CSCC11 Fall 2015
Assignment 3

Student Name: Maoting Zhang
Student Number: 999590633
UTORid(mathlab username): zhangmao

- # Matlab Scripts & functions
  - ## navie.m

```
function [P, a_1, a_0] = navie(X, y, data_train, labels_train, alpha, beta)
  % regularized form of Navie Baye's classification on C=1
  % X is a M x N matrix, columns are data vectors
  % y is a 1 x N vector, each labels the class of the corresponding data
  % alpha and beta are regularization parameters
  % P is a 1 x N vector where each p represents the classified class
  % i.e. for each p in P:
  % p(C=1|F_{1, 185}) = exp (a1 - r) / (exp (a1 - r) + exp (a0 - r))
  % where:
  % ak = sum_{i:F_i=1}(ln a_{i,k}) + sum_{i:F_i=0}(ln 1 - a_{k,1}) + ln b_k
  % for k = 0, 1
  % a_1 and a_0 are of size 1 x 185 representing a_{k, 1:185}

  M = size(data_train, 1);
  N = size(labels_train, 2);
  num_c1 = sum(labels_train);
  num_c0 = N - num_c1;

  % columns of data of different class
  c0_train = data_train(:, find(labels_train==0));
  c1_train = data_train(:, find(labels_train==1));

  % a_{i, 1} = (sum(F_i=1) + alpha)/ (num_C1 + 2*alpha)
  a_1 = zeros(0, M);
  a_0 = zeros(0, M);
  for i = 1:M
    a_1(i) = (sum(c1_train(i, :)) + alpha) / (num_c1 + 2*alpha);
    a_0(i) = (sum(c0_train(i, :)) + alpha) / (num_c0 + 2*alpha);
  end
  b_1 = (num_c1 + beta) / (N + 2*beta);
  b_0 = 1 - b_1;

  P = zeros(0, size(y, 2));
  for n = 1:size(y, 2)
      a0 = 0;
      a1 = 0;
      for m = 1:M
          if X(m, n) == 1
              a1 = a1 + log(a_1(m));
              a0 = a0 + log(a_0(m));
          else
              a1 = a1 + log(1 - a_1(m));
              a0 = a0 + log(1 - a_0(m));
          end
      end
      a1 = a1 + log(b_1);
      a0 = a0 + log(b_0);
      r = max(a1, a0);

      P(n) = exp(a1 - r) / (exp(a1 - r) + exp(a0 - r));
  end
  return;
```

- testNaive.m

```matlab
load('a3spam.mat')

figure(1);clf;
N_test = size(labels_test, 1);
error_naive_test = zeros(1, 9);
% WLOG we assume alpha == beta for simplicity
i = 1;
for regulator = 0.1:0.05:0.5

    P_c1_test = navie(data_test', labels_test', data_train', labels_train', regulator,
regulator);
    classes_test = P_c1_test > 0.5;
    error_naive_test(i) = (N_test - sum(classes_test==labels_test'));
    i = i + 1;
end

plot(0.1:0.05:0.5, error_naive_test, '--bs');

xlabel('regulator: alpha(==beta)');
ylabel('test error');
title('Test Error of Naive Bayes as a function of alpha==beta');


%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Top 10 features
% Idea:
%   separate the test data set according to the classification result,
% count the occurences frequencies of each feature in both classes.
%
% However, we noticed that there are words apprears to be common in both
% classes, which does not act as indicative, so instead we define the
% "indicativeness" of a feature to be of high difference in frequencies in
% two classes.

% choose alpha = beta = 0.1
regulator = 0.1;
[P_c1_test, a_1, a_0] = navie(data_test', labels_test', data_train', labels_train',
regulator, regulator);
classes_test = P_c1_test > 0.5;

% of size n x 185
c0 = data_test(find(classes_test==0), :);
c1 = data_test(find(classes_test==1), :);

% count occurrences of each feature in both classes
% of size 1 x 185
occur_c0 = sum(c0);
occur_c1 = sum(c1);
```

```
% frequencies occurrences / number of entries
freq_c0 = occur_c0 ./ size(c0, 1);
freq_c1 = occur_c1 ./ size(c1, 1);

diff_freq = freq_c0 - freq_c1;

[~, index] = sort(diff_freq);
% based on the context, we assume label==1 is spam
% features appears more often in C1 than C0:
top_spam_index = index(1:10);
top_spam = feature_names(top_spam_index)'
top_spam_weights = a_1(top_spam_index)

% features appears more often in C0 than C1:
% reverse the order of the indices (so from highest diff to lower)
top_ham_index = fliplr(index(176:185));
top_ham = feature_names(top_ham_index)'
top_ham_weights = a_0(top_ham_index)
```

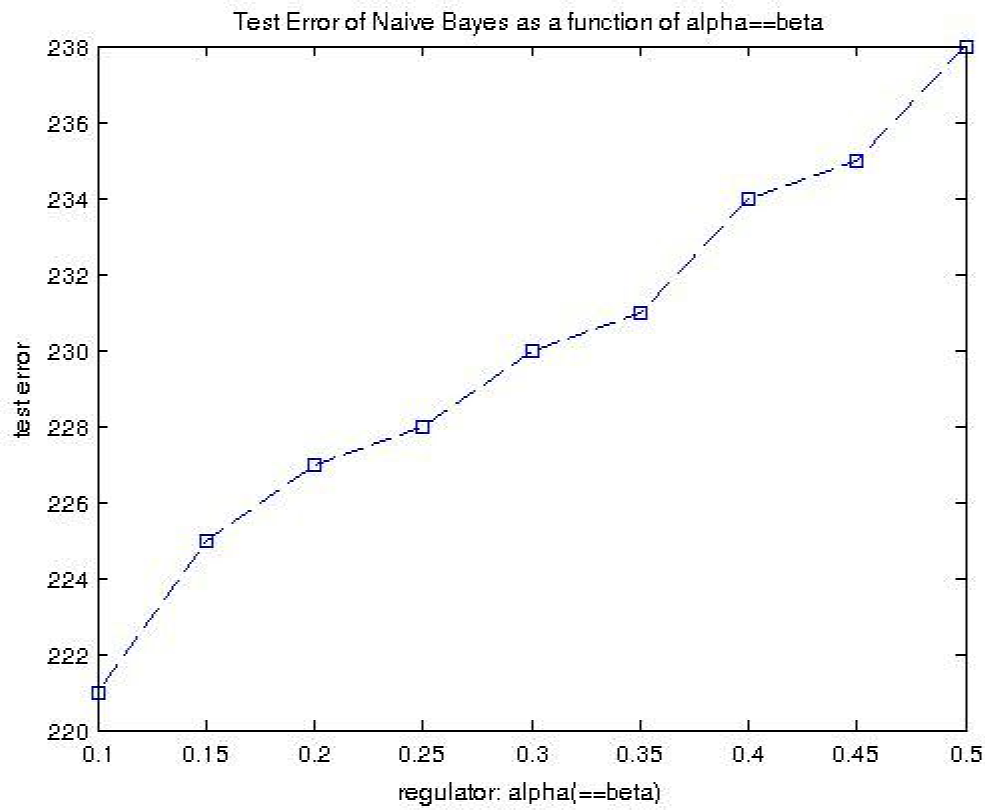
- ### testLR.m

Mostly of the same idea as in Naive approach so I removed some comments, please refer to mathlab submission for complete file. :)

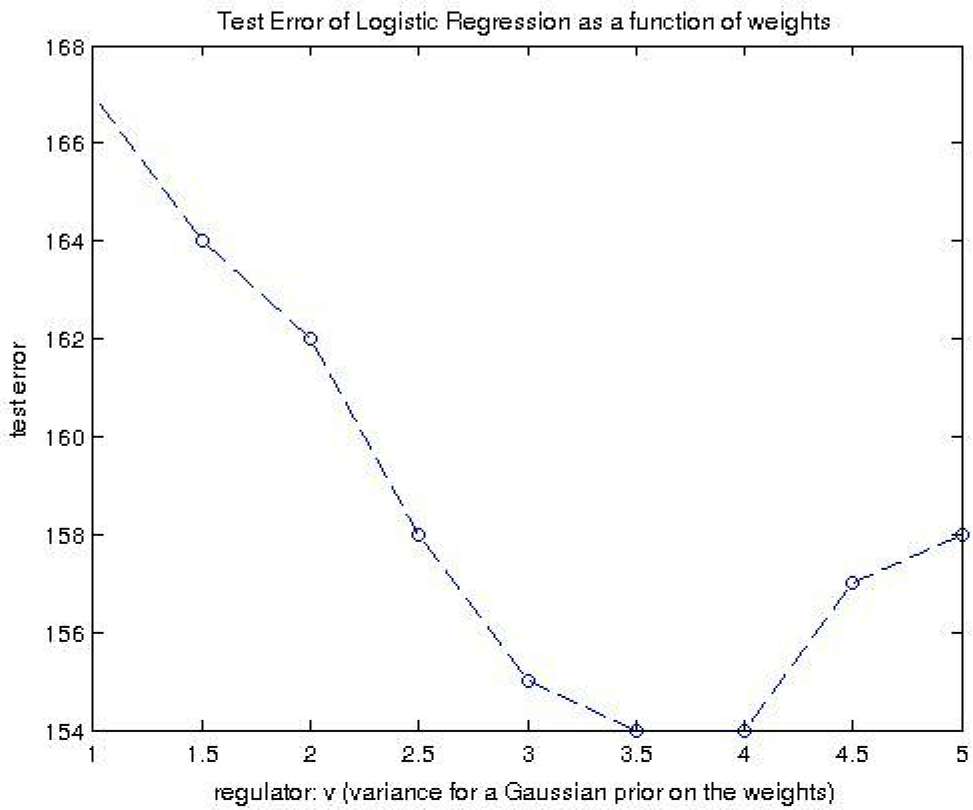
```
load('a3spam.mat')

figure(1);clf;
N = size(labels_test, 1);
error_lr = zeros(1, 9);

X = [ones(1, 1000); data_train'];
i = 1;
for v = 1:0.5:5
    [beta, converged] = logisticReg(X, labels_train', v);
    P_c1 = logistic([ones(1, 4000); data_test'], beta);
    classes = P_c1 > 0.5;
    error_lr(i) = N - sum(classes==labels_test');
    i = i + 1;
end
plot(1:0.5:5, error_lr, '--bo');

xlabel('regulator: v (variance for a Gaussian prior on the weights)');
ylabel('test error');
title('Test Error of Logistic Regression as a function of weights');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% choose v = 4
v=4;
[beta, converged] = logisticReg(X, labels_train', v);
P_c1 = logistic([ones(1, 4000); data_test'], beta);
classes = P_c1 > 0.5;
c0 = data_test(find(classes_test==0), :);
c1 = data_test(find(classes_test==1), :);
occur_c0 = sum(c0);
occur_c1 = sum(c1);
freq_c0 = occur_c0 ./ size(c0, 1);
freq_c1 = occur_c1 ./ size(c1, 1);
diff_freq = freq_c0 - freq_c1;
[~, index] = sort(diff_freq);
top_spam_index = index(1:10);
top_spam = feature_names(top_spam_index)'
top_spam_weights = beta(top_spam_index)'
top_ham_index = fliplr(index(176:185));
top_ham = feature_names(top_ham_index)'
top_ham_weights = beta(top_ham_index)'
```

- Error Plots



Total Error of **Naive** on **test data** as a function of alpha/beta



Total Error of **LR** on **test data** as a function of v

- Top 10 Features
  - Naive:

```
top_spam =

  Columns 1 through 6

    'alternative'    'public'    'quoted'    'transitional'    'alt'    'priority'

  Columns 7 through 10

    'express'    'related'    'microsoft'    'iso'

top_spam_weights =

  Columns 1 through 9

    0.3319    0.2781    0.3108    0.1987    0.1871    0.2851    0.2711    0.2104    0.2665

  Column 10

    0.3622

top_ham =

  Columns 1 through 6

    'toronto'    'sam'    'university'    'department'    'date'    'science'

  Columns 7 through 10

    'no_multipart'    'computer'    'research'    'status'

top_ham_weights =

  Columns 1 through 9

    0.6014    0.5245    0.3602    0.3584    0.4039    0.3113    0.9002    0.3130    0.2746

  Column 10

    0.2536
```

- LR:

```
top_spam =

  Columns 1 through 6

    'alternative'    'public'    'quoted'    'transitional'    'alt'    'priority'

  Columns 7 through 10

    'express'    'related'    'microsoft'    'iso'


top_spam_weights =

  Columns 1 through 9

   -0.7051    0.2036   -0.2687   -0.4662   -1.3087   -0.4136   -1.1955    0.0927   -0.8566

  Column 10

    0.7374


top_ham =

  Columns 1 through 6

    'toronto'    'sam'    'university'    'department'    'date'    'science'

  Columns 7 through 10

    'no_multipart'    'computer'    'research'    'status'


top_ham_weights =

  Columns 1 through 9

   -1.0024   -0.1718   -0.9389   -0.1934   -0.4959    0.2438   -0.6346    0.4076    0.1782

  Column 10

   -1.1033
```