A Series of Novel Ensemble Procedures with Tree Learners

Matt Corsetti
Advisor: Dr. Tanzy Love

Department of Biostatistics
and Computational Biology
University of Rochester

August 18, 2021

Overview

<u>Overview</u>:

- Brief Overview of Ensemble Methods
  - Tree learners
  - Popular preexisting procedures
- Grafted and Vanishing Random Subspaces
- Bagged Feature Weighted Random Forests
- Questions and Answers

Classification and Regression Trees (CART)

Background:

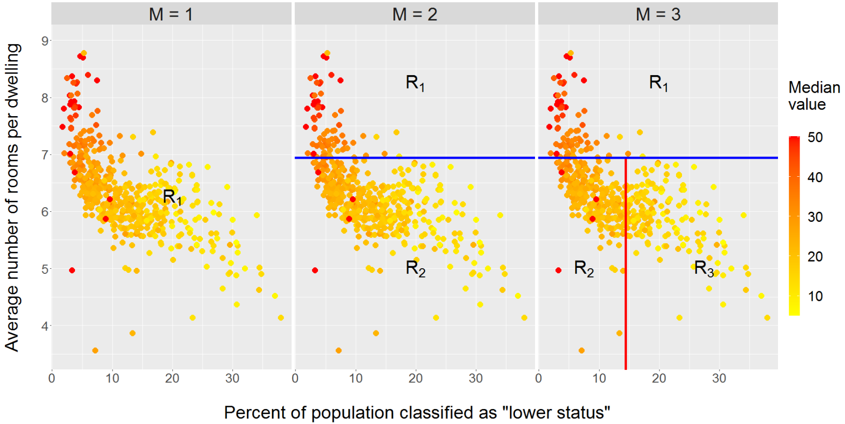
- Proposed by Leo Breiman in 1984
- **Problem**: Simple linear regression models often perform poorly with complex real-world data
- **Idea**: Try fitting simple regression models to different partitions of the covariate space to achieve a better fit
- **Solution**: Partition up the covariate space using a binary classification tree and fit a model to each subspace

Growing a Regression Tree:

- The data consists of p inputs and a response for each of N observations, that is, (x_i, y_i) for $i = 1, \ldots, N$, with $x_i = (x_{i1}, \ldots, x_{ip})$
- The algorithm sequentially identifies a variable on which to make a split/partition as well as the respective split point/value
- CART partitions the covariate space into M distinct, non-overlapping regions $R_1, \ldots, R_M$ and we model the response as a constant γ_m in each of the regions as follows:

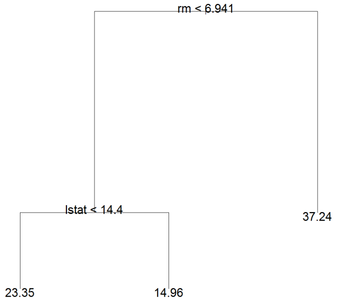
$$T(x) = \sum_{m=1}^{M} \gamma_m I(x \in R_m) \tag{1}$$

Boston Housing Data Example:



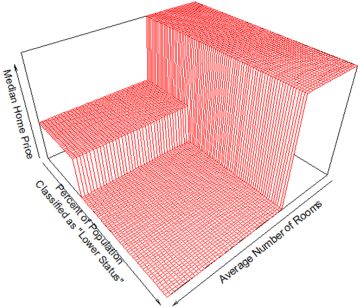
Classification and Regression Trees (CART)

- The first split (#rooms < 6.941) partitions the space into R_1 and R_2
- The second split (%lower status) further divides up the R_2 subspace



- The solution that minimizes the sum of squared errors uses the average of the y_i in the region R_m as the estimates for the γ_m's

$$\hat{\gamma}_m = \text{ave}(y_i | x_i \in R_M)$$



Classification and Regression Trees (CART)

How do we pick the splits?:

- Consider a split variable k and value at which to split it s and define the pair of half-planes

$$\min_{k,s}\left[\min_{\gamma_1}\sum_{x_i\in R_1(k,s)}(y_i-\gamma_1)^2+\min_{\gamma_2}\sum_{x_i\in R_2(k,s)}(y_i-\gamma_2)^2\right] \tag{2}$$

- Inner minimization is solved by $\hat{\gamma}_1=\text{ave}(y_i|x_i\in R_1(k,s))$ and $\hat{\gamma}_2=\text{ave}(y_i|x_i\in R_2(k,s))$
- The partition is made on the best available split (greedy) identified using equation 2, then the process is repeated

How/when do we stop?: Cost-Complexity Pruning

$$\sum_{m=1}^{|M|}\sum_{x_i\in R_m}(y_i-\hat{\gamma}_m)^2+\alpha|M| \tag{3}$$

Ensemble Methods

- **Ensemble learning**: methods that join together "simple" models or "weak" learners to form a committee or ensemble
- Ensembles leverage the combined strength of their base models to achieve increased predictive performance greater than that of the individual learners
- Generally speaking, ensembles are made stronger when there is disagreement and very little correlation among the learners
- "Diversity and independence are important because the best collective decisions are the product of disagreement and contest, not consensus or compromise." -James Surowiecki, The Wisdom of Crowds

Bootstrap Aggregation (Bagging)

Bagging:

- Ensemble procedure that reduces variance in the estimate $\hat{f}(x)$ by averaging over predictions from individual trees (reduces variance and leaves bias unchanged)
- Bagging with trees:
  - Draw B bootstrapped samples from the data $\boldsymbol{Z} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$
  - For each bootstrap sample $\boldsymbol{Z}_b$, $b = 1, \ldots, B$, fit a tree $T(x; \theta_b)$ and obtain predictions, then average predictions across trees:

$$\hat{f}_{\text{Bagged}}(x) = \frac{1}{B} \sum_{b=1}^{B} T(x; \theta_b), \tag{4}$$

  where θ_b characterizes the b^{th} tree (split variables, cut points, terminal node values)

Random Subspaces

Random Subspaces:

- Proposed by Tin Kam Ho in 1998, also known as "attribute bagging"
- Random subspaces with trees:
  - Draw B randomly chosen subsets of the predictor variables (referred to as "feature subsets") from the data, each of size $r < p$
  - For each feature subset $X^b_{\{n \times r\}}$ $b = 1, \ldots, B$ fit a tree $T(x; \theta_b)$ and obtain the predictions, then average predictions over trees:

$$\hat{f}_{\text{RSM}}(x) = \frac{1}{B} \sum_{b=1}^{B} T(x; \theta_b) \tag{5}$$

- Building trees on different feature subsets can reduce correlation between trees making for a stronger ensemble

Random Forests:

- Proposed by Leo Breiman in 2001, average over trees
- Improves the variance reduction of bagging by reducing correlation among trees in the ensemble
- RF achieves this by randomly selecting mtry $\leq p$ of the input variables as split candidates before each split
- Reducing mtry will reduce the correlation between trees thereby reducing the variance in the average

$$\hat{f}_{\text{RF}}(x) = \frac{1}{B} \sum_{b=1}^{B} T(x; \theta_b) \tag{6}$$

Boosting:

- Propsed by Robert Schapire in 1990, sum of trees ensemble
- Fit tree $T(x, \theta_b)$ to residuals from ensemble consisting of all trees that came before it (instead of y):

$$\hat{\theta}_b = \underset{\theta_b}{\text{argmin}} \left[\sum_{i=1}^{n} L(y_i, f_{b-1}(x_i) + T(x_i, \theta_b)) \right], \tag{7}$$

- Fitting each tree to ensemble residuals allows ensemble to improve in areas where it performs poorly
- Boosted tree model is the sum over these trees

$$\hat{f}_{\text{Boost}}(x) = \sum_{b=1}^{B} T(x; \theta_b) \tag{8}$$

Boosting

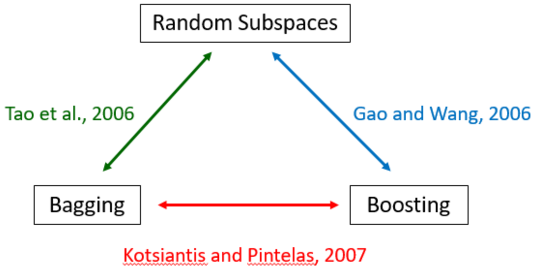
- Shrunken version of new tree is added to the ensemble

$$f_b(x) = f_{b-1}(x) + \omega T(x_i, \theta_b), \quad 0 \leq \omega \leq 1 \tag{9}$$

- Shrinkage prevents any one tree from being overly influential
- Shrinkage parameter ω controls rate at which boosting learns (smaller ω values with large forests sizes tend to work well)

Combining Ensemble Methods

- Much work has been done in combining data-partitioning methods with feature-partitioning methods and also with boosting



Overview:

- Ensemble Methods
  - Tree learners
  - Popular preexisting procedures
- Grafted and Vanishing Random Subspaces
- Bagged Feature Weighted Random Forests
- Questions and answers

Motivating Statement

Problem:

- Procedures that use random sampling of the input variables for split candidates (e.g. Random Forests, RSM) suffer when the # of truly informative features s is small relative to p
- Feature (input variable) subsets are likely to contain many non-informative features
- Learners built on these subsets can be harmful to ensemble

Solution:

- Allow each tree to share information regarding variable importance in its feature subset with the trees that come after it in the ensemble

Publication Status: In submission *Pattern Analysis and Applications* (3rd round review)

Grafting and Vanishing Random Subspaces

Grafting Random Subspaces (GRS):

1. Grow tree on bootstrapped sample and randomly chosen feature subset of size $r < p$
2. Identify most important variable in feature subset and recycle it across next q subsets

Main Idea:

- Grafting allows new trees to reuse informative features
- New trees explore how informative features interact with each other and features not yet randomly sampled

Grafting and Vanishing Random Subspaces

Vanishing Random Subspaces (VRS):

1. Grow tree on bootstrapped sample and randomly chosen feature subset of size $r < p$
2. Identify least important variable in feature subset
3. Temporarily exile feature from next q subsets

Main Idea:

- Exiling uninformative features creates a narrower but more enriched pool of variable candidates

Estimators:

- Both GRS and VRS take the average over trees as the estimator

$$\hat{f}_{\mathrm{GRS/VRS}}(x) = \frac{1}{B}\sum_{b=1}^{B} T(x;\theta_b) \tag{10}$$

- We also propose boosted versions of either algorithm (BoGRS and BoVRS) which are sum-of-shrunken-trees models

$$\hat{f}_{\mathrm{BoGRS/BoVRS}} = \sum_{b=1}^{B} \omega T(x;\theta_b) \tag{11}$$

Variable Importance

We consider three measures of variable importance:

1. **First split** - variable used to split root node
2. **Split frequency** - how often a variable is split on
3. **Contribution to explained deviance**

Contribution to explained deviance

- Each non-terminal node has an associated deviance

$$D_v = \sum_{x_i \in R_v} (y_i - \hat{\gamma}_v)^2$$

where $v = 1, \ldots, V$ indexes non-terminal nodes

- The gain in explained deviance from splitting a node is defined as

$$\Delta = \frac{D_{\text{parent}} - (D_{\text{left child}} + D_{\text{right child}})}{D_{\text{root}}}$$

- Contribution to explained deviance for variable j is the sum of the gains in explained deviance from non-terminal nodes that split on variable j

$$\text{AggDev}_j = \sum_{v=1}^{V} \Delta_v I(\text{node } v \text{ splits on variable } j)$$

Variable Importance

Optimal pairings:

1. **VRS**: deviance + least important
2. **VRS with boosting**: deviance + most important
3. **GRS**: most commonly split + most important
4. **GRS with boosting**: most commonly split + most important

The Grafting and Vanishing Parameter

Grafting/Vanishing Parameter q:

- q determines for how many successive feature subsets the variable is grafted or exiled

$$q_b \sim \max\{1, \text{Pois}(\sqrt{p}/2)\}$$

- Draw q_b after constructing the b^{th} learner
- Future work:
  - We'd like to make q_b a function of both p and B (outside the scope of this paper)

Simulation Design

We follow the simulation design of Hastie et al. 2017

Simulation parameters:

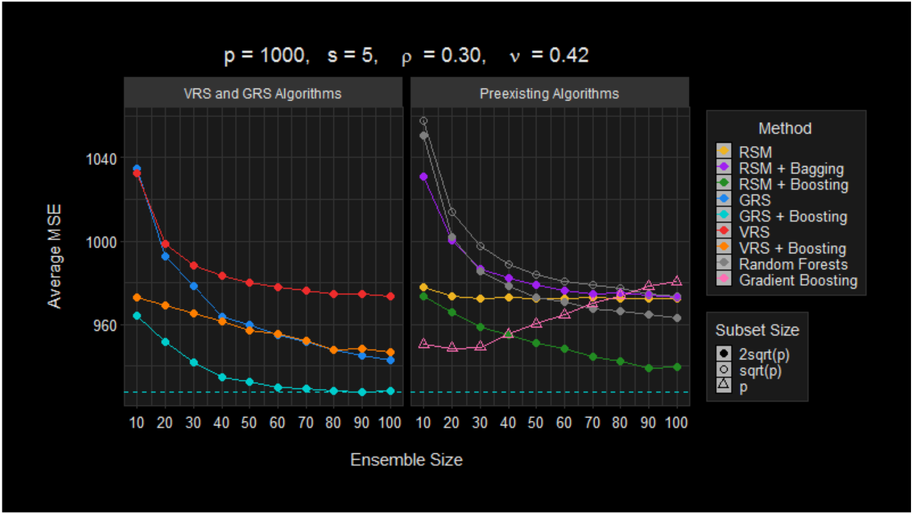
- $n = 100$ (fixed number of observations)
- $p = \{10, 100, 1000\}$ (number of predictor variables)
- $s = \{5, 50\}$ (sparsity level–number of truly informative variables)
- $\rho = \{0.30, 0.70\}$ (predictor autocorrelation level)
- $\nu = \{0.05, 0.42, 2.07\}$ (signal-to-noise level)
- 30 unique simulation settings

Simulation Design

Simulation data:

1. $\beta_0 =$ vector of true coefficients (first s values are sequence from 10 to 0.5)
2. Predictor matrix $X \sim N_p(0, \Sigma)$ i.i.d. where Σ has entry (i, j) equal to $\rho^{|i-j|}$
3. Response vector $y \sim N_n(X\beta_0, \sigma^2 I)$, where σ^2 defined to meet desired SNR level, $\sigma^2 = \beta_0^T \Sigma \beta_0 / \nu$
4. Acquire each ensemble's prediction error on validation set $(\tilde{X}, \tilde{y})$
5. Repeat these steps 1,000 times and average prediction error across repeats

Simulation Results



Simulation Results

p	s	ρ	ν	Best New Algorithm Method	B	Mean	Best Existing Algorithm Method	B	Mean	Paired T P-value	Result
1000	5	0.3	0.05	BoVRS	10	6119.44	BoRSM	10	6123.16	0.201	W
1000	5	0.3	0.42	BoGRS	100	928.24	BoRSM	90	939.18	0.000	W
1000	5	0.3	2.07	BoGRS	100	255.50	GBoost	90	260.89	0.000	W
1000	5	0.7	0.05	BoVRS	10	10141.88	BoRSM	10	10145.83	0.441	W
1000	5	0.7	0.42	BoGRS	100	1426.37	BoRSM	100	1455.47	0.000	W
1000	5	0.7	2.07	BoGRS	90	341.69	GBoost	50	357.14	0.000	W
1000	50	0.3	0.05	BoVRS	10	67936.47	BoRSM	10	67969.53	0.321	W
1000	50	0.3	0.42	BoVRS	10	10901.81	BoRSM	10	10912.24	0.082	W
1000	50	0.3	2.07	GRS	100	4627.08	GBoost	90	4619.40	0.510	L
1000	50	0.7	0.05	BoVRS	10	194650.78	BoRSM	10	194666.26	0.876	W
1000	50	0.7	0.42	GRS	100	29952.41	RF	100	30061.13	0.001	W
1000	50	0.7	2.07	BoGRS	100	10282.63	GBoost	90	9459.75	0.000	L

Table: HTT Simulation Results with CART-Based Ensembles

- At least one of our new procedures outperformed all preexisting ensemble competitors in 17 of the 30 simulation settings (7 by a statistically significant margin)

Experimental Design

Experimental Design:

- 200 individual experiments carried out on each of 12 real datasets
- Training/test set split of 2/3:1/3 drawn at onset of each experiment
- Ensemble predictive performances recorded at $B = 10$ to $B = 100$ trees in increments of 10
- MSE averaged across 200 individual experiments for each dataset and compared using paired T-tests

Experimental Results:

- New CART-based procedures outperformed preexisting ensembles in 6 of the 12 datasets (4 by statistically significant margin)

Experimental Results - Iranian Housing

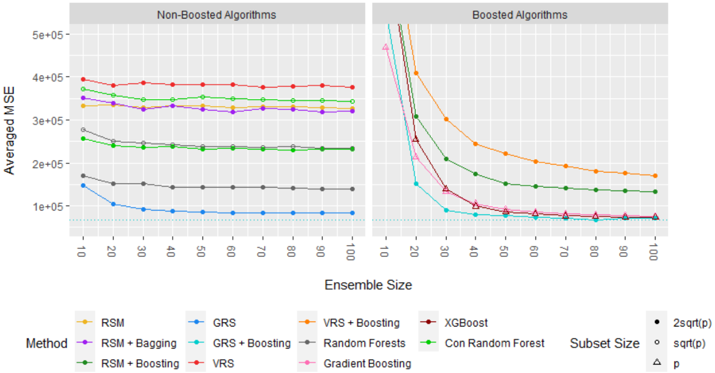


Figure: Iranian Housing Performance Results ($n = 372, \; p = 103$)

Experimental Results - Gait Speed

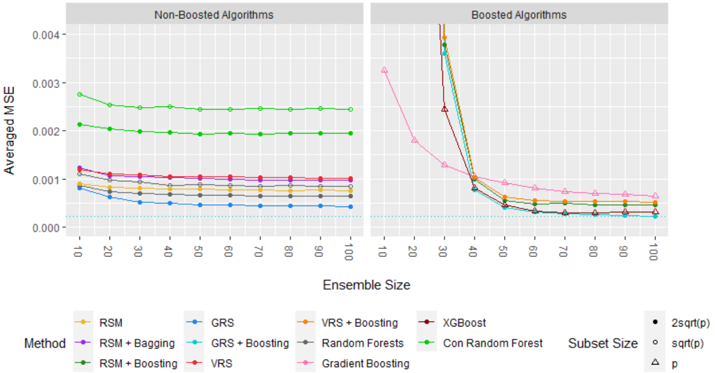


Figure: Gender Gait Speed Performance Results $(n = 48,\ p = 321)$

Summary of Findings

Summary:

- Grafting is better when there are few informative features
- Exiling tends to be better in situations with more informative features
- Boosted versions of GRS and VRS tended to outperform their non-boosted counterparts
- Grafting tends to be the more promising of the two
- Future work: combine grafting and vanishing into one algorithm

Overview:

- Ensemble Methods
  - Tree learners
  - Popular preexisting procedures
- Grafted and Vanishing Random Subspaces
- Bagged Feature Weighted Random Forests
- Questions and answers

Motivating Statement

Problem: (similar motivating problem as GRS/VRS)

- Random Forests suffers in settings where the number of truly informative features s is small relative to p
- When drawing feature subsets to split the nodes, many uninformative features will be randomly selected
- Sub-optimal solution: increase the mtry parameter to increase chance of including valuable predictors–increases computational burden

Solution:

- Use weighted random sampling instead of simple random sampling to draw feature subsets, with weights tilted in favor of informative features
- Establish weights in a pre-processing step before growing forest

Motivating Statement

Solution (continued):

- Using weighted random sampling to select the feature subsets for Random Forests is not a new idea:
  - Enriched Random Forests (Amaratunga et al., 2008)
  - Feature-Weighted Random Forests (Ye et al. 2008)
  - Iterative Random Forests (Basu et al., 2018)
- **New Idea**: use ensemble methods (specifically bagging) to establish better estimates of the feature weights in the pre-processing stage

Publication Status: In submission *Machine Learning*

Bagged Approach

Bagged Approach:

1. Draw Q bootstrapped samples from the training data
2. For each bootstrapped sample $\boldsymbol{Z}_q$, where $q = 1, \ldots, Q$, apply the same feature-weighting algorithm (e.g. ReliefF) to the bootstrapped sample and extract the p-vector of feature weights denoted w_q
3. Average over weight vectors to get ensemble estimate

$$\hat{w}(x, y) = \frac{1}{Q} \sum_{q=1}^{Q} w_q(x, y) \tag{12}$$

4. Construct the random forest using feature-weighted random sampling with the bagged ensemble-generated feature weights $\hat{w}(x, y)$

ReliefF Feature Weights

ReliefF Algorithm (Kononenko et al., 1997):

- An extension of the original Relief algorithm (Kira and Rendell, 1992) capable of handling missing data and multi-class problems
- Key idea of all Relief-based algorithms: estimate a variable's importance according to how well their values distinguish among observations that are near each other
- The algorithm should estimate the ability of attributes to separate each pair of classes regardless of which two classes are closest to each other
- ReliefF searches for k near misses from each different class and averages their contributions for updating W, weighted with the prior probability of each class

Simulation Design

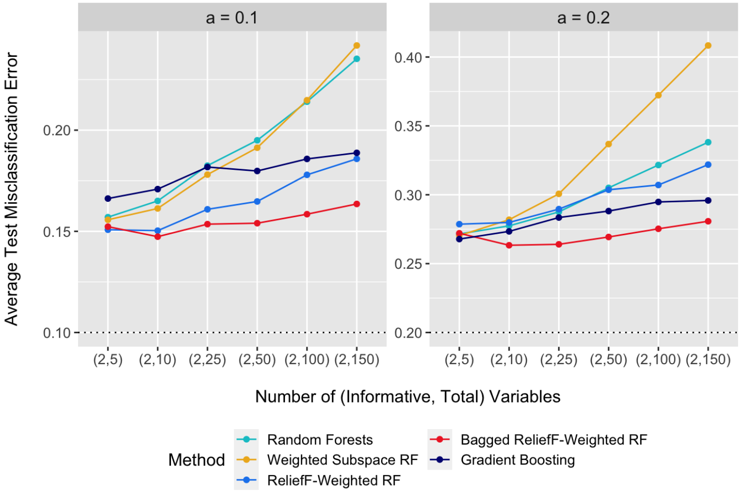
Simulation data:

- We use the simulation model of Mease and Wyner (2008)
- Probabilities of class membership for a binary response variable are generated using the model

$$P(y = 1|x) = a + (1 - 2a) \cdot I\left[\sum_{j=1}^{s} x_j > s/2\right]. \qquad (13)$$

- Input features follow a multivariate uniform distribution, $X \sim U[0,1]^p$
- Constant a denotes the Bayes error rate such that $0 \leq a \leq 1/2$
  - Bayes error rate is the best possible error rate an estimator could achieve (oracle error rate)
- Draw $n_{\text{train}} = 300$ and $n_{\text{test}} = 500$, 200 separate times for each combination of $a = \{0.1, 0.2\}$, $p = \{5, 10, 25, 50, 100, 150\}$, and $s = 2$ and averaged the misclassification error rates

Simulation Results



Experimental Design

Experimental Design:

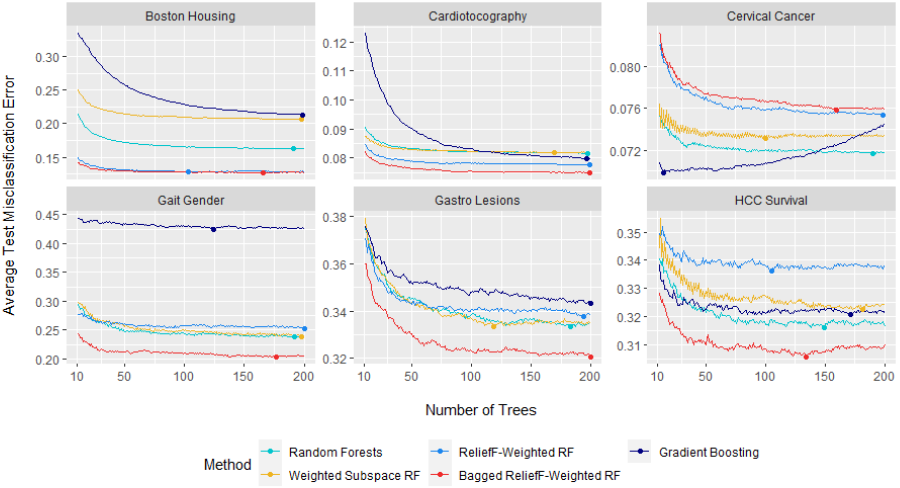
- 200 individual experiments carried out on 12 real datasets
- Training/testing split of 25%:75% drawn at onset of each experiment
- Average misclassification error rate trajectories recorded at ensemble sizes ranging from $B = 10$ through $B = 200$

Experimental Design

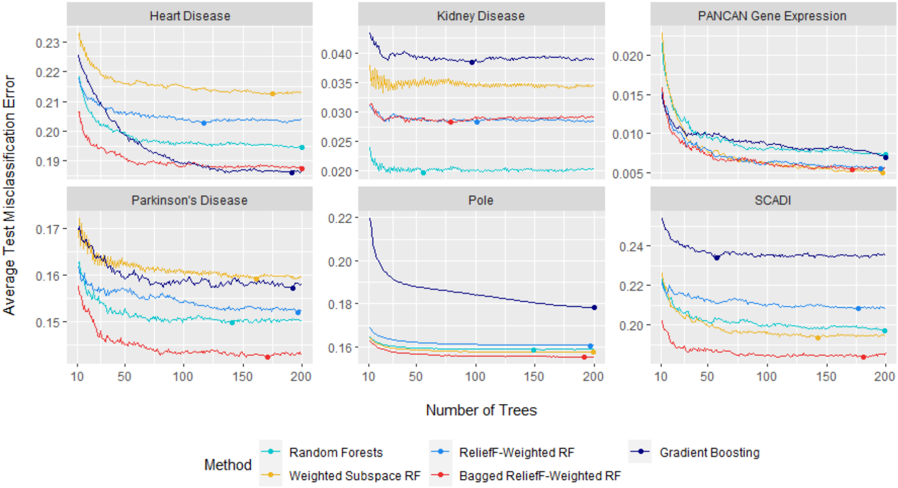
Dataset	n	p	Classes	% Largest Class	% Smallest Class
Boston Housing	506	13	9	26.09	3.36
Cardiotocography	2126	21	3	77.85	8.28
Cervical Cancer	668	28	2	93.26	6.74
Gait Gender	48	321	2	56.25	43.75
Gastro Lesions	152	699	3	52.63	19.74
HCC Survival	121	33	2	63.64	36.36
Heart Disease	297	13	2	53.87	46.13
Kidney Disease	242	19	2	50.41	49.59
PANCAN Gene	801	5000	5	37.45	9.73
Parkinson's Disease	195	22	2	75.38	24.62
Pole	5000	26	11	62.16	1.66
SCADI	70	206	4	41.43	14.29

Table: UCI data sets summary statistics

Experimental Results



Experimental Results



Experimental Results:

- BRWRF was the best performer on 8 of the 12 datasets (7 by statistically significant amounts)
- Key performance pattern occurs in 7 of the 12 datasets
  - Traditional Random Forests outperforms ReliefF Weighted Random Forests
  - Bagged ReliefF Weighted Random Forests outperforms traditional Random Forests
  - Demonstrates the need for better methods to estimate feature weights

Acknowledgements

Dr. Tanzy Love

Dr. Sally Thurston

Dr. Andrew McDavid

Dr. Jiebo Luo

Additional Thanks:

- Dr. Edwin van Wijngaarden and the SCDS Team
- T32 Training Grant (T32ES007271)
- Dr. Michael McDermott, Dr. Derick Peterson and Karin Gasaway

References

1. Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). Classification and regression trees. Boca Raton, FL: CRC Press.
2. Ho, T. K., (1995). Random Decision Forests. Proceedings of the 3rd International Conference on Document Analysis and Recognition, 278-282.
3. Hastie, T., Tibshirani, R., & Friedman, J. H. (2004). The elements of statistical learning: Data mining, inference, and prediction: With 200 full-color illustrations. New York: Springer.
4. Schapire, R. E. (1990). The strength of weak learnability. Machine Learning, 5(2), 197-227.

Questions and Answers

Thank you