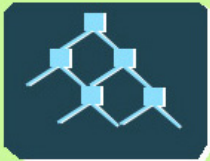


INTELLIGENT TRADING TECH



Discovering Edge using Machine Learning, Data Mining, and Bio Inspired Algorithms to augment traditional Systematic Development.

thursday, january 28, 2016

Kaggle Winton Stock Market Challenge - Post-Mortem

Recently, I participated in a [Kaggle contest](#) sponsored by [Winton Capital](#). The contest provided various market related data and asked participants to predict intraday and next two day return forecasts over unseen future data. Prizes included \$50,000 in monetary rewards and a chance to join Winton Capital's team of research scientists. I think these kinds of contests are a great way for modern data companies to scour hidden talent and in return, they offer great real world problems for both aspiring, as well as seasoned scientists to explore and validate ideas and methodologies with like minded individuals. I decided to write a post on my approach, so that users can repeat the methods and have some ideas behind the thought process.

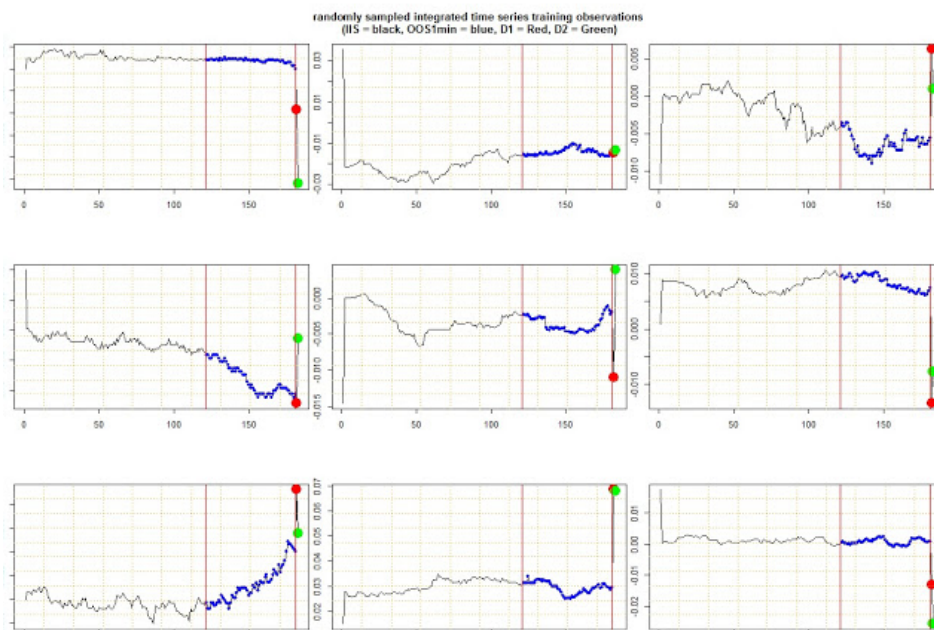


Fig 1. Sample observations of integrated training 1 min. time series with black iis intraday and blue oos intraday, D1 and D2 are red and green, respectively.

The bad news is that I didn't fare very well in the contest (edit* looks like I made top 25%), however, there were a lot of interesting issues that surfaced as

my blog list

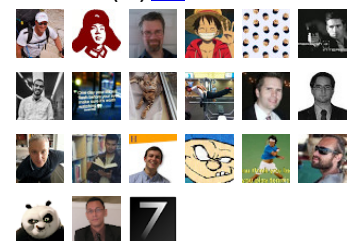
- [R bloggers](#) 6 hours ago
- [QuantShare](#) 5 weeks ago
- [CSS Analytics](#) 4 months ago
- [Tr8dr](#) 6 months ago
- [Systematic Investor](#) 3 years ago
- [Adaptive Trader](#) 4 years ago
- [Quantity](#) 5 years ago

related trading sites

- [quantstart](#)
- [quantocracy](#)
- [quantnews](#)
- [stackexchange.com](#)
- [www.nuclearphynance.com](#)
- [www.elitetrader.com](#)
- [www.portfolioworkstation.com](#)

followers

Followers (86) [Next](#)



Follow

blog archive

- ▼ 2016 (1)
- ▼ January (1)

the contest progressed. I joined the contest fairly late and read some posts about how the data was altered in midstream. To understand this better, it might help to describe the data in more detail. The original dataset was comprised of two subsets, a train dataset with 40000 obs. and 211 features, and a test dataset with 40000 obs. and 147 features. The features could be broken into an ID column, a mix of 25 unlabeled continuous and discrete features, and 183 ordered time series returns. The time series returns were further broken down into $-D_1, -D_2, 1minD, +D_1, +D_2$: the 1 min data represented a range of 179 intraday 1 min returns. The D_1, D_2 returns were prior overnight, and post daily returns, respectively. The goal was to predict the intraday and post two day returns beyond the original time series data points. So we were given roughly 2/3 of the prior returns and needed to predict 1/3 of the out of sample returns. Contestants were asked to submit the predicted test data results and received feedback on a reserved portion of the private dataset that were displayed as public leaderboard results. Typically, contestants can use the public leaderboard as a gauge of how well their models are performing on hidden, out of sample data. The overall final performance on the private leaderboard is withheld until the contest ends.

Many posters quickly figured out that the task was rather daunting and suspected that a zero baseline (just predict all zeros) or median of time series observations were pretty difficult to beat. However, what had changed in the contest, was that there was a possibility that some posters could have gamed the data and reverse engineered it to obtain very good scores on the public leaderboard. In response, the contest administrators decided to add additional test data to the original private test dataset, it was upped from 40000 unseen observations to 120000 observations. This was done to try to deter any gaming of the system. It also made gauging cross validation much more difficult as we were only given leaderboard feedback on 20000 of the observations. So we had to try to figure out the most robust model, given only 25% of training data, and make it robust to 75% of new hidden data! This with performance feedback on only 16.7% of the hidden data results being displayed on the public leaderboard. Additionally, some posters suspected that the data structures of the new dataset had been distorted from its original characteristics in order to avoid any gaming. I'm hoping to get more feedback from the contest and providers, in order to understand the data better.

My own approach was to use a gradient boosted model from the R gbm package. GBMs have historically performed well on kaggle competitions, have been shown to be one of the best performing models on a variety of datasets (see [Kuhn, Applied Predictive Modelling](#)), and are known to be a good choice for a robust out of the box model. Some of the benefits of GBMs, are ability to deal with NAs and handling of unscaled data, so we don't have to worry much about pre-processing data. As I didn't have much time to explore other options, I (perhaps wrongly) spent most of my time on improving the gbm

Kaggle Winton Stock
Market Challenge -
Post-Mortem...

- 2015 (1)
- 2013 (3)
- 2012 (6)
- 2011 (8)
- 2010 (25)

about me

Intelligent Trading

I've been trading full time for over 10 years and wish to share some of the knowledge I've acquired with others. I have a particular interest in machine learning and how the current research in this area holds much unexplored potential towards the area of systematic trading development. Although I've studied many different texts on machine learning, I've often found sparse practical examples related to trading. My goal is to share some concrete examples for the layman to be able to build and replicate. . . . intelligenttradingtech@yahoo.com

[View my complete profile](#)

[View My Stats](#)

cross-validated performance. Like others, I found that my public leaderboard score was consistently at the high range of my cv performance results. Since there was so much discussion about data manipulation, I (again likely wrongly) stubbornly believed in the cv results, and assumed that they might have biased the leaderboard set towards the worse performing data. In retrospect, I should have maybe spent more time on extremely simple models (like means and medians) as opposed to fitting a relatively small set of training data with a complex model. I also found many odd data characteristics, like intraday data columns that were perfectly trending up in the training set and down in the test set, some data had very little noise, others had lots of noise. My intuition was that not all of the time series data were from true financial time series, and that the distributions were not stable from training to test sets. This made the problem more difficult, IMO, then dealing with real data and having knowledge of the underlying data. I attempted to filter for different training and test samples, by using t-tests for comparisons, and setting a threshold for removing unusually different time series data columns.

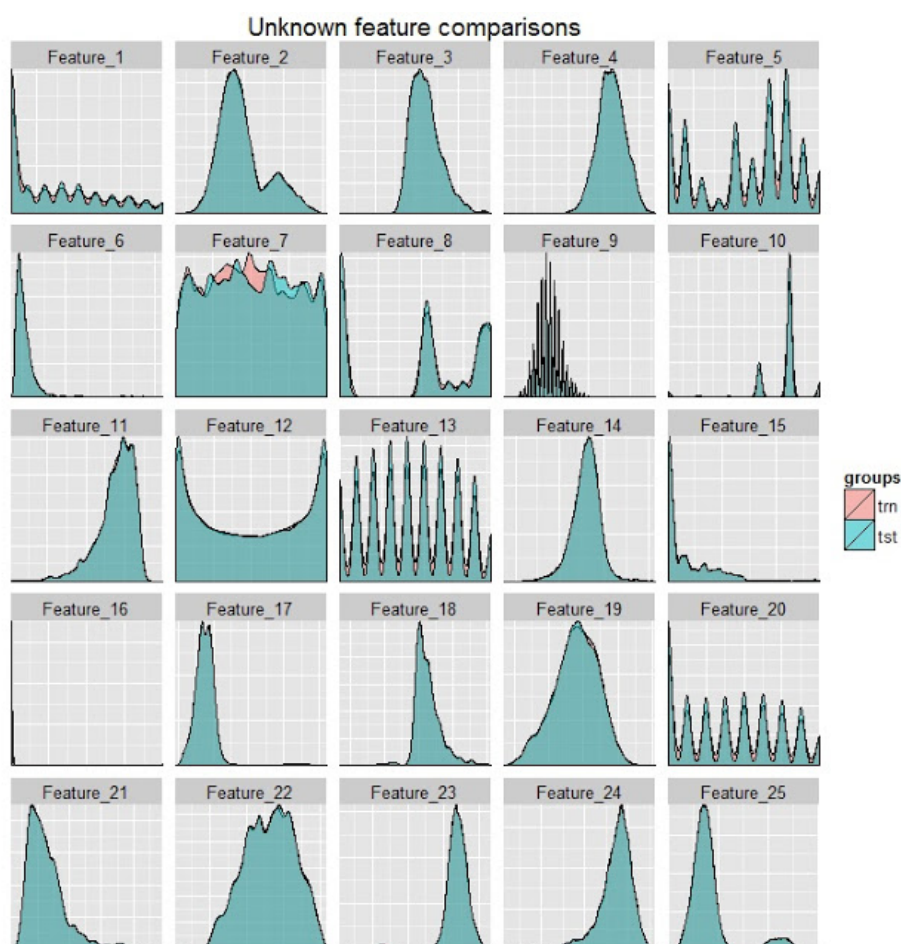


Fig. 2. First 25 unlabeled features highlighting unstable trn/tst distributions in Feature_7.

Some posters observed that Feature_7 was unstable between train and test samples. Unfortunately, that was also the same feature that was highlighted as the most important feature in both D1, and D2 predictions from my GBM

model. I left it in as I didn't have enough time to change strategies at that point and figured that I still had reasonable CV estimates that would beat the zero and median estimates on average. I ended up just using median estimates of training features for the out of sample 1 min predictions, and used a GBM to predict the next two out of sample days. We were also given a vector of loss weights for the training data and were instructed to use weighted mean absolute error as our loss function. R's GBM models allow us to set 'Laplace' as an absolute loss function. I also wrote out the loss function to manually record it in the cross validated loops and get a better feel for the sensitivity of the validation folds. A loop was created to record performance on 5 validation folds across several tree lengths. The optimal value was then used to rebuild final training models on all the training data, that could then be used to predict values for the test fold.

Fig. 3. below shows cross validated results from a simple GBM model using 5 fold validation, with shrinkage = 0.1, n.minobsinnode=50, interaction.depth =1. The red line shows the median value for the zero baseline as a comparison to the WMAE.val folds across different tree iterations. Notice that selecting, say n.trees = 900, yields a range of results that were all superior to the baseline median. I would expect somewhere around 1765 for this model; however, as explained earlier, the private test set was difficult to match performance as distribution of features were different. I used the run below to illustrate, but the actual model I built had a lower shrinkage of 0.01 and much better performance, but took a long time to run -- as I had to use about 8000 trees.

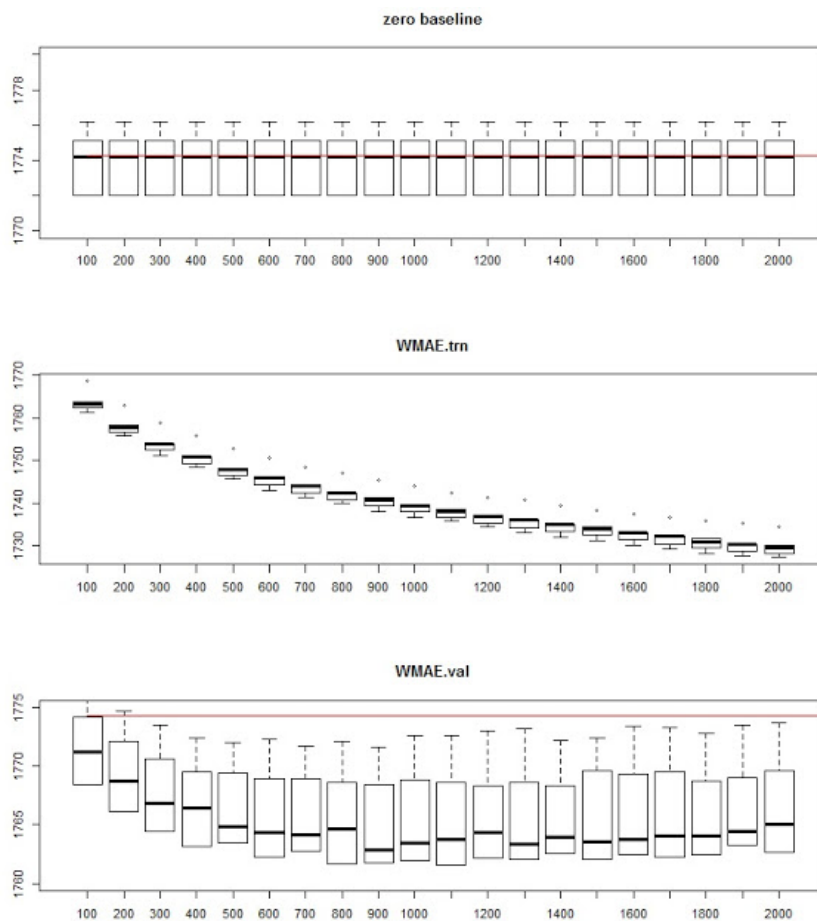


Fig 3. 5 fold validation results with simple GBM model, shrinkage=0.1
n.minobsinnode=50 interaction.depth = 1

In conclusion, I was able to use R to build a GBM model with good cross validated performance against the baseline zero metric, but the model did not perform very well on the hidden private data. I believe two big factors were differences in train and test datasets, as well as a much larger hold out dataset that was different enough that the true results were much worse than 5 fold CV would predict. It's equally possible that the model was too complex for the very noisy dataset.

Some posters were wondering about why WMAE was used as opposed to RMSE. In financial time series, MAE is often used as a loss function as it is more robust to outliers. You can imagine why large outliers will not only skew the data and fit, but their contributions will effectively be squared using RMSE.

Lastly, although it was a great exercise in machine learning and real world modeling. In my experience, targeting continuous variables and using weighted mean absolute value loss functions may not be the best fitness to target. The great thing that it illustrates is the practical difficulty in predicting financial time series. Thanks to Winton and Kaggle for a great experience!

#####

Code is below. Note* I used Pretty R to embed the code but there are color clashes with the background. You can copy and paste into a local browser to view more clearly. If anyone has suggestions on how to change the PrettyR font colors, please let me know.

```
library(gbm)
library(doParallel)
library(reshape2)

#trn <- read.csv("E:/Kaggle/Winton/train.csv") # 40k X 211
#tst <- read.csv("E:/Kaggle/Winton/test_2.csv") #120k X 147

trn <- readRDS("E:/Kaggle/Winton/trn.rds"); tst <- readRDS("E:/Kaggle/Winton/tst.rds")

interp <- function(x){
  x <- as.numeric(x)
  if(is.na(x[1])) x[1] <- x[!is.na(x)][1]
  #if(is.na(x[length(x)])) x[length(x)] <- x[length(x)-1]
  xsub <- x[2:length(x)]
  #gap.na <- which(is.na(xsub))

  for(k in 2:length(xsub)){
    if(is.na(xsub[k])) xsub[k] <- xsub[k-1]
  }
  x <- c(x[1],xsub)
  return(x)
}

disc.impute <- function(x){

  xfactor <- levels(factor(x))
  x[is.na(x)] <- sample(xfactor,length(which(is.na(x))),replace=TRUE)

  return(as.numeric(x))
}

cont.impute <- function(x){

  x[is.na(x)] <-
    runif(length(which(is.na(x))),
    min(range(na.omit(x))),max(range(na.omit(x))))
```

```

return(x)
}

generate.folds <- function(idx, k){

fold.idx <- holdout <- kfold <- list()

fold.len <- floor(length(idx)/k) # if idx does not divide evenly ignore
last few observations

for(i in 1:(k-1)){
holdout[[i]] <- sample(idx[!(idx%in%unlist(holdout))],fold.len)
kfold[[i]] <- idx[!(idx%in%holdout[[i]])]
}
holdout[[k]] <- idx[!(idx%in%unlist(holdout))]
kfold[[k]] <- idx[!(idx%in%holdout[[k]])]

folds <- list(kfold,holdout)
names(folds) <- c('trn','val')

return(folds)
}

qlimiter <- function(x,qlim) {
ifelse(x < quantile(x,(1-qlim),na.rm=TRUE),quantile(x,(1-qlim),na.rm
=TRUE),ifelse(
x > quantile(x,qlim,na.rm=TRUE),quantile(x,qlim,na.rm=TRUE),x))
}

feat.dist.sel <- function(mx,my,pth){
out <- 0
for(i in 1:dim(mx)[2]){
out[i] <- t.test(mx[,i],my[i])$p.value
}
sel <- which(out > pth)
return(sel)
}

interp.med <- function(x) {
x[is.na(x)] <- median(na.omit(x)); x
}

# column breakpoints
f1 <- 2; f25 <- 26; tsiis1 <- 27; tsiis2 <- 147; tsoos1 <- 148; tsoos2 <- 207;
tsD1 <- 208; tsD2 <- 209;
wtint <- 210; wtdly <- 211; trn.obs <- dim(trn)[1]; tst.obs <- dim(tst)[1];
clip.th <- 0.9999
discrete.features <- c(1,5,8,9,10,13,16,20)

```



```

continuous.features <- c(2,3,4,6,7,11,12,14,15,17,18,19,21,22,23,24,
25)

# preprocess and clean data
all.iis <- rbind(trn[,1:tsiis2], tst[,1:tsiis2])

all.trn.iis.ts <- trn[,tsiis1:tsiis2]
all.trn.oos.ts <- trn[,tsoos1:tsoos2]
all.trn.oos.ts.cln <- apply(all.trn.oos.ts,2,interp.med)

all.trn.oos.D1 <- trn[,tsD1]
all.trn.oos.D2 <- trn[,tsD2]
all.trn.oos.D1.clip <- qlimiter(all.trn.oos.D1,qlim=clip.th)
all.trn.oos.D2.clip <- qlimiter(all.trn.oos.D2,qlim=clip.th)

all.iis.ts <- all.iis[,tsiis1:tsiis2]
all.tst.iis.ts <- all.iis.ts[(trn.obs+1):(trn.obs+tst.obs),]

ts.sel <- feat.dist.sel(all.trn.iis.ts,all.tst.iis.ts,pth=.05)
all.iis.ts.sel <- all.iis.ts[,ts.sel]

all.iis.ts.cln <- apply(all.iis.ts.sel,2,interp.med)
all.trn.iis.ts.cln <- all.iis.ts.cln[1:trn.obs,]
all.tst.iis.ts.cln <- all.iis.ts.cln[(trn.obs+1):(trn.obs+tst.obs),]

# K FOLD CV
folds <- list(); n.fold <- 5
folds <- generate.folds(seq(dim(trn)[1]), n.fold)

#gbmGrid <- expand.grid(interaction.depth = c(5,9,15), n.trees = seq
(400,1000,by=200),
#shrinkage = 0.1, n.minobsinnode = 50)
out <- res <- list()
tune.par <- seq(10,200,10)

for(z in 1:length(tune.par)){
cl <- makeCluster(n.fold)
registerDoParallel(cl)

res <- list()
rmcol = 0

n.trees <- tune.par[z]

```



```

interaction.depth <- 1
n.minobsinnode <- 50
shrinkage <- 0.1

res <- foreach(k=1:n.fold, .combine=rbind) %dopar%
{

library(gbm)

trn.fld <- folds$trn[[k]]
val.fld <- folds$val[[k]]

# Naive estimates use zero baseline as predictions

trn.iis <- all.trn.iis.ts.cln[trn.fld,] ; val.iis <- all.trn.iis.ts.
cln[val.fld,]
trn.int.oos <- all.trn.oos.ts.cln[trn.fld,] ; val.int.oos <- all.trn
.oos.ts.cln[val.fld,]
trn.day.D1 <- all.trn.oos.D1.clip[trn.fld] ; val.day.D1 <- all.trn.o
os.D1.clip[val.fld]
trn.day.D2 <- all.trn.oos.D2.clip[trn.fld] ; val.day.D2 <- all.trn.o
os.D2.clip[val.fld]

wt.int.trn <- trn[trn.fld,wtint]; wt.int.val <- trn[val.fld,wtint]
wt.day.trn <- trn[trn.fld,wtdly]; wt.day.val <- trn[val.fld,wtdly]

trn.zro <- rep(0,dim(trn.iis)[1]); #zero pred for intraday
val.zro <- rep(0,dim(val.iis)[1]); #zero pred for intraday
trn.err.int <- wt.int.trn*abs(trn.int.oos-trn.zro)
val.err.int <- wt.int.val*abs(val.int.oos-val.zro)
trn.err.D1 <- wt.day.trn*abs(trn.day.D1-trn.zro)
val.err.D1 <- wt.day.val*abs(val.day.D1-val.zro)
trn.err.D2 <- wt.day.trn*abs(trn.day.D2-trn.zro)
val.err.D2 <- wt.day.val*abs(val.day.D2-val.zro)

WMAE.trn <- mean(c(as.matrix(trn.err.D1),as.matrix(trn.err.D2),as.ma
trix(trn.err.int))) #oos trn wmae
WMAE.val <- mean(c(as.matrix(val.err.D1),as.matrix(val.err.D2),as.ma
trix(val.err.int))) #oos tst wmae
WMAE.zro.all <- mean(c(WMAE.trn,WMAE.val))

# feature selection and cleaning outliers
all.feats <- all.iis[,f1:f25]
all.feats[,3] <- qlimiter(all.feats[,3],.9999)
all.feats[,11] <- qlimiter(all.feats[,11],.9999)
trn.feats <- all.feats[trn.fld,]

```

```

val.feats <- all.feats[val.fld,]

trn.day.1 <- data.frame(trn.feats,all.trn.iis.ts.cln[trn.fld,],all.trn.oos.D1.cln[trn.fld])
names(trn.day.1)[dim(trn.day.1)[2]]<-'D1'
trn.day.2 <- data.frame(trn.feats,all.trn.iis.ts.cln[trn.fld,],all.trn.oos.D2.cln[trn.fld])
names(trn.day.2)[dim(trn.day.2)[2]]<-'D2'

val.day.1 <- data.frame(val.feats,all.trn.iis.ts.cln[val.fld,],all.trn.oos.D1.cln[val.fld])
names(val.day.1)[dim(val.day.1)[2]]<-'D1'
val.day.2 <- data.frame(val.feats,all.trn.iis.ts.cln[val.fld,],all.trn.oos.D2.cln[val.fld])
names(val.day.2)[dim(val.day.2)[2]]<-'D2'

# Build gbm models, here: use same ntrees for both D1,D2
gbm.trn.mod.D1 <- gbm(D1 ~ .,data=trn.day.1,distribution="laplace",
  weights = wt.day.trn,
  n.trees=n.trees, interaction.depth=interaction.depth, shrinkage = shrinkage, n.minobsinnode =n.minobsinnode)

gbm.trn.mod.D2 <- gbm(D2 ~ .,data=trn.day.2,distribution="laplace",
  weights = wt.day.trn,
  n.trees=n.trees, interaction.depth=interaction.depth, shrinkage = shrinkage, n.minobsinnode =n.minobsinnode)

# TRN fold predictions

gbm.trn.pred.1 <- predict(gbm.trn.mod.D1, newdata=trn.day.1[, -dim(trn.day.1)[2]],
  n.trees=n.trees)
gbm.trn.pred.2 <- predict(gbm.trn.mod.D2, newdata=trn.day.2[, -dim(trn.day.2)[2]],
  n.trees=n.trees)
gbm.trn.act.1 <- all.trn.oos.D1[trn.fld]
gbm.trn.act.2 <- all.trn.oos.D2[trn.fld]
#par(mfrow=c(2,1))
#plot(gbm.trn.pred.1~gbm.trn.act.1)
#plot(gbm.trn.pred.2~gbm.trn.act.2)

trn.err.day.1 <- wt.day.trn*abs(gbm.trn.act.1 -gbm.trn.pred.1)
trn.err.day.2 <- wt.day.trn*abs(gbm.trn.act.2 -gbm.trn.pred.2)

```

```

trn.err.dly <- data.frame(trn.err.day.1,trn.err.day.2)
trn.err.int <- trn.err.int # from previous median estimates, wt.int.
trn*abs(trn.int.oos-trn.med)
WMAE.trn <- mean(c(as.matrix(trn.err.dly),as.matrix(trn.err.int)))

# VAL fold predictions

gbm.val.pred.1 <- predict(gbm.trn.mod.D1, newdata=val.day.1[, -dim(va
l.day.1)[2]],
n.trees=n.trees, interaction.depth=interaction.depth, shrinkage = sh
rinkage, n.minobsinnode =n.minobsinnode)
gbm.val.pred.2 <- predict(gbm.trn.mod.D2, newdata=val.day.2[, -dim(va
l.day.2)[2]],
n.trees=n.trees, interaction.depth=interaction.depth, shrinkage = sh
rinkage, n.minobsinnode =n.minobsinnode)
gbm.val.act.1 <- all.trn.oos.D1[val.fld]
gbm.val.act.2 <- all.trn.oos.D2[val.fld]

#par(mfrow=c(2,1))
#plot(gbm.val.pred.1~gbm.val.act.1)
#plot(gbm.val.pred.2~gbm.val.act.2)

val.err.day.1 <- wt.day.val*abs(gbm.val.act.1-gbm.val.pred.1)
val.err.day.2 <- wt.day.val*abs(gbm.val.act.2-gbm.val.pred.2)
val.err.dly <- data.frame(val.err.day.1,val.err.day.2)
val.err.int <- val.err.int # from previous median estimates, wt.int.
trn*abs(trn.int.oos-trn.med)
WMAE.val <- mean(c(as.matrix(val.err.dly),as.matrix(val.err.int)))

res[[k]] <- data.frame(k,interaction.depth,n.trees,shrinkage,n.minob
sinnode,WMAE.zro.all,WMAE.trn,WMAE.val)

}

out[[z]] <- t(do.call(rbind,res))
stopCluster(cl)
gc()
}

# aggregate and plot data
{
out.folds <- do.call(rbind,out)
mean.res <- aggregate(cbind(WMAE.zro.all,WMAE.trn,WMAE.val)~n.trees,
data=out.folds,mean)

plot(mean.res$WMAE.trn~mean.res$n.trees,type='o',pch=19,col='red')

```

```

lines(mean.res$WMAE.val~mean.res$n.trees,type='o',pch=19,col='blue')

yr <- range(mean.res$WMAE.zro.all,mean.res$WMAE.trn,mean.res$WMAE.val)

plot(mean.res$WMAE.zro.all~mean.res$n.trees,type='o',col='blue',pch=
19,ylim=yr,main=
'8 flds sweep',ylab='8 FOLD CV WMAE')
text(mean.res$WMAE.zro.all~mean.res$n.trees,cex=.8,col='black',pos=3
,label = round(mean.res$WMAE.zro.all,2))
lines(mean.res$WMAE.trn~mean.res$n.trees,type='o',pch=19,col='green'
,ylim=yr)
text(mean.res$WMAE.trn~mean.res$n.trees,cex=.8,col='black',pos=3,label
= round(mean.res$WMAE.trn,2))
lines(mean.res$WMAE.val~mean.res$n.trees,type='o',pch=19,col='red',ylim=yr)
text(mean.res$WMAE.val~mean.res$n.trees,cex=.8,col='black',pos=3,label
= round(mean.res$WMAE.val,2))
legend(800,1*yr[2],c("mean.res$WMAE.zro","mean.res$WMAE.trn","mean.res$WMAE.val"),
col=c('blue','green','red'),pch=19)

```

```
dev.new()
```

```

CV.df <- out.folds[,c(3,c(6:8))]
CV.df <- transform(CV.df,ID=seq(dim(CV.df)[1]))

```

```

df.m <- melt(CV.df, id.var=c('ID','n.trees'))
boxplot(value~variable+n.trees,df.m)
}

```

```

#####
#####      FINAL all TRN model generate/WMAE
#####

```

```

n.trees <- 900
interaction.depth <- 1
n.minobsinnode <- 50
shrinkage <- 0.1

```

```

all.feats <- all.iis[,f1:f25] #limit useless or sparse feature outliers

```

```

all.feats[,3] <- qlimiter(all.feats[,3],.9999)
all.feats[,11] <- qlimiter(all.feats[,11],.9999)
trn.feats <- all.feats[1:trn.obs,]

```

```

trn.day.1 <- data.frame(trn.feats,all.trn.iis.ts.cln,all.trn.oos.D1.clip)
names(trn.day.1)[dim(trn.day.1)[2]]<-'D1'
trn.day.2 <- data.frame(trn.feats,all.trn.iis.ts.cln,all.trn.oos.D2.clip)
names(trn.day.2)[dim(trn.day.2)[2]]<-'D2'

trn.day.1 <- data.frame(trn.feats,all.trn.oos.D1.clip)
names(trn.day.1)[dim(trn.day.1)[2]]<-'D1'
trn.day.2 <- data.frame(trn.feats,all.trn.oos.D2.clip)
names(trn.day.2)[dim(trn.day.2)[2]]<-'D2'

wt.day.trn <- trn[,wtdly]

gbm.trn.mod.D1 = gbm(D1 ~ .,data=trn.day.1, distribution="laplace",weights = wt.day.trn,n.trees=n.trees,shrinkage=0.1 ,cv.folds=5)
gbm.trn.mod.D2 = gbm(D2 ~ .,data=trn.day.2, distribution="laplace",weights = wt.day.trn,n.trees=n.trees,shrinkage=0.1 ,cv.folds=5)

#####
#####      OOS TST GENERATE
#####

tst.feats <- all.feats[(1+trn.obs):dim(all.feats)[1],]

tst.day.1 <- data.frame(tst.feats,all.tst.iis.ts.cln)
tst.day.2 <- data.frame(tst.feats,all.tst.iis.ts.cln)

all.oos.int.pred <- sapply(all.trn.oos.ts,median,na.rm=TRUE)
tst.pred.mtx <- t(matrix(rep(all.oos.int.pred,dim(tst)[1]),length(all.oos.int.pred),dim(tst)[1]))

gbm.tst.pred.1 <- predict(gbm.trn.mod.D1, newdata=tst.day.1,
n.trees = n.trees)
gbm.tst.pred.2 <- predict(gbm.trn.mod.D2, newdata=tst.day.2,
n.trees = n.trees)

tst.submission.mtx <- cbind(tst.pred.mtx,gbm.tst.pred.1,gbm.tst.pred.2)

tstpred <- as.vector(t(tst.submission.mtx))

submission <- read.csv("E:/Kaggle/Winton/sample_submission_2.csv")
submission$Predicted <- tstpred

```

```
write.csv(submission, "E:/Kaggle/Winton/submission1.csv", row.names=FALSE)
```

```
subtst <- read.csv("E:/Kaggle/Winton/submission1.csv")
```

Created by Pretty R at inside-R.org

posted by intelligent trading at 1:22 am 5 comments  links to this post
labels: kaggle, r, winton

friday, april 3, 2015

Review: Machine Learning An Algorithmic Perspective 2nd Edition

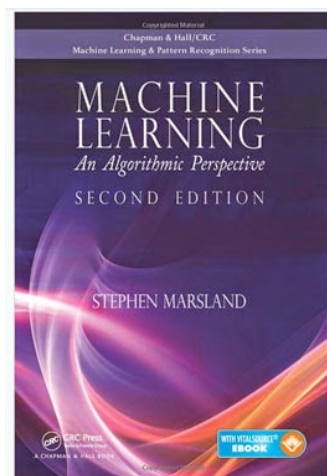


Fig 1. Machine Learning: An Algorithmic Perspective. 2nd Edition. Stephen Marsland

I just wanted to briefly share some initial impressions of the 2nd edition of Stephen Marsland's very hands on text, "Machine Learning, *An Algorithmic Perspective*." Having been a big fan of the first, I requested a review copy from Dr. Marsland, and with his help, the publishers were kind enough to send me a review copy. I spent the the last few months going over most of the newer topics and testing many of the newer scripts in Python. With that, I'll dive into some of my impressions of the text.

I've stated before, that I thought the 1st edition was hands down, one of the best texts covering applied Machine Learning from a Python perspective. I still consider this to be the case. The text, already extremely broad in scope, has been expanded to cover some very relevant modern topics, including:

- Particle Filtering (expanded coverage with working implementation in Python).
- Deep Belief Networks
- Gaussian Processes
- Support Vector Machines. Now includes working implementation with

Those topics alone should generate a significant amount of interest from readers. There are several things that separate this text's approach from many of the other texts covering Machine Learning. One, is that the text covers a very wide range of useful topics and algorithms. You rarely find a Machine Learning text with coverage in areas like evolutionary learning (genetic programming) or sampling methods (SIR, Metropolis-Hasting, etc). This is one reason I recommend the text highly to students of MOOC courses like Andrew Ng's excellent 'Machine Learning', or Hastie and Tibshirani's, 'An Introduction to Statistical learning'. Many of these students are looking to expand their set of skills in Machine Learning, with a desire to access working concrete code that they can build and run.

While the book does not overly focus on mathematical proofs and derivations, there is sufficient mathematical coverage that enables the student to follow along and understand the topics. Some knowledge of Linear Algebra and notation is always useful in any Machine Learning course. Also, the text is written in such a way, that if you simply want to cover a topic, such as particle filtering, you don't necessarily need to read all of the prior chapters to follow. This is useful for those readers looking to refresh their knowledge of more modern topics.

I did, occasionally, have to translate some of the Python code to work with Python 3.4. However, the editing was very minimal. For example, print statements in earlier versions did not require parentheses around the print arguments. So you can just change print 'this' to print('this'), for example.

I found the coverage of particle filters and sampling, highly relevant to financial time series-- as we have seen, such distributions often require models that depart from normality assumptions. I might possibly add a tutorial on this, sometime in the future.

In summary, I highly recommend this text to anyone that wants to learn Machine Learning, and finds the best way to augment learning is having access to working, concrete, code examples. In addition, I particularly recommend it to those students that have followed along from more of a Statistical Learning perspective (Ng, Hastie, Tibshirani) and are looking to broaden their knowledge of applications. The updated text is very timely, covering topics that are very popular right now and have little coverage in existing texts in this area.

*Anyone wishing to have a deeper look into topics covered and code, can find additional information and code on the [author's website](#).

monday, april 29, 2013

IBS Reversion (Stingray Plot) Intuition using R vioplot Package

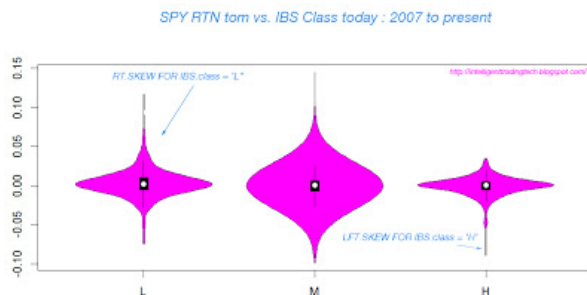



Fig 1. Vioplot of IBS filtered next day returns on SPY.

	Min	1st.Qu	Median	Mean	3rd.Qu	Max	Skewness
H	-0.0886	-0.0051	0.0004	-0.0008	0.0050	0.0349	-1.6000
L	-0.0743	-0.0049	0.0020830	0.0029	0.0102	0.11690	0.6394
M	-0.0985	-0.0064	0.0007	-0.0001	0.0067	0.14510	0.4364

Fig 2. Table of Summary of Results for rtn.tom vs. IBS.class (H,L,M)

A colleague and I were recently discussing ways to get intuition about the **IBS classification method** for reversion systems. I thought I'd share a **violin plot** I generated that might help to get some visual intuition about it. We can download and process next day returns for an asset like SPY and group classes into LOW (IBS < 0.2), HIGH (IBS > 0.8), and MID (all others). One thing you can see in the plots is the pronounced right skew in the LOW class, and left skew in the HIGH class (they sort of resemble opposing stingrays -- stingray plots might be a more apt term for the reversion phenomena); while the MID class tends to be more symmetrical. The nice thing about the vioplot visualization is that it includes the density shape of the return distribution, which adds intuition over more common box and whisker plots.

posted by intelligent trading at 10:24 pm 0 comments  links to this post
labels: ibs reversion (stingray plot) intuition using r vioplot package
ibs intuition using r vioplot package

[Home](#)

[Older Posts](#)

Subscribe to: [Posts \(Atom\)](#)