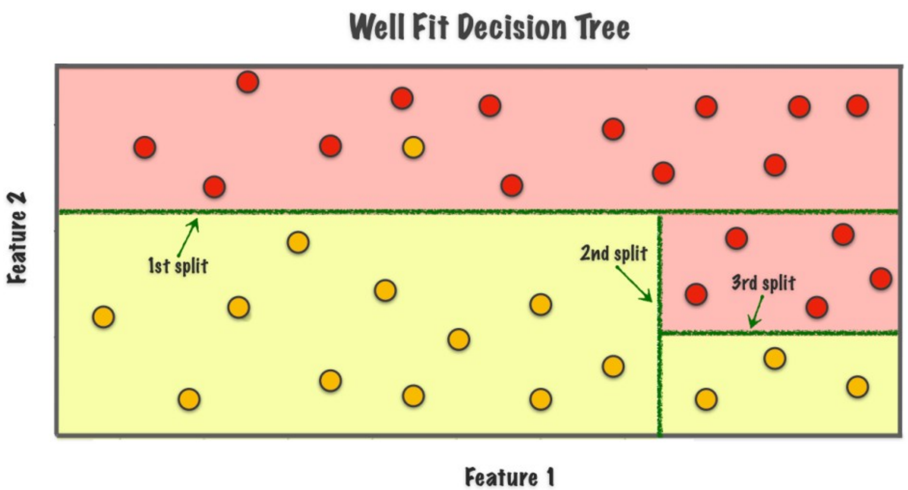


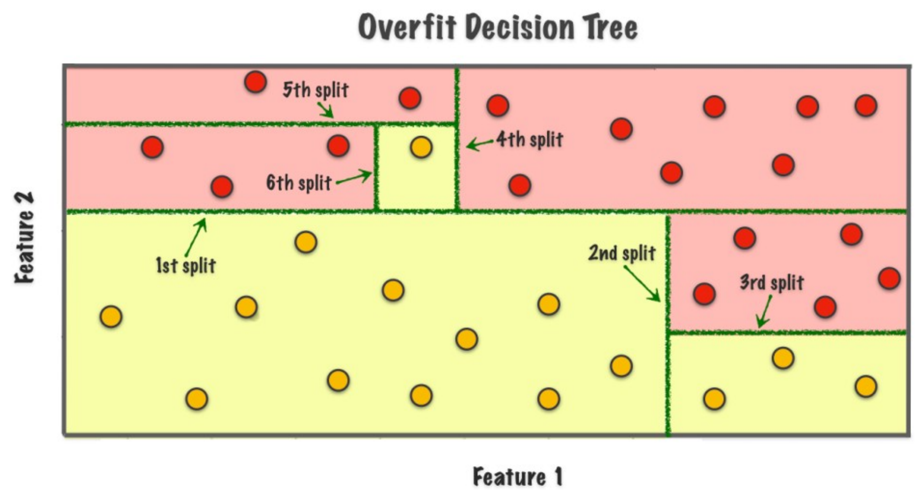
Rule #2 of Machine Learning

The *best* (i.e. the one that generalizes well) hypothesis almost never achieves 100% accuracy on the training data.

(Rule #1 was: you can't learn anything without inductive bias)

Overfitting





Avoid Overfitting

There are multiple approaches to preventing overfitting in decision trees:

1. **Early stopping:** Build the decision tree while applying some criterion that stops the decision trees growth before it overfits to the training data.
2. **Pruning**: Build the decision tree and allow it to overfit to the training data, then prune it back to remove the elements causing overfitting.
3. **Data preprocessing:** Making some changes to the initial data before ever building the tree

Early Stopping

Early stopping (or pre-pruning) is where we stop the growth of a decision tree early before it overfits to the training data.

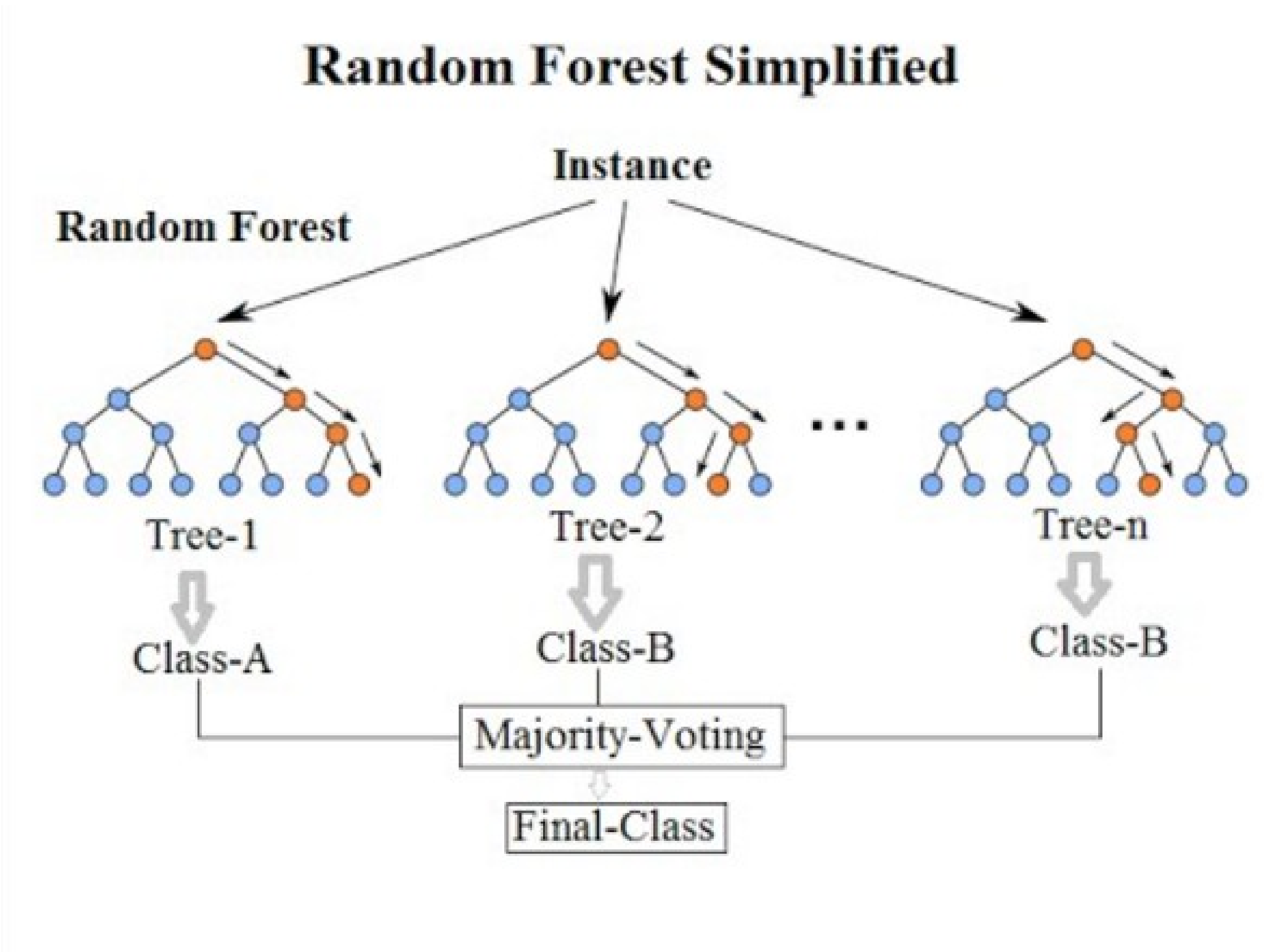
- **Maximum tree depth:** Simply predefine an arbitrary number for the maximum depth (or max number of splits) and once the tree reaches this value the growing process terminates.
- **Minimum number in node:** Define a minimum number of observations to appear in any child node for a split to be valid.
- **Minimum decrease in impurity:** Define a minimum acceptable decrease in impurity for a split to be accepted.
- **Maximum features:** Not strictly speaking a stopping rule but only considering a subset of the available features to split on may improve the final trees generalizability.
- **Validation accuracy:** Stop when model performance on the validation data stops improving

Pruning

Pruning (or post-pruning) takes a tree that has already been overfit and makes some adjustments to reduce/remove the observed overfitting.

- **Critical value pruning:** Retrospectively estimate the strength of each node from calculations done in the tree building stage. Nodes that don't achieve a certain critical value are pruned, unless a node further along the branch does reach it.
- **Error complexity pruning:** Generates a series of trees each made by pruning the full tree by different amounts and selects one of these by assessing its performance with an independent data set.
- **Reduced error pruning:** Runs the independent test data through the full tree and, for each non-leaf node, compares the number of errors if the sub tree from that node is kept vs removed. The pruned node will often make fewer errors using the new test data than the sub tree makes. The node that sees the biggest difference in performance is pruned and this process is continued until further pruning will increase the misclassification rate.

Random Forests



What you need to know about decision trees

Advantages to using decision trees:

1. Easy to interpret and make for straightforward visualizations
2. The internal workings are capable of being observed and thus make it possible to reproduce work
3. Can handle both numerical and categorical data
4. Can be used for classification and regression
5. Perform well on large datasets
6. Are extremely fast

Disadvantages to using decision trees:

1. Require algorithms capable of determining an optimal choice at each node
2. Prone to over-fitting, especially when a tree is particularly deep.

Random forests can be more accurate by reducing bias and variance

ID3 Example

ID3 – Example

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

	Color	Size	Points	Eatability
1	red	small	yes	toxic
2	brown	small	no	eatable
3	brown	large	yes	eatable
4	green	small	no	eatable
5	red	large	no	eatable

ID3 – Example

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

$D|_{\text{color}} =$

	toxic	eatable
red	1	1
brown	0	2
green	0	1

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

Estimated a-priori probabilities:

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

ID3 – Example

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

$$D|_{\text{color}} = \begin{array}{lcc} \hline & \text{toxic} & \text{eatable} \\ \hline \text{red} & 1 & 1 \\ \text{brown} & 0 & 2 \\ \text{green} & 0 & 1 \\ \hline \end{array} \quad \rightarrow \quad |D_{\text{red}}| = 2,\ |D_{\text{brown}}| = 2,\ |D_{\text{green}}| = 1$$

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:

$$\begin{aligned} H(C \mid \text{color}) &= -(\, 0.4 \cdot (\tfrac{1}{2} \cdot \log_2 \tfrac{1}{2} + \tfrac{1}{2} \cdot \log_2 \tfrac{1}{2}) + \\ &\quad\ 0.4 \cdot (\tfrac{0}{2} \cdot \log_2 \tfrac{0}{2} + \tfrac{2}{2} \cdot \log_2 \tfrac{2}{2}) + \\ &\quad\ 0.2 \cdot (\tfrac{0}{1} \cdot \log_2 \tfrac{0}{1} + \tfrac{1}{1} \cdot \log_2 \tfrac{1}{1})\ ) \ = \ 0.4 \end{aligned}$$

$$\begin{aligned} H(C \mid \text{size}) &\approx 0.55 \\ H(C \mid \text{points}) &= 0.4 \end{aligned}$$

ID3 – Example

Decision tree before the first recursion step:

attribute: points

yes

color	size	eatability
red	small	toxic
brown	large	eatable

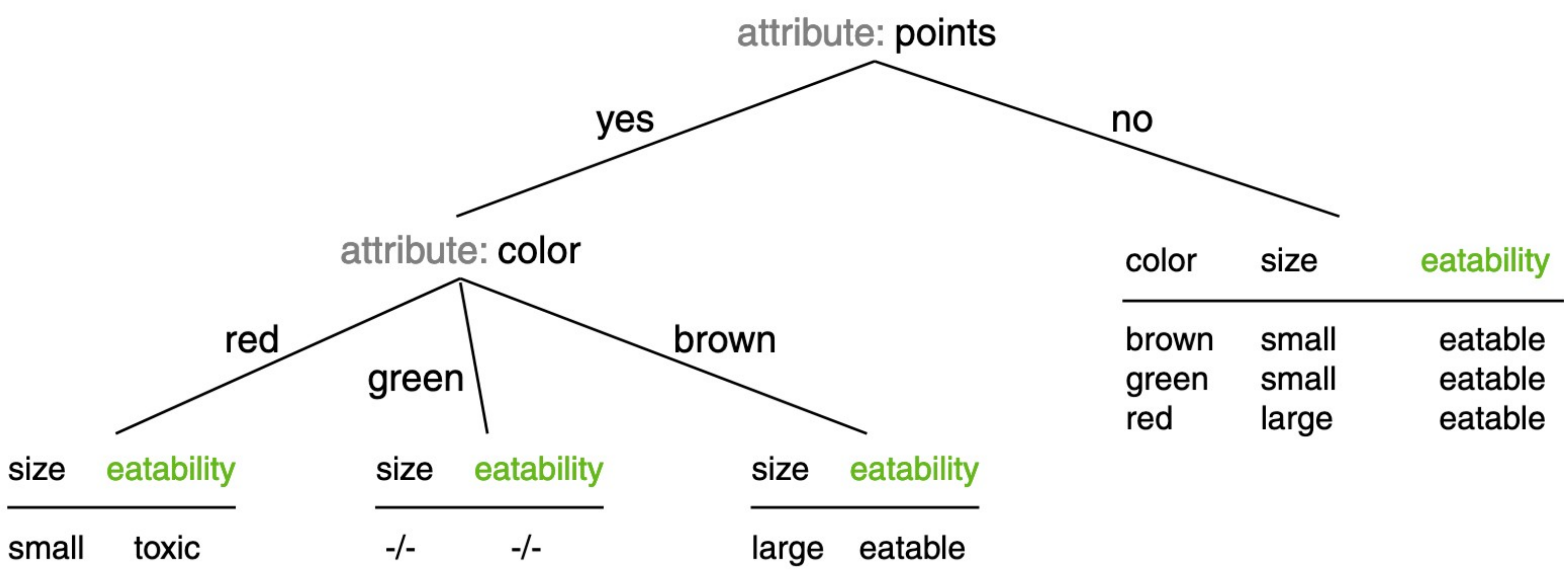
no

color	size	eatability
brown	small	eatable
green	small	eatable
red	large	eatable

The feature “points” was chosen in Step 5 of the ID3 algorithm.

ID3 – Example

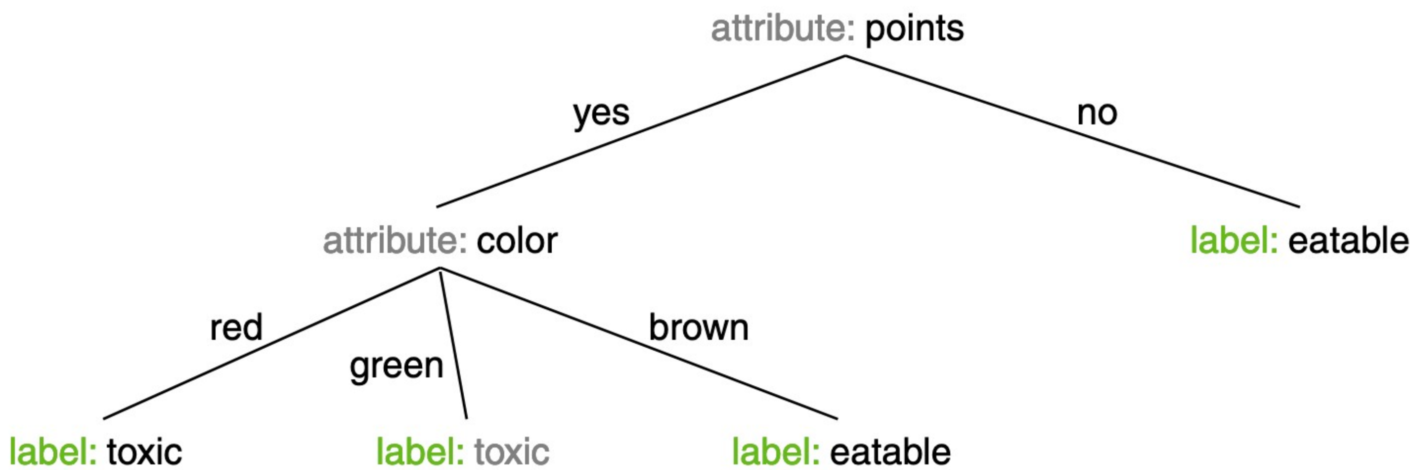
Decision tree before the second recursion step:



The feature “color” was chosen in Step 5 of the ID3 algorithm.

ID3 – Example

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.