دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)
دانشکده مهندسی برق

Statistical learning

Assignment 3

Mohammad hasan shammakhi

محمد حسن شماخی

۹۳۱۲۳۰۵۳

باعرض سلام خدمت شما:

خیلی خوشحالم از یک سو بخاطر اینکه این Assignment موجب شده تا به قلق های زیادی از نحوه بکارگیری الگوریتم های مختلف مشرف بشم و از طرفی با R بیشتر مانوس بشم. هرچند که از یک سو زمان زیادی از وقت لازم برای امتحان های پایان ترم را از دست دادم. ولی بازم می ارزید.

در طی این برنامه که شاید پیچیدگی آن یک صدم برنامه قبلیم نیز نباشد به خطای ۴ ٪ رسیدم. که علاوه بر خطای ۴ درصد به نکات دیگری هم پی بردم که به نظرم ارزش گفتن دارند.

اول اینکه تفاوت های این برنامه را با برنامه قبلی میگم و چی شد که یهو انقدر خوب جواب گرفتم.

یک مطلبی سر کلاس بیان کرده بودید تحت این عنوان که چرا نباید مسئله classification با بیش از ۲ کلاس را اینطور حل کرد:

- Encode three type classifier with a variable with three states:

$$Y = \begin{cases} 0 & \text{residential} \\ 1 & \text{agricultural} \\ 2 & \text{Lake/water} \end{cases}$$

- This coding is not appropriate as it suggests an ordering (i.e., classification error of residential to agricultural is half of residential to lake/water)
  - Fitting a linear regression model is not an option
- Multiclass Logistic Regression or Discriminant Analysis are more appropriate.

همانطور که اینجا بیان شده مشکل این بوده که مثلا خطا بین lake و residential با lake با agricultural متفاوت است.

خوب منم گفتم حل این مشکل که کاری نداره میایم کلاسا رو حدس می زنیم بعدش فقط تعداد اونایی که غلط گفته شده رو می شماریم و تقسیم بر تعداد می کنیم و اونوقت حل می شه دیگه و این مشکل رو نداره.

اما کلا مشکل دیگه هم داره و اینکه کلا تخمین کلاسبندی به وسیله مدل های رگرسیون خیلی بد پیش بین میشه جدا از خطاشون.

اینو دیروز توی یک مقاله دیدم که هم یک روش رگرسیون ارائه داده بود هم یک روش کلاسبندی. وگرنه همچین حسی نداشتم نسبت بهش

برای همین اومدم از روش one vs all استفاده کردم و بهترین احتمال رو در نظر گرفتم.

با این کار به یکباره خطا خیلی خوب شد و مثلا svm از ۱۰,۲۵ به ۴,۲ کاهش پیدا کرد همچنین خطا در روش RVM از ۱۱,۲ به ۳,۵۲ تغییر کرد.

این کار رو روی LDA و GLM هم کردم و خطای خوبی دریافت شد. که نتیجه ترکیب LDA و SVM با هم به ۴,۱ رسید.

روش بکار گرفته شده به مرحله به شرح زیر است:

MSPRL1

Tue Jun 02 23:28:48 2015

پس از پاکسازی workspace شروع به load کردن دیتا می کنیم.

```
rm(list=ls())
x_train=read.table("C:/Users/MSPRL1/Desktop/A3/UCI HAR Dataset/train/X_tra
in.txt", quote="\"")
y_train=read.table("C:/Users/MSPRL1/Desktop/A3/UCI HAR Dataset/train/y_tra
in.txt", quote="\"")
x_test=read.table("C:/Users/MSPRL1/Desktop/A3/UCI HAR Dataset/test/X_test.
txt", quote="\"")
y_test=read.table("C:/Users/MSPRL1/Desktop/A3/UCI HAR Dataset/test/y_test.
txt", quote="\"")
features=read.table("C:/Users/MSPRL1/Desktop/A3/UCI HAR Dataset/features.t
xt", quote="\"")
name=t(features[,2])
```

اسامی ستون های دیتا را درست کرده تا کار با دیتا راحت تر شود. در این مرحله نام label ها را y گذاشتیم.

```
colnames(x_train)=name;colnames(x_test)=name
colnames(y_train)=c("y");colnames(y_test)=c("y")
rm(name)
dat.train=data.frame(y=y_train,x_train)
dat.test=data.frame(y=y_test,x_test)
x.tra=model.matrix(y~.,data=dat.train)[,-1]
y.tra=dat.train$y
x.tes=model.matrix(y~.,data=dat.test)[,-1]
y.tes=dat.test$y
```

حال می خواهیم قسمتی از داده ها را با عنوان داده های پرت (outlier) دور بریزیم.

که معیار ما برای اینکار lofactor یا همان فاصله محلی است.

```
#####################outlier
library(DMwR)

## Warning: package 'DMwR' was built under R version 3.2.0

## Loading required package: lattice
## Loading required package: grid

t1=proc.time()
outlier.scores = lofactor(x.tra[,-1], k=5)
proc.time()-t1

##    user  system elapsed
## 1307.28  216.33 1528.77
```

در اینجا می توان دید که زمان لازم برای انجام Outlier Detection ۱۳۰۷ ثانیه معادل ۲۱ دقیقه و ۴۷ ثانیه می باشد.

حال دیتا های با ضریب بیشتر از ۱,۱ را دور می ریزیم و بقیه را با نام ndat. تولید می کنیم.

```
nx.tra=x.tra[outlier.scores<1.1,]
ny.tra=y.tra[outlier.scores<1.1]
ndat.train=data.frame(y=ny.tra,nx.tra)
ndat.test=data.frame(y=y.tes,x.tes)
```

حال می خواهیم ابتدا از لاسو استفاده کنیم تا برخی ویژگی ها را دور برزیم تا هم از redundancy جلوگیری کنیم هم اینکه در بکارگیری الگوریتم های مختلف بر داده ها زمان کمتری صرف کنیم.

نحوه استفاده کردنمان از لاسو به این صورت است که ابتدا بهترین لاندا را از روی لاسو با معیار cross validation پیدا می کنیم سپس تابع اصلی لاسو که میزان پارامترهای بیشتری در اختیار ما می گذارد را به ازای بهترین لاندا با رو ش CV تخمین می زنیم و پس از مشخص کردن ضرایب غیر صفر دیتای جدید را می سازیم.

```
###############laso
library (glmnet)
cv.lasso.fit=cv.glmnet(nx.tra,ny.tra,alpha=1)
best.landa=cv.lasso.fit$lambda.min
lasso.fit=glmnet(nx.tra,ny.tra,alpha=1,lambda=best.landa)
ind=lasso.fit$beta@i
nnx.tra=nx.tra[,ind]
nnx.tes=x.tes[,ind]
ndat.train=data.frame(y=ny.tra,nnx.tra)
ndat.test=data.frame(y=y.tes,nnx.tes)
```

حال می خواهیم داده های جدیدی به منظور استفاده از مدل ها به روش one vs all تولید کنیم.

```
######################one vs all
y1=ifelse(ndat.train$y==1,1,0);dat1=data.frame(y=y1,ndat.train[,-1])
y1=ifelse(ndat.train$y==2,1,0);dat2=data.frame(y=y1,ndat.train[,-1])
y1=ifelse(ndat.train$y==3,1,0);dat3=data.frame(y=y1,ndat.train[,-1])
y1=ifelse(ndat.train$y==4,1,0);dat4=data.frame(y=y1,ndat.train[,-1])
y1=ifelse(ndat.train$y==5,1,0);dat5=data.frame(y=y1,ndat.train[,-1])
y1=ifelse(ndat.train$y==6,1,0);dat6=data.frame(y=y1,ndat.train[,-1])
rm(y1)
```

حال می خواهیم ببینیم بر این داده ها SVM چقدر خوب است.

```
###############SVM
library(e1071)

## Warning: package 'e1071' was built under R version 3.2.0

t1=proc.time()
svm1=svm(y~.,data=dat1,family=binomial)
svm2=svm(y~.,data=dat2,family=binomial)
svm3=svm(y~.,data=dat3,family=binomial)
svm4=svm(y~.,data=dat4,family=binomial)
svm5=svm(y~.,data=dat5,family=binomial)
svm6=svm(y~.,data=dat6,family=binomial)
svm.pred1=predict(svm1,ndat.test,type="response")
svm.pred2=predict(svm2,ndat.test,type="response")
svm.pred3=predict(svm3,ndat.test,type="response")
svm.pred4=predict(svm4,ndat.test,type="response")
svm.pred5=predict(svm5,ndat.test,type="response")
svm.pred6=predict(svm6,ndat.test,type="response")
proc.time()-t1

##    user  system elapsed
##  243.60    1.08  245.92
```

زمان لازم برای بدست آوردن پارامترهای SVM برابر ۴ دقیقه و ۳ ثانیه می باشد.

```
svm.mat=data.frame(svm.pred1,svm.pred2,svm.pred3,svm.pred4,svm.pred5,svm.p
red6)
svm.class=max.col(svm.mat)
e_svm=mean(svm.class!=y.tes)
table(svm.class,y.tes)

##          y.tes
## svm.class   1   2   3   4   5   6
##         1 488   3   4   0   0   0
##         2   2 466  20   2   0   0
##         3   6   2 396   0   0   4
##         4   0   0   0 439  32   0
##         5   0   0   0  47 500   9
##         6   0   0   0   3   0 524

e_svm

## [1] 0.04546997
```

که خطای حاصل از SVM برابر ۴,۵ درصد می باشد. یعنی از ۱۰,۶ حالت قبلی ۶,۱ ٪ کاهش خطا داشتیم.

```
svm.out=data.frame(svm.class,apply(svm.mat, 1,max)/max(apply(svm.mat, 1,ma
x)))
```

حال می خواهیم از یک روش بر مبنای بیزین استفاده کنیم که در اینجا از RVM استفاده کرده ایم.

```
######################rvm
library("kernlab")
rvm1=rvm(y~.,data=dat1,family=binomial)
t2= proc.time ()
t2-t1
   user  system elapsed
4545.47   16.39 4565.19
t2= proc.time ()
t2-t1
    user   system  elapsed
 4545.53    16.39 31279.80
rvm2=rvm(y~.,data=dat1,family=binomial)
t3= proc.time ()
t3-t2
   user  system elapsed
4657.58   17.68 4678.61
rvm3=rvm(y~.,data=dat1,family=binomial)
t4= proc.time ()
t4-t3
   user  system elapsed
4522.75   16.72 4541.97
t4= proc.time ()
t4-t3
   user  system elapsed
4522.77   16.72 4744.88
rvm4=rvm(y~.,data=dat1,family=binomial)
t5= proc.time ()
t5-t4
   user  system elapsed
3743.55   13.51 3760.81
rvm5=rvm(y~.,data=dat1,family=binomial)
t6=proc.time()
t6-t5
   user  system elapsed
3696.02   12.61 3711.72
rvm6=rvm(y~.,data=dat1,family=binomial)
```

زمان طی شده برای انجام هر یک از RVM ها روی هم ۲۴۸۵۹ ثانیه معادل ۷ ساعت است

```
rvm.pred1=predict(rvm1,ndat.test,type="response")
rvm.pred2=predict(rvm2,ndat.test,type="response")
rvm.pred3=predict(rvm3,ndat.test,type="response")
rvm.pred4=predict(rvm4,ndat.test,type="response")
rvm.pred5=predict(rvm5,ndat.test,type="response")
rvm.pred6=predict(rvm6,ndat.test,type="response")

rvm.mat=data.frame(rvm.pred1,rvm.pred2,rvm.pred3,rvm.pred4,rvm.pred5,rvm.p
red6)
rvm.class=max.col(rvm.mat)
e_rvm=mean(rvm.class!=y.tes)
```

```
table (rvm.class,y.tes)
          y.tes
rvm.class   1   2   3   4   5   6
        1 490  12   1   0   0   0
        2   2 458  10   3   0   0
        3   4   1 409   0   0   0
        4   0   0   0 435  14   0
        5   0   0   0  52 518   4
        6   0   0   0   1   0 533
```

```
e_rvm
```

```
## [1] 0.03529013
```

خطای بدست آمده در این روش ۳,۵۲ درصد می باشد.

حال می خواهیم از روش LDA استفاده نماییم.

```
##############################lda
library(MASS)
t1=proc.time()
lda1=lda(y~.,data=dat1,family=binomial)
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear
```

```
lda2=lda(y~.,data=dat2,family=binomial)
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear
```

```
lda3=lda(y~.,data=dat3,family=binomial)
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear
```

```
lda4=lda(y~.,data=dat4,family=binomial)
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear
```

```
lda5=lda(y~.,data=dat5,family=binomial)
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear
```

```
lda6=lda(y~.,data=dat6,family=binomial)
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear
```

```
lda.pred1=predict(lda1,ndat.test,type="response")
lda.pred2=predict(lda2,ndat.test,type="response")
lda.pred3=predict(lda3,ndat.test,type="response")
lda.pred4=predict(lda4,ndat.test,type="response")
lda.pred5=predict(lda5,ndat.test,type="response")
lda.pred6=predict(lda6,ndat.test,type="response")
proc.time()-t1
```

```
##    user  system elapsed
##   32.26    0.64   33.01
```

```
lda.mat=data.frame(lda.pred1$posterior[,2],lda.pred2$posterior[,2],lda.pre
d3$posterior[,2],lda.pred4$posterior[,2],lda.pred5$posterior[,2],lda.pred6
$posterior[,2])
```

```
lda.class=max.col(lda.mat)
e_lda=mean(lda.class!=y.tes)
table(lda.class,y.tes)
```

```
##          y.tes
## lda.class   1   2   3   4   5   6
##         1 489  12   2   0   0   0
##         2   7 455  10   3   0   2
##         3   0   0 406   0   0   0
##         4   0   4   2 432  23   0
##         5   0   0   0  56 509   9
##         6   0   0   0   0   0 526
```

```
e_lda
```

```
## [1] 0.04411266
```

همانطور که مشاهده می شود خطا به میزان ۴,۴ ٪ می باشد.

```
lda.out=data.frame(lda.class,apply(lda.mat, 1,max))
```

حال خطای کلی را با استفاده از مدلی جدید بررسی میکنیم که به این صورت است:

احتمال ناشی از تخمین هر داده تست را در هر روش اندازه می گیریم و سپس lable را در نظر میگیریم که یک الگوریتم با اطمینان بیشتری نسبت به بقیه آن را مطرح کرده باشند.

```
####################total
total.class=ifelse(lda.out[,2]>svm.out[,2],lda.out[,1],svm.out[,1])
e_total=mean(total.class!=y.tes)
e_total
```

```
## [1] 0.04309467
```

```
table(total.class,y.tes)
```

```
##            y.tes
## total.class   1   2   3   4   5   6
##           1 491  11   2   0   0   0
##           2   3 457  14   3   0   2
##           3   2   1 404   0   0   0
##           4   0   2   0 433  23   0
##           5   0   0   0  55 509   9
##           6   0   0   0   0   0 526
```

```
proc.time()-t1
```

```
##    user  system elapsed
##   32.31    0.65   33.07
```

بنابراین خطای نهایی برابر ۳,۵۲می باشد.

```
##############################glm
library(MASS)
glm1=glm(y~.,data=dat1,family=binomial)
glm2=glm(y~.,data=dat2,family=binomial)
glm3=glm(y~.,data=dat3,family=binomial)
glm4=glm(y~.,data=dat4,family=binomial)
glm5=glm(y~.,data=dat5,family=binomial)
glm6=glm(y~.,data=dat6,family=binomial)
glm.pred1=predict(glm1,ndat.test,type="response")
glm.pred2=predict(glm2,ndat.test,type="response")
glm.pred3=predict(glm3,ndat.test,type="response")
glm.pred4=predict(glm4,ndat.test,type="response")
glm.pred5=predict(glm5,ndat.test,type="response")
glm.pred6=predict(glm6,ndat.test,type="response")
glm.mat=data.frame(glm.pred1,glm.pred2,glm.pred3,glm.pred4,glm.pred5,glm.p
red6)
glm.class=max.col(glm.mat)
e_glm=mean(glm.class!=y.tes)
table(glm.class,y.tes)
```

```
##           y.tes
## glm.class   1   2   3   4   5   6
##         1 473  14   2   0   0   0
##         2   5 430   9   3   0   9
##         3   0   3 388   1   1   0
##         4  14  14   3 413  21   3
##         5   4  10  18  74 509  14
##         6   0   0   0   0   1 511
```

```
e_glm
```

```
## [1] 0.07567017
```

خطای ناشی از روش logistic regression **برابر با ۷،۵ درصد می باشد.**

```
glm.out=data.frame(glm.class,apply(glm.mat, 1,max))
```

متاسفانه وقت بیشتری جهت ارائه کارهایی که در فایل قبلی انجام دادم جهت کاهش خطا ندارم و به همین خطای

چون ۶ بار RVM **زده ام لذا حجم آن حدود ۱۰۰ مگابایت شده که ارسال آن و از آن مهمترین برای شما دردسر**

میشه

بنابراین در برنامه ای که برای استفاده شما برای داده های تستتان قرار دادم RVM **را قرار نداده ام**

با تشکر

محمد حسن شاخی