

Problem Solutions

e-Chapter 9

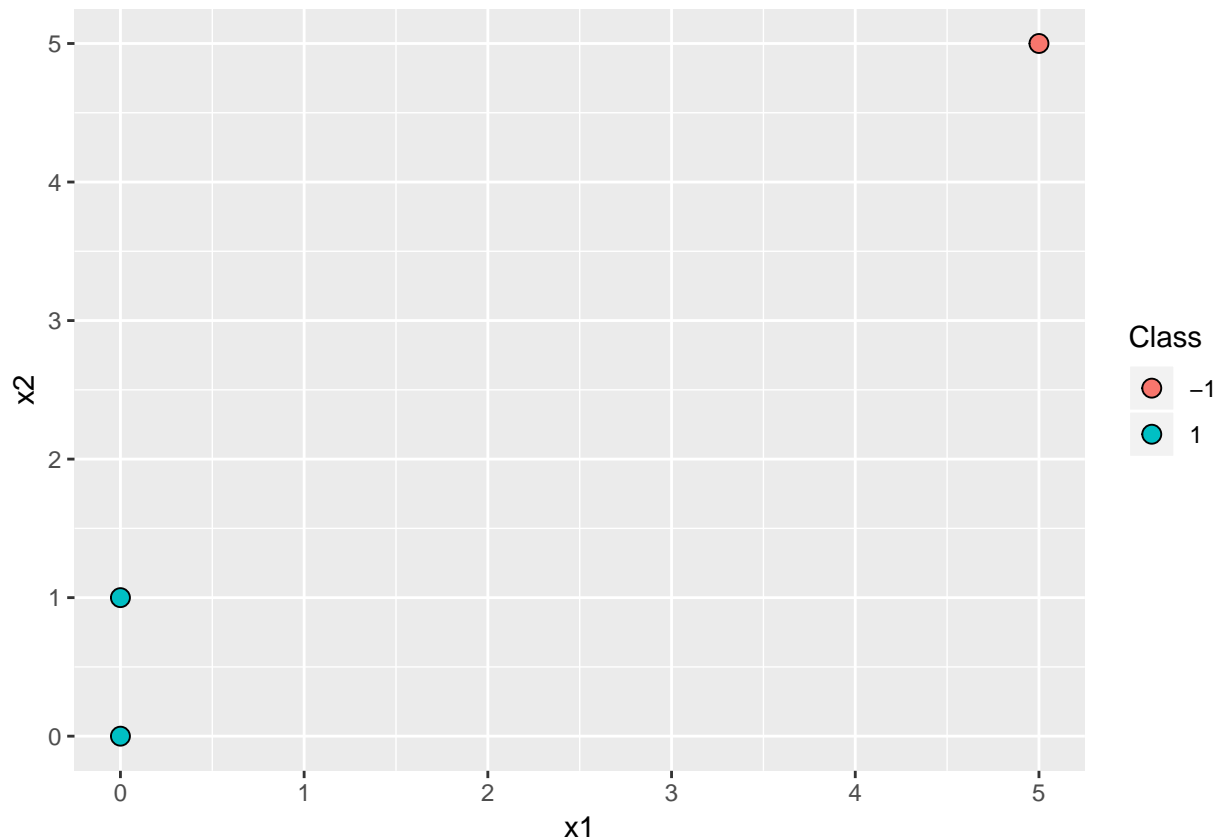
Pierre Paquay

Problem 9.1

(a) We begin by implementing the nearest neighbor method on the raw data.

```
data <- data.frame(x1 = c(0, 0, 5), x2 = c(0, 1, 5))
class <- as.factor(c(1, 1, -1))

ggplot(data, aes(x = x1, y = x2, fill = class)) + geom_point(size = 3, shape = 21) +
  guides(fill = guide_legend(title = "Class"))
```



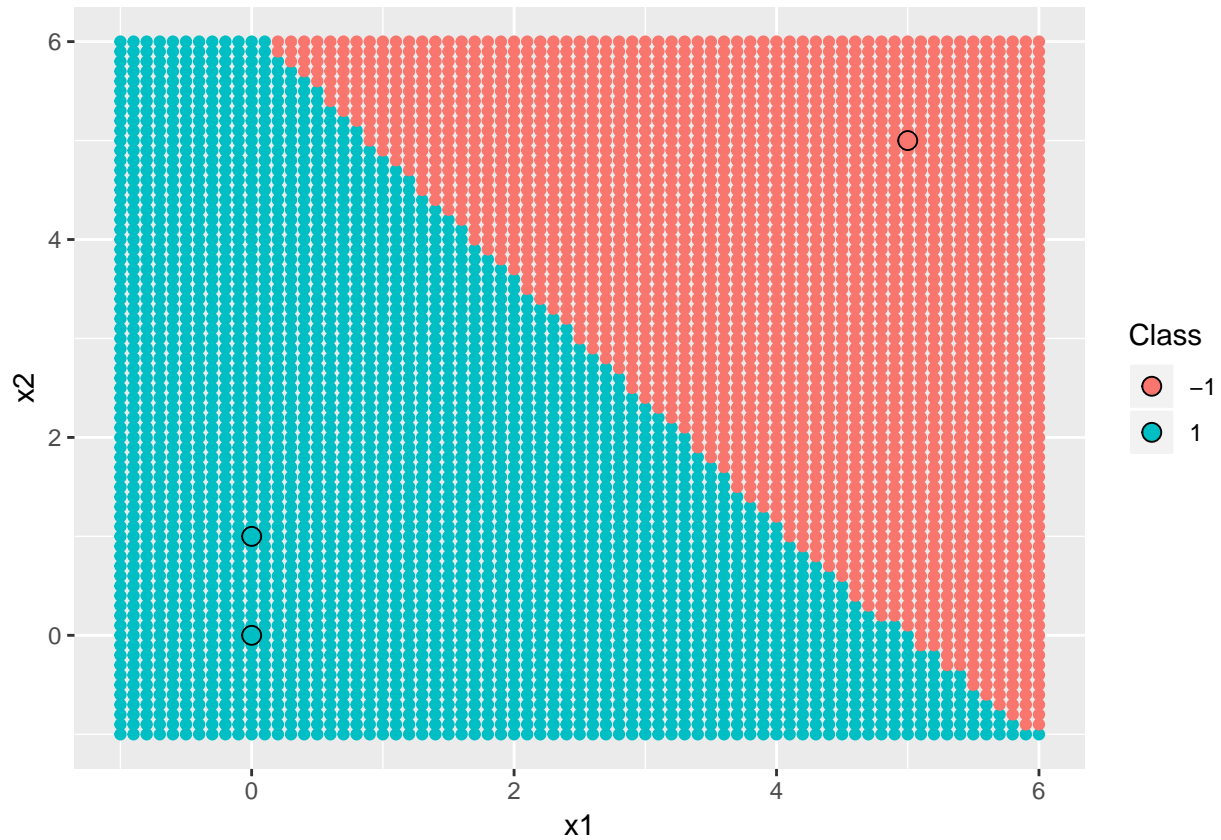
```
grid <- expand.grid(x1 = seq(min(data[, 1] - 1), max(data[, 1] + 1), by = 0.1),
  x2 = seq(min(data[, 2] - 1), max(data[, 2] + 1), by = 0.1))

knn_mod <- knn(data, grid, class, k = 1, prob = TRUE)
```

Below, we show the decision regions of the final hypothesis.

```
ggplot() + geom_point(data = grid, aes(x = x1, y = x2, col = knn_mod)) +
  geom_point(data = data, aes(x = x1, y = x2, fill = class), size = 3, shape = 21) +
  guides(fill = guide_legend(title = "Class")) +
```

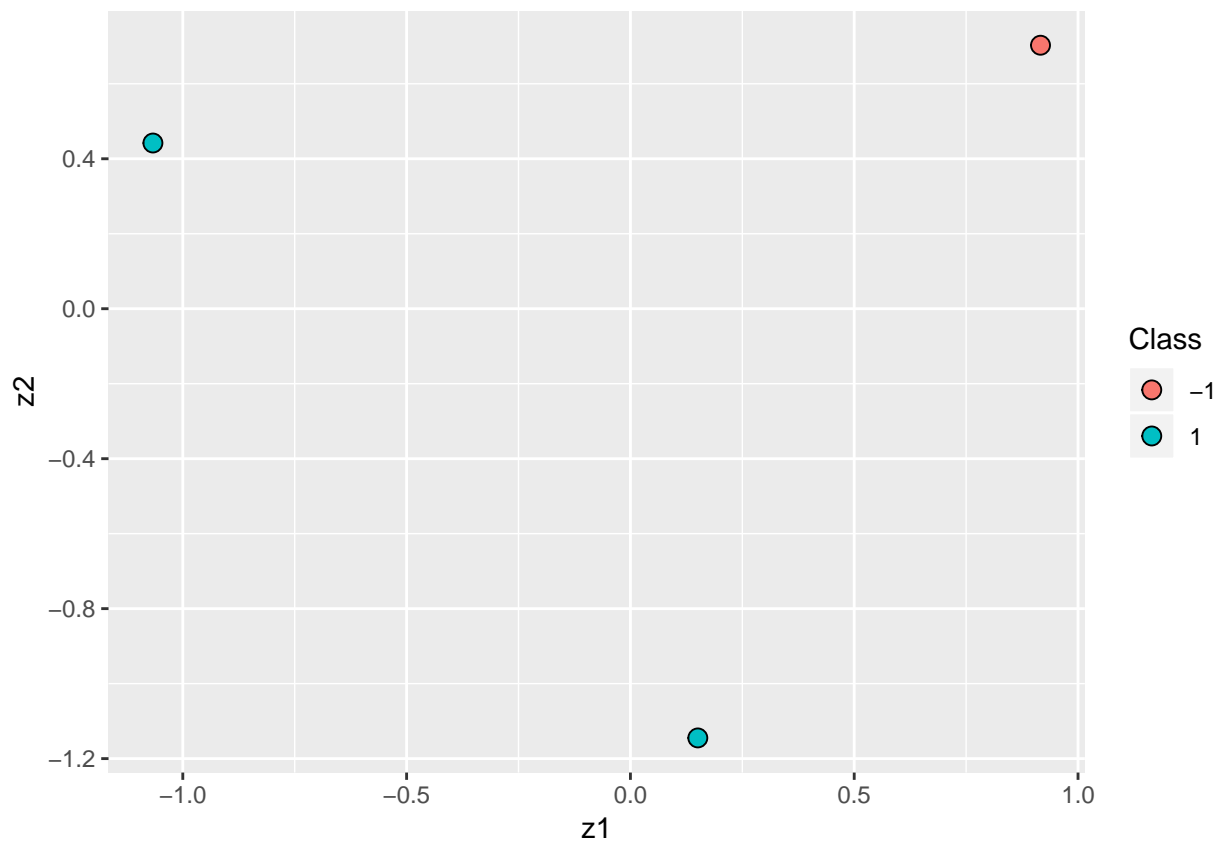
```
guides(col = guide_legend(title = "Class"))
```



(b) Here, we transform to whitened coordinates and we run the nearest neighbor rule.

```
data_centered <- apply(data, 2, function(y) y - mean(y))
sigma <- t(data_centered) %*% as.matrix(data_centered) / 2
sigma_sqr <- sqrtm(sigma)
sigma_sqr_inv <- solve(sigma_sqr)
data_whitened <- as.matrix(data_centered) %*% sigma_sqr_inv
data_whitened <- as.data.frame(data_whitened)
colnames(data_whitened) <- c("z1", "z2")

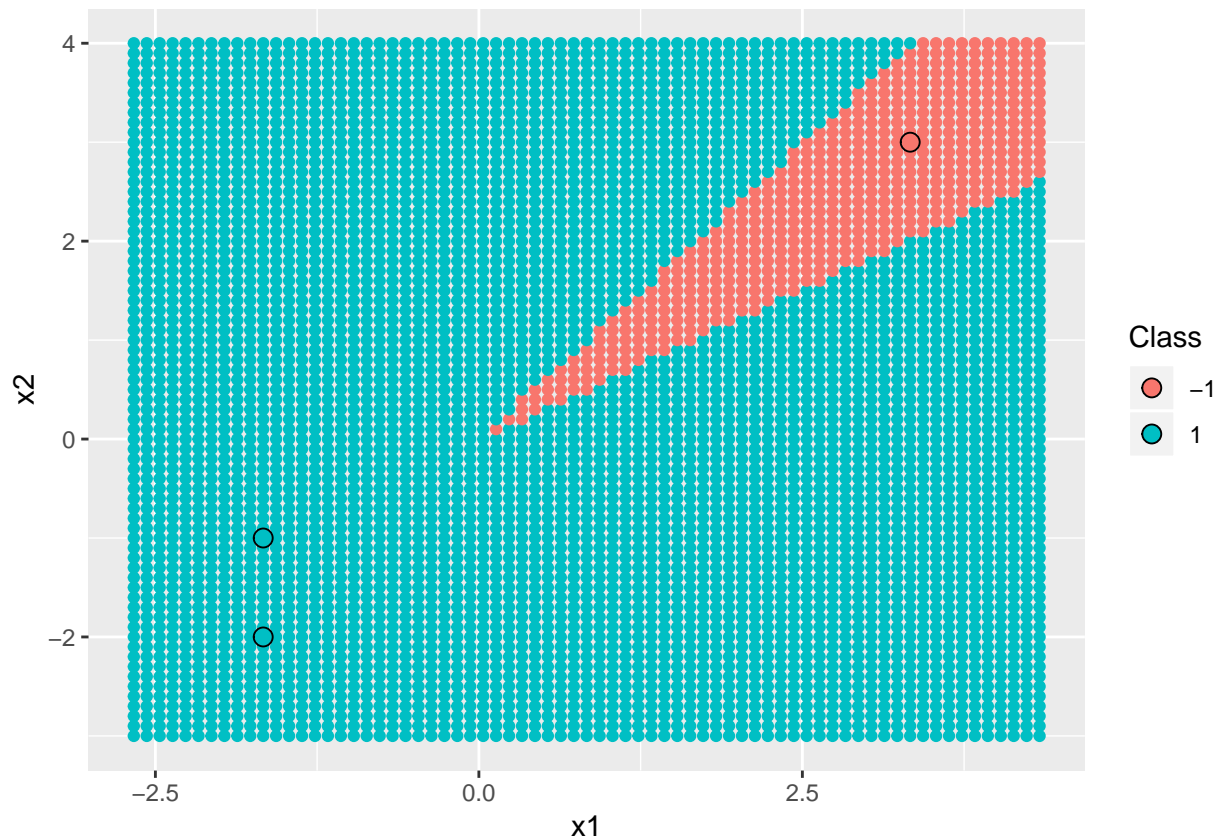
ggplot(data_whitened, aes(x = z1, y = z2, fill = class)) + geom_point(size = 3, shape = 21) +
  guides(fill = guide_legend(title = "Class"))
```



```
grid_centered <- expand.grid(x1 = seq(min(data_centered[, 1] - 1),
                                     max(data_centered[, 1] + 1), by = 0.1),
                           x2 = seq(min(data_centered[, 2] - 1),
                                     max(data_centered[, 2] + 1), by = 0.1))
grid_whitened <- as.matrix(grid_centered) %*% sigma_sqr_inv
knn_mod_whitened <- knn(data_whitened, grid_whitened, class, k = 1, prob = TRUE)
```

We show the decision region of the final hypothesis in the original space as well.

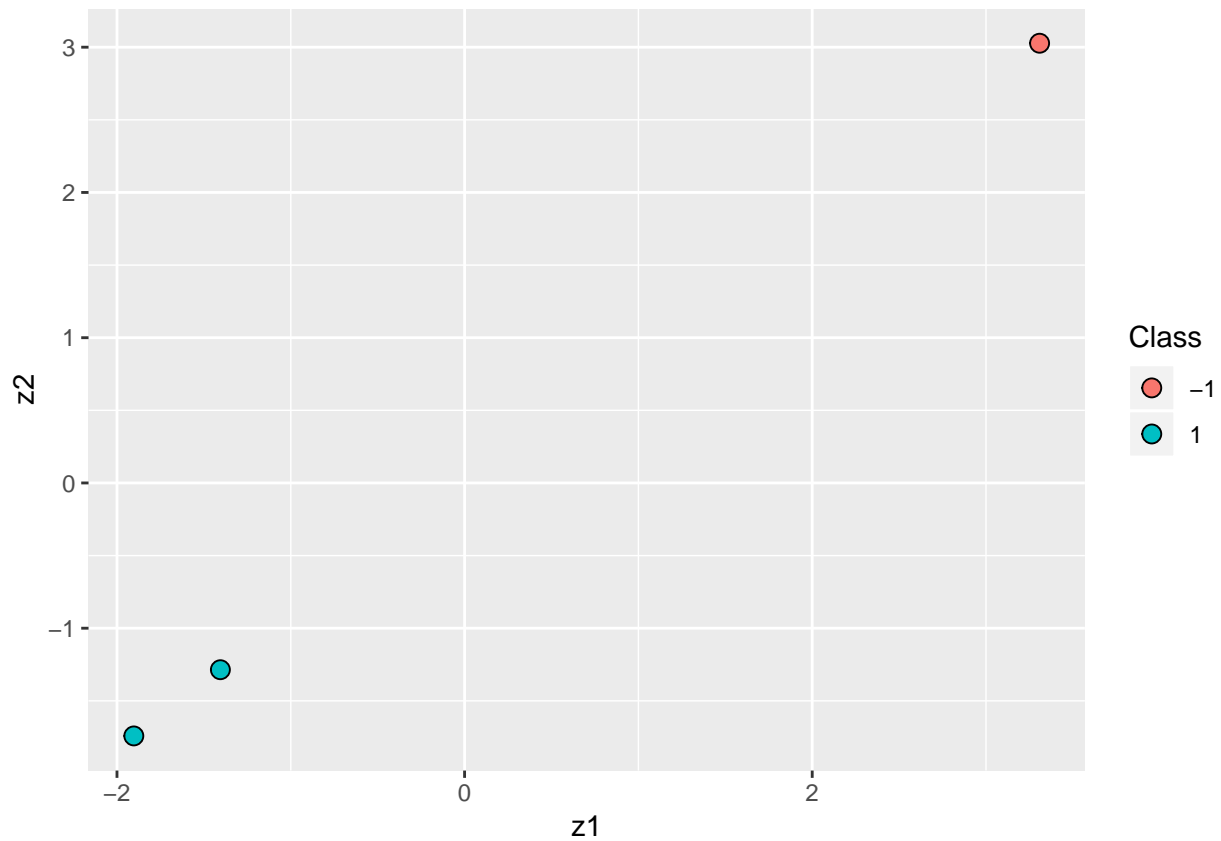
```
ggplot() + geom_point(data = grid_centered, aes(x = x1, y = x2, col = knn_mod_whitened)) +
  geom_point(data = as.data.frame(data_centered), aes(x = x1, y = x2, fill = class),
            size = 3, shape = 21) +
  guides(fill = guide_legend(title = "Class")) +
  guides(col = guide_legend(title = "Class"))
```



(c) Finally, we use principal component analysis to reduce the data to 1 dimension for our nearest neighbor classifier.

```
SVD_decomp <- svd(data_centered)
V1 <- SVD_decomp$v[, 1]
Z <- data_centered %*% V1
data_pca <- Z %*% t(V1)
data_pca <- as.data.frame(data_pca)
colnames(data_pca) <- c("z1", "z2")

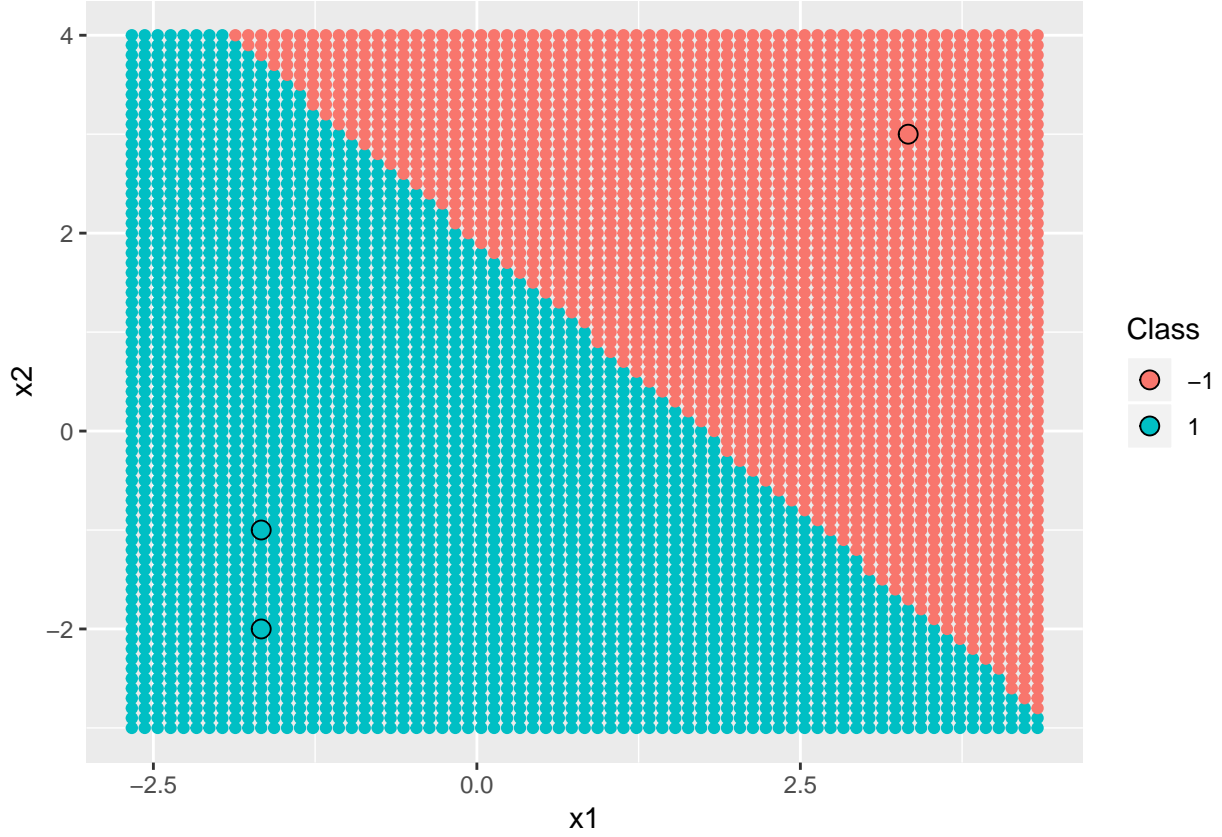
ggplot(data_pca, aes(x = z1, y = z2, fill = class)) + geom_point(size = 3, shape = 21) +
  guides(fill = guide_legend(title = "Class"))
```



```
grid_pca <- (as.matrix(grid_centered) %*% V1) %*% t(V1)
knn_mod_pca <- knn(data_pca, grid_pca, class, k = 1, prob = TRUE)
```

Once again, we show the decision regions of the final hypothesis in the original space.

```
ggplot() + geom_point(data = grid_centered, aes(x = x1, y = x2, col = knn_mod_pca)) +
  geom_point(data = as.data.frame(data_centered), aes(x = x1, y = x2, fill = class),
    size = 3, shape = 21) +
  guides(fill = guide_legend(title = "Class")) +
  guides(col = guide_legend(title = "Class"))
```



Problem 9.2

(a) Here, we have $x'_n = x_n + b$. Let us compute \bar{x}' , $\sigma_i'^2$ and Σ' . We have

$$\bar{x}' = \frac{1}{N} \sum_{n=1}^N x'_n = \frac{1}{N} \sum_{n=1}^N (x_n + b) = \bar{x} + b;$$

moreover, we have

$$\sigma_i'^2 = \frac{1}{N} \sum_{n=1}^N (x'_{ni} - \bar{x}'_i)^2 = \frac{1}{N} \sum_{n=1}^N (x_{ni} + b_i - \bar{x}_i - b_i)^2 = \sigma_i^2;$$

and finally

$$\Sigma' = \frac{1}{N} \sum_{n=1}^N (x'_n - \bar{x}')(x'_n - \bar{x}')^T = \frac{1}{N} \sum_{n=1}^N (x_n + b - \bar{x} - b)(x_n + b - \bar{x} - b)^T = \Sigma.$$

- **Centering.** We may write that

$$z'_n = x'_n - \bar{x}' = x_n + b - \bar{x} - b = z_n.$$

- **Normalization.** Here, we have that

$$z'_n = D'(x'_n - \bar{x}')$$

where

$$D' = \text{diag}(1/\sigma'_1, \dots, 1/\sigma'_d) = \text{diag}(1/\sigma_1, \dots, 1/\sigma_d) = D.$$

In that case, we have

$$z'_n = D(x'_n - \bar{x}') = D(x_n + b - \bar{x} - b) = z_n.$$

- **Whitening.** We have

$$z'_n = \Sigma'^{-1/2}(x'_n - \bar{x}') = \Sigma^{-1/2}(x_n + b - \bar{x} - b) = z_n.$$

(b) Here, we have $x'_n = \alpha x_n$ with $\alpha > 0$. Let us compute \bar{x}' , $\sigma_i'^2$ and Σ' . We have

$$\bar{x}' = \frac{1}{N} \sum_{n=1}^N x'_n = \frac{1}{N} \sum_{n=1}^N \alpha x_n = \alpha \bar{x};$$

moreover, we have

$$\sigma_i'^2 = \frac{1}{N} \sum_{n=1}^N (x'_{ni} - \bar{x}'_i)^2 = \frac{1}{N} \sum_{n=1}^N (\alpha x_{ni} - \alpha \bar{x}_i)^2 = \alpha^2 \sigma_i^2;$$

and finally

$$\Sigma' = \frac{1}{N} \sum_{n=1}^N (x'_n - \bar{x}')(x'_n - \bar{x}')^T = \frac{1}{N} \sum_{n=1}^N (\alpha x_n - \alpha \bar{x})(\alpha x_n - \alpha \bar{x})^T = \alpha^2 \Sigma.$$

- **Centering.** We may write that

$$z'_n = x'_n - \bar{x}' = \alpha x_n - \alpha \bar{x} = \alpha z_n \neq z_n.$$

- **Normalization.** Here, we have that

$$z'_n = D'(x'_n - \bar{x}')$$

where

$$D' = \text{diag}(1/\sigma'_1, \dots, 1/\sigma'_d) = \text{diag}(1/(\alpha\sigma_1), \dots, 1/(\alpha\sigma_d)) = \frac{1}{\alpha} D.$$

In that case, we have

$$z'_n = \frac{1}{\alpha} D(x'_n - \bar{x}') = \frac{1}{\alpha} D(\alpha x_n - \alpha \bar{x}) = D(x_n - \bar{x}) = z_n.$$

- **Whitening.** We have

$$z'_n = \Sigma'^{-1/2}(x'_n - \bar{x}') = \frac{1}{\alpha} \Sigma^{-1/2}(\alpha x_n - \alpha \bar{x}) = z_n.$$

(c) Here, we have $x'_n = Ax_n$ with $A = \text{diag}(a_1, \dots, a_d)$ ($a_i \neq 0$). Let us compute \bar{x}' , $\sigma_i'^2$ and Σ' . We have

$$\bar{x}' = \frac{1}{N} \sum_{n=1}^N x'_n = \frac{1}{N} \sum_{n=1}^N Ax_n = A\bar{x};$$

moreover, we have

$$\sigma_i'^2 = \frac{1}{N} \sum_{n=1}^N (x'_{ni} - \bar{x}'_i)^2 = \frac{1}{N} \sum_{n=1}^N (a_i x_{ni} - a_i \bar{x}_i)^2 = a_i^2 \sigma_i^2;$$

and finally

$$\Sigma' = \frac{1}{N} \sum_{n=1}^N (x'_n - \bar{x}')(x'_n - \bar{x}')^T = \frac{1}{N} \sum_{n=1}^N (Ax_n - A\bar{x})(Ax_n - A\bar{x})^T = A\Sigma A^T.$$

- **Centering.** We may write that

$$z'_n = x'_n - \bar{x}' = Ax_n - A\bar{x} = Az_n \neq z_n.$$

- **Normalization.** Here, we have that

$$z'_n = D'(x'_n - \bar{x}')$$

where

$$D' = \text{diag}(1/\sigma'_1, \dots, 1/\sigma'_d) = \text{diag}(1/(a_1\sigma_1), \dots, 1/(a_d\sigma_d));$$

which means that $D'A = D$. In that case, we have

$$z'_n = D'(x'_n - \bar{x}') = D'A(x_n - \bar{x}) = D(x_n - \bar{x}) = z_n.$$

- **Whitening.** We have

$$z'_n = \Sigma'^{-1/2}(x'_n - \bar{x}') = (A\Sigma A^T)^{-1/2}A(x_n - \bar{x}) \neq z_n.$$

(d) Here, we have $x'_n = Ax_n$ with $\det(A) \neq 0$. Let us compute \bar{x}' , $\sigma_i'^2$ and Σ' . We have

$$\bar{x}' = \frac{1}{N} \sum_{n=1}^N x'_n = \frac{1}{N} \sum_{n=1}^N Ax_n = A\bar{x};$$

moreover, we have

$$\sigma_i'^2 = \frac{1}{N} \sum_{n=1}^N (x'_{ni} - \bar{x}'_i)^2 = \frac{1}{N} \sum_{n=1}^N (l_i^T x_n - l_i^T \bar{x})^2 = \frac{1}{N} \sum_{n=1}^N l_i^T (x_n - \bar{x})(x_n - \bar{x})^T l_i = l_i^T \Sigma l_i$$

where l_i is the i th row of A ; and finally

$$\Sigma' = \frac{1}{N} \sum_{n=1}^N (x'_n - \bar{x}')(x'_n - \bar{x}')^T = \frac{1}{N} \sum_{n=1}^N (Ax_n - A\bar{x})(Ax_n - A\bar{x})^T = A\Sigma A^T.$$

- **Centering.** We may write that

$$z'_n = x'_n - \bar{x}' = Ax_n - A\bar{x} = Az_n \neq z_n.$$

- **Normalization.** Here, we have that

$$z'_n = D'(x'_n - \bar{x}')$$

where

$$D' = \text{diag}(1/\sigma'_1, \dots, 1/\sigma'_d) = \text{diag}(1/(l_1^T \Sigma l_1), \dots, 1/(l_d^T \Sigma l_d)).$$

In that case, we have

$$z'_n = D'(x'_n - \bar{x}') = D'A(x_n - \bar{x}) = D'Az_n \neq z_n.$$

- **Whitening.** We have

$$z'_n = \Sigma'^{-1/2}(x'_n - \bar{x}') = (A\Sigma A^T)^{-1/2}A(x_n - \bar{x}) \neq z_n.$$

Problem 9.3

We may write

$$\Gamma = \text{diag}(\lambda_1, \dots, \lambda_d)$$

where $\lambda_i > 0$. With this in mind, it is easy to describe $\Gamma^{1/2}$ and $\Gamma^{-1/2}$; we have

$$\Gamma^{1/2} = \text{diag}(\sqrt{\lambda_1}, \dots, \sqrt{\lambda_d}) \text{ and } \Gamma^{-1/2} = \text{diag}(1/\sqrt{\lambda_1}, \dots, 1/\sqrt{\lambda_d}).$$

To prove the first fact, we simply need to compute $\Gamma^{1/2}\Gamma^{1/2}$, we have

$$\Gamma^{1/2}\Gamma^{1/2} = \text{diag}(\sqrt{\lambda_1}, \dots, \sqrt{\lambda_d}) \text{diag}(\sqrt{\lambda_1}, \dots, \sqrt{\lambda_d}) = \text{diag}(\lambda_1, \dots, \lambda_d) = \Gamma;$$

to prove the second fact, we need to compute $\Gamma^{1/2}\Gamma^{-1/2}$ (the same reasoning can be applied to $\Gamma^{-1/2}\Gamma^{1/2}$), we have

$$\Gamma^{1/2}\Gamma^{-1/2} = \text{diag}(\sqrt{\lambda_1}, \dots, \sqrt{\lambda_d}) \text{diag}(1/\sqrt{\lambda_1}, \dots, 1/\sqrt{\lambda_d}) = I.$$

Now, to prove that $\Sigma^{1/2} = U\Gamma^{1/2}U^T$, we write that

$$\Sigma^{1/2}\Sigma^{1/2} = U\Gamma^{1/2} \underbrace{U^T U}_{=I} \Gamma^{1/2}U^T = U\Gamma U^T = \Sigma.$$

Then, to prove that $\Sigma^{-1/2} = U\Gamma^{-1/2}U^T$, we write that

$$\Sigma^{-1/2}\Sigma^{1/2} = U\Gamma^{-1/2} \underbrace{U^T U}_{=I} \Gamma^{1/2}U^T = UIU^T = I.$$

Problem 9.4

We know that $A = V\psi$, consequently we get

$$V^T A = \underbrace{V^T V}_{=I} \psi = \psi$$

because V is an orthogonal matrix. Moreover, we also have that

$$\psi^T \psi = A^T \underbrace{V V^T}_{=I} A = A^T A = I$$

since A is an orthogonal matrix as well; this means that ψ is also an orthogonal matrix.

Problem 9.5

(a) if A is a diagonal matrix ($N = d$), it suffices to define U , Γ and V as follows

$$U = V = I_N \text{ and } \Gamma = A.$$

In this case, we have

$$A = U\Gamma V^T.$$

(c) If A is a matrix with pairwise orthogonal columns, we may write that

$$A^T A = \text{diag}(a_1, \dots, a_d)$$

where $a_i > 0$. Now if we define D as follows

$$D = \text{diag}(1/\sqrt{a_1}, \dots, 1/\sqrt{a_d}),$$

it is easy to see that $U = AD$ is actually a matrix with orthonormal columns since

$$U^T U = D^T A^T A D = I_d.$$

In this case, we have

$$A = U\Gamma V^T$$

with $\Gamma = \text{diag}(\sqrt{a_1}, \dots, \sqrt{a_d})$ and $V = I_d$.

(d) If A has SVD UTV^T and $Q^T Q = I$, we may write that

$$QA = (QU)\Gamma V^T$$

with QU a matrix with orthonormal columns since

$$(QU)^T(QU) = U^T \underbrace{Q^T Q}_{=I} U = I_d.$$

(e) If A has blocks A_i along the diagonal such that A_i has SVD $U_i \Gamma_i V_i^T$, we simply have to define U , Γ , and V as follows

$$U = \begin{pmatrix} U_1 & & \\ & \ddots & \\ & & U_m \end{pmatrix}, \Gamma = \begin{pmatrix} \Gamma_1 & & \\ & \ddots & \\ & & \Gamma_m \end{pmatrix} \text{ and } V = \begin{pmatrix} V_1 & & \\ & \ddots & \\ & & V_m \end{pmatrix}.$$

In that case, we immediately get that

$$A = U\Gamma V^T$$

with

$$U^T U = \begin{pmatrix} U_1^T & & \\ & \ddots & \\ & & U_m^T \end{pmatrix} \begin{pmatrix} U_1 & & \\ & \ddots & \\ & & U_m \end{pmatrix} = I,$$

$$V^T V = \begin{pmatrix} V_1^T & & \\ & \ddots & \\ & & V_m^T \end{pmatrix} \begin{pmatrix} V_1 & & \\ & \ddots & \\ & & V_m \end{pmatrix} = VV^T = I,$$

and Γ diagonal.

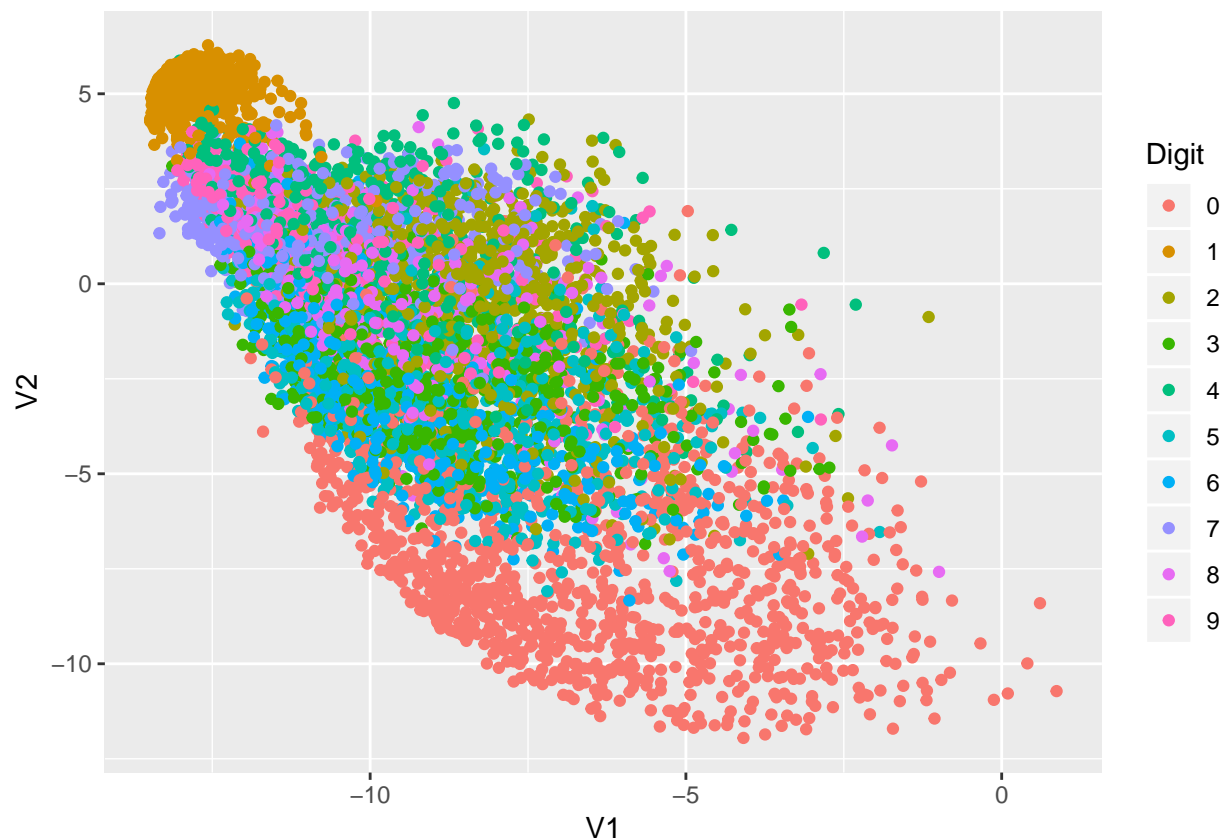
Problem 9.6

Here, we suppose that the digits data were not centered and we perform PCA to obtain a 2-dimensional feature vector.

```
digits <- read.delim("zip.train", header = FALSE, sep = " ")
digits$V258 <- NULL
colnames(digits) <- c("digit", 1:256)

X <- as.matrix(digits[, -1])
y <- digits[, 1]
SVD_decomp <- svd(X)
V2 <- SVD_decomp$v[, 1:2]
Z <- X %*% V2
digits_pca <- Z %*% t(V2)
digits_pca_proj <- digits_pca %*% V2
digits_pca_proj <- as.data.frame(digits_pca_proj)

ggplot(digits_pca_proj, aes(x = V1, y = V2, col = as.factor(y))) + geom_point() +
  guides(col = guide_legend(title = "Digit"))
```



The transformed data is not whitened since its covariance matrix is not the identity matrix.

```
cov(digits_pca_proj)
```

```
##          V1          V2
## V1  6.842064 -8.121955
## V2 -8.121955 17.058353
```

Problem 9.7

As we have that

$$z_n = \sqrt{N} \Gamma_k^{-1} V_k^T x_n,$$

we may let Z be

$$Z = \sqrt{N} X (\Gamma_k^{-1} V_k^T)^T = \sqrt{N} X V_k \Gamma_k^{-1}.$$

Moreover, since X is centered, it is easy to see that Z is centered as well because

$$\bar{z} = \frac{1}{N} Z^T \mathbf{1} = \frac{\sqrt{N}}{N} \Gamma_k^{-1} V_k^T \underbrace{X^T \mathbf{1}}_{=0} = 0.$$

To prove that the transformed data is actually whitened, we have to compute $Z^T Z$, if we use the SVD of $X = U\Gamma V^T$, we get

$$\begin{aligned}
\frac{1}{N} Z^T Z &= \frac{1}{N} (\sqrt{N} X V_k \Gamma_k^{-1})^T (\sqrt{N} X V_k \Gamma_k^{-1}) \\
&= \Gamma_k^{-1} V_k^T X^T X V_k \Gamma_k^{-1} \\
&= \Gamma_k^{-1} V_k^T V \Gamma \underbrace{U^T U}_{=I_d} \Gamma V^T V_k \Gamma_k^{-1} \\
&= \Gamma_k^{-1} V_k^T V \underbrace{\Gamma^2}_{=\text{diag}(\gamma_1^2, \dots, \gamma_d^2)} V^T V_k \Gamma_k^{-1}.
\end{aligned}$$

We also have that

$$V_k^T V = \begin{pmatrix} v_1^T \\ \vdots \\ v_k^T \end{pmatrix} (v_1, \dots, v_k \mid v_{k+1}, \dots, v_d) = (I_k \mid 0)$$

and

$$V^T V_k = \begin{pmatrix} I_k \\ 0 \end{pmatrix}.$$

This means that

$$\Gamma_k^{-1} (I_k \mid 0) = (\Gamma_k^{-1} \mid 0),$$

and in the same way

$$\begin{pmatrix} I_k \\ 0 \end{pmatrix} \Gamma_k^{-1} = \begin{pmatrix} \Gamma_k^{-1} \\ 0 \end{pmatrix}.$$

Consequently, we get that

$$\begin{aligned}
\frac{1}{N} Z^T Z &= (\Gamma_k^{-1} \mid 0) \Gamma^2 \begin{pmatrix} \Gamma_k^{-1} \\ 0 \end{pmatrix} \\
&= (\Gamma_k^{-1} \mid 0) \left(\begin{array}{c|c} \Gamma_k^2 & 0 \\ \hline 0 & * \end{array} \right) \begin{pmatrix} \Gamma_k^{-1} \\ 0 \end{pmatrix} \\
&= I_k,
\end{aligned}$$

which means that Z is whitened.

Problem 9.8

We begin by constructing a two dimensional feature as described in the book and we apply the algorithm to the digits data giving us the features z_1 and z_2 .

```

digits <- read.delim("zip.train", header = FALSE, sep = " ")
digits$V258 <- NULL
colnames(digits) <- c("digit", 1:256)

X <- as.matrix(digits[, -1])
y <- digits[, 1]
X_centered <- apply(X, 2, function(x) x - mean(x))
X_centered_1 <- X_centered[y == 1, ]
X_centered_not1 <- X_centered[y != 1, ]

SVD_decomp_1 <- svd(X_centered_1)
v1 <- SVD_decomp_1$v[, 1]

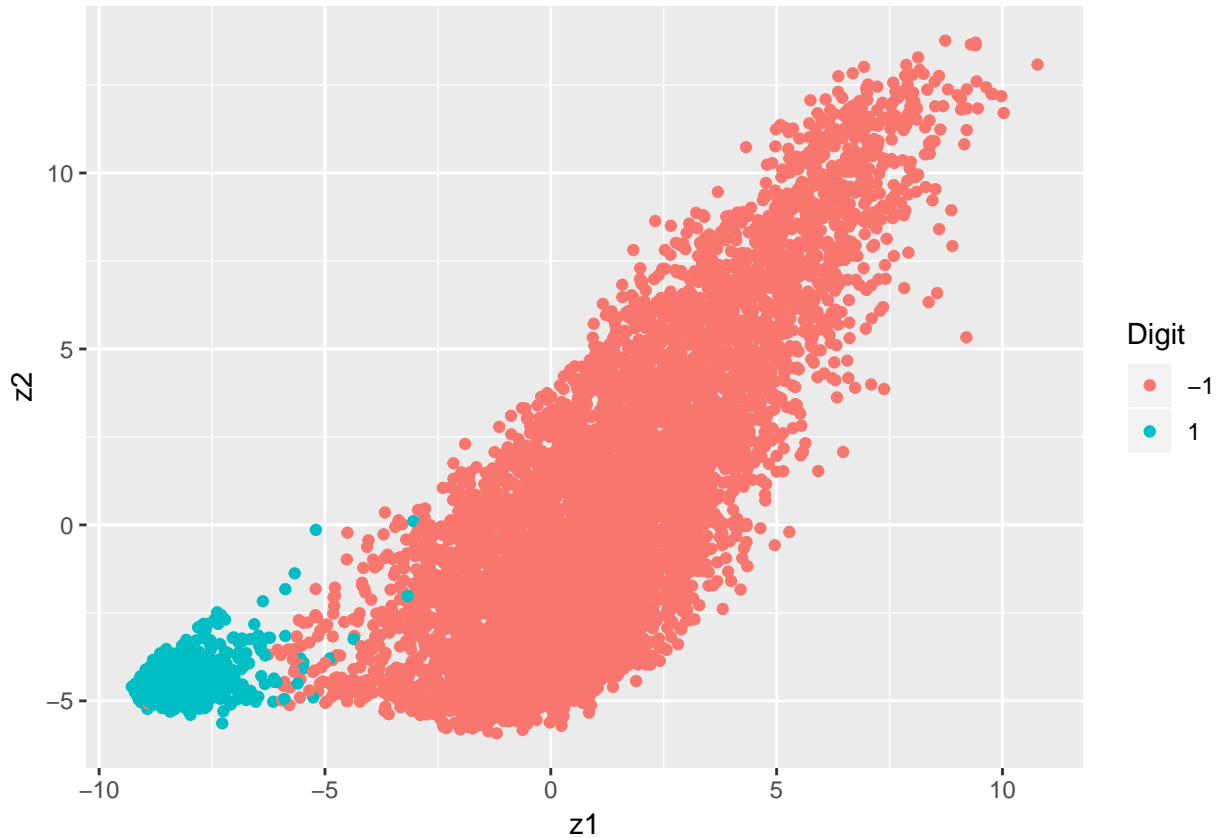
SVD_decomp_not1 <- svd(X_centered_not1)
v2 <- SVD_decomp_not1$v[, 1]

```

(a) We give below a scatter plot of our resulting two features.

```
z1 <- X_centered %*% v1
z2 <- X_centered %*% v2
y_1 <- ifelse(y == 1, +1, -1)
z <- data.frame(z1, z2, y_1)
```

```
ggplot(z, aes(x = z1, y = z2, col = as.factor(y_1))) + geom_point() +
  guides(col = guide_legend(title = "Digit"))
```



(b) No, the directions of v_1 and v_2 are not necessarily orthogonal.

```
v1 %*% v2 == 0
```

```
##      [,1]
## [1,] FALSE
```

(c) Since $\hat{V} = [v_1, v_2]$ has full rank, we know that

$$\hat{V}^+ = (\hat{V}^T \hat{V})^{-1} \hat{V}^T.$$

In this case, we have

$$\hat{X} = X \hat{V} (\hat{V}^T \hat{V})^{-1} \hat{V}^T.$$

(d) This method of constructing features is supervised since we need the target values to apply it.

Problem 9.9

(a) Since \hat{X} is a matrix whose rows are projected onto k basis vectors, it is pretty obvious that $\text{rank}(\hat{X}) = k$.

(b) We may write that

$$\|\Gamma - U^T \hat{X} V\|_F^2 = \|U^T X V - U^T \hat{X} V\|_F^2 = \|U^T (X - \hat{X}) V\|_F^2.$$

Moreover, we may also write (see exercise 9.9) that

$$\begin{aligned} \|U^T (X - \hat{X}) V\|_F^2 &= \|U(U^T (X - \hat{X}) V) V^T\|_F^2 \\ &= \|U U^T (X - \hat{X}) \underbrace{V V^T}_{=I}\|_F^2 \\ &= \text{trace}[(U U^T (X - \hat{X}))^T (U U^T (X - \hat{X}))] \\ &= \text{trace}[(X - \hat{X})^T U \underbrace{U^T U}_{=I} U^T (X - \hat{X})] \\ &= \|U^T (X - \hat{X})\|_F^2 \\ &\leq \underbrace{\|U^T\|_F^2}_{=\text{trace}(U^T U)=d} \|X - \hat{X}\|_F^2 \\ &\leq d \|X - \hat{X}\|_F^2 \end{aligned}$$

where we used the fact that $\|AB\|_F^2 \leq \|A\|_F^2 \|B\|_F^2$.

(c) Since the rank of a matrix product is less than each factor, we have that

$$\text{rank}(\hat{\Gamma}) = \text{rank}(U^T \hat{X} V) \leq \text{rank}(\hat{X}) = k.$$

(d) It is easy to see that

$$\begin{aligned} \|\Gamma - \hat{\Gamma}\|_F^2 &= \sum_{i,j=1}^d (\gamma_{ij} - \hat{\gamma}_{ij})^2 \\ &= \sum_{i=1}^d (\gamma_{ii} - \hat{\gamma}_{ii})^2 + \sum_{i \neq j} (\gamma_{ij} - \hat{\gamma}_{ij})^2 \\ &\geq \sum_{i=1}^d (\gamma_{ii} - \hat{\gamma}_{ii})^2. \end{aligned}$$

So, the optimal choice for $\hat{\Gamma}$ must have all off-diagonal elements equal to zero.

(e) Since we know that $\text{rank}(\hat{\Gamma}) \leq k$, we may conclude that the optimal $\hat{\Gamma}$ can have at most k non-zero diagonal elements.

(f) We may write for our optimal $\hat{\Gamma}$ that

$$\begin{aligned} \|\Gamma - \hat{\Gamma}\|_F^2 &= \sum_{i=1}^d (\gamma_{ii} - \hat{\gamma}_{ii})^2 \\ &= \sum_{i=1}^k (\gamma_{ii} - \hat{\gamma}_{ii})^2 + \underbrace{\sum_{i=k+1}^d (\gamma_{ii} - \hat{\gamma}_{ii})^2}_{\sum_{i=k+1}^d \gamma_{ii}^2} \\ &\geq \sum_{i=k+1}^d \gamma_{ii}^2. \end{aligned}$$

Consequently, the minimum reconstruction error is equal to $\sum_{i=k+1}^d \gamma_{ii}^2$.

(g) If $\hat{X} = XV_k V_k^T$, we choose our optimal $\hat{\Gamma}$ such that

$$\begin{aligned}\hat{\Gamma} &= U^T \hat{X} V \\ &= U^T X V_k V_k^T V \\ &= \underbrace{U^T U}_{=I} \Gamma V^T V_k V_k^T V \\ &= \Gamma (V^T V_k) (V_k^T V)\end{aligned}$$

where

$$V_k^T V = (I_k \mid 0) \text{ and } V^T V_k = \begin{pmatrix} I_k \\ 0 \end{pmatrix}.$$

This means that

$$\hat{\Gamma} = \Gamma \left(\begin{array}{c|c} I_k & 0 \\ \hline 0 & 0 \end{array} \right) = \left(\begin{array}{cccc|c} \gamma_1 & & & & 0 \\ & \ddots & & & \vdots \\ & & \gamma_k & & 0 \\ \hline 0 & \cdots & 0 & 0 & 0 \end{array} \right).$$

Thus, we finally have

$$\Gamma - \hat{\Gamma} = \left(\begin{array}{c|ccc} 0 & 0 & \cdots & 0 \\ \hline 0 & -\gamma_{k+1} & & \\ \vdots & & \ddots & \\ 0 & & & -\gamma_d \end{array} \right),$$

and consequently

$$\|\Gamma - \hat{\Gamma}\|_F^2 = \sum_{i=k+1}^d \gamma_i^2.$$

So, for this choice of $\hat{\Gamma}$, we actually check each of the conditions obtained in points (d), (e), and (f) to characterize the optimal choice of $\hat{\Gamma}$; so $\hat{X} = XV_k V_k^T$ is the optimal choice for the reconstructed vector. In this case, we know from point (a) that

$$\|X - \hat{X}\|_F^2 \geq \|\Gamma - \hat{\Gamma}\|_F^2,$$

so for our optimal choice of \hat{X} , the minimum reconstruction error is $\sum_{i=k+1}^d \gamma_i^2$.

Problem 9.10

(a) The main difference between Algorithm 1 and Algorithm 2 is that Algorithm 1 first performs SVD and then splits the data, and Algorithm 2 first splits the data and then performs SVD.

(b) Here, we generate N random normally distributed d -dimensional inputs x_1, \dots, x_N with their respective targets $y_n = w_f^T x_n + \epsilon_n$ where w_f is normally distributed and ϵ_n is independent Gaussian noise with variance 0.5.

```
set.seed(10)
d <- 5
N <- 40
k <- 3

target <- function(w, x) {

  return(sum(w * as.matrix(x)) + rnorm(1, sd = sqrt(0.5)))
}
```

```

data_gen <- function(d, N) {
  wf <- rnorm(d)
  X <- data.frame(x1 = rnorm(N), x2 = rnorm(N), x3 = rnorm(N),
                 x4 = rnorm(N), x5 = rnorm(N))
  y <- apply(X, 1, target, w = wf)

  return(list(X = X, y = y, wf = wf))
}

```

Now, we use Algorithms 1 and 2 to compute estimates of E_{out} .

```

E_cross_1 <- function(X, y) {
  SVD <- svd(X)
  Vk <- SVD$v[, 1:k]
  Z <- as.matrix(X) %*% Vk
  en <- numeric(N)
  for (n in 1:N) {
    Zn <- Z[-n, ]
    yn <- y[-n]
    wn_minus <- solve(t(Zn) %*% Zn) %*% t(Zn) %*% yn
    en[n] <- (sum(Z[n, ] * wn_minus) - y[n])^2
  }
  E1 <- mean(en)

  return(E1)
}

E_cross_2 <- function(X, y) {
  en <- numeric(N)
  for (n in 1:N) {
    Xn <- X[-n, ]
    yn <- y[-n]
    SVD_minus <- svd(Xn)
    Vk_minus <- SVD_minus$v[, 1:k]
    Zn <- as.matrix(Xn) %*% Vk_minus
    wn_minus <- solve(t(Zn) %*% Zn) %*% t(Zn) %*% yn
    wn_minus2 <- Vk_minus %*% wn_minus
    en[n] <- (sum(X[n, ] * wn_minus2) - y[n])^2
  }
  E2 <- mean(en)

  return(E2)
}

```

We compute E_{out} below.

```

E_out <- function(X, y, wf) {
  SVD <- svd(X)
  Vk <- SVD$v[, 1:k]
  Z <- as.matrix(X) %*% Vk
  w <- solve(t(Z) %*% Z) %*% t(Z) %*% y
  w2 <- Vk %*% w

  Ntest <- 1000
  X_test <- data.frame(x1 = rnorm(Ntest), x2 = rnorm(Ntest), x3 = rnorm(Ntest),

```



```

        x4 = rnorm(Ntest), x5 = rnorm(Ntest))
y_test <- apply(X_test, 1, target, w = wf)

E_out <- mean((as.matrix(X_test) %*% w2 - y_test)^2)

return(E_out)
}

```

Finally, we repeat this process 10^5 times and we report the averages \bar{E}_1 , \bar{E}_2 , and \bar{E}_{out} .

```

iter <- 100000
E1 <- numeric(iter)
E2 <- numeric(iter)
Eout <- numeric(iter)
for (i in 1:iter) {
  data <- data_gen(d, N)
  X <- data$X
  y <- data$y
  wf <- data$wf

  E1[i] <- E_cross_1(X, y)
  E2[i] <- E_cross_2(X, y)
  Eout[i] <- E_out(X, y, wf)
}
E1_avg <- mean(E1)
E2_avg <- mean(E2)
Eout_avg <- mean(Eout)

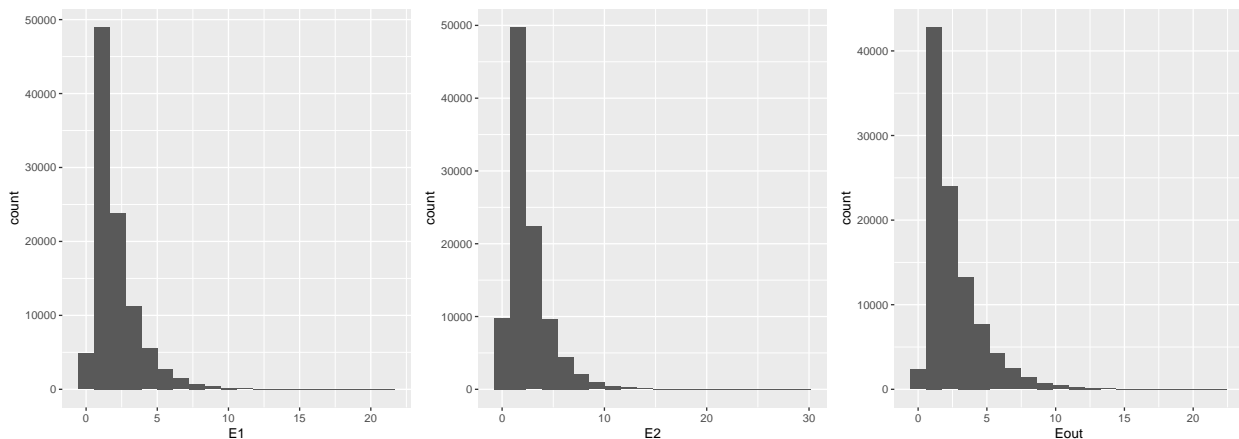
```

We plot below the histograms of E_1 , E_2 and E_{out} .

```

results <- data.frame(E1 = E1, E2 = E2, Eout = Eout)
plot1 <- ggplot(results, aes(x = E1)) + geom_histogram(bins = 20)
plot2 <- ggplot(results, aes(x = E2)) + geom_histogram(bins = 20)
plot3 <- ggplot(results, aes(x = Eout)) + geom_histogram(bins = 20)
grid.arrange(plot1, plot2, plot3, nrow = 1)

```



We get $\bar{E}_1 = 2.0445914$, $\bar{E}_2 = 2.5364857$, and $\bar{E}_{out} = 2.5386052$.

(c) As we may see, the average \bar{E}_1 is clearly not as close to \bar{E}_{out} as \bar{E}_2 is. This was to be expected since in cross-validation we have to compute every element of our hypothesis after the data split is done, not before.

(d) As stated in point (c), the correct estimate of E_{out} is E_2 .