Generic Algorithm + Bagging

Myra O'Regan

October 31, 2017

Generic Version

$$F(x) = c_0 + \sum_{m=1}^{M} c_m T_m(\mathbf{x})$$

- Use tree for $T_m(x)$
- Each Tree described by a set of parameters $T(\mathbf{x} : \mathbf{p_m})$ - splits in tree
- Each possible base learner can be thought as a point in a high-dimensional parameter space P

What we are trying to do

$$\{\widehat{c_m}, \widehat{\mathbf{p_m}}\} = \underset{\{c_m, \mathbf{p_m}\}_0^{\mathbf{M}}}{min} \sum_{i=1}^{N} L\left(y_i, c_0 + \sum_{m=1}^{M} c_m T(\mathbf{x} : \mathbf{p_m})\right)$$

- $y_i$are the target values
- We do this in two stages
- Step 1: Chose the points $\mathbf{p_m}$ - build the trees
- Step 2: We either find c_i or they are predetermined

How do we build the trees

Step 1- Choose $\{\mathbf{p_m}\}_1^{\mathbf{M}}$

$F_0(\mathbf{x}) = \mathbf{0}$

For m=1 to M $\{$

$$\mathbf{p_m} = \arg\min_{\mathbf{p}} \sum_{\mathbf{i} \in \mathbf{S_n}(\eta)} \mathbf{L}(\mathbf{y_i}, \mathbf{F_{m-1}}(\mathbf{x_i}) + \mathbf{T}(\mathbf{x_i}; \mathbf{p}))$$

$$T_m(\mathbf{x}) = \mathbf{T}(\mathbf{x_i}; \mathbf{p})$$

$$F_m(\mathbf{x}) = \mathbf{F_{m-1}}(\mathbf{x}) + \nu \mathbf{T_m}(\mathbf{x})$$

$\}$

write $\{T_m(x)\}_1^M$

Some notes

- L = Loss function
- η controls how to change the data sampling
- How large a sample do we take
- ν controls how much the approximation built up to the present iteration influences the next iteration.
- $F_{m-1}(x) = \nu \sum_{k=1}^{m-1} T_k(x)$

Ensembles - Determining $c_i's$

- We can use simple formula
- Determine them at each step along the way
- Calculate them afterwards using penalized regression function
- Post processing

Ensembles - Determining $c_i's$

- At this point all base learners have been selected
- Have to the determine the $c_i's$
- Do this by regularised regression

$$\{c_m\} = \underset{c_m}{\text{argmin}} \sum_{i=1}^{N} L\Big((y_i, c_0 + \sum_{m=1}^{M} c_m T_m(x_i))\Big) + \lambda.P(c) \tag{1}$$

- P(c) is the complexity penalty and λ controlling the amount of regularisation.

Bagging

- Stands for bootstrap aggregation by Brieman
- $L(y, \hat{y}) = (y - \hat{y})^2$
- $\upsilon = 0$
- $\eta = N/2$- discuss
- $T_m(x)$:large unpruned tree or ????
- $c_0 = 0, c_m = 1/M$
- Use **ipred package in R**

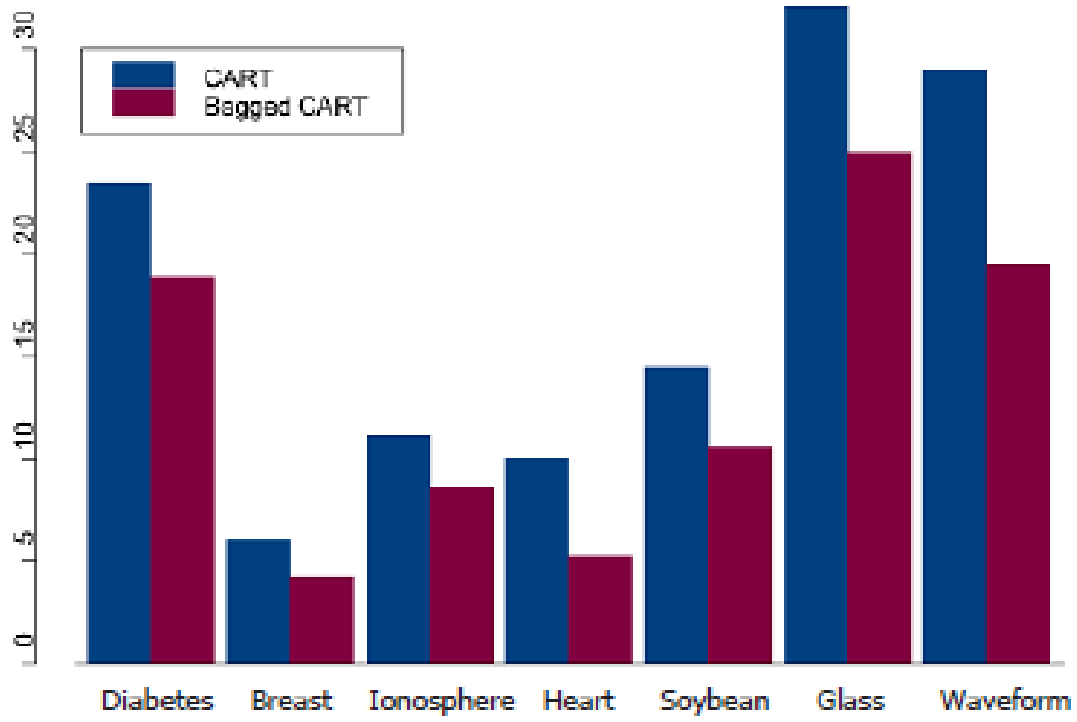
Bagging

- Simple structure of a tree
- Sample size of n
- Probability of being omitted in a single $=(1 - \frac{1}{n})$
- Probability of being omitted in all draws $=(1 - \frac{1}{n})^n$
- Approximately $=1/e = 0.368$
- Approximately 37% of sample excluded
- Can be used for test set
- Works best when predictors are unstable

More Bagging

- Reduces variance
- Loose simple structure
- Bagging may make something worse
- Works best when predictors are unstable

Comparison of trees vs bagging



Testing bagging on simulated data

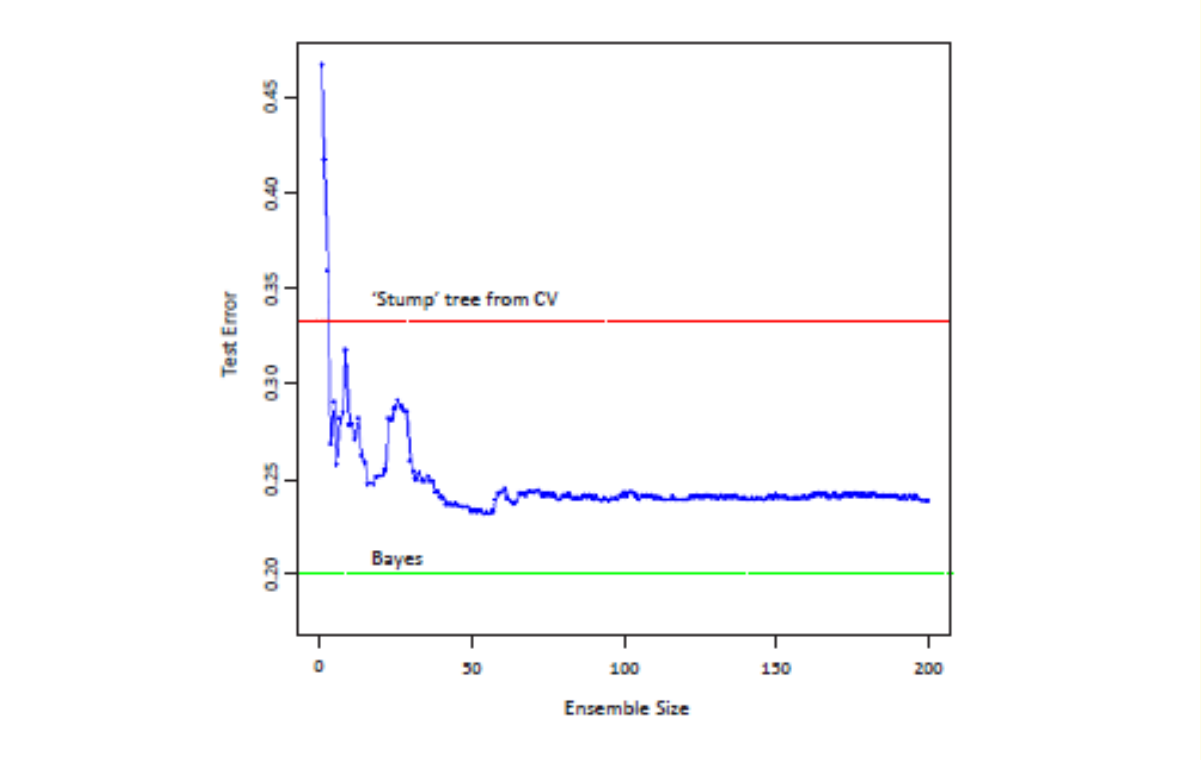
- 5 predictor variables - standard normal distribution
- Pairwise correlation of 0.95
- Response - 2 classes

$$P(y = 1)|x_1 \leq 0.5) = 0.2$$

$$P(y = 1)|x_1 > 0.5) = 0.8$$

- Minimum possible error rate =0.2
- Known as Bayes Error rate

Bayes Error rate



Experiment on bagging

- 3 independent variables
- Number of trees
- Pruning vs no pruning
- Size of sample to select

Some R code

```
expand.grid command
x1=c(1,2,3)
x2=c(10,20)
x=expand.grid(x1,x2)

baggerr=vector(length=??, mode="numeric")
```

- Up to you to look up the package ipred

Suggested R code

```
library(ipred)
churn=read.csv("D:/data mining 2012/churn.csv")
#set up parameters levels
cp<-c(?,?,?)
ntrees<-c(?,?,?)
rep<-c(?,?,?,?,?)
ps<-expand.grid(cp,ntrees,rep)
baggerr<-vector(length=60, mode="numeric")
```

And more ....

```
for (i  in 1:60 ){

obaggo=bagging(churn~.,data=churn[,3:18],
control=rpart.control(minsplit=2,cp=ps[i,1]
   ,xval=0),
  coob=TRUE,nbagg=ps[i,2])
baggerr[i]=obaggo$err
}
results=data.frame(baggerr,ps)
library(reshape)
results=rename(results,c(Var1="cp",
    Var2="Notrees" ,Var3="reps"))
```