Discovering Regression Structure with a Bayesian Ensemble

Ed George, University of Pennsylvania
(joint work with H. Chipman and R. McCulloch)

30 Years of Bootstrap & Recent Advances in Statistics
Rutgers University
December 4, 2009
A General Model Free Regression Setup

- Data: \( n \) observations on \( y \) and \( x = (x_1, \ldots, x_p) \)
- Suppose: \( Y = f(x) + \varepsilon, \quad \varepsilon \) symmetric around 0
- Unknowns: \( f \) and the distribution of \( \varepsilon \)

A Bayesian Ensemble approach can help us to:
- estimate \( f(x) = E(Y \mid x) \)
- obtain posterior intervals for \( f(x) \)
- estimate the effect of a particular \( x_j \)
- select an informative subset of \( x_1, \ldots, x_p \)
  (making no assumptions about \( f \))

Remark: In what follows we will assume \( \varepsilon \sim N(0, \sigma^2) \) for simplicity, but extension to a general DP process normal mixture model for \( \varepsilon \) works just fine.
A General Bayesian Ensemble Idea

Approximate \( f(x) = \mathbb{E}(Y | x) \) by something of the form

\[
\mathbb{E}(Y| x, \theta) = g(x; \theta_1) + g(x; \theta_2) + \ldots + g(x; \theta_m)
\]

\( \theta_1, \theta_2, \ldots, \theta_m \) iid \( \sim \pi \)

and use Bayes rule to obtain \( \mathbb{E}(Y| x, \text{data}) \)

This can work well when \( \pi \) is calibrated (using the data \( y \)) so that for every \( x \)

\[
\text{Var} \left[ g(x;\theta_1) + \ldots + g(x; \theta_m) \right] \approx \text{Var}(y)
\]

\[
[g(x;\theta_1) + \ldots + g(x; \theta_m)] \in (y_{\text{max}}, y_{\text{min}}) \text{ with high probability}
\]
Let's consider a Regression Tree for $g(x; \theta)$

Let $g(x; \theta), \theta = (T, M)$ be the regression tree function which assigns a $\mu$ value to $x$

$T$ denotes the tree structure including the decision rules

$M = \{\mu_1, \mu_2, \ldots, \mu_b\}$ denotes the set of terminal node $\mu$'s.

BART (Bayesian Additive Regression Trees) is a Bayesian Ensemble of such regression trees
Let $\theta = ((T_1,M_1), (T_2,M_2), \ldots, (T_m,M_m))$ identify a set of $m$ trees and their $\mu$'s.

$$Y = g(x;T_1,M_1) + g(x;T_2,M_2) + \ldots + g(x;T_m,M_m) + \sigma z, \quad z \sim N(0,1)$$

$E(Y \mid x, \theta)$ is the sum of all the corresponding $\mu$'s at each tree bottom node.

Such a model combines additive and interaction effects.
Completing the Model with a Regularization Prior

\[ Y = g(x;T_1,M_1) + ... + g(x;T_m,M_m) + \sigma z, \quad z \sim N(0,1) \]

\[ \pi((T_1,M_1),...(T_m,M_m),\sigma) \]

g(x;T_1,M_1), g(x;T_2,M_2), ..., g(x;T_m,M_m) is a highly redundant “over-complete basis” with many many parameters

We refer to \( \pi \) as a regularization prior because it is deliberately chosen to keep each \( g(x;T_i,M_i) \) effect small.
**BART Implementation**

Because BART is a fully Bayesian specification, information about all the unknowns, namely $\theta = ((T_1,M_1),..,(T_m,M_m),\sigma)$, is captured by the posterior

$$\pi(\theta | y) \propto p(y | \theta) \pi(\theta)$$

Thus, to implement BART we need to do

1. Construct the prior $\pi(\theta)$
   - Independent tree generating process on $T_1,..,T_m$
   - Use observed $y$ to properly scale $\pi(M | T)$
   - Extremely effective default choice is available

2. Calculate the posterior $\pi(\theta | y)$
   - Bayesian backfitting MCMC
   - Interweaving marginalization and regeneration of $\theta$

R package BayesTree available on CRAN
\[ Y = g(x;T_1,M_1) + \ldots + g(x;T_m,M_m) + \sigma z \]

plus

\[ \pi((T_1,M_1),\ldots,(T_m,M_m),\sigma) \]

**Connections to Other Modeling Ideas**

**Bayesian Nonparametrics:**
- Lots of parameters (to make model flexible)
- A strong prior to shrink towards simple structure (regularization)
- BART shrinks towards additive models with some interaction

**Dynamic Random Basis:**
- \( g(x;T_1,M_1), \ldots, g(x;T_m,M_m) \) are dimensionally adaptive

**Gradient Boosting:**
- Overall fit becomes the cumulative effort of many “weak learners”
\[
Y = g(x;T_1,M_1) + ... + g(x;T_m,M_m) + \sigma z \\
\text{plus} \\
\pi((T_1,M_1),...,(T_m,M_m),\sigma)
\]

Some Distinguishing Features of BART

BART is NOT Bayesian model averaging of a single tree model.

Unlike boosting, BART uses a FIXED number of trees \( m \).

Choose \( m \) large for best estimation of \( E[Y|x] \) and prediction
More trees yields more approximation flexibility

Choose \( m \) small for variable selection
Fewer trees forces the \( x \)'s to compete for entry
Estimation Comparison on 42 datasets

Out-sample-performance compared for 6 methods

Neural networks (single layer)
Random Forests
Boosting (Friedman's gradient boosting machine)
Linear regression with lasso
BART (Bayesian Additive Regression Trees)
BART/default - *NO* tuning of parameters

Data from Kim, Loh, Shih and Chaudhuri (2006)
Up to 65 predictors and 2953 observations

Train on 5/6 of data, test on 1/6
Tuning via 5-fold CV within training set
20 Train/Test replications per dataset
Relative RMSE Performance Ratios to Best

Performance Quantiles

<table>
<thead>
<tr>
<th>Method</th>
<th>(50%, 75%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lasso</td>
<td>(1.196, 1.762)</td>
</tr>
<tr>
<td>Boosting</td>
<td>(1.068, 1.189)</td>
</tr>
<tr>
<td>Neural Net</td>
<td>(1.055, 1.195)</td>
</tr>
<tr>
<td>Random Forest</td>
<td>(1.053, 1.181)</td>
</tr>
<tr>
<td>BART-default</td>
<td>(1.055, 1.164)</td>
</tr>
<tr>
<td>BART-cv</td>
<td>(1.037, 1.117)</td>
</tr>
</tbody>
</table>

BART is a very strong performer!
Example: Friedman’s Simulated Data

\[ Y = f(x) + \sigma z, \quad z \sim N(0,1) \]

where

\[ f(x) = 10 \sin (\pi x_1 x_2) + 20(x_3 - .5)^2 + 10x_4 + 5x_5 + 0x_6 + \ldots + 0x_{10} \]

10 x's, but only the first 5 matter!

Friedman (1991) used \( n = 100 \) observations from this model with \( \sigma = 1 \) to illustrate the potential of MARS
Comparison of BART with Other Methods

Performance measured on 1000 out-of-sample x’s by

$$\text{RMSE} = \sqrt{\frac{1}{1000} \sum_{i=1}^{1000} (\hat{f}(x_i) - f(x_i))^2}$$

<table>
<thead>
<tr>
<th>Method</th>
<th>average RMSE</th>
<th>se(RMSE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Forests</td>
<td>2.655</td>
<td>0.025</td>
</tr>
<tr>
<td>Linear Regression</td>
<td>2.618</td>
<td>0.016</td>
</tr>
<tr>
<td>Neural Nets</td>
<td>2.156</td>
<td>0.025</td>
</tr>
<tr>
<td>Boosting</td>
<td>2.013</td>
<td>0.024</td>
</tr>
<tr>
<td>MARS</td>
<td>2.003</td>
<td>0.060</td>
</tr>
<tr>
<td>BART-cv</td>
<td>1.787</td>
<td>0.021</td>
</tr>
<tr>
<td>BART-default</td>
<td>1.759</td>
<td>0.019</td>
</tr>
</tbody>
</table>
Applying BART to the Friedman Example

We applied BART with \( m = 100 \) trees to \( n = 100 \) observations of the Friedman example.

95\% posterior intervals vs true \( f(x) \)

\( \hat{f}(x) \)

in-sample \( f(x) \)

classified observations

Red
\( m = 1 \) model

Blue
\( m = 100 \) model

\( \sigma \) draws

out-of-sample \( f(x) \)

MCMC iteration
With only 100 observations on y and 1000 x's, BART yielded "reasonable" results !!!!

Added many useless x's to Friedman's example
High Dimensional Out-of-Sample RMSE Performance Comparisons

$\text{RMSE}$ Performance Comparisons

$p = 10$                              $p = 100$                           $p = 1000$

$n = 100$ throughout!
The Effect of Using More Trees in BART

Rapid decrease in RMSE followed by slow increase unless signal too weak
When \( Y = g(x; T_1, M_1) + ... + g(x; T_m, M_m) + \sigma z \) is fit to data, we can count how many times a predictor is used in the trees.

For example, in the tree here, \( x_2 \) and \( x_5 \) are each used once.

The importance of each \( x_k \) can thus be measured by its overall usage frequency.

This approach is most effective when \( m \) is small.
Variable Selection via BART

Variable usage frequencies as the number of trees $m$ is reduced
Variable Selection via BART

Using $m = 10$ trees, it remains effective with $n = 100$ and $p = 90$
These are estimates of $f_3(x_3) = \frac{1}{n} \sum_i f(x_3, x_{ic})$ where $x_c = x \setminus x_3$
Partial Dependence Plots for the Friedman Example
The Marginal Effects of $x_1 - x_{10}$
Example: HIV Data Analysis

\[ Y = \text{LDHL (log of hdl level)} \]

\[ x's = \text{CD4, Age, Sex, Race, Study, PI1, PI2, NNRTI2, NRTI1, NRTI2, ABI_349, CRC_71, CRC_72, CRC_55, CRC_73, CRC_10, ABI_383, ABI_387, ABI_391, ABI_395, ABI_400, ABI_401, CRC_66, CRC_67, CRC_68, CRC_69} \]

\[ n = 458 \text{ patients} \]

For this data

Least Squares yields \( R^2 = 26\% \)

BART yields \( R^2 = 42\% \)
BART suggests there is not a strong signal in x for this y.
Variable Selection via BART

with m = 5, 10, 20, 50, 200 trees
Partial Dependence Plots Suggest Genotype Effects

For example, the average predictive effect of ABI_383
Predictive Inference about Interaction of NNRTI2 Treatment and ABI_383 Genotype

There appears to be no interaction effect
Example: Drug Discovery

Goal: To predict the “activity” of a compound against a biological target.

Data: \( n = 2604 \) compounds of which 604 are active

\[ Y = 0 \text{ or } 1 \text{ according to whether compound is active} \]

\( x \)'s - 266 variables that characterize the compound’s molecular structure

BART is easily extended to estimate \( P[Y = 1|x] \) by Albert-Chib augmentation with a probit link

Applied BART to rank compounds by their \( P[Y = 1|x] \) estimates
BART Posterior Intervals for 20 Compounds with Highest Predicted Activity

In-sample

Out-of-Sample
Variable Importance in Drug Discovery

with $m = 5, 10, 20$ trees

All 266 $x$'s

Top 25 $x$'s
The Football Data

Each observation (n=245) corresponds to a football game.

\[ y = \text{Team A points} - \text{Team B points} \]

29 x’s. Each is the difference between the two teams on some measure. eg \( x_{10} \) is average points against defense per game for Team A for team B.
For each draw, for each variable calculate the percentage of time that variable is used in a tree. Then average over trees.

Subtle point: Can’t have too many trees. Variables come in without really doing anything.
Marginal Effects of the Variables

Just used variables 2, 7, 10, and 14.

Here are the four univariate partial-dependence plots.
A Bivariate Partial Dependence Plot
The joint effect of two of the x’s
A Sketch of the Prior

First, introduce prior independence as follows

\[ \pi((T_1, M_1), \ldots, (T_m, M_m), \sigma) = \prod \pi(T_j, M_j) \cdot \pi(\sigma) \]

\[ = \prod \pi(\mu_{ij} \mid T_j) \cdot \pi(T_j) \cdot \pi(\sigma) \]

Thus we only need to choose \( \pi(T) \), \( \pi(\mu \mid T) = \pi(\mu) \) and \( \pi(\sigma) \).
We specify a process that grows trees:

Step 1) Grow a tree structure with successive biased coin flips
Step 2) Randomly assign variables to decision nodes
Step 3) Randomly splitting rules to decision nodes

Hyperparameters chosen to put prior weight on small trees!!
\[ \pi(\mu \mid T) \]

For each bottom node \( \mu \), let \( \mu \sim N(\mu, \sigma^2) \)

which induces \( E[Y \mid x] \sim N(m \mu, m \sigma^2) \)

Since we are pretty sure that \( E[Y \mid x] \in (y_{\text{min}}, y_{\text{max}}) \), we chose \( \mu \) and \( \sigma \) such that for a prechosen \( k \) (such as 2 or 3)

\[
(m \mu - k\sqrt{m}\sigma, m \mu + k\sqrt{m}\sigma) = (y_{\text{min}}, y_{\text{max}})
\]

This conveniently done by shifting and rescaling y so that \( (y_{\text{min}}, y_{\text{max}}) = (-0.5, 0.5) \) and then setting \( \mu = 0 \) and

\[
k\sqrt{m}\sigma = .5 \Rightarrow \sigma = \frac{.5}{k\sqrt{m}}
\]

Note how the prior adapts to \( m \): \( \sigma \) gets smaller as \( m \) gets larger.

Default choice is \( k = 2 \).
Let \( \sigma^2 \sim \frac{\nu \lambda}{\chi^2_\nu} \) and consider \( \nu = 3, 5 \) or 10.

To set \( \lambda \), we use a rough overestimate of \( \sigma \) based on the data (such as \( \text{sd}(y) \) or the LS estimate for the saturated linear regression).

Determine \( \lambda \) by setting a quantile such as .75, .95 or .99 at this rough estimate.

The three priors we would consider when \( \sigma = 2 \):
BART is Robust to Prior Settings

On the Friedman (1991) example, BART’s robust RMSE performance is illustrated below where the prior hyperparameter \((\nu,q,k,m)\) choices are varied.
A Sketch of the MCMC algorithm

\[ Y = g(x; T_1, M_1) + g(x; T_2, M_2) + \ldots + g(x; T_m, M_m) + \sigma z \]

The "parameter" is: \( \theta = ((T_1, M_1), \ldots, (T_m, M_m), \sigma) \)

"Simple" Gibbs sampler:

(1) \( \sigma \mid \{T_j\}, \{M_j\}, \text{data} \)

(2) \( (T_j, M_j) \mid \{T_i\}_{i \neq j}, \{M_i\}_{i \neq j}, \sigma, \text{data} \) (Bayesian backfitting)

(1) Subtract all the g's from y to update \( \sigma \)
(2) Subtract all but the \( j^{\text{th}} \) g from y to update \( (T_j, M_j) \)
Using the decomposition

\[ p(T, M \mid \text{data}) = p(T \mid \text{data}) \cdot p(M \mid T, \text{data}) \]

and the fact that \( p(T \mid \text{data}) \) is available under our prior, we sample

\[ (T_j, M_j) \mid \{T_i\}_{i \neq j}, \{M_i\}_{i \neq j}, \sigma, \text{data} \]

by first drawing \( T \) from \( p(T \mid \text{data}) \), and then drawing \( M \) from \( p(M \mid T, \text{data}) \).

Drawing \( M \) from \( p(M \mid T, \text{data}) \) is routine

Just simulate \( \mu \)'s from the posterior under a conjugate prior
To draw $T$ from $p(T \mid \text{data})$, we use a Metropolis-Hastings algorithm.

Given the current $T$, we propose a modification and then either move to the proposal or repeat the old tree.

In particular we use proposals that change the size of the tree:

- Propose a more complex tree
- Propose a simpler tree

More complicated models will be accepted if the data's insistence overcomes the reluctance of the prior.
\[ Y = g(x; T_1, M_1) + g(x; T_2, M_2) + \ldots + g(x; T_m, M_m) + \sigma z, \quad z \sim N(0, 1) \]

Thus, at each iteration, \( T_i, M_i \) and \( \sigma \) are updated.

This is a Markov chain such that the stationary distribution is the posterior.

Each tree contributes a small part to the fit, and the fit is swapped around from tree to tree as the chain runs.

The Dynamic Random Basis in Action:

As we run the chain, we often observe that an individual tree grows quite large and then collapses back to a single node.

This illustrates how each tree is dimensionally adaptive.
Using the MCMC Output to Draw Inference

At iteration \(i\) we draw from the posterior of \(f\)

\[
\hat{f}_i(\cdot) = g(\cdot; T_{1i}, M_{1i}) + g(\cdot; T_{2i}, M_{2i}) + \cdots + g(\cdot; T_{mi}, M_{mi})
\]

To estimate \(f(x)\) we simply average the \(\hat{f}_i(\cdot)\) draws at \(x\)

Posterior uncertainty is captured by variation of the \(\hat{f}_i(x)\)

Similarly, it is straightforward to estimate functionals of \(f\) such as partial dependence functions

\[
f_k(x_k) = \frac{1}{n} \sum_i f(x_k, x_{ic}) \quad \text{where} \quad x_c = x \setminus x_k
\]
Where do we go from here?

BART (and other nonparametric methods) can give us a sense of
  • the location of $E(y \mid x)$
  • the uncertainty about $E(y\mid x)$
  • the individual effects of the $x_j$'s
  • a subset of $x_1,...,x_p$ related to $y$

This information would seem to be very valuable for model building. The next step is how?
To be continued…
One of the 42 Datasets is the well-known Boston Housing Data

Each observation corresponds to a geographic district

\[ y = \log(\text{median house value}) \]

13 x variables, stuff about the district

eg. crime rate, % poor, riverfront, size, air quality, etc.

n = 507 observations
Each boxplot depicts 20 rmse's out-of-sample for a version of a method.

eg. the method neural nets with a given number of nodes and decay value.

Smaller is better. **BART wins!**
Comparison of BART with Other Methods

50 simulations of 100 observations of Friedman example

The cross validation domain used to tune each method

<table>
<thead>
<tr>
<th>Method</th>
<th>Parameter</th>
<th>Values considered</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boosting</td>
<td># boosting iterations</td>
<td>n.trees= 1, 2, ..., 2000</td>
</tr>
<tr>
<td></td>
<td>Shrinkage (multiplier of each tree added)</td>
<td>shrinkage= 0.01, 0.05, 0.10, 0.25</td>
</tr>
<tr>
<td></td>
<td>Max depth permitted for each tree</td>
<td>interaction.depth= 1, 2, 3, 4</td>
</tr>
<tr>
<td>Neural Nets</td>
<td># hidden units</td>
<td>size= 10, 15, 20, 25, 30</td>
</tr>
<tr>
<td></td>
<td>Decay (penalty coef on sum-squared weights)</td>
<td>decay= 0.50, 1, 1.5, 2, 2.5</td>
</tr>
<tr>
<td></td>
<td>(Max # optimizer iterations, # restarts)</td>
<td>fixed at maxit= 1000 and 5</td>
</tr>
<tr>
<td>Random Forests</td>
<td># of trees</td>
<td>ntree= 200, 500, 1000</td>
</tr>
<tr>
<td></td>
<td># variables sampled to grow each node</td>
<td>mtry= 3, 5, 7, 10</td>
</tr>
<tr>
<td>MARS</td>
<td>GCV penalty coefficient</td>
<td>gcv= 1, 2, ..., 8</td>
</tr>
<tr>
<td>BART -cv</td>
<td>Sigma prior: ((\nu, q)) combinations</td>
<td>(3,0.90), (3,0.99), (10,0.75)</td>
</tr>
<tr>
<td></td>
<td>(\mu) Prior: (k) value for (\sigma_{\mu})</td>
<td>1, 1.5, 2, 2.5, 3</td>
</tr>
<tr>
<td></td>
<td>(# trees (m), iterations used, burn-in iterations)</td>
<td>fixed at (200, 1000,500)</td>
</tr>
<tr>
<td>BART -default</td>
<td>Sigma prior: ((\nu, q)) combinations</td>
<td>fixed at (3,0.90)</td>
</tr>
<tr>
<td></td>
<td>(\mu) Prior: (k) value for (\sigma_{\mu})</td>
<td>fixed at 2</td>
</tr>
<tr>
<td></td>
<td>(# trees (m), iterations used, burn-in iterations)</td>
<td>fixed at (200, 1000,500)</td>
</tr>
</tbody>
</table>

Table 1: Operational parameters for the various competing models. Names in last column indicate parameter names in R.