Untitled

Qianhua Zhou

2024-09-20

Question 2: Modeling High-Dimensional Data

We will use the `golub` dataset from the `multtest` package. This dataset contains 3051 genes from 38 tumor mRNA samples from the leukemia microarray study Golub et al. (1999). This package is not included in `R`, but on `bioconductor`. Install the latest version of this package from `bioconductor`, and read the documentation of this dataset to understand the data structure of `golub` and `golub.cl`.

a. [25 points] We will not use this data for classification (the original problem). Instead, we will do a toy regression example to show how genes are highly correlated and could be used to predict each. Carry out the following tasks:

    - Perform marginal association test for each gene with the response `golub.cl` using `mt.teststat()`. Use `t.equalvar` (two sample t test with equal variance) as the test statistic.
    - Sort the genes by their p-values and select the top 100 genes
    - Construct a dataset with the top 10 genes and another one (call it X) with the remaining genes
    - Perform principal component analysis (PCA) on the top 100 genes and extract the first principal component, **use this as the outcome** y. Becareful about the oriantation of the data matrix.
    - Perform ridge regression with 19-fold cross-validation on X and the outcome y. Does your model fit well? Can you provide detailed model fitting results to support your claim?
    - Fit ridge regression but use GCV as the criterion. Does your model fit well?

```r
#if (!requireNamespace("BiocManager", quietly = TRUE)) {
#    install.packages("BiocManager")
#}
#BiocManager::install("multtest")
library(multtest)
data(golub)
```

```r
t_stats <- mt.teststat(golub, golub.cl, test="t.equalvar")
p_values <- 2 * pt(-abs(t_stats), df = 36) # two-tailed p-values
sorted_indices <- order(p_values)
top_100_genes <- golub[sorted_indices[1:100], ]
top_10_genes <- golub[sorted_indices[1:10], ]
X <- golub[sorted_indices[11:100], ] # X
# PCA
pca_result <- prcomp(t(top_100_genes), scale. = TRUE)
# Extract the first few principal components
y <- pca_result$x[, 1]
# Load glmnet and perform ridge regression with cross-validation
library(glmnet)
```

```
## Loading required package: Matrix

## Loaded glmnet 4.1-8
```

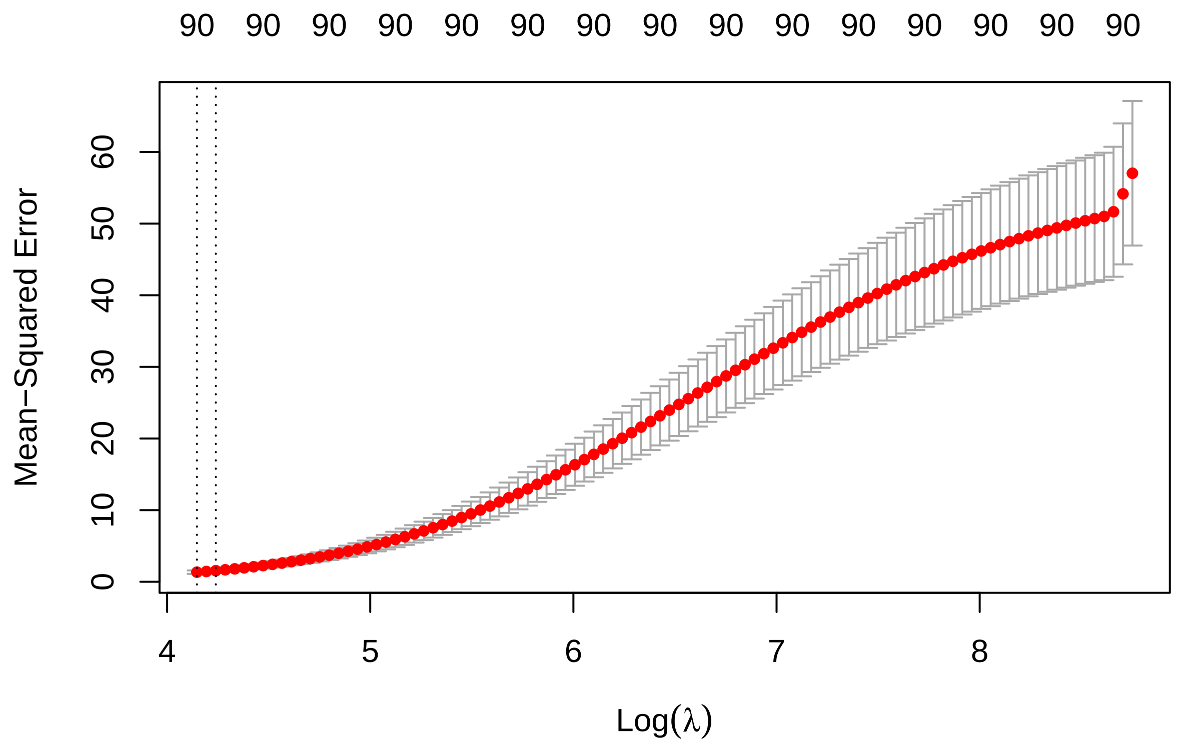
```r
set.seed(1)
cv_ridge <- cv.glmnet(t(X), y, alpha = 0, nfolds = 19)
```

```
## Warning: Option grouped=FALSE enforced in cv.glmnet, since < 3 observations per
## fold
```

```r
best_lambda <- cv_ridge$lambda.min
cat("Lambda value under 19-fold cross-validation:", best_lambda, "\n")
```

```
## Lambda value under 19-fold cross-validation: 63.22453
```

```r
# Plot to visualize lambda selection
plot(cv_ridge)
```



```r
# Extract coefficients at the optimal lambda
best_lambda <- cv_ridge$lambda.min
# Fit the final model with the best lambda
ridge_model <- glmnet(t(X), y, alpha = 0, lambda = best_lambda)
# Display coefficients
coef(ridge_model)
```

```
## 91 x 1 sparse Matrix of class "dgCMatrix"
##                       s0
## (Intercept)  3.75003331
## V1           0.22344308
## V2          -0.14724662
## V3           0.10096671
## V4           0.14810036
## V5           0.06678804
## V6           0.12122677
## V7          -0.11714696
## V8           0.12087425
## V9          -0.12508958
## V10         -0.15529850
## V11          0.13799593
## V12         -0.11466920
## V13         -0.18661617
## V14          0.09855022
## V15         -0.21397053
## V16          0.13420353
## V17          0.06236390
## V18          0.18348531
## V19          0.14296630
## V20         -0.25306392
## V21          0.10459252
## V22         -0.13880184
## V23         -0.08845951
## V24         -0.15844504
## V25          0.11333216
## V26          0.13555132
## V27          0.08183742
## V28          0.09968876
## V29          0.10101710
## V30         -0.10821511
## V31          0.16089232
## V32          0.11786225
## V33         -0.16114859
## V34          0.13276820
## V35         -0.12555321
## V36         -0.18420356
## V37          0.13743163
## V38          0.14320129
## V39          0.09175576
## V40          0.11431908
## V41          0.20881719
## V42          0.18589524
## V43          0.07547327
## V44          0.15256574
## V45         -0.16543395
## V46          0.10618685
## V47          0.10556113
## V48          0.20096384
## V49         -0.17664762
## V50         -0.17043693
## V51          0.12791491
## V52         -0.08337769
## V53         -0.16581938
## V54         -0.21597924
## V55          0.14273287
## V56         -0.15180050
## V57         -0.16110609
## V58         -0.15357680
## V59         -0.17943429
## V60          0.13188646
## V61         -0.15002133
## V62          0.11754486
## V63          0.06047721
## V64          0.17606099
## V65          0.15940233
## V66          0.09337962
## V67          0.11678008
## V68         -0.14147172
## V69         -0.06722247
## V70         -0.17972962
## V71         -0.08887630
## V72         -0.12962388
## V73         -0.09913002
## V74          0.09675740
## V75         -0.16274143
## V76          0.14423786
## V77         -0.20144932
## V78         -0.09314216
## V79         -0.11934818
## V80         -0.17083549
## V81         -0.15297295
## V82          0.08440054
## V83          0.17497310
## V84         -0.10051948
## V85         -0.22256635
## V86          0.09877243
## V87         -0.15733827
## V88         -0.23749983
## V89         -0.24227097
## V90         -0.13876792
```

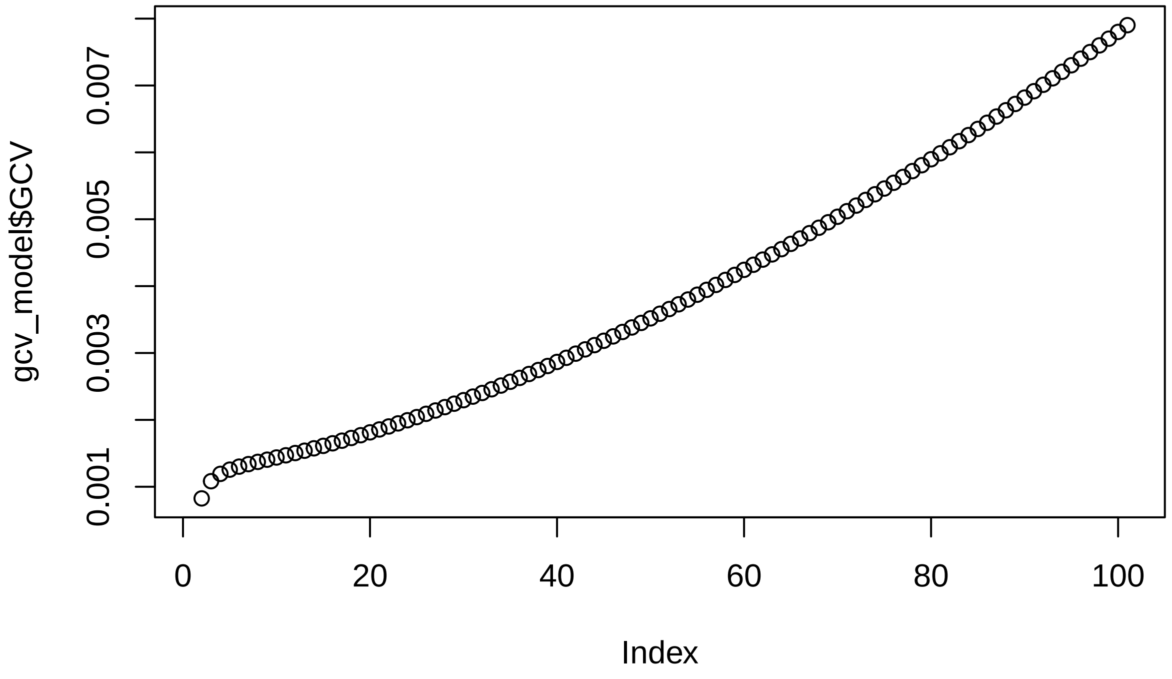
```r
predictions <- predict(ridge_model, newx = t(X))
mse <- mean((y - predictions)^2)
r_squared <- 1 - sum((y - predictions)^2) / sum((y - mean(y))^2)
cat("Mean Squared Error:", mse, "\n")
```

```
## Mean Squared Error: 1.249251
```

```r
cat("R-squared:", r_squared, "\n")
```

```
## R-squared: 0.9772938
```

```r
library(MASS)
# Fit ridge regression using GCV criterion
gcv_model <- lm.ridge(y ~ t(X), lambda = seq(0, 100, by = 1))
# Plot GCV values
plot(gcv_model$GCV)
```



```r
# Get the lambda with the lowest GCV
best_gcv_lambda <- gcv_model$lambda[which.min(gcv_model$GCV)]
cat("Lambda value under GCV:", best_gcv_lambda, "\n")
```

```
## Lambda value under GCV: 1
```

```r
# Fit the final ridge model using the best GCV lambda
gcv_final_model <- lm.ridge(y ~ t(X), lambda = best_gcv_lambda)

# Display coefficients
gcv_coef <- coef(gcv_final_model)
# Calculate fitted values using the GCV model
# Predictions
intercept <- gcv_coef[1]
coefficients <- gcv_coef[-1]
predictions_gcv <- intercept + as.matrix(t(X)) %*% coefficients
# Calculate MSE
mse <- mean((y - predictions_gcv)^2)
```

```r
# Calculate R-squared
ss_res <- sum((y - predictions_gcv)^2)
ss_tot <- sum((y - mean(y))^2)
r_squared <- 1 - (ss_res / ss_tot)
cat("Mean Squared Error:", mse, "\n")
```

```
## Mean Squared Error: 0.0002225726
```

```r
cat("R-squared:", r_squared, "\n")
```

```
## R-squared: 0.999996
```

For the ridge regression with 19-fold cross-validation, since the R-squared is 0.9772938, which is above 0.7,and very close to 1, the model is a good fit.

For the ridge regression using GCV, since the R-squared is0.999996, which is above 0.7, and very close to 1, also, mse is 0.0002225726, very close to 0, the model is agood fit.