CSC 522 : Automated Learning and Data Analysis

Homework 5

Roopak Venkatakrishnan - rvenkat7@ncsu.edu

March 25, 2013

1 Question 1 [57 Points (25 + 32)] - Regression

In this problem we will investigate various methods for tting a linear model for regression. Download the regprob.zip le from the course website.

1. Given a set of n real-valued responses y_i and a set of p predictors, we might try to model y_i as a linear combination of the p predictors. The form of this type of linear model is:

$$y_i = \beta_0 + \sum_{j=1}^{p} + \beta_j \times x_{ij}$$

where y_i is the the value of the response for the i^{th} observation, x_{ij} is the value for the j^{th} predictor for observation i, and β_0 is the intercept. To nd good values for all of the βs, one approach is to minimize the sum of squared errors (SSE), shown below:

$$SSE = \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} \beta_j \times x_{ij})$$

This approach is known as regression via ordinary least squares (OLS). Representing this model in matrix notation, the model can be written in an equivalent form as $Y = X\beta$. Now Y is an $n \times 1$ column vector containing the response variable, X is an $n \times (p + 1)$ matrix that contains the p predictors for all n observations as well as a column of all 1s to represent the intercept, and β is a $p+1$ vector. With some matrix calculus it can be shown the value of β that minimizes the SSE is given by:

$$\hat{\beta}_{OLS} = (X^T X)^{-1} X^T Y$$

where T indicates a matrix transpose. This formula will give a $(p+1)$ vector containing the estimated regression coecients.

Complete the following tasks:

- Load *train.csv*

```
> train <- read.csv(file.choose())
```

- Compute the OLS estimates using the data in train.csv. Do not use a package to do this, instead compute it directly from the formula given above. There are 10 predictors in the le, so your solution should contain 11 estimated regression coeffcients (1 for each predictor plus 1 for the intercept, 11 numbers in total).

```
> library(caret)
Loading required package: cluster
Loading required package: foreach
foreach: simple, scalable parallel programming from Revolution Analytics
Use Revolution R for scalability, fault tolerance and more.
http://www.revolutionanalytics.com
Loading required package: lattice
Loading required package: plyr
Loading required package: reshape2
> folds <- createFolds(train[["Y"]],k=5,list=FALSE)
> folds <- createFolds(train[["Y"]],k=5)
> t1 <- c(folds$Fold1,folds$Fold2,folds$Fold3,folds$Fold4)
> training_w5 <- train[t1,]
> t2 <- c(folds$Fold1,folds$Fold2,folds$Fold3,folds$Fold5)
> training_w4 <- train[t2,]
> t3 <- c(folds$Fold1,folds$Fold2,folds$Fold4,folds$Fold5)
> training_w3 <- train[t3,]
> t4 <- c(folds$Fold1,folds$Fold3,folds$Fold4,folds$Fold5)
> training_w2 <- train[t4,]
> t5 <- c(folds$Fold2,folds$Fold3,folds$Fold4,folds$Fold5)
> training_w1 <- train[t5,]
> X_w5 <- training_w5[2:11]
> X_w4 <- training_w4[2:11]
> X_w3 <- training_w3[2:11]
> X_w2 <- training_w2[2:11]
> X_w1 <- training_w1[2:11]
> Y_w5 <- training_w5[1]
> Y_w4 <- training_w4[1]
> Y_w3 <- training_w3[1]
> Y_w2 <- training_w2[1]
> Y_w1 <- training_w1[1]
> X0 <- rep(1,80)
> X_w5 <- cbind(X0,X_w5)
> X_w4 <- cbind(X0,X_w4)
> X_w3 <- cbind(X0,X_w3)
> X_w2 <- cbind(X0,X_w2)
> X_w1 <- cbind(X0,X_w1)
> X_w5_t <- t(X_w5)
> X_w4_t <- t(X_w4)
> X_w3_t <- t(X_w3)
> X_w2_t <- t(X_w2)
> X_w1_t <- t(X_w1)
> xtx_w5 <- as.matrix(X_w5_t) %*% as.matrix(X_w5)
> xtx_w4 <- as.matrix(X_w4_t) %*% as.matrix(X_w4)
> xtx_w3 <- as.matrix(X_w3_t) %*% as.matrix(X_w3)
> xtx_w2 <- as.matrix(X_w2_t) %*% as.matrix(X_w2)
> xtx_w1 <- as.matrix(X_w1_t) %*% as.matrix(X_w1)
> xty_w5 <- as.matrix(X_w5_t) %*% as.matrix(Y_w5)
> xty_w4 <- as.matrix(X_w4_t) %*% as.matrix(Y_w4)
> xty_w3 <- as.matrix(X_w3_t) %*% as.matrix(Y_w3)
> xty_w2 <- as.matrix(X_w2_t) %*% as.matrix(Y_w2)
> xty_w1 <- as.matrix(X_w1_t) %*% as.matrix(Y_w1)
> xtx_w5_i <- solve(xtx_w5)
> xtx_w4_i <- solve(xtx_w4)
> xtx_w3_i <- solve(xtx_w3)
```

```
> xtx_w2_i <- solve(xtx_w2)
> xtx_w1_i <- solve(xtx_w1)
> beta_w5 <- as.matrix(xtx_w5_i) %*% xty_w5
> beta_w4 <- as.matrix(xtx_w4_i) %*% xty_w4
> beta_w3 <- as.matrix(xtx_w3_i) %*% xty_w3
> beta_w2 <- as.matrix(xtx_w2_i) %*% xty_w2
> beta_w1 <- as.matrix(xtx_w1_i) %*% xty_w1
>########## The 5 sets of Beta OLS ###################
> beta_w5
                Y
X0    1.987023233
X1    1.477377154
X2   -1.960537048
X3    3.002801541
X4    1.760115960
X5   -0.492587970
X6   -0.026595553
X7    0.010870996
X8   -0.004196558
X9   -0.014542570
X10   0.039114341
> beta_w4
                 Y
X0    1.9991479285
X1    1.4852979072
X2   -1.9891755456
X3    3.0158514848
X4    1.7838078750
X5   -0.5056817162
X6   -0.0540927802
X7    0.0002767814
X8    0.0011992199
X9   -0.0106488097
X10   0.0086930600
> beta_w3
                Y
X0    2.009039939
X1    1.478394960
X2   -1.960978114
X3    2.988727432
X4    1.750058132
X5   -0.493569447
X6   -0.023029988
X7    0.030725486
X8    0.015261200
X9    0.001889546
X10  -0.004013025
> beta_w2
                Y
X0    2.003628822
X1    1.489469121
X2   -1.962818726
X3    3.013862887
X4    1.764748262
X5   -0.514227186
X6   -0.016802510
```

```
X7   −0.003583540
X8   −0.022725361
X9   −0.004960674
X10   0.008137362
> beta_w1
                Y
X0    2.005209639
X1    1.489807861
X2   −1.937527185
X3    3.015534910
X4    1.748714536
X5   −0.475419570
X6   −0.037666679
X7    0.025749127
X8    0.012059920
X9   −0.008382290
X10  −0.001508069
```

- Estimate the mean squared error on an unseen test set by performing 5-fold crossvalidation. Recall the MSE for a set of y observations and $\hat{y}$ predictions is dened as

$$MSE = \frac{1}{N}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

```
> test_w5 <- train[folds$Fold5,]
> test_w4 <- train[folds$Fold4,]
> test_w3 <- train[folds$Fold3,]
> test_w2 <- train[folds$Fold2,]
> test_w1 <- train[folds$Fold1,]
> X_text_w5 <- test_w5[2:11]
> X_text_w4 <- test_w4[2:11]
> X_text_w3 <- test_w3[2:11]
> X_text_w2 <- test_w2[2:11]
> X_text_w1 <- test_w1[2:11]
> xpred_w5 <- mapply("*",t(beta_w5)[2:11],X_text_w5)
> xpred_w4 <- mapply("*",t(beta_w4)[2:11],X_text_w4)
> xpred_w3 <- mapply("*",t(beta_w3)[2:11],X_text_w3)
> xpred_w2 <- mapply("*",t(beta_w2)[2:11],X_text_w2)
> xpred_w1 <- mapply("*",t(beta_w1)[2:11],X_text_w1)
> xpred_w5 <- cbind(t(beta_w5)[1],xpred_w5)
> xpred_w4 <- cbind(t(beta_w4)[1],xpred_w4)
> xpred_w3 <- cbind(t(beta_w3)[1],xpred_w3)
> xpred_w2 <- cbind(t(beta_w2)[1],xpred_w2)
> xpred_w1 <- cbind(t(beta_w1)[1],xpred_w1)
> y_pred_w5 <- rowSums(xpred_w5)
> y_pred_w4 <- rowSums(xpred_w4)
> y_pred_w3 <- rowSums(xpred_w3)
> y_pred_w2 <- rowSums(xpred_w2)
> y_pred_w1 <- rowSums(xpred_w1)
> ydiff_w5 <- cbind(test_w5[1],y_pred_w5)
> ydiff_w4 <- cbind(test_w4[1],y_pred_w4)
> ydiff_w3 <- cbind(test_w3[1],y_pred_w3)
> ydiff_w2 <- cbind(test_w2[1],y_pred_w2)
> ydiff_w1 <- cbind(test_w1[1],y_pred_w1)
> ydiff_w5$diff <- ydiff_w5$Y - ydiff_w5$y_pred_w5
> ydiff_w4$diff <- ydiff_w4$Y - ydiff_w4$y_pred_w4
```

```
> ydiff_w3$diff <- ydiff_w3$Y - ydiff_w3$y_pred_w3
> ydiff_w2$diff <- ydiff_w2$Y - ydiff_w2$y_pred_w2
> ydiff_w1$diff <- ydiff_w1$Y - ydiff_w1$y_pred_w1
> yd_sq_w5 <- ydiff_w5$diff^2
> yd_sq_w4 <- ydiff_w4$diff^2
> yd_sq_w3 <- ydiff_w3$diff^2
> yd_sq_w2 <- ydiff_w2$diff^2
> yd_sq_w1 <- ydiff_w1$diff^2
> mse_w5 <- sum(yd_sq_w5)/20
> mse_w4 <- sum(yd_sq_w4)/20
> mse_w3 <- sum(yd_sq_w3)/20
> mse_w2 <- sum(yd_sq_w2)/20
> mse_w1 <- sum(yd_sq_w1)/20
> mse_final <- (mse_w5+mse_w4+mse_w3+mse_w2+mse_w1)/5
> mse_final
[1] 0.04863818
```