Parties, Models, Mobsters
Methods and Software for Model-Based Recursive Partitioning

Achim Zeileis

http://eeecon.uibk.ac.at/~zeileis/
Parties, Models, Mobsters

Motivation
Motivation: Trees


- **Data models:** Stochastic models, typically parametric. → Classical strategy in statistics. Regression models are still the workhorse for many empirical analyses.

- **Algorithmic models:** Flexible models, data-generating process unknown. → Less applications in many fields, e.g., social sciences or economics.

**Classical example:** Trees, i.e., modeling of dependent variable $y$ by “learning” a recursive partition w.r.t explanatory variables $z_1, \ldots, z_l$. 

Example: Recursive partitioning (RPart) for dependence of survival on the Titanic w.r.t. gender, age, and class of passengers.
Motivation: Trees


- **Data models**: Stochastic models, typically parametric.  
  → Classical strategy in statistics. Regression models are still the workhorse for many empirical analyses.

- **Algorithmic models**: Flexible models, data-generating process unknown.  
  → Less applications in many fields, e.g., social sciences or economics.

**Classical example**: Trees, i.e., modeling of dependent variable $y$ by “learning” a recursive partition w.r.t explanatory variables $z_1, \ldots, z_l$.

**Example**: Recursive partitioning (RPart) for dependence of survival on the Titanic w.r.t. gender, age, and class of passengers.
Motivation: Trees

1. Gender
   - Male
   - Female

2. Age
   - Adult
   - Child

3. Class
   - 3rd
   - 1st, 2nd

Node 3 (n = 1667)

Node 5 (n = 48)

Node 6 (n = 16)

Node 8 (n = 196)

Node 9 (n = 274)
Motivation: Leaves

**Key features:**

1. Predictive power in nonlinear regression relationships.
2. Interpretability (enhanced by visualization), i.e., no “black box” methods.

**Typically:** Simple models for univariate $y$, e.g., mean.
Motivation: Leaves

**Key features:**

1. Predictive power in nonlinear regression relationships.
2. Interpretability (enhanced by visualization), i.e., no “black box” methods.

**Typically:** Simple models for univariate $y$, e.g., mean.

**Idea:** More complex models for more complex $y$, e.g., regression models, multivariate normal model, item responses, etc.

**Here:** Synthesis of parametric data models and algorithmic tree models.

**Goal:** Fitting local models by partitioning of the sample space.
Model-based recursive partitioning
Model-based recursive partitioning

**MOB algorithm:**

1. Fit the parametric model in the current subsample.
2. Assess the stability of the parameters across each splitting variable \( z_j \).
3. Split sample along the \( z_{j^*} \) with strongest instability: Choose breakpoint with highest improvement of the model fit.
4. Repeat steps 1–3 recursively in the subsamples until some stopping criterion is met.
Model-based recursive partitioning

**Example:** Logistic regression, assessing differences in the effect of “preferential treatment” (“women and children first”?) in the Titanic survival data.

**In R:** Generalized linear model tree with binomial family (and default logit link).

```r
R> library("partykit")
R> mb <- glmtree(Survived ~ Treatment | Age + Gender + Class,
+ data = ttnc, family = binomial, alpha = 0.05, prune = "BIC")
R> plot(mb)
R> print(mb)
```

**Result:** Log-odds ratio of survival given treatment differs across classes (slope), as does the survival probability of male adults (intercept).
Model-based recursive partitioning

Node 2 (n = 706)

Node 4 (n = 285)

Node 5 (n = 1210)

Class

p < 0.001

1

3rd

1st, 2nd, Crew

Yes

No

Normal (Male&Adult)

Preferential (Female|Child)

Normal (Male&Adult)

Preferential (Female|Child)

Normal (Male&Adult)

Preferential (Female|Child)
Model-based recursive partitioning

Generalized linear model tree (family: binomial)

Model formula:
Survived ~ Treatment | Age + Gender + Class

Fitted party:
[1] root
|   (Intercept) Treatment Preferential
|   -1.641 1.327
| [3] Class in 1st, 2nd, Crew
| |   (Intercept) Treatment Preferential
| |   -2.398 4.477
| | [5] Class in 1st, Crew: n = 1210
| |   (Intercept) Treatment Preferential
| |   -1.152 4.318

Number of inner nodes: 2
Number of terminal nodes: 3
Number of parameters per node: 2
Objective function (negative log-likelihood): 1061
1. Model estimation

Models: $M(y, x, \theta)$ with (potentially multivariate) observations $y$, optionally regressors $x$, and $k$-dimensional parameter vector $\theta \in \Theta$.

Parameter estimation: $\hat{\theta}$ by optimization of additive objective function $\Psi(y, x, \theta)$ for $n$ observations $y_i$ ($i = 1, \ldots, n$):

$$\hat{\theta} = \arg\min_{\theta \in \Theta} \sum_{i=1}^{n} \psi(y_i, x_i, \theta).$$

Special cases: Maximum likelihood (ML), weighted and ordinary least squares (WLS and OLS), quasi-ML, CRPS, and other M-estimators.
1. Model estimation

**Estimating function:** \( \hat{\theta} \) can also be defined in terms of

\[
\sum_{i=1}^{n} \psi(y_i, x_i, \hat{\theta}) = 0,
\]

where \( \psi(y, x, \theta) = \frac{\partial \Psi(y, x, \theta)}{\partial \theta} \) is the model score function.

**Central limit theorem:** If there is a true parameter \( \theta_0 \) and given certain weak regularity conditions

\[
\sqrt{n}(\hat{\theta} - \theta_0) \xrightarrow{d} \mathcal{N}(0, V(\theta_0)),
\]

where \( V(\theta_0) = \{A(\theta_0)\}^{-1}B(\theta_0)\{A(\theta_0)\}^{-1} \). \( A \) and \( B \) are the expectation of the derivative of \( \psi \) and the variance of \( \psi \), respectively.
1. Model estimation

**Idea:** In many situations, a single global model $M(y, x, \theta)$ that fits all $n$ observations cannot be found. But it might be possible to find a partition w.r.t. the variables $z_1, \ldots, z_l$ so that a well-fitting model can be found locally in each cell of the partition.

**Tools:**
- Assess parameter instability w.r.t to splitting variables $z_j (j = 1, \ldots, l)$.
- A general measure of deviation from the model is the score or estimating function $\psi(y, x, \theta)$. 
2. Tests for parameter instability
2. Tests for parameter instability

**Test statistics:** Scalar functional $\lambda(W_j)$ that captures deviations from zero.

**Null distribution:** Asymptotic distribution of $\lambda(W^0)$.

**Special cases:** Class of test encompasses many well-known tests for different classes of models. Certain functionals $\lambda$ are particularly intuitive for numeric and categorical $z_j$, respectively.

**Advantage:** Model $\mathcal{M}(y, x, \hat{\theta})$ just has to be estimated once. Empirical estimating functions $\psi(y_i, x_i, \hat{\theta})$ just have to be re-ordered and aggregated for each $z_j$. 
2. Tests for parameter instability

Class of tests: Generalized M-fluctuation tests capture instabilities in $\hat{\theta}$ for an ordering w.r.t $z_j$.

Basis: Empirical fluctuation process of cumulative deviations w.r.t. to an ordering $\sigma(z_{ij})$.

$$W_j(t, \hat{\theta}) = B^{-1/2} n^{-1/2} \sum_{i=1}^{\lfloor nt \rfloor} \psi(y_{\sigma(z_{ij})}, x_{\sigma(z_{ij})}, \hat{\theta}) \quad (0 \leq t \leq 1)$$

Functional central limit theorem: Under parameter stability $W_j(\cdot) \xrightarrow{d} W^0(\cdot)$, where $W^0$ is a $k$-dimensional Brownian bridge.
2. Tests for parameter instability

**Splitting numeric variables:** Assess instability using supLM statistics.

\[
\lambda_{supLM}(W_j) = \max_{i=\hat{i}, \ldots, \bar{i}} \left( \frac{i}{n} \cdot \frac{n-i}{n} \right)^{-1} \left\| W_j \left( \frac{i}{n} \right) \right\|_2^2.
\]

**Interpretation:** Maximization of single shift LM statistics for all conceivable breakpoints in \([\hat{i}, \bar{i}]\).

**Limiting distribution:** Supremum of a squared, \(k\)-dimensional tied-down Bessel process.

**Potential alternatives:** Many other parameter instability tests from the same class of tests, e.g., a Cramér-von Mises test (or Nyblom-Hansen test), MOSUM tests, etc.
2. Tests for parameter instability

**Splitting categorical variables:** Assess instability using $\chi^2$ statistics.

$$\lambda_{\chi^2}(W_j) = \sum_{c=1}^{C} \frac{n}{|I_c|} \left\| \Delta_{I_c} W_j \left( \frac{i}{n} \right) \right\|^2.$$

**Feature:** Invariant for re-ordering of the $C$ categories and the observations within each category.

**Interpretation:** Capture instability for split-up into $C$ categories.

**Limiting distribution:** $\chi^2$ with $k \cdot (C - 1)$ degrees of freedom.
2. Tests for parameter instability

**Splitting ordinal variables:** Several strategies conceivable. Assess instability either as for categorical variables (if $C$ is low), or as for numeric variables (if $C$ is high), or via a specialized test.

$$
\lambda_{\text{maxLMo}}(W_j) = \max_{i \in \{i_1, \ldots, i_{C-1}\}} \left( \frac{i}{n} \cdot \frac{n-i}{n} \right)^{-1} \left\| W_j \left( \frac{i}{n} \right) \right\|_2^2,
$$

$$
\lambda_{\text{WDMo}}(W_j) = \max_{i \in \{i_1, \ldots, i_{C-1}\}} \left( \frac{i}{n} \cdot \frac{n-i}{n} \right)^{-1/2} \left\| W_j \left( \frac{i}{n} \right) \right\|_\infty.
$$

**Interpretation:** Assess only the possible splitpoints $i_1, \ldots, i_{C-1}$, based on $L_2$ or $L_\infty$ norm.

**Limiting distribution:** Maximum from selected points in a squared Bessel process or multivariate normal distribution, respectively.
2. Tests for parameter instability

**Alternative inference frameworks:** Classic association tests for independence of $y$ and $z_j$ can be turned into model-based tests by using model scores $\psi(y, x, \hat{\theta})$ instead of just $y$ (Schlosser, Hothorn, Zeileis 2019, arXiv).


- Based on conditional inference (or permutation tests).
- Originally nonparametric.


- Based on $\chi^2$ tests.
- Originally based on residuals only (not full model scores).
- Categorizes both $z_j$ and the model residuals (or scores) into bins.
2. Tests for parameter instability

**Alternative inference frameworks:** Classic association tests for independence of $y$ and $z_j$ can be turned into model-based tests by using model scores $\psi(y, x, \hat{\theta})$ instead of just $y$ (Schlosser, Hothorn, Zeileis 2019, arXiv).

**CTree:** Hothorn, Hornik, Zeileis (2006, *JCGS*).
- Based on conditional inference (or permutation tests).
- Originally nonparametric.
2. Tests for parameter instability

**Alternative inference frameworks:** Classic association tests for independence of \( y \) and \( z_j \) can be turned into model-based tests by using model scores \( \psi (y, x, \hat{\theta}) \) instead of just \( y \) (Schloesser, Hothorn, Zeileis 2019, arXiv).

**CTree:** Hothorn, Hornik, Zeileis (2006, *JCGS*).
- Based on conditional inference (or permutation tests).
- Originally nonparametric.

**GUIDE:** Loh (2002, *Statistica Sinica*).
- Based on \( \chi^2 \) tests.
- Originally based on residuals only (not full model scores).
- Categorizes both \( z_j \) and the model residuals (or scores) into bins.
3. Splitting

**Goal:** Split model into \( b = 1, \ldots, B \) subsamples along the splitting variable \( z_j \) associated with the highest parameter instability. Local optimization of

\[
\sum_b \sum_{i \in I_b} \Psi(y_i, x_i, \theta_b).
\]

\( B = 2 \): Exhaustive search of order \( O(n) \).

\( B > 2 \): Exhaustive search is of order \( O(n^{B-1}) \), but can be replaced by dynamic programming of order \( O(n^2) \). Different methods (e.g., information criteria) can choose \( B \) adaptively.

**Here:** Binary splitting. Optionally, \( B = C \) can be chosen (without search) for categorical variables.
3. Splitting

Alternatively:

- Selecting the optimal split w.r.t. the objective function $\Psi(y, x, \theta)$ requires refitting the model and may be costly.
- Employ a maximally-selected score-based test statistic instead.
- Avoids refitting the model and is thus much cheaper to compute.
4. Pruning

**Goal:** Avoid overfitting.

**Pre-pruning:**
- Internal stopping criterium.
- Stop splitting when there is no significant parameter instability.
- Based on Bonferroni-corrected $p$ values of the parameter instability tests.

**Post-pruning:**
- Grow large tree (e.g., with high significance level).
- Prune splits that do not improve the model fit based on information criteria (e.g., AIC or BIC).

**Hyperparameters:** Significance level and information criterion penalty can be chosen manually (or possibly through cross-validation etc.).
Parties, Models, Mobsters

Software
Software

**Workhorse function:** mob() for
- data handling,
- calling model fitters,
- carrying out parameter instability tests and
- recursive partitioning algorithm.

**Required functionality:**
- **Parties:** Class and methods for recursive partytions.
- **Models:** Model fitting functions (optimizing suitable objective function).
- **Mobsters:** High-level interfaces (lmtree(), glmtree(), bttree(), ...) that call lower-level mob() with suitable options and methods.
Software

Parties: S3 class ‘modelparty’ built on ‘party’.
  • Separates data and tree structure.
  • Inherits generic infrastructure for printing, predicting, plotting, …

Models: Plain functions with input/output convention.
  • Basic and extended interface for rapid prototyping and for speeding up computings, respectively.
  • Only minimal glue code required if models are well-designed.

Mobsters:
  • mob() completely agnostic regarding models employed.
  • Separate interfaces lmtree(), glmtree(), …
  • New interfaces typically need to bring their model fitter and adapt the main methods print(), plot(), predict() etc.
Software

**Input:** Basic model interface.

\[ \text{fit}(y, x = \text{NULL}, \text{start} = \text{NULL}, \text{weights} = \text{NULL}, \text{offset} = \text{NULL}, \ldots) \]

\( y, x, \text{weights}, \text{offset} \) are (the subset of) the preprocessed data. Starting values are in \( \text{start} \) and further fitting arguments in \( \ldots \).

**Output:** Fitted model object of class with suitable methods.

- `coef()`: Estimated parameters \( \hat{\theta} \).
- `logLik()`: Maximized log-likelihood function \(- \sum_i \Psi(y_i, x_i, \hat{\theta})\).
- `estfun()`: Empirical estimating functions \( \Psi'(y_i, x_i, \hat{\theta}) \).
Software

**Input:** Extended model interface.

\[
\text{fit}(y, x = \text{NULL}, \text{start} = \text{NULL}, \text{weights} = \text{NULL}, \text{offset} = \text{NULL}, \ldots, \\
\text{estfun} = \text{FALSE}, \text{object} = \text{FALSE})
\]

**Output:** List.

- **coefficients:** Estimated parameters \(\hat{\theta}\).
- **objfun:** Minimized objective function \(\sum_i \Psi(y_i, x_i, \hat{\theta})\).
- **estfun:** Empirical estimating functions \(\Psi'(y_i, x_i, \hat{\theta})\).
  Only needed if \(\text{estfun} = \text{TRUE}\), otherwise optionally \text{NULL}.
- **object:** A model object (providing further methods).
  Only needed if \(\text{object} = \text{TRUE}\), otherwise optionally \text{NULL}.

**Internally:** Extended interface constructed from basic interface if supplied. Efficiency can be gained through extended approach.
Mobsters:

- Distributions: Parametric, multivariate, circular, transformation (*disttree, cirtree, trtf*).
- Linear and generalized linear models (*partykit, palmtree*).
- Linear and generalized linear mixed effects models (*glmertree*).
- Survival models (*partykit, model4you*).
- Beta regression (*betareg*).
- Psychometric models: Bradley-Terry, item response theory, multinomial processing trees (*psychotree*).
- Structural equation models (*partykit, semtree*).
- Network models (*networktree*).
- Spatial lag models (*lagsarImtree*).
Parties, Models, Mobsters

Bradley-Terry trees
Bradley-Terry trees

**Question:** Which of these women is more attractive?

**And:** How does the answer depend on age, gender, and the familiarity with the associated TV show *Germany’s Next Topmodel*?
Bradley-Terry trees

Data: Paired comparisons of attractiveness.

- Germany’s Next Topmodel 2007 finalists: Barbara, Anni, Hana, Fiona, Mandy, Anja.
- Survey with 192 respondents at Universität Tübingen.
- Available covariates: Gender, age, familiarity with the TV show.
- Familiarity assessed by yes/no questions:
  1. Do you recognize the women?/Do you know the show?
  2. Did you watch it regularly?
  3. Did you watch the final show?/Do you know who won?
Bradley-Terry trees

**Model:** Bradley-Terry (or Bradley-Terry-Luce) model.

- Standard model for paired comparisons in social sciences.
- Parametrizes probability $\pi_{ij}$ for preferring object $i$ over $j$ in terms of corresponding “ability” or “worth” parameters $\theta_i$.

$$
\pi_{ij} = \frac{\theta_i}{\theta_i + \theta_j}
$$

$$
\text{logit}(\pi_{ij}) = \log(\theta_i) - \log(\theta_j)
$$

- Maximum likelihood as a logistic or log-linear GLM.
Bradley-Terry trees

**Mobster:** Bradley-Terry trees.

- **Core infrastructure:** Model-fitting function `btmodel()` in `psychotools`.
- **High-level interface:** `bttree()` in `psychotree`.
- **Here:** Recreation from scratch using only `mob()` and `btmodel()`.

**Illustration:**

```r
R> library("psychotree")
R> data("Topmodel2007", package = "psychotree")
R> bt <- bttree(preference ~ gender + age + q1 + q2 + q3, data = Topmodel2007)
R> plot(bt)
R> print(bt)
```
Bradley-Terry trees

1. age
   - $p < 0.001$
   - $\leq 52$
   - $> 52$

2. $q^2$
   - $p = 0.017$
   - yes
   - no

3. gender
   - $p = 0.007$
   - male
   - female

Node 3 (n = 35)
- B
- Ann
- H
- F
- M
- Anj

Node 5 (n = 71)
- B
- Ann
- H
- F
- M
- Anj

Node 6 (n = 56)
- B
- Ann
- H
- F
- M
- Anj

Node 7 (n = 30)
- B
- Ann
- H
- F
- M
- Anj
Bradley-Terry trees

Bradley-Terry tree

Model formula:
preference ~ gender + age + q1 + q2 + q3

Fitted party:
[1] root
  | [2] age <= 52
  |   | [3] q2 in yes: n = 35
  |   | Barbara  Anni  Hana  Fiona  Mandy
  |   | 1.3378  1.2318  2.0499  0.8339  0.6217
  |   | [4] q2 in no
  |   |   | [5] gender in male: n = 71
  |   |   | Barbara  Anni  Hana  Fiona  Mandy
  |   |   | 0.43866  0.08877  0.84629  0.69424 -0.10003
  |   |   | Barbara  Anni  Hana  Fiona  Mandy
  |   |   | 0.9475  0.7246  0.4452  0.6350 -0.4965
  | [7] age > 52: n = 30
  | Barbara  Anni  Hana  Fiona  Mandy
  | 0.2178 -1.3166 -0.3059 -0.2591 -0.2357
Bradley-Terry trees

Number of inner nodes: 3
Number of terminal nodes: 4
Number of parameters per node: 5
Objective function (negative log-likelihood): 1829
Bradley-Terry trees

Number of inner nodes: 3
Number of terminal nodes: 4
Number of parameters per node: 5
Objective function (negative log-likelihood): 1829

From scratch: Only need basic model fitting function because btmodel() provides all necessary methods.

R> btfit1 <- function(y, x = NULL, start = NULL, weights = NULL, offset = NULL, ...) {
+   btmodel(y, weights = weights, ...)
+ }
R> system.time(
+   bt1 <- mob(preference ~ gender + age + q1 + q2 + q3, data = Topmodel2007,
+             fit = btfit1)
+ )
  user  system elapsed
4.873   2.879   1.962
Bradley-Terry trees

**More efficient:** Extended model fitting function.

```r
R> btfit2 <- function(y, x = NULL, start = NULL, weights = NULL, offset = NULL, ..., 
+   estfun = FALSE, object = FALSE) {
+   rval <- btmodel(y, weights = weights, ..., estfun = estfun, vcov = object)
+   list(
+     coefficients = rval$coefficients,
+     objfun = -rval$loglik,
+     estfun = if(estfun) rval$estfun else NULL,
+     object = if(object) rval else NULL
+   )
+ }
R> system.time(
+   bt2 <- mob(preference ~ gender + age + q1 + q2 + q3, data = Topmodel2007,
+             fit = btfit2)
+ )

   user  system elapsed
1.407   0.467   1.064
```

Bradley-Terry trees

**Infrastructure:**
- Basics readily available: `print()`, `plot()`, `predict()`, `coef()`, ...
- Customizable, e.g., model-specific plots, predictions, ...

**Here:**
```
R> plot(bt2)
R> print(bt2)
R> print(bt2)
```
Model-based recursive partitioning (btfit2)

Model formula:
```
preference ~ gender + age + q1 + q2 + q3
```
Bradley-Terry trees

Fitted party:
[1] root
  | [2] age <= 52
  |   | [3] q2 in yes: n = 35
  |   |   Barbara  Anni  Hana  Fiona  Mandy
  |   |   1.3378  1.2318  2.0499  0.8339  0.6217
  |   | [4] q2 in no
  |   |   | [5] gender in male: n = 71
  |   |   |   Barbara  Anni  Hana  Fiona  Mandy
  |   |   |   0.43866  0.08877  0.84629  0.69424 -0.10003
  |   |   |   Barbara  Anni  Hana  Fiona  Mandy
  |   |   |   0.9475  0.7246  0.4452  0.6350 -0.4965
  | [7] age > 52: n = 30
  |   Barbara  Anni  Hana  Fiona  Mandy
  |   0.2178 -1.3166 -0.3059 -0.2591 -0.2357

Number of inner nodes: 3
Number of terminal nodes: 4
Number of parameters per node: 5
Objective function: 1829
Bradley-Terry trees

1. age
   - p < 0.001
   - ≤ 52
   - > 52

2. $q^2$
   - p = 0.017
   - yes
   - no

3. n = 35
   - Estimated parameters:
     - Barbara 1.3378
     - Anni 1.2318
     - Hana 2.0499
     - Fiona 0.8339
     - Mandy 0.6217

4. gender
   - p = 0.007
   - yes
   - no

5. male
   - n = 71
   - Estimated parameters:
     - Barbara 0.43866
     - Anni 0.08877
     - Hana 0.84629
     - Fiona 0.69424
     - Mandy −0.10003

6. female
   - n = 56
   - Estimated parameters:
     - Barbara 0.9475
     - Anni 0.7246
     - Hana 0.4452
     - Fiona 0.6350
     - Mandy −0.4965

7. n = 30
   - Estimated parameters:
     - Barbara 0.2178
     - Anni −1.3166
     - Hana −0.3059
     - Fiona −0.2591
     - Mandy −0.2357
Model-based random forests
Model-based random forests

Tree:

- **Idea:** Automatic detection of steps and abrupt changes.
- **Goal:** Capture non-linear and non-additive effects and interactions.
- **Result:** Yields $B$ subsamples $B_b$ with $b = 1, \ldots, B$ in which separate local models are estimated.
Model-based random forests

Tree:
- **Idea**: Automatic detection of steps and abrupt changes.
- **Goal**: Capture non-linear and non-additive effects and interactions.
- **Result**: Yields $B$ subsamples $B_b$ with $b = 1, \ldots, B$ in which separate local models are estimated.

Forest:
- **Idea**: Ensemble of $T$ trees based on resampling the learning data.
- **Goal**: Stabilization and regularization, smoother effects.
- **Strategies**: Bootstrap or subsamples. Random input variable sampling.
- **Result**: Yields subsamples $B^t_b$ with $b = 1, \ldots, B^t$ and $t = 1, \ldots, T$ for adaptive local model estimation.
Model-based random forests

Tree:
For predicting a (potentially new) observation \( z \) only consider observations corresponding to \( z_i \) in the learning data.

Forest:
Obtain a finer similarity measure between new observation \( z \) and \( z_i \).

Weights:
Average over trees, e.g., 2 out of 3 for \( z_i \).
Model-based random forests

**Tree:** For predicting a (potentially new) observation \( z \) only consider observations corresponding to \( z_i \) in the learning data.
Model-based random forests

Tree: For predicting a (potentially new) observation $z$ only consider observations corresponding to $z_i$ in the learning data.

Forest: Obtain a finer similarity measure between new observation $z$ and $z_i$. 

$z_i, Z$
Tree: For predicting a (potentially new) observation $z$ only consider observations corresponding to $z_i$ in the learning data.

Forest: Obtain a finer similarity measure between new observation $z$ and $z_i$.

Weights: Average over trees, e.g., 2 out of 3 for $z_i$. 

Model-based random forests
Model-based random forests

Parameter estimator for a global model with learning data \( \{(y_i, x_i)\}_{i=1,...,n} \):

\[
\hat{\theta} = \arg\min_{\theta \in \Theta} \sum_{i=1}^{n} w_i(z) \cdot \Psi(y_i, x_i, \theta)
\]
Model-based random forests

**Parameter estimator for a global**
model with learning data\(\{(y_i, x_i, z_i)\}_{i=1, \ldots, n}\):

\[
\hat{\theta}(z) = \arg\min_{\theta \in \Theta} \sum_{i=1}^{n} w_i(z) \cdot \Psi(y_i, x_i, \theta)
\]
Model-based random forests

Parameter estimator for a global model with learning data \( \{(y_i, x_i, z_i)\}_{i=1,...,n} \):

\[
\hat{\theta}(z) = \arg\min_{\theta \in \Theta} \sum_{i=1}^{n} w_i(z) \cdot \Psi(y_i, x_i, \theta)
\]

Weights:

\[
w_{i}^{\text{base}}(z) = 1
\]
Model-based random forests

Parameter estimator for an adaptive local model with learning data \{ \( (y_i, x_i, z_i) \) \}_{i=1,\ldots,n} :

\[
\hat{\theta}(z) = \arg\min_{\theta \in \Theta} \sum_{i=1}^{n} w_i(z) \cdot \Psi(y_i, x_i, \theta)
\]

Weights:

\[
w_{i}^{\text{base}}(z) = 1 \\
w_{i}^{\text{tree}}(z) = \sum_{b=1}^{B} I((z_i \in B_b) \land (z \in B_b))
\]
Model-based random forests

Parameter estimator for an adaptive local model with learning data \(\{(y_i, x_i, z_i)\}_{i=1,...,n}\):

\[
\hat{\theta}(z) = \arg\min_{\theta \in \Theta} \sum_{i=1}^{n} w_i(z) \cdot \Psi(y_i, x_i, \theta)
\]

Weights:

\[
w_i^{\text{base}}(z) = 1
\]

\[
w_i^{\text{tree}}(z) = \sum_{b=1}^{B} I((z_i \in \mathcal{B}_b) \land (z \in \mathcal{B}_b))
\]

\[
w_i^{\text{forest}}(z) = \frac{1}{T} \sum_{t=1}^{T} \sum_{b=1}^{B_t} \frac{1}{|\mathcal{B}_b^t|} I((z_i \in \mathcal{B}_b^t) \land (z \in \mathcal{B}_b^t))
\]
Model-based random forests

Software:

- `cforest()` based on `ctree()` in `partykit`.
- Redesign of `partykit` internals in development to facilitate “plug & play” trees and forests.
- `pmforest()` for personalized treatment effects in `model4you`.
- `traforest()` for transformation forests in `trtf`.
- `distforest()` for distributional forests in `disttree` on R-Forge.
- `circforest()` for circular forests in `circtree` on R-Forge.
Parties, Models, Mobsters

References
References


