多元统计分析课程论文

匡亚明学院 统计学 马宇恒 171240510

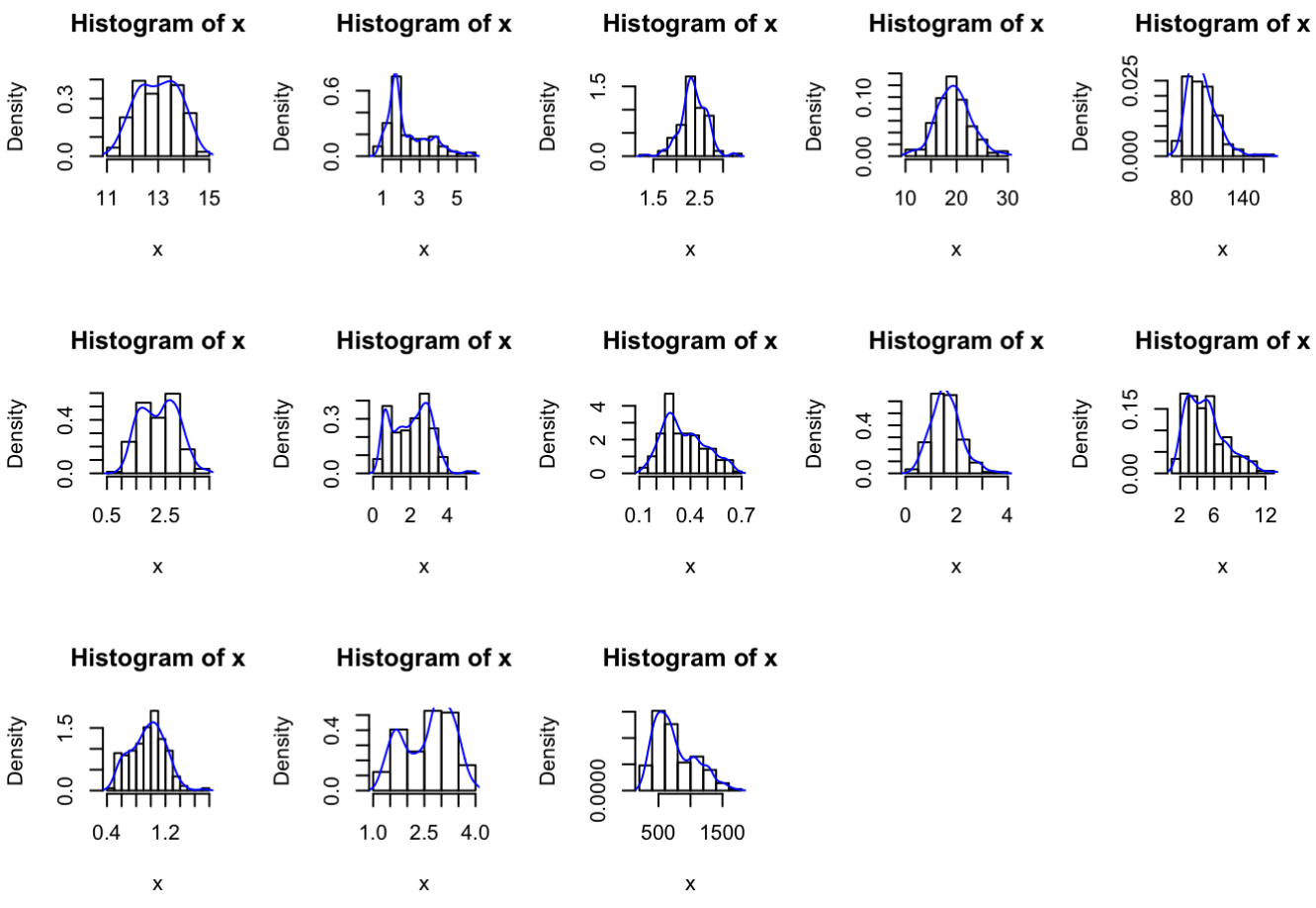
引言

报告采用多元统计分析课程中所包含的多种分类器对红酒数据进行分类。该数据集是1991年由Institute of Pharmaceutical and Food Analysis and Technologies, Genoa, Italy的M. Forina等人提供，是机器学习领域经典的分类任务数据集。数据集记录了来自意大利同一地区、由相同工艺和三个品种葡萄制作的红酒的十三种化学成分含量，所有的指标均为数值变量。

数据集整理与探索

```
wine<-read.csv("./wine/winedata",header = FALSE)
colnames(wine)<-c("Type","Alcohol","Malicacid","Ash","Alcalinityofash","Magnesium","T
otalphenols","Flavanoids","Nonflavanoidphenols","Proanthocyanins","Colorintensity","H
ue","OD280.OD315diluted","Proline")
```

```
par(mfrow=c(3,5))
for (i in 2:14){
  x=wine[[i]]
hist(x,probability=T)
lines(density(x, bw = "sj"),col="blue")
}
```



给出每个变量的分布直方图，接下来对每个类别进行正态性检验。

```
multinormtest<-function(D,i){
  U<-D[D[1]==i,][2:14]
  mshapiro.test(t(U))
}
```

```
multinormtest(wine,1)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  Z
## W = 0.82894, p-value = 9.091e-07
```

第一种红酒数据是正态分布，第二、三种结果类似，所以我们大致认为数据有正态性，且生成模型为混合高斯。接下来检验每个总体之间是否有显著性差异。首先我们需要验证方差是否相等，所以做如下假设检验。

```
multi.var.test=function(data){
  n=nrow(data)
  p=ncol(data)-1
  lambda=(det((n-1)*var(data[2:14])/n))^(-n/2)
  d=0
  k=3
  for(i in 1:k){
    datatemp=data[data$Type==i,2:14]
    tempn=nrow(datatemp)
    d=d+1/(tempn-1)
    lambda=lambda/(det((tempn-1)*var(datatemp)/tempn))^(-tempn/2)
  }

  d=(2*p^2+3*p-1)*(d-1/(n-k))/(6*(p+1)*(k-1))
  Chi=-2*(1-d)*log(lambda)
  p.value=1-pchisq(Chi, 0.5*p*(p+1)*(k-1))
  return(p.value=p.value)

}
multi.var.test(wine)
```

```
## [1] 0
```

p值输出为0，通过在console中读取参数值可知检验统计量已经达到1325，远超自由度为182的Chi平方分布的置信区间。所以三组协方差矩阵显著的不相等。所以我们采取协方差矩阵不相等时的检验方法。

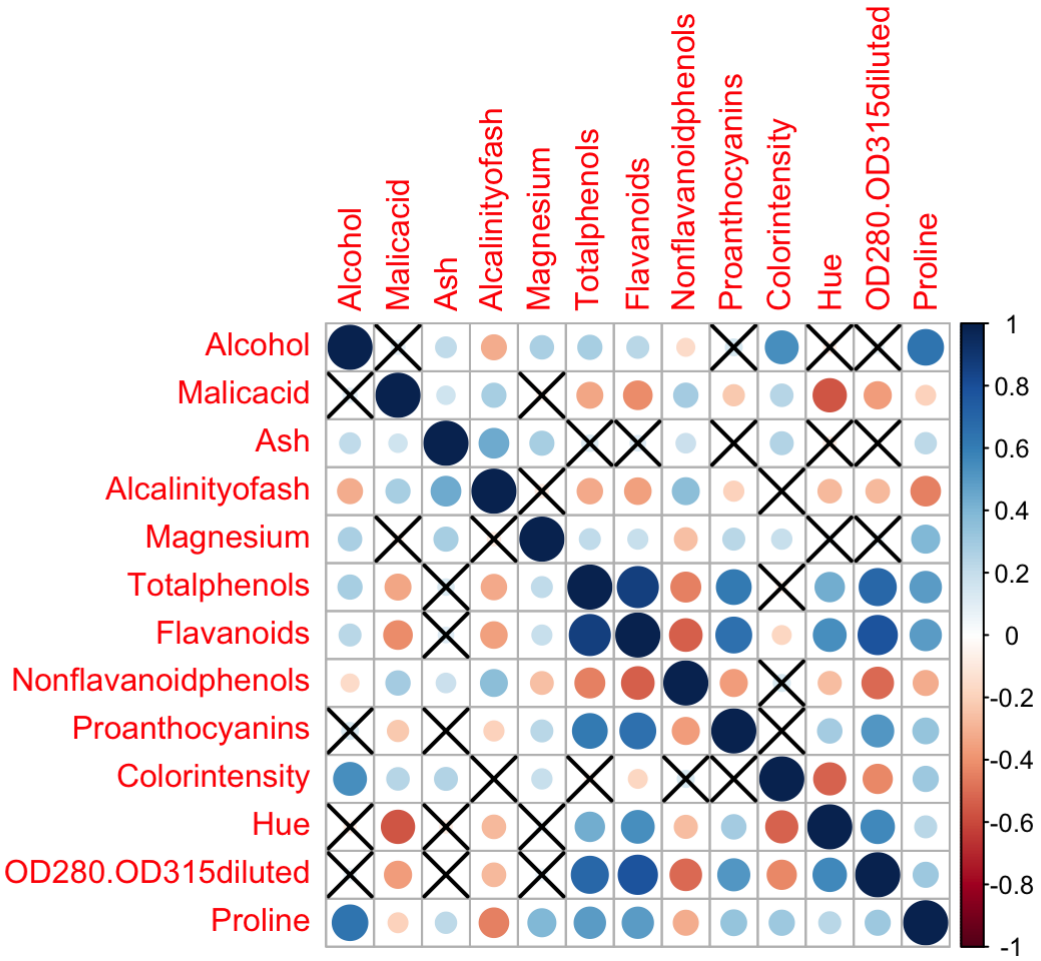
```
multi.difmu.test<-function(data,k,h){
  p=ncol(data)-1
  if (ncol(data[data$Type==k,])<ncol(data[data$Type==h,])){
    datan<-data[data$Type==k,2:14]
    datam<-data[data$Type==h,2:14]
    n=ncol(data[data$Type==k,])
    m=ncol(data[data$Type==h,])
  }
  else{
    datan<-data[data$Type==h,2:14]
    datam<-data[data$Type==k,2:14]
    n=ncol(data[data$Type==h,])
    m=ncol(data[data$Type==k,])
  }
  dataz<-datan
  for (i in 1:n){
    dataz[i,]=datan[i,]-sqrt(n/m)*datam[i,]+sqrt(1/m*n)*apply(datam[1:n,],2, mean)-ap
ply(datam,2,mean)/m
  }
  mu<-as.matrix(apply(dataz,2,mean))
  sigmainv=as.matrix(solve(var(dataz)))
  Fstatistics=sum(diag((n-p)*n*t(mu)%*%sigmainv%*%mu/((n-1)*p)))
  p.value=1-pf(Fstatistics, p,n-p)
  return(p.value=p.value)
}
multi.difmu.test(wine,1,2)
```

```
## [1] 0.09944742
```

第二、三组数据与第一组结果类似。虽然不愿意承认，但是我们的三类数据都没有显著的区别于彼此。不过我们并不能以此接受原假设。事实上，当我们使用不同的分类器进行分类时，取得的效果相当理想。这也是众多论文采用本数据集进行算法初步检验的原因。课本中提到，当协方差矩阵差异巨大时，这种两样本检验并不准确，这更加坚定了我们继续下去的决心。

除了总体的区分度外，我们还要查看因变量之间的相关性。

```
res1 <- cor.mtest(wine[c(2:14)], conf.level = 0.95)
corrplot(cor(wine[2:14]),sig.level = .05,p.mat = res1$p)
```



可以看到不同变量之间的相关性比较显著。这给了我们采取PCA进行分类的提示。

训练集拆分

我们按照7:3的比例随机拆分训练集和测试集合。

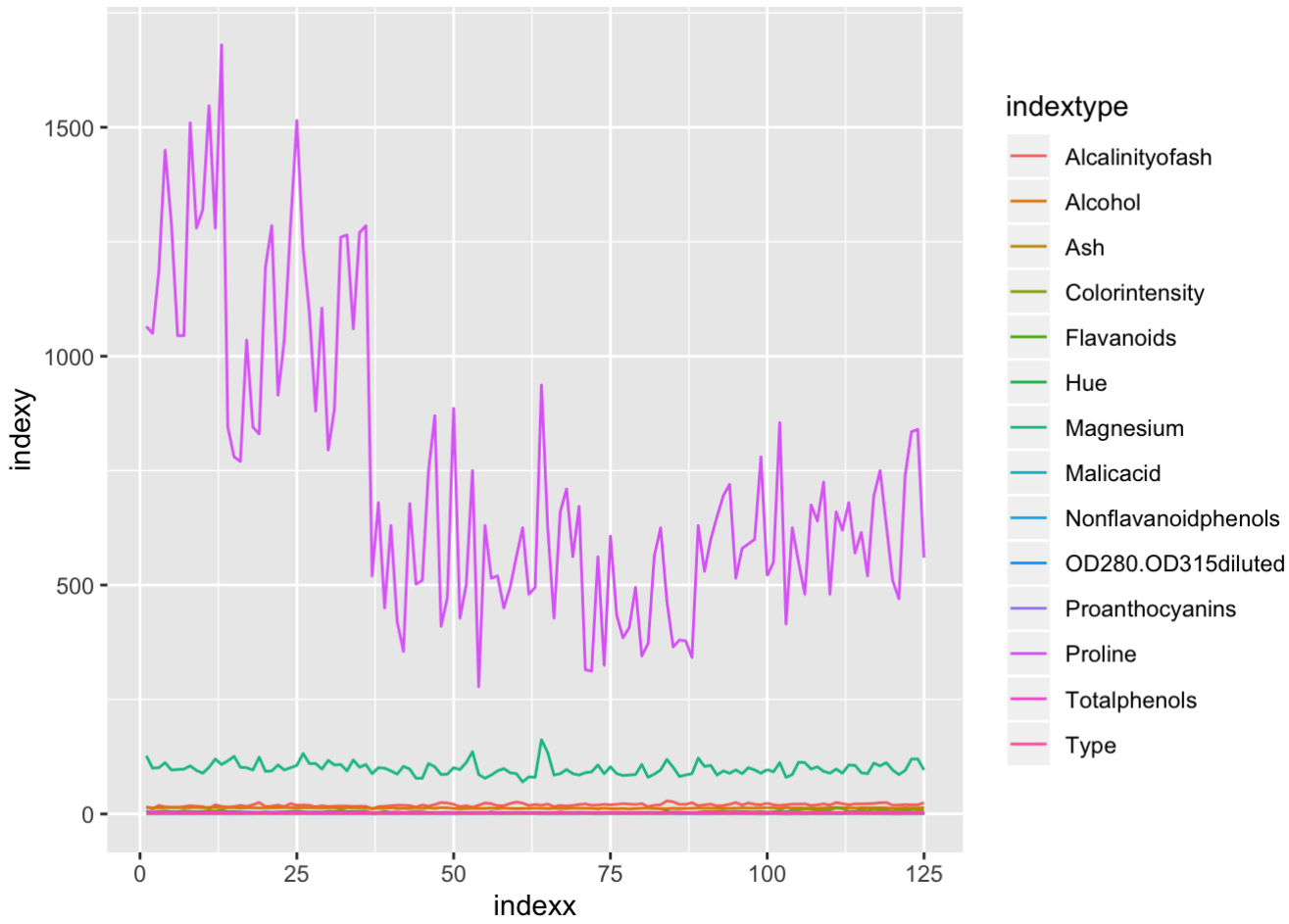
```
set.seed(521)
index<-sample(1:nrow(wine),round(nrow(wine)*3/10))
wine_test<-wine[index,]
wine_train<-wine[-index,]
wine_test
wine_train
```

数据中心化与标准化

拆分数据集之后，我们可以选择直接使用数据训练模型，但如果我们对数据的量级进行绘图：

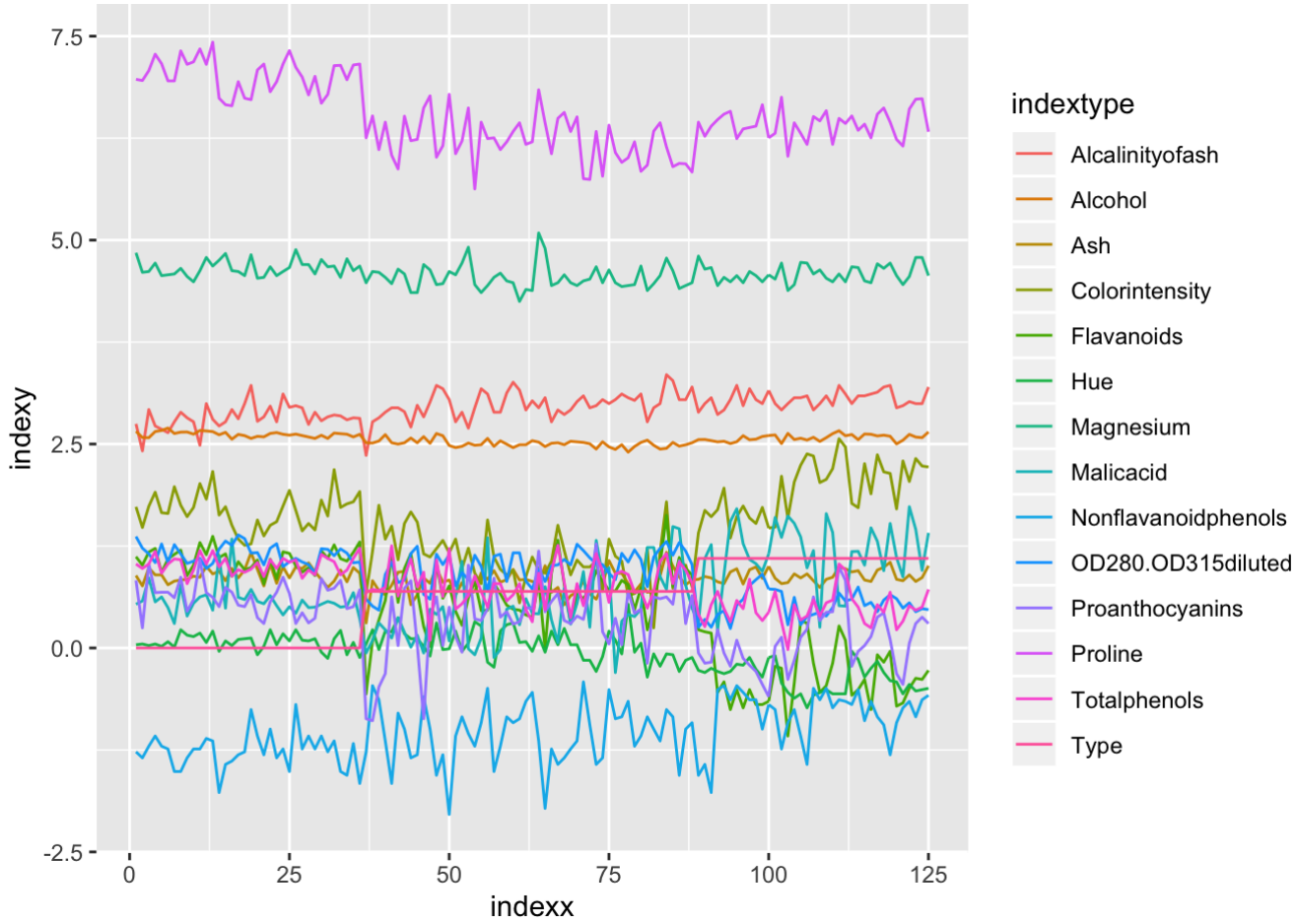
```
scaleplot<-function(data,name,iflog=0){
  data<-data[c(1,which(colnames(data)==name))]
  indexx=NULL
  indexy=NULL
  indextype=NULL
  for (i in 1:length(name)){
    indexx=c(indexx,1:nrow(data))
    indexy=c(indexy,data[[name[i]]])
    indextype=c(indextype,rep(name[i],nrow(data)))
  }
  if(iflog==1){
    indexy=log(indexy)
  }
  plotdata<-data.frame(indexx,indexy,indextype)
  p<-ggplot(plotdata,aes(x=indexx,y=indexy,group=indextype,colour=indextype))+geom_li
ne()
  p
}
```

```
scaleplot(wine_train,colnames(wine_train))
```



这张图由于一个量级很大的数据压缩了大部分变量的显示效果，我们取对数后显示。

```
scaleplot(wine_train,colnames(wine_train),iflog=1)
```



可以看到数据量级的差异非常巨大，这将会极大程度上使得PCA等模型受到量级大的变量的影响，使得模型不稳定。另外，许多模型的可解释性也会随着中心化与标准化有质的改变。因此，我们对数据进行中心化与标准化。

```
sigma=1
mean=0
for ( i in 2:ncol(wine_train)){
  sigma=c(sigma,sqrt(var(wine_train[[i]])))
  mean=c(mean,mean(wine_train[[i]]))
}
wine_trainst<-data.frame(wine_train[[1]],as.data.frame(scale(wine_train[2:14])))
colnames(wine_trainst)[1]="Type"
```

注意到我们保存了均值和方差（标签列设为0和1），是因为中心化和标准化也是模型的一部分，需要对测试集使用相同的中心化、标准化方法。

总函数

本文之后的各种模型均可以通过此函数进行调用和测试。

```
#输出判别结果的函数
printdiscriminant<-function(original,predicted,ifprint=1){
  count<-matrix(rep(0,9),3,3)
  original<-as.numeric(original)
  predicted<-as.numeric(predicted)
  for (i in 1:length(original)){
    count[original[i],predicted[i]]=1+count[original[i],predicted[i]]
  }
  accuracy=sum(diag(count))/length(original)
  count<-as.data.frame(count)
  rownames(count)=c("original1","original2","original3")
  colnames(count)=c("predicted1","predicted2","predicted3")
  if(ifprint==0){  return(accuracy)}
  print(count)
  print("accuracy:")
  print(accuracy)

}
# 线性判别的判别函数，后文相应处有解释
decidegroup<-function(x){
  if(x[1]< -1.891911 && x[2]>-0.354111){
    return(1)
  }else if(x[1]>1.794088 && x[2]>-0.6310445){
    return(3)
  }else{
    return(2)
  }
}
Mdistance<-function(x,group){
  sigmainv=solve(var(group[-1]))
  mu=sapply(group[-1],mean)
  result=(as.matrix(x[-1]-mu))%*%sigmainv%*%t(as.matrix(x[-1]-mu))
  result[1,1]
}
GenearalMdistance<-function(x,group){
  group1<-group[group$Type==1,]
  group2<-group[group$Type==2,]
  group3<-group[group$Type==3,]
  sigma<-c(log(det(var(group1[-1]))),log(det(var(group2[-1]))),log(det(var(group3[-1
]))))
  prior<-c(-2*log(nrow(group1)/nrow(group)),-2*log(nrow(group2)/nrow(group)),-2*log(n
row(group3)/nrow(group)))
  distance<-c(Mdistance(x,group1),Mdistance(x,group2),Mdistance(x,group3))+sigma+prio
r
  distance
}
#trainingdata为原始训练集（未标准化中心化）
#testingdata为原始测试集
#参数先后为是否标准化中心化，是否做pca，使用的分类器，是否计时
discriminant<-function(trainingdata,testingdata,ifstandarize=0,ifpca=0,method,ifprint
=1){
  #标准化
  if(ifstandarize==1){
    sigma=1
    mean=0
    for ( i in 2:ncol(trainingdata)){
      sigma=c(sigma,sqrt(var(trainingdata[[i]])))
```

```
      mean=c(mean,mean(trainingdata[[i]]))
    }
    trainingdata<-data.frame(trainingdata[[1]],as.data.frame(scale(trainingdata[-1
])))
    colnames(trainingdata)[1]="Type"
    for(i in 1:ncol(testingdata)){
      testingdata[[i]]<-(testingdata[[i]]-mean[i])/sigma[i]
    }
  }
  #对pca数据进行处理
  if(ifpca==1){
    trainingdata.pca<-prcomp(trainingdata[-1])
    pcadata<-data.frame(trainingdata$Type,trainingdata.pca$x[,1:3])
    pcatestdata<-data.frame(testingdata$Type,as.matrix(testingdata[-1])%*%as.matrix(t
rainingdata.pca$rotation[,1:3]))
  trainingdata=pcadata
  colnames(trainingdata)[1]<-"Type"
  testingdata=pcatestdata
  colnames(testingdata)[1]<-"Type"
  }
  #未PCA的分类器
  #LDA
  if(method=="LDA"){
    training.lda<-lda(Type~.,data = trainingdata)
    testing.lda.predict<-predict(training.lda,testingdata)
    prediction<-NULL
    for(i in 1:nrow(testingdata)){
      prediction<-c(prediction,decidegroup(as.numeric(testing.lda.predict$x[i,])))
    }
    result<-printdiscriminant(testingdata$Type,prediction,ifprint)
  }
  #QDA
  if (method=="QDA"){
    trainingdata.qda<-qda(Type~., trainingdata,prior=c(1,1,1)/3)
    testing.qda.predict<-predict(trainingdata.qda,testingdata)
    prediction<-NULL
    for(i in 1:nrow(testingdata)){
      prediction<-c(prediction,which.max(testing.qda.predict$posterior[i,]))
    }
    result<-printdiscriminant(testingdata$Type,prediction,ifprint)

  }
  #贝叶斯分类器
  if (method=="Bayes"){
    prediction<-NULL
    for(i in 1:nrow(testingdata)){
      prediction<-c(prediction,which.min(GenearalMdistance(testingdata[i,],trainingda
ta)))
    }
    result<-printdiscriminant(testingdata$Type,prediction,ifprint)
  }
  result
}
```

```
#交叉验证函数
crossvalidation<-function(wholeset=wine,ifstandarize=1,ifpca=0,method,kfold,iftime=0)
{
  a=Sys.time()
  set.seed(250)
  index<-sample(1:nrow(wholeset),nrow(wholeset))
  indexs<-list(index[1:round(nrow(wholeset)/kfold)])
  for ( i in 2:(kfold-1)){
    indexs<-append(indexs,list(index[(round(nrow(wholeset)/kfold)*(i-1)+1):(round(nro
w(wholeset)/kfold)*i)]))
  }
  indexs<-append(indexs,list(index[(round(nrow(wholeset)/kfold)*(kfold-1)+1):length(i
ndex)]))
  accuracy<-NULL
  for (i in 1:10){
    temp=discriminant(wine[setdiff(index,indexs[[i]]),],wine[indexs[[i]],],ifstandari
ze=ifstandarize,ifpca=ifpca,method=method,ifprint = 0)
    accuracy<-c(accuracy,temp)
  }
  if(iftime==1){
    b=Sys.time()
    print(b-a)
  }
mean(accuracy)
}
```

LDA

虽然已经知道协方差矩阵差异巨大，我们仍先对数据进行最简单的线性分类，观察效果。

```
wine.lda<-lda(Type~.,data = wine_trainst)
```

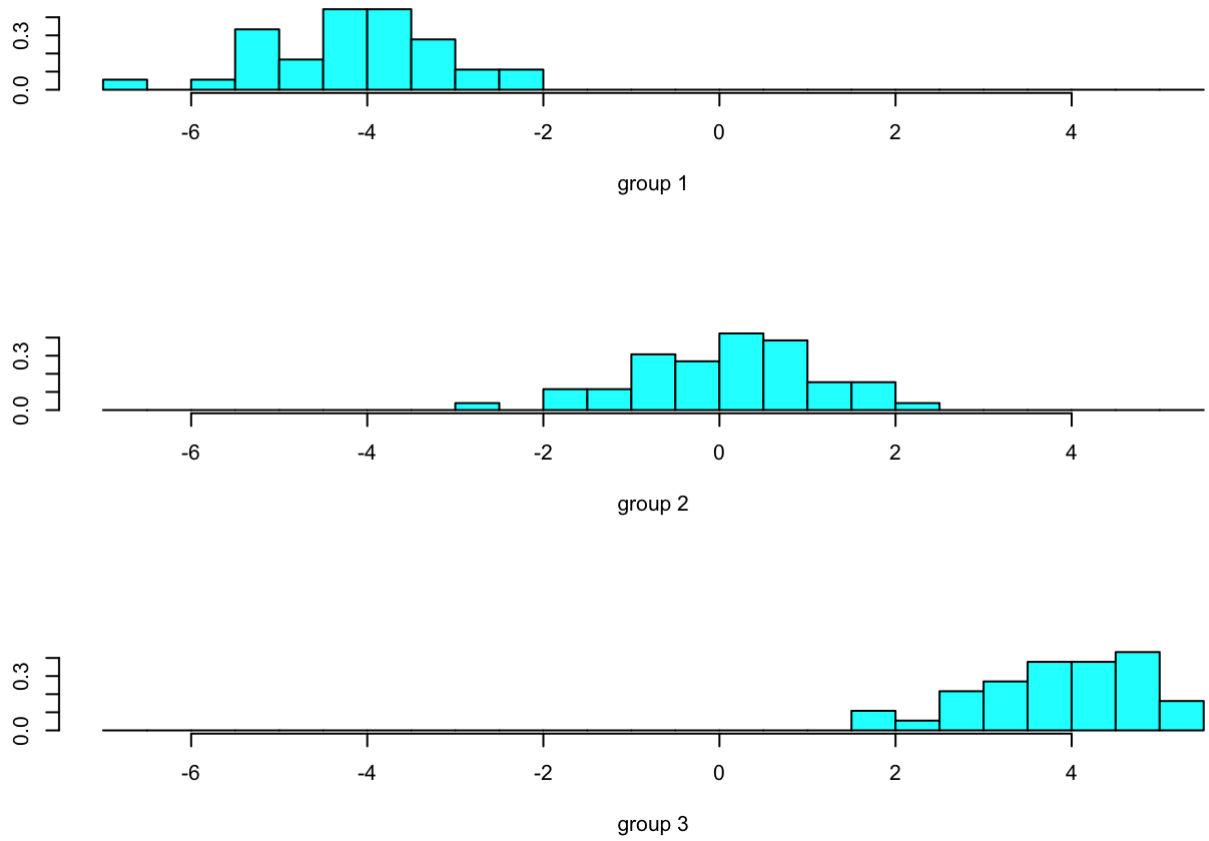
在console输入wine.lda就可以看到所展示的先验、每组数据针对每个变量的平均值、系数和区分度贡献。我们下面来确定判别准则。

```
wine.lda.predict<-predict(wine.lda,wine_trainst)
groupmean<-function(wine_trainst){
  wine.lda<-lda(Type~.,data = wine_trainst)
  wine.lda.predict<-predict(wine.lda,wine_trainst)
  mean1=NULL
  mean2=NULL
  for (i in 1:3){
  mean1=c(mean1,mean(wine.lda.predict$x[wine_trainst$Type==i,1]))
  mean2=c(mean2,mean(wine.lda.predict$x[wine_trainst$Type==i,2]))
  }
  result<-data.frame(mean1,mean2)
  rownames(result)=c("1","2","3")
  result
}
groupmean(wine_trainst)
```

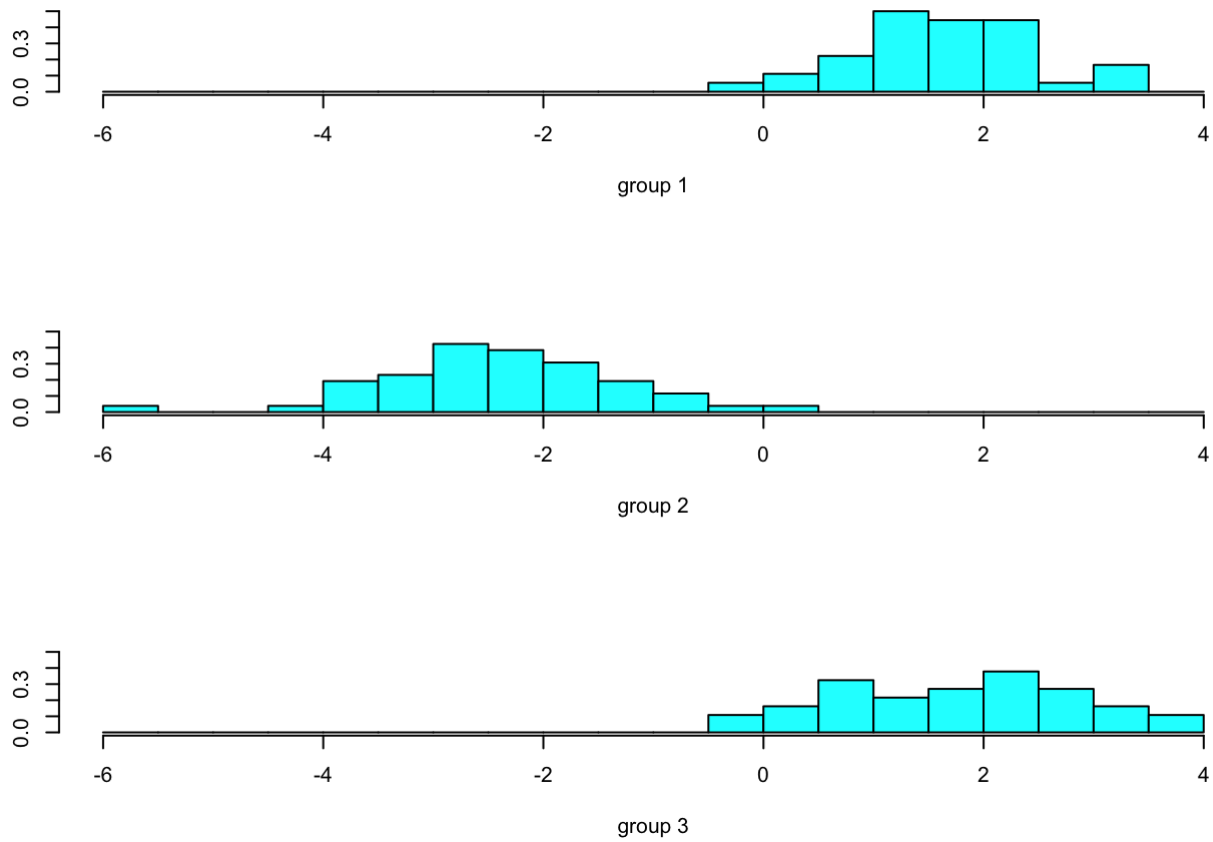
```
##        mean1     mean2
## 1 -4.1250122  1.632261
## 2  0.1016345 -2.397249
## 3  3.8706877  1.780960
```

为了更好的展示数据在两条投影直线上的分布，我们进行柱状图可视化。

```
ldahist(data = wine.lda.predict$x[,1], g=wine_trainst$Type)
```




```
ldahist(data = wine.lda.predict$x[,2], g=wine_trainst$Type)
```



即便表面上数据在第一个维度已经可以很好的使用两个临界值进行判别，但是区分度显示第二个维度还是提供了33%的区分度。所以我们仍旧要使用第二个维度的信息。此处我们采取最简单的欧式距离判别,即使用均值的平均值作为临界值。

```
decidegroup<-function(x){
  if(x[1]< -1.891911 && x[2]>-0.354111){
    return(1)
  }else if(x[1]>1.794088 &&x[2]>-0.6310445){
    return(3)
  }else{
    return(2)
  }
}
```

我们对判别准则进行测试。此处调用的函数为开始建模时写好的总函数。

```
discriminant(wine_train,wine_test,ifstandarize=1,method="LDA")
```

```
##           predicted1 predicted2 predicted3
## original1         20          3          0
## original2          0         19          0
## original3          0          0         11
## [1] "accuracy:"
## [1] 0.9433962
```

```
## [1] 0.9433962
```

可以看到判别结果相当不错，只有三个类型1被判断为了类型2，其他的均正确，正确率达到94.3%。使用10-fold交叉验证得到正确率大致为96.1%。

```
crossvalidation(wholeset = wine,kfold = 10,method = "LDA")
```

```
## [1] 0.9611111
```

我们此时再查看一下只采用一个维度信息的判别标准的效果。

```
decidegroup<-function(x){
  if(x[1]< -1.891911){
    return(1)
  }else if(x[1]>1.794088){
    return(3)
  }else{
    return(2)
  }
}
```

```
crossvalidation(wholeset = wine,kfold = 10,method = "LDA")
```

```
## [1] 0.9111111
```

果然，效果比原先下降了许多，第二个维度的信息是至关重要的。

QDA

考虑到原先每组数据方差不同，我们使用QDA进行后验概率计算，并且采取后验概率最大判别。

```
crossvalidation(wholeset = wine,kfold = 10,method = "QDA")
```

```
## [1] 0.9944444
```

正确率显著的好于线性判别，几乎已经没有可以提升的空间了。

Bayes

考虑到方差和先验均不一样，但是数据却具有正态性，我们使用平均损失最小的贝叶斯分类器（此时等价于广义平方距离判别）。

```
crossvalidation(wholeset = wine,kfold = 10,method = "Bayes")
```

```
## [1] 0.9944444
```

贝叶斯分类的准确率和二次判别相同，原因是贝叶斯仅仅比二次判别多考虑了先验分布，而这个数据集的三类数据数量十分接近，所以并没有显著差异。

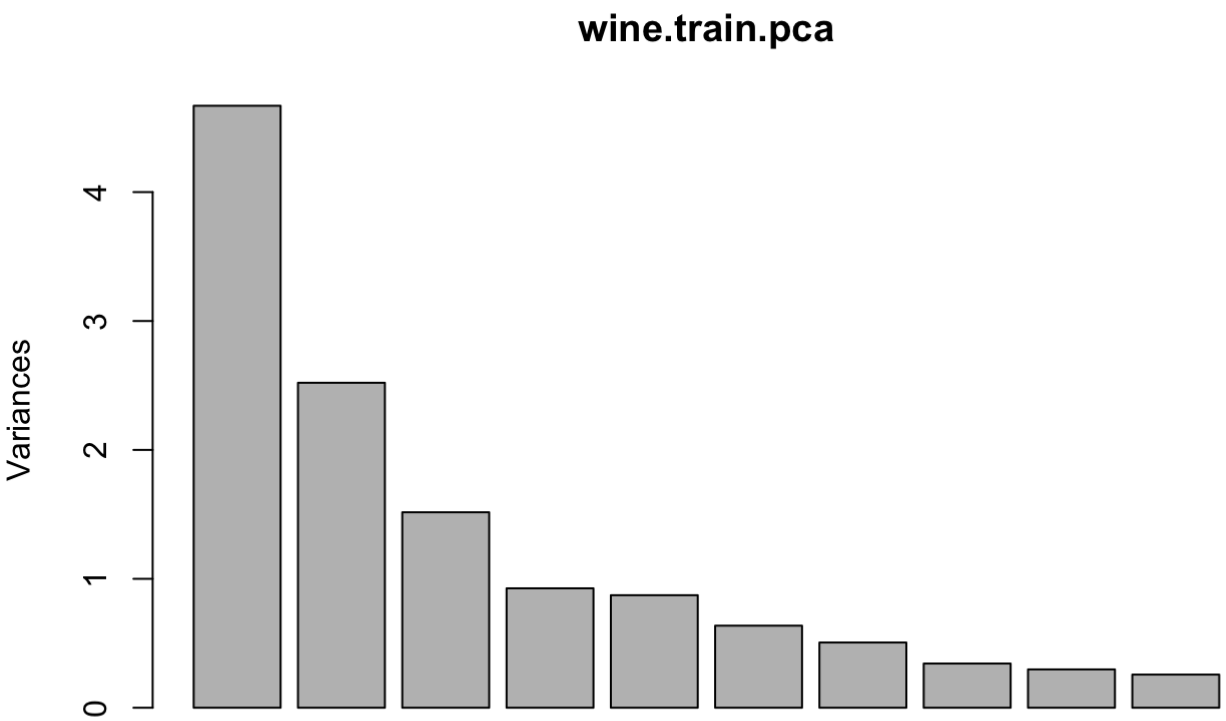
PCA

变量之间很强的相关性提示我们数据的内在维度可能远小于实际维度，并且可能存在噪声。我们尝试使用PCA去除噪声，并使用不同的分类器完成任务。

```
wine.train.pca<-prcomp(wine_trainst[2:14])
summary(wine.train.pca)
```

```
## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5     PC6
## Standard deviation     2.1609 1.5877 1.2314 0.96247 0.93431 0.79784
## Proportion of Variance 0.3592 0.1939 0.1166 0.07126 0.06715 0.04897
## Cumulative Proportion  0.3592 0.5531 0.6698 0.74101 0.80816 0.85713
##                           PC7     PC8     PC9    PC10    PC11    PC12
## Standard deviation     0.71131 0.58548 0.54523 0.50701 0.43431 0.40846
## Proportion of Variance 0.03892 0.02637 0.02287 0.01977 0.01451 0.01283
## Cumulative Proportion  0.89605 0.92242 0.94528 0.96506 0.97957 0.99240
##                          PC13
## Standard deviation     0.3143
## Proportion of Variance 0.0076
## Cumulative Proportion  1.0000
```

```
screeplot(wine.train.pca)
```

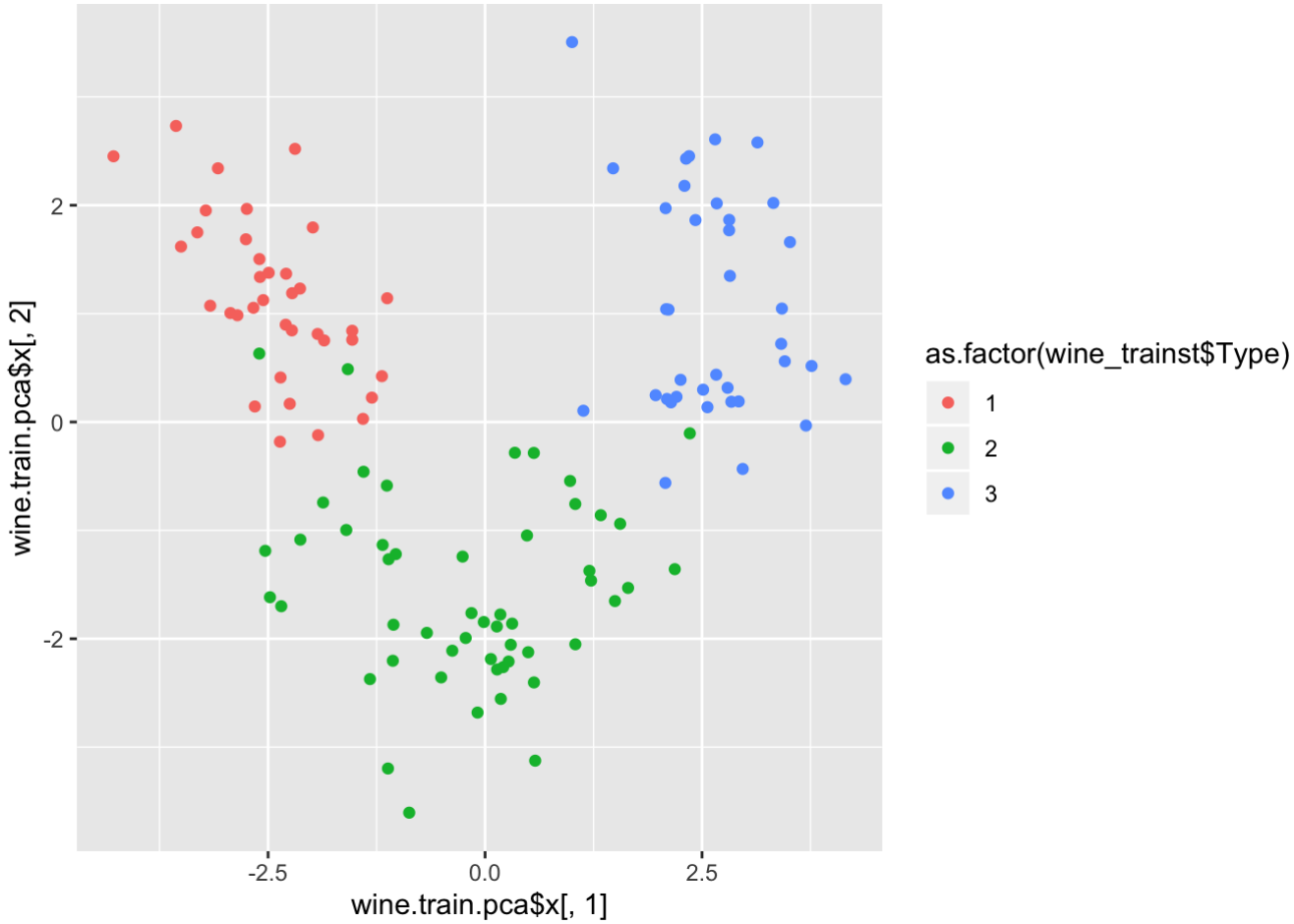



```
accu<-0
for (i in 2:14){
  accu<-c(accu,wine.train.pca$sdev[i-1]^2+accu[i-1])
}
accu[2:14]/13
```

```
##  [1] 0.3592064 0.5531206 0.6697557 0.7410129 0.8081619 0.8571273 0.8960476
##  [8] 0.9224161 0.9452831 0.9650566 0.9795664 0.9924000 1.0000000
```

从图中我们可以得知，前三个主成分已经提供了绝大多数信息，占总比例的68%。而后十个只提供了32%。所以我们可以合理的将数据维度降至三维。出于绘图的方便，我们先将前两维进行可视化。

```
ggplot(wine_trainst,aes(x=wine.train.pca$x[,1],y=wine.train.pca$x[,2],,group=Type,col
or=as.factor(wine_trainst$Type)))+geom_point()
```



可以看出，即使仅仅考虑前两维的信息，这已经是一个相当明显的可以完成的分类任务。下面我们使用不同的分类器在三个维度上进行训练,并在测试集上测试效果。

```
crossvalidation(wholeset = wine,kfold = 10,method = "QDA",ifpca=1)
```

```
## [1] 0.9555556
```

```
crossvalidation(wholeset = wine,kfold = 10,method = "Bayes",ifpca=1)
```

```
## [1] 0.9611111
```

原先接近全对的二次判别和贝叶斯判别在准确率方面有了一些下降。不过当我们考察运行的时间。

```
crossvalidation(wholeset = wine,kfold = 10,method = "QDA",iftime=1)
```

```
## Time difference of 0.08146191 secs
```

```
## [1] 0.9944444
```

```
crossvalidation(wholeset = wine,kfold = 10,method = "QDA",ifpca=1,iftime = 1)
```

```
## Time difference of 0.07454395 secs
```

```
## [1] 0.9555556
```

```
crossvalidation(wholeset = wine,kfold = 10,method = "Bayes",iftime=1)
```

```
## Time difference of 1.968237 secs
```

```
## [1] 0.9944444
```

```
crossvalidation(wholeset = wine,kfold = 10,method = "Bayes",ifpca=1,iftime=1)
```

```
## Time difference of 0.892978 secs
```

```
## [1] 0.9611111
```

可以看到，尽管准确率有了小幅下降，但随着数据维度增大而运算量快速增大的贝叶斯分类器速度被PCA大幅提升。

结论

我们使用了三种分类器对红酒数据进行分类。准确率最高可以达到99%以上。