CUNY School of Professional Studies

SPS.CUNY.EDU

Lecture 12
2020 Spring Data-622
Ensemble: bagging, boosting

Raman Kannan

Instructor Email Address: Raman.Kannan@sps.cuny.edu

Acknowledgements:
Generous support from IBM Power Systems Academic Initiative
IBM PSAI provides computing infrastructure for free

Script for Algorithms

Develop the Intuition
Understand the assumptions
Develop the mathematics
Run the algorithms
Learn to interpret the result/output
Predict using the model
Learn to determine the performance
Distinguish training/testing error
Differentiate between overfitting/underfitting
Techniques to improve performance

Intuition (Resampling)

So far we have seen single algorithm strategies at work, in an effort to minimize variance and bias. Averaging is a proven smoothing technique.

Now we want to incorporate hybrid strategies where we leverage multiple algorithms.

We first create random disjoint datasets, run the same model and then take the average (Cross Validation, Bagging).

Run a model, identify the observations which are misclassified and fine tune to correct the misclassification. This is called boosting.

Bagging – Bootstrap AGGregation

Bootstrapping involves – resampling with replacement. Effective when we left with a small sample to begin with – CV may not be possible.

Bagging – run model on many bootstrapped samples and take the average for regression (majority voting for classification)

Generate a Model (bagging)

```
path<-"C:/Users/rk215/cuny/L11-tree/binary.csv"
admit_data<-read.csv(path,head=TRUE);
head(admit_data)

#make some columns factors
fad<-data.frame(as.factor(admit_data$admit),admit_data$gre,
admit_data$gpa,as.factor(admit_data$rank))
names(fad)<-names(admit_data)
# create test and training set
set.seed(43)
tstset<-sample(400,120,replace=FALSE) # 30% hold out test set
admit_trdata<-fad[-tstset,]
admit_tstdata<-fad[tstset,]

# generate model
set.seed(43)
iter=200
bagfit.admit<-bagging(admit~.,data=admit_trdata,coob=T,nbagg=iter)
```

Performance

```
bag.pred<-predict(bagfit.admit,admit_tstdata[,-1])
# probabilities -->
bag.pred.result<-data.frame(actual=admit_tstdata[,1],predicted=bag.pred)
# confusion matrix (aka contingency table)
table(actual=admit_tstdata[,1],predicted=bag.pred)
pradmit.number<-as.numeric(predicted_admit)

prediction.admit.bag<-prediction(as.numeric(bag.pred),admit_tstdata$admit)
performance.admit.bag<-
performance(prediction.admit.bag,measure='tpr', x.measure='fpr')
auc.admit.bag<-performance(prediction.admit.bag,measure='auc’)

# plot and display AUC
plot(performance.admit.bag, main="ROC Curve for Bagged ADMIT data")
auc.admit.bag@y.values[[1]]
```

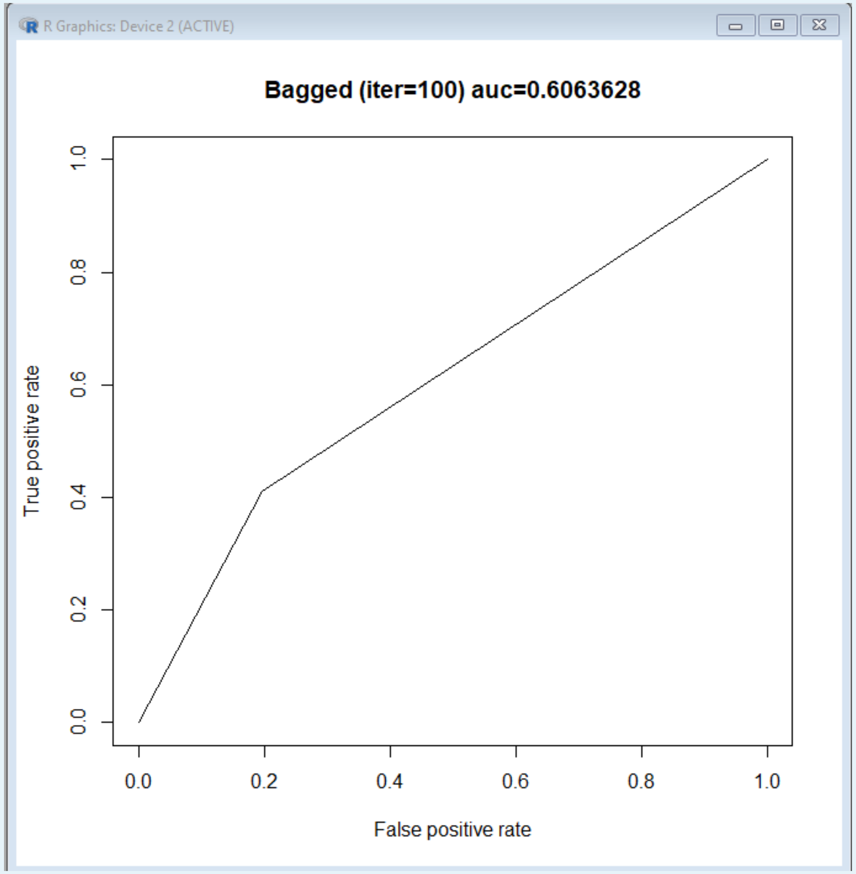
More need not necessarily result in improvement

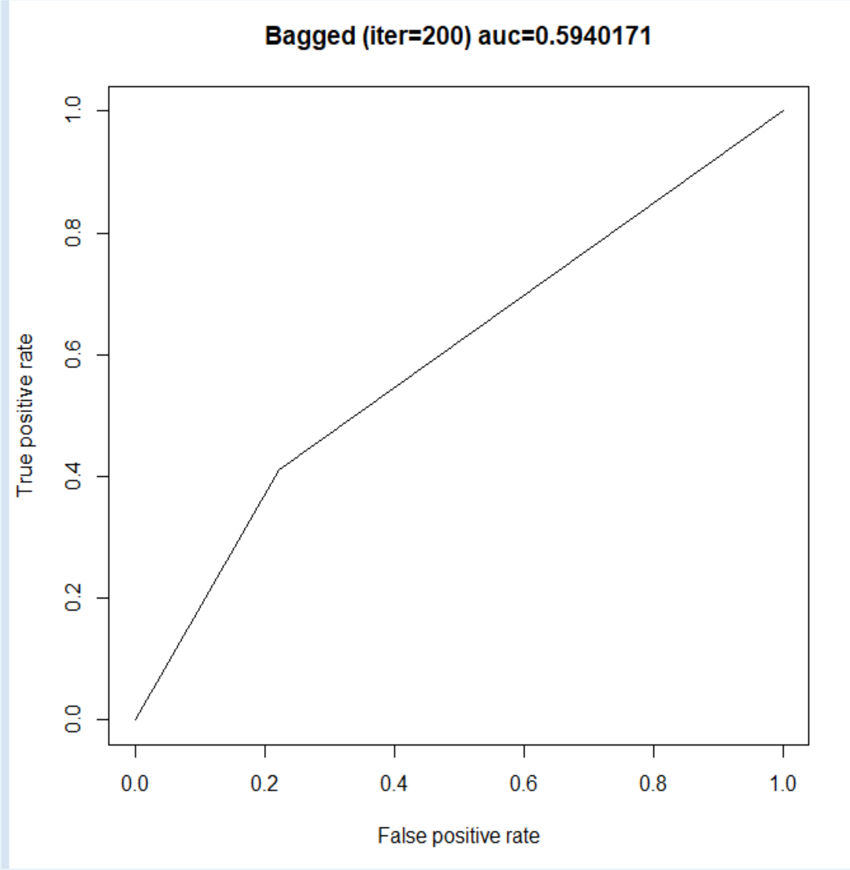
```
> set.seed(43)
> iter=100
> bagfit.admit<-bagging(admit~.,data=admit_trdata,coob=T,nbagg=iter)
>
> bag.pred<-predict(bagfit.admit,admit_tstdata[,-1])
> # probabilities -->
> bag.pred.result<-data.frame(actual=admit_tstdata[,1],predicted=bag.pred)
> table(actual=admit_tstdata[,1],predicted=bag.pred)
      predicted
actual  0  1
     0 65 16
     1 23 16
>
> prediction.admit.bag<-prediction(as.numeric(bag.pred),admit_tstdata$admit)
> performance.admit.bag<-performance(prediction.admit.bag,measure='tpr', x.measure='fpr')
> auc.admit.bag<-performance(prediction.admit.bag,measure='auc')
> plot(performance.admit.bag)
> auc.admit.bag@y.values[[1]]
[1] 0.6063628
```

```
> set.seed(43)
> iter=200
> bagfit.admit<-bagging(admit~.,data=admit_trdata,coob=T,nbagg=iter)
>
> bag.pred<-predict(bagfit.admit,admit_tstdata[,-1])
> # probabilities -->
> bag.pred.result<-data.frame(actual=admit_tstdata[,1],predicted=bag.pred)
> table(actual=admit_tstdata[,1],predicted=bag.pred)
      predicted
actual  0  1
     0 63 18
     1 23 16
>
> prediction.admit.bag<-prediction(as.numeric(bag.pred),admit_tstdata$admit)
> performance.admit.bag<-performance(prediction.admit.bag,measure='tpr', x.measure='fpr')
> auc.admit.bag<-performance(prediction.admit.bag,measure='auc')
> plot(performance.admit.bag)
> auc.admit.bag@y.values[[1]]
[1] 0.5940171
```

Plot (iter=100)

```
plot(performance.admit.bag, main="Bagged (iter=100) auc=0.6063628")
```





Code (iter=200)

```
 set.seed(43)
iter=200
bagfit.admit<-bagging(admit~.,data=admit_trdata,coob=T,nbagg=iter)

 bag.pred<-predict(bagfit.admit,admit_tstdata[,-1])
 # probabilities -->
 bag.pred.result<-data.frame(actual=admit_tstdata[,1],predicted=bag.pred)
 table(actual=admit_tstdata[,1],predicted=bag.pred)
 prediction.admit.bag<-prediction(as.numeric(bag.pred),admit_tstdata$admit)
 performance.admit.bag<-performance(prediction.admit.bag,measure='tpr', x.measure='fpr')
 auc.admit.bag<-performance(prediction.admit.bag,measure='auc')


auc.admit.bag@y.values[[1]]
main_title<-paste("Bagged (iter=",iter,") auc=",auc.admit.bag@y.values[[1]],sep="")
plot(performance.admit.bag, main=main_title)
```

XGBoost and gbm:gbm data load

```
path<-"C:/Users/rk215/cuny/L11-tree/binary.csv"

#ad<-read.csv(path,head=T)
#fad<-data.frame(as.factor(ad$admit),ad$gre,ad$gpa,as.factor(ad$rank))

admit_data<-read.csv(path,head=TRUE);
head(admit_data)

#make some columns factors

fad<-data.frame(as.factor(admit_data$admit),admit_data$gre,
admit_data$gpa,as.factor(admit_data$rank))
names(fad)<-names(admit_data)
set.seed(43)
tstset<-sample(400,120,replace=FALSE) # 30% hold out test set

admit_trdata<-fad[-tstset,]
admit_tstdata<-fad[tstset,]
```

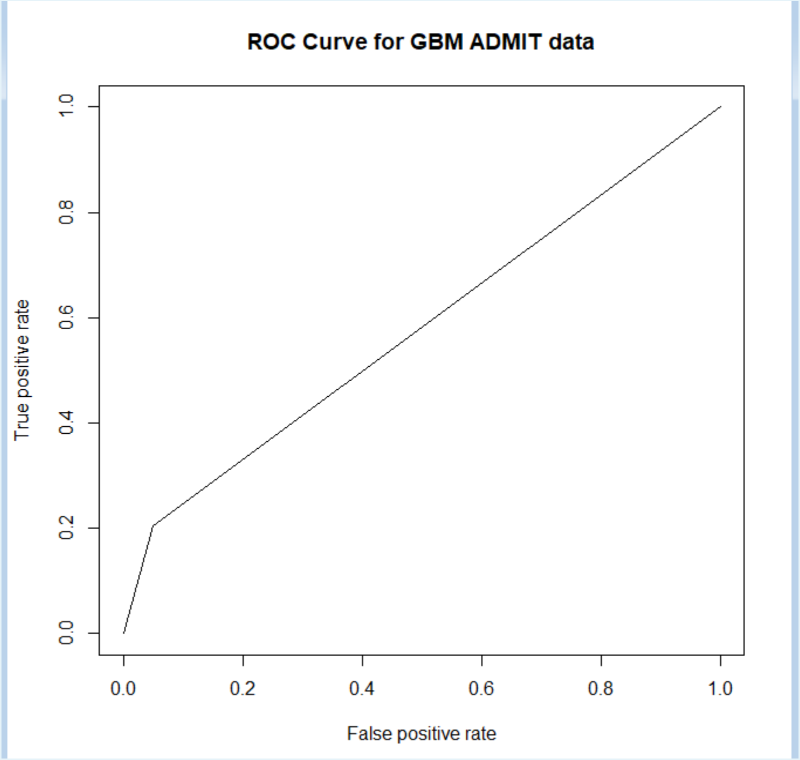
XGBoost and gbm

```
# model
mod_gbm = gbm(admit ~.,
            data = admit_trdata,
            distribution = "multinomial",
            cv.folds = 10,
            shrinkage = .01,
            n.minobsinnode = 10,
            n.trees = 200)

print(mod_gbm)

pred = predict.gbm(object = mod_gbm,
                newdata = admit_tstdata,
                n.trees = 200,
                type = "response")

labels<-colnames(pred)[apply(pred,1,which.max)]
result<-data.frame(admit_tstdata$admit,labels)
```



Performance

```
confusionMatrix<-
table(actual=result$admit_tstdata.admit,
predicted=result$labels)
pradmit.number<-as.numeric(result$labels)
actual.number<-as.numeric(result$admit_tstdata.admit)
pr<-prediction(pradmit.number,actual.number)
auc_data<-performance(pr,"tpr","fpr")
plot(auc_data,main="ROC Curve for GBM ADMIT data")
aucval<-performance(pr,measure="auc")
aucval@y.values[[1]]
```

```
> # confusionMatrix
>
> confusionMatrix<-table(actual=result$admit_tstdata.admit,predicted=result$lab$
>
> pradmit.number<-as.numeric(result$labels)
>
> actual.number<-as.numeric(result$admit_tstdata.admit)
> pr<-prediction(pradmit.number,actual.number)
> auc_data<-performance(pr,"tpr","fpr")
> plot(auc_data,main="ROC Curve for GBM ADMIT data")
> aucval<-performance(pr,measure="auc")
> aucval@y.values[[1]]
[1] 0.5778727
```

References

http://uc-r.github.io/gbm_regression
https://www.24tutorials.com/machine-learning/xgboost-for-classification/
https://github.com/dmlc/xgboost/blob/master/R-package/demo/caret_wrapper.R
http://www.jmlr.org/proceedings/papers/v42/chen14.pdf
https://www.hackerearth.com/practice/machine-learning/machine-learning-algorithms/beginners-tutorial-on-xgboost-parameter-tuning-r/tutorial