BART::wbart: BART for Numeric Outcomes

Robert McCulloch and Rodney Sparapani

Contents

1 BART

BART is Bayesian Additive Regression Trees (see Chipman, George, and McCulloch).

We fit the basic model:

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

We use Markov Chain Monte Carlo to get draws from the posterior distribution of the parameter (f, σ).

In this vignette we look at BART::wbart which is the basic function in the R package **BART**.

1.1 Boston Housing Data

Let's just use the good old Boston housing data.

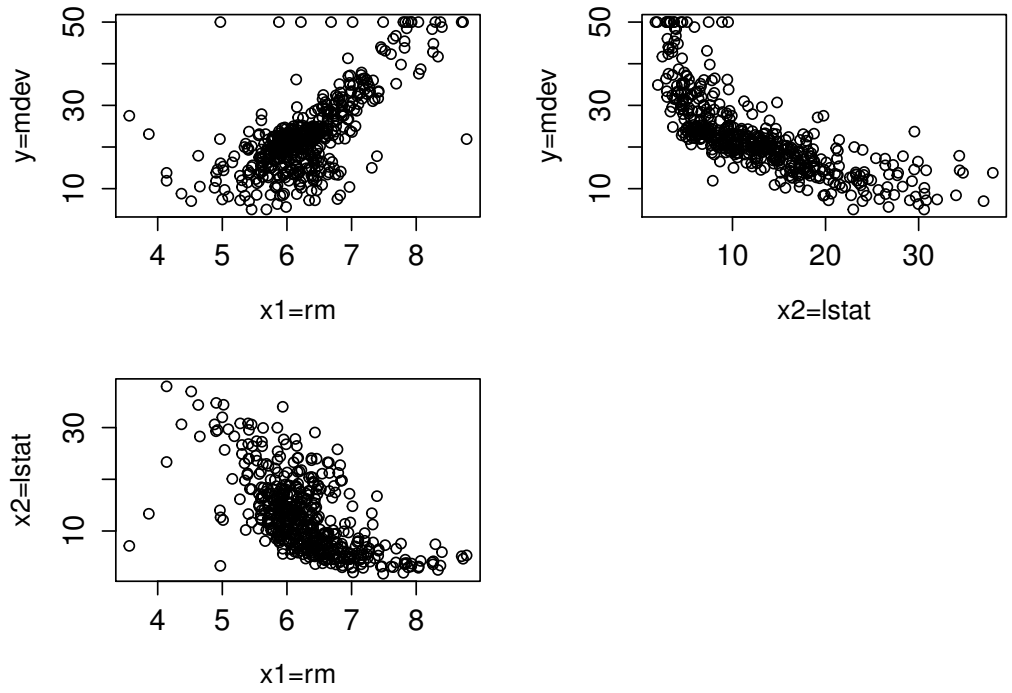
We'll predict the median house value, y=mdev, from x1 = rm (number of rooms) and x2=lsat (% lower status).

```
library(MASS)
x = Boston[,c(6,13)] #rm=number of rooms and lstat= percent lower status
y = Boston$medv # median value
head(cbind(x,y))
```

```
##      rm lstat    y
## 1 6.575  4.98 24.0
## 2 6.421  9.14 21.6
## 3 7.185  4.03 34.7
## 4 6.998  2.94 33.4
## 5 7.147  5.33 36.2
## 6 6.430  5.21 28.7
```

1.2 A Quick Look at the Data

```
par(mfrow=c(2,2))
par(mai=c(.8,.8,.2,.2))
plot(x[,1],y,xlab="x1=rm",ylab="y=mdev",cex.axis=1.3,cex.lab=1.2)
plot(x[,2],y,xlab="x2=lstat",ylab="y=mdev",cex.axis=1.3,cex.lab=1.2)
plot(x[,1],x[,2],xlab="x1=rm",ylab="x2=lstat",cex.axis=1.3,cex.lab=1.2)
```



1.3 Run wbart

```
library(BART) #BART package
set.seed(99) #MCMC, so set the seed
nd=200 # number of kept draws
burn=50 # number of burn in draws
bf = wbart(x,y,nskip=burn,ndpost=nd)
```

```
## *****Into main of wbart
## *****Data:
## data:n,p,np: 506, 2, 0
## y1,yn: 1.467194, -10.632806
## x1,x[n*p]: 6.575000, 7.880000
## *****Number of Trees: 200
## *****Number of Cut Points: 100 ... 100
## *****burn and ndpost: 50, 200
## *****Prior:beta,alpha,tau,nu,lambda: 2.000000,0.950000,0.795495,3.000000,5.979017
## *****sigma: 5.540257
## *****w (weights): 1.000000 ... 1.000000
## *****Dirichlet:sparse,theta,omega,a,b,rho,augment: 0,0,1,0.5,1,2,0
## *****nkeeptrain,nkeeptest,nkeeptestme,nkeeptreedraws: 200,200,200,200
## *****printevery: 100
## *****skiptr,skipte,skipteme,skiptreedraws: 1,1,1,1
##
## MCMC
## done 0 (out of 250)
## done 100 (out of 250)
## done 200 (out of 250)
## time: 0s
## check counts
## trcnt,tecnt,temecnt,treedrawscnt: 200,0,0,200
```

1.4 Results returned with a list

We stored the results of running wbart in the list **bf**.

```
names(bf)
##  [1] "sigma"           "yhat.train.mean" "yhat.train"
##  [4] "yhat.test.mean"  "yhat.test"       "varcount"
##  [7] "varprob"         "treedraws"       "proc.time"
## [10] "mu"              "varcount.mean"   "varprob.mean"
## [13] "rm.const"
length(bf$sigma)
## [1] 250
length(bf$yhat.train.mean)
## [1] 506
dim(bf$yhat.train)
## [1] 200 506
```

Remember, the training data has $n = 506$ observations, we had burn=50 burn-in draws and nd=200 kept draws.

Let's look at a couple key list components:

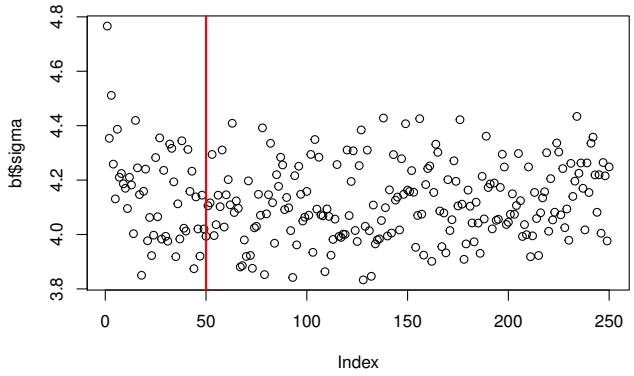
- **sigma**: burnin + kept (burn+nd) draws of σ.
- **yhat.train.mean**: j^{th} value is posterior mean of $f(x_j)$, f evaluated at the j^{th} training observation.
- **yhat.train**: i, j value is the i^{th} kept MCMC draw of $f(x_j)$.

1.5 Assess Convegence with σ Draws

As with any high-dimensional MCMC, assessing convergence may be tricky.

A nice simple thing to look at is the draws of σ. The parameter σ is the only identified parameter in the model and it also gives us a sense of the size of the errors.

```
plot(bf$sigma)
abline(v=burn,lwd=2,col="red")
```



Look's like it burned in almost right away.
Just one initial draw looking a bit bigger than the rest. Hopefully, subsequent variation is legitimate posterior variation.

In a more difficult problem you may see the σ draws initially declining as the MCMC search for fit.

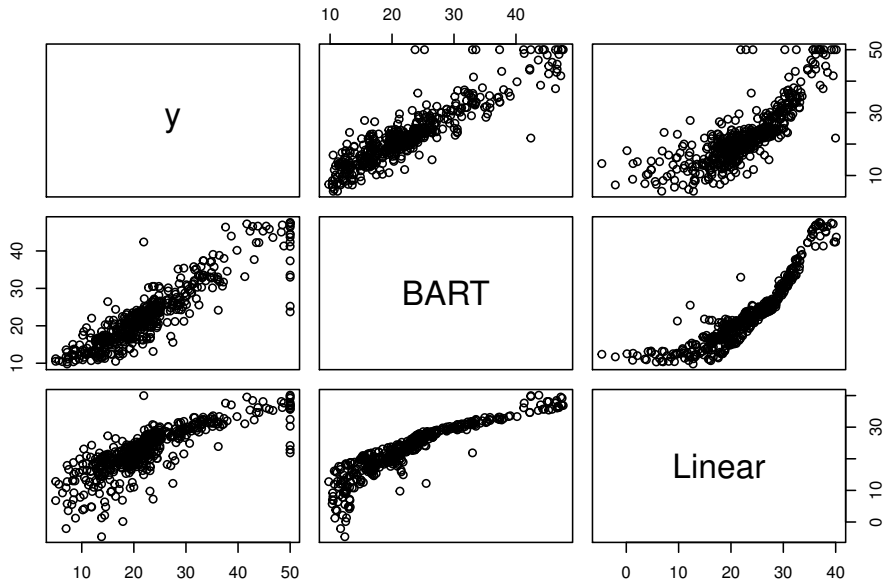
1.6 Look at in-sample Fit and Compare to a Linear Fit

Let's look at the in-sample BART fit (yhat.train.mean) and compare it to y=medv and the fits from a multiple linear regression.

```
lmf = lm(y~.,data.frame(x,y))
fitmat = cbind(y,bf$yhat.train.mean,lmf$fitted.values)
colnames(fitmat)=c("y","BART","Linear")
cor(fitmat)
```

```
##                y      BART    Linear
## y      1.0000000 0.9051200 0.7991005
## BART   0.9051200 1.0000000 0.8978003
## Linear 0.7991005 0.8978003 1.0000000
```

```
pairs(fitmat)
```

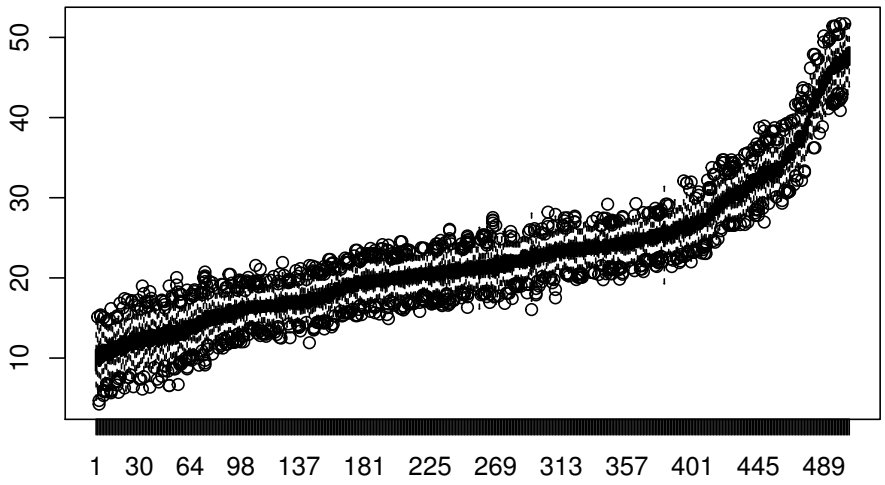


The BART fit is noticeably different from the linear fit.

1.7 A Quick Look at the Uncertainty

We order the observations by the fitted house value (yhat.train.mean) and then use boxplots to display the draws of $f(x)$ in each column of yhat.train.

```
ii = order(bf$yhat.train.mean) #order observations by predicted value
boxplot(bf$yhat.train[,ii]) #boxplots of f(x) draws
```



Substantial predictive uncertainty, but you are still pretty sure some houses should cost more that others!!

2 Using predict.wbart

We can get out of sample predictions two ways.

First, we can can just ask for them when we call wbart by supply a matrix of test x value.

Second, we can call a predict method.

2.1 Train and Test Data Sets

Let's split our data into train and test subsets.

```
n=length(y) #total sample size
set.seed(14)  # Dave Keon, greatest Leaf of all time!
ii = sample(1:n,floor(.75*n)) # indices for train data, 75% of data
xtrain=x[ii,]; ytrain=y[ii] # training data
xtest=x[-ii,]; ytest=y[-ii] # test data
cat("train sample size is ",length(ytrain)," and test sample size is ",length(ytest),"\n")
```

```
## train sample size is  379  and test sample size is  127
```

And now we can run `wbart` using the train to learn and predict at `xtest`.

First, we'll just give `xtest` to the `wbart` call.

```
set.seed(99)
bfp1 = wbart(xtrain,ytrain,xtest) #predict.wbart wants a matrix
```

```
dim(bfp1$yhat.test)
## [1] 1000  127
length(bfp1$yhat.test.mean)
## [1] 127
```

Now

- **yhat.test**: i, j value is the i^{th} kept MCMC draw of $f(x_j)$ where x_j is the j^{th} row of `xtest`.
- **yhat.test.mean**: j^{th} value is posterior mean of $f(x_j)$, f evaluated at the j^{th} row of `xtest`.

Alternatively, we could run wbart saving all the MCMC results and then call predict.wbart.

```
set.seed(99)
bfp2 = wbart(xtrain,ytrain)
yhat = predict(bfp2,as.matrix(xtest)) #predict wants a matrix
```

Then `yhat` and `bfp1$yhat.test` are the same.

```
dim(yhat)
## [1] 1000  127
summary(as.double(yhat-bfp1$yhat.test))
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
## -9.091e-09 -1.188e-09  2.455e-11  1.559e-12  1.186e-09  6.789e-09
```

#Thining ***

In our simple Boston housing data set `wbart` runs pretty fast.

But with more data and longer runs you may want to speed things up by saving less and then using predict.

Let's just keep a thinned subset of 200 tree ensembles.

```
set.seed(4) #Bobby Orr, let's change the seed
bfthin = wbart(xtrain,ytrain,nskip=1000,ndpost=10000,
                    nkeeptrain=0,nkeeptest=0,nkeeptestmean=0,nkeeptreedraws=200)
yhatthin = predict(bfthin,as.matrix(xtest)) #predict wants a matrix
```

```
dim(bfthin$yhat.train)
## [1]   0 379
dim(yhatthin)
## [1] 200 127
```

Now there are no kept draws of $f(x)$ for training x, and we have 200 tree ensembles to use in predict.wbart.

##The thinning arguments:

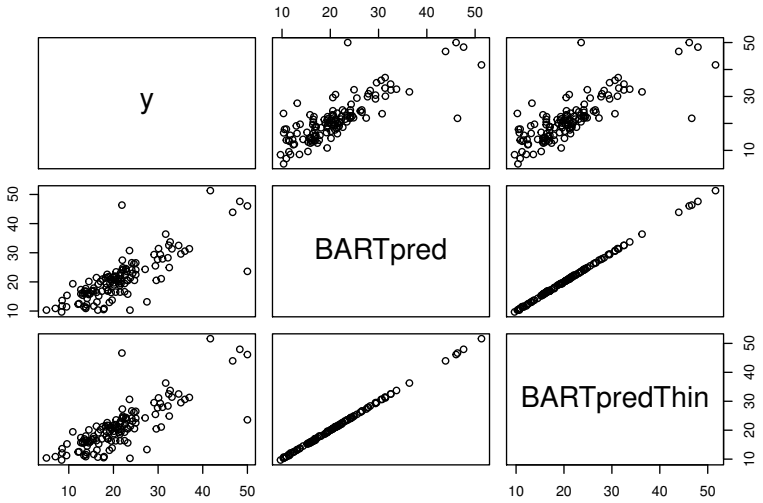
- **nkeeptrain** : number of $f(x)$ draws to save for training x.
- **nkeeptest** : number of $f(x)$ draws to save for test x.
- **nkeeptestmeam** : number of draws to use in computing yhat.test.mean.
- **nkeeptreedraws** : number of tree ensembles to keep.

The default values are to keep all the draws (e.g. nkeeptrain=ndpost).

Of course, if you keep 100 out of 100,000, you keep every 1,000th draw.

2.2 Let's have a look at the predictions

```
fmat=cbind(ytest,bfp1$yhat.test.mean,apply(yhatthin,2,mean))
colnames(fmat) = c("y","BARTpred","BARTpredThin")
pairs(fmat)
```



Recall, the **BARTpred** predictions are from a run are from a BART run with seed=99 and all default values.

The **BARTpredThin** are from 200 kept trees out of a long run with 1,000 burn-in and 10,000 kept draws and seed=4.

Interesting how similar they are !!!!