Dimension Reduction

The following code imports the pre-processed data model matrices and responses. It also adds a dummy predictor variable to the test set for prediction function purposes.

```
load("train.mat.JV.RData"); load("test.mat.JV.RData")
load("train.JV.RData"); load("test.JV.RData")

test.mat <- cbind(test.mat, rep(1, nrow(test.mat)))  # create dummy column
colnames(test.mat) <- c(colnames(test.mat)[-1], "SalePrice")
```

The `pls` library has `pcr` and `plsr` functions that can be used to fit principal components regression and partial least squares models. The `Metrics` library has a `rmsle` function that can be used to evaluate error (the same metric that Kaggle uses).

```
library(pls)
library(Metrics)
```

The following code scales the quantitative predictors for use in the dimension reduction functions.

```
train.mat.scale <- train.mat  # initialize object

for (i in 1:ncol(train.mat.scale)) {
  if (is.numeric(train.mat.scale[i])) {
    train.mat.scale[i] <- scale(train.mat.scale[i])  # scale non-factors
  }
}

train.mat.scale <- cbind(train.mat.scale, train$SalePrice)  # need a model matrix that includes
 the response for PCR/PLS
colnames(train.mat.scale) <- c(colnames(train.mat), "SalePrice")
train.mat.scale <- train.mat.scale[,-1]
```

Principal Components Regression

Principal components regression is a combination of using least squares estimation with principal components analysis, which uses linear combinations of the original predictors to reduce the number of predictors (yielding a less variable model).

The following code uses cross-validation to find a good M parameter for PCR, and then writes the prediction to a csv file for submission.

```
pcr.reg <- pcr(SalePrice~., data=as.data.frame(train.mat.scale), scale=F, validation="CV")  # fit PCR regression and use CV
pcr.id <- c(0:ncol(train.mat.scale))[which.min(RMSEP(pcr.reg)$val[1,1,])]  # store value of m to minimize estimated RMSE
pcr.pred <- predict(pcr.reg, newdata=as.data.frame(test.mat), ncomp=pcr.id)  # make prediction

pcr.pred.df <- cbind(Id=test$Id, SalePrice=pcr.pred)
write.csv(pcr.pred.df, "pcr.pred.csv", row.names=F)
```

The following code estimates the RMSLE using 5-fold cross-validation for PCR.

```
set.seed(1)  # consistency of k-fold validation breaks
fold.index <- cut(sample(1:nrow(train.mat.scale)), breaks=5, labels=FALSE)  # split data into 5 folds

pcr.rmslek <- c()  # initialize storage of the k RMSLE's
for (k in 1:5) {
    train.x <- train.mat.scale[fold.index != k,]  # fold training set
    test.x <- train.mat.scale[fold.index == k,]  # fold test set
    true.y <- train$SalePrice[fold.index == k]  # fold test response

    pcr.regk <- pcr(SalePrice~., data=as.data.frame(train.x), scale=F, validation="CV")  # fit PC regression using training data
    pcr.idk <- c(0:ncol(train.mat.scale))[which.min(RMSEP(pcr.reg)$val[1,1,])]  # extract number of components to use
    pcr.predk <- predict(pcr.regk, newx=test.x, M=pcr.idk)  # predict response for test data
    pcr.rmslek <- c(pcr.rmslek, rmsle(actual=true.y, predicted=pcr.predk))  # store the RMSLE metric for this test fold
}

pcr.rmsle <- mean(pcr.rmslek, na.rm=T)  # calculate the average RMSLE
```

Summary of Results

Method	Components	Estimated RMSLE	Actual RMSLE
PCR	161	0.55844	0.19129

Partial Least Squares

Partial least squares is a combination of using least squares estimation with supervised principal components analysis, which uses linear combinations of the original predictors as they relate to the response to reduce the number of predictors (yielding a less variable model).

The following code uses cross-validation to find a good M parameter for PLS, and then writes the prediction to a csv file for submission.

```
pls.reg <- plsr(SalePrice~., data=as.data.frame(train.mat.scale), scale=F, validation="CV")  # f
it pls regression and use CV
pls.id <- c(0:ncol(train.mat.scale))[which.min(RMSEP(pls.reg)$val[1,1,])]  # store value of m to
 minimize estimated RMSE
pls.pred <- predict(pls.reg, newdata=as.data.frame(test.mat), ncomp=pls.id)  # make prediction

pls.pred.df <- cbind(Id=test$Id, SalePrice=pls.pred)
write.csv(pls.pred.df, "pls.pred.csv", row.names=F)
```

The following code estimates the RMSLE using 5-fold cross-validation for PLS.

```
set.seed(1)  # consistency of k-fold validation breaks
fold.index <- cut(sample(1:nrow(train.mat.scale)), breaks=5, labels=FALSE)  # split data into 5
 folds

pls.rmslek <- c()  # initialize storage of the k RMSLE's
for (k in 1:5) {
    train.x <- train.mat.scale[fold.index != k,]  # fold training set
    test.x <- train.mat.scale[fold.index == k,]  # fold test set
    true.y <- train$SalePrice[fold.index == k]  # fold test response

    pls.regk <- plsr(SalePrice~., data=as.data.frame(train.x), scale=F, validation="CV")  # fit
 PLS regression using training data
    pls.idk <- c(0:ncol(train.mat.scale))[which.min(RMSEP(pls.reg)$val[1,1,])]  # extract number
 of components to use
    pls.predk <- predict(pls.regk, newx=test.x, M=pls.idk)  # predict response for test data
    pls.rmslek <- c(pls.rmslek, rmsle(actual=true.y, predicted=pls.predk))  # store the RMSLE me
tric for this test fold
}

pls.rmsle <- mean(pls.rmslek, na.rm=T)  # calculate the average RMSLE
```

Summary of Results

Method	Components	Estimated RMSLE	Actual RMSLE
PLS	26	0.56133	0.18627