Modern Statistical Prediction and Machine Learning

Homework 4 (due March 24): Linear Regression, Classification

Theoretical Exercise

Problem 1: Basic properties of PCA

Suppose we have a sample of N vectors in $\mathbb{R}^p$, denoted by $x = (x_1, ..., x_p)$. We want to perform PCA on this set.

- Define that sample covariance matrix Σ_N and show that it is positive semi-definite.

  The sample covariance matrix Σ_N is

  $$\Sigma_N = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})(x_i - \bar{x})^T$$

  where $\bar{x}$ is the sample mean vector.

  So, let $v \in \mathbb{R}^p$ be a non-zero vector. Then,

  $$v^T\Sigma_N v = v^T\left(\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})(x_i - \bar{x})^T\right)v = \frac{1}{N-1}\sum_{i=1}^{N}((x_i - \bar{x})^T v)^2.$$

  Since in the last equality, the square of a real number is always non-negative. It follows that $v^T\Sigma_N v \geq 0$ for any non-zero vector v. Therefore, Σ_N is positive semi-definite.

- Let $\lambda_1, ..., \lambda_p$ denote the eigenvalues of Σ_N , and let $v_1, v_2, ..., v_p$ denote the corresponding unit norm eigenvectors. Prove that the eigenvectors are orthogonal, i.e., $\langle v_i + v_j \rangle = 0$ if $i \neq j$.

  To prove this, let's consider two distinct eigenvectors v_i and v_j with corresponding eigenvalues λ_i and λ_j. Since v_i and v_j are eigenvectors of Σ_N, we have:

  $$\Sigma_N v_i = \lambda_i v_i$$

  and

  $$\Sigma_N v_j = \lambda_j v_j$$

  Taking the transpose of the first equation and multiplying both sides by v_j, we get:

  $$v_j^T\Sigma_N v_i = \lambda_i v_j^T v_i$$

  Similarly, taking the transpose of the second equation and multiplying both sides by v_i, we get:

  $$v_i^T\Sigma_N v_j = \lambda_j v_i^T v_j$$

  Since Σ_N is symmetric, we have $\Sigma_N^T = \Sigma_N$, so $v_j^T\Sigma_N v_i = v_i^T\Sigma_N^T v_j = v_i^T\Sigma_N v_j$. Therefore, we can equate the right-hand sides of the two equations above to get:

  $$\lambda_i v_j^T v_i = \lambda_j v_i^T v_j$$

  Since $\lambda_i \neq \lambda_j$, this implies that $v_j^T v_i = 0$, which means that the eigenvectors are orthogonal.

- Let $k < p$ be an integer. Show that the first k principal components capture at least as much variance as any other k-dimensional subspace of the data. More formally, prove that for any k-dimensional subspace V of the data, the sum of the variances of the data projected onto V is less than or equal to the sum of the variances of the data projected onto the first k principal components.

  Let X be an $n \times p$ data matrix with n observations and p variables. Assume that the X is centered, i.e., the mean of each variable is 0:

  $$Var(X) = tr(X^T X)$$

  Let V be a k-dimensional subspace of the data, and let P_V be the projection matrix onto V. Then the total variance of the data projected onto V is given by:

  $$Var(XP_V) = tr(P_V^T X^T X P_V)$$

  Let U be the $p \times k$ matrix whose columns are the first k principal components of X. The variance of the data projected onto the first k principal components is given by:

  $$Var(XU) = tr(U^T X^T X U)$$

  U is chosen to maximize this variance, because principal components are the directions that capture the maximum variance, the sum of the variances of the data projected onto the first k principal components is greater than or equal to the sum of the variances of the data projected onto any other k-dimensional subspace, we have:

  $$Var(XU) >= Var(XP_V)$$

  This proves that the sum of the variances of the data projected onto the first k principal components is greater than or equal to the sum of the variances of the data projected onto any k-dimensional subspace V.

- Suppose we have a new observation x_0 that we want to project onto the first k principal components. Explain how we can obtain the projected vector y_0 using the eigenvectors and eigenvalues of the sample covariance matrix S.

  1. Center the observation x_0 by subtracting the sample mean vector from it.
  2. Compute the k eigenvectors corresponding to the k largest eigenvalues of the sample covariance matrix S, where the eigenvectors are the first k PCs.
  3. Form a matrix P_k by concatenating the k principal components column-wise.
  4. Project the centered observation onto the k principal components by computing the dot product of the transpose of P_k with the centered observation: $y_0 = P_k^T(x_0 - \bar{X})$

  This give the k-dimensional vector y_0 that represents the projection of x_0 onto the first k principal components.

Problem 2: Kernels and feature transformation

- Is there a transformation from $\mathbb{R} \to \mathbb{R}^2$ that makes these points linearly separable?

  Yes. We can use $K(x) = x^2$

- Is there a transformation from $\mathbb{R} \to \mathbb{R}^2$ that makes these points linearly separable?

  No.

- Consider the following sample. Is there a transformation from $\mathbb{R}^2 \to \mathbb{R}^2$ that makes this set linearly separable?

  Yes. $K(x, y) = x^2 + y^2$

- Is the linear separation possible for some mapping from $\mathbb{R}^2 \to \mathbb{R}$?

  No, because that would be a dimensionality reduction. If the data is not linearly separable in higher dimension, then it would not be linearly separable in lower dimension.

- How to linearly separate the following set? What is the corresponding kernel?

$$K(x, x') = \exp(-\frac{||x - x'||^2}{2\sigma^2})$$

Problem 3: k-means clustering and Lloyd's algorithm

- Conceptual Exercise 1 in Section 12.6 of ISL book

  (a) Prove

$$\frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{i,j} - x_{i',j})^2 = 2 \sum_{i \in C_k} \sum_{j=1}^{p} (x_{ij} - \bar{x}_{kj})^2$$

  where $\bar{x}_{kj} = \frac{1}{|C_k|} \sum_{i \in C_k} x_{ij}$ is the mean for feature j in cluster C_k.
  Expand the left-hand side:

$$\frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{i,j}^2 - 2x_{i,j}x_{i',j} + x_{i',j}^2)$$

  Distribute the summation:

$$\frac{1}{|C_k|} \left(\sum_{i,i' \in C_k} \sum_{j=1}^{p} x_{i,j}^2 - 2 \sum_{i,i' \in C_k} \sum_{j=1}^{p} x_{i,j}x_{i',j} + \sum_{i,i' \in C_k} \sum_{j=1}^{p} x_{i',j}^2 \right)$$

  Substitute the mean for feature j in cluster C_k: $\bar{x}kj = \frac{1}{|C_k|} \sum i \in C_k x_{ij}$.
  For the first term, we have:

$$\sum_{i,i' \in C_k} \sum_{j=1}^{p} x_{i,j}^2 = |C_k| \sum_{i \in C_k} \sum_{j=1}^{p} x_{ij}^2$$

  For the third term, we have:

$$\sum_{i,i' \in C_k} \sum_{j=1}^{p} x_{i',j}^2 = |C_k| \sum_{i \in C_k} \sum_{j=1}^{p} x_{ij}^2$$

  For the second term, we have:

$$-2 \sum_{i,i' \in C_k} \sum_{j=1}^{p} x_{i,j}x_{i',j} = -2 \sum_{i \in C_k} \sum_{j=1}^{p} x_{ij} \sum_{i' \in C_k} x_{i',j} = -2|C_k| \sum_{i \in C_k} \sum_{j=1}^{p} x_{ij}\bar{x}_{kj}$$

  Substitute terms back into the equation:

$$\frac{1}{|C_k|} \left(|C_k| \sum_{i \in C_k} \sum_{j=1}^{p} x_{ij}^2 - 2|C_k| \sum_{i \in C_k} \sum_{j=1}^{p} x_{ij}\bar{x}kj + |C_k| \sum i \in C_k \sum_{j=1}^{p} x_{ij}^2 \right)$$

Simplify and cancel out $|C_k|$:

$$2\sum_{i\in C_k}\sum_{j=1}^{p}x_{ij}^2 - 2\sum_{i\in C_k}\sum_{j=1}^{p}x_{ij}\bar{x}_{kj}$$

Factor out the 2:

$$2\left(\sum_{i\in C_k}\sum_{j=1}^{p}x_{ij}^2 - \sum_{i\in C_k}\sum_{j=1}^{p}x_{ij}\bar{x}_{kj}\right)$$

Now, we get the right-hand side of the equation:

$$2\sum_{i\in C_k}\sum_{j=1}^{p}(x_{ij}-\bar{x}_{kj})^2$$

(b) On the basis of this identity, argue that the K-means clustering algorithm (Algorithm 12.2) decreases the objective (12.17) at each iteration.

This identity relates to minimizing the within-cluster variation in step (2b) of the K-means algorithm (Algorithm 12.2). In this step, a new centroid is just updated from step (a), we minimize the variation by calculating the Euclidean distance between each sample into the centroids and assigning the sample to the centroid that has the smallest distance. Thus, the clusters with its respective sample points in that clusters will have the smallest within-cluster variation.

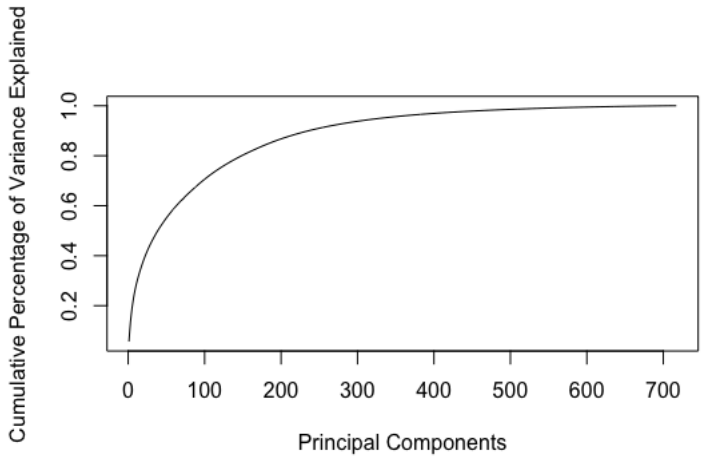
Problem 4: PCA on MNIST

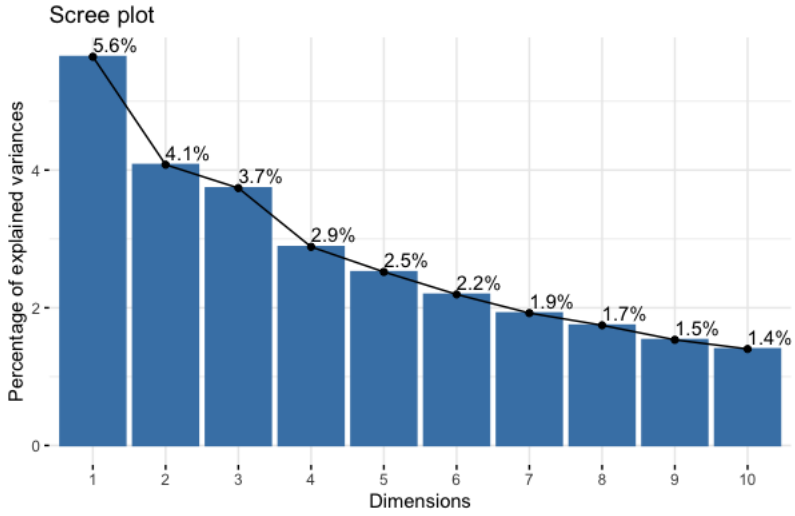
Download the MNIST training data from the class's drive folder.

1. Perform a PCA on the training data. Remember to center and scale the data before perform- ing a PCA.

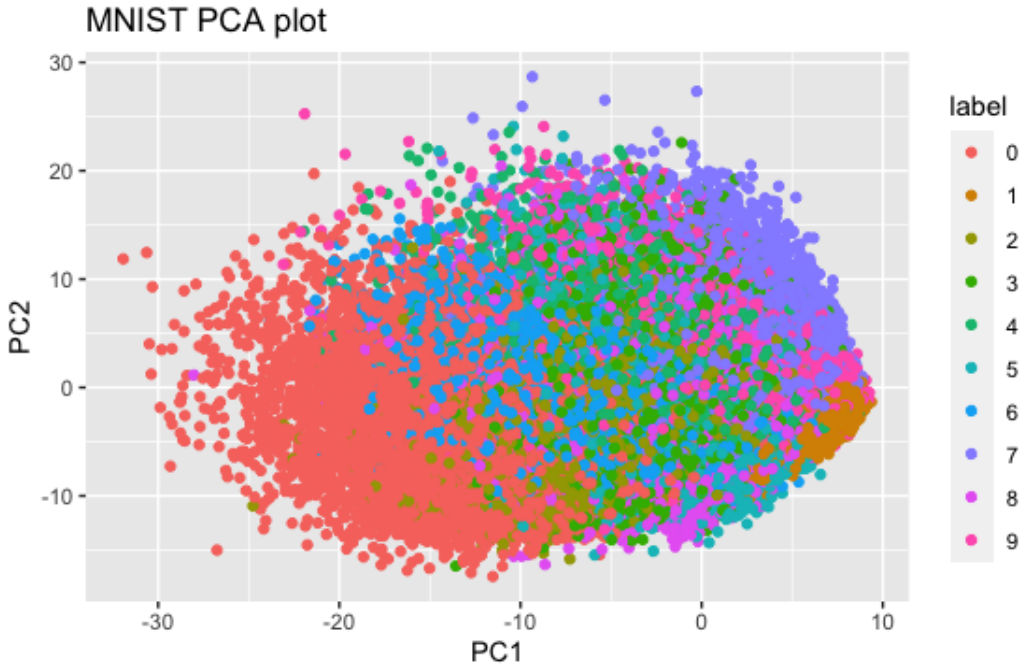
   In R code attached.

2. Plot the variances explained by the different principal components through scree plots as discussed in the lab. Also, plot the cumulative percentage of variance explained against the principal components.



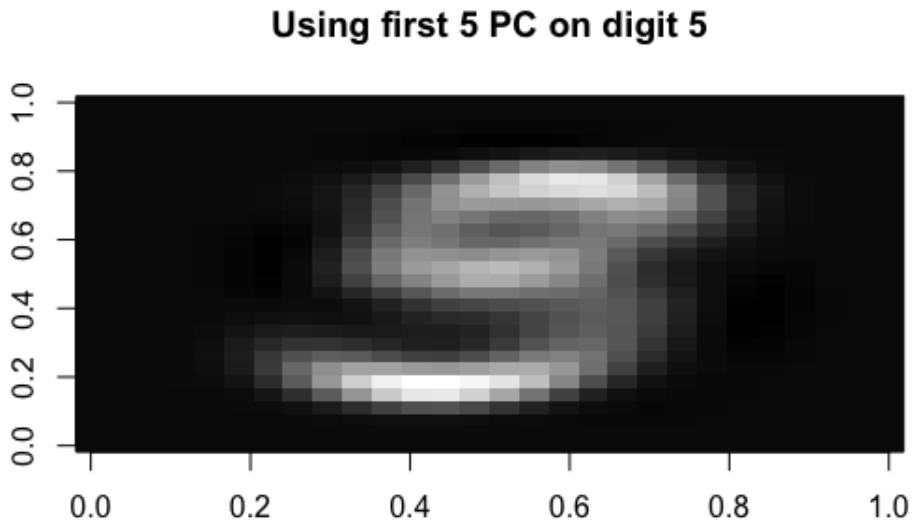


3. Visualize the data in two dimensions color-coding with the class labels. Explain any pattern you see.



The data points are separated into different sections, it means that the principal components are capturing meaningful differences between the classes. However, the separate is not distinct between in class, the overlapping means that using only two principal components will only give a good over picture of class separation, but there will certainly be ambiguous cases between the classes. Overall, we think the first two PC did well, we are keeping in mind that PCA does not have the class information.

4. Plot the first five principal components as gray-scale images. Comment on any pattern you see.

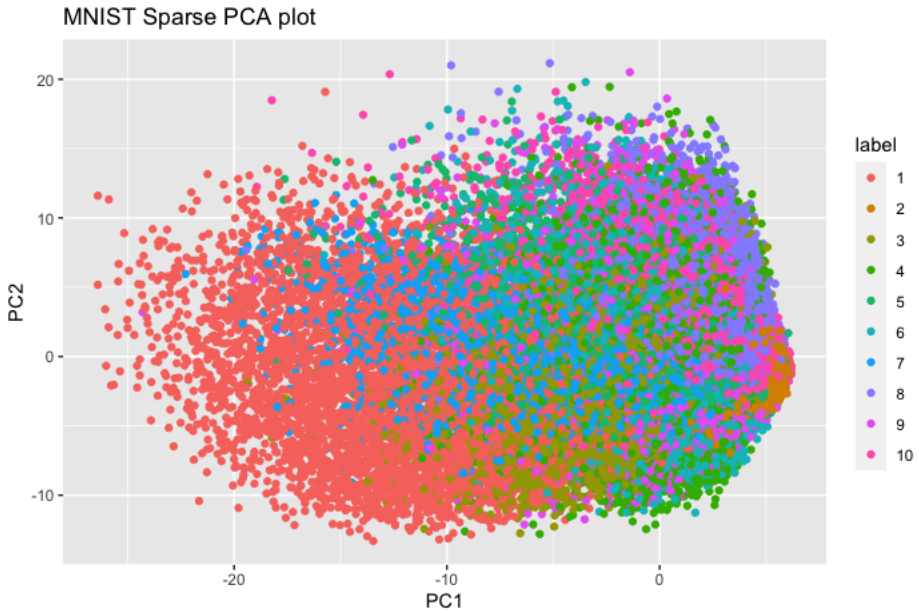


The first 5 PCs capture basic edges/outlines of digits. We can also see the width of digits, and slant/tilt, we see some intricate patterns within digits and some finer details.

5. How many principal components are required to explain 70, 80, 90, and 95% of the variance?

   For 70%, it requires 98 PCs. For 80%, it requires 149 PCs. For 90%, it requires 236 PCs.

6. Plot the data in two dimensions after applying kernel PCA and sparse PCA with your choice of kernel and corresponding hyperparameter.



Problem 5: Classification on MNIST using SVMs

In the previous home assignment, you built different classification models on MNIST data. In this exercise,

we will tune SVMs using a 10-fold cross-validation on MNIST. Download the training and the test data from the classes drive folder.

1. Perform a PCA on the entire data (by first combining the training and test data) and reduce the dimensions of the data that explain 90% of the total variance. Remember to center and scale the data before performing a PCA.

   238 PCs

2. Train the SVM using the radial basis functions

sigma	C	Accuracy	Kappa	AccuracySD	KappaSD
0.004689845	0.25	0.9325997	0.9250865	0.003643884	0.004049874
0.004689845	0.50	0.9463330	0.9403502	0.003540210	0.003934797
0.004689845	1.00	0.9567165	0.9518912	0.002794284	0.003105768

3. Train an SVM with a polynomial kernel

   See R code provided.

4. Compare the performance of the two models on the test set by computing the accuracy on the test set.

   SVM was definitely more computationally efficient.

Problem 6: Clustering on Libras Movement

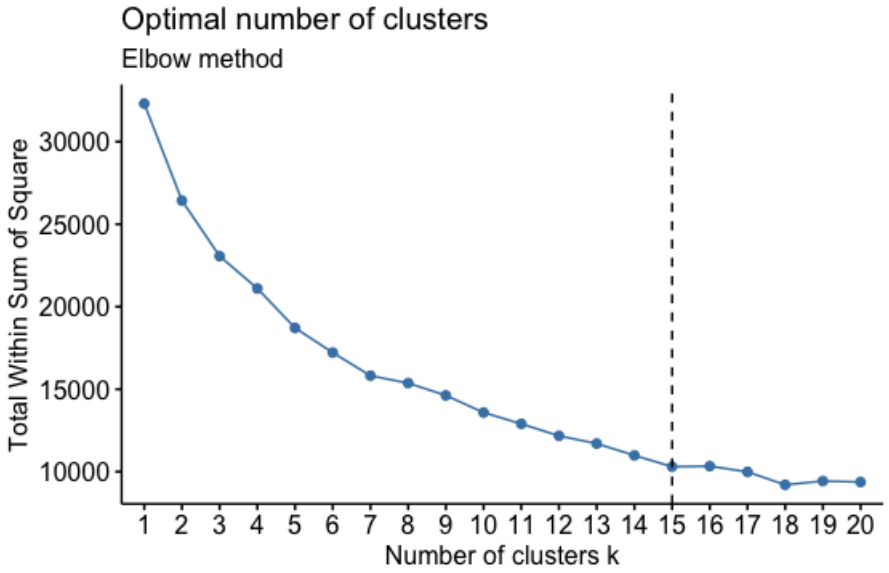
1. Perform a k-means clustering on the data after centering and scaling with k = 15. How sensitive is k-means on this data? Repeat the experiment 100 times and report the mean and standard deviation of the clustering obtained by k-means in terms of Adjusted Rand Index (ARI) and Normalized Mutual Information (NMI) between the ground truth and the obtained partition.

   Adjusted Rand Index: Mean = 0.3130 SD = 0.0091

   Normalized Mutual Information: Mean = 0.5866 SD = 0.0092

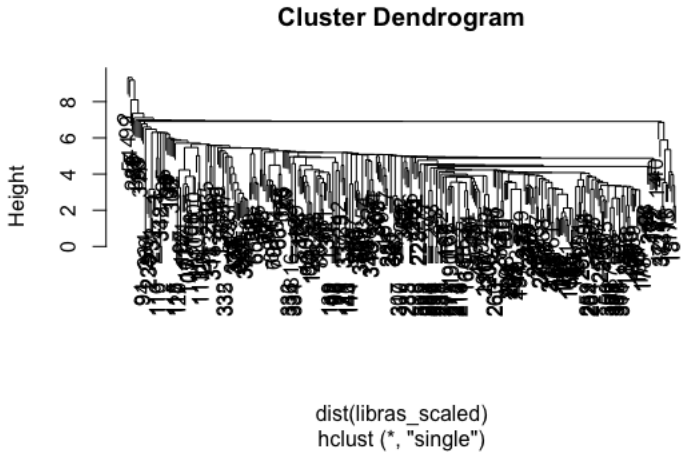
   Because of intial different cetroid initilization, the k-means is known to be sensitive to the initial starting points, and thus, repeating the experiment multiple times can help us obtain more stable results. In our case, the ARI and NMI values give us an idea of how well the obtained partition matches the true labels of the data. A higher value of ARI and NMI indicates a better clustering result. Since our ARI is 0.3130 and MMI is 0.5886, we say that the clusters are not every well separated.

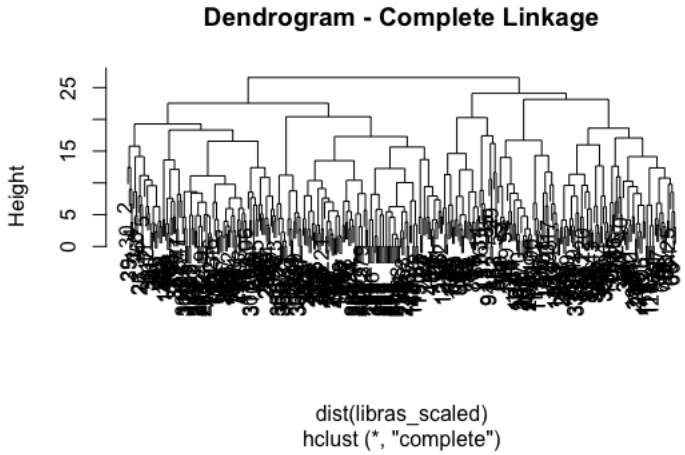
2. Suitably select k by the elbow method. Is your selection close to 15 or something very different?



The resulting plot shows a steep decrease in the WCSS up to around 15 clusters, after which the decrease becomes more gradual. In fact, 15 and 16 have the same WCSS.

3. Perform a hierarchical clustering on the data using single and complete linkages and plot the dendrograms. Make a suitable cut and report the performance of the methods in terms of ARI and NMI.





The ARI and NMI values for the single linkage method were 0.002 and 0.064, respectively, while for the complete linkage method, they were 0.290 and 0.561, respectively

4. 4. Repeat part 1 of the experiment after reducing the dimensions of the data to explain 80% of the variation in the data via PCA.

   Adjusted Rand Index = 0.264

   Normalized Mutual Information = 0.53

   Comparing to the earlier simulation of ARI=0.3130 and NMI=0.5866. Both ARI and NMI has decreased slightly using PCA. This could be that PCA is only extracting and retaining high variations, and leave out some information. So, parition may not be as good because we have less information to work with.

R code

```
############################
# Problem 4: PCA on MNIST
############################
# loading library
```

```
library(factoextra)
library(ggfortify)
library(reshape2)
library(SDEFSR) # read keel data
library(aricode) # adjusted rand index, normalized mutual information

# import data
train <- read.csv("mnist_train.csv")
test <-read.csv("mnist_test.csv")

X_train <- train[, -1]
idx <- which(colSums(X_train)==0)
X_reduced <- X_train[,-idx] # dropping columns with only zeros inside
label <- train[, 1]
label <- as.factor(label)

#################
# 1. Perform PCA
#################
pca <- prcomp(X_reduced, center = TRUE, scale = TRUE)
pca_summary <- summary(pca)

#########################
# 2. Plot the variances
#########################
variance_explained <- cumsum(pca$sdev^2/sum(pca$sdev^2))
plot(variance_explained, type = "l", xlab = "Principal_Components",
     ylab = "Cumulative_Percentage_of_Variance_Explained")

fviz_eig(pca, addlabels = TRUE)

#########################
# 3. Visualize the data in two dimensions color-coding
#########################
library(ggfortify)
scores <- as.data.frame(pca$x[,1:2])
scores <- cbind(scores, label)
scores$label <- as.factor(scores$label)
ggplot(scores, aes(x = PC1, y = PC2, color = label)) +
  geom_point() +
  labs(title = "MNIST_PCA_plot", x = "PC1", y = "PC2")

# Explain any patterns your see.
# The data points are separated into different sections, it means that the principal compo
# meaningful differences between the classes. However, the separate is not distinct betwee
# the overlapping means that using only two principal components will only give a good ove
# but there will certainly be ambiguous cases between the classes. Overall, we think the f
# we are keeping in mind that PCA does not have the class information.


#################################################
# 4. Plot the first five PC as gray-scale image
#      and comment
#################################################
```

```
# calculate total variance explained by each principal component
df_pca <- data.frame(t(pca_summary$importance))

# with reconstruction
n_comp = 5
recon <- pca$x[, 1:n_comp] %*% t(pca$rotation[, 1:n_comp])
recon <- scale(recon, center = FALSE, scale = 1/pca$scale)
recon <- scale(recon, center = -1 * pca$center, scale = 1/pca$scale)
# recon_df <- data.frame(cbind(1:nrow(recon), recon))
# colnames(recon_df) <- c("x", 1:(ncol(recon_df)-1))
recon <- as.data.frame(recon)
zero_cols <- setdiff(names(X_train), names(recon))
# as logical vector: True or False
# zero_cols <- names(X_train) %in% names(recon)
recon[, c(zero_cols)] <- 0
# reorder the columns to the same as original data
# before displaying image
recon <- recon[, colnames(X_train)]
# check if we did it correctly
all(names(recon) == names(X_train)) # yes

# plot the reconstruction
r_samp = 1
image(matrix(as.matrix(recon[r_samp,]), nrow = 28, ncol = 28)[, 28:1],
      col = gray((0:255)/255), main = paste0("Using first 5 PC on digit ", label[r_samp]))

# Comment
# The first 5 PCs seem to capture basic edges/outlines of digits
# It captures width of digits, and slant/tilt, we can see
# some intricate patterns within digits along with some finer details


# 5. How many principal components are required to explain 70, 80, 90, and 95% of the varia
variance_explained <- cumsum(pca$sdev^2/sum(pca$sdev^2))
required_var <- c(0.7, 0.8, 0.9)
for (i in required_var){
  num_components <- which(variance_explained >= i)[1]
  print(paste0("For ", i*100, "% it requires ", + num_components, " PCs."))
}
# for component
# num_components <- which(variance_explained >= 0.9)[1]


# 6. Plot the data in two dimensions after applying
# kernel PCA and sparse PCA with your choice of kernel
# and corresponding hyperparameter.
library(kernlab)
library(sparsepca)
# terminology for self-reference
# scores: The scores in PCA refer to the transformed data points that have been
# projected onto the principal components (PCs). These transformed data points
# represent the new coordinate system that is defined by the PCs. The scores are
# sometimes referred to as the "principal component scores."
# loading: The loading object contains the loadings (or weights) for each variable
```

```
# (or feature) in the original data set, with respect to each principal component.
# The loadings represent the contribution of each variable to the corresponding principal
# component. Each row in the loading matrix corresponds to one variable in the original da
# and each column corresponds to one principal component.
# x: The x object is the centered and scaled data matrix that is used as input to the PCA.
# This matrix contains the observations (rows) and variables (columns) of the original dat

# library(usethis)
# usethis::edit_r_environ()
# R_MAX_VSIZE=5Gb

# kernel PCA
X_small <- X_reduced[1:3000, ]
kpc <- kpca(~.,data=X_small, kernel="rbfdot", kpar=list(sigma=0.2))
plot(rotated(kpc), col=as.factor(label[1:2000]),
     xlab="1st Principal Component",ylab="2nd Principal Component")

## sparse PCA
pca_sparse <- spca(X_reduced, center = TRUE, scale = TRUE)
scores <- pca_sparse$scores[, 1:2]
scores <- cbind(scores, label)
scores <- as.data.frame(scores)
names(scores) <- c("PC1", "PC2", "label")
scores$label <- as.factor(scores$label)
ggplot(scores, aes(x = PC1, y = PC2, color = label)) +
  geom_point() +
  labs(title = "MNIST Sparse PCA plot", x = "PC1", y = "PC2")


################################################
## Problem 5: Classification on MNIST using SVMs
################################################
rm(list = ls())
train <-read.csv("mnist_train.csv")
test <-read.csv("mnist_test.csv")
dim_train <- dim(train) # 60000 by 785
dim_test <- dim(test) # 10000 by 785
mnist <- rbind(train, test) # combine the data

### Part 1: perform PCA on entire data, reduce the dimensions to 90% of total variance
X <- mnist[, -1]
label <- mnist[, 1]
label <- as.factor(label)

# Some columns only have all zeros, we need to drop them
# otherwise, standardization produces NaN values in the column
X_reduced <- X[,-(which(colSums(X)==0))] # dropping columns with only zeros inside
pca <- prcomp(X_reduced, center = TRUE, scale = TRUE)

# determine the number of components to explain 90% of the total variance
variance_explained <- cumsum(pca$sdev^2/sum(pca$sdev^2))
num_components <- which(variance_explained >= 0.9)[1]

# reduce dimensions to 90% of the total variance
```

```
X_reconstruct <- predict(pca, newdata = X_reduced)[, 1:num_components]

# X_reduced: does not include the y label, it is used in dimensionality reduction in PCA
# mnist_reduced: a data frame includes x feature, which are the PC's, and y label
mnist_reduced <- cbind(label, X_reconstruct) # add back y label to the dataset
mnist_reduced <- as.data.frame(mnist_reduced) # Convert matrix to data.frame: factor only
mnist_reduced$label <- as.factor(mnist_reduced$label) # change it to factor so the algorith

### Part 2
# load the necessary libraries
library(e1071)
library(caret)

# set up the cross-validation scheme
# cv <- trainControl(method = "repeatedcv",
#                    number = 10,
#                    repeats = 1
#                    # repeats = 3,
#                    verboseIter = TRUE)

# define the search grid for tuning the SVM parameters
# grid <- expand.grid(sigma = seq(0.01, 2, length = 20),
#                     cost = 2^(seq(-5, 10, length = 16)))

X_train_recon <- X_reconstruct[1:nrow(train), ]
label_train_recon <- label[1:nrow(train)]
X_test_recon <- X_reconstruct[(nrow(train)+1):nrow(mnist), ]
label_test_recon <- label[(nrow(train)+1):nrow(mnist)]
svm_radial <- train(y = label_train_recon,
                    x = X_train_recon,
                    method = "svmRadial",
                    trControl = cv,
                    # preProc = c("center", "scale"),
                    metric = "Accuracy")

# print the best parameters and corresponding accuracy
svm_radial$bestTune
# 3 0.004689845 1

# svm_radial$results[svm_radial$bestTune[1], ]
svm_radial$results

# sigma     C   Accuracy      Kappa   AccuracySD      KappaSD
# 1 0.004689845 0.25 0.9325997 0.9250865 0.003643884 0.004049874
# 2 0.004689845 0.50 0.9463330 0.9403502 0.003540210 0.003934797
# 3 0.004689845 1.00 0.9567165 0.9518912 0.002794284 0.003105768

# use the best model to make predictions on the test set
predictions <- predict(svm_radial, newdata = test)

### another try using without using caret
num_components <- which(variance_explained >= 0.9)[1]
reduced_train_x <- pca$x[, 1:num_components]
```

```
# Train SVM with radial basis function kernel
tuned_parameters <- tune(svm, train.x = reduced_train_x, train.y = label,
                         kernel = "radial", ranges = list(cost = 2^(-1:1), gamma = 2^(-1:1
svm_model <- tuned_parameters$best.model

# Evaluate SVM on test set
reduced_test_x <- predict(pca, newdata = test_x)[, 1:num_components]
predicted_labels <- predict(svm_model, newdata = reduced_test_x)
accuracy <- mean(predicted_labels == test_y)
cat("Test accuracy:", accuracy)

### 3. Train SVM with polynomial kernel
train_control <- trainControl(method="cv", number=10)
svm_poly <- train(y = label_train_recon,
                  x = X_train_recon,
                  method = "svmPoly",
                  trControl = train_control,
                  # preProc = c("center", "scale"),
                  metric = "Accuracy")


### another example
library(e1071)
library(kernlab)

# Load MNIST data
rm(list = ls())
train <-read.csv("mnist_train.csv")
test <-read.csv("mnist_test.csv")
mnist <- rbind(train, test) # combine the data

# Split data into training and test sets
set.seed(123)
mnist_reduced <- mnist[,-(which(colSums(mnist)==0))] # dropping columns with only zeros in

##########
### Part 1: Perform a PCA on the entire data (by first combining the training and test data
pca <- prcomp(mnist_reduced[,-1], center = TRUE, scale. = TRUE)
variance_explained <- cumsum(pca$sdev^2 / sum(pca$sdev^2))
num_components <- which(variance_explained >= 0.9)[1] # 238
train_pca <- as.data.frame(pca$x[1:60000, 1:num_components])
train_pca$label <- as.factor(train$label)

# Perform PCA on test data using training data PCA object
# test_pca <- as.data.frame(predict(pca, newdata = test[,-1])[,1:num_components])
# test_pca$label <- test$label
test_pca <- as.data.frame(pca$x[60001:70000, 1:num_components])
test_pca$label <- as.factor(test$label)



# Define cross-validation function
cv_function <- function(cost) {
  folds <- cut(seq(1,nrow(train_pca)),breaks=10,labels=FALSE)
```

```
  cv_results <- lapply(1:10, function(i) {
    cat('fold:', i)
    test_fold_index <- which(folds==i,arr.ind=TRUE)
    train_fold_index <- setdiff(1:nrow(train_pca), test_fold_index)
    train_fold <- train_pca[train_fold_index,]
    test_fold <- train_pca[test_fold_index,]
    cat('train_SVM', i)
    svm_model <- svm(label ~ ., data = train_fold, kernel = 'polynomial', cost = cost, deg
    cat('Done', i)
    predictions <- predict(svm_model, test_fold[,-ncol(test_fold)])
    accuracy <- mean(predictions == test_fold$label)
    return(accuracy)
  })
  mean(unlist(cv_results))
}

# Tune cost parameter using cross-validation
cost_values <- seq(0.1, 10, by = 0.1)
cv_results <- sapply(cost_values, cv_function)
best_cost <- cost_values[which.max(cv_results)]

# Train SVM using best cost value
svm_model <- svm(label ~ ., data = train_pca, kernel = 'polynomial', cost = best_cost, deg

# Make predictions on test data
predictions <- predict(svm_model, test_pca[,-ncol(test_pca)])

# Evaluate model performance
confusionMatrix(predictions, test_pca$label)
```