# Stat 432 Homework 4

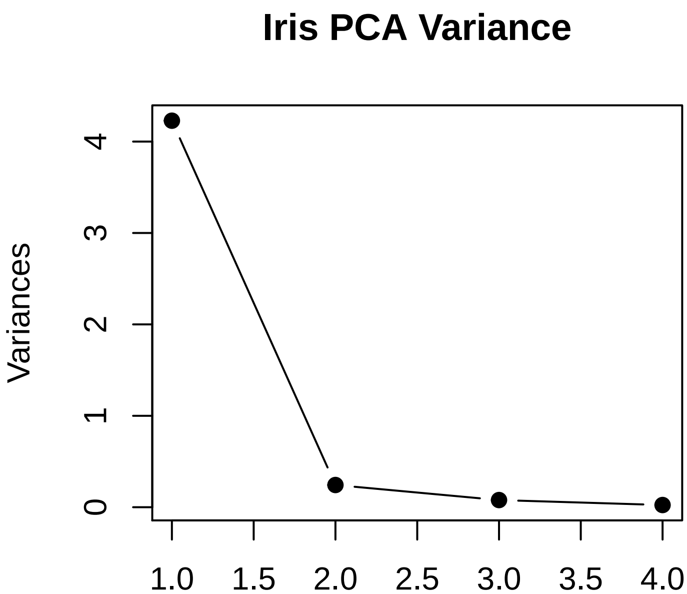
*Assigned: Sep 20, 2018; Due: Sep 28, 2018*

Question 1 (more on PCA)

[3 points] Take the first four columns of the `iris` data to verify the connection of PCS with the singular value decomposition (`svd`).

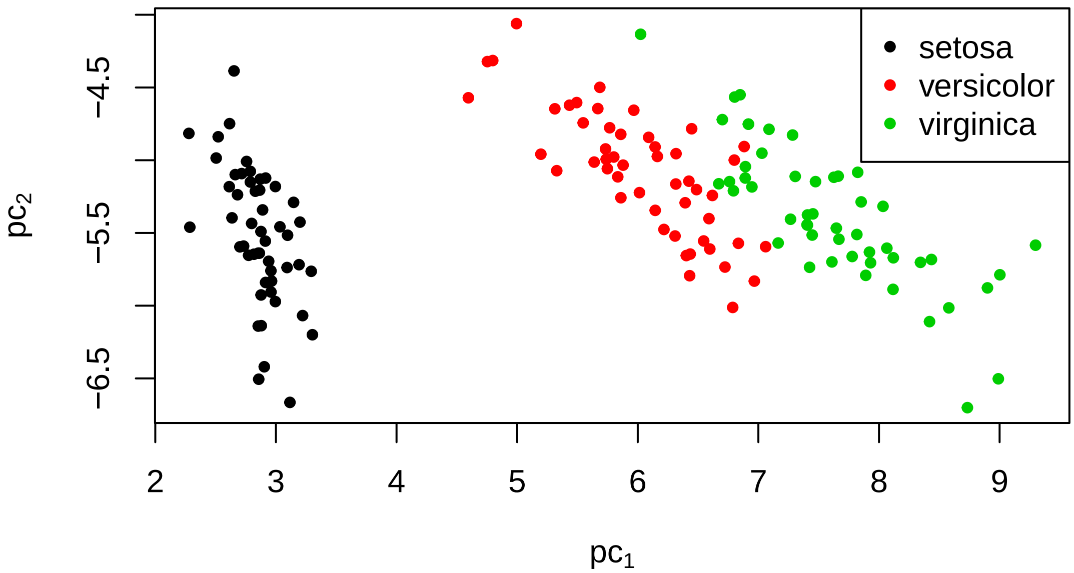
Load `iris` data. We use `svd` function to obtain the singular values $\mathbf{D}$. Then PCA variance is the eigenvalues of $\widehat{\Sigma} = \mathbf{X}^{\mathrm{T}}\mathbf{X}/(n-1)$, which is $D^2/(n-1)$, and we plot them in the decreasing order. (1 point)

```
# load 'iris'
data(iris)
n=nrow(iris)
# SVD
irissvd=svd(scale(iris[, 1:4], center = TRUE, scale = FALSE))
plot(irissvd$d^2/(n-1), type = 'b', pch = 19, xlab ='', ylab= 'Variances', main = "Iris PCA Variance")
```




Now we obtain the principal components by $\mathbf{X}^{\mathrm{T}}\mathbf{V}$ and plot the first two on a figure with colors associated with species. (1 point) And we print the rotation matrix $\mathbf{V}$. (1 point)

```
# principal components
irispc=as.matrix(iris[, 1:4])%*%irissvd$v
plot(irispc[,1], irispc[,2], pch=20, col=c(1:3)[iris$Species], xlab=expression(pc[1]), ylab=expression(
legend('topright',pch=20,legend=unique(iris$Species),col=c(1:3))
```




```
# rotation matrix
irissvd$v
```

```
##             [,1]        [,2]        [,3]       [,4]
## [1,]  0.36138659 -0.65658877  0.58202985  0.3154872
## [2,] -0.08452251 -0.73016143 -0.59791083 -0.3197231
## [3,]  0.85667061  0.17337266 -0.07623608 -0.4798390
## [4,]  0.35828920  0.07548102 -0.54583143  0.7536574
```

Question 2 ($k$-NN for classification)

[4 points] Consider again the zip code digits data. And we will use the Eucleadian distance. We want to predict the digit of the 4th observation in the testing dataset.

```
    library(ElemStatLearn)
    train.x = zip.train[, -1]
    train.y = as.factor(zip.train[, 1])
    test.x.one = zip.test[4, -1]
```

One way to find 15 nearest neighbors using`sweep` and `rowSums` could be as follows: (1 point)

```
k=15
# find k nn
(idx_knn=which(rank(rowSums((sweep(train.x, 2, test.x.one, FUN = "-"))^2)) <= k))
```

```
##  [1]  389  521 1619 1825 2240 3471 3988 4187 4188 5106 5143 5198 6450 6774
## [15] 6976
```

Then we can find the most frequent digit among these 15 observations. We find out that the majority vote is “0”, not the true label “6”. (1 point)

```
# majority vote
(major_vote=names(which.max(table(train.y[idx_knn]))))
```

```
## [1] "0"
```

```
# true label
(test.y.one=zip.test[4,1])
```

```
## [1] 6
```

Now we repeat the process for a sequence of $k$ and list the results as follows (1 point. No need to list for multiple $k$'s for this ste; you get full credict as long as you get correct prediction for some $k$.)

```
# try different k's
pred=rep(NA,15)
for(k in 1:15){
  idx_knn=which(rank(rowSums((sweep(train.x, 2, test.x.one, FUN = "-"))^2)) <= k)
  major_vote=names(which.max(table(train.y[idx_knn])))
  pred[k]=as.numeric(major_vote)==test.y.one
}
# print out results
names(pred)<-1:15
pred
```

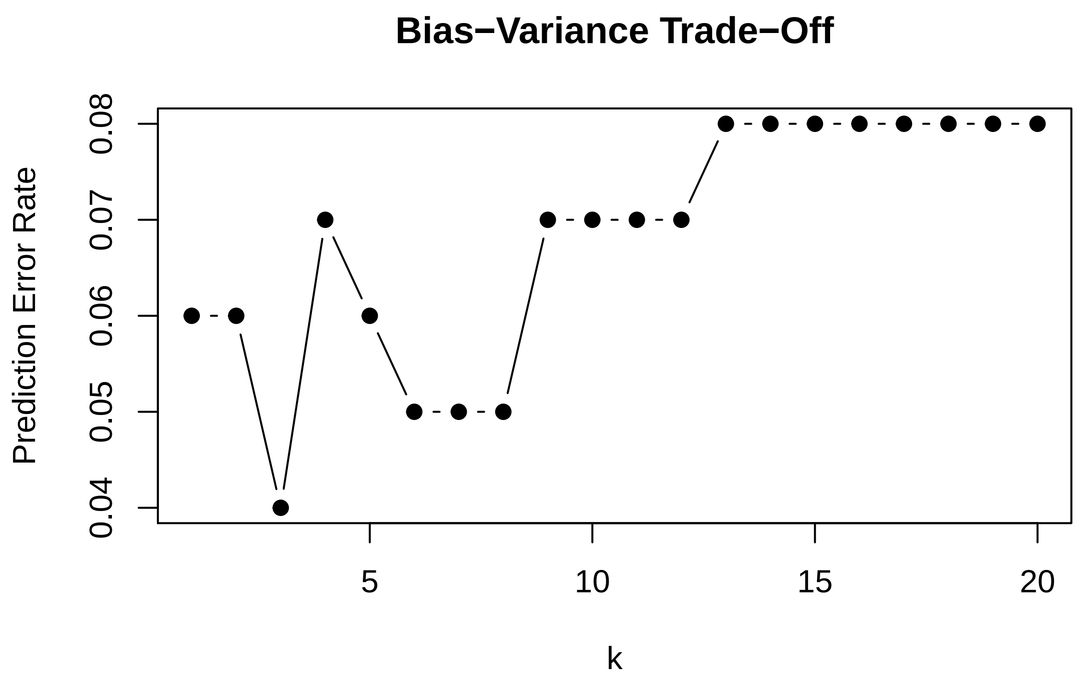
```
##     1     2     3     4     5     6     7     8     9    10    11    12
##  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE
##    13    14    15
## FALSE FALSE FALSE
```

For this given testing case, we see that with small $k$, it happens to predict the correct labels though it fails at $k = 4$. With increasing $k$, it gets correct prediction until $k > 10$. Now we repeat the process for the first 100 observations with $k$ ranging from 1 to 20, and plot the error rate wrt $k$.

```
# pairwise distance function
# cited from https://www.r-bloggers.com/pairwise-distances-in-r/
pdist <- function(A,B) {
  an = apply(A, 1, function(rvec) crossprod(rvec,rvec))
  bn = apply(B, 1, function(rvec) crossprod(rvec,rvec))

  m = nrow(A)
  n = nrow(B)

  tmp = matrix(rep(an, n), nrow=m)
  tmp = tmp +  matrix(rep(bn, m), nrow=m, byrow=TRUE)
  # sqrt( tmp - 2 * tcrossprod(A,B) )
  tmp - 2 * tcrossprod(A,B)
}
# obtain the first 100 observations as testing data
test.x.hundred = zip.test[1:100, -1]
test.y.hundred = zip.test[1:100, 1]
# repeat for k from 1 to 20
errate=rep(NA,20)
dist_trte=pdist(train.x,test.x.hundred)
for(k in 1:20){
  idx_knn=apply(dist_trte,2,function(d) which(rank(d)<=k))
  if(k==1){idx_knn=t(idx_knn)}
  major_vote=apply(idx_knn,2,function(id)names(which.max(table(train.y[id]))))
  errate[k]=mean(as.numeric(major_vote)!=test.y.hundred)
}
# plot the error rate
plot(errate,type = 'b', pch = 19, xlab =expression(k), ylab= 'Prediction Error Rate', main = "Bias-Vari
```




Seen from the resutls, the best prediction result (lowest error rate) is attained at $k = 3$. (1 point. You can also come up other criterion for selecting the best $k$. You lose 0.5 points if not reporting the best $k$ as the final answer.)

Question 3 (Cross-validation using the `caret` package)

[3 points]

Install and load the `caret` package. We use it to do CV (1 point) to choose the best $k$. And then we test the best fit on testing data.

```
# load 'caret' which requires the package 'e1071' that needs to be installed
library(caret)
# prepare for the training
TrainData = data.frame(x = train.x)
TrainClasses = as.factor(train.y)
# train the knn model for classification using CV, taking several minutes
knnFit <- train(TrainData, TrainClasses, method = "knn", tuneGrid = data.frame("k" = c(1:10)),
                trControl = trainControl(method = "cv", number = 3))
# best k
knnFit$bestTune
```

```
##   k
## 1 1
```

```
# prepare for the testing
TestData = data.frame(x = zip.test[,-1])
TestClasses = as.factor(zip.test[,1])
# test
# predy=extractPrediction(list(knn=knnFit), testX = TestData, testY = TestClasses, unkX = TRUE)
predy=predict(knnFit, newdata=TestData)
# confusion matrix
table(predy,TestClasses)
```

```
##      TestClasses
## predy   0   1   2   3   4   5   6   7   8   9
##     0 355   0   6   3   0   2   0   0   5   0
```

```
##      1   0 255   1   0   3   1   0   1   0   0
##      2   2   0 183   2   1   2   1   1   1   1
##      3   0   0   2 154   0   4   0   1   6   0
##      4   0   6   1   0 182   0   2   4   1   2
##      5   0   0   0   5   1 145   3   0   1   0
##      6   0   2   0   0   2   2 164   0   0   0
##      7   1   1   2   0   2   0   0 139   1   4
##      8   0   0   3   0   1   3   0   0 148   1
##      9   1   0   0   2   8   1   0   1   3 169
```

The best $k = 1$ (1 point), which usually tends to overfit data. This may be due to the small number folds. The confusion matrix (1 point) shows it works well in prediction.