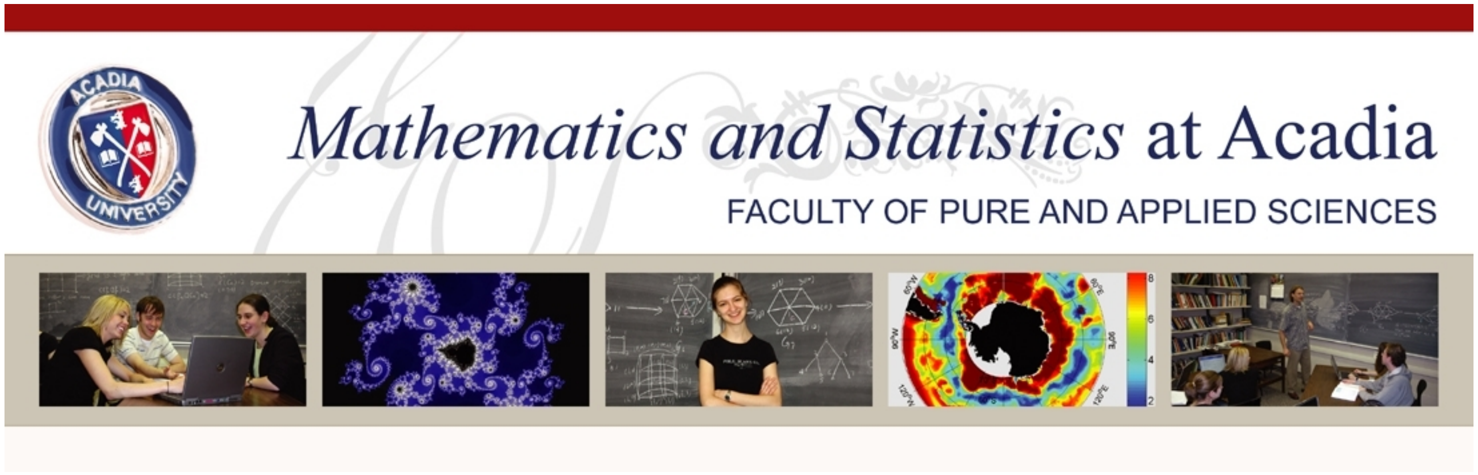
Bayesian Additive Regression Trees (BART)

Hugh Chipman, Acadia University
Edward George, University of Pennsylvania and
Robert McCulloch, University of Chicago

Software is available! Google "Hugh" and "Acadia".


Mathematics and Statistics at Acadia
FACULTY OF PURE AND APPLIED SCIENCES
ACADIA UNIVERSITY

Outline:

1. Introduction to Ensemble models
2. A train/test bake-off comparison
3. BART: A Bayesian Ensemble
4. Examples and other cool stuff (fake & real examples, active learning)

Part 1: Introduction to Ensemble models

An Introduction to Ensembles:

- Basic problem: Function estimation with data
- Model is

$$y = f(x) + \text{noise}$$

  with
  - y a one dimensional variable
  - x a p-dimensional variable
  - Observed data are N pairs $(x_1, y_1), \ldots, (x_N, y_N)$.
  - $f(x)$ estimated using the observed data.
- Ensemble models assume that $f(x)$ is actually a sum of m (often many) functions:

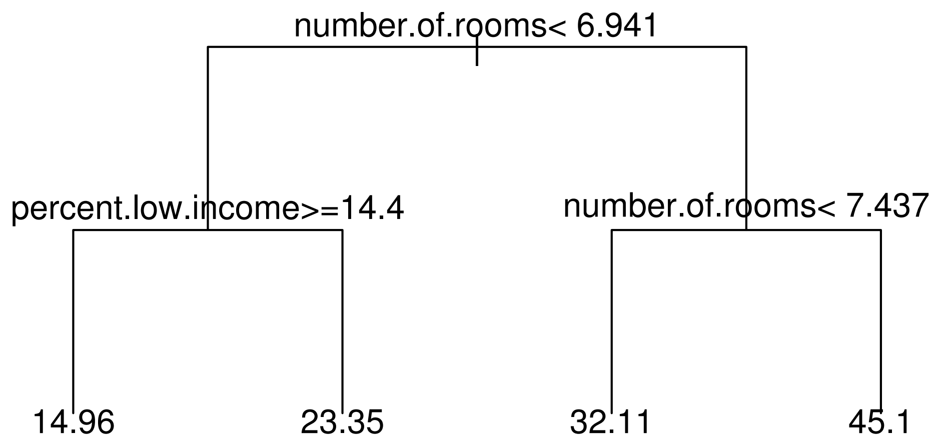
$$f(x) = g_1(x) + g_2(x) + \ldots + g_m(x)$$

  - Examples: Linear model, Generalized Additive Model, MARS, Neural net, ...
  - Original ensemble motivation: we get a better prediction by averaging a "committee" of individual models (g_i's).

An Introduction to Ensembles:

$$f(x) = g_1(x) + g_2(x) + \ldots + g_m(x)$$

- In principal, the individual g_i's could be any model.
- In practice, they're often **Trees**.



- Trees have several advantages:
  1. Selection of relevant X's.
  2. Able to represent interactions.
  3. Can handle missing values, categorical X's.

An Introduction to Ensembles:

$$f(x) = g_1(x) + g_2(x) + \ldots + g_m(x)$$

- Each tree g_i has parameters we must learn from the data:
  - Tree structure (topology and splitting rules): T_i
  - Predictions in terminal node: M_i (e.g. node constants $\mu_1, \mu_2, \ldots, \mu_b$ if there are b terminal nodes)

$$f(x) = g(x; T_1, M_1) + g(x; T_2, M_2) + \ldots + g(x; T_m, M_m)$$

- Simultaneous optimization of $T_1, ..., T_m, M_1, ..., M_m$ infeasible.

An Introduction to Ensembles:

Estimation: *"The algorithm is the model"* - Breiman, 2001

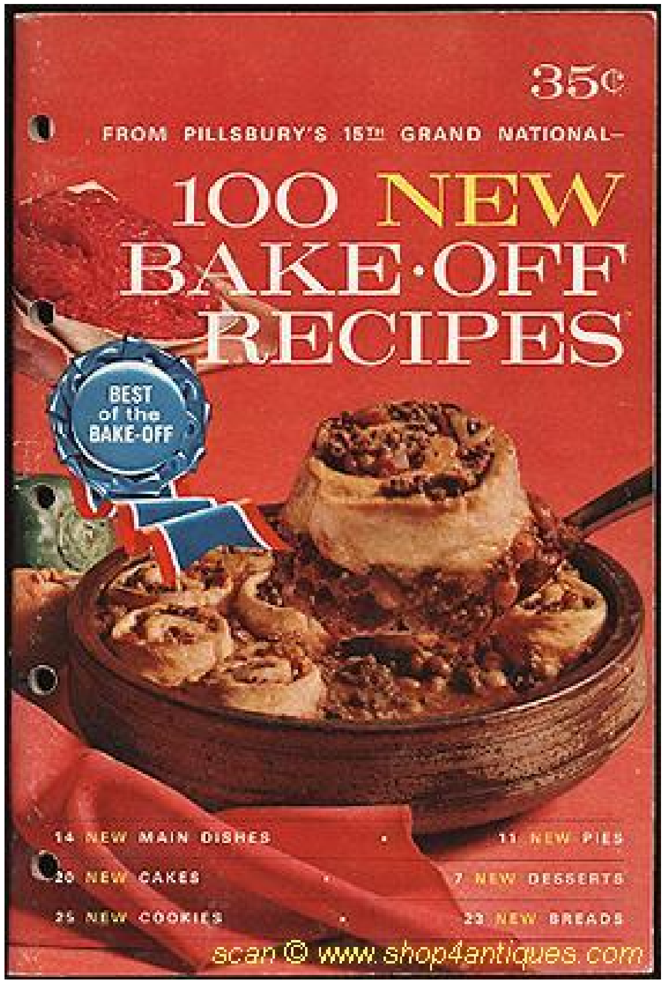
Several ways to estimate $T_1, ..., T_m, M_1, ..., M_m$:

- Bagging (Breiman 1996), Bayesian Tree models (Chipman, George, McCulloch 1998) and Random Forests (Breiman 2001):
  - Use randomized training (data resampling/stochastic search) to identify multiple trees that fit well.
  - Prediction is an average across individual tree predictions.
- Boosting (Freund and Schapire 1997, Friedman 2001)
  - has individual g_i that fit poorly (*weak learners*)
  - but they are chosen so that when combined they predict well.
- Both classes of models produce

$$f(x) = g(x; T_1, M_1) + g(x; T_2, M_2) + \ldots + g(x; T_m, M_m)$$

Part 2: Does it work?: A large empirical study

("Bake-Off", conducted for *Neural Information Processing Systems 2006)*


35¢
FROM PILLSBURY'S 15TH GRAND NATIONAL–
100 NEW BAKE·OFF RECIPES
BEST of the BAKE-OFF
14 NEW MAIN DISHES • 11 NEW PIES
20 NEW CAKES • 7 NEW DESSERTS
25 NEW COOKIES • 23 NEW BREADS
scan © www.shop4antiques.com

Experimental comparison: 6 learners $\times$ 42 datasets

- **Learners**:
  - Random Forests
  - Boosting (Friedman's gradient boosting machine)
  - BART-default - (Bayesian Additive Regression Trees)
  - BART-cv (BART, but treat prior parameters like tuning parameters via cross-validation)
  - Linear regression with lasso
  - Neural networks (single hidden layer)
- **Datasets:**
  - From Kim, Loh, Shih and Chaudhuri (2006)
  - Up to 65 predictors and 6806 observations
- **Details:**
  - Train on 5/6 of data, test on 1/6
  - Learners tuned via 5-fold CV within training set.
  - 20 Train/Test replications per dataset

Results: Root Mean Square Errors

Average test set RMSE across 42 datasets (after standardizing Y):

$$\text{RMSE} = \sqrt{\sum_{i=1}^{N} (Y_i - \hat{f}(x_i))^2 / N}$$

BART-CV:	0.5042
Boosting:	0.5089
BART:	0.5093
Random Forest:	0.5097
Neural Net:	0.5160
Lasso:	0.5896

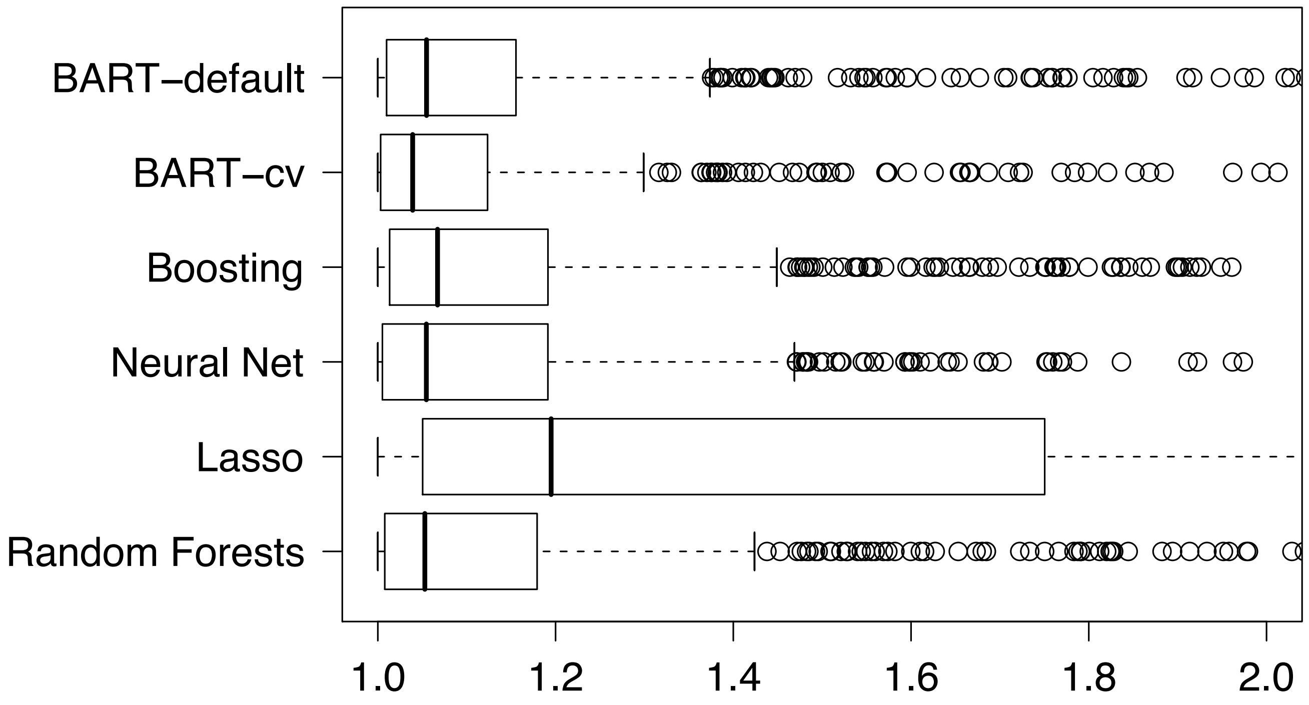
Some comments:

- BART does quite well
- Treating BART like a machine learner only gives a modest improvement.
- It's actually pretty surprising how close the different models are.

Results: Relative Root Mean Square Errors

"Relative" $\Rightarrow$ for each replicate on each data set, we identify best model, and all RMSEs are divided by the error of the best model.

$\Rightarrow$ "1.0" is best, "2.0" is a RMSE twice as large as best model.



Results: Relative Root Mean Square Errors

- The other ensembles may be doing well for different reasons:
  - Boosting forces each learner to model different structure in the data
  - Random Forests use model averaging to reduce variability
- Neither ensemble gives any prediction inference.
  - This will be our goal: combine the strengths of boosting and random forests in a model that allows inference.
- Extra bonus(es):
  - Bayesian machinery largely removes need for tuning model parameters.
  - Pointwise uncertainty in predictions.
  - Uncertainty for the magnitude of the effect of a predictor.
  - Diagnostics for model checking.

Part 3: Bayesian Additive Regression Trees (BART)

- Ensembles as a statistical model.
- Prior specification
- MCMC estimation

Ensembles as a statistical model:
Bayesian Additive Regression Trees (BART)

Our data model is

$$Y = f(x) + \epsilon, \ \epsilon \sim N(0, \sigma^2)$$

or more specifically,

$$Y = g(x, T_1, M_1) + g(x, T_2, M_2) + \ldots + g(x, T_m, M_m) + \epsilon.$$

with the errors being iid Normal$(0, \sigma^2)$

The parameters we have to estimate are:

- m Trees $T_1, T_2, \ldots, T_m$.
- m sets of terminal node parameters $M_1, \ldots M_m$ (with b_j nodes in tree j).
- A single scale parameter σ for the residual variance.

Details:

1. Prior specification
2. MCMC sampling of the posterior

One important point. The model

$$Y = g(x, T_1, M_1) + g(x, T_2, M_2) + \ldots + g(x, T_m, M_m) + \epsilon.$$

is different from Bayesian model averaging of a single tree model.

We are obtaining a posterior for a "sum of m trees" model, with a joint posterior on m trees and m terminal node parameter vectors.

Prior specification:

Need to specify a prior on T's, M's, and σ.

- Assume prior structure:

$$p((T_1, M_1), (T_2, M_2), \ldots, (T_m, M_m), \sigma) \\ = p(T_1, T_2, \ldots, T_m)\, p(M_1, M_2, \ldots, M_m \,|\, T_1, T_2, \ldots, T_m)\, p(\sigma).$$

- Since the dimension of the M depends on the T, this conditional structure is essential.
- We simplify even further by imposing independence whenever possible.

$$p(T_1, T_2, \ldots, T_m) = \prod p(T_j)$$

$$p(M_1, M_2, \ldots, M_m \,|\, T_1, T_2, \ldots, T_m) = \prod p(M_j \,|\, T_j)$$

$$p(M_j \,|\, T_j) = \prod p(\mu_{i,j} \,|\, T_j)$$

Prior Specification:

Semi-automatic choices motivated by Empirical Bayes methods.

Basic ideas:

- T's: how big a tree is probable?
- σ: how much noise in the response?
- $M's$: How much can each individual tree contribute?
  Clever trick: Make this depend on the number of trees.
- Number of trees could be a parameter, but we fix it instead.

Prior Specification:

1. Residual variance σ^2:

   Prior for σ is simplest and most important.

   We use the standard conjugate prior:

$$\sigma^2 \sim \frac{\nu\,\lambda}{\chi^2_\nu}.$$

   - ν determines spread of the prior
   - λ determines location of the prior

   Instead of eliciting ν, λ directly we:

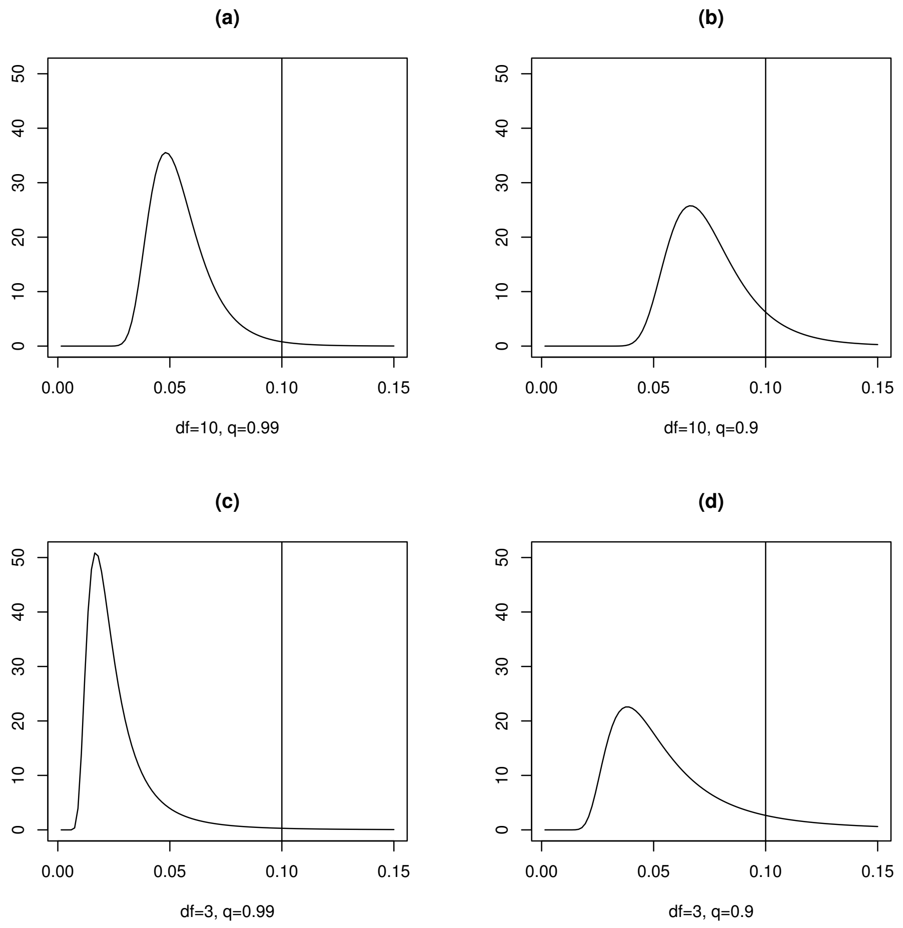
   (a) Guess at an upper quantile of σ, say 90% or 99%. Set this equal to least squares linear regression estimate of σ

   (b) Choose ν to give good spread of σ prior.

Prior Specification:

Vertical line indicates $\hat{\sigma}$, the rough estimate of σ from linear least squares model.

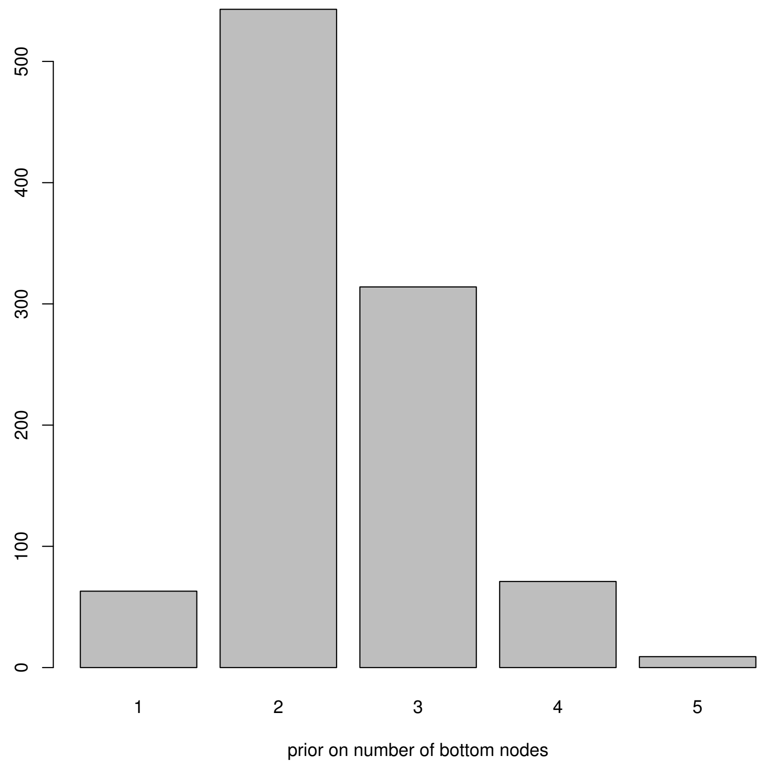
- Top: $\nu = 10$
- Bottom:$\nu = 3$
- Left: $\hat{\sigma}$ at 99th quantile.
- Right: $\hat{\sigma}$ at 90th quantile.



Prior Specification:

2. Prior on tree structure T:

- Basically a prior on tree size*.
- Actual prior used in examples later gives tree size prior in plot.
- NOTE that unlike Boosting, we don't fix tree depth. We put a prior on it and let the data determine tree depth.



- Tree size determines how many variables are used in each "weak learner" $g(x; T, M)$.

* Actually a distribution on whether you split, which variable you split on, and the splitting rule, for each node.

Prior Specification:

3. Terminal node parameters μ_i

- Suppose we have $m = 200$ trees.
- For any x, the prediction $f(x)$ will be a sum of 200 μ's, one from each tree.

$$\theta = E(Y|x) = f(x) = \sum_{i=1}^{200} \mu_i$$

$$\text{Var}(\theta) = \sum_{i=1}^{200} \text{Var}(\mu_i) = 200\text{Var}(\mu_i)$$

(assuming μ_i's independent given the trees)

- So, we can specify how much we expect the mean of y given x (i.e., θ) to vary, and take

$$\text{Var}(\mu_i) = \text{Var}(\theta)/200$$

$$\text{sd}(\mu_i) = \text{sd}(\theta)/\sqrt{200}$$

Prior Specification:

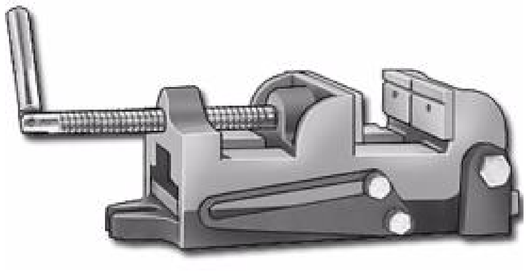
3. Terminal node parameters μ_i

- Assume μ_i is normally distributed, with mean 0, and standard deviation

$$\text{sd}(\mu_i) = \frac{\text{range(Y)}}{4\sqrt{200}}, \qquad \text{for 200 trees in sum}$$

  NOTE: The amount of shrinkage of the μ's *depends on the number of trees* (here $m = 200$).

- Each term g(x;T,M) will be "regularized" so it contributes only a small part of the overall fit.



So the model is adaptively regularized in several ways: Tree prior and terminal node prior.

MCMC Estimation:

Instead of explicitly maximizing the posterior, we simulate from it, via Markov chain Monte Carlo (MCMC).

In a nutshell:

Let $T_{(-j)}$ be all trees *except* T_j, define $M_{(-j)}$ similarly.
Repeat $k = 1, ..., 1000$ (say)

- Repeat $j = 1, ..., m$ times
  - Metropolis-Hastings step: Draw T_j conditional on $Y, T_{(-j)}, \sigma$
  - Draw M_j given $Y, T_1, \dots T_m, M_{(-j)}, \sigma$
- Draw σ given Y and all other parameters.

Note that the sample of T_j at step k is actually a modification of the T_j sample at step $k-1$.

MCMC Estimation:

Final prediction:

- Each sweep of algorithm yeilds a draw from the posterior of

$$f(x) = g(x, T_1, M_1) + g(x, T_2, M_2) + \ldots + g(x, T_m, M_m)$$

- Average the draws - gives the posterior average of $f(x)$.
- Uncertainty in $f(x)$ is also available, from the posterior distribution on $f(x)$.

Connections to other learning algorithms:

1. Bayesian Backfitting (Hastie and Tibshirani) is a similar MCMC approach.
2. Like Boosting, each of our "weak learners" $g(x; T_j, M_j)$ learns structure that the other weak learners do not capture.
3. Like Random Forests and Bagging, we model average over multiple draws of the sum of trees model.

Part 4: Examples with Data (Simulated, Boston, Active Learning)

Simulated example: Friedman (1991)

$$y = f(x) + \epsilon, \qquad \epsilon \sim N(0,1)$$

where

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

(10 x's but only the first 5 matter)

$N = 100$ observations

BART settings:

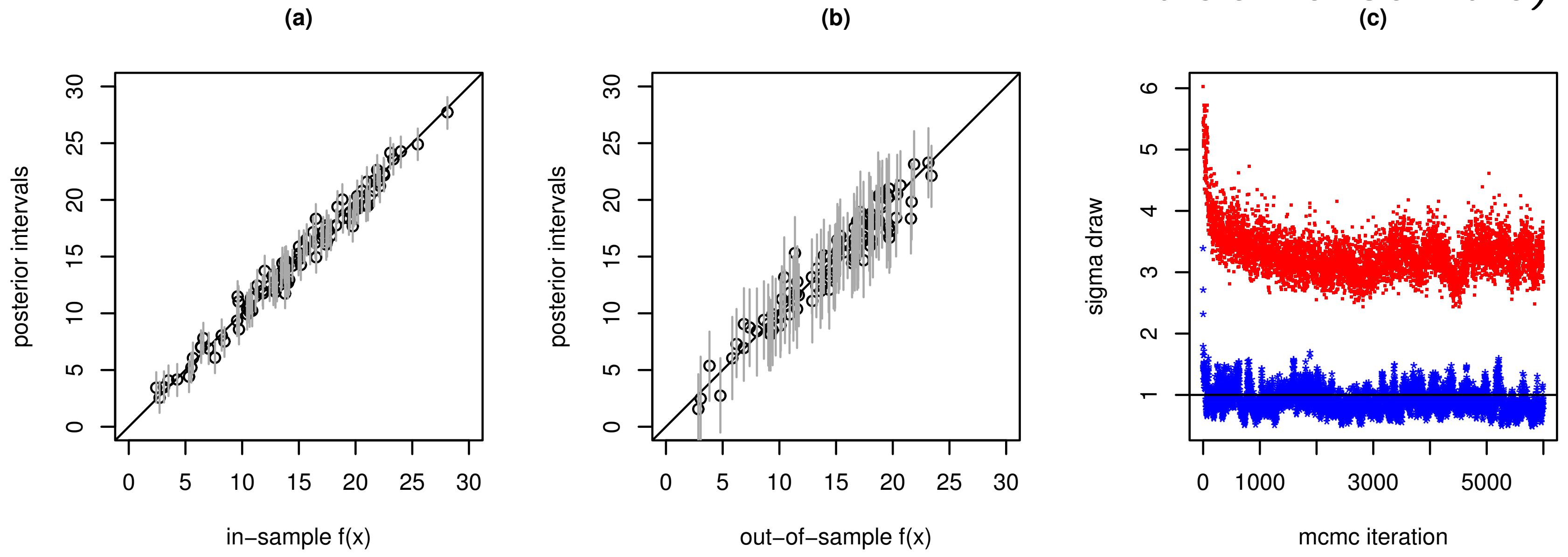
- $m = 100$ trees
- σ prior uses $\hat{\sigma}$ from linear least squares regression as 90th quantile, $\nu = 3$.
- Tree prior puts most probability on 2, 3 terminal nodes.
- Automatic choice of $M = \{\mu_j\}$ prior just discussed.

Simulated example:

Training sample predictions

Test sample predictions

posterior for σ (red= 1 tree, blue=ensemble)



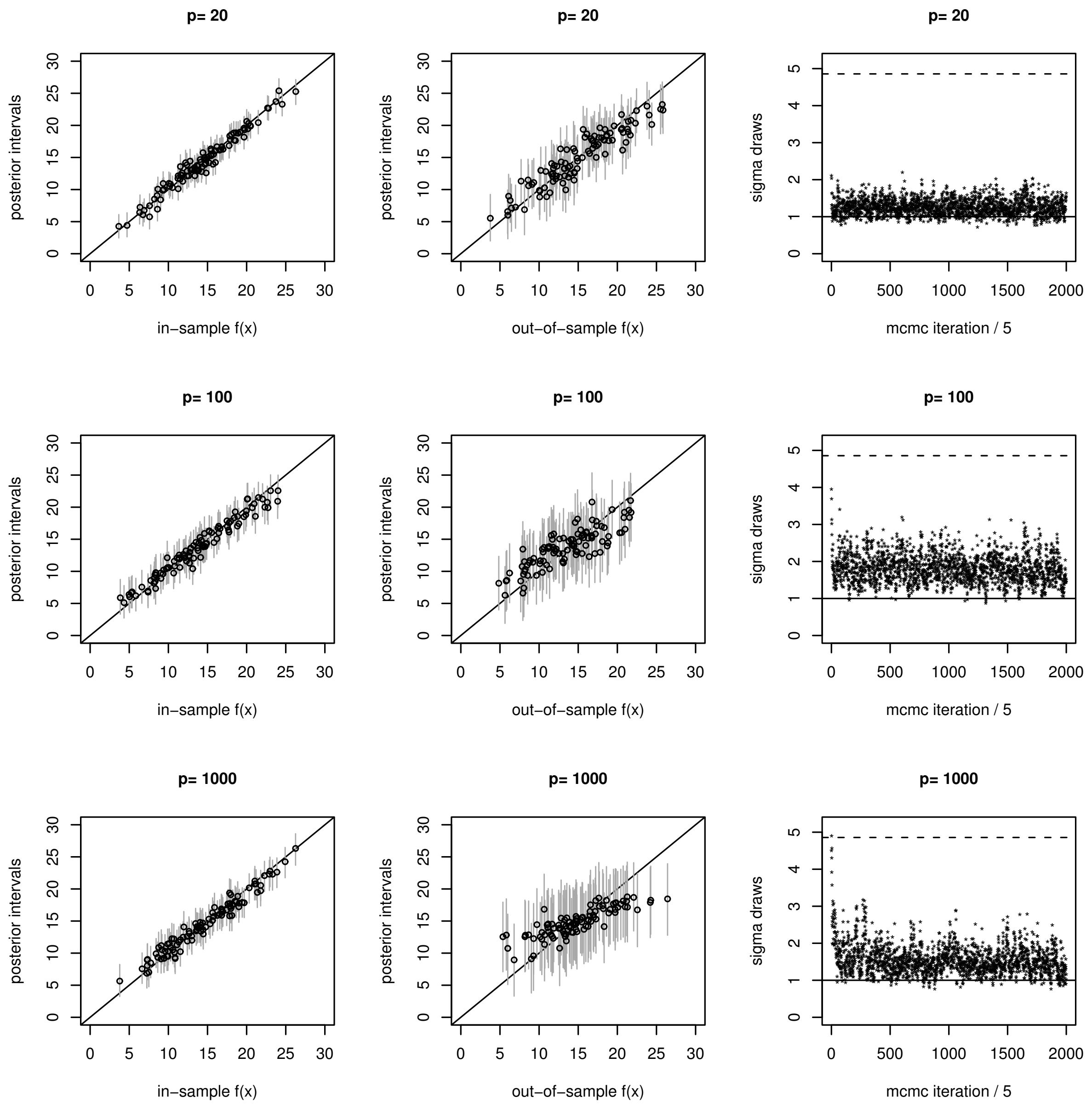
- Chain converges quickly and mixes well.
- Note that the model is not identifiable, but we are really only interested in identifiability of predictions.

Simulated example:

$p = 20$ dimensions

$p = 100$ dimensions

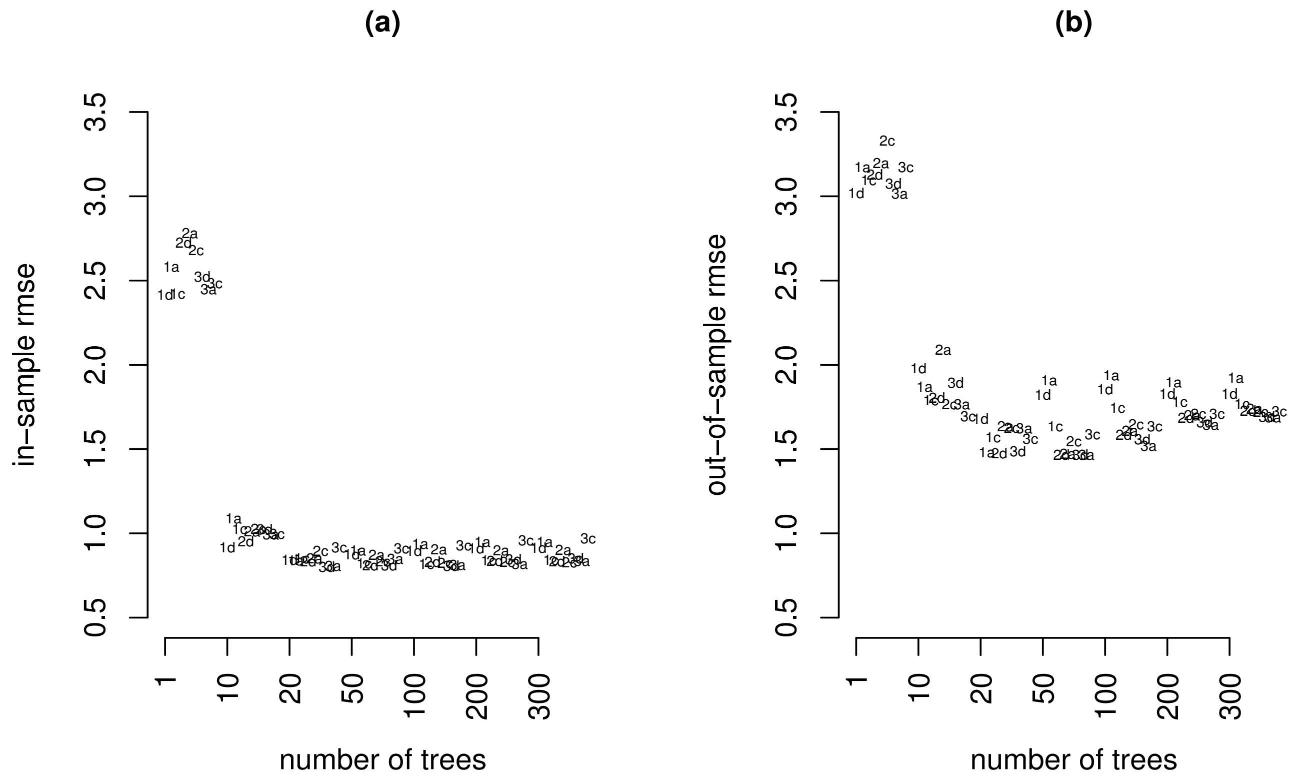
$p = 1000$ dimensions



Simulated example:

Previous page: BART capable of extracting low-dimensional signal with many x's. (Even $n \ll p$, i.e. $n = 100$ observations in $p = 1000$ dimensions!)

Also reasonable robustness to prior settings:



Training Results   Test Results

Additional Goodies: The Boston Housing Example

- Goal: Predict neighbourhood house price using demographic variables.
- Data:
  - y=log median house price in the region (the response)
  - X is 13 predictors, measuring pollution, crime, house sizes, commute distance, racial diversity, tax rates, etc.
- Common "benchmark" problem.

Additional Goodies: The Boston Housing Example

Posterior distribution on the number of terminal nodes of the 200 trees (actually a draw from the posterior).

Number nodes	1	2	3	4	5	6
Relative freq (%)	3.5	54.5	32.5	8.5	0.5	0.5

This can be interesting because

- a 1-node tree doesn't contribute to the model
- a 2 node tree is a main effect for one variable
- a 3 node tree is a two-way interaction
- ... etc.

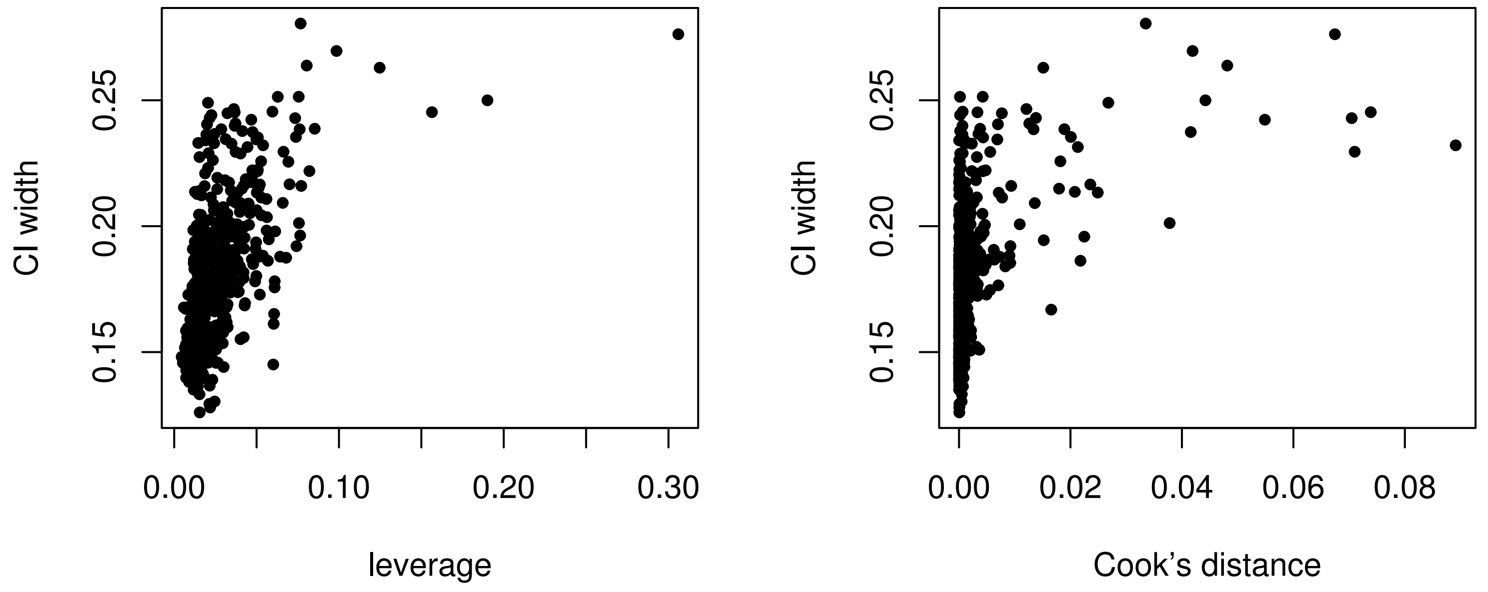
In this case, there seems to be mostly main effects and some two-way interactions.

(still a somewhat dodgy way to measure interaction order)

Additional Goodies: The Boston Housing Example

Relation to model diagnostics

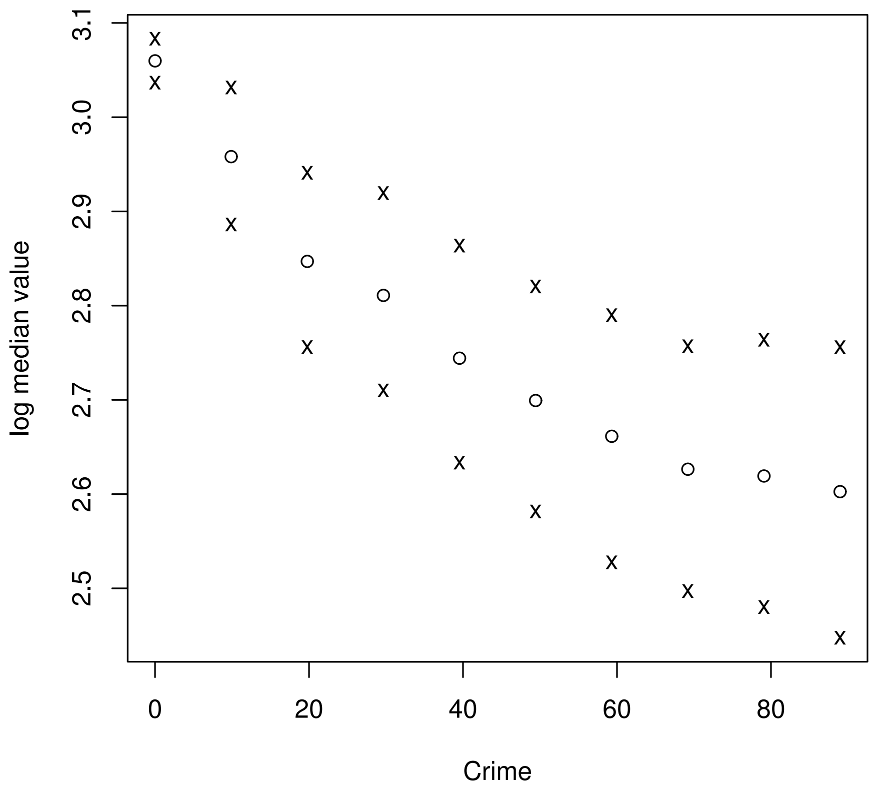
- Consider predicting y. For each point, plot the posterior interval width against traditional regression diagnostics (left: leverage, right: Cook's distance).
- Influential points tend to have larger posterior intervals.
- Posterior gives information about influential points.



Additional Goodies: The Boston Housing Example

Partial Dependence Plots

- Want to measure the effect of one or two x's on $f(x)$.
- Basically we margin over the other variables (Friedman 2001).
- Full posterior inference for such a plot is straightforward.
- Example: `crime rate`



- Almost all crime rates are in the 0-5 range.
- Bounds widen as we have less data (high crime rate).

And Now for Something Completely Different....

Active Learning

Active Learning

The game we play:

- Same "regression" scenario as before: predict Y using X.
- The difference is that **we can sequentially choose the x's at which we measure Y**.
- That is, we assume that all potential x's are known, and we need to choose which ones we measure Y at.
- By "actively learning" (ie sequentially gathering data) we hope to build a better model with less data.
- This is essentially experimental design.
- Much of the theory for design applies to linear models, here we show how adaptive models can be used for sequential design.

Active Learning

Sketch of the active learning algorithm:

1. Select an initial design (i.e., initial set of observations) X_0 with n_0 points via some criterion.
2. Obtain response values Y_0 for data.
3. Build a model using data $D_0 = (X_0, Y_0)$.
4. Repeat $j = 1, ..., n$:
   (a) For each potential design point $x_i \in$ candidate set C, calculate the design criterion
   (b) Select point x_{i^*} with best design criterion yielding design $X_j = (X_{j-1}, x_{i^*})^T$.
   (c) Measure response at X_j, giving $Y_j = (Y_{j-1}, y_{i^*})^T$
   (d) Build model M_j using $D_j = (X_j, Y_j)$.

Note:

- At the end of this algorithm, we will have n_0+n observations.
- Details on calculation in 4(a) on subsequent pages.

Active Learning

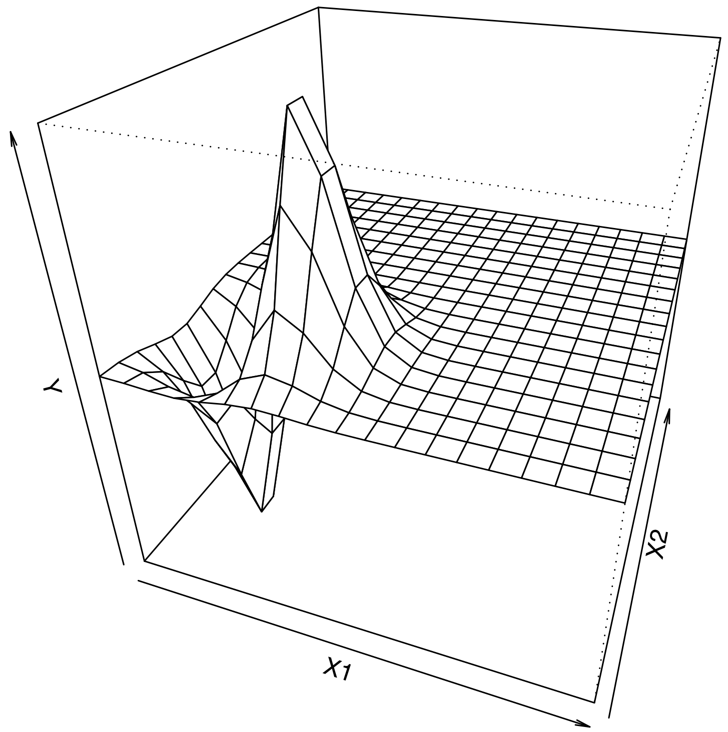
Two possible design criteria:

1. Maximize variance of predicted response ("ALM" - MacKay (1992))
   (*where do I know the least about Y?*).
2. Maximize expected reduction in variance of predicted response, averaged over a candidate set C ("ALC" - Cohn (1996))
   (*what data point will improve my model's predictions most?*)

We'll use # 1 here

2-D example (Gramacy and Lee 2006)

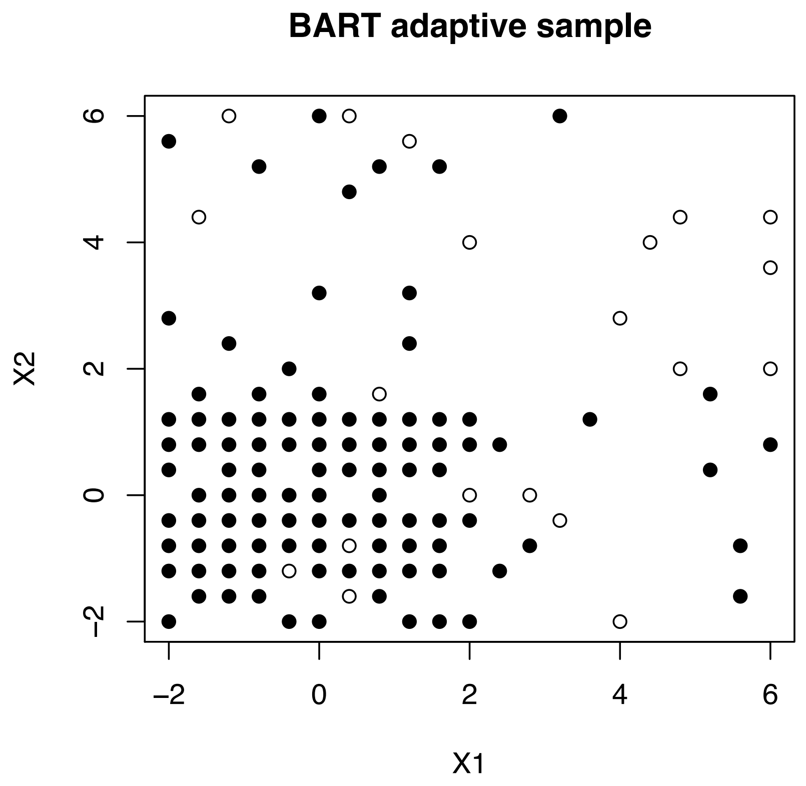
- Two predictors
- Function (right) nearly constant in 75% of input space
- Initial SRS of 20 observations, followed by adaptive sampling of 100 observations.
- All observations on a 21×21 grid.

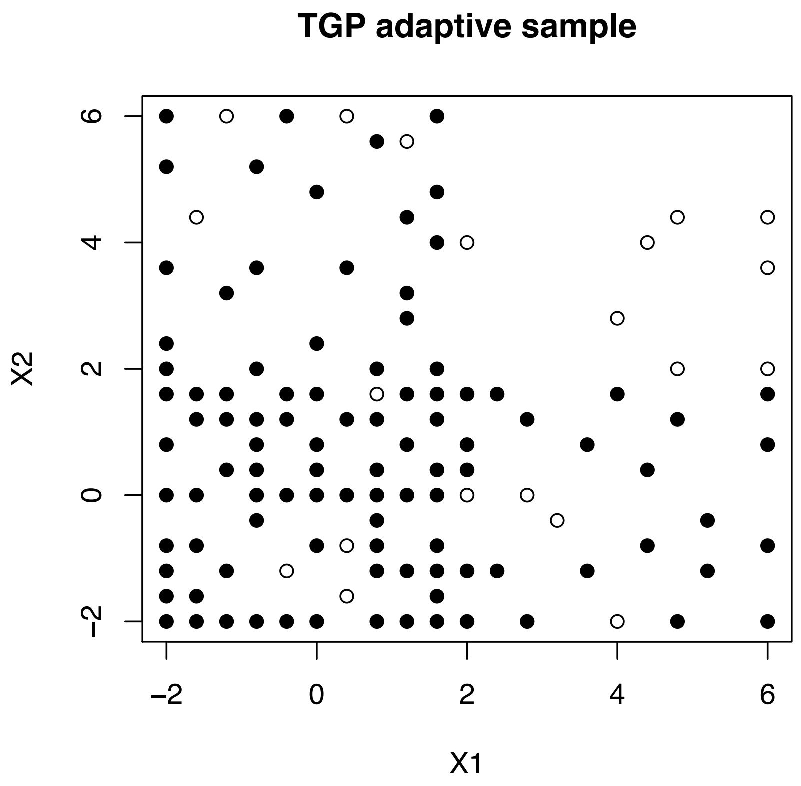


We'll make comparisons with Gramacy and Lee's "Treed Gaussian Processes" (TGP)

2-D example (Gramacy and Lee 2006)

Points sampled by BART and TGP:





Test-set MSE's (right) indicate

- Both select good samples
- TGP fits better (smooth)

MSE	Model	
	TGP	BART
SRS sample	3.66	15.96
BART sample	0.35	2.84
TGP sample	0.40	2.93

Active Learning

- This may look like a dead heat, but ...
  - BART scales well to large n and large p.
  - Ability of BART to discard irrelevant variables may be handy.
  - BART can handle categorical X's

Summary and future work:

1. It is possible to have a flexible predictive model, but still use it to make statistical inferences.
   - There is some computational cost.
   - Some derivations of models necessary.
   - But it's worth it: Cross-validation not necessary.
2. Extension to classification: 2-class problem is immediate: view binary outcome as corresponding to a latent continuous variable.
3. We plan to do extension to exponential family (similarities with Hastie and Tibshirani's Bayesian Backfitting).
4. Because we have a probability model, we can build in many interesting features. (e.g., different response data types, hi-erarchical models, outliers, modelling of σ as well as μ,...)