Predictive Modeling Process

Abbey Chaver and Tina Huang

Introduction

- Credit.csv dataset

Table 1: Table continues below

Income	Limit	Rating	Cards	Age	Education	Gender	Student	Married	Ethnicity	Balance
14.89	3606	283	2	34	11	Male	No	Yes	Caucasian	333
106	6645	483	3	82	15	Female	Yes	Yes	Asian	903

- Want to predict Balance given ten predictors

Methods

Summaries of Variables

- Quantitative
  - Summary table of max, min, range, median, 1st and 3rd quartiles, IQR, mean, and sd
  - Histograms
  - Boxplots
- Qualitative
  - Frequency and Relative Frequency Tables
  - Barplots of Frequencies
  - Conditional Boxplots between Balance and the qualitative variable

Example: Income

Example: Student (yes or no)

Preparing the Data

- Dummy out the categorical variables

```
new_data <- model.matrix(Balance ~ ., data=data)
new_data <- cbind(new_data[ ,-1], Balance = data$Balance)
```

- Mean-center and standardize all variables

```
scaled_data <- scale(new_data, center = TRUE, scale = TRUE)
```

- Divide scaled data into training and test sets

Regression Models

Ordinary Least Squares

- Use `lm()` function
- Then find the coefficients and mse of the model

```
ols = lm(Balance~Income+Limit+Rating+Cards+Age+Education+Ge
+StudentYes+MarriedYes+EthnicityAsian+EthnicityCaucasian, d

ols_coeff = ols$coefficients[-1]

ols_mse = mean(ols$residuals^2)
```

Ridge

- Use glmnet R package
- First perform cross validation on the training set to find the value of lambda that results in the lowest cross validation error

```
grid <- grid <- 10^(seq(10, -2, length = 100))

cv <- cv.glmnet(trainX, trainY, intercept = FALSE, standard
alpha=0)

lambda = cv$lambda.min
```

Ridge: Continued

- Find mean squared error of the best model with this value of lambda on the test set

```
predictedY <- predict(cv, newx = testX, s = "lambda.min")
ssq <- sum((predictedY - testY)^2)
mse = ssq/nrow(testX)
```

- Using that value of lambda, refit the model on the whole dataset to find the official coefficients

```
fcv <- glmnet(dataX, dataY, intercept = FALSE, standardize
alpha=0)
coeff <- coef(fcv)
```

Lasso

- Similar to Ridge, first perform cross validation to find best value of lambda

```
cv_out = cv.glmnet(train_x, train_y, alpha=1, intercept=FAL
lambda=grid)
lasso_lambda = cv_out$lambda.min
```

- Find the mse, then refit the model to find the official coefficients

```
lasso_pred = predict(lasso_mod, s = lasso_lambda, newx=test
lasso_mse = mean((lasso_pred-test_y)^2)
lasso_out = glmnet(scaled_x, scaled_y, alpha=1, lambda=gri
lasso_coeff = predict(lasso_out,type="coefficients",s=lasso
```

Principal Components

- Use pls R package
- Peform 10-fold cross-validation on the training set
- Find the number of components which yields the lowest mse
- Apply the model to the whole dataset with the number of components, and find the official coefficients

```
pcr_model <- pcr(Balance ~ ., data=training, validation = '
test_mses <- c(rep(0, 12))
for (i in 1:12) {
  pcr_pred <- predict(pcr_model, test, ncomp = i)
  test_mses[i] = mean((pcr_pred - testY)^2)
}
which.min(test_mses)
full_pcr_model <- pcr(Balance ~ ., data=data)
best_coefs <- full_pcr_model$coefficients[, , 12][-1]
```

Partial Least Squares

- Similar to Principal Components, use pls package
- Perform 10-fold cross validation to find the best number of components
- Fit the model with this number of components to the whole dataset and find the coefficients

```
pls_fit = plsr(Balance ~ ., data=training_data,scale=TRUE,
pls_comp = which.min(pls_fit$validation$PRESS)

pls_pred = predict(pls_fit, test_x, ncomp=pls_comp)
pls_mse = mean((pls_pred-test_y)^2)

pls_out = plsr(Balance ~ ., data=scaled_data ,scale=TRUE,nc
pls_coeff = pls_out$coefficients[, , pls_comp]
```

Comparison of All Models

Plot of Coefficients

Table of MSEs

Table 3: MSE for Different Models

	OLS	Ridge	Lasso	PCR	PLSR
MSE	0.04479	0.05103	0.04899	0.04926	0.04863

- We can see that ols and pls have the lowest mean squared errors, and would therefore be the best models for predicting Balance

Thanks!