Korea Advanced Institute of Science and Technology
School of Electrical Engineering
EE488 Intro. to Machine Learning, Spring 2018

Issued: Apr. 6, 2018
Due: Apr. 13, 2018

Assignment II-part1

Policy

Group study is encouraged; however, assignment that you hand-in must be of your own work. Anyone suspected of copying others will be penalized. The homework will take considerable amount of time so start early.

1. **Logistic regression**: We consider here a discriminative approach for solving the classification problem illustrated in Figure 1. We attempt to solve the binary classification task depicted in

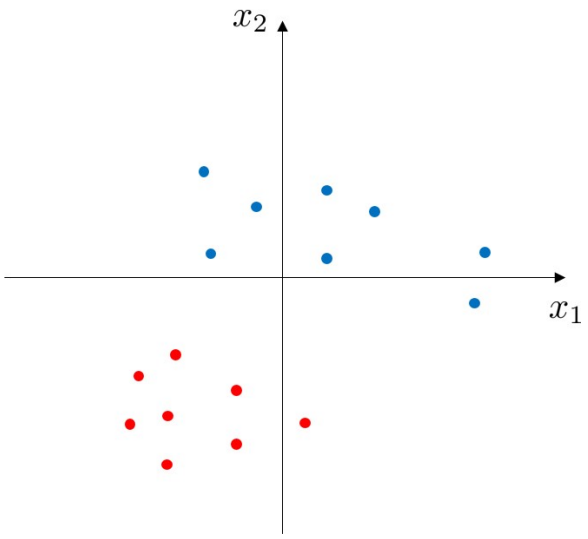


Figure 1: The 2-dimensional labeled training set, where blue dots corresponds to class $y = 1$ and red dots corresponds to class $y = 0$.

Figure 1 with the simple linear logistic regression model

$$p(y = 1|\boldsymbol{x}, \boldsymbol{\theta}) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2) = \frac{1}{1 + exp(-\theta_0 - \theta_1 x_1 - \theta_2 x_2)}$$

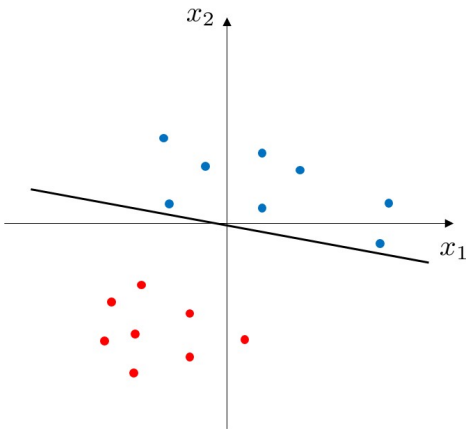
Notice that the training data can be separated with zero training error with a linear separator. Consider training regularized linear logistic regression models where we try to maximize

$$\sum_{i=1}^{n} \log(P(y_i|x_i, \theta_0, \theta_1, \theta_2)) - C\theta_j^2$$

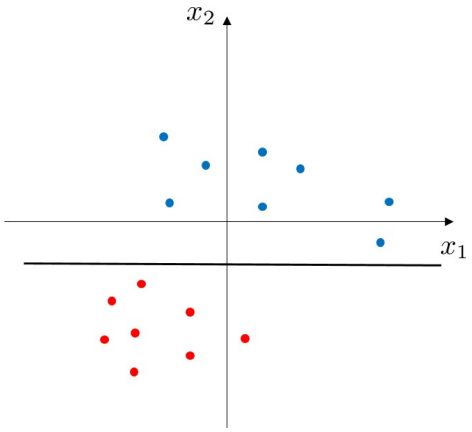
for very large C. The regularization penalty used in penalized conditional log-likelihood estimation is $-C\theta_j^2$ where $j = \{0, 1, 2\}$. In other words, only one of the parameters is regularized

in each case. Given the training data in Figure 1, Draw the decision line with regularization of each parameter θ_j. State whether the training error increases or stays the same (zero) for each θ_j for very large C. Provide a brief justification for each of your answers.

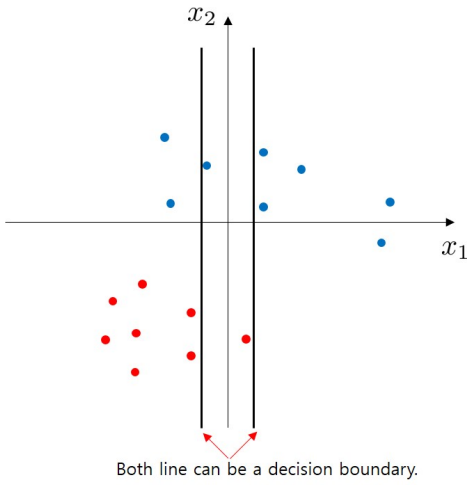
(solution)
When regularization is placed on θ_0, it is restricted to a small value. Assume $\theta_0 \approx 0$, then the decision boundary will pass through a point close to the origin. Decision boundary is a set of points where the conditional probability of different classes are equal: $p(y = 1|\boldsymbol{x}, \boldsymbol{\theta}) = p(y = 0|\boldsymbol{x}, \boldsymbol{\theta}) = 0.5$, and when $(x_1, x_2) = (0, 0)$, $g(\theta_0 + \theta_1 x_1 + \theta_2 x_2) = g(0) = 0.5$. Therefore, the θ_0 regularized logistic regression will find the decision boundary that passes through $(0, 0)$, and the training error will stay the same(0 error).



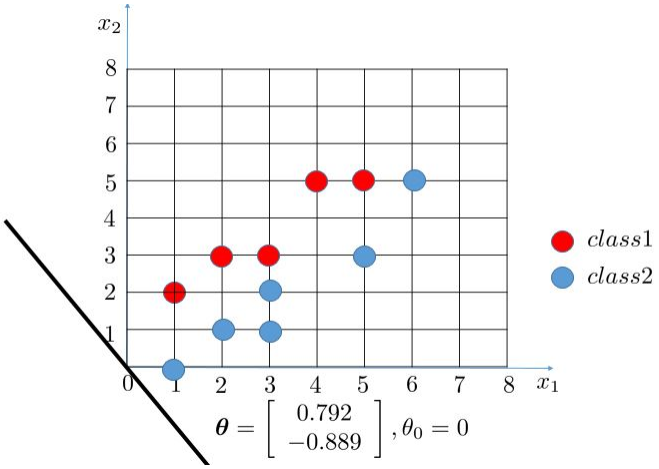
When regularization is placed on θ_1, θ_1 is restricted to a small value. Assume $\theta_1 \approx 0$, then the decision boundary will be a horizontal line: x_1 will not influence classification.



When regularization is placed on θ_2, θ_2 is restricted to a small value. Assume $\theta_2 \approx 0$, then the decision boundary will be a vertical line: There will be three classification errors.



2. **Fisher's Linear Discriminative Analysis (FDA)**:
Given 11 training data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{11}$ where $\boldsymbol{x}_i \in \mathbb{R}^2$ and label $y_i \in \{-1, +1\}$ respectively as shown in the figure below. Fisher's linear discriminant is considered to project each $\boldsymbol{x}_i$ on to a line such that prediction y is equal to $sign(\boldsymbol{\theta}^T \boldsymbol{x} + \theta_0)$. Here $\boldsymbol{\theta}$ is obtained by maximizing $J(\boldsymbol{\theta}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2} = \frac{\boldsymbol{\theta}^T \mathbf{S}_B \boldsymbol{\theta}}{\boldsymbol{\theta}^T \boldsymbol{S}_\theta \boldsymbol{\theta}}$ where m_i, s_i^2 are the projected mean and the variance of ith class respectively. The between-class covariance matrix $\boldsymbol{S}_B$ and within-class covariance matrix $\boldsymbol{S}_\theta$ are respectively defined as follows: $\boldsymbol{S}_B = (\boldsymbol{m}_2 - \boldsymbol{m}_1)(\boldsymbol{m}_2 - \boldsymbol{m}_1)^T$ and $\boldsymbol{S}_\theta = \sum_{n \in C_1} (\boldsymbol{x}_n - \boldsymbol{m}_1)(\boldsymbol{x}_n - \boldsymbol{m}_1)^T + \sum_{n \in C_2} (\boldsymbol{x}_n - \boldsymbol{m}_1)(\boldsymbol{x}_n - \boldsymbol{m}_2)^T$.



(i) Set $\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = 0$ to find the maximum of $J(\boldsymbol{\theta})$ and estimate the projection directions $\boldsymbol{\theta}$.
(solution)
The mean vector of two data samples

$$\boldsymbol{m}_1 = \frac{1}{N_1} \sum_{i \in C_1} \boldsymbol{x}_i = \begin{bmatrix} 3.0000 \\ 3.6000 \end{bmatrix},$$

$$\boldsymbol{m}_2 = \frac{1}{N_2} \sum_{i \in C_2} \boldsymbol{x}_i = \begin{bmatrix} 3.3333 \\ 2.0000 \end{bmatrix}.$$

The within-class co-variance matrices

$$\begin{aligned}\boldsymbol{S_\theta} = \boldsymbol{S}_{-1} + \boldsymbol{S}_{+1} &= 4\begin{bmatrix} V[x_1] & Cov[x_1,x_2] \\ Cov[x_1,x_2] & V[x_2] \end{bmatrix}_{-1} + 5\begin{bmatrix} V[x_1] & Cov[x_1,x_2] \\ Cov[x_1,x_2] & V[x_2] \end{bmatrix}_{+1} \\ &= \begin{bmatrix} 10.0 & 8.0 \\ 8.0 & 7.2 \end{bmatrix}_{-1} + \begin{bmatrix} 17.3 & 16.0 \\ 16.0 & 16.0 \end{bmatrix}_{+1} = \begin{bmatrix} 27.3 & 24.0 \\ 24.0 & 23.2 \end{bmatrix}\end{aligned}$$

The inverse of $\boldsymbol{S}_\theta$ is $\boldsymbol{S}_\theta^{-1} = \begin{bmatrix} 0.3991 & -0.4128 \\ -0.4128 & 0.4702 \end{bmatrix}$.

Set the derivative

$$\begin{aligned}\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} &= 0 \\ &= (\boldsymbol{\theta}^T \boldsymbol{S_\theta} \boldsymbol{\theta}) \frac{\partial (\boldsymbol{\theta}^T \boldsymbol{S}_B \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} - (\boldsymbol{\theta}^T \boldsymbol{S}_B \boldsymbol{\theta}) \frac{\partial (\boldsymbol{\theta}^T \boldsymbol{S_\theta} \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = 0 \\ &= (\boldsymbol{\theta}^T \boldsymbol{S_\theta} \boldsymbol{\theta}) 2\boldsymbol{S}_B \boldsymbol{\theta} - (\boldsymbol{\theta}^T \boldsymbol{S}_B \boldsymbol{\theta}) 2 S_{\boldsymbol{\theta}} \boldsymbol{\theta} = 0\end{aligned}$$

$$\begin{aligned}\rightarrow (\boldsymbol{\theta}^T \boldsymbol{S}_B \boldsymbol{\theta}) S_{\boldsymbol{\theta}} \boldsymbol{\theta} &= (\boldsymbol{\theta}^T \boldsymbol{S_\theta} \boldsymbol{\theta})(\boldsymbol{m}_2 - \boldsymbol{m}_1)(\boldsymbol{m}_2 - \boldsymbol{m}_1)^T \boldsymbol{\theta} \\ \boldsymbol{S_\theta} \boldsymbol{\theta} &= \frac{\boldsymbol{\theta}^T S_{\boldsymbol{\theta}} \boldsymbol{\theta} (\boldsymbol{m}_2 - \boldsymbol{m}_1)^T \boldsymbol{\theta}}{\boldsymbol{\theta}^T \boldsymbol{S}_B \boldsymbol{\theta}} (\boldsymbol{m}_2 - \boldsymbol{m}_1) \\ &= \alpha(\boldsymbol{m}_2 - \boldsymbol{m}_1)\end{aligned}$$

$$\begin{aligned}\rightarrow \boldsymbol{\theta} &= \alpha \boldsymbol{S}_{\boldsymbol{\theta}}^{-1} (\boldsymbol{m}_2 - \boldsymbol{m}_1) \\ &= \begin{bmatrix} 0.3991 & -0.4128 \\ -0.4128 & 0.4702 \end{bmatrix} \begin{bmatrix} 0.3333 \\ -1.6000 \end{bmatrix} \\ &= \begin{bmatrix} 0.7936 \\ -0.8899 \end{bmatrix}.\end{aligned}$$

As long as the line has the right direction, its exact position does not matter.

(ii) Scatters plots the data samples and computes output $y_n = sign(\boldsymbol{\theta}^T \boldsymbol{x}_n + \theta_0)$ and represents class -1 samples in red while the class +1 samples in blue.

Last step is to compute the actual 1D vector $\boldsymbol{y}$ as follows as:

$$\begin{aligned}\boldsymbol{y}^{-1} &= sign(\boldsymbol{\theta}^T \boldsymbol{x} + \theta_0) \\ &= sign\left(\begin{bmatrix} 0.7936 & -0.8899 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 3 & 5 & 5 \end{bmatrix} \right) \\ &= sign\left(\begin{bmatrix} -0.9862, -1.0826, -0.2890, -1.2752, -0.4817 \end{bmatrix} \right) \\ &= \begin{bmatrix} -1, -1, -1, -1, -1 \end{bmatrix}\end{aligned}$$

$$
\begin{aligned}
\boldsymbol{y}^{+1} &= sign(\boldsymbol{\theta}^T \boldsymbol{x} + \theta_0) \\
&= sign\left(\begin{bmatrix} 0.7936 & -0.8899 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 & 3 & 5 & 6 \\ 0 & 1 & 1 & 2 & 3 & 5 \end{bmatrix} \right) \\
&= sign\left(\begin{bmatrix} 0.7936, 0.6972, 1.4908, 0.6009, 1.2982, 0.3119 \end{bmatrix} \right) \\
&= \begin{bmatrix} 1, 1, 1, 1, 1, 1 \end{bmatrix}
\end{aligned}
$$

The optimal $\boldsymbol{\theta}$ and $\theta_0 = 0$ draw the projection line in the above figure.

3. K**-fold cross validation**: To evaluate a prediction model we use K-fold cross validation method. First we split dataset into K roughly equal sized sets and we fit the model for $K-1$ sets of data and test it for k^{th} set. We do this for $k = 1, 2, ..., K$ and take the average. When $K = N$(N is number of data), it is known as leave-one-out cross validation(LOOCV).

   (i) Consider a case in that the label is chosen at random according to $\mathbb{P}[y = 1] = \mathbb{P}[y = 0] = 1/2$. Consider a learning algorithm that outputs the constant predictor $h(\mathbf{x}) = 1$ if the parity of the labels on the training set is 1 and otherwise the algorithm output the constant predictor $h(\mathbf{x}) = 0$. This case is known as failure case of LOOCV, explain why.
   [**Solution**] Lets assume some binary classification dataset $\{(\mathbf{x}_1, 1), (\mathbf{x}_2, 0), ..., (\mathbf{x}_N, 0)\}$. Our predictor is that if we have odd number of train data that has label 1 then we always predicts 1 whenever test data are given as inputs and vice versa. So we divide into 2 cases.

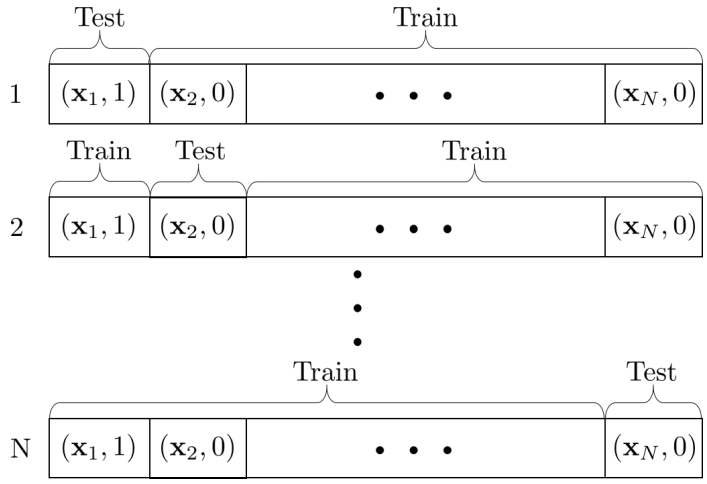


Figure 2: LOOCV

   a. Case 1: we have odd number of train data that has label 1.
   Using LOOCV method, we use first set as test set and use rest as a training set like top figure in figure 2. Since we used 1 data that has label 1 as a test set, now we have even number of data that has label 1. So our trained classifier is $h(\mathbf{x}) = 0$. When we test our trained classifier on test set, we get error of 1.
   Now we use data that has label 0 as a test data like second figure in figure 2. Since the number of data that has label 1 is not changed, our trained classifier is $h(\mathbf{x}) = 1$. When we test our classifier on test set, we get error of 1.

b. Case 2: we have even number of train data that has label 1.
Using LOOCV method, we use first set as test set and use rest as a training set like top figure in figure 2. Since we used 1 data that has label 1 as a test set, now we have odd number of data that has label 1. So our trained classifier is $h(\mathbf{x}) = 1$. When we test our trained classifier on test set, we get error of 0.
Now we use data that has label 0 as a test data like second figure in figure 2. Since the number of data that has label 1 is not changed, our trained classifier is $h(\mathbf{x}) = 0$. When we test our classifier on test set, we get error of 0. So when we use LOOCV as evaluation method, our predicted error will be always 0.

Considering above cases, when we use LOOCV as evaluation method our predicted error will be always 1 or 0. However, the real error is $\frac{1}{2}$, because $\mathbb{P}[y = 1] = \mathbb{P}[y = 0] = 1/2$. Therefore when we use LOOCV method, we cannot predict the real error. So this is the failure case of LOOCV.

(ii) Assume above case, this time use K-fold cross validation($K \neq N$) instead of LOOCV. Does it fails again, explain you answer.

[**Solution**] If we have multiple samples in each K-fold set, the expected number of sample that has label 1 and label 0 are same. So the expected error for each K-fold case is $\frac{1}{2}$. Therefore for K-fold cross validation($K \neq N$) it does not fail.

4. **Implementing k-NN algorithm(Matlab Programming Assignment)** In this problem, we will implement a k-NN algorithm. The following equation is the mathematical representation for k-NN algorithm where given data is noted as $(\mathbf{x_i}, y_i)$, and the Algorithm 1 explains step by step procedure for k-NN algorithm.

$$\hat{f}(\mathbf{x}) = \text{argmax}_t \Big(\sum_{i \in N_K(\mathbf{x})} \mathbb{1}(t = y_i) | N_K(\mathbf{x}) : \text{A set of } k \text{ nearest data to } \mathbf{x} \Big)$$

Algorithm 1: k-NN algorithm

Goal : Predict class y
Parameter : k
for $t = 1, ..., T$ **do**
  Calculate distance(d_t) between input $\mathbf{x}$ and data $\mathbf{x}_t$
end
Choose k number of samples that are nearest from the input $\mathbf{x}$
Predict class into the majority class
Output : Majority voted class y

For implementation, a skeleton code is provided in the "knn" folder. The main function `main_knn.m` calls the following 4 functions:(1) `[x_train, y_train, x_test, y_test] = createDataset ('train.csv', 'test.csv')`, (2) `[class] = classify_knn(x_test,x_train,y_train,k)`,(3) `[acc] = test_knn(class,y_test)`, (4) `plot_knn(x_train,y_train,x_test,y_test,k)`.

(i) `[x_train,y_train,x_test,y_test] = createDataset('train.csv','test.csv')` loads both the training and test dataset contained respectively in `train.csv` and `test.csv`.

```
function [x_train,y_train,x_test,y_test] = createDataset(train_data,test_data)
%% Arguments %%
% train_data : name of train data
% test_data : name of test data
% x_train : train data
% y_train : train label
% x_test : test data
```

```
% y_test : test label
%% Codes %%
train = load(train_data);
test = load(test_data);
x_train = train(:,1:2);
y_train = train(:,3);
x_test = test(:,1:2);
y_test = test(:,3);
end
```

(ii) `[class] = classify_knn(x_test,x_train,y_train,k)` takes both the training set {`x_train,y_train`} where `x_train` is the data array while `y_train` is the corresponding label array of training dataset and test set `x_test` as inputs to the k-NN and outputs the predicts class of test data `class`.

```
function class = classify_knn(x_input,x_train,y_train,k)
%% Arguments %%
% x_input: input to classify
% x_train: training data
% y_train: training label
% k: number of nearest neighbor
% class: predicted class
%% Your code here %%
class = zeros(length(x_input),1);
for j = 1:size(x_input, 1)
dist = sum((x_input(j,:) - x_train).^2,2);
[~,dist_sort] = sort(dist, 'ascend');
k_class = y_train(dist_sort(1:k));
class(j) = 1;
labels = unique(k_class);
for i = 1:length(labels)
l = labels(i);
if sum(k_class==l)>sum(k_class==class(j))
class(j) = l;
end
end
end
end
```

(iii) `[acc]=test_knn(class,y_test)` takes test label `y_test`, predicted class `class` and estimates the classification accuracy in `acc`.

```
function [acc] = test_knn(class,y_test)
%% Arguments %%
% class: predicted label
% y_test: test label
% acc: accuracy
%% Your code here %%
% Calculate acc
acc = sum(class == y_test) / length(y_test);
end
```

(iv) `plot_knn(x_train,y_train,x_test,y_test,k)` takes the training set {`x_train,y_train`} to scatter plots the loaded data and draw a decision boundary for training set. The positive samples should be represented as red dots while the negative samples should be in blue dots.

```
function plot_knn(x_train,y_train,x_test,y_test,k)
%% Arguments %%
```

```
% x_train: training data
% y_train: training label
% x_test: test data
% y_test: test label
% theta: weight parameter
%% Your code here %%
% Divide data according to label
x_train_pos = x_train(y_train==1,:);
x_train_neg = x_train(y_train==-1,:);
x_test_pos = x_test(y_test==1,:);
x_test_neg = x_test(y_test==-1,:);

% Plot decision boundary
max_x = max([x_train; x_test]);
min_x = min([x_train; x_test]);
[x1, x2] = meshgrid(linspace(min_x(1),max_x(1)), linspace(min_x(2),max_x(2)));
x_grid = [reshape(x1,[],1) reshape(x2,[],1)];
y = classify_knn(x_grid,x_train,y_train,k);

% plot figure
figure('pos',[100 300 1200 500])
subplot(1,2,1)
hold on
contour(x1,x2,reshape(y,size(x1)),1.5);
plot(x_train_pos(:,1), x_train_pos(:,2), 'or', x_train_neg(:,1), x_train_neg(:,2), 'ob')
title('Train data')
subplot(1,2,2)
hold on
contour(x1,x2,reshape(y,size(x1)),1.5);
plot(x_test_pos(:,1), x_test_pos(:,2), 'or', x_test_neg(:,1), x_test_neg(:,2), 'ob')
title('Test data')
end
```

Submit Instructions for Programming Assignment

- Please submit in .zip file to KLMS named **ee488_assignment2_student#.zip**, for example, "ee488_assignment2_20181234.zip".

- This file should contain one folder - **knn** and document file for the result with analysis.

- In matlab code, the comment explaining your code **must be** included, or you will not get a full grade even if your code works fine. Please also include all the files that are required to run the code in the zip file. Do not change the name of the folder and comments should be written in English. Additionally submitting unexecutable code will receive no points.