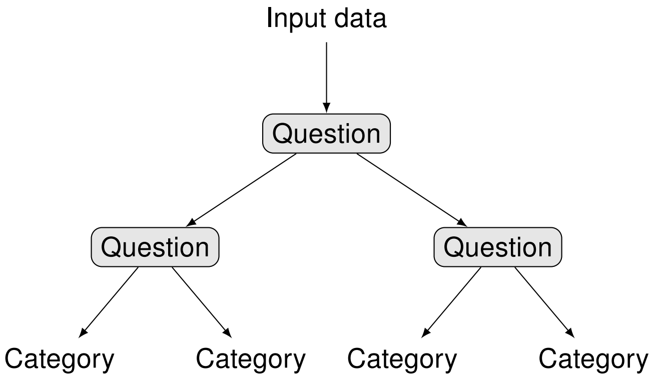
Decision Trees and Random Forests

W2W Machine Learning Workshop

W2W Machine Learning Workshop

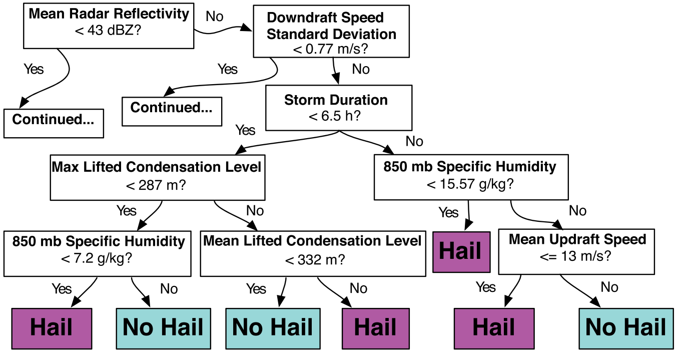
Popular ML algorithms: Decision trees



Decision trees (DT) are flowchart-like structures to classify input data or predict output values.

Decision trees in meteorology

Human-readable, classical tools used for forecasting since the 1950s.



Mean Radar Reflectivity < 43 dBZ?
No
Yes
Continued...
Downdraft Speed Standard Deviation < 0.77 m/s?
Yes
No
Continued...
Storm Duration < 6.5 h?
Yes
No
Max Lifted Condensation Level < 287 m?
850 mb Specific Humidity < 15.57 g/kg?
Yes
No
850 mb Specific Humidity < 7.2 g/kg?
Mean Lifted Condensation Level < 332 m?
Hail
Mean Updraft Speed <= 13 m/s?
Yes
No
Hail
No Hail
No Hail
Hail
Hail
No Hail

FIG. 1. An example of a decision tree for predicting if hail will occur. A version of this decision tree first appeared in Gagne (2016).

from McGovern et al. (BAMS, 2016) https://doi.org/10.1175/BAMS-D-16-0123.1

Tree-based regression and classification

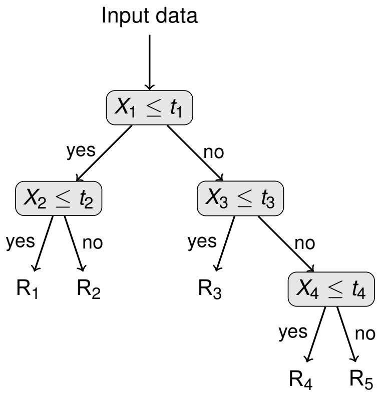
Tree-based methods partition the feature space into a set of rectangles, and then fit a simple model (a constant) in each one.

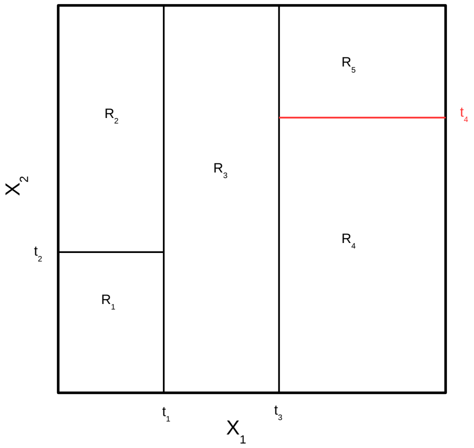
Example: Regression problem for Y based on predictors $\boldsymbol{X} = (X_1, X_2)$.

Split the space (X_1, X_2 plane) into two regions, and model the response by a constant (the mean of Y) in each region. Choose variable and split-point to achieve the best fit.

Split one or both regions into two more regions, and iterate until stopping criterion is reached.

Example: Growing a decision tree





Example decision tree: Prediction

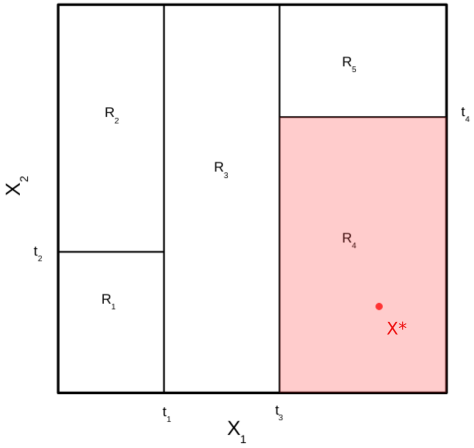
The decision tree results in a partition of the X_1, X_2-plane.

For a new set of features (x_1^*, x_2^*), a prediction is made as follows:

1. Find the set (the rectangle, or leaf) in which (x_1^*, x_2^*) lies.

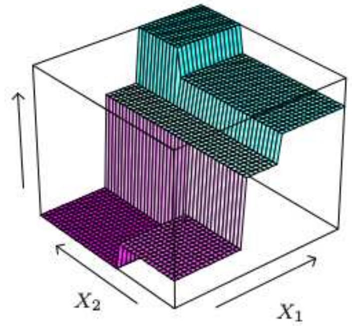
2. Average over all training observations within that set to obtain a prediction $\hat{y}^*$.

- regression: average over target values
- classification: most frequent class, or empirical distribution over classes



Example: Tree-based regression

For regression, the prediction model induced by the decision tree is a combination of constant functions.



adapted from: Hastie et al. (2009). Elements of Statistical Learning. Springer. Figure 9.2.

Resulting model:

$$\hat{y}^* = \sum_{k=1}^{K} c_k \mathbb{1}\left\{\left(X_1^*, X_2^*\right) \in R_k\right\},$$

where c_k is the average of all observations y in the training set for which $(X_1, X_2) \in R_k$.

How to grow a tree?

Algorithms for decision tree learning usually work top-down, by choosing a variable at each step that best splits the set of items, resulting in a recursive partitioning of the training set.

Different algorithms use different metrics for measuring "best". These generally measure the homogeneity of the target variable within the subsets.

Splitting rules depend on task (regression/classification), various choices available.

Umbrella term for both: CART (classification and regression trees), Breiman (1984).

Regression trees

Here: One popular variant for regression.

Finding the best (overall) partition in terms least squares is computationally infeasible.

Instead: greedy algorithm to determine optimal splitting variable j and split point s (inducing half-planes R_1, R_2) minimizing the squared error,

$$\min_{j,s} \left[\sum_{\boldsymbol{X}^i \in R_1} \left(Y_i - \overline{Y}_{R_1} \right)^2 + \sum_{\boldsymbol{X}^i \in R_2} \left(Y_i - \overline{Y}_{R_2} \right)^2 \right]$$

The above criterion is iteratively applied in each step until some stopping criterion is reached.

When to stop growing a (single) tree?

Tree size is a capacity hyperparameter (tuning parameter governing the model's complexity):

- a large tree might overfit the data (extreme case: terminal leaf size 1)
- a small tree might not capture important structures

A minimum node size at which the splitting process is stopped can be determined based on a hold-out validation set.

Alternative: Pruning a decision tree: Grow a large tree T_0, and stop the splitting process only when some minimum node size (say 5) is reached. Then T_0 is pruned using cost-complexity pruning:

Go through subtrees $T \subset T_0$ that can be obtained by collapsing non-terminal nodes, and optimizing a cost function balancing tree size and goodness of fit to the data via cross-validation.

Advantages of decision trees

- simple to display, interpret and understand
- able to handle numerical and categorical data, little data preparation necessary
- white box model: any condition a given situation is observable in a model is easily explained
- mirrors human decision making more closely than other approaches
- built-in feature selection (irrelevant features will be used less)
- (relatively) computationally efficient

Disadvantages of decision trees

- prone to overfitting, requires regularization such as pruning
- can be very non-robust: small change in the training data can result in a large change in the tree and consequently the final predictions
- $2^{q-1} - 1$ possible partitions for categorical predictors with q possible values: potentially computationally prohibitive for large q,

  additional technical difficulties if categorical predictors have different numbers of levels
- greedy algorithm makes locally optimal decisions at each node, but cannot guarantee globally optimal decision tree
- generally lower predictive ability compared to other approaches discussed later

Decision trees as building blocks

Today, decision trees are mostly used as building blocks for ensemble methods, which construct more than one decision tree:

- random forests repeatedly resample the training data to build multiple decision trees
- boosted trees incrementally build an ensemble by training each new instance to emphasize the training instances previously mis-modeled

Bagging trees

Bagging = “bootstrap aggregation”. Aim: Average prediction over a collection of randomly re-sampled training sets, thereby reducing its variance and improving robustness.

Basic idea: Instead of fitting a model (here: a tree) to the entire training data $\mathcal{D}$, proceed as follows

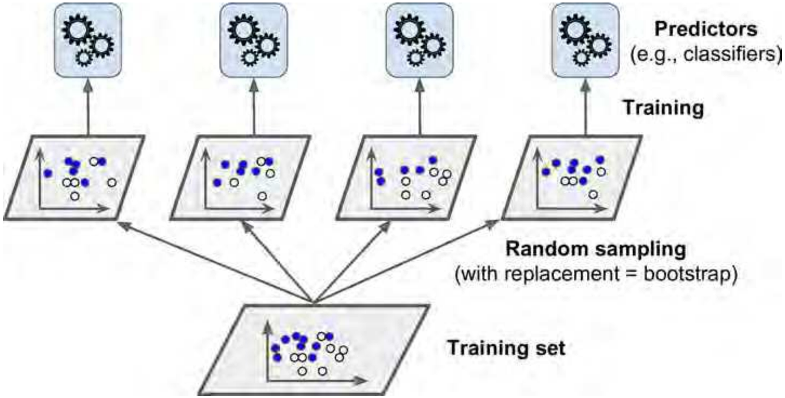
1. draw a random subset from $\mathcal{D}$, denoted by $\mathcal{D}_b$
2. fit a model $\hat{f}_b$ based on this subset $\mathcal{D}_b$ of the training data
3. repeat the above B times

The bagged model prediction (for regression) is the average of the subset-based models:

$$\hat{f}_{\text{bagged}}(\boldsymbol{X}^*) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}_b(\boldsymbol{X}^*)$$

Classification: Majority vote over B subset-based models.

Illustration of bagging



[14.1]

Random forests

Random forests (Breiman, 2001) extend the idea of bagged trees by additionally only considering a random subset of candidate predictor variables for each split (typically $m/3$ (regression) or $\sqrt{m}$ (classification), where m is the total number of predictors).

Motivation: Improve variance reduction of bagging by reducing the correlation between the trees.

Rationale: If one very strong predictor is available, most trees will use this predictor in the first split, leading to similar bagged trees. In this setting, predictions from bagged trees would be highly correlated.

Random forest algorithm

1. For $b = 1$ to B:
   - Draw a bootstrap sample $\mathcal{D}_b$ of size L from the training data.
   - Grow a tree T_b to $\mathcal{D}_b$ by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.
     1. Randomly select m_{split} of the m predictor variables.
     2. Pick the best variable and split point among the m_{split}.
     3. Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_b\}_{b=1}^{B}$.

To make a prediction for a new data point $\boldsymbol{X}^*$:

- regression:

$$\hat{y}^* = \frac{1}{B}\sum_{b=1}^{B} T_b(\boldsymbol{X}^*).$$

- classification: majority vote over class predictions of individual trees

Hyperparameters and regularization of RF

The RF algorithm involves the following hyperparameters that need to be chosen by the user.

Optimal choices of course always depend on the application at hand, and may be based on tuning on hold-out validation sets.

- m_{split}: number of candidate predictors used for each split, effects the correlation between trees

  Typically set to $\approx m/3$ (regression) or $\approx \sqrt{m}$ (classification), where m is the total number of predictors.

  However, optimal choice strongly depends on presence of correlation among predictors and number of useful predictors.

- n_{min}: minimum node size (number of elements), effects complexity of individual trees

  Typically set to 5 (regression) or 1 (classification).

Hyperparameters and regularization of RF

- L: bootstrap sample size, effects correlation between trees

  Settings depend on specific implementation, typically sampling with replacement and $L = n_{\text{train}}$ is used, leading to $\approx \frac{2}{3} n_{\text{train}}$ unique examples.

- B: number of trees, effects computational costs

  Increase of B leads to better approximation of the “true” tree model conditional on the training data.

  RF cannot overfit the data (in terms of increasing B).

  Choice of B can be based on out-of-bag samples, see below.

Interpreting random forests

Alternative: At each split in each tree, improvement in the split-criterion is the importance measure attributed to the splitting variable, and is accumulated over all the trees separately for each variable.

Improvement in the split-criterion for regression: improvement in squared error risk obtained by splitting over that for a constant fit over the entire region.

Advantages of RFs

- simple, variety of implementations available
- flexible and versatile (applicable for different tasks and input data types)
- can be generalized beyond regression and classification (e.g., quantile regression)
- implicit feature selection
- quick to train, can be parallelized
- relatively easy to tune hyperparameters
- not very prone to overfitting
- good results in applications

Disadvantages of RFs

- mathematically not well understood
- less interpretable than DTs (can be mitigated to some extent)
- problems for categorical variables with many levels inherited from DTs
- potentially large model size (trees need to be stored in memory for prediction), might be too slow for real-time prediction
- outperformed by gradient boosting machines in many practical applications