Machine Learning PS4:ML trading Algo

Hao Ran Li, Feiwen Liang, Leila Lan, Susu Zhu, Yitao Hu

22/04/2020

```
#import data and libraries
library(readr)
library(data.table)
library(foreign)
library(PerformanceAnalytics)
library(glmnet)
library(ggplot2)
StockRetAcct_DT= as.data.table(read.dta("StockRetAcct_insample.dta")) #set the key as firm ID and year
setkey(x = StockRetAcct_DT,FirmID,year)
StockRetAcct_DT=na.omit(StockRetAcct_DT)
```

Question 1

a

```
#create the new feasures
Features_df=StockRetAcct_DT[,list(
                                  lnIssue=lnIssue,
                                  lnIssue2=lnIssue^2,
                                  lnIssueME=lnIssue*lnME,
                                  lnProf=lnProf,
                                  lnProf2=lnProf^2,
                                  lnProfME=lnProf*lnME,
                                  lnInv=lnInv,
                                  lnInv2=lnInv^2,
                                  lnInv=lnInv*lnME,
                                  lnME=lnME,
                                  lnME2=lnME^2
                                  ),
                               by=list(FirmID,year)]

Returns=StockRetAcct_DT[,1:4,]
```

i

Note in order to use the formula written in 48 page, we need to first normalize all our features cross-sectionally (because we assume zero mean and 1 variance.), and then using the formula to compute feature-based long-short portfolio returns. The method is similar to using Fama-MacBeth Reregression for Portfolio construction after reviewing notes of Lecture 1. The mean return of Factor portfolios are shown below.

```r
#define a function to normalize the features
normalize=function(vec){
  return((vec-mean(vec)))
}
FactorRet=data.frame(matrix(NA,35,12))
i=1
#cross-sectionally normalize
for (t in min(Features_df$year):max(Features_df$year)){
  Xt=as.matrix(Features_df[Features_df$year==t,3:ncol(Features_df)])
  #get excess Return
  Rt=exp(Returns[year==t,,]$lnAnnRet)-exp(Returns[year==t,,]$lnRf)
  #normalize
  Xt=apply(Xt, 2, FUN=normalize)
  #insert 1s
  Xt=cbind(1,Xt)
  #compute factor returns
  FactorRet[i,]=t(Xt)%*%Rt/nrow(Xt)
  #FactorRet[i,]=solve(t(Xt)%*%Xt)%*%t(Xt)%*%Rt
  i=i+1
}
colnames(FactorRet)=c(colnames(Features_df[,3:ncol(Features_df),]),'EWMkt')
row.names(FactorRet)=1980:2014
colMeans(FactorRet)
```

```
##      lnIssue     lnIssue2    lnIssueME       lnProf      lnProf2     lnProfME
##  0.094134960 -0.004068936 -0.003294340 -0.056419662  0.004698409 -0.003181903
##        lnInv       lnInv2        lnInv         lnME        lnME2        EWMkt
##  0.060412797 -0.005955226 -0.007420974 -0.082455281 -0.007062238 -0.196818058
```

The Var-Cov matrix is shown below

```r
cov(FactorRet)
```

```
##                lnIssue      lnIssue2     lnIssueME        lnProf       lnProf2
## lnIssue    0.0347240252  3.645178e-04  3.096695e-04  0.0047560506 -6.699780e-05
## lnIssue2   0.0003645178  1.019909e-04  5.603283e-05  0.0014287212 -1.470876e-04
## lnIssueME  0.0003096695  5.603283e-05  4.281769e-05  0.0007802892 -6.980533e-05
## lnProf     0.0047560506  1.428721e-03  7.802892e-04  0.0200752652 -2.078163e-03
## lnProf2   -0.0000669978 -1.470876e-04 -6.980533e-05 -0.0020781630  3.111782e-04
## lnProfME   0.0001641671  2.022108e-04  9.618960e-05  0.0028629373 -4.334618e-04
## lnInv     -0.0020315024 -1.978640e-03 -9.407068e-04 -0.0279041868  4.151813e-03
## lnInv2     0.0004549639  1.035092e-04  5.980499e-05  0.0014522091 -1.428421e-04
## lnInv      0.0006973567  1.524596e-04  9.136096e-05  0.0021513230 -2.287884e-04
## lnME       0.0058897021  1.533814e-03  8.774739e-04  0.0216099856 -2.183989e-03
## lnME2     -0.0039851466  7.713502e-05  2.387457e-05  0.0013713655 -2.505919e-04
## EWMkt     -0.1048570506  2.328682e-03  8.147081e-04  0.0407491508 -7.642730e-03
##                lnProfME         lnInv        lnInv2         lnInv          lnME
## lnIssue    0.0001641671 -0.0020315024  4.549639e-04  6.973567e-04  0.0058897021
## lnIssue2   0.0002022108 -0.0019786397  1.035092e-04  1.524596e-04  0.0015338139
## lnIssueME  0.0000961896 -0.0009407068  5.980499e-05  9.136096e-05  0.0008774739
## lnProf     0.0028629373 -0.0279041868  1.452209e-03  2.151323e-03  0.0216099856
## lnProf2   -0.0004334618  0.0041518130 -1.428421e-04 -2.287884e-04 -0.0021839892
## lnProfME   0.0006389543 -0.0057736988  1.871172e-04  2.978334e-04  0.0028856688
## lnInv     -0.0057736988  0.0555568254 -1.924314e-03 -3.074780e-03 -0.0293040864
## lnInv2     0.0001871172 -0.0019243135  1.619222e-04  2.265714e-04  0.0023396880
```

```
## lnInv      0.0002978334 -0.0030747795  2.265714e-04  3.379761e-04  0.0033099463
## lnME       0.0028856688 -0.0293040864  2.339688e-03  3.309946e-03  0.0340282726
## lnME2      0.0004100152 -0.0027538948  7.420493e-05  1.521792e-04  0.0016772944
## EWMkt      0.0124118509 -0.0858934445  2.495149e-03  4.890535e-03  0.0529434142
##                    lnME2         EWMkt
## lnIssue   -3.985147e-03 -0.1048570506
## lnIssue2   7.713502e-05  0.0023286815
## lnIssueME  2.387457e-05  0.0008147081
## lnProf     1.371365e-03  0.0407491508
## lnProf2   -2.505919e-04 -0.0076427297
## lnProfME   4.100152e-04  0.0124118509
## lnInv     -2.753895e-03 -0.0858934445
## lnInv2     7.420493e-05  0.0024951485
## lnInv      1.521792e-04  0.0048905348
## lnME       1.677294e-03  0.0529434142
## lnME2      2.781341e-03  0.0768017021
## EWMkt      7.680170e-02  2.1317060415
```

The sample SR is shown below.

```
FactorRet=xts(FactorRet,order.by=seq(from=as.Date('1980-01-01'),to=as.Date('2014-01-01'),by='years'))

SharpeRatio(FactorRet,FUN = 'StdDev')
```

```
##                                  lnIssue   lnIssue2  lnIssueME     lnProf
## StdDev Sharpe (Rf=0%, p=95%): 0.5051681 -0.4029026 -0.5034505 -0.3981987
##                                 lnProf2   lnProfME     lnInv     lnInv2
## StdDev Sharpe (Rf=0%, p=95%): 0.266346 -0.1258786 0.2563069 -0.4679991
##                                    lnInv       lnME      lnME2      EWMkt
## StdDev Sharpe (Rf=0%, p=95%): 0.2563069 -0.4469907 -0.1339107 -0.1348035
```

ii

Here, we perform C.V to choose the optimal lambda.

```
#define a Cost function
Costfn=function(Fitted,Truths){
  return(mean((Fitted-Truths)^2))
}
TrainsetCV=FactorRet[1:25,]
Testset=FactorRet[26:35,]

#further divide the train data into training set and cv set
CV_coefs=c()

for (t in seq(1,21,by=5))
  {
  CVset=TrainsetCV[t:(t+4),]
  Trainset=TrainsetCV[-t:-(t+4),]
  Fbar_train=colMeans(Trainset)
  Fbar_cv=colMeans(CVset)
  X_train=cov(Trainset)
  X_cv=cov(CVset)
  Model=glmnet(x = X_train,y = Fbar_train,family = 'gaussian',alpha = 0.5)
  Fiited_cv=predict(Model,X_cv)
  Costs=apply(Fiited_cv, 2, FUN=Costfn,Truths=Fbar_cv)
```

```
  CV_coef=cbind(Model$lambda,Costs)
  CV_coefs=rbind(CV_coefs,CV_coef)
}
#compute mean Cost
colnames(CV_coefs)=c('Lambda','Cost')
CV_coefs=data.frame(CV_coefs)
CV_coefs=aggregate(CV_coefs,by=list(CV_coefs$Lambda),FUN=mean)
#find opt labmda
opt_lambda=CV_coefs[CV_coefs$Cost==min(CV_coefs$Cost),]$Lambda
```

Then we use optimal lambda to re-estimate the Model, the betas are shown below

```
OptModel=glmnet(x=cov(TrainsetCV),y = colMeans(TrainsetCV),family = 'gaussian',lambda = opt_lambda,inte
bs=OptModel$beta
bs
```

```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                       s0
## lnIssue     1.019056901
## lnIssue2    .
## lnIssueME -64.984226350
## lnProf      .
## lnProf2     .
## lnProfME    .
## lnInv       .
## lnInv2     -2.514362501
## lnInv       .
## lnME        .
## lnME2       .
## EWMkt       0.001842944
```

iii

Finally we can compute the MVE Portfolio returns, which is given by:

$$R_t^{Port} = b^T F_t$$

```
MVE_Ret=t(bs)%*%t(as.matrix(Testset))
```

Then, we can compute the out-of-sample Performance statistics

```
out_sample_test=list(Mean=mean(MVE_Ret),SD=sd(MVE_Ret),SR=mean(MVE_Ret)/sd(MVE_Ret))
out_sample_test
```

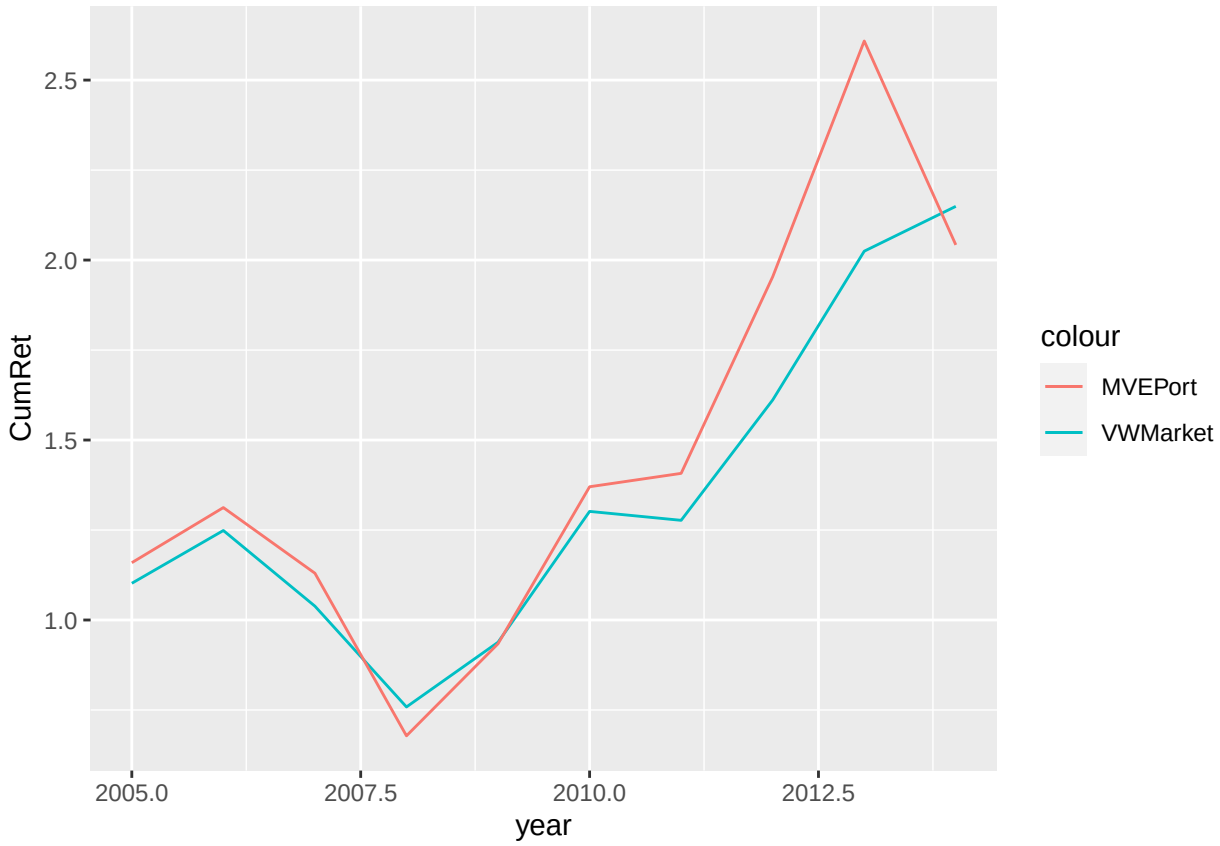
```
## $Mean
## [1] 0.1723909
##
## $SD
## [1] 0.2107025
##
## $SR
## [1] 0.8181722
```

iv

Note, to be realistic, we real-estimate our model each year and compute the MVE Portfolio in accorance.

```
#define a function to map from trainCVset to coeffs
getbs=function(TrainsetCV)
  {
#further divide the train data into training set and cv set
CV_coefs=c()

for (t in seq(1,(nrow(TrainsetCV)-4),by=round(nrow(TrainsetCV)/5)))
  {
  CVset=TrainsetCV[t:(t+4),]
  Trainset=TrainsetCV[-t:-(t+4),]
  Fbar_train=colMeans(Trainset)
  Fbar_cv=colMeans(CVset)
  X_train=cov(Trainset)
  X_cv=cov(CVset)
  Model=glmnet(x = X_train,y = Fbar_train,family = 'gaussian',alpha = 0.5)
  Fiited_cv=predict(Model,X_cv)
  Costs=apply(Fiited_cv, 2, FUN=Costfn,Truths=Fbar_cv)
  CV_coef=cbind(Model$lambda,Costs)
  CV_coefs=rbind(CV_coefs,CV_coef)
}
#compute mean Cost
colnames(CV_coefs)=c('Lambda','Cost')
CV_coefs=data.frame(CV_coefs)
CV_coefs=aggregate(CV_coefs,by=list(CV_coefs$Lambda),FUN=mean)
#find opt labmda
opt_lambda=CV_coefs[CV_coefs$Cost==min(CV_coefs$Cost),]$Lambda
OptModel=glmnet(x=cov(TrainsetCV),y = colMeans(TrainsetCV),family = 'gaussian',lambda = opt_lambda,inte
bs=OptModel$beta
return(bs)
}
MVE_Port_Ret=c()
#loop through time
for (t in 1:10){
  bs=getbs(TrainsetCV = FactorRet[1:(t+24),])
  Ret=as.numeric(t(bs)%*%t(as.matrix(Testset[t,])))
  MVE_Port_Ret=c(MVE_Port_Ret,Ret)
}
MVE_Port=cumprod(1+MVE_Port_Ret)
#compute VW port Ret
StockRetAcct_DT[,EXRet:=exp(lnAnnRet)-exp(lnRf),]
VWRET=StockRetAcct_DT[,list(VWRet=weighted.mean(x = EXRet,w = lnME)),by=year]
VWRET=VWRET[order(year)]
VWRET=VWRET[year>2004,]
VWRET[,CumRet:=cumprod(VWRet+1),]
#set the leverage factor
k=sd(VWRET$VWRet)/sd(MVE_Port_Ret)
MVE_Port_Ret=k*MVE_Port_Ret
#plot the returns
plt=ggplot(VWRET,aes(x = year,y = CumRet,color='VWMarket'))
plt+geom_line()+geom_line(aes(y=MVE_Port,color='MVEPort'))
```


2.5
2.0
1.5
1.0
CumRet
2005.0
2007.5
2010.0
2012.5
year
colour
MVEPort
VWMarket