Bayesian Ensemble Learning and Parallel Computation

Rob McCulloch

Intro

Trees and Ensemble Methods

BART

PBART: Parallel Bayesian Additive Trees

Other Parallel Approaches

End

Outline:

(i)

Trees and ensemble methods.

(ii)

BART: a Bayesian ensemble method.

(iii)

BART and “big data”:

- Parallel versions of BART.

The Hockey Data

Glen Healey, commenting on an NHL broadcast:

> *Referees are predictable. The flames have had three penalties, I guarantee you the oilers will have three.*

Well, *guarantee* seems a bit strong,
but there is something to it.

> *How predictable are referees?*

Got data on every penalty in every
(regular season) game for 7 seasons around the time they switched from one referee to two.

For each penalty (after the first one in a game) let

`revcall` =

1 if current penalty and previous penalty are on different teams,

0 otherwise.

You know a penalty has just been called,
which team is it on?
is it a reverse call on the other team???

Mean of `revcall` is .6 !

Intro
Trees and Ensemble Methods
BART
PBART: Parallel Bayesian Additive Trees
Other Parallel Approaches
End

For every penalty (after the first one in a game) we have:

Table : Variable Descriptions

Variable	Description	Mean	Min	Max
Dependent variable				
`revcall`	1 if current penalty and last penalty are on different teams	0.589	0	1
Indicator-Variable Covariates				
`ppgoal`	1 if last penalty resulted in a power-play goal	0.157	0	1
`home`	1 if last penalty was called on the home team	0.483	0	1
`inrow2`	1 if last two penalties called on the same team	0.354	0	1
`inrow3`	1 if last three penalties called on the same team	0.107	0	1
`inrow4`	1 if last four penalties called on the same team	0.027	0	1
`tworef`	1 if game is officiated by two referees	0.414	0	1
Categorical-variable covariate				
`season`	Season that game is played		1	7

Intro
Trees and Ensemble Methods
BART
PBART: Parallel Bayesian Additive Trees
Other Parallel Approaches
End

Table : Variable Descriptions

Variable	Description	Mean	Min	Max
Other covariates				
`timeingame`	Time in the game (in minutes)	31.44	0.43	59.98
`dayofseason`	Number of days since season began	95.95	1	201
`numpen`	Number of penalties called so far (in the game)	5.76	2	21
`timebetpens`	Time (in minutes) since the last penalty call	5.96	0.02	55.13
`goaldiff`	Goals for last penalized team minus goals for opponent	-0.02	-10	10
`gf1`	Goals/game scored by the last team penalized	2.78	1.84	4.40
`ga1`	Goals/game allowed by the last team penalized	2.75	1.98	4.44
`pf1`	Penalties/game committed by the last team penalized	6.01	4.11	8.37
`pa1`	Penalties/game by opponents of the last team penalized	5.97	4.33	8.25
`gf2`	Goals/game scored by other team (not just penalized)	2.78	1.84	4.40
`ga2`	Goals/game allowed by other team	2.78	1.98	4.44
`pf2`	Penalties/game committed by other team	5.96	4.11	8.37
`pa2`	Penalties/game by opponents of other team	5.98	4.33	8.25

$n = 57,883.$

How is `revcall` related to the variables?

	inrow2=0	inrow2=1
revcall=0	0.44	0.36
revcall=1	0.56	0.64

`inrow2=1`:
If the last two calls were on the same team
then 64% of the time, the next call will *reverse* and be on the other team.

`inrow2=0`:
If the last two calls were on different teams, then the frequency of reversal is only 56%.

Of course,

we want to relate revcall to all the other variables jointly!

Well, we could just run a logit,

> *but with all the info, can we,*
> *fairly automatically,*
> *get a better fit than a logit gives?*

What could we try?

Intro
Trees and Ensemble Methods
BART
PBART: Parallel Bayesian Additive Trees
Other Parallel Approaches
End

Data Mining Certificates Online
Stanford Center for Professional Development

```
Data Mining and Analysis
STATS202
Description

In the Information Age, there is an unprecedented amount of
data being collected and stored  by banks, supermarkets,
internet retailers, security services, etc.
So, now that we have all this data, what do we with it?

The discipline of data mining and analysis provides crunchers
with the tools and framework to discover meaningful patterns
in data sets of any size and scale.  It allows us to turn all of
this data into valuable, actionable information.
In this course, learn how to explore, analyze, and leverage data.

Topics Include

    Decision trees
    Neural networks
    Association rules
    Clustering
    Case-based methods
    Data visualization
```


Intro
Trees and Ensemble Methods
BART
PBART: Parallel Bayesian Additive Trees
Other Parallel Approaches
End

Modern Applied Statistics: Data Mining
STATS315B

Online

Description

Examine new techniques for predictive and descriptive learning using concepts that bridge gaps among statistics, computer science, and artificial intelligence.
This second sequence course emphasizes the statistical application of these areas and integration with standard statistical methodology.
The differentiation of predictive and descriptive learning will be examined from varying statistical perspectives.

Topics Include

- Classification & regression trees
- Multivariate adaptive regression splines
- Prototype & near-neighbor methods
- Neural networks

Instructors
Jerome Friedman, Professor Emeritus, Statistics

http://www.sas.com/events/aconf/2010/bdmci61.html

```
Advanced Analytics for Customer Intelligence Using SAS

Predictive Modeling for Customer Intelligence: The KDD Process Model
A Refresher on Data Preprocessing and Data Mining
Advanced Sampling Schemes

    cross-validation (stratified, leave-one-out)
    bootstrapping

Neural networks

    multilayer perceptrons (MLPs)
    MLP types (RBF, recurrent, etc.)
    weight learning (backpropagation, conjugate gradient, etc.)
    overfitting, early stopping, and weight regularization
    architecture selection (grid search, SNC, etc.)
    input selection (Hinton graphs, likelihood statistics, brute force, etc.)
    self organizing maps (SOMs) for unsupervised learning
    case study: SOMs for country corruption analysis

Support Vector Machines (SVMs)

    linear programming
    the kernel trick and Mercer theorem
    SVMs for classification and regression
    multiclass SVMs (one versus one, one versus all coding)
    hyperparameter tuning using cross-validation methods
    case study: benchmarking SVM classifiers
```

Opening up the Neural Network and SVM Black Box

- rule extraction methods (pedagogical versus decompositional approaches such as neurorule, neurolinear, trepan, etc.
- two-stage models

A Recap of Decision Trees (C4.5, CART, CHAID)

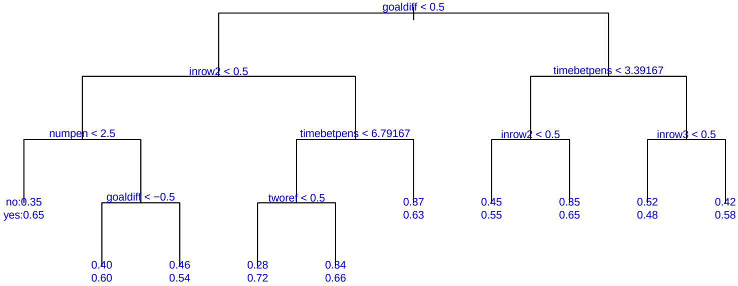
Regression Trees

- splitting/stopping/assignment criteria

Ensemble Methods

- bagging
- boosting
- stacking
- random forests

A Tree



- Last penalized was not ahead
- Last two penalties on same team
- Not long since last call
- one ref

$\Rightarrow$

72% revcall.

- Last penalized was ahead
- it has been a while since last penalty
- last three calls not on same team

$\Rightarrow$

48% revcall.

Ensemble Methods:

A single tree can be interpretable, but it does not give great in-sample fit or out-of-sample predictive performance.

Ensemble methods combine fit from many trees to give an overall fit.

They can work great!

Intro
Trees and Ensemble Methods
BART
PBART: Parallel Bayesian Additive Trees
Other Parallel Approaches
End

Let's try Random Forests (Leo Brieman).

- Randomly resample data with replacement (like bootstrapping)
- For each sample, build a big tree
- In choosing each decision rule, randomly sample a subset of variables to try.
- To predict, average (or vote) the result of the trees.

For example, *build a thousand trees !!!*

Have to choose the number of trees in the forest and the randomization scheme.

Wow! (crazy or **brilliant**?)

My impression is that Random Forests is the most popular method.

Let's try Random Forests and trees of various sizes.

For Random Forests we have to choose the number of trees to use and the number of variables to sample.

I'll use the default for number of variables and try 200,500, and 1500 trees in the forest.

We have **57,883** observations and a small number of variables so let's do a simple train-test split.

train:

use **47,883** observations to fit the models.

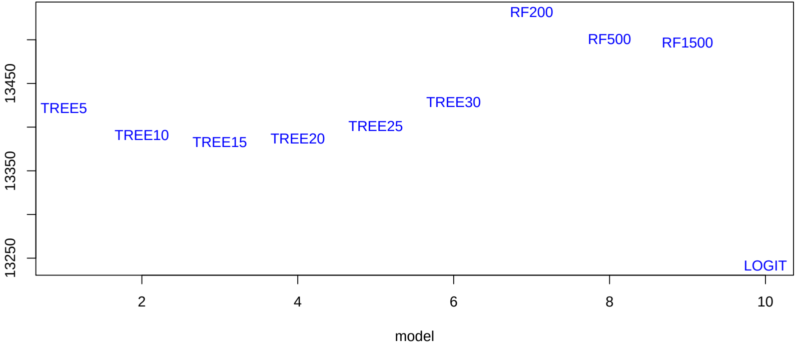
test:

use **10,000** out-of-sample observations to see how well we predict.

Loss for trees of different sizes, and 3 forests of different sizes (each forest has many big trees!).

Smaller loss is better.

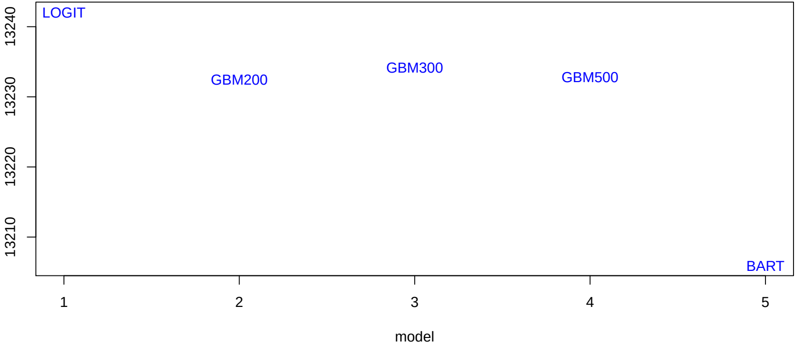
Loss is measured by the deviance
(-2*log-likelihood (out-of-sample)).



Logit does best!

Let's try **boosting** and **BART**.

Intro
Trees and Ensemble Methods
BART
PBART: Parallel Bayesian Additive Trees
Other Parallel Approaches
End



GBM is Jerry Friedman's boosting.
The number is the number of trees used.
Other parameters left at default.

BART is Bayesian Additive Regression Trees.
We used the default prior and 200 trees.

Intro
Trees and Ensemble Methods
BART
PBART: Parallel Bayesian Additive Trees
Other Parallel Approaches
End

BART

We want to "fit" the fundamental model:

$$Y_i = f(X_i) + \epsilon_i$$

BART is a Markov Monte Carlo Method that draws from

$$f \mid (x, y)$$

We can then use the draws as our inference for f.

To get the draws, we will have to:

- Put a prior on f.
- Specify a Markov chain whose stationary distribution is the posterior of f.

Intro
Trees and Ensemble Methods
BART
PBART: Parallel Bayesian Additive Trees
Other Parallel Approaches
End

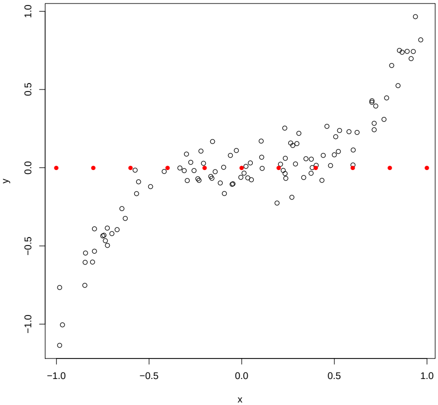
Simulate data from the model:

$$Y_i = x_i^3 + \epsilon_i \quad \epsilon_i \sim N(0, \sigma^2) \;\; iid$$

```
--------------------------------------------------------
n = 100
sigma = .1
f = function(x) {x^3}
set.seed(14)
x = sort(2*runif(n)-1)
y = f(x) + sigma*rnorm(n)
xtest = seq(-1,1,by=.2)
--------------------------------------------------------
```

Here, *xtest* will be the *out of sample* x values at which we wish to infer f or make predictions.

```
plot(x,y)
points(xtest,rep(0,length(xtest)),col='red',pch=16)
```



Red is xtest.

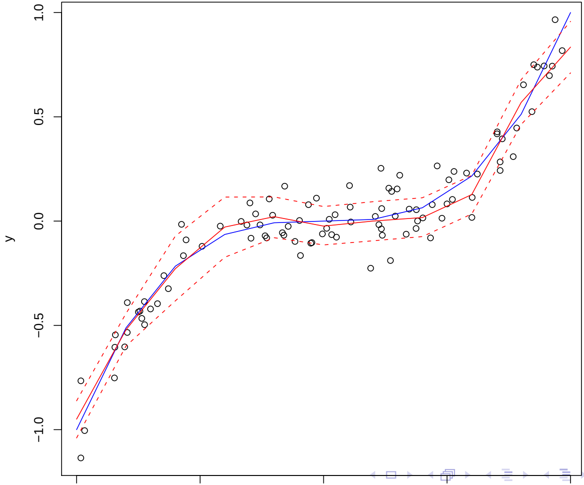
```
--------------------------------------------------
library(BayesTree)
rb = bart(x,y,xtest)
length(xtest)
[1] 11
dim(rb$yhat.test)
[1] 1000   11
--------------------------------------------------
```

The (i, j) element of yhat.test is

the i^{th} draw of f evaluated at the j^{th} value of xtest.

1,000 draws of f, each of which is evaluated at 11 xtest values.

```
--------------------------------------------------
plot(x,y)
lines(xtest,xtest^3,col='blue')
lines(xtest,apply(rb$yhat.test,2,mean),col='red')
qm = apply(rb$yhat.test,2,quantile,probs=c(.05,.95))
lines(xtest,qm[1,],col='red',lty=2)
lines(xtest,qm[2,],col='red',lty=2)
--------------------------------------------------
```



Intro

Trees and Ensemble Methods

BART

PBART: Parallel Bayesian Additive Trees

Other Parallel Approaches

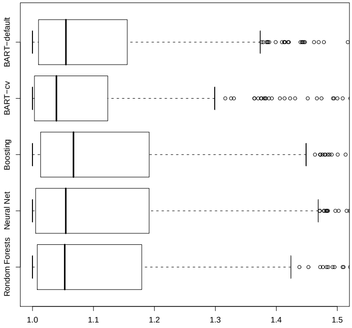
End

Example: Out of Sample Prediction

Did out of sample predictive comparisons on 42 data sets.
(*thanks to Wei-Yin Loh!!*)

- p=3 − 65, $n = 100 - 7,000$.
- for each data set 20 random splits into 5/6 train and 1/6 test
- use 5-fold cross-validation on train to pick hyperparameters (except BART-default!)
- gives 20*42 = 840 **out-of-sample predictions**, for each prediction, divide rmse of different methods by the smallest

+ each boxplots represents 840 predictions for a method
+ 1.2 means you are 20% worse than the best
+ BART-cv best
+ BART-default (use default prior) does amazingly well!!



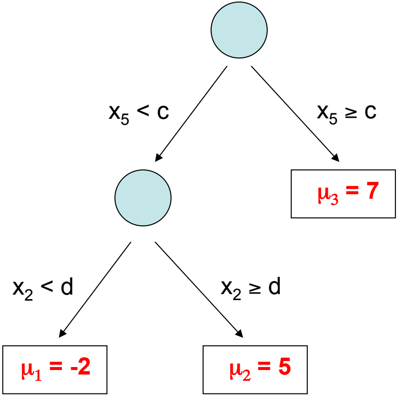
Intro
Trees and Ensemble Methods
BART
PBART: Parallel Bayesian Additive Trees
Other Parallel Approaches
End

A Regression Tree Model

Let T denote the tree structure including the decision rules.

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

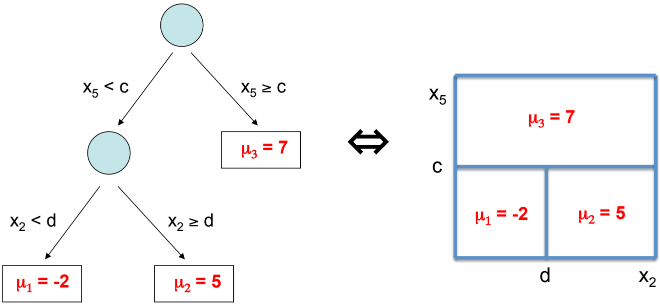
Let $g(x; T, M)$, be a regression tree function that assigns a μ value to x.



A single tree model:

$$y = g(x; T, M) + \epsilon.$$

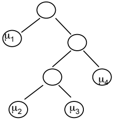
A coordinate view of $g(x; T, M)$



Easy to see that $g(x; T, M)$ is just a step function.

The BART Model

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



$m = 200, 1000, \ldots, \text{big}, \ldots.$

$f(x \mid \cdot)$ is the sum of all the corresponding μ's at each bottom node.

Such a model combines additive and interaction effects.

Complete the Model with a Regularization Prior

$$\pi(\theta) = \pi((T_1, M_1), (T_2, M_2), \ldots, (T_m, M_m), \sigma).$$

π wants:

- Each T small.
- Each μ small.
- "nice" σ (smaller than least squares estimate).

We refer to π as a regularization prior because it keeps the overall fit from getting "too good".

In addition, it keeps the contribution of each $g(x; T_i, M_i)$ model component small, each component is a "weak learner".

BART MCMC

$$Y = g(x;T_1,M_1) + \ldots + g(x;T_m,M_m) + \sigma z$$
plus
$$\pi((T_1,M_1),\ldots,(T_m,M_m),\sigma)$$

First, it is a "simple" Gibbs sampler:

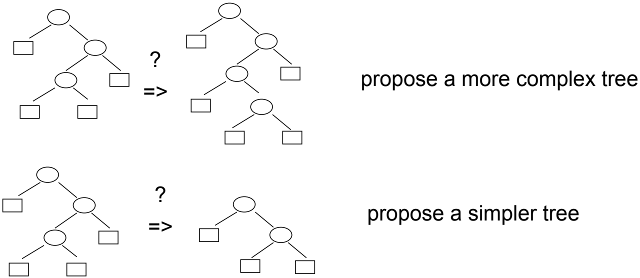
$$\begin{aligned} (T_i, M_i) \quad &| \quad (T_1, M_1, \ldots, T_{i-1}, M_{i-1}, T_{i+1}, M_{i+1}, \ldots, T_m, M_m, \sigma) \\ \sigma \quad &| \quad (T_1, M_1, \ldots, \ldots, T_m, M_m) \end{aligned}$$

To draw $(T_i, M_i) \mid \cdot$ we subtract the contributions of the other trees from both sides to get a simple one-tree model.

We integrate out M to draw T and then draw $M \mid T$.

Intro
Trees and Ensemble Methods
BART
PBART: Parallel Bayesian Additive Trees
Other Parallel Approaches
End

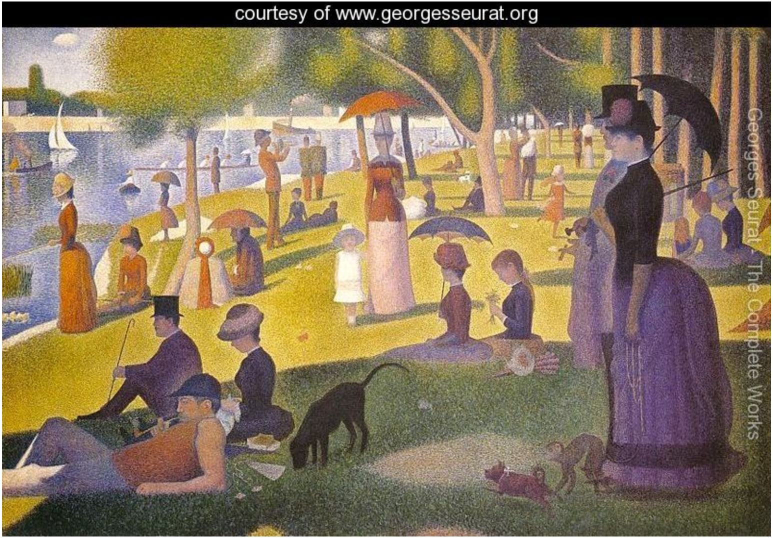
To draw T we use a Metropolis-Hastings within Gibbs step. We use various moves, but the key is a "birth-death" step.



... as the MCMC runs, each tree in the sum will grow and shrink, swapping fit amongst them ....

Intro
Trees and Ensemble Methods
BART
PBART: Parallel Bayesian Additive Trees
Other Parallel Approaches
End

Build up the fit, by adding up tiny bits of fit ..



Nice things about BART:

- don't have to think about x's
  (compare: add x_j^2 and use lasso).
- don't have to prespecify level of interaction
  (compare: boosting in R)
- competitive out-of-sample.
- stable MCMC.
- stochastic search.
- simple prior.
- uncertainty.
- small p and big n.

Intro
Trees and Ensemble Methods
BART
PBART: Parallel Bayesian Additive Trees
Other Parallel Approaches
End

PBART: Parallel Bayesian Additive Trees

Dave Higdon said,

> *we tried your stuff (the* `R` *package* `BayesTree`*) on the analysis of computer experiments and it seemed promising but it is too slow".*

Recode with MPI to make it faster!!

MPI: Message Passing Interface.

Two Steps.

Step 1. Rewrote serial code so that it is "leaner".

Intro
Trees and Ensemble Methods
BART
PBART: Parallel Bayesian Additive Trees
Other Parallel Approaches
End

Lean Code:

```
class tree {
public:
...
private:
   //--------------------------------
   //parameter for node
   double mu;
   //--------------------------------
   //rule: left if x[v] < xinfo[v][c]
   size_t v;
   size_t c;
   //--------------------------------
   //tree structure
   tree_p p; //parent
   tree_p l; //left child
   tree_p r; //right child
};
```

1 double, two integers, three pointers.

n	bart/BayesTree MCMC	new MCMC
1,000	57.725	14.057
2,000	136.081	27.459
3,000	211.799	40.483
4,000	298.712	54.454
5,000	374.971	66.900
6,000	463.861	82.084
7,000	545.995	95.737
8,000	651.683	107.911
9,000	724.577	120.778
10,000	817.711	135.764

The new code is 4 to 6 times faster!!

```
> 57.725/14.057
[1] 4.106495
> 817.711/135.764
[1] 6.023033
```

Note: We also use a more limited set of MCMC moves in the new code but we find we get the same fits.

Step 2:

Parallel MPI implementation.

Have $p+1$ processor cores.
Split data up into p equal chunks.

- core 0 is the master. It runs the MCMC.
- core i, $i = 1, 2, \ldots, p$ has data chunk i in memory.
- Each core has the complete model $((T_j, M_j)_{j=1}^m, \sigma)$ in memory.

Note: with MPI cores and associated memory may be on different machines.

Compare with openmp where the memory must be shared.

- core 0 is the master. It runs the MCMC.
- core i, $i = 1, 2, \ldots, p$ has data chunk i in memory.
- Each core has the complete model $((T_j, M_j)_{j=1}^m, \sigma)$ in memory.

Each MCMC step involves:

1. master core 0, initiates an MCMC step
   (e.g. change a single tree, draw σ).
2. master core 0, sends out a compute request (needed for MCMC step) to each slave node $i = 1, 2, \ldots p$.
3. Each slave core i computes on it's part of the data and sends the results back to master core 0.
4. master core 0 combines the results from the p slaves and updates the model using the results (e.g. changes a tree, obtains new σ draw).
5. master core 0, copies new model state out to all the slave cores.

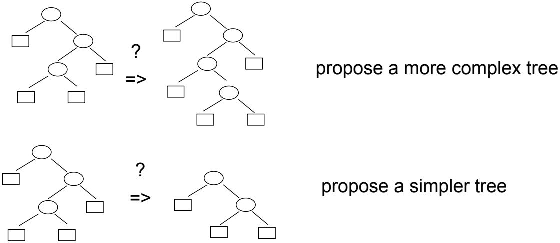
Keys to Parallel implementation:

- Lean model representation is cheap to copy out to all the slave cores.
- MCMC draws all depend on simple conditionally sufficient statistics which may be computed on the separate slave cores, cheaply sent back to the master, and then combined.

Even though the the overall model is complex, each local move is simple !!

Simple Sufficient Statistics:

Consider the birth-death step:



Given a tree, we just have $\{R_{ij}\} \sim N(\mu_j, \sigma^2)$ iid in j^{th} bottom node, where R is resids from the the other trees..

Evaluating a birth-death is just like testing equality of two normal means with an independent normal prior.

Sufficient statistic is just $\sum_i r_{ij}$ for two different j corresponding to left/right child bottom nodes.

Timing

We did lot's (I mean way too many..) timing experiments.

One that summarizes things nicely is as follows.

Look at the *efficiency* which is

$$E = \frac{T_{ser}/(p+1)}{T_{par}}$$

If $T_{par} = \frac{T_{ser}}{(p+1)}$ this would equal 1.

Intro

Trees and Ensemble Methods

BART

PBART: Parallel Bayesian Additive Trees

Other Parallel Approaches

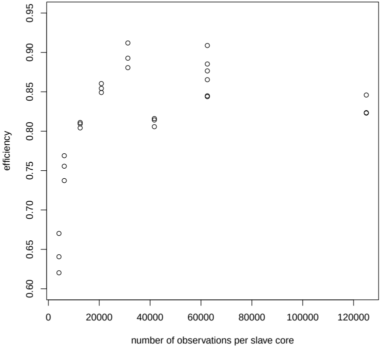
End

We timed 27 runs with $m \in \{50, 100, 200\}$,
$n \in \{100000, 500000, 1000000\}$, $p + 1 \in \{9, 17, 25\}$.

Efficiency ranges from .6 to .95.

pretty good !!!

If you have too few observations on a core the cost of message passing eats into the speed.



Other Parallel Approaches

In R

The easiest way to parallelize an MCMC is to just run several chains at the same time.

R (and lots of other environments) now support ways to do this easily at a high level (opposite of mpi).

To see use of R package parallel, look at:

```
~/do/research/bart_monotonic/repo_1-13-2015/code:
rpmonbart.R,
monbart.R,
cmonbart.cpp.
```

Probably easier with doParallel and foreach R packages.

Intro
Trees and Ensemble Methods
BART
PBART: Parallel Bayesian Additive Trees
Other Parallel Approaches
End

code snippet, mc.cores is the number of compute cores.

```
#--------------------------------------------------
   RNGkind("L'Ecuyer-CMRG")
   set.seed(seed)
   mc.reset.stream()

   mc.ndpost <- (ndpost %/% mc.cores)
   cat("ndpost: ",ndpost,"\n")
   cat("mc.ndpost: " ,mc.ndpost,"\n")

...

   cat("mc.cores: ",mc.cores,"\n")
   for(i in 1:mc.cores) {
      mcparallel(
         monbart(x.train,y.train, x.test,
...
               fmean=fmean,
               ntree=ntree,
               ndpost=mc.ndpost,
...
               printevery=printevery
         ),
         silent=(i!=1)
      )
   }

   post.list <- mccollect()
....
```

openmp

To get an R package out the effectively uses parallel computation I am trying openmp.

Much easier to use than mpi.

For example, just to get predictions (evaluations of $f(x)$ for many x can take time.

Easy to parallelize by just letting each core do a subset of the x's.

```
RcppExport SEXP cpwbart(
   SEXP _itrees,              //treedraws list from fbart
   SEXP _ix,                  //x matrix to predict at
   SEXP _itc                  //thread count
)
{
   Rprintf("*****In main of C++ for bart prediction\n");

   //--------------------------------------------------
   //get threadcount
   int tc = Rcpp::as<int>(_itc);
   cout << "tc (threadcount): " << tc << endl;
...

   Rcpp::NumericMatrix yhat(nd,np);
   std::fill(yhat.begin(), yhat.end(), 0.0);
...
   # ifndef _OPENMP
   cout << "\n***using serial code\n";
   getpred(0, nd-1, p, m, np,  xi,  tmat, px,  yhat);
   #else
   if(tc==1) {
      cout << "\n***using serial code\n";
      getpred(0, nd-1, p, m, np,  xi,  tmat, px,  yhat);
   } else {
      cout << "\n***using parallel code\n";
#     pragma omp parallel num_threads(tc)
      local_getpred(nd,p,m,np,xi,tmat,px,yhat);
   }
   #endif
...
}
```

```
void getpred(int beg, int end, size_t p, size_t m, size_t np,
                xinfo& xi, std::vector<vtree>& tmat, double *px, Rcpp::NumericMatrix& yhat)
{
   double *fptemp = new double[np];

   for(int i=beg;i<=end;i++) {
      for(int j=0;j<m;j++) {
         fit(tmat[i][j],xi,p,np,px,fptemp);
         for(int k=0;k<np;k++) yhat(i,k) += fptemp[k];
      }
   }

   delete [] fptemp;
}


void local_getpred(size_t nd, size_t p, size_t m, size_t np,
                 xinfo& xi, std::vector<vtree>& tmat, double *px, Rcpp::NumericMatrix& yhat)
{


   int my_rank = omp_get_thread_num();
   int thread_count = omp_get_num_threads();
   int h = nd/thread_count; int beg = my_rank*h; int end = beg+h-1;

   getpred(beg,end,p,m,np,xi,tmat,px,yhat);
}
#endif
```

End

Currently working on several extensions of the BART model (e.g. multinomial outcomes, heterskedastic modeling...)

I never use the old R package because it is so slow.

I am working out putting out new R package(s) with openmp to do the parallel computation.

Is openmp a good idea?

What else do I need to learn?

C++ 11 has "concurrency" built in.

Lot's of other things going on,... Julia ...GPUs ....

Intro
Trees and Ensemble Methods
BART
PBART: Parallel Bayesian Additive Trees
Other Parallel Approaches
End