Ensemble Methods: Bagging, Boosting and Random Forests

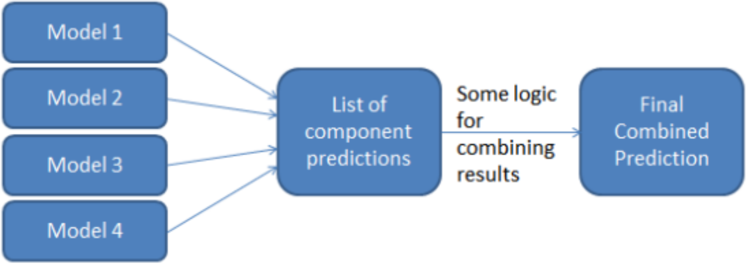
June 17, 2015

Agenda

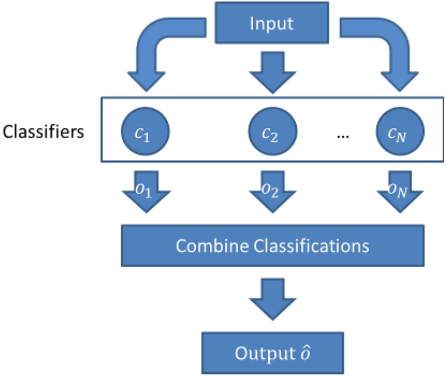
- Ensemble models: What are they?
- Bagging: What, Why and How?
- Randomforests
- Boosting: adaBoost

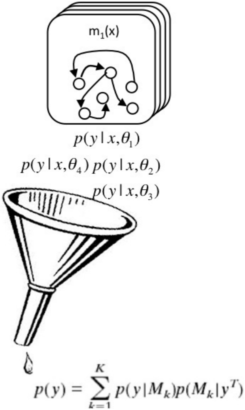
Ensemble Methods

- Combine learners: Obtain prediction from this family
- Individual learners are not that great sometimes
- Train weak learners and combine them to create an aggregate model.



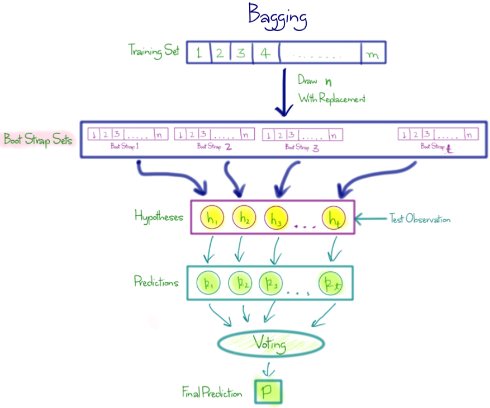
- How models are created determines the type of ensemble
- Bootstrap samples, Samples with different weights





Bagging: What?

- Create ensembles from : bootstrapped samples(with replacement)
- Classification problem : Majority vote
- Regression problem : Average the result



Bagging-Process

- 1. Take 'n' bootsrapped sampeles
- 2. Train models (regression-classification) on each of these 'n' samples
- 3. Aggregate the results- Take average(regression) or Majority vote (classification)

Basic Idea of Bootstrap

Using the original sample as the population, and draw N samples from the original sample (which are the bootstrap samples). Defining the estimator using the bootstrap samples.

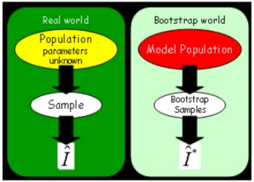


Figure: Real World versus Bootstrap World

Formal Definition

- Suppose we have a regression problem, with training data Λ=$\{(x_1, y_1), (x_2, y2)......(x_n, y_n)\}$
- Take 'B' bootstrapped samples out of Λ, let the set of these samples be represented by $\left\{\Lambda^{Bi}\right\}; i = 1, 2, 3, ...B$
- For each bootstrapped sample Λ^{Bi} fit a model to get $\hat{f}(x)^{Bi}$, thus obtaining a set $\left\{\hat{f}(x)^{Bi}\right\}; i = 1, 2, 3, ...B$
- In order to generate the bagged regression estimate, for each $xi \in \Lambda$, compute $\frac{1}{B}\Sigma\hat{f}(x)^{Bi}; i = 1, 2, 3......B$.
- The set of all predcitions will be $\left\{\frac{1}{B}\Sigma\hat{f}(x)^{B}; i = 1, 2, 3......n\right\}, This$ *will become the bagged regression estimate*

Formal Definition

- Suppose we now have a classification problem. Let $C(\Lambda, x)$ be the classifier, eg a tree, producing a class label on our data Λ at point x.
- To bag $C(\Lambda, x)$ we take B bootstrap samples $\Lambda^{*1}, \Lambda^{*2}, ...\Lambda^{*B}$ from our training data set Λ
- Then bagged estimate $\hat{C}(x) = \text{Majority Vote}\left\{C(\Lambda^{*B}, x)_{b=1}^{B}\right\}$

Bagging-Where it works best?

- Bagging works best with unstable models such as CART, Nueral Nets and Subset selection linear regression models. With stable algorithms such as KNN bagging doesn't improve any performance.

Bagging-Code demo

- Let us take a look at how bagging improves perfromance
- Dataset: Ionosphere from UCI (Classification task)
- Build a tree model (CART)
- Bagg this model and see the difference in performance
- Performance measures will be kappa and summary confusiion matrix

```
#Downloading data from UCI sever
library(RCurl)
url<-c("https://archive.ics.uci.edu/ml/machine-learning-databases/ionosphere/ionosphere.data")
url<-getURI(url)
ionosphere<-read.csv(textConnection(url),header=T)
names(ionosphere)[35]<-"Good_Bad"

#Data partition
library(caret)
index<-createDataPartition(y = ionosphere$Good_Bad,times = 1,p = 0.70,list = FALSE)
train_i<-ionosphere[index,]
test_i<-ionosphere[-index,]

#CART Tree model##
library(rpart)
tree_m<-rpart(Good_Bad~.,data=train_i)

#Accuracy tree model #

pred<-unname(predict(tree_m,test_i,type="class"))
str(test_i$Good_Bad)


##  Factor w/ 2 levels "b","g": 1 1 1 1 1 2 1 2 2 1 ...


str(pred)


##  Factor w/ 2 levels "b","g": 1 2 1 1 2 2 1 2 2 1 ...


confusionMatrix(pred,test_i$Good_Bad,positive = "g")


## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction  b  g
##          b 26  2
##          g 11 65
##
##                Accuracy : 0.875
##                  95% CI : (0.7957, 0.9317)
##     No Information Rate : 0.6442
##     P-Value [Acc > NIR] : 9.994e-08
##
##                   Kappa : 0.7116
##  Mcnemar's Test P-Value : 0.0265
##
##             Sensitivity : 0.9701
##             Specificity : 0.7027
##          Pos Pred Value : 0.8553
##          Neg Pred Value : 0.9286
##              Prevalence : 0.6442
##          Detection Rate : 0.6250
##    Detection Prevalence : 0.7308
##       Balanced Accuracy : 0.8364
##
##        'Positive' Class : g
##

#Bagged model##

library(adabag)

mod_bag<-bagging(Good_Bad~.,data = train_i)

p<-predict.bagging(mod_bag,test_i)
```

```
predicted<-p$class

predicted<-factor(predicted,levels=c("b","g"))

confusionMatrix(predicted,test_i$Good_Bad,positive = "g")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  b  g
##          b 33  4
##          g  4 63
##
##                Accuracy : 0.9231
##                  95% CI : (0.854, 0.9662)
##     No Information Rate : 0.6442
##     P-Value [Acc > NIR] : 3.604e-11
##
##                   Kappa : 0.8322
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9403
##             Specificity : 0.8919
##          Pos Pred Value : 0.9403
##          Neg Pred Value : 0.8919
##              Prevalence : 0.6442
##          Detection Rate : 0.6058
##    Detection Prevalence : 0.6442
##       Balanced Accuracy : 0.9161
##
##        'Positive' Class : g
##
```

```
#Problem of few significant predictors dominating
variable<-rep("a",100)
for(i in 1:length(mod_bag$trees))
{
  variable[i]<-as.character(mod_bag$trees[[i]]$frame[1,1])
}
variable
```

```
##   [1] "X0.85243" "X0.85243" "X0.85243" "X0.99539" "X0.85243" "X0.85243"
##   [7] "X0.41078" "X0.85243" "X0.85243" "X0.41078" "X0.85243" "X0.85243"
##  [13] "X0.85243" "X0.85243" "X0.99539" "X0.85243" "X0.41078" "X0.83398"
##  [19] "X0.85243" "X0.85243" "X0.85243" "X0.85243" "X0.85243" "X0.85243"
##  [25] "X0.83398" "X0.85243" "X0.85243" "X0.41078" "X0.83398" "X0.85243"
##  [31] "X0.41078" "X0.99539" "X0.85243" "X0.85243" "X0.85243" "X0.41078"
##  [37] "X0.85243" "X0.85243" "X0.85243" "X0.85243" "X0.83398" "X0.85243"
##  [43] "X0.85243" "X0.85243" "X0.85243" "X0.41078" "X0.85243" "X0.99539"
##  [49] "X0.85243" "X0.85243" "X0.83398" "X0.83398" "X0.85243" "X0.85243"
##  [55] "X0.85243" "X0.99539" "X0.99539" "X0.41078" "X0.83398" "X0.41078"
##  [61] "X0.85243" "X0.85243" "X0.85243" "X0.85243" "X0.41078" "X0.85243"
##  [67] "X0.85243" "X0.85243" "X0.41078" "X0.99539" "X0.99539" "X0.83398"
##  [73] "X0.83398" "X0.99539" "X0.41078" "X0.41078" "X0.99539" "X0.85243"
##  [79] "X0.85243" "X0.83398" "X0.85243" "X0.83398" "X0.83398" "X0.85243"
##  [85] "X0.85243" "X0.99539" "X0.83398" "X0.83398" "X0.85243" "X0.85243"
##  [91] "X0.85243" "X0.85243" "X0.99539" "X0.85243" "X0.85243" "X0.85243"
##  [97] "X0.99539" "X0.41078" "X0.85243" "X0.85243"
```

```
table(variable)
```

```
## variable
## X0.41078 X0.83398 X0.85243 X0.99539
##       14       14       59       13
```

- Now we will use bagging to solve regression problem and see how bagging improves accuracy
- Dataset: Ionosphere
- Error metric to be analyzed will be RMSE

```
#Predict burnt area
setwd("/media/ramius/E2A02905A028E1B1/Work/Misc/Ensemble/")
forestfires<-read.csv("forestfires.csv")
set.seed(100)
index<-sample(nrow(forestfires),0.70*(nrow(forestfires)))
library(rpart)
reg_tree<-rpart(area~.,forestfires[index,])

predicted<-predict(reg_tree,newdata = forestfires[-index,-13])
actual<-forestfires[-index,13]
sum((predicted-actual)^2/(nrow(forestfires)-length(index)))


## [1] 2161.086
```

```
library(ipred)
set.seed(100)
reg_bag<-ipredbagg(forestfires[index,13],forestfires[index,-13])

sum((predict(reg_bag,forestfires[-index,])-actual)^2/(nrow(forestfires)-length(index)))


## [1] 634.1497
```

```
#Problem of few significant predictors dominating
variable<-rep("a",length(reg_bag$mtrees))
for(i in 1:length(reg_bag$mtrees))
{
  variable[i]<-as.character(reg_bag$mtrees[[i]][2]$btree$frame[1,1])
}
variable
```

```
## [1] "DMC"   "FFMC"  "day"   "DMC"   "temp"  "month" "Y"     "temp"
## [9] "DMC"   "temp"  "temp"  "temp"  "temp"  "temp"  "temp"  "temp"
## [17] "DMC"  "DMC"   "temp"  "temp"  "temp"  "month" "temp"  "temp"
## [25] "temp"

table(variable)

## variable
##   day   DMC  FFMC month  temp     Y
##     1     5     1     2    15     1
```

Bagging-R pipline

- R has a lots of packages that impliment bagging: ipred, adabag and caret
- If the one wants to build ensembles of tree models then ipred or adabag would suffice
- caret has a lot of bagging algorithms: MARS, LDA
- caret also allows tuning bagged model created using ipred or adabag
- The code demo will include the detailed implimentation of bagging using both ipred and adabag.
- German Credit card data set will be used for the demo

```
##Bagging pipeline demo##

#Read data
setwd("/media/ramius/E2A02905A028E1B1/Work/Misc/Ensemble/")
gc<-read.csv("germancredit.csv")
gc$Default<-factor(gc$Default,levels = c(0,1),labels = c("Good","Bad"))
#Use caret to partition the data
library(caret)
index<-createDataPartition(y = gc[,1],p = 0.70,list = FALSE,times=1)
gc_train<-gc[index,]
gc_test<-gc[-index,]

#Use ipred to create bagged models

library(ipred)
bag1<-ipredbagg(gc_train$Default,gc_train[,-1],nbagg = 1000)

#Checking the accuracy on test data
pred<-predict(bag1,gc_test[,-1],type="class")
pred<-factor(pred,levels=c("Good","Bad")) #Need to relevel the factor variable
confusionMatrix(pred,reference = gc_test[,1],positive = "Good")
#Detach ipred
detach("package:ipred", unload=TRUE)

#Use adabag to create bagged predictors
library(adabag)
bag2<-bagging(Default~.,gc_train,mfinal = 1000)

pred<-predict.bagging(bag2,gc_test)$class
pred<-factor(pred,levels=c("Good","Bad"))
confusionMatrix(pred,reference = gc_test[,1],positive = "Good")
detach("package:adabag", unload=TRUE)
#Comparison with a single tree model
library(rpart)
```

```
tree1<-rpart(Default~.,data=gc_train)
pred<-predict(tree1,gc_test[,-1],type="class")
pred<-factor(pred,levels=c("Good","Bad"))
confusionMatrix(pred,reference = gc_test[,1],positive = "Good")
```