EECS 445 F14 Homework 2

WU Tongshuang
40782356

Group mate:
Eric Shin
Steven Mikes
Kevin Wegienka

1.

(a) According to Bayes rule:

$$p(t=1 \mid x;\emptyset,\Sigma,\mu_0,\mu_1)$$

$$= \frac{p(t=1 \mid x;\emptyset,\Sigma,\mu_0,\mu_1)p(t=1;\emptyset,\Sigma,\mu_0,\mu_1)}{p(x;\emptyset,\Sigma,\mu_0,\mu_1)}$$

$$= \frac{p(t=1 \mid x;\emptyset,\Sigma,\mu_0,\mu_1)p(t=1;\emptyset,\Sigma,\mu_0,\mu_1)}{p(x;\emptyset,\Sigma,\mu_0,\mu_1)}$$

$$= \frac{p(t=1 \mid x;\emptyset,\Sigma,\mu_0,\mu_1)p(t=1;\emptyset,\Sigma,\mu_0,\mu_1)}{p(x|t=1;\emptyset,\Sigma,\mu_0,\mu_1)p(t=1;\emptyset,\Sigma,\mu_0,\mu_1) + p(x|t=0;\emptyset,\Sigma,\mu_0,\mu_1)p(t=0;\emptyset,\Sigma,\mu_0,\mu_1)}$$

$$= \frac{p(t=1 \mid x;\emptyset,\Sigma,\mu_0,\mu_1)p(t=1;\emptyset,\Sigma,\mu_0,\mu_1)}{p(x|t=1;\emptyset,\Sigma,\mu_0,\mu_1)p(t=1;\emptyset,\Sigma,\mu_0,\mu_1) + p(x|t=0;\emptyset,\Sigma,\mu_0,\mu_1)p(t=0;\emptyset,\Sigma,\mu_0,\mu_1)}$$

$$= \frac{exp(-\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1))\emptyset}{exp(-\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1))\emptyset + exp(-\frac{1}{2}(x-\mu_0)^T\Sigma^{-1}(x-\mu_0))(1-\emptyset)}$$

$$= \frac{1}{1+\frac{1-\emptyset}{\emptyset}exp(\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1) - \frac{1}{2}(x-\mu_0)^T\Sigma^{-1}(x-\mu_0))}$$

$$= \frac{1}{1+exp(log(\frac{1-\emptyset}{\emptyset}) + \frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1) - \frac{1}{2}(x-\mu_0)^T\Sigma^{-1}(x-\mu_0))}$$

$\because \quad (a-b)^T\Sigma^{-1}(a-b) = a^T\Sigma^{-1}a - b^T\Sigma^{-1}a - a^T\Sigma^{-1}b + b^T\Sigma^{-1}b,\ {\mu_0}^T\Sigma^{-1}\boldsymbol{x} = ({\mu_0}^T\Sigma^{-1}\boldsymbol{x})^T$

$$= \frac{1}{1+exp(-[-log(\frac{1-\emptyset}{\emptyset}) + \frac{1}{2}({\mu_0}^T\Sigma^{-1}\mu_0 - {\mu_1}^T\Sigma^{-1}\mu_1) - {\mu_0}^T\Sigma^{-1}\boldsymbol{x} + {\mu_1}^T\Sigma^{-1}\boldsymbol{x}])}$$

Redefine the $x^{(i)}$'s to be $\mathcal{M}+1$-dimensional vectors by adding the extra coordinate $x_0 = 1$, we can see $p(t=1 \mid x;\emptyset,\Sigma,\mu_0,\mu_1)$ is in the form of logistic function, and can be written as $p(t=1 \mid x;\emptyset,\Sigma,\mu_0,\mu_1) = \frac{1}{1+exp(-w^Tx)}$,where

$$w = \begin{bmatrix} -log(\frac{1-\emptyset}{\emptyset}) + \frac{1}{2}({\mu_0}^T\Sigma^{-1}\mu_0 - {\mu_1}^T\Sigma^{-1}\mu_1) \\ \Sigma^{-1}\mu_1 - \Sigma^{-1}\mu_0 \end{bmatrix}$$

(b)

$$\ell(\emptyset,\Sigma,\mu_0,\mu_1)$$

$$= log\prod_{i=0}^{N} p\left(x^{(i)} \mid t^{(i)};\emptyset,\Sigma,\mu_0,\mu_1\right)p(y^{(i)};\emptyset)$$

$$= \sum_{i=1}^{N} log(p\left(x^{(i)} \mid t^{(i)};\emptyset,\Sigma,\mu_0,\mu_1\right)) + \sum_{i=1}^{N} log(p(y^{(i)};\emptyset))$$

$$= \sum_{i=1}^{N} log(p(x^{(i)} \mid t^{(i)}; \emptyset, \Sigma, \mu_0, \mu_1)) + \sum_{i=1}^{N} log(p(y^{(i)}; \emptyset))$$

where

$$(p(x^{(i)} \mid t^{(i)}; \emptyset, \Sigma, \mu_0, \mu_1) = \frac{1}{(2\pi)^{\frac{M}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x^{(i)} - \mu_{t^{(i)}})^T \Sigma^{-1}(x^{(i)} - \mu_{t^{(i)}})\right),$$

$$p(t^{(i)}; \emptyset) = \emptyset^{t^{(i)}}(1 - \emptyset)^{1-t^{(i)}}$$

$$\mathrm{M} = 1, \Sigma = [\sigma^2], |\Sigma| = \sigma^2, |\Sigma|^{\frac{1}{2}} = \sigma \text{ with } \sigma > 0,$$

$$(x^{(i)} - \mu_{t^{(i)}})^T = (x^{(i)} - \mu_{t^{(i)}}), \textit{which is just real number because it's 1 by 1}$$

$\therefore$ equation:

$$\sum_{i=1}^{N} [\tfrac{1}{2} log \tfrac{1}{2\pi} + log \tfrac{1}{\sigma} - \tfrac{1}{2}(x^{(i)} - \mu_{t^{(i)}})^2 [\sigma^2]^{-1} + t^{(i)} log\emptyset + (1 - t^{(i)}) log(1 - \emptyset)]$$

Taking derivatives:

$$\frac{\partial \ell}{\partial \phi} = \sum_{i=1}^{N} (\frac{t^{(i)}}{\emptyset} - \frac{1 - t^{(i)}}{1 - \emptyset})$$

$$= \sum_{i=1}^{N} (\frac{t^{(i)}}{\emptyset} - \frac{1 - t^{(i)}}{1 - \emptyset})$$

$$= \frac{\sum_{i=1}^{N} 1(t^{(i)} = 1)}{\emptyset} - \frac{N - \sum_{i=1}^{N} 1(t^{(i)} = 1)}{1 - \emptyset} = 0$$

$$\therefore (1 - \emptyset) \sum_{i=1}^{N} 1(t^{(i)} = 1) = \emptyset N - \emptyset \sum_{i=1}^{N} 1(t^{(i)} = 1)$$

Resulting in:

$$\emptyset = \frac{1}{N} \sum_{i=1}^{N} 1(t^{(i)} = 1)$$

$$\nabla_{\mu_0} \ell = \sum_{i:\, t^{(i)}=0} (-\frac{1}{2} \nabla_{\mu_0} (x^{(i)} - \mu_0)^2 [\sigma^2]^{-1})$$

$$= -\frac{1}{2} \sum_{i:\, t^{(i)}=0} 2(\mu_0 - \boldsymbol{x^{(i)}})$$

$$= -\sum_{i=i}^{N} 1(t^{(i)} = 0)\mu_0 - 1(t^{(i)} = 0)\boldsymbol{x^{(i)}}) = 0$$

Resulting in:

$$\mu_0 = \frac{\Sigma^{-1} \sum_{i=i}^{N} 1(t^{(i)} = 0)\boldsymbol{x^{(i)}}}{\Sigma^{-1} \sum_{i=i}^{N} 1(t^{(i)} = 0)} = \frac{\sum_{i=i}^{N} 1(t^{(i)} = 0)\boldsymbol{x^{(i)}}}{\sum_{i=i}^{N} 1(t^{(i)} = 0)}$$

Similarly:

$$\mu_1 = \frac{\sum_{i=i}^{N} 1(t^{(i)} = 1)\boldsymbol{x^{(i)}}}{\sum_{i=i}^{N} 1(t^{(i)} = 1)}$$

$\nabla_{\Sigma^{-1}}\ell = 0$ since Σ^{-1} is just a real number.

$\Sigma = [\sigma^2]$, by definition,

$$\Sigma = \frac{1}{N}\sum_{i=1}^{N}\left(x^{(i)} - \mu_{t^{(i)}}\right)^2$$

$$= \frac{1}{N}\sum_{i=1}^{N}\left(x^{(i)} - \mu_{t^{(i)}}\right)^T\left(x^{(i)} - \mu_{t^{(i)}}\right)$$

(See c for more general form.)

(c)

As in (b), equation:

$$= \sum_{i=1}^{N}\left[\frac{M}{2}log\frac{1}{2\pi} + \frac{1}{2}log\frac{1}{|\Sigma|} - \frac{1}{2}\left(x^{(i)} - \mu_{t^{(i)}}\right)^T\Sigma^{-1}\left(x^{(i)} - \mu_{t^{(i)}}\right)\right.$$
$$\left. + t^{(i)}log\emptyset + (1 - t^{(i)})log(1 - \emptyset)\right]$$

Taking derivatives:

$$\frac{\partial\ell}{\partial\phi} = \sum_{i=1}^{N}\left(\frac{t^{(i)}}{\emptyset} - \frac{1 - t^{(i)}}{1-\emptyset}\right)$$

$$= \sum_{i=1}^{N}\left(\frac{t^{(i)}}{\emptyset} - \frac{1 - t^{(i)}}{1-\emptyset}\right)$$

$$= \frac{\sum_{i=1}^{N} 1(t^{(i)} = 1)}{\emptyset} - \frac{N - \sum_{i=1}^{N} 1(t^{(i)} = 1)}{1 - \emptyset} = 0$$

$$\therefore (1 - \emptyset)\sum_{i=1}^{N} 1(t^{(i)} = 1) = \emptyset N - \emptyset\sum_{i=1}^{N} 1(t^{(i)} = 1)$$

Resulting in:

$$\emptyset = \frac{1}{N}\sum_{i=1}^{N} 1(t^{(i)} = 1)$$

$$\nabla_{\mu_0}\ell = \sum_{i:\, t^{(i)}=0}\left(-\frac{1}{2}\nabla_{\mu_0}\left(x^{(i)} - \mu_0\right)^T\Sigma^{-1}\left(x^{(i)} - \mu_0\right)\right)$$

$$= -\frac{1}{2}\sum_{i:\, t^{(i)}=0}\nabla_{\mu_0}\left(x^{(i)} - \mu_0\right)^T\Sigma^{-1}\left(x^{(i)} - \mu_0\right)$$

$$= -\frac{1}{2}\sum_{i:\, t^{(i)}=0}\nabla_{\mu_0}\left(\mu_0{}^T\Sigma^{-1}\mu_0 - 2\mu_0{}^T\Sigma^{-1}\boldsymbol{x^{(i)}}\right)$$

$$= -\frac{1}{2}\sum_{i:\, t^{(i)}=0}\left(2\Sigma^{-1}\mu_0 - 2\Sigma^{-1}\boldsymbol{x^{(i)}}\right)$$

$$= -\sum_{i=i}^{N}\left(1(t^{(i)} = 0)\Sigma^{-1}\mu_0 - 1(t^{(i)} = 0)\Sigma^{-1}\boldsymbol{x^{(i)}}\right)$$

$$= -\sum_{i=i}^{N}\left(1(t^{(i)} = 0)\Sigma^{-1}\mu_0\right) - \sum_{i=i}^{N}\left(1(t^{(i)} = 0)\Sigma^{-1}\boldsymbol{x^{(i)}}\right) = 0$$

Resulting in:

$$\mu_0 = \frac{\Sigma^{-1}\sum_{i=i}^{N} 1\left(t^{(i)} = 0\right)\boldsymbol{x^{(i)}}}{\Sigma^{-1}\sum_{i=i}^{N} 1(t^{(i)} = 0)} = \frac{\sum_{i=i}^{N} 1\left(t^{(i)} = 0\right)\boldsymbol{x^{(i)}}}{\sum_{i=i}^{N} 1(t^{(i)} = 0)}$$

Similarly:

$$\mu_1 = \frac{\sum_{i=i}^{N} 1\left(t^{(i)}=1\right)\boldsymbol{x}^{(i)}}{\sum_{i=i}^{N} 1(t^{(i)}=1)}$$

For Σ, here we directly taken derivation on the inverse matrix Σ^{-1} for convenience, since they should change accordingly and make no differences on the result.

$$\nabla_{\Sigma^{-1}}\ell = \sum_{i=1}^{N} \nabla_{\Sigma^{-1}}[\frac{1}{2}log|\Sigma|^{-1} - \frac{1}{2}\left(x^{(i)}-\mu_{t^{(i)}}\right)^T\Sigma^{-1}\left(x^{(i)} - \mu_{t^{(i)}}\right)]$$

$$= \sum_{i=1}^{N} [\frac{1}{2}\nabla_{\Sigma^{-1}}log|\Sigma|^{-1} - \frac{1}{2}\nabla_{\Sigma^{-1}}\left(x^{(i)}-\mu_{t^{(i)}}\right)^T\Sigma^{-1}\left(x^{(i)} - \mu_{t^{(i)}}\right)]$$

$$= \sum_{i=1}^{N} [\frac{1}{2|\Sigma|^{-1}}\nabla_{\Sigma^{-1}}|\Sigma|^{-1} - \frac{1}{2}\nabla_{\Sigma^{-1}}\left(x^{(i)}-\mu_{t^{(i)}}\right)^T\Sigma^{-1}\left(x^{(i)} - \mu_{t^{(i)}}\right)]$$

$$= \sum_{i=1}^{N} [\frac{1}{2|\Sigma|^{-1}}\nabla_{\Sigma^{-1}}|\Sigma|^{-1} - \frac{1}{2}\left(x^{(i)}-\mu_{t^{(i)}}\right)^T\nabla_{\Sigma^{-1}}\Sigma^{-1}\left(x^{(i)} - \mu_{t^{(i)}}\right)]$$

Since $\frac{\partial|x|}{\partial x} = \frac{x^T}{|x|}$, $\Sigma^T = \Sigma$:

$$= \sum_{i=1}^{N} [\frac{1}{2|\Sigma|^{-1}}|\Sigma|^{-1}\Sigma^T - \frac{1}{2}\left(x^{(i)}-\mu_{t^{(i)}}\right)^T\left(x^{(i)} - \mu_{t^{(i)}}\right)]$$

$$= \frac{1}{2}\sum_{i=1}^{N} [\Sigma - \left(x^{(i)}-\mu_{t^{(i)}}\right)^T\left(x^{(i)} - \mu_{t^{(i)}}\right)]$$

$$= \frac{1}{2}(N\Sigma - \sum_{i=1}^{N}\left(x^{(i)}-\mu_{t^{(i)}}\right)^T\left(x^{(i)} - \mu_{t^{(i)}}\right))$$

Resulting in:

$$\Sigma = \frac{1}{N}\sum_{i=1}^{N}\left(x^{(i)}-\mu_{t^{(i)}}\right)^T\left(x^{(i)} - \mu_{t^{(i)}}\right)$$

(d)

The computed parameters are:

$$\emptyset = 0.4300$$

$$\mu_0 = [1.1731 \quad 1.7420]$$

$$\mu_1 = [-0.0914 \quad -0.6088]$$

$$\Sigma = \begin{bmatrix} 0.6570 & 0.6878 \\ 0.6878 & 0.8956 \end{bmatrix}$$

The resulted accuracy is: 93.00%

2.

(a)

$\because$ For k = K: $\frac{\exp\left(w_k^T \emptyset(x)\right)}{1+\exp\left(w_j^T \emptyset(x)\right)} = \frac{\exp(0 \cdot \emptyset(x))}{1+\sum_{j=1}^{K-1} exp(w_j^T \emptyset(x))} = \frac{1}{1+\sum_{j=1}^{K-1} exp(w_j^T \emptyset(x))}$

$\therefore$ The following session will just consider k = K as the normal cases.

Indicator function: $I(t^{(i)} = m) = 1, I(t^{(i)} \neq m) = 1,$

Case 1: $t^{(i)} = k \neq m,$

For i = 1 to N, $p\left(t^{(i)} = m \mid x^{(i)}\right) = 0, I(t^{(i)} = m) = 0$

$l(w) = 0$

$\therefore \nabla_{w_m} l(w) = 0$

$$= \sum_{i=1}^{N} \emptyset(x^{(i)})[I(t^{(i)} = m) - p\left(t^{(i)} = m \mid x^{(i)}\right)]$$

Case 2: $t^{(i)} = k = m,\ I(t^{(i)} = m) = 1$

$l(w) = \sum_{i=1}^{N} log[p\left(t^{(i)} \mid x^{(i)}; w\right)]$

$\nabla_{w_m} l(w) = \sum_{i=1}^{N} \nabla_{w_m} log[p\left(t^{(i)} \mid x^{(i)}; w\right)]$

$$= \sum_{i=1}^{N} \nabla_{w_m} log \frac{exp(w_m^T \emptyset(x^{(i)}))}{1+\sum_{j=1}^{K-1} exp(w_j^T \emptyset(x^{(i)}))}$$

$$= \sum_{i=1}^{N} \nabla_{w_m} (-1 \cdot log \frac{1+\sum_{j=1}^{K-1} exp(w_j^T \emptyset(x^{(i)}))}{exp(w_m^T \emptyset(x^{(i)}))})$$

$$= \sum_{i=1}^{N} \nabla_{w_m} (-1 \cdot log(1 + \frac{1+\sum_{j=1,\neq m}^{K-1} exp(w_j^T \emptyset(x^{(i)}))}{exp(w_m^T \emptyset(x^{(i)}))}))$$

$$= \sum_{i=1}^{N} -1 \cdot (\frac{1}{1+\frac{1+\sum_{j=1,\neq m}^{K-1} exp(w_j^T \emptyset(x^{(i)}))}{exp(w_m^T \emptyset(x^{(i)}))}} \cdot (-\emptyset(x^{(i)})) \cdot (1 + \sum_{j=1,\neq m}^{K-1} exp(w_j^T \emptyset(x^{(i)}))) \cdot \frac{1}{exp(w_m^T \emptyset(x^{(i)}))})$$

$$= \sum_{i=1}^{N} (\frac{1+\sum_{j=1,\neq m}^{N} exp(w_j^T \emptyset(x^{(i)}))}{exp(w_m^T \emptyset(x^{(i)})) + 1 + \sum_{j=1,\neq m}^{N} exp(w_j^T \emptyset(x^{(i)}))} \cdot (\emptyset(x^{(i)})))$$

$$= \sum_{i=1}^{N} \emptyset(x^{(i)}) \frac{1+\sum_{j=1,\neq m}^{K-1} exp(w_j^T \emptyset(x^{(i)}))}{1+\sum_{j=1}^{K-1} exp(w_j^T \emptyset(x^{(i)}))}$$

$$= \sum_{i=1}^{N} \emptyset(x^{(i)})[1 - \frac{exp(w_m^T \emptyset(x^{(i)}))}{1+\sum_{j=1}^{K-1} exp(w_j^T \emptyset(x^{(i)}))}]$$

$$= \sum_{i=1}^{N} \emptyset(x^{(i)})[1 - p\left(t^{(i)} = m \mid x^{(i)}\right)]$$

$$= \sum_{i=1}^{N} \emptyset(x^{(i)})[I(t^{(i)} = m) - p\left(t^{(i)} = m \mid x^{(i)}\right)]$$

∴Overall:

$$\nabla_{w_m} l(w) = \sum_{i=1}^{N} \emptyset(x^{(i)})[I(t^{(i)} = m) - p(t^{(i)} = m \mid x^{(i)})]$$

(b)
Computer parameter (as introduced, w_3 is always fixed to 0):

$$\mathrm{w} = \begin{bmatrix} 7.2584 & 17.2115 & 0 \\ -22.4094 & -4.7467 & 0 \\ -2.9808 & -0.6819 & 0 \\ 8.3283 & 1.2255 & 0 \\ -0.0437 & -1.4462 & 0 \end{bmatrix}$$

Accuracy on test example:

Accuracy = 92%

Which is the same as got from *mnrfit.*

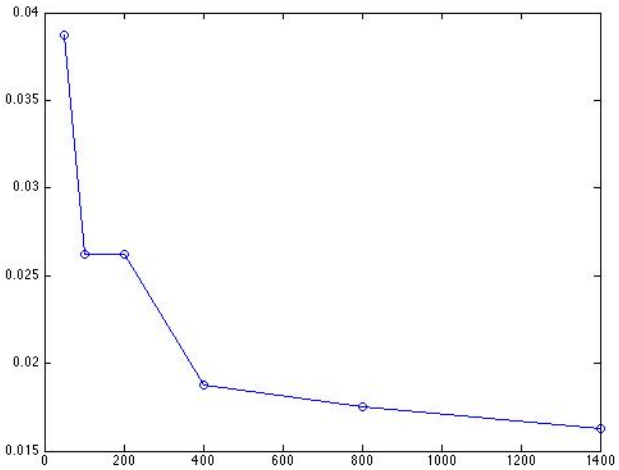
3.
(a) The test error:

$$e = 1.625\% \approx 1.63\%$$

(b) The five most indicative words are:

httpaddr, spam, unsubscribe, ebai, valet

(c) Plot for test set error vs. training set size:



Training set size	Test set error
50	3.87%
100	2.62%
200	2.62%
400	1.87%
800	1.75%
1400	1.63%
Full size	1.63%

Training size of 1400 (and probably larger) gives th best test set error.

Matlab code for Q1(d)

%- Run code: q1solution -%

```
%% EECS 445 - HW 02 - Q1 Gaussian Discriminate Analysis

%  Declaration
%  ------------
%  Date: 2014 / 10 / 02
%  Author: WU Tongshuang, 40782306

%  used helper function
%  --------------------
%  computeParameter(x_train, t_train, N_train, m)
%  computeTest(x_test, t_test, N_test, phi, u0, u1, cov)

%  Instructions
%  ------------
%  Using the M.L.E of the parameters given in part (b), compute each
%  parameter¡¯s value. Load the file q1 data.mat, which contains the variables
%  q1x train, q1y train, q1x test, q1y test. First, use training data to
%  estimate parameter values. Report your estimation result. Then, use your
%  result and your answer in (a) to classify test data.
%  Report your test accuracy.

%% Initialization
clear; close all; clc
%% ======================== Part 0: Load data ============================
fprintf('Running HW2 Q1 exercise ... \n');
load('q1_data.mat');
x_train = q1x_train; t_train = q1y_train;
x_test = q1x_test; t_test = q1y_test;
N_train = size(x_train, 1); % the number of training examples
N_test = size(x_test, 1);   % the number of training examples
m = size(x_train, 2);       % the dimention of x

%% ======================== Part 1: Compute parameter ======================
fprintf('Compute parameter ... \n');
[phi,u0,u1,cov] = computeParameter(x_train, t_train, N_train, m)

%% ======================== Part 2: compute accuracy =======================
fprintf('Compute accuracy for test ...\n')
accuracy = computeTest(x_test, t_test, N_test, phi, u0, u1, cov)
```

%- Helper function : **computeTest** -%

```
function [accuracy] = computeTest(x_test, t_test, N_test, phi, u0, u1, cov)
% EECS 445 HW2_Q1_d helper function
% use your result and your answer in (a) to classify test data.
% Report your test accuracy.

covU0 = cov \ u0';
covU1 = cov \ u1';
w1 = -log(1/phi-1) + 1/2 * (u0 * covU0 - u1 * covU1);
w2 = covU1 - covU0;
w  = [w1 ; w2];
x0 = ones(N_test,1);

x_compute = [x0 x_test];
accuracy = 0;

p = 1./(1+exp(-x_compute*w));
for i = 1: N_test
   if(round(p(i)) == t_test(i))
       accuracy = accuracy + 1;
   end
end
accuracy = accuracy / N_test;

end
```

%- Helper function : **computeParameter** -%

```
function [phi,u0,u1,cov] = computeParameter(x_train, t_train, N_train, m)
% EECS 445 HW2_Q1_d helper function
% Using the M.L.E of the parameters given in part (b),
% compute each parameter¡¯s value

x_positive = x_train(t_train==1, :);
x_negative = x_train(t_train==0, :);
N_positive = size(x_positive,1);
N_negative = size(x_negative,1);

phi = N_positive / N_train;
u1 = sum(x_positive) / N_positive;
u0 = sum(x_negative) / N_negative;
u  = [u0; u1];

cov = zeros(m,m);
for i = 1:N_train
    cov_element = (x_train(i,:)' - u(t_train(i)+1,:)');
    newMatrix = cov_element * cov_element';
    cov = cov + newMatrix;
end

cov = cov ./ N_train;
end
```

Matlab code for Q2(b)

%- Run code: q2solution -%

```
%% EECS 445 - HW 02 - Q2 Gaussian Discriminate Analysis
%  Declaration
%  ------------
%  Date: 2014 / 10 / 02
%  Author: WU Tongshuang, 40782306

%  used helper function
%  --------------------
%  softmaxGradient(x_train, t_train, N_train, w, alpha, category, m)
%  softmaxPredict(x_test, t_test,N_test, w)

%  Instructions
%  ------------
%  Using the gradient computed in part (a), implement Gradient Ascent for
%  Softmax Regression in Matlab. Use a learning rate a = 0.0005.
%  Load the file q2 data.mat, which contains the variables q2x train,
%  q2t train, q2x test, q2t test. Train your classifier on the training
%  data

%% Initialization
clear; close all; clc
%% ======================== Part 0: Load data ============================
fprintf('Running HW2 Q2 exercise ... \n');
load('q2_data.mat');
x_train = q2x_train; t_train = q2t_train;
x_test = q2x_test; t_test = q2t_test;
N_train = size(x_train, 1); % the number of training examples
N_test = size(x_test, 1);   % the number of test examples
x_train = [ones(N_train,1) x_train];
x_test = [ones(N_test,1) x_test];
m = size(x_train, 2);      % the dimention of x
category = max(t_train);
%% ======================== Part 1: Compute parameter ====================
fprintf('Compute parameter ... \n');
w = 0.005 * randn((category-1) * m, 1);
alpha = 0.0005;
w = softmaxGradient(x_train, t_train, N_train, w, alpha, category, m);
w_extract = [w; zeros(1,m)]' % since w3 is fixed to 0
%% ======================== Part 2: compute accuracy =====================
fprintf('Compute accuracy for test ...\n')
[accuracy, index] = softmaxPredict(x_test, t_test,N_test, w);
accuracy
```

%- Helper function : **softmaxGradient** -%

```
function [w, J_history] = softmaxGradient(x_train, t_train, N_train, w, alpha, category,
m)
% 2(d)
% implement Gradient Ascent for Softmax Regression

% dimention:
% x_train: N * dimention
% y_train: N * 1
% indicator: #category * N
% w: #category-1 * dimention

% indicator function, #category * N
indicator = full(sparse(t_train, 1:N_train, 1));
w = reshape(w, category-1, m);
iter = 1;

while(iter < 300000)

    multi = exp(x_train*w'); % N * category
    multi_train = [multi ones(N_train,1)];
    sumHx = repmat(sum(multi_train,2),1,3);
    hx = multi_train./sumHx;
    cost = sum(sum(indicator' .* log(hx)));
    J_history(iter) = cost;
    if(iter > 1)
        if(J_history(iter)-J_history(iter-1)< 1e-7)
            break;
        end
    end
    thetagrad = (indicator - hx') * x_train;
    w = w + alpha * thetagrad(1:2,:);
    iter =iter+1;
end

end
```

%- Helper function : **softmaxPredict** -%

```
function [accuracy index] = softmaxPredict(x_test, t_test,N_test, w)
% 2(b)
% implement Gradient Ascent for Softmax Regression

% x_test: N * dimention
% t_test: N * 1
% w: #category-1 * dimention
multi = exp(x_test*w'); % N * category
multi_train = [multi ones(N_test,1)];
sumHx = repmat(sum(multi_train,2),1,3);

hx = multi_train./sumHx;
[prob, index] = max(hx');
accuracy = 0;
index = index';
for i = 1:N_test
    if index(i) == t_test(i)
        accuracy = accuracy +1;
    end
end
accuracy = accuracy / N_test;
end
```

Matlab code for Q3

%- Run code: q3solution_a -%

```
%% EECS 445 - HW 02 - Q3_a Naive Bayes

%  Declaration
%  ------------
%  Date: 2014 / 10 / 02
%  Author: WU Tongshuang, 40782306

%  used helper function
%  --------------------
%  nb_train(range)
%  nb_test(range, phi_0, phi_1, phi_0_set, phi_1_set)

%  Instructions
%  ------------
%  Implement a naive Bayes classier for spam classification, using the
%  multinomial event model and Laplace smoothing

%% Initialization
clear; close all; clc

%% ======================== Part 1: Compute parameter ======================
fprintf('Compute parameter ... \n');
[phi_0, phi_1, phi_0_set, phi_1_set] = nb_train(0);

%% ======================== Part 2: compute accuracy =======================
fprintf('Compute accuracy for test ...\n')
error = nb_test(phi_0, phi_1, phi_0_set, phi_1_set);
fprintf('Computed error: %f \n',error);
```

%- Run code: q3solution_b -%

```
%% EECS 445 - HW 02 - Q3_b Naive Bayes for finding indicative words

%  Declaration
%  ------------
%  Date: 2014 / 10 / 02
%  Author: WU Tongshuang, 40782306

%  used helper function
%  --------------------
%  nb_train(range)
%  nb_test(range, phi_0, phi_1, phi_0_set, phi_1_set)

%  Instructions
%  ------------
%  find the 5 tokens that are most indicative of the SPAM class (i.e.,
%  have the highest positive value on the measure above)

%% Initialization
clear; close all; clc

%% ======================= Part 1: Compute parameter ======================
fprintf('Compute parameter ... \n');
[token_index token_text] = textread('TOKENS_LIST','%d %s');
[phi_0, phi_1, phi_0_set, phi_1_set] = nb_train(0);
error = nb_test(phi_0, phi_1, phi_0_set, phi_1_set);

%% ======================= Part 2: find indicative token ==================
fprintf('Extracting indicative words ...\n')

indicator = phi_1_set - phi_0_set;
[indicator,index] = sort(indicator, 'descend');
index = index(1:5);
for i = 1: size(index,2)
   token_text(index(i))
end
```

%- Run code: q3solution_c -%

```
%% EECS 445 - HW 02 - Q3_c Naive Bayes for different trainig set size

%  Declaration
%  ------------
%  Date: 2014 / 10 / 02
%  Author: WU Tongshuang, 40782306

%  used helper function
%  --------------------
%  nb_train(range)
%  nb_test(range, phi_0, phi_1, phi_0_set, phi_1_set)

%  Instructions
%  ------------
%  Implement a naive Bayes classier for spam classification, using the
%  multinomial event model and Laplace smoothing

%% Initialization
clear; close all; clc
errors = zeros(6,2);
trainSet = [50, 100, 200, 400, 800, 1400];
%% ======================== Part 1: Compute parameter ======================
fprintf('Compute error ... \n');

for i =1:size(trainSet,2)
    [phi_0, phi_1, phi_0_set, phi_1_set] = nb_train(trainSet(i));
    error = nb_test(phi_0, phi_1, phi_0_set, phi_1_set);
    errors(i,1) = trainSet(i);
    errors(i,2) = error;
end
errors

%% ======================== Part 2: plot error ======================
fprintf('Plot error ...\n')
plot(errors(:,1), errors(:,2),'-o');
```

%- Helper function : **nb_train** -%

```
function [phi_0, phi_1, phi_0_set, phi_1_set] = nb_train(range)

if(range == 0)
    [spmatrix, tokenlist, trainCategory] = readMatrix('MATRIX.TRAIN');
elseif (range == 50)
    [spmatrix, tokenlist, trainCategory] = readMatrix('MATRIX.TRAIN.50');
elseif (range == 100)
    [spmatrix, tokenlist, trainCategory] = readMatrix('MATRIX.TRAIN.100');
elseif (range == 200)
    [spmatrix, tokenlist, trainCategory] = readMatrix('MATRIX.TRAIN.200');
elseif (range == 400)
    [spmatrix, tokenlist, trainCategory] = readMatrix('MATRIX.TRAIN.400');
elseif (range == 800)
    [spmatrix, tokenlist, trainCategory] = readMatrix('MATRIX.TRAIN.800');
elseif (range == 1400)
    [spmatrix, tokenlist, trainCategory] = readMatrix('MATRIX.TRAIN.800');
end
trainMatrix = full(spmatrix);
numTrainDocs = size(trainMatrix, 1);
numTokens = size(trainMatrix, 2);

% …

% YOUR CODE HERE

negSet = trainMatrix(trainCategory == 0,:);
posSet = trainMatrix(trainCategory == 1,:);

negNumWord = sum(sum(negSet));
posNumWord = sum(sum(posSet));

for i = 1:numTokens
    phi_0_set(i) = (sum(negSet(:, i)) + 1) / (negNumWord + numTokens);
    phi_1_set(i) = (sum(posSet(:, i)) + 1) / (posNumWord + numTokens);
end
phi_0 = log(size(negSet,1) / numTrainDocs);
phi_1 = log(size(posSet,1) / numTrainDocs);

phi_0_set = log(phi_0_set);
phi_1_set = log(phi_1_set);

end
```

%- Helper function : **nb_test** -%

```
function [error] = nb_test(phi_0, phi_1, phi_0_set, phi_1_set)

[spmatrix, tokenlist, category] = readMatrix('MATRIX.TEST');

testMatrix = full(spmatrix);
numTestDocs = size(testMatrix, 1);
numTokens = size(testMatrix, 2);

% …
output = zeros(numTestDocs, 1);

%---------------
% YOUR CODE HERE
for i = 1:numTestDocs
    [r, c, v] = find(testMatrix(i,:));
    negP = sum(v .* phi_0_set(c)) + phi_0;
    posP = sum(v .* phi_1_set(c)) + phi_1;
    if (negP <= posP)
        output(i) = 1;
    end
end
%---------------


% Compute the error on the test set
error=0;
for i=1:numTestDocs
  if (category(i) ~= output(i))
    error=error+1;
  end
end

%Print out the classification error on the test set
error = error/numTestDocs;
```