Introduction to the bartMachine R package

Saint Louis R User Group

John Snyder

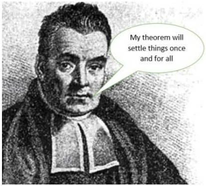
April 11, 2019

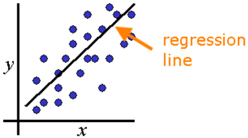
Outline

1. Brief BART overview
2. Installation and features
3. Demo
4. Further Considerations

What is BART?

Bayesian **A**dditive **R**egression **T**rees





How it works

- Emsamble method which is the sum of many shallow trees.
- Complexity is regularized via Bayesian "priors."
  - This frees us from ad hoc decisions
- Uses "Bayesian Backfitting"
  - Each tree is sequentially exposed to the residuals when all other trees are used to predict

Results:

- Each tree describes a tiny amount of the structure
- The Bayesian structure means variation is fully quantified.
  - Intervals, p-values, and model selection oh my!
- Outperforms many common models in out of sample prediction.

Powerful Predictive Performance

- Test RMSE of 100 random datasets simulated from various nonlinear functions (added noise with s=1)

Function	BART	XGBoost*	Random Forest*	Linear Reg(lol)
Friedman	1.08	1.21	1.64	2.61
Mirsha's Bird	1.53	2.78	2.90	26.59
Weird Exp	1.04	1.05	1.07	6.08
Linear	1.025	1.032	1.034	1.004

- bartMachine is relatively unknown
  - xgboost: ~43k downloads per month
  - randomForest: ~88k downloads per month
  - bartMachine: ~2k downloads per month

Package Features:

- Functions for Cross Validation
- Model fitting:
  - Is done in parallel[1]
  - Can incorporate missing data
- Lots of fun statistical things
  - Credible iterval calculation
  - Diagnostic plots/tests
- Variable selection
- Interaction detection
- Export fit trees

[1] MCMC sampling is used, so speedups during model fitting aren't great

Installation and loading steps

1. Google "How to install rJava on [your OS]"
2. Do that
3. Run the following

```r
install.packages("bartMachine")
```

To load the package with:

- 10GB of memory
- All but one core available for compute

```r
options(java.parameters = "-Xmx10g")
library(bartMachine)
numcores <- parallel::detectCores()
set_bart_machine_num_cores(numcores - 1)
```

Boston Data

```
data(Boston)
Boston %>% round(digits=2) %>% head
```

```
##   crim zn indus chas  nox   rm  age  dis rad tax ptratio  black lstat medv
## 1 0.01 18  2.31    0 0.54 6.58 65.2 4.09   1 296    15.3 396.90  4.98 24.0
## 2 0.03  0  7.07    0 0.47 6.42 78.9 4.97   2 242    17.8 396.90  9.14 21.6
## 3 0.03  0  7.07    0 0.47 7.18 61.1 4.97   2 242    17.8 392.83  4.03 34.7
## 4 0.03  0  2.18    0 0.46 7.00 45.8 6.06   3 222    18.7 394.63  2.94 33.4
## 5 0.07  0  2.18    0 0.46 7.15 54.2 6.06   3 222    18.7 396.90  5.33 36.2
## 6 0.03  0  2.18    0 0.46 6.43 58.7 6.06   3 222    18.7 394.12  5.21 28.7
```

- Target: Predict home value(medv)

Fitting BART

```
y <- Boston$medv
X <- Boston %>% dplyr::select(-c("medv"))

#Fit BART model
bart.model <- bartMachine(X,y,
                          num_trees = 200,
                          num_burn_in = 1000,
                          num_iterations_after_burn_in = 5000)
```

BART is fit with MCMC, which requires a “burnin” set of initial iterations.

- NOTE: with 10 cores, each thread would fit $1000 + 5000/10$ or 1500 iterations

BART object

```
> bart.model
bartMachine v1.2.3 for regression

training data n = 506 and p = 13
built in 19.4 secs on 31 cores, 200 trees, 1000 burn-in and 5000 post.

sigsq est for y beforehand: 21.938
avg sigsq estimate after burn-in: 2.70319

in-sample statistics:
 L1 = 430.35
 L2 = 654.72
 rmse = 1.14
 Pseudo-Rsq = 0.9847
p-val for shapiro-wilk test of normality of residuals: 0
p-val for zero-mean noise: 0.98467
```

Did we overfit?

K fold CV using our fit model

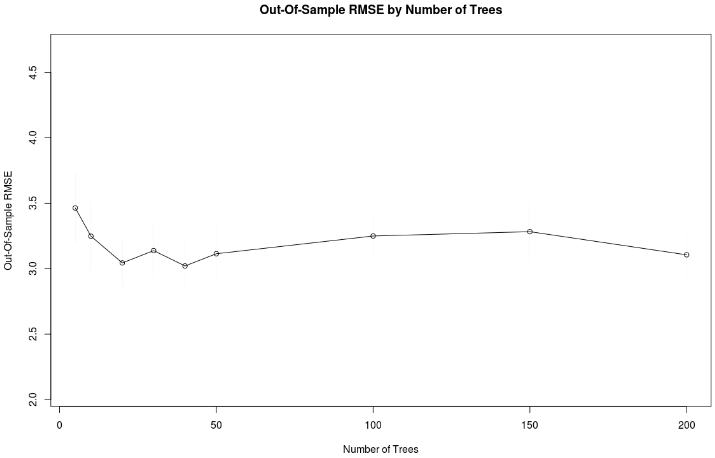
```
k_fold_cv(X, y, k_folds = 10,
          num_trees = 200,
          num_burn_in = 1000,
          num_iterations_after_burn_in = 5000)
```

```
..........
$y_hat
  [1] 24.932807 20.522438 31.149051 36.071931...

$L1_err
[1] 1022.238

$L2_err
[1] 4624.004

$rmse
[1] 3.02297

$PseudoRsq
[1] 0.8917508

$folds
  [1]  8 10  3  1 ....
```

Select the number of Trees

```
rmse_by_num_trees(bart.model, num_replicates = 20)
```

- This fits the model 20 times for a number of various numbers of trees
- Aggregates out of sample RMSE



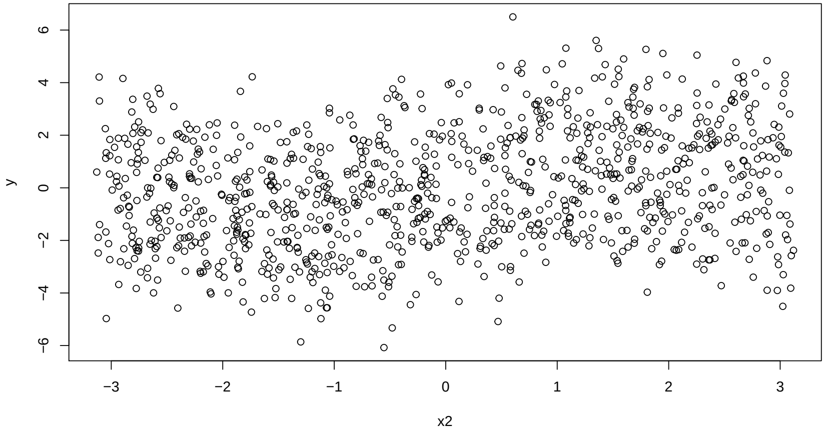
Cross Validate

```
bart.model.cv <- bartMachineCV(X, y,
                               num_burn_in = 1000,
                               num_iterations_after_burn_in = 5000)
```

- To automatically cross validate, we just need to add a CV to the end of the function!
- This takes a while

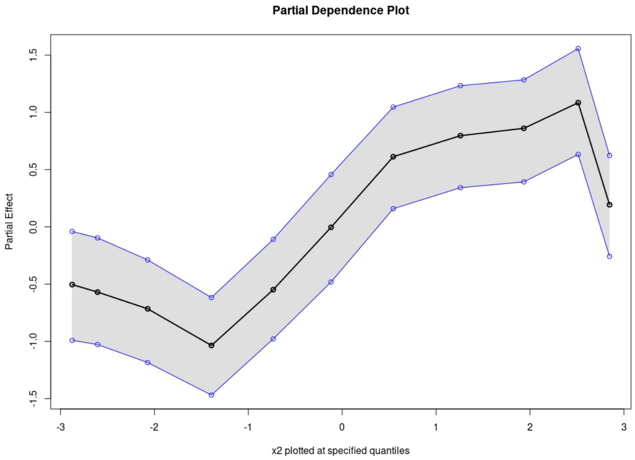
Aside: Partial Dependence

```r
nsim <- 1000
x1 <- runif(nsim,-3.14,3.14)
x2 <- runif(nsim,-3.14,3.14)

y <- x1+sin(x2) + rnorm(nsim)
plot(x2,y)
```



Aside: Partial Dependence

```
bart.model <- bartMachine(data.frame(x1,x2),y,
                          num_trees = 200,
                          num_burn_in = 1000,
                          num_iterations_after_burn_in = 4000)
pd_plot(bart.model, j = "x2")
```



Code Time

Coding demo

John's Final Thought

- BART is a powerful technique which brings many advantages
  - At the expense of computational efficiency.

```r
# n is 10000, p is 100
bart.model <- bartMachine(X,y,
                          num_trees = 100,
                          num_burn_in = 1000,
                          num_iterations_after_burn_in = 5000,
                          mem_cache_for_speed = TRUE)
```

- Statistical advantages are numerous
- Great for small to mid sized data
- Good results with removing expected variation and feeding residuals into BART.