STAT542 Statistical Learning Homework 5

Huamin Zhang

Dec 19, 2017

Name: Huamin Zhang (huaminz2@illinois.edu)

Question 1 [15 points]

Answer:

Since A is an invertible square matrix, we get $AA^{-1} = A^{-1}A = I_{n\times n}$. And b is a column vector, so $1 - b^T A^{-1} b$ is a scalar. Thus, if $A - bb^T$ is invertible, we can get

$$\begin{aligned}
&(A - bb^T)(A^{-1} + \frac{A^{-1}bb^TA^{-1}}{1 - b^TA^{-1}b}) \\
&= AA^{-1} - bb^TA^{-1} + \frac{AA^{-1}bb^TA^{-1}}{1 - b^TA^{-1}b} - \frac{bb^TA^{-1}bb^TA^{-1}}{1 - b^TA^{-1}b} \\
&= I - bb^TA^{-1} + \frac{bb^TA^{-1} - bb^TA^{-1}bb^TA^{-1}}{1 - b^TA^{-1}b} \\
&= I - bb^TA^{-1} + \frac{(b - bb^TA^{-1}b)b^TA^{-1}}{1 - b^TA^{-1}b} \\
&= I - bb^TA^{-1} + \frac{b(1 - b^TA^{-1}b)b^TA^{-1}}{1 - b^TA^{-1}b} \qquad 1 - b^TA^{-1}b \text{ is a scalar} \\
&= I - bb^TA^{-1} + bb^TA^{-1} \\
&= I
\end{aligned}$$

Thus, we can get

$$(A - bb^T)(A^{-1} + \frac{A^{-1}bb^TA^{-1}}{1 - b^TA^{-1}b}) = I \Rightarrow (A - bb^T)^{-1} = A^{-1} + \frac{A^{-1}bb^TA^{-1}}{1 - b^TA^{-1}b}$$

Question 2 [30 points]

Answer:

```
# Function Sliced_Inverse
# Usage: Sliced_Inverse(X, Y, nslices = 10)
```

```r
# Arguments
# X: N by p design matrix
# Y: N by 1 response variable
# nslices: number of slices in partioning data. Default 10..
# Value(Output)
# M: p by p matrix; Used to find raw eigenvector
# raw.evectors: p by p matrix; eigen vectors of  M
# evalues: eigen values of M
# evectors:  p by p matrix, transform raw.evectors to beta
Sliced_Inverse <- function(X, Y, nslices = 10){
  X.row = nrow(X); X.col = ncol(X)
  # Center the design matrix
  X.center = scale(X, center=T, scale=F)
  # Calculate sample covariance matrix
  cov_matrix = cov(X.center)*(X.row-1)/X.row
  eigenvector <- eigen(cov_matrix)
  # Calculate cov^-1/2
  V = eigenvector$vectors
  cov_inv_sqrt = V %*% diag(eigenvector$values^(-1/2)) %*% t(V)
  # Calculate Z matrix
  Z = X.center %*% cov_inv_sqrt
  # Sort the dataset
  Z = Z[order(Y),]
  # Divide the dataset and calculate zh
  Z_h = matrix(NA, ncol= X.col, nrow=nslices)
  slice_size = rep(NA, nslices)
  size = round(X.row / nslices)
  for (i in 1:nslices){
    if(i != nslices){ index = ((i-1)*size+1):(i*size)
    }else{ index = ((i-1)*size+1):X.row }
    Z_h[i,] = colMeans(Z[index,])
    slice_size[i] = length(index)
  }
  # Compute the covariance matrix for the slice means of Z,
  # weighted by the slice sizes
  Z_mean = colMeans(Z_h); Center_Z = Z_h - Z_mean
  M = t(Center_Z) %*% apply(Center_Z, 2, "*", slice_size) / X.row
  # Perform PCA on M
  pca.M = eigen(M)
  evalues = pca.M$values
  raw.evectors = pca.M$vectors
  evectors = cov_inv_sqrt %*% raw.evectors
  # normalize beta by column
  normalizer = t(1/apply(evectors,2, function(x) sqrt(sum(x^2))) %*% t(rep(1,X.col)))
  evectors = evectors * normalizer
  return(list(M = M, raw.evectors = raw.evectors,
              evectors = evectors, evalues = evalues))
}
```

a) [10 points]

Here we use an underlying model that can be detected by SIR. Suppose the underlying true model is

$$y = 10000 * (x_1 + x_2)^5 + 100 * (x_2 + x_3)^3 + \epsilon$$

First, we compare our results of β with dr package results based on the model.

```
library(dr)
# Generate 1000 observations
n <- 1000; p <- 10
set.seed(5)
X <- matrix(rnorm(n*p), n, p)
beta1 <- matrix(c(1, 0, 1, rep(0, p-3)))
beta2 <- matrix(c(0, 1, 0, rep(0, p-3)))
Y <-  10000 * (X %*% beta1)^5 +  100 * (X %*% beta2)^3 + 0.1 * rnorm(n)
# fit the SIR model with dr package
fit.sir <- dr(Y~., data = data.frame(X, Y), method = "sir", nslices=10, numdir=p)
# fit the SIR model with our code
myfit <- Sliced_Inverse(X, Y, nslices=10)
# The mean difference
mean(abs(myfit$evectors) - abs(fit.sir$evectors))
```

```
## [1] 2.260605e-14
```

```
# The row eigen values.
round(myfit$evalues,8)
```

```
##  [1] 0.94538553 0.08869129 0.03143086 0.01790409 0.00872885 0.00380978
##  [7] 0.00142676 0.00043480 0.00002482 0.00000000
```

```
# Here we only show the top two eigenvectors
result <- cbind(myfit$evectors[,1:2],fit.sir$evectors[,1:2])
colnames(result) <- c("My.evectors.1","My.evectors.2","dr.evectors.1","dr.evectors.2")
round(result,8)
```

```
##     My.evectors.1 My.evectors.2 dr.evectors.1 dr.evectors.2
## X1     0.70507941   -0.00738079    0.70507941   -0.00738079
## X2     0.04958382    0.94231831    0.04958382    0.94231831
## X3     0.70712422   -0.07106001    0.70712422   -0.07106001
## X4     0.00651881  -0.14892247   0.00651881  -0.14892247
## X5     0.00403295  -0.00226775   0.00403295  -0.00226775
## X6     0.00236615  -0.18251629   0.00236615  -0.18251629
## X7     0.01327442   0.08958131   0.01327442   0.08958131
## X8     0.01037527  -0.19162230   0.01037527  -0.19162230
## X9     0.00004449  -0.05306241   0.00004449  -0.05306241
## X10   -0.00562126   0.06226896  -0.00562126   0.06226896
```

We can see our results are almost the same as the result of dr package, which means the code is correct. Now we compare our estimated direction with the truth.

```r
# normalize for direction beta so that we can compare easily
normalize = function(X){return(X/X[which.max(abs(X))])}
true_dir1 = normalize(beta1)
true_dir2 = normalize(beta2)
esti_dir1 = normalize(myfit$evectors[,1])
esti_dir2 = normalize(myfit$evectors[,2])
result = round(cbind(true_dir1,true_dir2,esti_dir1,esti_dir2),3)
colnames(result) <- c("True dir 1","True dir 2",
                      "Estimated dir 1","Estimated dir 2")
round(result,8)
```

```
##       True dir 1 True dir 2 Estimated dir 1 Estimated dir 2
##  [1,]          1          0           0.997          -0.008
##  [2,]          0          1           0.070           1.000
##  [3,]          1          0           1.000          -0.075
##  [4,]          0          0           0.009          -0.158
##  [5,]          0          0           0.006          -0.002
##  [6,]          0          0           0.003          -0.194
##  [7,]          0          0           0.019           0.095
##  [8,]          0          0           0.015          -0.203
##  [9,]          0          0           0.000          -0.056
## [10,]          0          0          -0.008           0.066
```

Thus, we see our estimated direction is similiar with the truth, which means the SIR method and our code perform well in detecting the underlying model.

b) [10 points]

Here we use an underlying model that can be detected by SIR. Suppose the underlying true model is

$$y = 10000 * (|x_1 + x_2|)^5 + 100 * (x_2 + x_3)^4 + \epsilon$$

First, we compare our results of β with dr package results based on the model.

```r
# Generate 1000 observations
n <- 1000; p <- 10
set.seed(6)
X <- matrix(rnorm(n*p), n, p)
beta1 <- matrix(c(1, 0, 1, rep(0, p-3)))
beta2 <- matrix(c(0, 1, 0, rep(0, p-3)))
Y <-  10000 * (abs(X %*% beta1))^5 +  100 * (X %*% beta2)^4 + 0.1 * rnorm(n)
# fit the SIR model with dr package
fit.sir <- dr(Y~., data = data.frame(X, Y), method = "sir", nslices=10, numdir=p)
# fit the SIR model with our code
myfit <- Sliced_Inverse(X, Y, nslices=10)
# The mean difference
mean(abs(myfit$evectors) - abs(fit.sir$evectors))
```

```
## [1] -9.668048e-17
```

```r
# The row eigen values.
round(myfit$evalues,8)
```

```
##  [1] 0.03694467 0.02284099 0.01777122 0.00791281 0.00645281 0.00345852
##  [7] 0.00119634 0.00039209 0.00002694 0.00000000
```

```r
# Here we only show the top two eigenvectors
result <- cbind(myfit$evectors[,1:2],fit.sir$evectors[,1:2])
colnames(result) <- c("My.evectors.1","My.evectors.2","dr.evectors.1","dr.evectors.2")
round(result,8)
```

```
##     My.evectors.1 My.evectors.2 dr.evectors.1 dr.evectors.2
## X1    -0.20112906   -0.00872315   -0.20112906   -0.00872315
## X2    -0.29950636   -0.08985453   -0.29950636   -0.08985453
## X3     0.03795304   -0.06314545    0.03795304   -0.06314545
## X4     0.39225994   -0.34173076    0.39225994   -0.34173076
## X5    -0.34748087   -0.65222486   -0.34748087   -0.65222486
## X6     0.68475251    0.03661265    0.68475251    0.03661265
## X7    -0.21116670    0.14986605   -0.21116670    0.14986605
## X8    -0.22294742    0.21699837   -0.22294742    0.21699837
## X9     0.07650982   -0.31045846    0.07650982   -0.31045846
## X10   -0.15733757    0.52764806   -0.15733757    0.52764806
```

We can see our results are almost the same as the result of dr package, which means the code is still correct. Now we compare our estimated direction with the truth.

```r
# normalize for direction beta so that we can compare easily
normalize = function(X){return(X/X[which.max(abs(X))])}
true_dir1 = normalize(beta1)
true_dir2 = normalize(beta2)
```

```r
esti_dir1 = normalize(myfit$evectors[,1])
esti_dir2 = normalize(myfit$evectors[,2])
result = round(cbind(true_dir1,true_dir2,esti_dir1,esti_dir2),3)
colnames(result) <- c("True dir 1","True dir 2",
                      "Estimated dir 1","Estimated dir 2")
round(result,8)
```

```
##       True dir 1 True dir 2 Estimated dir 1 Estimated dir 2
##  [1,]          1          0          -0.294           0.013
##  [2,]          0          1          -0.437           0.138
##  [3,]          1          0           0.055           0.097
##  [4,]          0          0           0.573           0.524
##  [5,]          0          0          -0.507           1.000
##  [6,]          0          0           1.000          -0.056
##  [7,]          0          0          -0.308          -0.230
##  [8,]          0          0          -0.326          -0.333
##  [9,]          0          0           0.112           0.476
## [10,]          0          0          -0.230          -0.809
```

Thus, we see all the eigenvalues are small, and the our estimated direction is totally difference with the truth, which means the SIR method and our code perform bad in detecting the underlying model.

Question 3 [55 points]

Answer:

Before the report, let me introduce the data clean part.

First, we delete the text data features, such as `original_title`, `overview`, `title`, `tagline`.

Second, we delete the `homepage` feature.

Third, we delete information after the release date, such as `popularity`, `vote_count`.

Fourth, some of the **feature engineering** part. We abstract the information from nested features, such as `genres`, `keywords` and `production_companies` etc. For example, if the raw observation A is `{attr1:(A,B)}` B is `{attr1:(A,C)}`, we transform it to A: `{attr1.A:1, attr1.B:1, attr1.C:0}` and B:`{attr1.A:1, attr1.B:0, attr1.C:1}`.

Fifth, we select the variables that appears in more than five observations. We think if one variables that appears in only one observation doesn't make sense.

Sixth, we abstract the year and month information from the `release date`.

Seventh, we delete these obsearvation whose `budget` and `revenue` is 0 which means there is missing value.

Part I. Predict Revenue

a) [5 points]

In this question, we will use two models, **group lasso regression and random forest.**

First, we select the features whose correlation with y is not less than 0.1. Finally, we have 91 features, and there are 1627 observation in train data adn 1562 observation in test data.

For group lasso regression model, the measure we use is mean absolute error. We use 10-fold cross-validation to tune the parameter `lambda`. The code and tuning result can be find in HW5_STAT542_huaminz2.Rmd and regression_tune.Rdata.

For random forest model, the measure we use is mean absolute error. We tune the parameter `mtry`, `nodesize` and `ntree`. The code and tuning result can be find in HW5_STAT542_huaminz2.Rmd and regression_tune.Rdata.

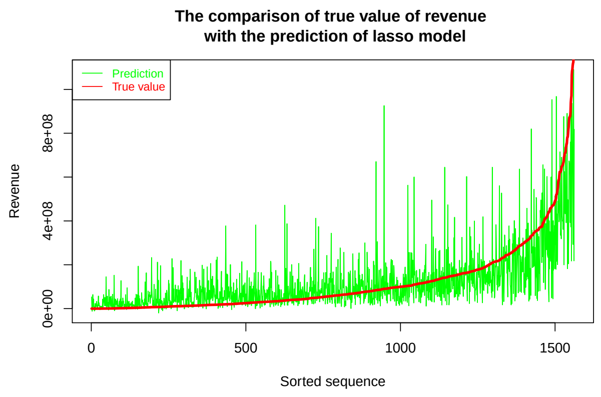
For feature engineering, we also create a feature as `revenue_budget_ratio` as the prediction target.

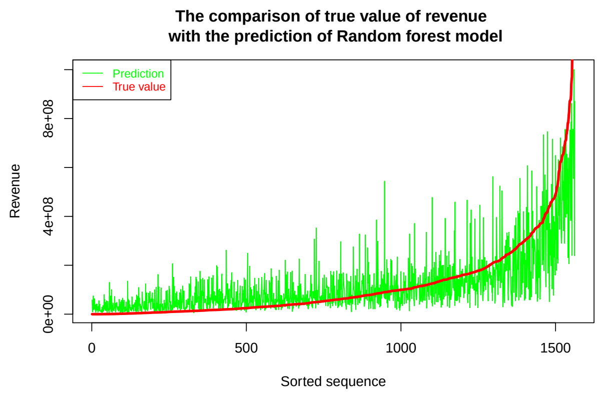
b) [5 points]

Here use the mean absolute error to compare these two model and select the best one with the smallest error.

```
##                revenue revenue_budget_ratio
## Group Lasso   69909136            168858496
## Random Forest 65174753            130964962
```

The mae of group lasso regression model is 69.9091364 million and the mae of random forest model 65.1747526 million. And the error with `revenue_budget_ratio` is about 168.8584963 million with lasso model, and 130.9649617 million with Random Forest model. Thus, we use the random forest model with `revenue` as target. The parameter of random forest is `ntree` = 500, `mtry` = 30 and `nodesize` = 5. Here we show the plot of prediction of random forest and group lasso regression model.



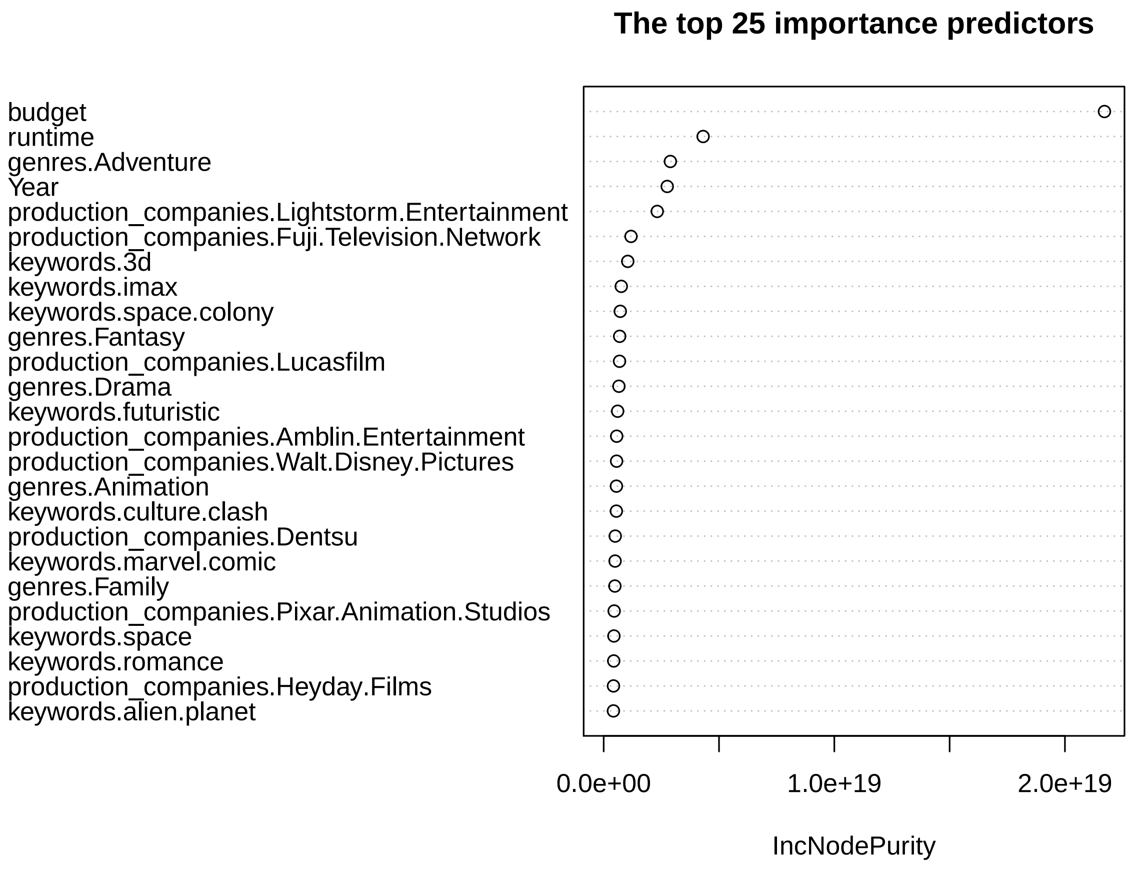


In this plot, we see the trend of test data is well fitted by prediction values.

The importance of predictors (top 10) given by random forest can be shown as the following:

```
##                                              IncNodePurity
## budget                                        2.171290e+19
## runtime                                       4.310900e+18
## genres.Adventure                              2.891530e+18
## Year                                          2.754997e+18
## production_companies.Lightstorm.Entertainment  2.322185e+18
## production_companies.Fuji.Television.Network   1.185320e+18
## keywords.3d                                   1.043335e+18
## keywords.imax                                 7.627126e+17
## keywords.space.colony                         7.195222e+17
## genres.Fantasy                                6.933324e+17
```

Also, we make a plot of the importance predictors in Random Forest model.



From the result, we can say the `budget`, `runtime`, `genres.Adventure`(whether the genres is Adventure or not), `Year` are the most important predictors. And more, for example, the `revenue` can be affected much whether the file genres is Fantasy

or whether the file is a 3D imax film.

Part II. Predict vote_average

a) [5 points]

In this question, we will use four models, **group lasso regression, logistic regression, SVM and random forest.**

For both model, we first select the features whose correlation with y is not less than 0.05. Finally, we have 203 features.

Since in the raw training data, we have 325 positive observations and 1302 negative observations, which means it is an unbalanced dataset. **Here we try three different strategy to deal with the problem, oversampling, undersampling and SMOTE algorithm.** To compare these model, we will calculate accuracy, sensitivity, specificity, precision, F1 score.

For oversampling method, we choose 1302 observations from positive observations with replacement, thus we have 1302 positive observations and 1302 negative observations.

For undersampling method, we choose 325 observations from negative observations, thus we have 325 positive observations and 325 negative observations.

For SMOTE method(package 'DMwR'), we artificially generate new examples of the minority class using the nearest neighbors of these cases. Furthermore, the majority class examples are also under-sampled, leading to a more balanced dataset. Here we use the parameters `perc.over = 200` and `perc.under = 150`. Finally we have 910 positive observations and 910 negative observations.

For group lasso regression model, we use 10-fold cross-validation to tune the parameter `lambda` and the measure we use is auc. The code and tuning result can be find in HW5_STAT542_huaminz2.Rmd and classification_tune.Rdata.

For logistic regression model, The code can be find in HW5_STAT542_huaminz2.Rmd and classification_tune.Rdata.

For SVM model, we use tune the parameter `kernel` and the constant of the regularization term and the measure we use is sensitivity. The code and tuning result can be find in HW5_STAT542_huaminz2.Rmd and classification_tune.Rdata.

For random forest model, the measure we use is sensitivity. We tune the parameter `mtry`, `nodesize` and `ntree`. The code and tuning result can be find in HW5_STAT542_huaminz2.Rmd and classification_tune.Rdata.

b) [5 points]

Here let show the result.

The result of SMOTE dataset.

```
##             lasso      logistic  SVM        Random Forest
## sensitivity 0.4026846  0.6208054 0.3724832  0.3959732
## specificity 0.8686709  0.6693038 0.8623418  0.9193038
## accuracy    0.7797695  0.6600512 0.768886   0.8194622
## precision   0.09493671 0.1463608 0.08781646 0.09335443
## F1 score    0.4109589  0.4106548 0.380789   0.4555985
```

The result of over sampling dataset.

```
##             lasso     logistic   SVM       Random Forest
## sensitivity 0.5469799 0.3959732  0.4395973 0.3959732
## specificity 0.8156646 0.8583861  0.8243671 0.9185127
## accuracy    0.7644046 0.7701665  0.7509603 0.818822
## precision   0.1289557 0.09335443 0.1036392 0.09335443
## F1 score    0.4697406 0.3966387  0.4024578 0.4547206
```

The result of under sampling dataset.

```
##             lasso     logistic  SVM        Random Forest
## sensitivity 0.5805369 0.4630872 0.409396   0.3959732
## specificity 0.7998418 0.7745253 0.7966772  0.9177215
## accuracy    0.7580026 0.7151088 0.7227913  0.8181818
## precision   0.1368671 0.1091772 0.09651899 0.09335443
## F1 score    0.4779006 0.3828017 0.3604136  0.4538462
```

The result of raw dataset.

```
##             lasso      logistic   SVM        Random Forest
## sensitivity 0.2516779  0.2684564  0.2583893  0.3959732
## specificity 0.9343354  0.9129747  0.9082278  0.9185127
## accuracy    0.8040973  0.7900128  0.784251   0.818822
## precision   0.05933544 0.06329114 0.06091772 0.09335443
## F1 score    0.3289474  0.3278689  0.3136456  0.4547206
```

According to F1 score which considers both the precision and the recall, the final model we choose is lasso regression with under sampling dataset. The lambda we will is 0.009511977.

The importance of predictors (top 10)(According to the coefficient of variable, the larger the more influence) given by lasso regression can be shown as the following:

```
##  [1] "spoken_languages.en"
##  [2] "production_companies.Westdeutscher.Rundfunk..WDR."
##  [3] "genres.Thriller"
##  [4] "genres.Family"
##  [5] "genres.Action"
##  [6] "genres.Comedy"
##  [7] "keywords.vampire"
##  [8] "genres.Horror"
##  [9] "production_countries.United.States.of.America"
## [10] "keywords.duringcreditsstinger"
```

Thus, the most important predictor is whether the language of the film is English. Also, we see the genres of a film affects a lot.

d) [5 points]

According to the information of Star Wars(StarWar.csv), we make the classification and regression. The prediction(see the code in rmd file) is that we think the total revenue will be 737310139 and the rating of this file will be greater than 7.(The truth is the rating on IMDB is 8.1 and 93% on Rotten Tomatoes until now, which fit our model)