Supervised Learning

Chapter VI: Decision Tree Learning 2

Johannes Jurgovsky

Outline

Decision Tree Learning 2

1. Algorithms
2. Pruning
3. Ensemble Learning
4. Random Forest

1. Algorithms

Algorithms:

ID3 Algorithm [Quinlan 1986] [CART Algorithm]

Characterization of the model (model world)

- X is a set of feature vectors, also called feature space.
- Y is a set of classes.
- $y : X \to Y$ is the ideal classifier for X.
- $D = \{(\mathbf{x}_1, y(\mathbf{x}_1)), \ldots, (\mathbf{x}_n, y(\mathbf{x}_n))\} \subseteq X \times Y$ is a set of examples.

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

Algorithms:

ID3 Algorithm [Quinlan 1986] [CART Algorithm]

Characterization of the model (model world)

- X is a set of feature vectors, also called feature space.
- Y is a set of classes.
- $y : X \to Y$ is the ideal classifier for X.
- $D = \{(\mathbf{x}_1, y(\mathbf{x}_1)), \ldots, (\mathbf{x}_n, y(\mathbf{x}_n))\} \subseteq X \times Y$ is a set of examples.

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

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 $\{a_1, \ldots, a_k\}$:

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

2. Splitting criterion is the Information Gain:
$IG(D, A) = \Delta\iota_{entropy}(D, \{D_1, \ldots, D_k\}_A) = \iota_{entropy}(D) - \sum_j^k \frac{|D_j|}{|D|}\iota_{entropy}(D_j)$

Algorithms:

ID3 Algorithm (pseudo code) [algorithm template]

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

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

Algorithms:

ID3 Algorithm (pseudo code) [algorithm template]

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

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

Algorithms:

ID3 Algorithm (pseudo code) [algorithm template]

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

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

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

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

   ELSE

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

   ENDIF

   ENDDO
7. *return*(t)

Decision Tree Algorithms

ID3 Algorithm (pseudo code) [algorithm template]

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

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

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

   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)

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 be comprised of multiple (more than two) classes.
- Step 3 of of the ID3 algorithm checks the purity of D and, given this case, assigns the unique class c, $c \in$ *dom*(*Target*), as label to the respective node.

Algorithms:

ID3 Algorithm: 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

Algorithms:

ID3 Algorithm: Example (continued)

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

$D =$

	toxic	eatable
red	1	1
brown	0	2
green	0	1

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

$$p(\text{color} = \text{red}) = \frac{2}{5}, \quad p(\text{color} = \text{brown}) = \frac{2}{5}, \quad p(\text{color} = \text{green}) = \frac{1}{5}$$

$$p(\text{eatability} = \text{toxic}) = \frac{1}{5}, \quad p(\text{eatability} = \text{eatable}) = \frac{4}{5}$$

Impurity (entropy) of dataset D:

$$\iota(D) = -(\frac{1}{5}\log_2\frac{1}{5} + \frac{4}{5}\log_2\frac{4}{5}) \approx 0.72$$

Impurities (entropy) of subsets $D_{\text{color=red}}, D_{\text{color=brown}}, D_{\text{color=green}}$:

$$\iota(D_{\text{color=red}}) = -(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}) = 1$$

$$\iota(D_{\text{color=brown}}) = -(\frac{0}{2}\log_2\frac{0}{2} + \frac{2}{2}\log_2\frac{2}{2}) = 0$$

$$\iota(D_{\text{color=green}}) = -(\frac{0}{1}\log_2\frac{0}{1} + \frac{1}{1}\log_2\frac{1}{1}) = 0$$

Algorithms:

ID3 Algorithm: Example (continued)

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

$D =$

	toxic	eatable
red	1	1
brown	0	2
green	0	1

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

$$p(\text{color} = \text{red}) = \frac{2}{5}, \quad p(\text{color} = \text{brown}) = \frac{2}{5}, \quad p(\text{color} = \text{green}) = \frac{1}{5}$$

Impurity reductions (Information gain) obtained for attributes color, size and points:

$$\Delta\iota(D, \{D_{\text{red}}, D_{\text{brown}}, D_{\text{green}}\}) = \iota(D) - (\frac{2}{5}\iota(D_{\text{red}}) + \frac{2}{5}\iota(D_{\text{brown}}) + \frac{1}{5}\iota(D_{\text{green}})) \approx 0.72 - 0.4 = 0.32$$

$$\Delta\iota(D, D|_{\text{size}}) \approx 0.72 - 0.55 = 0.17$$
$$\Delta\iota(D, D|_{\text{points}}) \approx 0.72 - 0.4 = 0.32$$

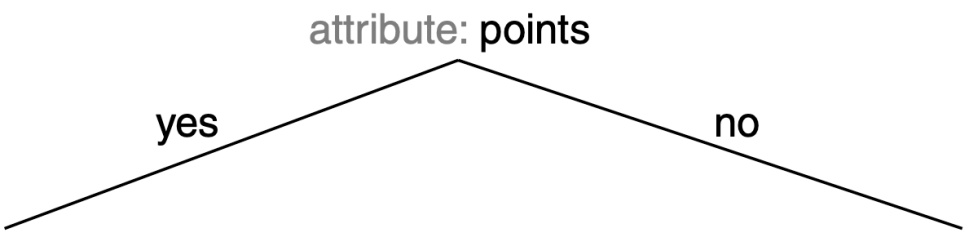
Remarks:

- The smaller the weighted impurity of the subsets, the larger the impurity reduction (Information gain).
- In the example, the information gain in the first recursion step is maximum for the two features “color” and “points”.

Algorithms:

ID3 Algorithm: Example (continued)

Decision tree before the first recursion step:



color	size	eatability
red	small	toxic
brown	large	eatable

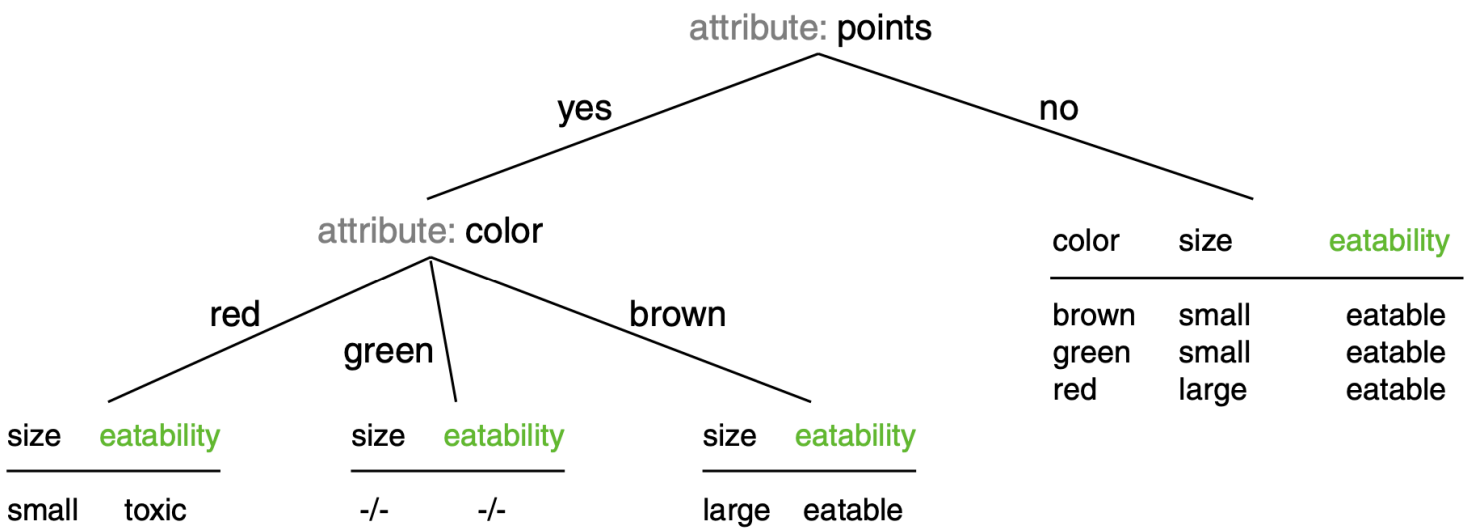
color	size	eatability
brown	small	eatable
green	small	eatable
red	large	eatable

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

Algorithms:

ID3 Algorithm: Example (continued)

Decision tree before the second recursion step:

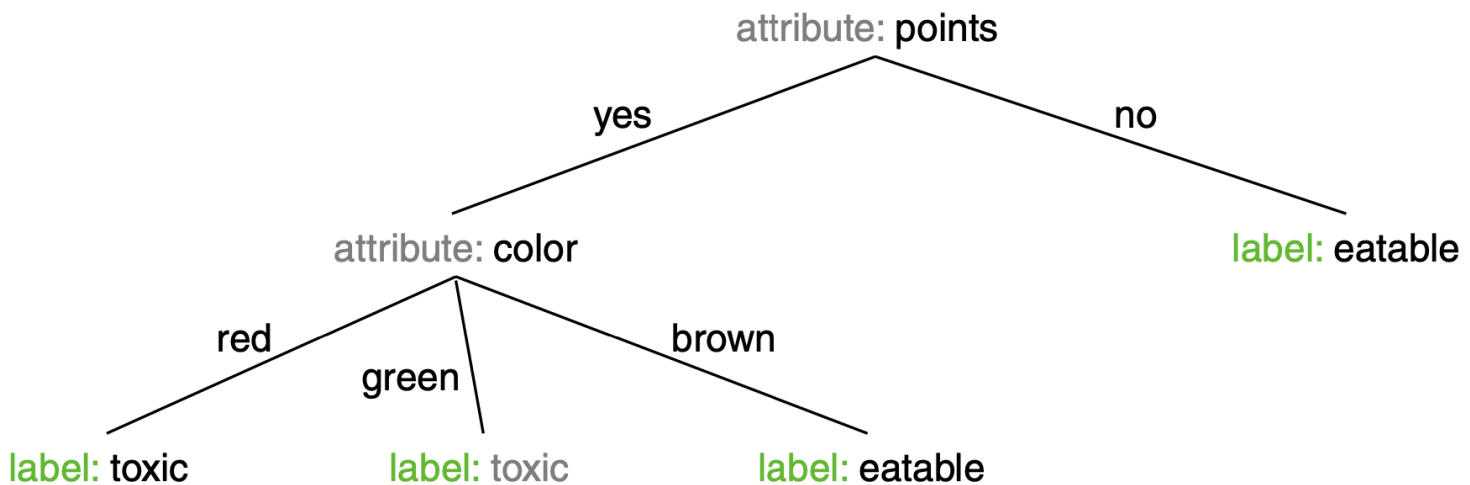


The feature “color” was chosen in Step 5 of the ID3 algorithm.

Algorithms:

ID3 Algorithm: Example (continued)

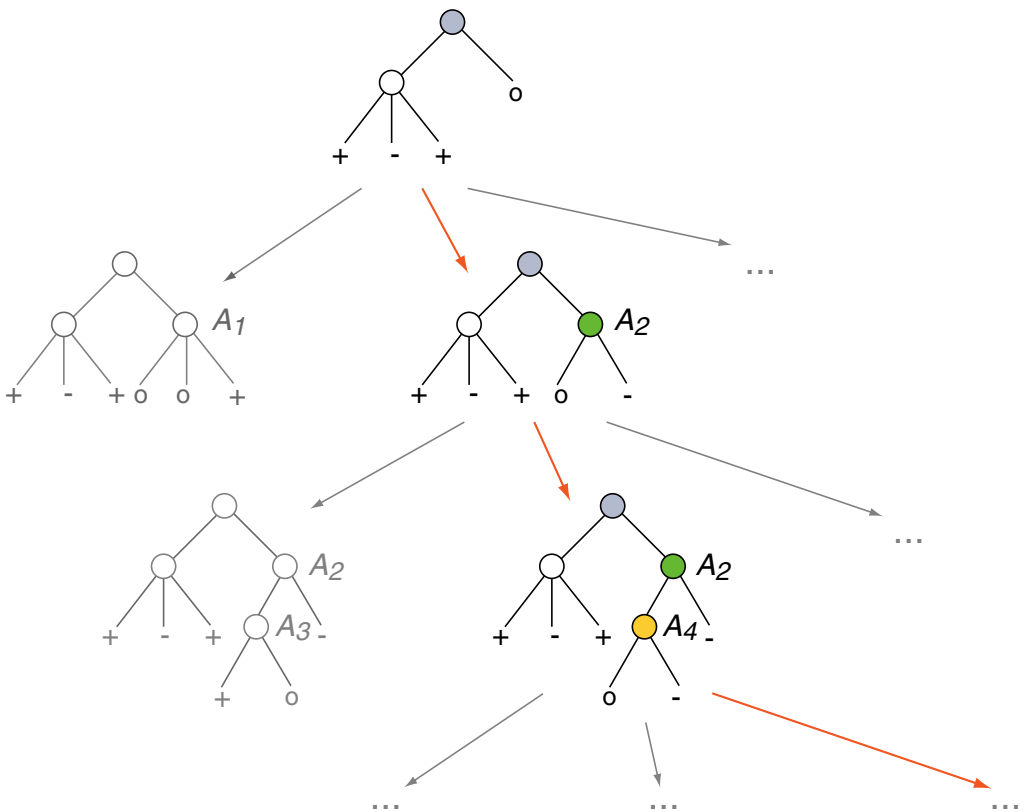
Final decision tree after second recursion step:



Break of a tie: choosing the class "toxic" for D_{green} in Step 6 of the ID3 algorithm.

Algorithms:

ID3 Algorithm: Hypothesis Space



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.

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

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

Inductive bias of the ID3 algorithm:

- Prefers to construct small decision trees (top-down approach)
- Highly discriminative features (with many levels) tend to be closer to the root.

Is this justified?

Algorithms:

CART Algorithm [Breiman 1984] [ID3 Algorithm]

Characterization of the model (model world)

- X is a set of feature vectors, also called feature space. No restrictions are presumed for the measurement scales of the features.
- Y is a set of classes.
- $y : X \to Y$ is the ideal classifier for X.
- $D = \{(\mathbf{x}_1, y(\mathbf{x}_1)), \ldots, (\mathbf{x}_n, y(\mathbf{x}_n))\} \subseteq X \times Y$ is a set of examples.

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

Algorithms:

CART Algorithm [Breiman 1984] [ID3 Algorithm]

Characterization of the model (model world)

- X is a set of feature vectors, also called feature space. No restrictions are presumed for the measurement scales of the features.
- Y is a set of classes.
- $y : X \to Y$ is the ideal classifier for X.
- $D = \{(\mathbf{x}_1, y(\mathbf{x}_1)), \ldots, (\mathbf{x}_n, y(\mathbf{x}_n))\} \subseteq X \times Y$ is a set of examples.

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

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.

Algorithms:

CART Algorithm (continued)

1. Let A be a feature with domain $\mathbf{A}$. Ensure a finite number of binary splittings for X by applying the following domain partitioning rules:
   - If A is nominal, choose a subset $\mathbf{A}'$ from $\mathbf{A}$. The decision rule for a new data point then becomes $\mathbf{x}|_A \in \mathbf{A}'$?
   - If A is ordinal, choose a threshold $a \in \mathbf{A}$ 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. The decision rule for a new data point then becomes $\mathbf{x}|_A \geq a$?

Algorithms:

CART Algorithm (continued)

1. Let A be a feature with domain $\mathbf{A}$. Ensure a finite number of binary splittings for X by applying the following domain partitioning rules:
   - If A is nominal, choose a subset $\mathbf{A}'$ from $\mathbf{A}$. The decision rule for a new data point then becomes $\mathbf{x}|_A \in \mathbf{A}'$?
   - If A is ordinal, choose a threshold $a \in \mathbf{A}$ 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. The decision rule for a new data point then becomes $\mathbf{x}|_A \geq a$?

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 $\Delta\iota$:

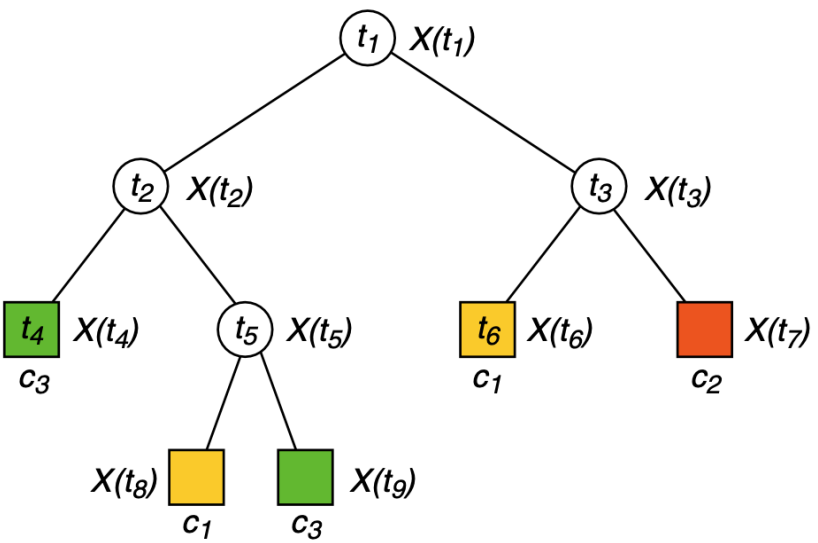
$$\underline{\Delta\iota}(D(t), \{D(t_L), D(t_R)\}) = \iota(t) - \frac{|D_L|}{|D|} \cdot \iota(t_L) - \frac{|D_R|}{|D|} \cdot \iota(t_R),$$

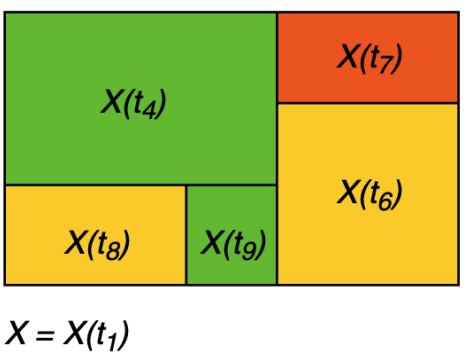
where t_L and t_R denote the left and right successor of t.

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.

2. Pruning

Pruning:
Overfitting

Overfitting: Let D be a set of examples and let H be a hypothesis space. The hypothesis $h \in H$ is said to **overfit** D if an $h' \in H$ exists with:

$$\mathit{Err}(h, D) < \mathit{Err}(h', D) \quad \text{and} \quad \mathit{Err}^*(h) > \mathit{Err}^*(h'),$$

The error $\mathit{Err}^*(h)$ denotes the true misclassification rate of h, while $\mathit{Err}(h, D)$ denotes the error of h on an example set D.

Assessing Overfitting in Practice: The $\mathit{Err}^*(h)$ is estimated with a validation set $D_{vs} \subset D$ that has not been used during training: $D_{vs} \cap D_{tr} = \emptyset$. Estimating the true error using out-of-sample data is called holdout estimation. Thus, the assessment relevant in practice becomes:

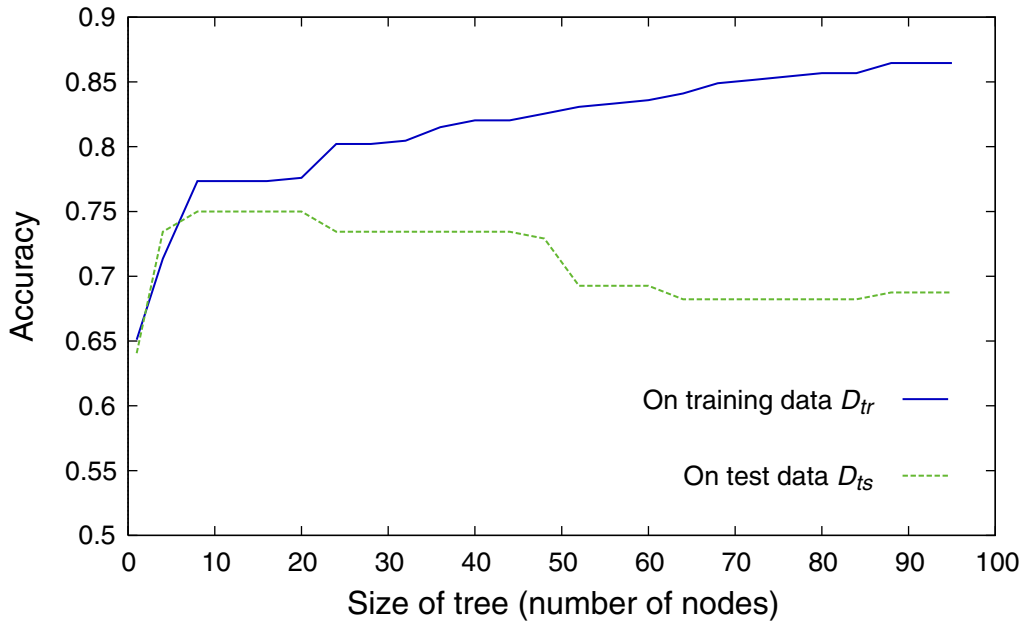
$$\mathit{Err}(h, D_{tr}) < \mathit{Err}(h', D_{tr}) \quad \text{and} \quad \mathit{Err}(h, D_{vs}) > \mathit{Err}(h', D_{vs})$$

Decision Trees are high-Variance models. They tend to overfit if the dataset D is:

- D is noisy The learner fits the noise instead of data points.
- D is biased and hence non-representative
- D is too small The parametrization of the learner is largely unconstrained.

Pruning:

Overfitting (continued)



[Mitchell 1997]

Remarks:

- Accuracy is the percentage of correctly classified examples. That is: 1 - Misclassification rate.
- The terms validation set and test set are often used interchangeably. The important aspect in this figure is that D_{ts} has not been used during training.
- When does the accuracy of a decision tree become One? If there are no mislabeled examples, the accuracy of a decision tree approaches One as the size of the tree increases.

Pruning:

Overfitting (continued)

Approaches to counter overfitting:

1. Stopping of the decision tree construction process *during training*.
2. Pruning of a decision tree *after training*: Reduced Error Pruning
   - Partitioning of D into three sets for training, validation and test.

Pruning:

Stopping

Possible criteria for stopping [splitting criteria]:

1. Size of $D(t)$.
   $D(t)$ will not be partitioned further if the number of examples, $|D(t)|$, is below a certain threshold.

2. Purity of $D(t)$.
   $D(t)$ will not be partitioned further if all induced splittings yield no significant impurity reduction $\Delta\iota$.

Problems:

ad 1) A threshold that is too small results in oversized decision trees.

ad 1) A threshold that is too large omits useful splittings.

ad 2) $\Delta\iota$ cannot be extrapolated with regard to the tree height.

Pruning:

Pruning

The pruning principle:

1. Construct a sufficiently large decision tree T_{max}.
2. Prune T_{max}, starting from the leaf nodes towards the tree root.

Each leaf node t of T_{max} fulfills one or more of the following conditions:

- $D(t)$ is sufficiently small. Typically, $|D(t)| \leq 5$.
- $D(t)$ is comprised of examples of only one class.
- $D(t)$ is comprised of examples with identical feature vectors.

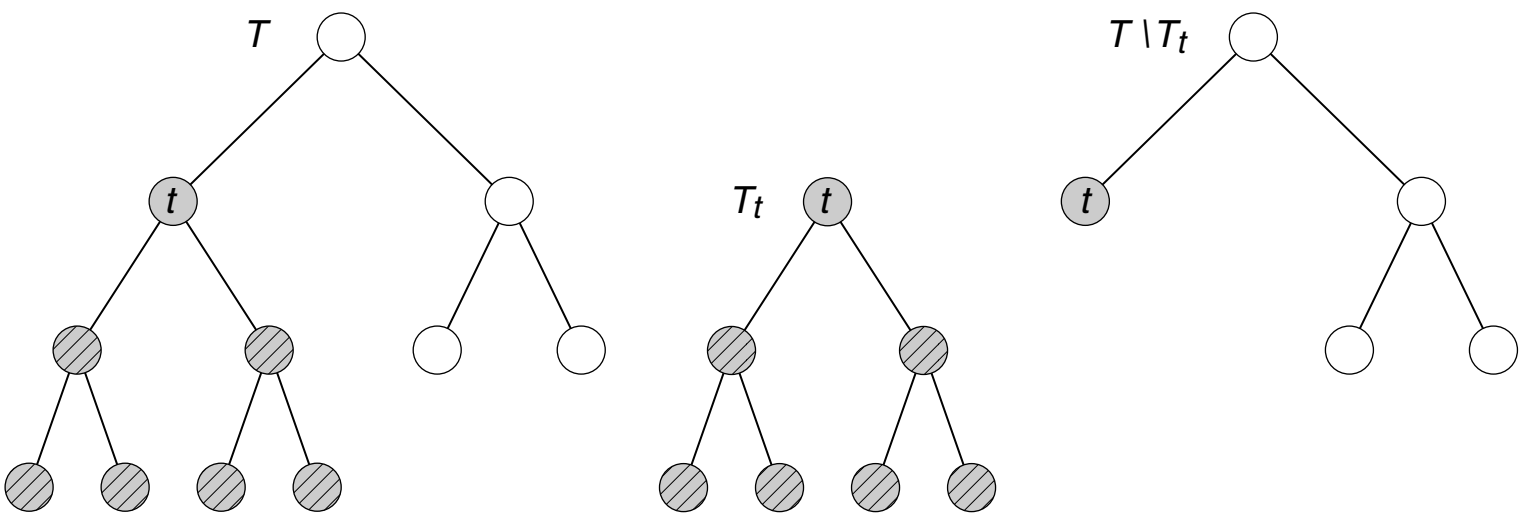
Pruning:

Pruning (continued)

Definition 1 (Decision Tree Pruning)

Given a decision tree T and an inner (non-root, non-leaf) node t. Then pruning of T wrt. t is the deletion of all successor nodes of t in T. The pruned tree is denoted as $T \setminus T_t$. The node t becomes a leaf node in $T \setminus T_t$.

Illustration:



Pruning:

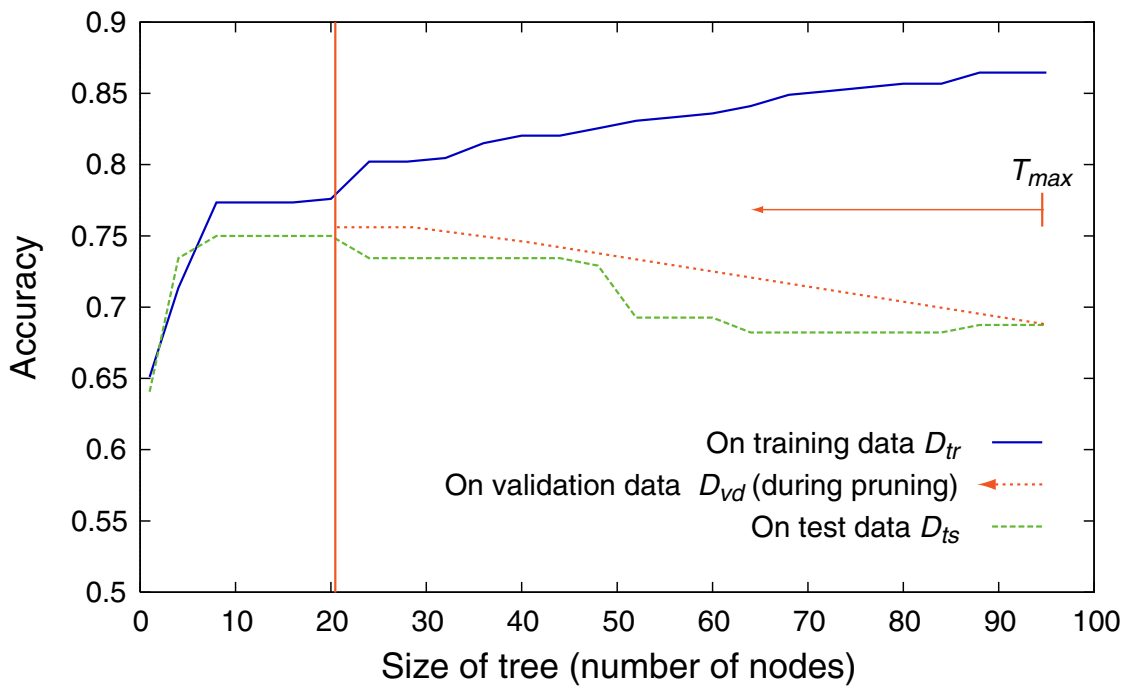
Pruning: Reduced Error Pruning

Reduced error pruning for decision tree T_{max} and validation set D_{vs} :

1. $T = T_{\text{max}}$
2. Choose an inner node t in T.
3. Tentative pruning of T wrt. t : $T' = T \setminus T_t$.
   Based on $D(t)$ assign class to t. [*DT-construct*]
4. If *Err*$(T', D_{vs}) \leq$ *Err*(T, D_{vs}) then accept pruning: $T = T'$.
5. Continue with Step 2 until all inner nodes of T are tested.

Pruning:

Pruning: Reduced Error Pruning (continued)



[Mitchell 1997]

Decision Trees

Extensions

Pruning

- If D is small, its partitioning into three sets for training, validation, and test will discard valuable information for decision tree construction. An alternative pruning strategy used in the C4.5 algorithm is called Rule post pruning which converts the tree into an equivalent set of rules. One rule for each path from a leaf to the root node. Rule post pruning then aims to simplify the resulting disjunction without sacrificing too much accuracy.
- In addition to the accuracy, pruning can also be guided by reducing misclassification costs

Decision Tree Extensions

- To cope with missing values in new data points, one can introduce "surrogate splittings" for each node. A surrogate splitting is an alternative splitting on some other attribute that has the second best information gain. If a new data point has a missing value in the primary attribute, the data point gets treated according to the surrogate splitting.
- Splittings based on (linear) combinations of features: Splittings on single attributes lead to axis-aligned partitions. This restriction can lead to unnecessarily complicated decision boundaries.
- Regression trees: Use mean of target values in leaf node as estimate for the value of the dependent variable.

3. Ensemble Learning

Ensemble Learning: Overview

Ensemble Learning

Motivation for practitioners: High-variance learning algorithms, like decision trees or neural networks, are hard to manage in real-world tasks: Unreliable predictions due to overfitting.

Ensemble Learning builds an ensemble of different predictors and combines their predictions.

It is based on the hypothesis, that predictions from a group of (different) experts outperform the prediction of a single expert.

The individual predictors can be obtained from very different learning algorithms (e.g. Decision Tree, Naive Bayes, Neural Network) or from the same learning algorithms with parameter and data set variations (e.g. Random Forest).

Ensemble Learning: Overview

Ensemble Learning: Variants

Model stacking:

- Use a collection of arbitrary learners (preferably different ones)
- Fit all learners on the training data
- Fit a "combiner" model on the training data using all the predictions of the other models as additional inputs.
- The "combiner" is often a single-layer Logistic Regression classifier.

Boosting:

- Use a collection of weak learners (high-bias/low-variance) such as very shallow decision tress (Decision Stumps)
- Train the learners sequentially
- Let the current learner focus on those data points, that the earlier learners got wrong (*the hard data points*).

Ensemble Learning: Overview

Ensemble Learning: Variants (2)

Bagging (Bootstrap Aggregating):

- Used to combine a collection of high-variance/low-bias predictors into a single predictor with less variance while keeping low-bias
- Draw new training data sets from the training data
- Fit one learner on each subset
- Aggregate the predictions of the learners (e.g. plurality vote in classification, weighted mean in regression)
- Popular algorithm: Random Forests

4. Random Forest

Random Forest:

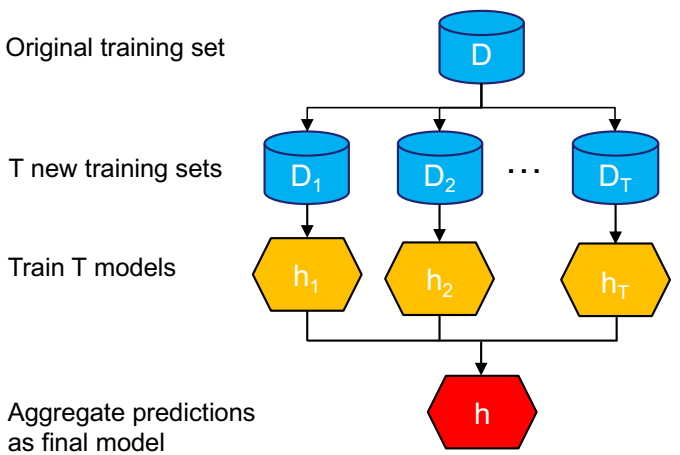
Builds on Bootstrap Aggregating

Bootstrap Aggregating: A group of predictors, obtained from the same learning algorithm, can overcome the limitations of a single predictor

Problem: The learning algorithm needs different training data in order to fit different predictors.

Solution: Generate different data sets by using Bootstrapping (Bootstrap-Sampling):

- Generate T new training sets D_i of size n by sampling with replacement (Bootstrap Samples) from training set D of size n.
- Uniform random sampling with replacement; some observations may be repeated in each D_i
- D_i expected to contain $(1 - \frac{1}{e}) \approx 63.2\%$ of unique observations of D (for large n)



Random Forest:
Bagging Decision Trees

Random Forest: An extension to Bagging with decision trees as learners.

- Use bootstrap sampling for the training data sets
- At each split decision, select a random subset of attributes (feature bagging)

- $\rightarrow$ Goal: Decorrelate the induced decision trees. In a simple bootstrap sample: If one or several attributes are very strong indicators for the target attribute, these attributes will be selected in many of the T trees, causing the trees to become correlated.

Random Forest:

Algorithm [Bre01a]

Algorithm: Random Forest

Input: Training data $D = \{(\mathbf{x}_1, y(\mathbf{x}_1)), \ldots, (\mathbf{x}_n, y(\mathbf{x}_n))\}$, $n \in \mathbb{N}$, $\mathbf{x}_i$ having $|A|$ attributes.
with $\mathbf{x}_i \in X$ and $y(\mathbf{x}_i) \in \{1, \ldots, C\}$ for $1 \leq i \leq n$;
$T \in \mathbb{N}$, the number of predictors.

Output: Ensemble predictor h for X.

1. For $t = 1, \ldots, T$:
   (a) Sample D_t by using Bootstrapping from D
   (Draw n examples from D with Replacement)
   (b) Train a decision tree h_t, i.e. $h_t : X \to Y$ with training set D_t as follows
   - When growing the tree, randomly pick a subset $A' \subset A$ of m attributes at each node
   - Calculate impurity reduction and conduct node splitting using the randomly selected attributes A'
2. Return the classifier:

$$h(\mathbf{x}) = \underset{c \in \{1,\ldots,C\}}{\text{argmax}} \sum_t 1_{[h_t(\mathbf{x})=c]}$$

Random Forest:

Random Forest Performance

Random Forests retain the benefits of decision trees: Handling missing values, handling a mix of continuous and categorical attributes, suitability to classification and regression. In addition, Random Forests

- Do not require (expensive) pruning to generalize
- Can be trained efficiently (linear, parallel)
- Provide good prediction accuracy ("off-the-shelf")

Test set misclassification error (%)

Data set	Forest	Single tree
Breast cancer	2.9	5.9
Ionosphere	5.5	11.2
Diabetes	24.2	25.3
Glass	22.0	30.4
Soybean	5.7	8.6
Letters	3.4	12.4
Satellite	8.6	14.8
Shuttle $\times 10^3$	7.0	62.0
DNA	3.9	6.2
Digit	6.2	17.1

From [Bre01b]

Random Forest:

Properties

Hyperparameters:

- Select number of trees T based on validation error
- Rule of thumb: For $|A|$ attributes, select $m = \sqrt{|A|}$ for classification / select $m = \frac{|A|}{3}$ for regression.

Byproduct: Feature Importance

- When growing the forest, count how often attribute A_k is used as splitting attribute in any node
- Normalize the count by the number of trees T

Similarity among examples $\mathbf{x}_i, \mathbf{x}_j \in D$:

- When growing the forest, count how often $\mathbf{x}_i$ and $\mathbf{x}_j$ end up in the same leaf node
- Normalize the count by the number of trees T

$\Rightarrow$ The normalized count can be used to judge the similarity of the two data points.

References

[Bre96] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[Bre01a] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[Bre01b] Leo Breiman. Statistical modeling: The two cultures. *Statistical Science*, 16(3):199–231, 2001.

[HNPA14] Md Al Mehedi Hasan, Mohammed Nasser, Biprodip Pal, and Shamim Ahmad. Support vector machine and random forest modeling for intrusion detection system (ids). *Journal of Intelligent Learning Systems and Applications*, 6(01):45, 2014.

[M+97] Tom M Mitchell et al. *Machine learning*. McGraw-Hill Boston, MA, 1997.