Research Task #3:

Based on the information and examples from the Part B required reading assignments from James et al, perform PLSR experiments using the *Hitters* data set. (Please note that you will need to trace some preparations to the data set and definitions of some variables to experiments described in the text before the PLSR lab section.) Using cross-validation, identify an optimal number of PLS components, and then test the optimal model on the test data.

Overview

This task utilizes the Hitters data set from the ISLR library that contains records and salaries of baseball players and is described in Section 6 of James et al. (2013). The experiment is supervised and uses the partial least squares regression learning algorithm to predict the dependent variable, salary. Partial Least Squares Regression (PLSR) is a supervised learning algorithm that can be used for dimensionality reduction, including for predictive modeling. The data set is broken into a training and test set and the training set is used to develop the model.

PLRS is similar to multiple regression and is more suitable when there are many highly correlated variables, which is often the case with high-dimensional data. Multiple regression can predict an expected value of the dependent variable E(y) given an intercept of B_0 and independent variables x_p with slopes of B_p.

$$E(y) = B_0 + B_1x_1 + B_2x_2 + \ldots + B_px_p$$

(quoted from Dziuda, 2024).[1]

The regression line is determined by using training data and comparing the predicted value of response variable to the actual value in the training data (the error). The individual error residuals are then squared and summed, and the regression model will be the hyperplane that minimizes the sum of those squared errors (Dziuda, 2024).[2] While this may be the best model that predicts the dependent variable using the independent variables, it becomes unreliable when many independent variables are highly correlated against other independent variables or groups of independent variables. Partial least squares regression may be preferrable in such situations, because PLRS can perform supervised dimensionality reduction to condense the variables to a few, uncorrelated components (Dziuda, 2024).[3]

In partial least squares regression, we will be deriving components that are specifically constructed to retain most of such information in our independent variables that helps predict our response variable, while at the same time reducing the dimensionality of the regression. This method computes the direction of the first component by calculating coefficients from simple linear regression of the response onto each independent variable and summing their products with the observations for that variable. This means that the variables that are more strongly correlated with the response will have a greater influence on the resulting direction of the first component (Dziuda, 2024).[4] The second PLS component will be orthogonal (uncorrelated) with the first and identify the direction with the second most correlation with the response variable.

---

[1] Pg. 18
[2] Pg. 19
[3] Pg. 19
[4] Pg. 21

With PSLR we can evaluate models by using cross-validation or using a separate set of data, like the test set, to determine how effective the models predicts the dependent variable (James et al., 2013).[5] In cross-validation we choose the model with the number of components that minimizes the mean squared error. The model with this number of components would then be evaluated using the test set as we will illustrate below in this research task.

Note:
I initially struggled with this experiment because many of the results that used random number generator did not result in the same R outputs as the James et al. (2013) textbook. However, I referenced the 2nd edition of the textbook, James et al. (2021) and the new edition agrees with my R results.

I believe that the 2013 edition was using an older version of R. I found a discussion on Stackoverflow.com that provides examples of different random number results between the different versions of R (Bschneidr, 2019)[6] and verified it with the R release notice. The 3.6.0 notice states that the default method for generating from a discrete uniform distribution was changed with that release (Microsoft, 2023).[7]

Experimental Steps
a) Data Exploration & Preparation

The first step in the experiment was to load the data set and view the data. We note that there are 322 observations and 20 variables, including the dependent variable, salary. This is clearly not an example of high-dimensional data, since the number of observations is much more than the number of variables. We also note that most of the variables are continuous, except for League, Division and NewLeague. Furthermore, we note that some of the salary numbers are completed as NA instead of with a number.

R code adapted from page 244 of James et al. (2013):

```
# Load ISLR package
library(ISLR)
# Display Variables of Hitters Data (Columns)
fix(Hitters)
names(Hitters)
# Count rows and columns of Hitters data
dim(Hitters)
```

R Output

```
[1]"AtBat" "Hits" "HmRun" "Runs" "RBI" "Walks"
[7]"Years" "CAtBat" "CHits" "CHmRun" "CRuns" "CRBI"
[13]"CWalks" "League" "Division" "PutOuts" "Assists" "Errors"
[19]"Salary" "NewLeague"

322 20
```

[5] Pg. 374
[6] https://stackoverflow.com/questions/47199415/is-set-seed-consistent-over-different-versions-of-r-and-ubuntu
[7] https://cran.microsoft.com/snapshot/2019-05-02/doc/manuals/r-devel/NEWS.html

We count the number of records missing salary information and remove them from data set before we perform our analysis. We note that 59 NA results initially were present and after processing none were present. The number of records was reduced to 263.

R code adapted from page 244 of James et al. (2013):

```
# Count missing salary amounts (NA)
sum(is.na(Hitters$Salary))
```

R Output:

```
59
```

R code adapted from page 244 of James et al. (2013):

```
# Remove records for players with missing salary
Hitters=na.omit(Hitters)
dim(Hitters)
sum(is.na(Hitters))
```

R Output:

```
263 20
0
```

The least squares criterion will be used to create an m-dimensional regression hyperplane that best fits the predictions of the training data points onto an (m+1) dimensional space (Dziuda, 2024).[8] To enable this analysis, we specify that Salary is the dependent Y variable and create a matrix of the 19 predictor variables. The model.matrix function also changes our three qualitative variables into dummy variables.

R code adapted from page 251 of James et al. (2013):

```
# Use model.matrix function to create matrix for 19 predictors
# Predict Salary on Hitters Data for y vector
x=model.matrix(Salary~.,Hitters)[,-1]
y=Hitters$Salary
```

b) Preparation of Training and Test subsets

We then divide the data set into both a training and test data set by randomly choosing a subset of numbers between the 1 and 263 observations with the random number generator set to 1.

R code adapted from page 253 of James et al. (2013):

```
# Split samples into training and test set by randomly choosing subset between 1 and n
set.seed(1)
train=sample(1:nrow(x), nrow(x)/2)
test=(-train)
y.test=y[test]
```

[8] Pg. 21

We note that with 263 observations in the data set, the train set will include 131 observations (nrow(x)/2), and thus the test set will have 132 observations.

c) Preparation of Partial Least Squares Models & 10-Fold Cross-Validation

We use the PLS function to create 19 PLSR models (with number of PLS components from 1 to 19) using the training data and to calculate 10-fold cross-validation errors for each of the models. We note that the pls.fit function standardizes the predictors as indicated with the "scale=TRUE." This limits the adverse effects of different unit scales (James, 2013).[9]

R code adapted from page 258 of James et al. (2013):

```
# Load pls package and apply to Hitters data while standardizing each predictor
library(pls)
set.seed(1)
pls.fit=plsr(Salary~., data=Hitters,subset=train,scale=TRUE, validation="CV")
# Compute ten-fold cross validation error for each possible number of the components
summary(pls.fit)
```

R Output:

```
Data:   X dimension: 131 19
        Y dimension: 131 1
Fit method: kernelpls
Number of components considered: 19
```

VALIDATION: RMSEP
Cross-validated using 10 random segments.

	(Intercept)	1 comps	2 comps	3 comps	4 comps	5 comps	6 comps	7 comps	8 comps	9 comps
CV	428.3	325.5	329.9	328.8	339.0	338.9	340.1	339.0	347.1	346.4
adjCV	428.3	325.0	328.2	327.2	336.6	336.1	336.6	336.2	343.4	342.8

	10 comps	11 comps	12 comps	13 comps	14 comps	15 comps	16 comps	17 comps	18 comps	19 comps
CV	343.4	341.5	345.4	356.4	348.4	349.1	350.0	344.2	344.5	345.0
adjCV	340.2	338.3	341.8	351.1	344.2	345.0	345.9	340.4	340.6	341.1

TRAINING: % variance explained

	1 comps	2 comps	3 comps	4 comps	5 comps	6 comps	7 comps	8 comps	9 comps	10 comps
X	39.13	48.80	60.09	75.07	78.58	81.12	88.21	90.71	93.17	96.05
Salary	46.36	50.72	52.23	53.03	54.07	54.77	55.05	55.66	55.95	56.12

	11 comps	12 comps	13 comps	14 comps	15 comps	16 comps	17 comps	18 comps	19 comps
X	97.08	97.61	97.97	98.70	99.12	99.61	99.70	99.95	100.00
Salary	5 6.47	56.68	57.37	57.76	58.08	58.17	58.49	58.56	58.62

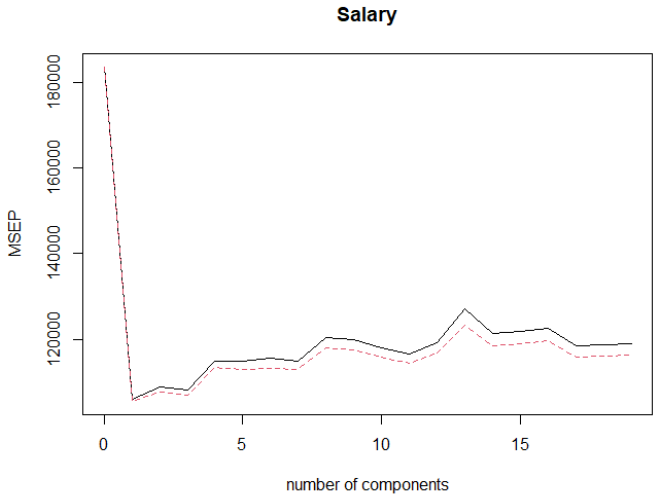
We can see that the root mean square error is minimized for the PLSR model with only one PLS component (highlighted in results above). This gives us a mean squared error of approximately 105,628. One component also explains 46.36% of the variance in salary. This also is clearly apparent if we plot the cross-validation mean squared errors for the number components as shown

[9] Pg. 256

in the R output below. We also note that one component is only marginally better than two or three components, as clearly illustrated in the validation error plot below.

R code adapted from page 258 of James et al. (2013):
# Plot cross-validation scores
validationplot(pls.fit,val.type="MSEP")

R Output:



c) Evaluate Optimal Partial Least Squares Model with Test Set

The optimal PLSR model (with one PLS component) is then evaluated on the test data.

R code adapted from page 258 of James et al. (2013):
# Evaluate corresponding test set MSE
pls.pred=predict(pls.fit,x[test,],ncomp=1)
mean((pls.pred-y.test)^2)

R Output:
151995.3

The mean squared error is higher than that which was observed for the training set, however it is not far out of line. James et al. (2021) indicates that in the prior labs performed in the textbook, ridge regression and lasso also produced comparable mean squared error amounts.[10]

d) Build Partial Least Squares Model on Full Data Set

R code adapted from page 258 of James et al. (2013):
# Perform PLS using full data set using M = 1 components from cross-validation
pls.fit=plsr(Salary~., data=Hitters,scale=TRUE,ncomp=1)

[10] Pg. 282

summary(pls.fit)

R Output:
Data: X dimension: 263 19
Y dimension: 263 1
Fit method: kernelpls
Number of components considered: 1
TRAINING: % variance explained
1 comps
X 38.08
Salary 43.05

1 component also explains 43.05% of variance in salary of the full Hitters set, which is slightly less than the 46.36% salary of the training.

Conclusion

Overall, one PLS component was the optimal number of PLS components identified from our PLSR experiments. One PLS component explains 43.05% of the variance in salary of the full Hitters set. While the PLSR model with one PLS component minimized the mean squared error, we note that there was not a huge difference in the mean squared error for models with one to three components when we performed the cross-validation using the training set. One PLS component was only marginally better than two and three components, as apparent in the cross-validation plot. With such a close result, I would not be surprised if different seeds selected a PLSR model with two or three PLS components simply through the randomness of the 10-fold cross-validation process.

References:

Bschneidr. (2019). Consistency over versions of R: Not necessarily. <https://stackoverflow.com/questions/47199415/is-set-seed-consistent-over-different-versions-of-r-and-ubuntu>

Dziuda, D.M. (2024). DATA 514 – Unit 1: Multivariate analytics lecture notes. CCSU Blackboard LMS.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An introduction to statistical learning with applications in R. Springer 1st ed. 8th corrected printing.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). An introduction to statistical learning with applications in R. Springer 2nd ed.

Microsoft. (2023). R News: Changes in R 3.6.0: Significant user visual changes. <https://cran.microsoft.com/snapshot/2019-05-02/doc/manuals/r-devel/NEWS.html>