Computer Assignment 3 - Matrix Algebra and PCA

Machine Learning, Spring 2020

YOUR NAME

In this assignment, we will explore how to run a few simple (but incredibly useful) linear algebra commands in **R**. Then, we will examine how one calculates principle components using built-in **R** functions.

Transpose and Checking Equality

Given the following 5 vectors, use the `cbind` and `rbind` functions to create the matrices X and Y, where the *rows* of X are x.1-x.5 and the *columns* of Y are x.1-x.5.

```
x.1 = c(2, .5, 4, 2)
x.2 = c(1, 1, 1, 1)
x.3 = c(-1, 0, 2, 1)
x.4 = c(1, -.5, .25, 3)
x.5 = c(3, 6, 9, 12)

# Concatenate the vectors as rows
X = YOUR CODE HERE
# Concatenate the vectors as columns
Y = YOUR CODE HERE
```

We can calculate the transpose of a matrix by using the *t( )* command. For example, try typing *t(X)* and look at the output.

```
YOUR CODE HERE
```

Based on our construction, t(X) = Y. We can check that this is true by using a logical equivalence command ==. When checking the equality of two matrices, the == command will check the entry by entry equality of the two matrices. To do this, the two matrices must be of the same dimensions. Verify that t(X) = Y by using the command *t(X) == Y*. Verify that **R** will return an error if you type *t(x.1) == Y* because *x.1* and Y are not the same dimension.

```
YOUR CODE HERE
```

Matrix Multiplication

Matrix multiplication is performed in **R** using the %*% command. For example, to obtain the product AB of two matrices A and B you can type A%*%B. Calculate the empircial covariance matrix of X using the following code:

```
#calculate the column means of X
col.means = colMeans(X)

#create a matrix of these column means repeated across the rows
X.bar = matrix(rep(col.means,4),ncol = 4, byrow = TRUE)

#column center X
X.col.centered = X - X.bar
```

```
#calculate the empirical covariance matrix of X
sigma = 1/4 * t(X.col.centered) %*% X.col.centered
```

Verify that *sigma* is symmetric using the *t( )* and == commands.

```
YOUR CODE HERE
```

It is important to distinguish the difference between the %*% command and the $*$ command in **R**. The $*$ command is used for scalar multiplication and for entry-wise multiplication of matrices. Knowning this, answer the following questions:

Questions

1. Write out *sigma*, the empirical covariance matrix of X
2. Based on *sigma*, what is $Cov(x.1, x.3)$?
3. What is $X * X$? Note that this is different than what one obtains when typing $t(X)\% * \%X$.

YOUR ANSWERS HERE 1. 2. 3.

Eigenvalues and Eigenvectors

The eigenvalues and eigenvectors of a matrix A can be calculated simply by typing the *eigen(A)* command. The output of this command will contain two components: the eigenvectors and the eigenvalues of the matrix A. Using the *eigen( )* command, answer the following questions:

Questions

1. What are the first and second eigenvectors of *sigma*? What are the first and second eigenvalues of *sigma*?

```
YOUR CODE HERE
```

2. Let *eig.1* be the first eigenvector of *sigma*. Based on the equations given in class, what is the variance of the projections of $x.1, x.2, \ldots, x.4$ onto *eig.1*? What is the relationship between this variance and the first eigenvalue? Without calculations, what do you expect the variance of the projections of $x.1, x.2, \ldots, x.4$ onto the second eigenvector of *sigma* to be?

Norms

The Euclidean norm $||\cdot||$ of a vector can be calculated in **R** using the *norm( )* command. For example, to calculate the norm of the a vector u, we can type *norm(u)*. Note that $||\mathbf{x}||^2$ is the sum of squared distance of entries of $\mathbf{x}$. For example,

```
norm(as.matrix(x.1))
```

The default in **R** is to use the Euclidean norm; however, one can specify other types of norms by using the *type* argument in *norm(u, type =)*. Use *help(norm)* for more information. Answer the following questions:

Questions

1. Project *z.1* onto the first eigenvector of *sigma*. What is the sum of squared distance between *z.1* and its projection (calculated below)?

```
proj = t(z.1)* %*% eig.1 %*% eig.1
YOUR CODE HERE
```

PCA Basics

Running a principal components analysis in **R** is fast and simple, and only makes use of one function which is available to all versions of **R**. The below description is partially taken from the publicly available **R** documentation. This documentation can be seen in full if you type *?prcomp* or *help(prcomp)* in your **R** console.

```
require('stats')
?prcomp
```

prcomp performs a principal components analysis on the given numeric data matrix **x** and returns the following outputs:

Outputs:

- *sdev*: the standard deviations of the principal components.
- *rotation*: the matrix of variable loadings (i.e., a matrix whose columns contain the eigenvectors).
- *x*: the rotated data; i.e., a matrix whose columns are the principal components.
- *center*: the means that were subtracted.
- *scale*: the scalings applied to each variable.

Let's now see PCA in action on a real dataset. We first consider the *iris* dataset available in **R**. Load the data and separate the species names and quantitative data using the following commands:

```
data(iris)

#Separate the species names from the quantitative data
data = iris[,1:4]
species = iris$Species
```

Questions

1. How many variables does this data set contain? What are their names?
2. Why might PCA be helpful for this data set?
3. Run a principal components analysis on the iris dataset. What proportion of the total variation in the data is explained by each of the principal components?

```
#Run principal components analysis
pcs = prcomp(data)

#Summarize the pcs
summary(pcs)
```

YOUR ANSWERS HERE 1. 2. 3.

4. Plot a scree plot of the variance explained:

```
# Make scree plot
screeplot(pcs, type = "lines", main = "Variance explained by PC")
```

Based on the variation explained for each of these components, would you be comfortable projecting the *iris* data set down to only one dimension?

YOUR ANSWER HERE

The PCs

Let's first look at each of these PCs individually. One way to do this is to look at the distribution of the PC scores for each PC. This is a good time to introduce the *for()* loop command. The *for()* loop allows one to sequentially call a command over an index specified in (). For example, try

```
for(i in 1:4){
  print(i)
}
```

We will plot the four pcs using a *for* loop:

```
#Partition the image space into a 2 x 2 grid
par(mfrow = c(2,2))

#Now run the for loop to plot each image separately
#For each plot, we will label the points according to species

for(i in 1:4){
  plot(pcs$x[,i],
       main = paste("PC", eval(i)),
       xlab = "Sample Index",
       ylab = paste("PC", eval(i), "score"),
       col = species)
}
```

Note that the *col = species* command automatically colors the points by the flower species. Here's a quick trick for figuring out which colors correspond to which species.

```
barplot(table(species), col = unique(species))
```

Questions

1. Comment on the four pc plots. Do any of the plots reveal any interesting structure?
2. Do you think that you could distinguish the species of plant if you projected the original data onto PC4? How about PC1?

YOUR ANSWERS HERE 1. 2.

Biplots

Let's see if we have any further evidence of clustering in the data by looking at a biplot of the PCs. That is, we will plot the scores of the first PC against the scores of the second. In this way, we are illustrating a 2-dimensional view of this 4-dimensional data. Create a biplot with labeled points using the following code:

```
plot(pcs$x[,1:2],
     main = "Biplot of Iris Data",
     xlab = "PC 1", ylab = "PC 2",
     col = species)
```

Next, let's plot a pairwise biplots for each pair of pcs. Do this by using the *pairs()* command as follows:

```r
pairs(pcs$x, col = species)
```

Questions

1. What features do you notice from the resulting biplot? Do the data appear to "cluster?"
2. Compare your pairwise plots to the plot of PC 1 alone. Does there appear to be any added benefit in projecting the data into more than one direction? In conclusion, to how many dimensions can we project this data and still capture the features of the species?

YOUR ANSWERS HERE 1. 2.