More real data examples

Fatty Acid Composition Data

Let's try to use LDA on the fatty acid composition data, which aims to classify seven commercial oils. First, we load the data from the package.

```
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
#load oil data from caret package
data(oil)
```

Then, we divide the data to training and test sets.

```
# create training and test sets
set.seed(32)
trainIndex = createDataPartition(oilType, p = 0.7, list = FALSE, times = 1)
train.feature=fattyAcids[trainIndex,] # training features
train.label=oilType[trainIndex] # training labels
test.feature=fattyAcids[-trainIndex,] # test features
test.label=oilType[-trainIndex] # test labels
```

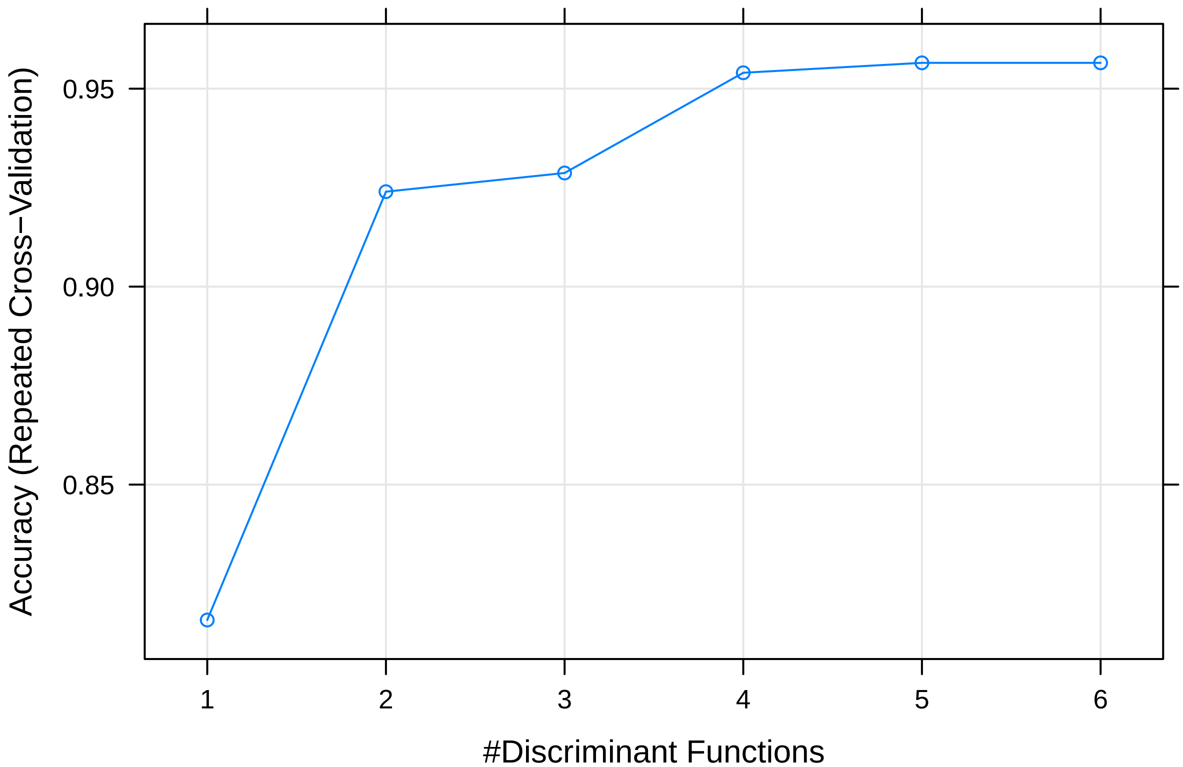
The parameter in LDA is the number of linear discriminants. For binary classification, we cannot tune this parameter, because we can only have one (`2-1`) linear discriminants. However, for multi-class classification, we need to tune this parameter, because the maximum number of linear discriminants is `C-1` where `C` is the number of classes. Here we can tune it via 10-fold CV.

```
#### set up train control
fitControl <- trainControl(## 10-fold CV
  method = "repeatedcv",
  number = 10,
  ## repeated five times
  repeats = 5)
#### training process
set.seed(5)
ldaFit1=train(train.feature,train.label, method = "lda2",
             trControl = fitControl,
             metric = "Accuracy",
             tuneLength=10)
ldaFit1
```

```
## Linear Discriminant Analysis
##
## 70 samples
##  7 predictor
##  7 classes: 'A', 'B', 'C', 'D', 'E', 'F', 'G'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 62, 63, 62, 63, 62, 65, ...
## Resampling results across tuning parameters:
##
```

```
##   dimen  Accuracy   Kappa
##   1      0.8157937  0.7490790
##   2      0.9239841  0.9019950
##   3      0.9287063  0.9076874
##   4      0.9540238  0.9398278
##   5      0.9565238  0.9430278
##   6      0.9565238  0.9430278
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was dimen = 5.
```

```r
plot(ldaFit1)
```



Based on the trained model, we can predict the labels of the test instances.

```r
#### test process
pred=predict(ldaFit1,test.feature)
acc=mean(pred==test.label)
acc
```

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

```r
table(pred,test.label)
```

```
##     test.label
## pred  A  B  C  D  E  F  G
##    A 11  0  0  0  0  0  0
##    B  0  7  0  0  0  0  0
##    C  0  0  0  0  0  0  0
##    D  0  0  0  2  0  0  0
##    E  0  0  0  0  3  0  0
##    F  0  0  0  0  0  3  0
```

```
##    G  0  0  0  0  0  0  0
```

German Credit Data

Now we try to classify the German credit data by LDA. Since we just need to distinguish between two classes, good or bad, there is no parameter to be tuned for this task.

```r
library(caret)
#load german credit data from caret package
data(GermanCredit)
# classify two classes: good or bad
## Show the first 10 columns
str(GermanCredit[, 1:10])
```

```
## 'data.frame':    1000 obs. of  10 variables:
##  $ Duration                 : int  6 48 12 42 24 36 24 36 12 30 ...
##  $ Amount                   : int  1169 5951 2096 7882 4870 9055 2835 6948 3059 5234 ...
##  $ InstallmentRatePercentage: int  4 2 2 2 3 2 3 2 2 4 ...
##  $ ResidenceDuration        : int  4 2 3 4 4 4 4 2 4 2 ...
##  $ Age                      : int  67 22 49 45 53 35 53 35 61 28 ...
##  $ NumberExistingCredits    : int  2 1 1 1 2 1 1 1 1 2 ...
##  $ NumberPeopleMaintenance  : int  1 1 2 2 2 2 1 1 1 1 ...
##  $ Telephone                : num  0 1 1 1 1 0 1 0 1 1 ...
##  $ ForeignWorker            : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ Class                    : Factor w/ 2 levels "Bad","Good": 2 1 2 2 1 2 2 2 2 1 ...
```

```r
## Delete two variables where all values are the same for both classes
GermanCredit[,c("Purpose.Vacation","Personal.Female.Single")] <- list(NULL)
# create training and test sets
set.seed(12)
trainIndex = createDataPartition(GermanCredit$Class, p = 0.7, list = FALSE, times = 1)
train.feature=GermanCredit[trainIndex,-10] # training features
train.label=GermanCredit$Class[trainIndex] # training labels
test.feature=GermanCredit[-trainIndex,-10] # test features
test.label=GermanCredit$Class[-trainIndex] # test labels
#### training process
ldaFit=train(train.feature,train.label, method = "lda",
             trControl = trainControl(method = "none"))
```

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

```r
ldaFit$finalModel
```

```
## Call:
## lda(x, y)
##
## Prior probabilities of groups:
##  Bad Good
##  0.3  0.7
##
## Group means:
##      Duration   Amount InstallmentRatePercentage ResidenceDuration
## Bad  24.72381 4109.448                  3.119048          2.819048
## Good 18.63265 2852.037                  2.975510          2.851020
##           Age NumberExistingCredits NumberPeopleMaintenance Telephone
## Bad  33.55714              1.361905                1.142857 0.6380952
```

```
## Good 36.48571              1.424490                1.171429 0.5857143
##      ForeignWorker CheckingAccountStatus.lt.0
## Bad      0.9809524                  0.4571429
## Good     0.9469388                  0.2020408
##      CheckingAccountStatus.0.to.200 CheckingAccountStatus.gt.200
## Bad                        0.352381                   0.02380952
## Good                       0.244898                   0.06938776
##      CheckingAccountStatus.none CreditHistory.NoCredit.AllPaid
## Bad                   0.1666667                     0.10000000
## Good                  0.4836735                     0.02244898
##      CreditHistory.ThisBank.AllPaid CreditHistory.PaidDuly
## Bad                      0.09523810              0.5571429
## Good                     0.03061224              0.5244898
##      CreditHistory.Delay CreditHistory.Critical Purpose.NewCar
## Bad           0.08571429              0.1619048      0.2952381
## Good          0.07755102              0.3448980      0.2122449
##      Purpose.UsedCar Purpose.Furniture.Equipment Purpose.Radio.Television
## Bad       0.04285714                   0.1904762                0.2142857
## Good      0.10816327                   0.1877551                0.3163265
##      Purpose.DomesticAppliance Purpose.Repairs Purpose.Education
## Bad                 0.01428571      0.01904762        0.07619048
## Good                0.01224490      0.01632653        0.04081633
##      Purpose.Retraining Purpose.Business Purpose.Other
## Bad         0.004761905       0.12380952   0.019047619
## Good        0.010204082       0.08979592   0.006122449
##      SavingsAccountBonds.lt.100 SavingsAccountBonds.100.to.500
## Bad                   0.7238095                      0.1285714
## Good                  0.5489796                      0.1040816
##      SavingsAccountBonds.500.to.1000 SavingsAccountBonds.gt.1000
## Bad                       0.03809524                  0.01904762
## Good                      0.07755102                  0.05918367
##      SavingsAccountBonds.Unknown EmploymentDuration.lt.1
## Bad                   0.09047619               0.2333333
## Good                  0.21020408               0.1367347
##      EmploymentDuration.1.to.4 EmploymentDuration.4.to.7
## Bad                  0.3476190                 0.1285714
## Good                 0.3204082                 0.2163265
##      EmploymentDuration.gt.7 EmploymentDuration.Unemployed
## Bad                0.2142857                    0.07619048
## Good               0.2714286                    0.05510204
##      Personal.Male.Divorced.Seperated Personal.Female.NotSingle
## Bad                        0.04761905                 0.3428571
## Good                       0.04081633                 0.2938776
##      Personal.Male.Single Personal.Male.Married.Widowed
## Bad             0.5380952                    0.07142857
## Good            0.5857143                    0.07959184
##      OtherDebtorsGuarantors.None OtherDebtorsGuarantors.CoApplicant
## Bad                    0.9095238                         0.05238095
## Good                   0.9122449                         0.02244898
##      OtherDebtorsGuarantors.Guarantor Property.RealEstate
## Bad                        0.03809524           0.2000000
## Good                       0.06530612           0.3183673
##      Property.Insurance Property.CarOther Property.Unknown
## Bad           0.2476190         0.3285714        0.2238095
```

```
## Good          0.2285714         0.3306122       0.1224490
##      OtherInstallmentPlans.Bank OtherInstallmentPlans.Stores
## Bad                   0.2095238                   0.07142857
## Good                  0.1142857                   0.04081633
##      OtherInstallmentPlans.None Housing.Rent Housing.Own Housing.ForFree
## Bad                   0.7190476    0.2333333   0.6238095      0.14285714
## Good                  0.8448980    0.1530612   0.7551020      0.09183673
##      Job.UnemployedUnskilled Job.UnskilledResident Job.SkilledEmployee
## Bad               0.02380952             0.1714286           0.6380952
## Good              0.01836735             0.2081633           0.6510204
##      Job.Management.SelfEmp.HighlyQualified
## Bad                                0.1666667
## Good                               0.1224490
##
## Coefficients of linear discriminants:
##                                          LD1
## Duration                       -0.0216856021
## Amount                         -0.0001254021
## InstallmentRatePercentage      -0.1477892602
## ResidenceDuration               0.0059335640
## Age                             0.0129255354
## NumberExistingCredits          -0.2356809987
## NumberPeopleMaintenance         0.1039756437
## Telephone                      -0.3037740810
## ForeignWorker                  -0.7555334926
## CheckingAccountStatus.lt.0     -0.6802651221
## CheckingAccountStatus.0.to.200 -0.1566746314
## CheckingAccountStatus.gt.200    0.6204429507
## CheckingAccountStatus.none      0.5822805598
## CreditHistory.NoCredit.AllPaid -0.8002929257
## CreditHistory.ThisBank.AllPaid -0.8999539610
## CreditHistory.PaidDuly         -0.0910125573
## CreditHistory.Delay             0.2346992535
## CreditHistory.Critical          0.4083883730
## Purpose.NewCar                 -0.5229639366
## Purpose.UsedCar                 0.7796256642
## Purpose.Furniture.Equipment     0.2419758273
## Purpose.Radio.Television        0.1226363293
## Purpose.DomesticAppliance      -0.5208304623
## Purpose.Repairs                -0.1380512748
## Purpose.Education              -0.6722419773
## Purpose.Retraining              0.4130997040
## Purpose.Business                0.0190668746
## Purpose.Other                   0.6246648314
## SavingsAccountBonds.lt.100     -0.2836749290
## SavingsAccountBonds.100.to.500 -0.0742162810
## SavingsAccountBonds.500.to.1000 0.0751536157
## SavingsAccountBonds.gt.1000     0.4764658400
## SavingsAccountBonds.Unknown     0.3384464680
## EmploymentDuration.lt.1        -0.2307035092
## EmploymentDuration.1.to.4      -0.0926359542
## EmploymentDuration.4.to.7       0.4548014521
## EmploymentDuration.gt.7        -0.0577882211
## EmploymentDuration.Unemployed  -0.1124774496
```

```
## Personal.Male.Divorced.Seperated       -0.1276359750
## Personal.Female.NotSingle              -0.0990109104
## Personal.Male.Single                    0.0886405260
## Personal.Male.Married.Widowed           0.0652361362
## OtherDebtorsGuarantors.None            -0.1492637088
## OtherDebtorsGuarantors.CoApplicant     -0.6843004264
## OtherDebtorsGuarantors.Guarantor        0.6096420316
## Property.RealEstate                     0.0609739066
## Property.Insurance                      0.0119492109
## Property.CarOther                       0.1110520535
## Property.Unknown                       -0.3053798731
## OtherInstallmentPlans.Bank             -0.2787415310
## OtherInstallmentPlans.Stores           -0.2033191150
## OtherInstallmentPlans.None              0.2837016837
## Housing.Rent                           -0.2718830803
## Housing.Own                             0.0062887175
## Housing.ForFree                         0.3995714048
## Job.UnemployedUnskilled                -0.1791354529
## Job.UnskilledResident                   0.0737468347
## Job.SkilledEmployee                     0.0089103399
## Job.Management.SelfEmp.HighlyQualified -0.0870595478
```

Now we can predict the labels of test instances:

```
#### test process
pred=predict(ldaFit,test.feature)
acc=mean(pred==test.label)
acc
```

```
## [1] 0.7433333
```

```
table(pred,test.label)
```

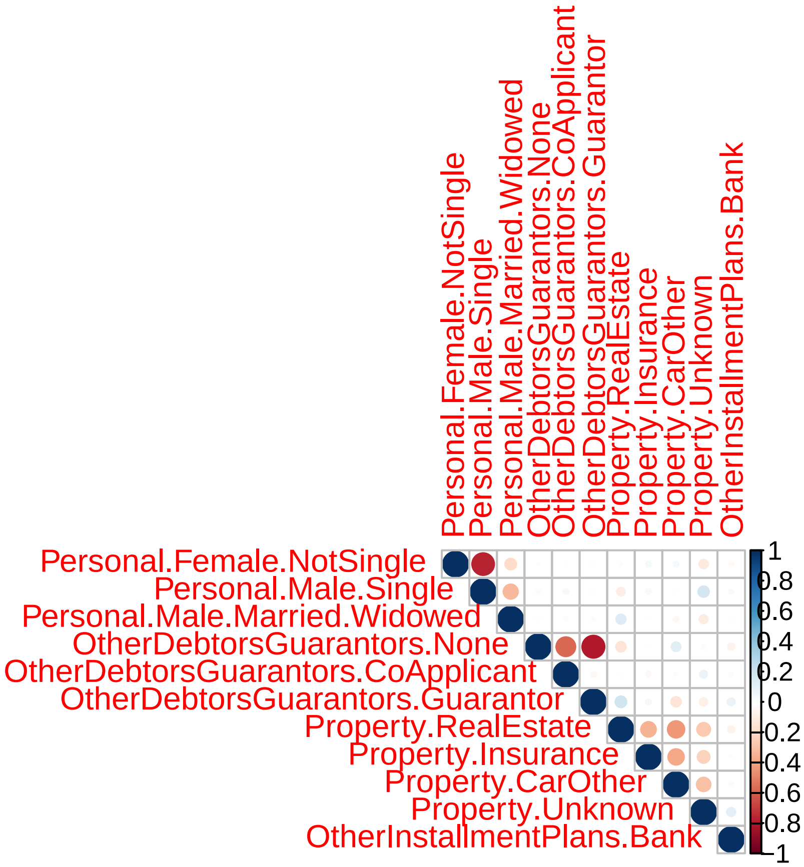
```
##       test.label
## pred   Bad Good
##   Bad   40   27
##   Good  50  183
```

We can see that there is a warning for variable colinearity given by the `lda` function. This means that there are variables with correlations of 1 or -1. Now let's have a look at variable colinearity. We can do this by plot the pairwise correlations. Given that there are 59 variables, a plot of all variables will be too crowded. We can creat a plot that contain a subset of the variables.

```
#### check variable colinearity
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
corrplot(round(cor(train.feature[,40:50]),2), type = "upper")
```



Pima Indians Diabetes Data

We now try to classify the diabetes data by LDA. Similarly to the German credit data, we only have two classes here, so there is no parameter to be tuned in LDA.

```
library(mlbench)
library(caret)
#load Pima Indians Diabetes data from mlbench package
data(PimaIndiansDiabetes)
dim(PimaIndiansDiabetes)
```

```
## [1] 768   9
```

```
levels(PimaIndiansDiabetes$diabetes)
```

```
## [1] "neg" "pos"
```

```
table(PimaIndiansDiabetes$diabetes)
```

```
##
## neg pos
## 500 268
```

```
# create training and test sets
set.seed(76)
trainIndex = createDataPartition(PimaIndiansDiabetes$diabetes, p = 0.7, list = FALSE, times = 1)
train.feature=PimaIndiansDiabetes[trainIndex,-9] # training features
train.label=PimaIndiansDiabetes$diabetes[trainIndex] # training labels
```

```
test.feature=PimaIndiansDiabetes[-trainIndex,-9] # test features
test.label=PimaIndiansDiabetes$diabetes[-trainIndex] # test labels
#### training process
ldaFit=train(train.feature,train.label, method = "lda",trControl = trainControl(method = "none"))
ldaFit$finalModel
```

```
## Call:
## lda(x, y)
##
## Prior probabilities of groups:
##       neg       pos
## 0.6505576 0.3494424
##
## Group means:
##     pregnant  glucose pressure  triceps   insulin     mass  pedigree
## neg 3.262857 111.1029 68.19714 18.98857  68.71143 29.79971 0.4401314
## pos 4.787234 142.4681 71.93617 23.54787 103.90957 35.40479 0.5463564
##          age
## neg 31.36286
## pos 36.63298
##
## Coefficients of linear discriminants:
##                    LD1
## pregnant  0.1105303578
## glucose   0.0269578458
## pressure -0.0104081197
## triceps   0.0071594610
## insulin  -0.0008296858
## mass      0.0603986613
## pedigree  0.4637060543
## age       0.0070345560
```

The labels of the test instances can be predicted via the trained LDA model.

```
#### test process
pred=predict(ldaFit,test.feature)
acc=mean(pred==test.label)
acc
```

```
## [1] 0.7652174
```

```
table(pred,test.label)
```

```
##      test.label
## pred  neg pos
##   neg 131  35
##   pos  19  45
```