THU-70250043, Pattern Recognition (Spring 2016) **Homework: 6**

Feature Selection

Lecturer: Changshui Zhang zcs@mail.tsinghua.edu.cn

Student: Qingfu Wen wqf15@mails.tsinghua.edu.cn

Problem 1

Relation between Fisher Criterion and Least Squares

It's interesting to see that under some circumstances, the fisher criterion can be obtained as a special case of least squares. Consider the binary classification problem, let's unify the expression at the very beginning for convenience of the following steps. And you are required to obey the notations given below.

Suppose we have N_1 points of class $\mathcal{C}_1$ and N_2 of class $\mathcal{C}_2$, then the mean vectors of the two classes are given by:

$$\mathbf{m}_1 = \frac{1}{N_1}\sum_{n\in\mathcal{C}_1}\mathbf{x}_n, \qquad \mathbf{m}_2 = \frac{1}{N_2}\sum_{n\in\mathcal{C}_2}\mathbf{x}_n \tag{1}$$

In the lecture notes, we have defined *between-class* covariance matrix and *within-class* covariance matrix:

$$S_B = (\mathbf{m}_2-\mathbf{m}_1)(\mathbf{m}_2-\mathbf{m}_1)^T, \qquad S_W = \sum_{n\in\mathcal{C}_1}(\mathbf{x}_n-\mathbf{m}_1)(\mathbf{x}_n-\mathbf{m}_1)^T + \sum_{n\in\mathcal{C}_2}(\mathbf{x}_n-\mathbf{m}_2)(\mathbf{x}_n-\mathbf{m}_2)^T \tag{2}$$

Now, let's turn to the least square problem. We take the targets for $\mathcal{C}_1$ to be N/N_1 and $\mathcal{C}_2$ to be $-N/N_2$ where $N = N_1 + N_2$(This may be a little confusing, but you will see the reasons of doing so in a short time). Then the sum-of-square error function can be written as:

$$E = \frac{1}{2}\sum_{n=1}^{N}(\mathbf{w}^\mathbf{T}\mathbf{x}_n + w_0 - t_n)^2 \tag{3}$$

Where, $(\mathbf{x}_n, t_n)$ are the points we have. Target t_n equals to N/N_1 or $-N/N_2$ according to its class. Our goal is to estimate $\mathbf{w}$ and w_0.

1.1 Show that the optimal w_0 is: $w_0 = -\mathbf{w}^T\mathbf{m}$.
Set

$$\frac{\partial E}{\partial w_0} = \sum_{n=1}^{N}(\mathbf{w}^\mathbf{T}\mathbf{x}_n + w_0 - t_n) = 0$$

we got

$$\begin{aligned} w_0 &= \frac{1}{N}\sum_{n=1}^{N}(t_n - \mathbf{w^T x}_n) \\ &= \frac{1}{N}\sum_{n=1}^{N}(t_n - \mathbf{w^T m}) \\ &= \frac{1}{N}(\sum_{n\in\mathcal{C}_1}\frac{N}{N_1} + \sum_{n\in\mathcal{C}_2} -\frac{N}{N_2}) - \mathbf{w^T m} \\ &= -\mathbf{w^T m} \end{aligned}$$

1.2 Derive the equation that the optimal $\mathbf{w}$ should obey:

$$(S_W + \frac{N_1 N_2}{N} S_B)\mathbf{w} = N(\mathbf{m_1} - \mathbf{m_2}) \tag{4}$$

Set

$$\begin{aligned} \frac{\partial E}{\partial \mathbf{w}} &= \sum_{n=1}^{N}(\mathbf{w^T x}_n + w_0 - t_n)x_n = 0 \\ \sum_{n=1}^{N}(\mathbf{w^T x}_n + w_0)x_n - \sum_{n=1}^{N} t_n x_n &= 0 \\ (S_W + \frac{N_1 N_2}{N} S_B)\mathbf{w} - N(\mathbf{m_1} - \mathbf{m_2}) &= 0 \end{aligned}$$

Thus

$$(S_W + \frac{N_1 N_2}{N} S_B)\mathbf{w} = N(\mathbf{m_1} - \mathbf{m_2})$$

1.3 Show me that $\mathbf{w}$ satisfies: $\mathbf{w} \propto S_W^{-1}(\mathbf{m_2} - \mathbf{m_1})$ from equation (4), which means we've got the same form as that of fisher criterion.

SOLUTION:
$\because S_B\mathbf{w} = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^T \propto (\mathbf{m_2} - \mathbf{m_1})$
$\therefore S_W\mathbf{w} \propto (\mathbf{m_2} - \mathbf{m_1})$.
Thus, $\mathbf{w} \propto S_W^{-1}(\mathbf{m_2} - \mathbf{m_1})$.

Problem 2

Fisher's discriminant for multiple classes

Consider the generalization of the Fisher discriminant to $K > 2$ classes, and assume that the dimensionality of the input space is greater than the number K of classes. Next, we introduce $D' > 1$ linear 'features' $y_k = \mathbf{w}_k^T\mathbf{x}$. The weight vectors $\{\mathbf{w}_k\}'s$ can be considered to be the columns of a matrix $\mathbf{W}$, so that:

$$\mathbf{y} = \mathbf{W}^T\mathbf{x}$$

Where, $\mathbf{x} \in \mathcal{R}^D$ and $\mathbf{y} \in \mathcal{R}^{D'}$. By this means, we have projected the D-dimensional $\mathbf{x}$-space onto the D'-dimensional $\mathbf{y}$-space, in which we can better separate the data.

The generalization of the *within-class* covariance matrix to the case of K classes is:

$$S_W = \sum_{k=1}^{K} \sum_{n \in \mathcal{C}_k} (\mathbf{x}_n - \mathbf{m}_k)(\mathbf{x}_n - \mathbf{m}_k)^T \tag{5}$$

Where $\mathbf{m_k} = \frac{1}{N_k} \sum_{n \in \mathcal{C}_k} \mathbf{x}_n$.

The total covariance matrix is:

$$S_T = \sum_{n=1}^{N} (\mathbf{x}_n - \mathbf{m})(\mathbf{x}_n - \mathbf{m})^T \tag{6}$$

Where $\mathbf{m} = \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}_n = \frac{1}{N} \sum_{k=1}^{K} N_k \mathbf{m}_k$.

2.1 Decompose the total covariance matrix S_T into *within-class* covariance matrix S_W and *between-class* covariance matrix S_B, and show that S_B has the form: $S_B = \sum_{k=1}^{K} N_k (\mathbf{m}_k - \mathbf{m})(\mathbf{m}_k - \mathbf{m})^T$

$$\begin{aligned}
S_B =& S_T - S_W \\
=& \sum_{n=1}^{N} (\mathbf{x}_n - \mathbf{m})(\mathbf{x}_n - \mathbf{m})^T - \sum_{k=1}^{K} \sum_{n \in \mathcal{C}_k} (\mathbf{x}_n - \mathbf{m}_k)(\mathbf{x}_n - \mathbf{m}_k)^T \\
=& \sum_{n=1}^{N} (\mathbf{x}_n \mathbf{x}_n^T - \mathbf{m}\mathbf{m}^T) - \sum_{k=1}^{K} \sum_{n \in \mathcal{C}_k} (\mathbf{x}_n \mathbf{x}_n^T - \mathbf{m}_k \mathbf{m}_k^T) \\
=& \sum_{k=1}^{K} N_k \mathbf{m}_k \mathbf{m}_k^T - N \mathbf{m}\mathbf{m}^T \\
=& \sum_{k=1}^{K} N_k (\mathbf{m}_k - \mathbf{m})(\mathbf{m}_k - \mathbf{m})^T
\end{aligned}$$

2.2 Write down the *within-class* covariance matrix s_W and *between-class* covariance matrix s_B of the projected D'-dimensional $\mathbf{y}$-space.

$$s_W = \sum_{k=1}^{K} \sum_{n \in \mathcal{C}_k} W^T (\mathbf{x}_n - \mathbf{m}_k)(\mathbf{x}_n - \mathbf{m}_k)^T W$$

$$s_B = \sum_{k=1}^{K} N_k W^T (\mathbf{m}_k - \mathbf{m})(\mathbf{m}_k - \mathbf{m})^T W$$

2.3 Many possible choices of criterion can be implemented to decide the weight matrix $\mathbf{W}$, more than 5 examples are shown in Professor Zhang's ppt. Here, we are using another criterion:

$$J(W) = \frac{\Pi_{diag} s_B}{\Pi_{diag} s_W} \tag{7}$$

Where, $\Pi_{diag} A$ means multiplication of the diagonal elements of matrix A.

Represent $J(\mathbf{W})$ explicitly with $\mathbf{W}$, S_W and S_B.

SOLUTION:

$$J(W) = \frac{\Pi_{diag} s_B}{\Pi_{diag} s_W} = \frac{\Pi_{diag} \mathbf{W}^T S_B \mathbf{W}}{\Pi_{diag} \mathbf{W}^T s_W \mathbf{W}}$$

2.4 As is stated above, we now want to project the original data space onto a space with D' dimensions, while at the same time trying to maximize $J(\mathbf{W})$ represented by equation (7). Write down the equations that columns of weight matrix $\mathbf{W}$ should obey(which means the selected projection directions).

2.5 As is stated in the problem, we have K classes in all and we are trying to find D' linear 'features'(or projection directions) by maximizing $J(\mathbf{W})$. How many such 'features' at most are we able to find? Give me your reason.

Problem 3

An intuitive understanding between features and error-rates

Let's review the definition of binary-class Bayesian error-rate at first.

In classification problems, our goal is always to make as few misclassifications as possible. We need a rule that assigns each value of $\mathbf{x}$ to one of the available classes. Such a rule will divide the input space into regions $\mathcal{R}_k$ called *decision regions*, one for each class, such that all points in $\mathcal{R}_k$ are assigned to class $\mathcal{C}_k$. Take binary classification as an example: A mistake occurs when an input vector belonging to class $\mathcal{C}_1$ is assigned to class $\mathcal{C}_2$ or vice versa. The error-rate is then given by:

$$p(mistake) = p(\mathbf{x} \in \mathcal{R}_1, \mathcal{C}_2) + p(\mathbf{x} \in \mathcal{R}_2, \mathcal{C}_1) \tag{8}$$

$$= \int_{\mathcal{R}_1} p(\mathbf{x}, \mathcal{C}_2) d\mathbf{x} + \int_{\mathcal{R}_2} p(\mathbf{x}, \mathcal{C}_1) d\mathbf{x} \tag{9}$$

To minimize the above error-rate, we have to make use of posterior distribution: If $p(\mathcal{C}_1|\mathbf{x}) > p(\mathcal{C}_2|\mathbf{x})$, then we assign that $\mathbf{x}$ to class $\mathcal{C}_1$, and vice versa. Thus leading to the Bayesian error-rate.

3.1 Suppose we consider a K-class problem, derive the corresponding error rate as that of equation (9).

SOLUTION:

$$\begin{aligned} p(mistake) &= \sum_{i=1}^{K} \sum_{j \neq i} \int_{\mathcal{R}_i} p(\mathbf{x}, \mathcal{C}_j) d\mathbf{x} \\ &= \sum_{i=1}^{K} (1 - \int_{\mathcal{R}_i} p(\mathbf{x}, \mathcal{C}_i) d\mathbf{x}) \end{aligned}$$

3.2 Let $x_i, i = 1, 2, 3$ be independent binary-valued features, and $P(x_i = 1|w_1) = \alpha_i$, $P(x_i = 1|w_2) = \beta_i$, $P(w_1) = P(w_2)$. Assume that $\beta_1 - \alpha_1 > \beta_2 - \alpha_2 > \beta_3 - \alpha_3$ and $\alpha_i < \beta_i$, $\forall i = 1, 2, 3$.

Prove that the Bayesian error-rates with only one feature will satisfy $e(x_1) < e(x_2) < e(x_3)$. Give me your explanation of this phenomenon based on the three features.

SOLUTION:

$P(w_1) = P(w_2) = 0.5$,$P(x_i = 1) = P(x_i = 1|w_1)P(w_2) + P(x_i = 1|w_2)P(w_2) = 0.5 * (\alpha_i + \beta_i)$

$P(w_1|x_i = 1) = \frac{P(w_1, x_i=1)}{P(x_i=1)} = \frac{0.5*\alpha_i}{0.5*(\alpha_i+\beta_i)} = \frac{\alpha_i}{\alpha_i+\beta_i}$,$P(w_2|x_i = 1) = \frac{\beta_i}{\alpha_i+\beta_i}$.

$P(w_1|x_i = 0) = \frac{1-\alpha_i}{2-\alpha_i-\beta_i}$, $P(w_2|x_i = 0) = \frac{1-\beta_i}{2-\alpha_i-\beta_i}$.

The classification criterion is if it belongs to w_2 when $x_i = 1$, otherwise, it belongs to w_1.

$$\begin{aligned} e(x_i) =& P(w_1|x_i = 1)P(x_i = 1) + P(w_2|x_i = 0)P(x_i = 0) \\ =& \frac{\alpha_i}{2} + \frac{1 - \beta_i}{2} \\ =& \frac{1 - (\beta_i - \alpha_i)}{2} \end{aligned}$$

Since $\beta_1 - \alpha_1 > \beta_2 - \alpha_2 > \beta_3 - \alpha_3$, $e(x_1) < e(x_2) < e(x_3)$.
For only one feature, the feature with largest $\beta_i - \alpha_i$ has lowest error rate.

3.3 With the following parameters:

$$\alpha_1 = 0.1, \qquad \alpha_2 = 0.05, \qquad \alpha_3 = 0.01$$
$$\beta_1 = 0.9, \qquad \beta_2 = 0.8, \qquad \beta_3 = 0.7$$

Calculate $e(x_1)$, $e(x_2)$, $e(x_3)$; $e(x_1, x_2)$, $e(x_1, x_3)$, $e(x_2, x_3)$. Compare the values of different error-rates and present your explanation from the view of feature selection.

SOLUTION:
$e(x_1) = \frac{1-(0.9-0.1)}{2} = 0.1$,$e(x_2) = \frac{1-(0.8-0.05)}{2} = 0.125$,$e(x_3) = \frac{1-(0.7-0.01)}{2} = 0.155$.
$P(w_1, x_i = 1, x_j = 1) = \alpha_i\alpha_j$,$P(w_1, x_i = 1, x_j = 0) = \alpha_i(1 - \alpha_j)$.
$P(w_1, x_i = 0, x_j = 1) = (1 - \alpha_i)\alpha_j$,$P(w_2, x_i = 0, x_j = 0) = (1 - \beta_i)(1 - \beta_j)$.
$e(x_1, x_2) = 0.1 * 0.05 + 0.1 * 0.95 + 0.9 * 0.05 + 0.2 * 0.1 = 0.165$.
$e(x_1, x_3) = 0.1 * 0.01 + 0.1 * 0.99 + 0.9 * 0.01 + 0.1 * 0.3 = 0.139$.
$e(x_2, x_3) = 0.05 * 0.01 + 0.05 * 0.99 + 0.95 * 0.01 + 0.2 * 0.3 = 0.1195$.
Since $e(x_1) < e(x_2, x_3) < e(x_2) < e(x_1, x_3) < e(x_3) < e(x_1, x_2)$, we can see that feature selection is not easy to handle.

Problem 4: Programming

Fisher discriminant on the ORL Database of Faces

In this task, you are required to cope with the ORL Database of Faces, which contains a set of face images. This is a famous database, there are ten different images of each of 40 distinct persons. An *orl_faces.mat* file is provided, with a 400×4096 matrix *data* and a 400×1 vector *label*. Each row of *data* corresponds to a face, you can view the i-th face by matlab command "*imshow(uint8(reshape(data(i, :), 64, 64)))* ". Each element of *label* ranges from 1 to 40, indicating the person's id.

4.1 Use Fisher's linear discriminant with "one-versus-one" strategy and Fisher's discriminant for multiple classes discussed in **Problem2** to reduce the dimensions of the feature space separately(The original feature space is 4096).

Hint: The standard fisher linear discriminant is based on binary-class problem, so each round we implement this criterion, we will get an optimal direction $\mathbf{w}$ *determined by class* $\mathcal{C}_i$ *and class* $\mathcal{C}_j$*. Thus, "one-versus-one" strategy means that we should implement fisher linear discriminant on every two classes: since we have 40 classes in all, this strategy will lead to* $C_{40}^2 = 780$ *binary discriminant functions in total. Each point is then classified according to a majority vote amongst the discriminant functions. But if your laptop is slow in the computation of matrix inverse, you can just experiment on 10 of the 40 classes.*

SOLUTION:
I don't know how to reduce the dimension using "one-versus-one" strategy and only implemented Fisher's discriminant for multiple classes, you may see the matlab code of `fisherLDA.m`.

4.2 Design a classifier based on the new feature space.

Hint: Recall what you have learned in the previous lectures, you now have a few choices of classifiers such as softmax regression, bayesian inference, SVM and KNN. With this course going on, you'll learn more classifiers, our suggestion for you is to keep these methods in mind and become more skilled at them.

SOLUTION:
Based on Fisher's linear discriminant analysis above, I implemented kNN(k=3) and SVM(linear kernel) classifiers for new feature space with different dimension. The table below shows classification accuracy of two classifiers.

dim	kNN	SVM
1	0.925	0.9875
2	0.925	0.9875
4	0.925	0.9875
8	0.925	0.9875
16	0.925	0.9875
32	0.925	0.9875
64	0.925	0.9875
128	0.925	0.9875
256	0.925	0.9875
512	0.925	0.9875

Table 1: Classification accuracy of different dimension

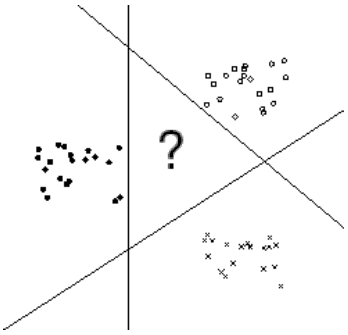


Figure 1: "one-versus-one" strategy

4.3 Discuss the results and compare the two strategy of reducing data dimensions. What's more, the "one-versus-one" strategy may lead to some ambiguous results, you can illustrate this fact with some simple hand-drawn images.
SOLUTION:
From figure 2, we can see that "one-versus-one" strategy may have some unknown area that does not belong to any classes.

Problem 5: Programming

Feature selection with L1-norm regularization

This programming task seems to be heavy, but don't be afraid. This is actually an easy work as long as you follow the steps and hints provided.

In this problem, let's consider a regularized approach to feature selection in a simple regression context. Unlike fisher criterion, we are not going to reduce dimensions, but just select features from the existed ones. Suppose we have training inputs $X = (\mathbf{x}_1, ..., \mathbf{x}_n)^T$ and corresponding outputs $\mathbf{y} = (y_1, ..., y_2)^T$, where $\mathbf{x}_i \in \mathcal{R}^D$ and $y_i \in \mathcal{R}$. We want to train a linear predictor $\hat{y}(\mathbf{x}; \mathbf{w}) = \mathbf{w}^T \phi(\mathbf{x})$ with $\phi(\mathbf{x})$ indicating the M features $\phi(\mathbf{x}) = (\phi_1(\mathbf{x}), ..., \phi_M(\mathbf{x}))^T$. The objective is regularized least-squares as following:

$$J(\mathbf{w}; \lambda) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{2} (y_i - \mathbf{w}^T \phi(\mathbf{x}_i))^2 + \lambda ||\mathbf{w}||_1 \tag{10}$$

Where $||\mathbf{w}||_1$ is the L1-norm:

$$||\mathbf{w}||_1 = \sum_{k=1}^{M} |w_k| \tag{11}$$

Thus, our task is to minimize the regularized objective in order to seek for the optimal parameters $\hat{\mathbf{w}} = \hat{\mathbf{w}}(\lambda)$:

$$\hat{\mathbf{w}} = argmin_{\mathbf{w} \in \mathcal{R}^M} J(\mathbf{w}; \lambda) \tag{12}$$

Implementation of the L1-norm will lead to a sparse $\mathbf{w}$. What's more, with λ increasing, more elements of weight vector $\mathbf{w}$ are forced to zero, which means the corresponding features are ignored in the predictor.

Coordinate descent will be used to solve this L1-regularized least-squares problem. In this approach, we adjust one parameter at a time so as to minimize the objective while keeping the remaining parameters fixed:

$$\hat{w}_k = argmin_{w_k} J(\mathbf{w}; \lambda) \tag{13}$$

With k iterating over all indices {1,...,M}, and repeating the process for many times, parameter $\mathbf{w}$ asymptotically converges to the global minimum of the convex objective (Not familiar with definition of convex? wiki or google).

Since the L1 regularization is not smooth, we can't take derivatives as the differentiable functions to minimize the objective. Instead, we make use of the *subdifferential of a convex function*, which is defined as:

$$\partial f(w) = \{s | f(w + \Delta) \geq f(w) + s\Delta, \forall \Delta \in \mathcal{R}\} \tag{14}$$

This is a set-valued generalization of the normal derivative and reduces to the normal derivative $\partial f(w) = \{\frac{\partial f(w)}{\partial w}\}$ whenever f is differentiable. Taking the absolute value function $f(w) = |w|$ as an example:

$$\partial f(w) = \begin{cases} \{-1\}, & w < 0 \\ [\text{-}1, +1], & w = 0 \\ \{+1\}, & w > 0 \end{cases}$$

We will use the following result from non-smooth analysis:

Optimality Condition: $\hat{w}$ is a global minimizer of a convex function $f(w)$ if and only if $0 \in \partial f(\hat{w})$.

For example, the optimality condition $0 \in \partial f(w)$ for the absolute value function $f(w) = |w|$ holds if and only if $w = 0$. Hence, $w = 0$ is the unique global minimizer of $|w|$.

Below, we will guide you through the analysis to solve the L1-norm regularized least-squares problem.

5.1 Show that the sub-differential of $J(\mathbf{w}; \lambda)$ with respect to parameter w_k is:

$$\partial_{w_k} J(\mathbf{w}; \lambda) = (a_k w_k - c_k) + \lambda \partial_{w_k} |w_k| = \begin{cases} \{a_k w_k - (c_k + \lambda)\}, & w_k < 0 \\ [\text{-}c_k\text{-}\lambda, \text{-}c_k + \lambda], & w_k = 0 \\ \{a_k w_k - (c_k - \lambda)\}, & w_k > 0 \end{cases}$$

With:

$$a_k = \frac{1}{n} \sum_{i=1}^{n} \phi_k^2(\mathbf{x}_i) \tag{15}$$

$$c_k = \frac{1}{n} \sum_{i=1}^{n} \phi_k(\mathbf{x}_i)(y_i - \mathbf{w}_{-k}^T \phi_{-k}(\mathbf{x}_i)) \tag{16}$$

Where $\mathbf{w}_{-k}$(respectively ϕ_{-k}) denote the vector of all parameters (features) except for parameter w_k (feature ϕ_k). Observe equations (15) and (16), you can find that a_k's are non-negative and constant parameters while each c_k depends on all parameters except for the parameter w_k that we are to update. Try to interpret the coefficient c_k.

SOLUTION:

$$\begin{aligned} J(\mathbf{w};\lambda) &= \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}(y_i - \mathbf{w}^T\phi(\mathbf{x}_i))^2 + \lambda||\mathbf{w}||_1 \\ &= \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}(y_i - \mathbf{w}_{-k}^T\phi_{-k}(\mathbf{x}_i) - w_k\phi_k(\mathbf{x}_i))^2 + \lambda||\mathbf{w}||_1 \\ &= \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}\left[(y_i - \mathbf{w}_{-k}^T\phi_{-k}(\mathbf{x}_i))^2 - 2w_k\phi_k(\mathbf{x}_i)(y_i - \mathbf{w}_{-k}^T\phi_{-k}(\mathbf{x}_i)) + w_k^2\phi_k^2(\mathbf{x}_i)\right] + \lambda||\mathbf{w}||_1 \\ &= \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}(y_i - \mathbf{w}_{-k}^T\phi_{-k}(\mathbf{x}_i))^2 - c_kw_k + \frac{1}{2}a_kw_k^2 + \lambda||\mathbf{w}||_1 \end{aligned}$$

Thus

$$\partial_{w_k}J(\mathbf{w};\lambda) = (a_kw_k - c_k) + \lambda\partial_{w_k}|w_k|$$

and

$$\partial_{w_k}|w_k| = \begin{cases} \{-1\}, & w_k < 0 \\ [\text{-}1,\ +1], & w_k = 0 \\ \{+1\}, & w_k > 0 \end{cases}$$

Finally

$$\partial_{w_k}J(\mathbf{w};\lambda) = (a_kw_k - c_k) + \lambda\partial_{w_k}|w_k| = \begin{cases} \{a_kw_k - (c_k + \lambda)\}, & w_k < 0 \\ [\text{-}c_k\text{-}\lambda,\ \text{-}c_k\text{+}\lambda], & w_k = 0 \\ \{a_kw_k - (c_k - \lambda)\}, & w_k > 0 \end{cases}$$

5.2 Solve the non-smooth optimality condition for $\hat{w}_k$ s.t. $0 \in \partial_{w_k}J(\hat{w}_k)$. It's helpful to consider each of the following cases separately:
(a) $c_k < -\lambda$
(b) $c_k \in [-\lambda, +\lambda]$
(c) $c_k > +\lambda$
In each case, provide a plot $\partial_{w_k}J(w_k)$ versus w_k and label the zero-crossing $\hat{w}_k$. Then, with the help of those plots, you can express $\hat{w}_k$ as a function of a_k, c_k and λ. Finally, provide a plot of $\hat{w}_k$ versus c_k. What role does the regularization parameter λ play in this context relative to the coefficient c_k ?

Hint: There is no need for you to spend too much time generating those plots required above. Since those plots are just small tools helping you understand the final equations. You can just provide me with your hand-drawn plots, either inserted in your report, or included in the codes' folder.

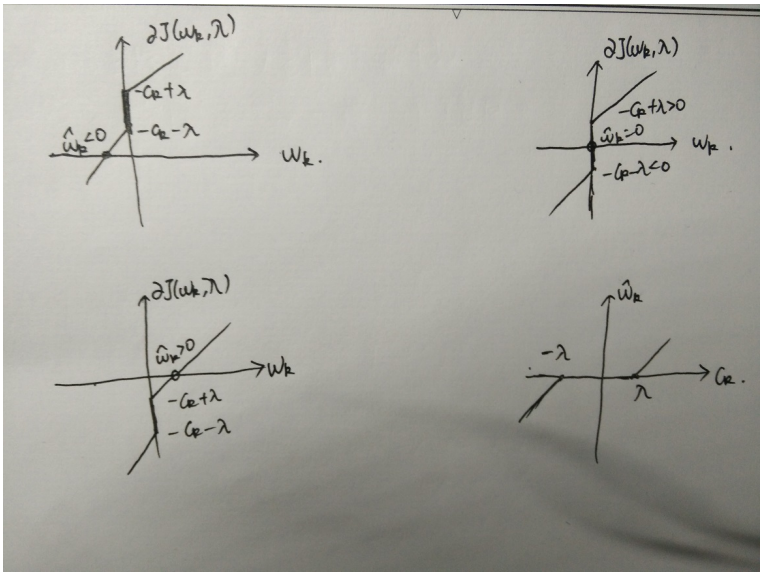
SOLUTION:

(a) if $c_k < -\lambda$, $c_k + \lambda < 0$. By solving $a_kw_k - (c_k + \lambda) = 0$, we have

$$\hat{w_k} = \frac{c_k + \lambda}{a_k} < 0$$

(b) if $c_k \in [-\lambda, +\lambda]$, $-\lambda < c_k < \lambda$. Thus, $\partial_{w_k}J(\mathbf{w};\lambda)$ occurs when

$$\hat{w_k} = 0$$



(c) if $c_k > \lambda$, $-c_k + \lambda < 0$. By solving $a_k w_k - (c_k - \lambda) = 0$, we have

$$\hat{w_k} = \frac{c_k - \lambda}{a_k} > 0$$

Above all, we can write w_k into a function of c_k form.

$$\hat{w_k} = \begin{cases} \frac{c_k+\lambda}{a_k}, & c_k < -\lambda \\ 0, & c_k \in [-\lambda, +\lambda] \\ \frac{c_k-\lambda}{a_k}, & c_k > \lambda \end{cases}$$

we can see that, if $|c_k| < \lambda$, w_k goes to 0, which means $\mathbf{w}$ is sparse; if $|c_k| > \lambda$, $w_k \neq 0$. Thus, the larger λ is, the sparser $\mathbf{w}$ will be.

5.3 We have provided you with training and testing data in the file **least_sq.mat**. Load this file into MATLAB workspace and you will find 4 structures **test**, **train_large**, **train_mid**, **train_small** each containing an $n \times M$ matrix X and a $n \times 1$ vector $\mathbf{y}$.

For simplicity, we only consider the linear regression, say $\phi(\mathbf{x}) = \mathbf{x}$. Write a MATLAB subroutine based on the skeleton **least_sq_L1.m** to solve for the optimal parameter $\hat{\mathbf{w}}$ given the training data $(X, \mathbf{y})$.

Hint: The skeleton code is just a suggestion, you are not strictly required to follow it as long as you can realize the algorithm.

What's more, before writing the code, review the equations derived in 5.2, you may find that the expression for $\hat{w}_k$ considered under different cases can be included into just one simple equation, which can accelerate your code by eliminating "if-else" snippets.

5.4 Complete the skeleton **mainFunc.m** to run your algorithm for a sequence of λ values [0.01: 0.01: 2.0]. And we use $\mathbf{w}_0 = (X^T X)^{-1} X^T \mathbf{y}$, the usual least-squares parameter estimation as the initial state for $\mathbf{w}$. Also, a function **least_sq_multi.m** is provided, you should read the code carefully.

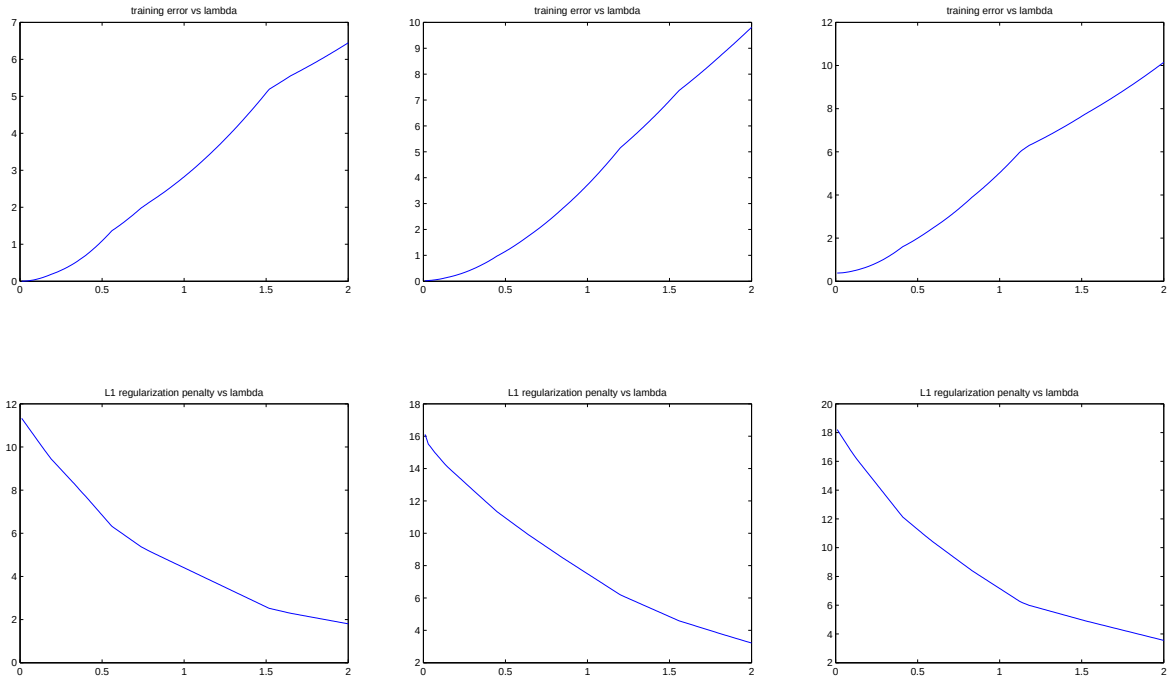
Plot each of the following versus λ:
(a) the training error $J(\hat{\mathbf{w}}(\lambda); 0)$

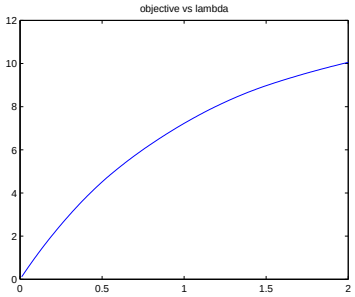
(b) the regularization penalty $||\hat{\mathbf{w}}(\lambda)||_1$
(c) the minimized objective $J(\hat{\mathbf{w}}(\lambda); \lambda)$
(d) the number of non-zero parameters $||\hat{\mathbf{w}}(\lambda)||_0$(the L0-norm)
(e) the test error

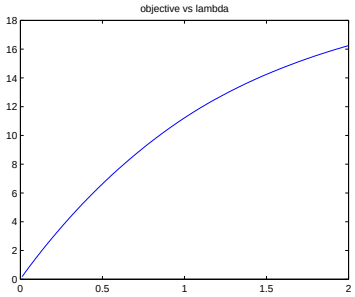
Run this experiment for each of the three training set **train_large**, **train_mid** and **train_small**, then test on set **test**. Comment on the behavior of each of these quantities as a function of λ. For each training set, what value of λ minimizes the test error? How does this vary with the size of the training set? How could we estimate from the training data the appropriate value of λ to use so as to approximately minimize the test error?

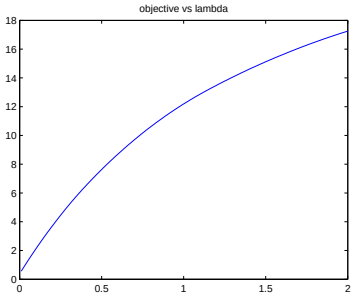
Hint: Above all, we've guided you through the basic idea of optimizing the L1-norm problem. L1-regularization is greatly related to the field of compress sensing or sparse representation which was a hot topic for a few years, to learn more, wiki or google.

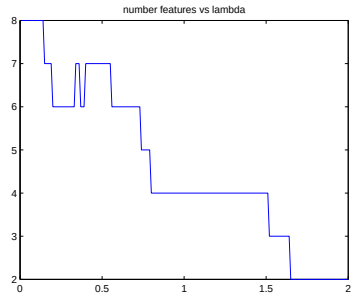
SOLUTION:
we can see that when λ increases, the training error increase and regularization penalty consequently decrease.
The objective function increases smoothly when λ increases.
The test error gets local minimum for a certain λ, with multi-times running of different λ, we can find the minimum of test error.
When λ gets larger enough, L0 norm becomes 0, which means w_k is 0. The larger λ is, the more 0s are in $\mathbf{w}$.

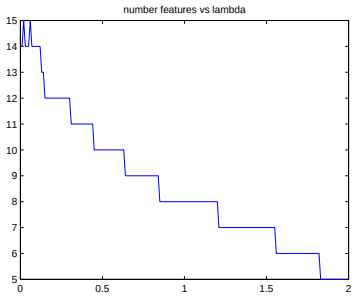


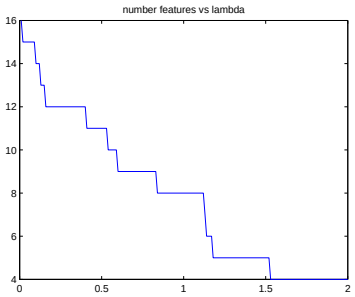


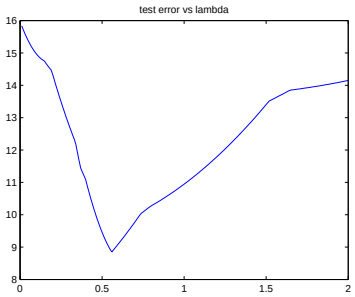


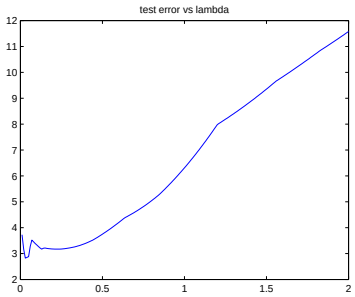












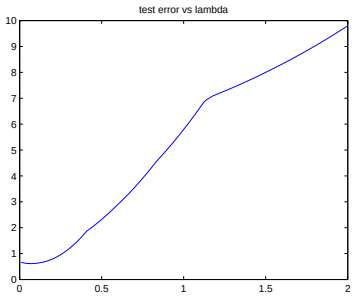


Figure 2: Small

Figure 3: Medium

Figure 4: Large