Traditional machine learning methods: From linear regression to random forests

Sebastian Lerch

W2W ML Workshop, Zugspitze, October 2018

Heidelberg Institute for Theoretical Studies HITS

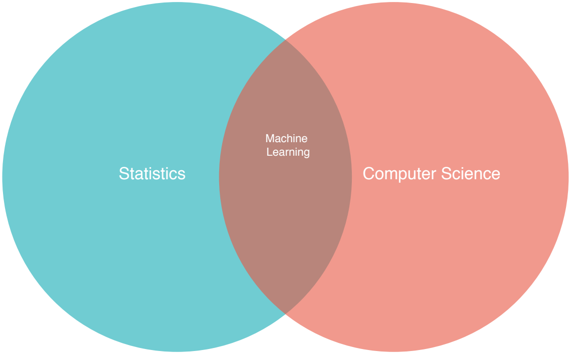
Karlsruhe Institute of Technology

WAVES TO WEATHER

Outline

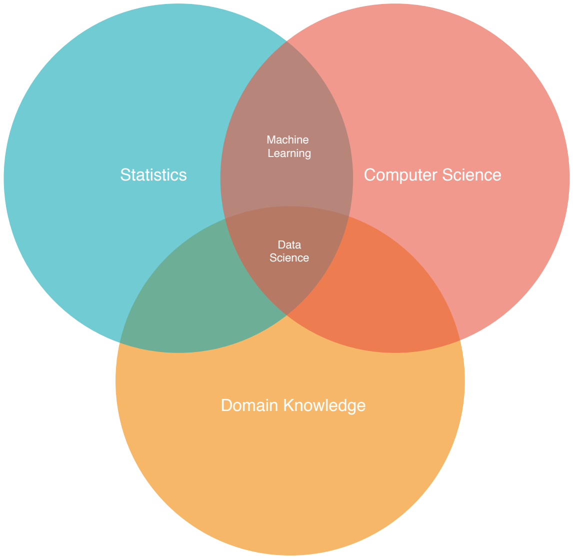
- Statistics vs. machine learning
- Regression, classification, and model evaluation
- Classical statistical models: Linear regression
- Random forests
- Outlook: Advanced regression tasks

Statistics, machine learning and data science



credits to Bin Yu

Statistics, machine learning and data science



credits to Bin Yu

Why should I care?

arXiv.org

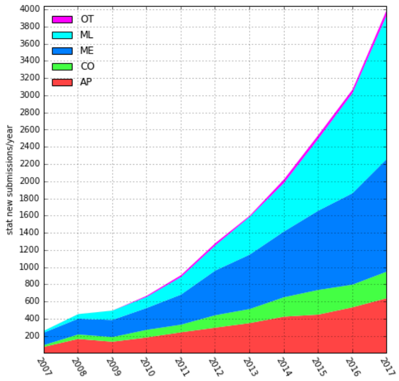
- repository of electronic preprints, mostly from physics, mathematics, statistics, and computer science
- ~ 1.5 million total papers with ~ 1.2 billion downloads
- $> 10\,000$ monthly submissions

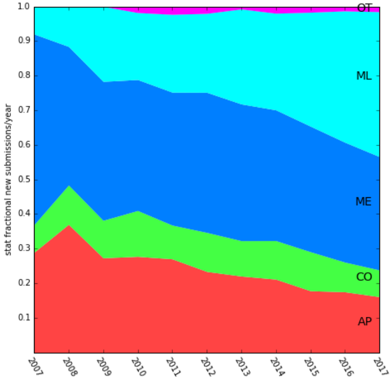
Why should I care?

arXiv.org

- repository of electronic preprints, mostly from physics, mathematics, statistics, and computer science
- ~ 1.5 million total papers with ~ 1.2 billion downloads
- $> 10\,000$ monthly submissions
- *statistics* sub-categories:
  - Applications (AP)
  - Computation (CO)
  - Machine Learning (ML)
  - Methodology (ME)
  - Other Topics (OT)

Relevance of ML in statistics





increase of fraction of ML submissions illustrates shift of research focus in statistics

Regression tasks

Supervised learning task: Given a set of (measured or preset) input variables, some of which have influence on one or more outputs, predict values of the outputs.

Regression tasks

Supervised learning task: Given a set of (measured or preset) input variables, some of which have influence on one or more outputs, predict values of the outputs.

In the language of modern machine learning: Given

- predictors (or features) $X_1, \ldots, X_m$

make a

- prediction (or forecast) $\hat{Y}$

for the

- independent variable (or response / predictand) Y

Specifically: Build a model that produces $\hat{Y}$ by relating Y to $X_1, \ldots, X_m$, based on training data.

Regression and classification

Regression and classification differ in type of response variable:

- classification is based on qualitative data (e.g., rain / no rain; presence of a car in an image; anmial/plant species; (hand-written) digit; school grades; . . .)
- regression is based on quantitative data (e.g., real-valued atmospheric measurements; . . .)

Essentially, classification is regression with qualitative data.

Regression and classification

Regression and classification differ in type of response variable:

- classification is based on qualitative data (e.g., rain / no rain; presence of a car in an image; anmial/plant species; (hand-written) digit; school grades; . . .)
- regression is based on quantitative data (e.g., real-valued atmospheric measurements; . . .)

Essentially, classification is regression with qualitative data.

more details in Martin's talk, I will focus on regression in the following

Forecasting/Prediction often refers to out-of-sample modeling, while regression often refers to in-sample modeling. I will use the two terms interchangeably.

Application example

- response: 00 UTC 2-meter temperature measurements at around 500 DWD observation stations in Germany
- predictors: NWP model output and station characteristics:
  - 48h-ahead ECMWF ensemble mean forecasts of several variables (temperature, pressure, cloud cover, soil moisture, radiation, ...), interpolated to station locations
  - station altitude, longitude, latitude, orography information

Goal: Predict 2-meter temperature based on predictors.

Note: NWP model output changes from day to day, but station characteristics stay the same

based on:

Rasp, S. and Lerch, S. (2018). **Neural networks for post-processing ensemble weather forecasts**, https://arxiv.org/abs/1805.09091.

Mathematical formulation

Input: For past forecast cases $t = 1, \dots, T$, we have

- vectors-valued predictors $\boldsymbol{X}^1, \dots, \boldsymbol{X}^T$, where for $i = 1, \dots, T$

$$\boldsymbol{X}^i = X_1^i, \dots, X_m^i$$

- corresponding values of the response $Y_1, \dots, Y_T$

Mathematical formulation

Input: For past forecast cases $t = 1, \ldots, T$, we have

- vectors-valued predictors $\boldsymbol{X}^1, \ldots, \boldsymbol{X}^T$, where for $i = 1, \ldots, T$

$$\boldsymbol{X}^i = X_1^i, \ldots, X_m^i$$

- corresponding values of the response $Y_1, \ldots, Y_T$

Task: For an out-of-sample data point (forecast case t^*), i.e., predictor values

$$\boldsymbol{X}^{t^*}$$

generate a prediction $\hat{Y}_{t^*}$.

This is done by building a model relating predictors $\boldsymbol{X}$ to responses Y, and estimating model parameters based on past data points (or training data) $(\boldsymbol{X}^1, Y_1), \ldots, (\boldsymbol{X}^T, Y_T)$.

Application example

Past forecast cases include dates $d = 1, \dots, D$ and stations $s = 1, \dots, S$.

- past vectors of predictors $\boldsymbol{X}^{s,d} = \left(X^{s,d}_{\text{t2m\_fc}}, X^{s,d}_{\text{tcc\_fc}}, X^{s,d}_{\text{sp\_fc}}, \dots, X^{s,d}_{\text{station\_altitude}}\right)$
- past observations of temperature $Y_{s,t}$

Aim: Build a model to predict a future (d^*) temperature at some station (s^*) based on the corresponding predictor $\boldsymbol{X}^{s^*,d^*}$.

This is achieved by building a model that relates $\boldsymbol{X}^{s,d}$ to $Y_{s,d}$.

For ease of notation, dates and stations will be summarized via multi-index $t = (d, s)$

Model evaluation

A model $\mathcal{M}(\boldsymbol{X})$ produces a prediction $\hat{Y}$ of the response Y.

To measure the quality of $\mathcal{M}$ based on an evaluation (or test) set

$$E = \{(\boldsymbol{X}^1, Y_1), \ldots, (\boldsymbol{X}^n, Y_n)\},$$

produce forecasts $\hat{Y}_1, \ldots, \hat{Y}_n$ via $\mathcal{M}$ based on a training set distinct from E, and compare them to the actual values $Y_1, \ldots, Y_n$.

Model evaluation

A model $\mathcal{M}(\boldsymbol{X})$ produces a prediction $\hat{Y}$ of the response Y.

To measure the quality of $\mathcal{M}$ based on an evaluation (or test) set

$$E = \{(\boldsymbol{X}^1, Y_1), \ldots, (\boldsymbol{X}^n, Y_n)\},$$

produce forecasts $\hat{Y}_1, \ldots, \hat{Y}_n$ via $\mathcal{M}$ based on a training set distinct from E, and compare them to the actual values $Y_1, \ldots, Y_n$.

For example, by computing

- the mean squared error $\frac{1}{n}\sum_{i=1}^{n}\left(\hat{Y}_i - Y_i\right)^2$
- the mean absolute error $\frac{1}{n}\sum_{i=1}^{n}\left|\hat{Y}_i - Y_i\right|$

Linear regression

Standard statistical model, models Y as linear combination of predictors $X_1, \ldots, X_m$

$$Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_m X_m + \varepsilon$$

Assumptions: ε is an unobserved random error with mean 0 and finite variance.

Model predictions:

$$\hat{Y} = \beta_0 + \beta_1 X_1 + \cdots + \beta_m X_m = \boldsymbol{X}^T \boldsymbol{\beta}$$

Parameter estimation in LR models

Reminder: $Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_m X_m + \varepsilon$.

Parameters $(\beta_0, \beta_1, \ldots, \beta_m)$ are estimated by least squares, i.e., by minimizing squared residuals (the squared error)

$$\sum_{i=1}^{T} \left(Y_i - \boldsymbol{X^i}^T \boldsymbol{\beta} \right)^2$$

over a training set.

The (mathematically provable) optimal solution $(\hat{\beta} = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T Y)$ can be computed explicitly.

In-sample model checking: Coefficient of determination

Given a fitted LR model, the coefficient of determination

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{\sum_{i=1}^{n}(Y_i - \overline{Y})^2}$$

is often used to assess the in-sample model fit.

$R^2 = 1$ indicates perfect model fit.

R^2 is often interpreted as the proportion of response variation "explained" by the regressors in the model.

In case of a single regressor, fitted by least squares, R^2 is the square of the correlation coefficient.

Advantages and disadvantages of LR models

Advantages

- simple (overfitting less likely)
- mathematically well understood
- explicit optimal parameter estimates available
- computationally simple

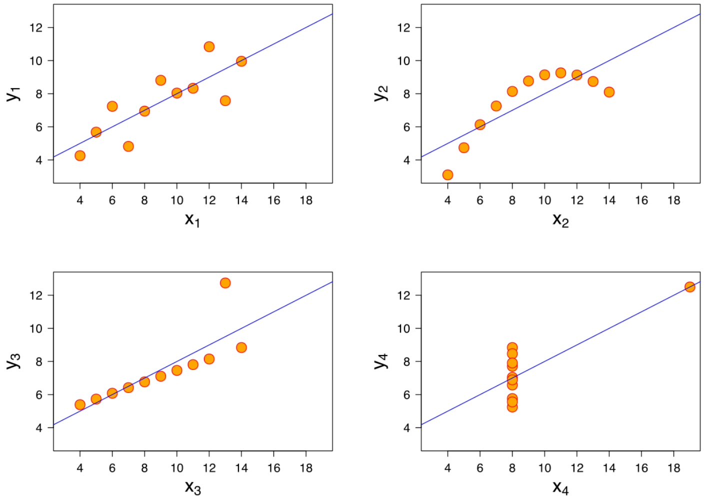
Disadvantages

- assumes linear relation and uncorrelated predictors
- often complicated predictor selection required, e.g. via ridge regression or LASSO methods

Potentially a “high bias, low variance” model.

Anscombe's quartett

Four datasets with almost identical R^2 of LR model



https://en.wikipedia.org/wiki/File:Anscombe27s_quartet_3.svg

A simple alternative: k-nearest neighbor regression

simple nonparametric regression approach, proceeds as follows to produce a forecast $\hat{Y}_{t^*}$ based on $\boldsymbol{X}^{t^*}$:

1. compute similarities of all past predictor values to $\boldsymbol{X}^{t^*}$

A simple alternative: k-nearest neighbor regression

simple nonparametric regression approach, proceeds as follows to produce a forecast $\hat{Y}_{t^*}$ based on $\boldsymbol{X}^{t^*}$:

1. compute similarities of all past predictor values to $\boldsymbol{X}^{t^*}$
2. find k training cases $(u_1, \dots u_k)$ with smallest distances

A simple alternative: k-nearest neighbor regression

simple nonparametric regression approach, proceeds as follows to produce a forecast $\hat{Y}_{t^*}$ based on $\boldsymbol{X}^{t^*}$:

1. compute similarities of all past predictor values to $\boldsymbol{X}^{t^*}$
2. find k training cases $(u_1, \dots u_k)$ with smallest distances
3. determine regression estimate $\hat{Y}_{t^*}$ as (weighted) average of the past observations corresponding from those cases:

$$\hat{Y}_{t^*} = \frac{1}{k} \sum_{j=1}^{k} Y_{u_j}$$

A simple alternative: k-nearest neighbor regression

simple nonparametric regression approach, proceeds as follows to produce a forecast $\hat{Y}_{t^*}$ based on $\boldsymbol{X}^{t^*}$:

1. compute similarities of all past predictor values to $\boldsymbol{X}^{t^*}$
2. find k training cases $(u_1, \dots u_k)$ with smallest distances
3. determine regression estimate $\hat{Y}_{t^*}$ as (weighted) average of the past observations corresponding from those cases:

$$\hat{Y}_{t^*} = \frac{1}{k} \sum_{j=1}^{k} Y_{u_j}$$

Related meteorological approach: Historical analogs.

Potentially low bias, high variance approach (prone to overfitting).

Predictor selection potentially required, but not obvious. Motivates more complex approaches, such as random forests.

Random forests for regression (and classification)

Random forests provide popular and flexible nonparametric tools for regression and classification.

They combine decision trees and bootstrap aggregation.

In the following, we review

- decision trees
- tree-based regression models
- growing and pruning trees
- bagging and random forests

Decision trees

human-readable, classical tools for meteorological forecasting since the 1950s

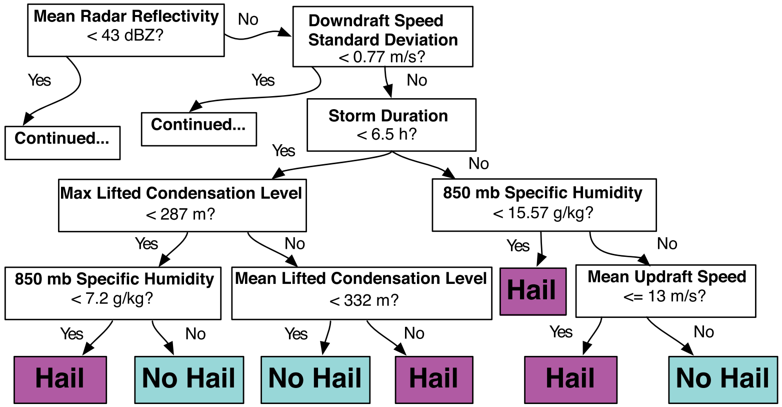

Mean Radar Reflectivity < 43 dBZ?
No
Downdraft Speed Standard Deviation < 0.77 m/s?
Yes
Continued...
Yes
Continued...
No
Storm Duration < 6.5 h?
Yes
No
Max Lifted Condensation Level < 287 m?
850 mb Specific Humidity < 15.57 g/kg?
Yes
No
Yes
No
850 mb Specific Humidity < 7.2 g/kg?
Mean Lifted Condensation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

Tree-based regression and classification (CART)

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 X_1, X_2.

Tree-based regression and classification (CART)

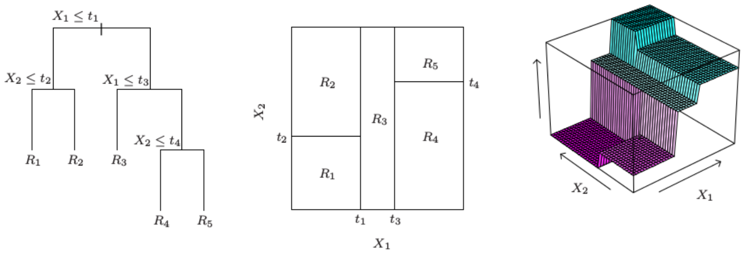
Tree-based methods partition the feature space into a set of rectangles, and then fit a simple model (a constant) in each one.

Example: Regression problem for Y based on predictors X_1, X_2.

Split the space (X_1, X_2 plane) into two regions, and model the response by 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: Tree-based regression



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

Resulting model: $\hat{Y} = \sum_{k=1}^{K} c_k \mathbb{1}\left\{(X_1, X_2) \in R_k\right\}$, where c_k is the average of all observations Y for which $(X_1, X_2) \in R_k$.

How to grow a tree?

Depends on task (regression/classification), various choices available. Here: One popular variant for regression.

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

How to grow a tree?

Depends on task (regression/classification), various choices available. 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,

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

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 tuning parameter: A large tree might overfit the data (extreme case: terminal leaf size 1.), but a small tree might not capture important structures.

When to stop growing a (single) tree?

Tree size is a tuning parameter: A large tree might overfit the data (extreme case: terminal leaf size 1.), but a small tree might not capture important structures.

Pruning: Grow a large tree T_0 , 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.

When to stop growing a (single) tree?

Tree size is a tuning parameter: A large tree might overfit the data (extreme case: terminal leaf size 1.), but a small tree might not capture important structures.

Pruning: Grow a large tree T_0 , 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.

Note: RFs are an alternative form of mitigation of potential overfitting and usually do not require pruning.

Bagging trees

Bagging = "bootstrap aggregation". Aim: Average prediction over a collection of bootstrap samples, thereby reducing its variance.

Basic idea: Instead of fitting a model $\hat{f}$ (here: a tree) to the entire training data $\mathcal{D} = \{(\boldsymbol{X}^{\mathbf{1}}, Y_1), \ldots, (\boldsymbol{X}^{\boldsymbol{T}}, Y_T)\}$, proceed as follows

1. draw a random subset of size L 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

Bagging trees

Bagging = "bootstrap aggregation". Aim: Average prediction over a collection of bootstrap samples, thereby reducing its variance.

Basic idea: Instead of fitting a model $\hat{f}$ (here: a tree) to the entire training data $\mathcal{D} = \{(\boldsymbol{X}^{\mathbf{1}}, Y_1), \ldots, (\boldsymbol{X}^{\boldsymbol{T}}, Y_T)\}$, proceed as follows

1. draw a random subset of size L 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 predictions is obtained as average of the subset-based models:

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

Each bootstrap tree will involve different features than the full tree.

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 $m/3$).

Motivation: Improve variance reduction of bagging by reducing the correlation between the trees, without increasing the variance too much.

Random forest algorithm

1. For $b = 1$ to B:
   - Draw a bootstrap sample $\mathcal{D}_b$ of size L from the training data.

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.
     1.1 Randomly select n_{split} of the m predictor variables.

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.
     1.1 Randomly select n_{split} of the m predictor variables.
     1.2 Pick the best variable and split point among the n_{split}.

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.
     1.1 Randomly select n_{split} of the m predictor variables.
     1.2 Pick the best variable and split point among the n_{split}.
     1.3 Split the node into two daughter nodes.

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.
     1.1 Randomly select n_{split} of the m predictor variables.
     1.2 Pick the best variable and split point among the n_{split}.
     1.3 Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_b\}_{b=1}^{B}$.

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.
     1.1 Randomly select n_{split} of the m predictor variables.
     1.2 Pick the best variable and split point among the n_{split}.
     1.3 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 $\boldsymbol{X}^{t^*}$:

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

Tuning parameters: $n_{\text{min}}, n_{\text{split}}, B, L$.

Predictor importance measures for RF

Two alternative (in-sample) measures of predictor importance:

- At each split in each tree, the 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.

Predictor importance measures for RF

Two alternative (in-sample) measures of predictor importance:

- At each split in each tree, the 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.
- When fitting the b-th tree, record prediction accuracy in the "out of bag" (OOB) samples (those not in $\mathcal{D}_b$). Then randomly permute values for the j-th variable in OOB samples, and re-compute prediction accuracy. Average resulting decrease in accuracy over all trees measures the importance of variable j in the random forest.

Advantages and disadvantages of RFs

Advantages

- simple, variety of implementations available
- flexible and versatile (applicable for different tasks and input data types)
- interpretable (implicit feature selection)
- quick to train, can be parallelized
- good results in applications

Disadvantages

- mathematically not well understood (e.g., Athey et al., 2016)
- potentially large model size (trees need to be stored in memory for prediction)

Outlook: More complex regression tasks

- logistic regression: model probability of occurrence of a binary response
- quantile regression: estimate conditional quantiles of response variable
- distributional regression: estimate conditional distribution of response variable (e.g., statistical post-processing of NWP ensembles)

All advanced regression tasks call for the use of modern ML methods and lead to interesting statistical questions, e.g., regarding suitable loss functions for model evaluation and training.