Stats 315a: Statistical Learning
Problem Set 2

Dong-Bang Tsai[*]
Department of Applied Physics, Stanford University, Stanford, California 94305, USA

Problem 1 (ESL 3.12 & 3.30)

(ESL 3.12)

Let's say original data set is $X = \begin{pmatrix} x_1^T \\ \vdots \\ x_n^T \end{pmatrix}$, and original output is y. Now we augment the centered matrix X with additional rows $\sqrt{\lambda} I$, and augment y with p zeros. By doing so, we can get new training data set, $y' = \begin{pmatrix} y \\ 0 \\ \vdots \\ 0 \end{pmatrix}$, and $X' = \begin{pmatrix} x_1^T \\ \vdots \\ x_n^T \\ \sqrt{\lambda} I \end{pmatrix}$. Let's apply ordinary least squares regression on the augmented data set, and the solution is (in textbook Eq. (3.3)

$$\hat{\beta}' = ({X'}^T X)^{-1} {X'}^T y' \tag{1}$$

$$= \left(\begin{pmatrix} x_1 & \dots & x_n & \sqrt{\lambda} I \end{pmatrix} \begin{pmatrix} x_1^T \\ \vdots \\ x_n^T \\ \sqrt{\lambda} I \end{pmatrix} \right)^{-1} \begin{pmatrix} x_1 & \dots & x_n & \sqrt{\lambda} I \end{pmatrix} \begin{pmatrix} y \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{2}$$

$$= \left(X X^T + \lambda I \right)^{-1} X y \tag{3}$$

which is the solution of ridge regression in Eq. (3.44) in textbook.

(ESL 3.30)

Let's say original data set is $X = \begin{pmatrix} x_1^T \\ \vdots \\ x_n^T \end{pmatrix}$, and original output is y. Now we augment the X with additional rows $\sqrt{\lambda} I$ and multiply $(1+\lambda)^{-1/2}$, and augment y with p zeros. By doing so, we can get new training data set, $y' = \begin{pmatrix} y \\ 0 \\ \vdots \\ 0 \end{pmatrix}$, and $X' = (1+\lambda\alpha)^{-1/2} \begin{pmatrix} x_1^T \\ \vdots \\ x_n^T \\ \sqrt{\lambda\alpha} I \end{pmatrix}$, insert this augmented data set into elastic-net optimization problem,

[*]Electronic address: `dbtsai@stanford.edu`

also use some tricks from ESL 3.12, using $\gamma = \frac{(1-\alpha)}{\alpha}$, $\phi = (1+\lambda\alpha)^{-1/2}$

$$\min_{\beta'}(y' - X'\beta')^T(y' - X'\beta') + \lambda(\alpha|\beta'|_2^2 + (1-\alpha)|\beta'|_1) \tag{4}$$

$$= \min_{\beta'}(y - \phi X\beta')^T(y - \phi X\beta') + \frac{(1-\alpha)}{\alpha}\phi|\beta'|_1 \tag{5}$$

$$= \min_{\beta'}(y - X\beta)^T(y - X\beta) + \frac{(1-\alpha)}{\alpha}\phi|\beta|_1 \tag{6}$$

where $\phi\beta' = \beta$. As a result, a elastic-net optimization problem can turn into a lasso problem with a proper augmented data.

Problem 2 (ESL 3.23)

(a)

As we discussed in page 46 in textbook, the orthogonality lemma is given by $< \hat{y}, y - \hat{y} >= 0$. With $u(\alpha) = \alpha X\hat{\beta} = \alpha\hat{y}$,

$$< x_j, y - u(\alpha) > =< x_j, y - \alpha\hat{y} > \tag{7}$$

$$=< x_j, (1-\alpha)y + \alpha_y - \alpha\hat{y} > \tag{8}$$

$$= (1-\alpha) < x_j, y > +\alpha < x_j, y - \hat{y} > \tag{9}$$

$$= (1-\alpha) < x_j, y > \tag{10}$$

therefore,

$$\frac{1}{N}| < x_j, y - u(\alpha) > | = \frac{1}{N}(1-\alpha)| < x_j, y > | = (1-\alpha)\lambda \tag{11}$$

As a result, the correlations of each x_j with the residuals remain equal in magnitude as we progress toward u.

(b)

The correlations can be given by

$$C_j(\alpha) = \frac{\frac{1}{N}| < x_j, y - u(\alpha) > |}{\sqrt{\frac{1}{N} < x_j, x_j >}\sqrt{\frac{1}{N} < y - u(\alpha), y - u(\alpha) >}} \tag{12}$$

$$= \frac{(1-a)\lambda}{\sqrt{\frac{1}{N} < y - u(\alpha), y - u(\alpha) >}} \tag{13}$$

Let's figure out the term in denominator. Using the trick we used before, $y - u(\alpha) = (1-\alpha)y + \alpha_y - \alpha\hat{y}$

$$< y - u(\alpha), y - u(\alpha) > =< (1-\alpha)y + \alpha_y - \alpha\hat{y}, (1-\alpha)y + \alpha_y - \alpha\hat{y} > \tag{14}$$

$$= (1-\alpha)^2 < y, y > +2\alpha(1-\alpha) < y, y - \hat{y} > +\alpha^2 < y - \hat{y}, y - \hat{y} > \tag{15}$$

$$= (1-\alpha)^2 N + 2\alpha < y - \hat{y}, y - \hat{y} > +2\alpha < \hat{y}, y - \hat{y} > +\alpha^2 RSS \tag{16}$$

$$= (1-\alpha)^2 N + 2\alpha < y - \hat{y}, y - \hat{y} > +\alpha^2 RSS \tag{17}$$

$$= N\left((1-\alpha)^2 + \frac{\alpha}{N}(2-\alpha)RSS\right) \tag{18}$$

Therefore, the correlations are

$$C_j(\alpha) = \frac{(1-a)\lambda}{\sqrt{(1-\alpha)^2 + \frac{\alpha}{N}(2-\alpha)RSS}} \tag{19}$$

which will be decreased monotonically to zero.

(c)

The result from part (a) shows that in LAR algorithm, when we move β_j from 0 towards its least-squares coefficient $< x_j, r >$, the correlations remain equal in magnitude; therefore, it keeps the correlations tied. Also, as α increased, the correlation function will be monotonically decreased to zero.

Problem 3 (ESL 4.2)

(a)

If the LDA rule classifies to class 2, it implies that $\hat{\delta}_2(x) > \hat{\delta}_1(x)$, where $\hat{\delta}_k(x)$ is defined in Eq. (4.10) in textbook,

$$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2}\mu_k^T \Sigma^{-1} \mu_k + \log \pi_k \tag{20}$$

where π, μ, Σ are estimated from our training data. Insert the condition in the problem set into this equation, we have

$$x^T \hat{\Sigma}^{-1} \hat{\mu}_2 - \frac{1}{2}\hat{\mu}_2^T \hat{\Sigma}^{-1} \hat{\mu}_2 + \log \frac{N_2}{N} > x^T \hat{\Sigma}^{-1} \hat{\mu}_1 - \frac{1}{2}\hat{\mu}_1^T \hat{\Sigma}^{-1} \hat{\mu}_1 + \log \frac{N_1}{N} \tag{21}$$

$$\therefore \quad x^T \hat{\Sigma}^{-1}(\hat{\mu}_2 - \hat{\mu}_1) > \frac{1}{2}\hat{\mu}_2^T \hat{\Sigma}^{-1} \hat{\mu}_2 - \frac{1}{2}\hat{\mu}_1^T \hat{\Sigma}^{-1} \hat{\mu}_1 + \log \frac{N_1}{N} - \log \frac{N_2}{N} \tag{22}$$

The other case will simply change ">" to "<" which can be easily proven.

(b)

Let's define X is $N \times (p+1)$ matrix, and the extra dimension is from adding the constant variable 1 in X, and $\hat{\Theta} = \begin{pmatrix} \hat{\beta}_0 \\ \hat{\beta} \end{pmatrix}$ will include $\hat{\beta}_0$ which is the intercept. Therefore, the normal equation given in Eq. (2.5) in textbook will be $X^T(\hat{y} - X\hat{\Theta}) = 0$.

Let's calculate all the terms in the normal equation, and see if we can simplify the equation. The first term will be

$$X^T \hat{y} = \begin{pmatrix} 1 & 1 & \dots & 1 \\ x_1 & x_2 & \dots & x_n \end{pmatrix} \begin{pmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_n \end{pmatrix} = \begin{pmatrix} \sum \hat{y}_i \\ \sum x_i \hat{y}_i \end{pmatrix} = \begin{pmatrix} \sum \hat{y}_i \\ \sum x_i \hat{y}_i \end{pmatrix} = \begin{pmatrix} N_1(-\frac{N}{N_1}) + N_2 \frac{N}{N_2} \\ -\frac{N}{N_1} N_1 \hat{\mu}_1 + \frac{N}{N_2} N_2 \hat{\mu}_2 \end{pmatrix} \tag{23}$$

$$= \begin{pmatrix} 0 \\ N(\hat{\mu}_2 - \hat{\mu}_1) \end{pmatrix} \tag{24}$$

The second term involved in the normal equation will be

$$X^T X = \begin{pmatrix} 1 & \dots & 1 \\ x_1 & \dots & x_n \end{pmatrix} \begin{pmatrix} 1 & x_1^T \\ \vdots & \vdots \\ 1 & x_n^T \end{pmatrix} = \begin{pmatrix} N & \sum x_i^T \\ \sum x_i & \sum x_i x_i^T \end{pmatrix} = \begin{pmatrix} N & N_1 \hat{\mu}_1^T + N_2 \hat{\mu}_2^T \\ N_1 \hat{\mu}_1 + N_2 \hat{\mu}_2 & \sum x_i x_i^T \end{pmatrix} \tag{25}$$

where $\sum x_i x_i^T$ is a $p \times p$ matrix. The covariance $\hat{\Sigma}$ is defined in textbook on page 109,

$$\hat{\Sigma} = \frac{1}{N-2} \sum_{k=1}^{2} \sum_{g_i=k} (x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T \tag{26}$$

$$= \frac{1}{N-2} \sum_{k=1}^{2} \sum_{g_i=k} (x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T \tag{27}$$

$$= \frac{1}{N-2} \sum_{k=1}^{2} \sum_{g_i=k} x_i x_i^T - x_i \hat{\mu}_k^T - \hat{\mu}_k x_i^T + \hat{\mu}_k \hat{\mu}_k^T \tag{28}$$

$$= \frac{1}{N-2} \left(N_1 \hat{\mu}_1 \hat{\mu}_1^T + N_2 \hat{\mu}_2 \hat{\mu}_2^T - 2 \left(N_1 \hat{\mu}_1 \hat{\mu}_1^T + N_2 \hat{\mu}_2 \hat{\mu}_2^T \right) + \sum_{k=1}^{2} \sum_{g_i=k} x_i x_i^T \right) \tag{29}$$

$$= \frac{1}{N-2} \left(- \left(N_1 \hat{\mu}_1 \hat{\mu}_1^T + N_2 \hat{\mu}_2 \hat{\mu}_2^T \right) + \sum_{i=1}^{N} x_i x_i^T \right) \tag{30}$$

therefore, we have

$$\sum_{i=1}^{N} x_i x_i^T = (N-2)\hat{\Sigma} + \left(N_1 \hat{\mu}_1 \hat{\mu}_1^T + N_2 \hat{\mu}_2 \hat{\mu}_2^T \right) \tag{31}$$

Now, the normal equation $X^T X \hat{\Theta} = X^T y$ can rewritten as

$$\begin{pmatrix} N & N_1 \hat{\mu}_1^T + N_2 \hat{\mu}_2^T \\ N_1 \hat{\mu}_1 + N_2 \hat{\mu}_2 & (N-2)\hat{\Sigma} + \left(N_1 \hat{\mu}_1 \hat{\mu}_1^T + N_2 \hat{\mu}_2 \hat{\mu}_2^T \right) \end{pmatrix} \begin{pmatrix} \hat{\beta}_0 \\ \hat{\beta} \end{pmatrix} = \begin{pmatrix} 0 \\ N(\hat{\mu}_2 - \hat{\mu}_1) \end{pmatrix} \tag{32}$$

therefore,

$$\hat{\beta}_0 = - \left(\frac{N_1}{N} \hat{\mu}_1^T + \frac{N_2}{N} \hat{\mu}_2^T \right) \hat{\beta} \tag{33}$$

and

$$\left[-\frac{1}{N} \left(N_1 \hat{\mu}_1 + N_2 \hat{\mu}_2 \right) \left(N_1 \hat{\mu}_1^T + N_2 \hat{\mu}_2^T \right) + (N-2)\hat{\Sigma} + \left(N_1 \hat{\mu}_1 \hat{\mu}_1^T + N_2 \hat{\mu}_2 \hat{\mu}_2^T \right) \right] \hat{\beta} = N(\hat{\mu}_2 - \hat{\mu}_1) \tag{34}$$

$$\therefore \quad \left[(N-2)\hat{\Sigma} + \frac{N_1 N_2}{N} (\hat{\mu}_2 - \hat{\mu}_1)(\hat{\mu}_2 - \hat{\mu}_1)^T \right] \hat{\beta} = N(\hat{\mu}_2 - \hat{\mu}_1) \tag{35}$$

$$\therefore \quad \left[(N-2)\hat{\Sigma} + \frac{N_1 N_2}{N} \hat{\Sigma}_B \right] \hat{\beta} = N(\hat{\mu}_2 - \hat{\mu}_1) \tag{36}$$

where $\hat{\Sigma}_B = (\hat{\mu}_2 - \hat{\mu}_1)(\hat{\mu}_2 - \hat{\mu}_1)^T$

(c)

$$\hat{\Sigma}_B \hat{\beta} = (\hat{\mu}_2 - \hat{\mu}_1)(\hat{\mu}_2 - \hat{\mu}_1)^T \hat{\beta} = (\hat{\mu}_2 - \hat{\mu}_1)c \tag{37}$$

where $c = (\hat{\mu}_2 - \hat{\mu}_1)^T \hat{\beta}$ is a constant number. Therefore, $\hat{\Sigma}_B \hat{\beta}$ is in the direction $(\hat{\mu}_2 - \hat{\mu}_1)$. Insert this fact into the normal equation,

$$(N-2)\hat{\Sigma}\hat{\beta} = N(\hat{\mu}_2 - \hat{\mu}_1) - \frac{cN_1 N_2}{N} (\hat{\mu}_2 - \hat{\mu}_1) \tag{38}$$

therefore

$$\hat{\beta} \propto \hat{\Sigma}^{-1} (\hat{\mu}_2 - \hat{\mu}_1) \tag{39}$$

(d)

Assume that the relation between new coding and old coding is $y' = ay + b$, the cost function and corresponding coefficients will be

$$\mathbf{RSS}(\beta') = \sum_i (\hat{y}' - \hat{\beta}'_0 - \hat{\beta}'^T x_i)^2 \tag{40}$$

$$= \sum_i (ay + b - \hat{\beta}'_0 - \hat{\beta}'^T x_i)^2 \tag{41}$$

$$= a^2 \sum_i (y + \frac{b - \hat{\beta}'_0}{a} - \frac{\hat{\beta}'^T}{a} x_i)^2 \tag{42}$$

note that the minimization of new cost function should give the same result since they have similar form; therefore,

$$\hat{\beta}_0 = \frac{b}{a} - \hat{\beta}'_0 \tag{43}$$

$$\hat{\beta} = \frac{\hat{\beta}'}{a} \tag{44}$$

As a result, we can conclude that $\hat{\beta}' \propto \hat{\Sigma}^{-1}(\hat{\mu}_2 - \hat{\mu}_1)$, and $\hat{\Sigma}_B \hat{\beta}'$ is in the direction $(\hat{\mu}_2 - \hat{\mu}_1)$.

(e)

$$\hat{\beta} = \gamma \hat{\Sigma}^{-1}(\hat{\mu}_2 - \hat{\mu}_1) \tag{45}$$

where $\gamma = \frac{N^2 - cN_1N_2}{N(N-2)} > 0$. Therefore, we can construct prediction function

$$\hat{f} = \hat{\beta}_0 + x^T \hat{\beta} \tag{46}$$

$$= -\left(\frac{N_1}{N}\hat{\mu}_1^T + \frac{N_2}{N}\hat{\mu}_2^T\right)\hat{\beta} + x^T\hat{\beta} \tag{47}$$

$$= \gamma \left[x^T - \left(\frac{N_1}{N}\hat{\mu}_1^T + \frac{N_2}{N}\hat{\mu}_2^T\right)\right] \hat{\Sigma}^{-1}(\hat{\mu}_2 - \hat{\mu}_1) \tag{48}$$

Consider the following rule, we classify to class 2 if $\hat{y}_i > 0$; therefore, we can write down

$$\left[x^T - \left(\frac{N_1}{N}\hat{\mu}_1^T + \frac{N_2}{N}\hat{\mu}_2^T\right)\right] \hat{\Sigma}^{-1}(\hat{\mu}_2 - \hat{\mu}_1) > 0 \tag{49}$$

$$\therefore \quad x^T \hat{\Sigma}^{-1}(\hat{\mu}_2 - \hat{\mu}_1) > \left(\frac{N_1}{N}\hat{\mu}_1^T + \frac{N_2}{N}\hat{\mu}_2^T\right) \hat{\Sigma}^{-1}(\hat{\mu}_2 - \hat{\mu}_1) \tag{50}$$

which is not the same as the original LDA rule classifier. However, when $N_1 = N_2$, we have

$$\therefore \quad x^T \hat{\Sigma}^{-1}(\hat{\mu}_2 - \hat{\mu}_1) > \left(\frac{1}{2}\hat{\mu}_1^T + \frac{1}{2}\hat{\mu}_2^T\right) \hat{\Sigma}^{-1}(\hat{\mu}_2 - \hat{\mu}_1) \tag{51}$$

$$= \frac{1}{2}\hat{\mu}_2^T \hat{\Sigma}^{-1}\hat{\mu}_2 - \frac{1}{2}\hat{\mu}_1^T \hat{\Sigma}^{-1}\hat{\mu}_1 \tag{52}$$

which is the same as the original LDA classifier.

Problem 4

Suppose that $Z_1 = a_1^T X$ and $Z_2 = a_2^T X$ are the discriminant variables for the first two discriminant coordinates, a_1 and a_2. The between-class variance of Z_i is $a_i^T B a_i$, and the within-class variance $a_i^T W a_i$. Those are discussed in textbook on page 114. Note that $B + W = T$, where T is the total covariance matrix of X.

According to the textbook on page 116, a_i is an eigenvector of $W^{-1}B$; therefore,

$$W^{-1}Ba_2 = \lambda_2 a_2 \tag{53}$$

which implies

$$a_1^T Ba_2 = \lambda_2 a_1^T W a_2 \tag{54}$$

As a result,

$$\begin{aligned} a_1^T T a_2 &= a_1^T (B+W) a_2 && (55)\\ &= (\lambda_2 + 1) a_1^T W a_2 && (56)\\ &= 0 && (57) \end{aligned}$$

where)$a_1^T W a_2 = 0$ due that a_1 and a_2 are orthogonal in W which has been proven in page 116.

So far, we prove that Z_1 and Z_2 are uncorrelated for population version; however, if we estimate W and B from the sample covariance, the only difference will be a factor. As a result, the conclusion still holds for sample version.

Problem 5

(a)

The solution of the ridge regression coefficient vector $\hat{\beta}_\lambda$ is given by Eq. (3.44) in the textbook,

$$\hat{\beta}_\lambda = (X^T X + \lambda I)^{-1} X^T y \tag{58}$$

Here, we would like to invert $(X^T X + \lambda I)$, and this matrix has dimension of 10,000 . As we know that the complexities of matrix multiplication and matrix inversion are all $O(p^3)$ where n is the dimension of matrix. Therefore, the total cost of computation will be the order of 10^{12}.

(b)

Start from

$$\begin{aligned} \hat{\beta}_\lambda &= (X^T X + \lambda I)^{-1} X^T y && (59)\\ \therefore (X^T X + \lambda I)\hat{\beta}_\lambda &= X^T y && (60)\\ & && (61)\\ \therefore \hat{\beta}_\lambda &= \frac{1}{\lambda}(X^T y - X^T X \hat{\beta}) && (62)\\ &= \frac{1}{\lambda} X^T (y - X\hat{\beta}) && (63)\\ &= X^T \alpha && (64) \end{aligned}$$

where $\alpha = \frac{1}{\lambda}(y - X\hat{\beta})$

(c)

If we write $X = UDV^T$, the solution will be

$$\begin{aligned} \hat{\beta}_\lambda &= (X^T X + \lambda I)^{-1} X^T y && (65)\\ &= (VDU^T UDV^T + \lambda I)^{-1} VDU^T y && (66)\\ &= (VD^2 V^T + \lambda I)^{-1} VDU^T y && (67)\\ &= V(D^2 + \lambda I)^{-1} V^T VDU^T y && (68)\\ &= V(D^2 + \lambda I)^{-1} DU^T y && (69)\\ &= V\theta && (70) \end{aligned}$$

where $\theta = (D^2 + \lambda I)^{-1} D U^T y$ Since we are going to calculate $(D^2 + \lambda I)^{-1}$ which is diagonal matrix, the inversion of this matrix is very fast, and it's $O(p)$, and not depend on λ. Calculate $DU^T y$ is $O(np)$, and multiply $(D^2 + \lambda I)^{-1} D$ and $U^T y$ is $O(p^2)$; therefore, the complexity of whole calculation will be $O(p^2)$.

As long as we can compute SVD of X, the whole computation will be faster than approach (a).

(d)

Simply calculate $\hat{y}_0 = x_0 \hat{\beta}_\lambda$, where x_0 is the new measurement vector.

Problem 6

(a)

LDA on the original 256 dimensional space using MATLAB classify routine gives the following error.

Error of training data: 1.5376%
Error of testing data: 8.3333%

(b)

LDA on the leading 49 principal components of the features using MATLAB classify routine gives the following error.

Error of training data: 4.4989%
Error of testing data: 8.5366%

It gives the worse result compared with directly performing LDA on the original data.

(c)

LDA on the averaged non-overlapping 2×2 pixel blocks using MATLAB classify routine gives the following error.

Error of training data: 3.3030%
Error of testing data: 7.7236%

It gives better testing error result since the fitting in part (a) will gives a over-fitting result to training set. You can see it by comparing the error of training data. As a result, the filtering will reduce the model complexity which decreases the problem of over-fitting in part (a).

(d)

I downloaded glmnet for matlab from http://www-stat.stanford.edu/ tibs/glmnet-matlab/

The glmnet performed on filtered data from part (c) gives the following error (at the end of the path).

Error of training data: 1.5376%
Error of testing data: 8.1301%

(Analytical analysis)

Now, we would like to prove that fitting a linear model with the full set of pixels with the constrain that the coefficients are piecewise constant is equivalent to the filtered data result in part (c).

For the unconstrained model, we try to optimize

$$f(x) = x^T\beta \tag{71}$$

In order to transits to constrained model, let's define separate x vector into 4 groups which represents $k-th$ pixel in 2×2 block. For each groups, there are 8×8 different positions. Therefore, the vector can be rewritten as a tensor with this notation, x_{ij}^k where k denotes which pixels at one position, and ij denotes the position ranged from 1 to 8. Now the original unconstrained model can be given by

$$f(x) = x^T\beta = \sum_k \beta^k x^k \tag{72}$$

$$= \sum_{ij}\sum_k \beta_{ij}^k x_{ij}^k \tag{73}$$

Now, consider that the coefficients are constrained to be piecewise constant, ie, $\beta^1 = \beta^2 = \beta^3 = \beta^4$, the model can be rewritten as

$$f(x) = \sum_{ij}\sum_k \beta_{ij}^k x_{ij}^k \tag{74}$$

$$= \sum_{ij}\beta_{ij}^1 \sum_k x_{ij}^k \tag{75}$$

$$= \sum_{ij}\beta_{ij}^1 4\bar{x}_{ij} \tag{76}$$

$$= \bar{x}^T\bar{\beta} \tag{77}$$

where $\bar{x}$ is the average of original x within the 2×2 area. Therefore, constrained model is the same as fitting a filtered data.

(Source code for part (a) to part (d))

```
% HW2 Q6
% Dong-Bang Tsai
% ========================================================================
clear all;

% Generate training set
zip_train_raw_3 = dlmread('train.3');
zip_train_raw_5 = dlmread('train.5');
zip_train_raw_8 = dlmread('train.8');
X = [zip_train_raw_3; zip_train_raw_5; zip_train_raw_8];
Y = [3*ones(size(zip_train_raw_3,1),1); ...,
     5*ones(size(zip_train_raw_5,1),1); 8*ones(size(zip_train_raw_8,1),1)];
clear zip_train_raw_3 zip_train_raw_5 zip_train_raw_8;

% Generate testing set
zip_test_raw = dlmread('zip.test');
zip_test_raw_3 = zip_test_raw( find(zip_test_raw(:,1)==3), 2:257);
zip_test_raw_5 = zip_test_raw( find(zip_test_raw(:,1)==5), 2:257);
zip_test_raw_8 = zip_test_raw( find(zip_test_raw(:,1)==8), 2:257);
X_test = [zip_test_raw_3; zip_test_raw_5; zip_test_raw_8];
Y_test = [3*ones(size(zip_test_raw_3,1),1); ...,
     5*ones(size(zip_test_raw_5,1),1); 8*ones(size(zip_test_raw_8,1),1)];
clear zip_test_raw zip_test_raw_3 zip_test_raw_5 zip_test_raw_8;


% a)
% The error of training set:
error_train = 100*sum(classify(X, X, Y,'linear')~=Y)/length(Y)
% The error of testing set:
error_test  = 100*sum(classify(X_test, X, Y,'linear')~=Y_test)/length(Y_test)

% b)
[U,S,V] = svd(X);
V_leading = V(:,1:49);
clear U S V;
X_leading = X*V_leading;
X_test_leading = X_test*V_leading;
% The error of training set:
error_leading_train = 100*sum(classify(X_leading, X_leading, Y,'linear') ...,
                      ~=Y)/length(Y)
% The error of testing set:
error_leading_test  = 100*sum(classify(X_test_leading, X_leading, Y,'linear') ...,
                        ~=Y_test)/length(Y_test)

% c)
X_square=reshape(X,length(Y),16,16);
X_test_square=reshape(X_test,length(Y_test),16,16);
% Average the non-operlapping 2x2 pixel blocks
X_square_filtered = zeros(length(Y),8,8);
X_test_square_flitered = zeros(length(Y_test),8,8);
for i=1:8
    for j=1:8
        X_square_filtered(:,i,j) = (X_square(:,2*i-1,2*j-1) +X_square(:,2*i,2*j-1) ...,
                                    +X_square(:,2*i-1,2*j) +X_square(:,2*i,2*j))/4;
        X_test_square_flitered(:,i,j) = (X_test_square(:,2*i-1,2*j-1) ...,
                           +X_test_square(:,2*i,2*j-1) +X_test_square(:,2*i-1,2*j) ...,
                           +X_test_square(:,2*i,2*j))/4;
    end
end
X_filtered = reshape(X_square_filtered, length(Y), 64);
X_test_filtered = reshape(X_test_square_flitered, length(Y_test), 64);
clear X_square X_test_square X_square_filtered X_test_square_flitered;
% The error of training set:
error_filtered_train = 100*sum(classify(X_filtered, X_filtered, Y,'linear') ...,
                       ~=Y)/length(Y)
% The error of testing set:
error_filtered_test  = 100*sum(classify(X_test_filtered, X_filtered, Y,'linear') ...,
                       ~=Y_test)/length(Y_test)

% d)

% Since glmnet can not take Y is not from 1 to n where 1 to n must be
% contious, I remap it to 1-3 in this case.
Y_glmnet = Y;
for i=1:length(Y)
    if Y(i) == 3
      Y_glmnet(i) =1;
    end
    if Y(i) == 5
        Y_glmnet(i) = 2;
    end
    if Y(i) == 8
        Y_glmnet(i) = 3;
    end
```

```
end
Y_test_glmnet = Y_test;
for i=1:length(Y_test)
    if Y_test(i) == 3
      Y_test_glmnet(i) =1;
    end
    if Y_test(i) == 5
        Y_test_glmnet(i) = 2;
    end
    if Y_test(i) == 8
        Y_test_glmnet(i) = 3;
    end
end

coeff = glmnet(X_filtered, Y_glmnet, 'multinomial', glmnetSet);
result = glmnetPredict(coeff, 'class', X_filtered);
% The error of training set:
error_glmnet_train = 100*sum(result(:,length(coeff.lambda))  ...,
                       ~=Y_glmnet)/length(Y_glmnet)

result = glmnetPredict(coeff, 'class', X_test_filtered);
% The error of testing set
error_glmnet_train = 100*sum(result(:,length(coeff.lambda))  ...,
                       ~=Y_test_glmnet)/length(Y_test_glmnet)
```