```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
% Austin Welch
% EC503 HW6.1a
% SVM Classifier for Text Documents
% dataset: data_20news.zip
% using svmtrain, svmclassify

% clear variables/console and suppress warnings
clear; clc;
id = 'stats:obsolete:ReplaceThisWithMethodOfObjectReturnedBy';
id2 = 'stats:obsolete:ReplaceThisWith';
warning('off',id);
warning('off',id2);

% load data
disp('Loading data...');
traindata = importdata('train.data');
trainlabel = importdata('train.label');
testdata = importdata('test.data');
testlabel = importdata('test.label');
vocab = importdata('vocabulary.txt'); % all words in docs,
 line#=wordID
stoplist = importdata('stoplist.txt'); % list of commonly used stop
 words
classes = importdata('newsgrouplabels.txt'); % names of the 20 classes

% determine wordIDs in vocabulary that are not in train/test data
IDsNotInTrain = setdiff(1:length(vocab),unique(traindata(:,2)));
IDsNotInTest = setdiff(1:length(vocab),unique(testdata(:,2)));

% determine stop words' wordIDs
[~, stopIDs, ~] = intersect(vocab, stoplist);

% change stop word counts to zero
traindata(ismember(traindata(:,2),stopIDs),3) = 0;
testdata(ismember(testdata(:,2),stopIDs),3) = 0;

% add missing words to train/test data, but with zero counts
appendRows = zeros(length(IDsNotInTrain),3);
appendRows(:,1) = 1; appendRows(:,2) = IDsNotInTrain; appendRows(:,3)
 = 0;
traindata = [appendRows; traindata];
appendRows = zeros(length(IDsNotInTest),3);
appendRows(:,1) = 1; appendRows(:,2) = IDsNotInTest; appendRows(:,3) =
 0;
testdata = [appendRows; testdata];
clear appendRows;

% rearrange train/test data to dimensions (doc#, vocab#) with count
 values
Mtrain = sparse(accumarray(traindata(:,1:2), traindata(:,3)));
```

```
Mtest = sparse(accumarray(testdata(:,1:2), testdata(:,3)));

% calculate frequencies by dividing each count by the word totals
Mtrain = Mtrain ./ sum(Mtrain,2);
Mtest = Mtest ./ sum(Mtest,2);

% when removing stop words, couple docs end up with total word counts
 of
% zero, which causes division by 0 when calculating frequencies and
 results
% in nans. need to find these nans and replace with zeros.
Mtrain(sum(Mtrain,2)==0,:) = 0;
Mtest(sum(Mtest,2)==0,:) = 0;

Loading data...
```

part (a) : binary SVM with linear kernel

```
% select classes 1 & 20
twoClassRowsTrain = (trainlabel==1 | trainlabel==20);
twoTrainData = Mtrain(twoClassRowsTrain,:);
twoTrainLabel = trainlabel(twoClassRowsTrain);
twoClassRowsTest = (testlabel==1 | testlabel==20);
twoTestData = Mtest(twoClassRowsTest,:);
twoTestLabel = testlabel(twoClassRowsTest);

% 5-fold cross-validation for boxconstraint (cost) parameter
fprintf('Beginning part (a)...\n\n');
K = 5;
CV = cvpartition(twoTrainLabel, 'KFold', K);
ccrs = zeros(CV.NumTestSets,1);
cRange = -5:15;
CV_CCRs = zeros(length(cRange),1);
h = waitbar(0,'Cross-validating boxconstraint parameter...', ...
    'Name','Part (a)');
for i=1:length(cRange)
    waitbar(i/length(cRange));
    C = 2^cRange(i);
    for j = 1:CV.NumTestSets
        vectorC = C*ones(CV.TrainSize(j),1);
        trIdx = CV.training(j);
        teIdx = CV.test(j);
        SVMStruct = svmtrain(twoTrainData(trIdx,:), ...
            twoTrainLabel(trIdx), 'kernel_function', 'linear', ...

  'boxconstraint',C*ones(CV.TrainSize(j),1), 'autoscale', ...
            'false', 'kernelcachelimit', 20000);
        yPredictions = svmclassify(SVMStruct, twoTrainData(teIdx,:));
        ccrs(j) = sum(yPredictions == twoTrainLabel(teIdx))/
CV.TestSize(j);
    end
    CV_CCRs(i) = mean(ccrs);
    fprintf('C = 2^%d, CV-CCR: %0.4f\n\n', cRange(i), CV_CCRs(i));
```

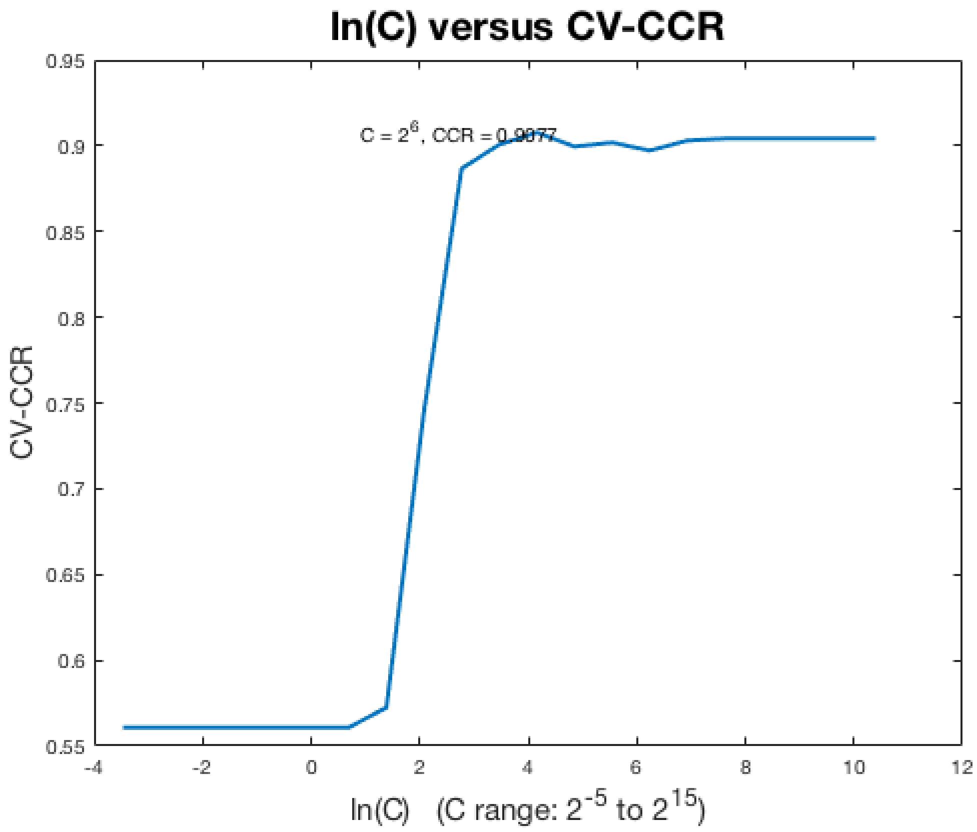
```
end
delete(h);

% Determine best CV_CCR and boxconstraint
[bestCCR, bestCIndex] = max(CV_CCRs);
bestC = cRange(bestCIndex);
fprintf('C* is 2^%d and corresponding CCR value is %0.4f\n', ...
    bestC, bestCCR);

% plot ln(C) vs. CV-CCR
figure(1);
graph1 = plot(log(2.^cRange),CV_CCRs);
set(graph1,'LineWidth',2)
title('ln(C) versus CV-CCR','FontSize',20);
xlabel('ln(C)   (C range: 2^{-5} to 2^{15})','FontSize',15);
ylabel('CV-CCR','FontSize',15);
text(CV_CCRs(bestCIndex), bestCCR, sprintf('C = 2^%d, CCR =
 %6.4f', ...
    cRange(bestCIndex), bestCCR), 'FontSize',10);

% print comments
fprintf(['\nC* seems to range from 2^6 to 2^9 on different runs, with
\n'...
    'the most common value seen from repeated trials being 2^7.\n',...
    'The CV-CCR starts at 0.5589 (with C^-5) and stays there until
\n', ...
    'C reaches 2^3, upon which time the CV-CCR begins to rapidly
\n', ...
    'increase. It peaks at approximately C = 2^7, then drops very
\n',...
    'slightly and levels off.\n\n'])

% Now that I have C*, train on all class 1 & 20 training data
SVMStruct = svmtrain(twoTrainData,
 twoTrainLabel, 'kernel_function', ...

  'linear','boxconstraint',2^(bestC)*ones(length(twoTrainLabel),1), ...
    'autoscale','false', 'kernelcachelimit', 60000);

% Then test on all class 1 & 20 test data and report CCR
yPredictions = svmclassify(SVMStruct, twoTestData);
CCR = sum(yPredictions==twoTestLabel)/length(twoTestLabel);
fprintf('CCR on entire test data for classes 1 & 20: %0.4f\n', CCR);

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

```
Beginning part (a)...

C = 2^-5, CV-CCR: 0.5608

C = 2^-4, CV-CCR: 0.5608

C = 2^-3, CV-CCR: 0.5608
```

```
C = 2^-2, CV-CCR: 0.5608

C = 2^-1, CV-CCR: 0.5608

C = 2^0, CV-CCR: 0.5608

C = 2^1, CV-CCR: 0.5608

C = 2^2, CV-CCR: 0.5724

C = 2^3, CV-CCR: 0.7453

C = 2^4, CV-CCR: 0.8867

C = 2^5, CV-CCR: 0.9007

C = 2^6, CV-CCR: 0.9077

C = 2^7, CV-CCR: 0.8995

C = 2^8, CV-CCR: 0.9018

C = 2^9, CV-CCR: 0.8972

C = 2^10, CV-CCR: 0.9030

C = 2^11, CV-CCR: 0.9042

C = 2^12, CV-CCR: 0.9042

C = 2^13, CV-CCR: 0.9042

C = 2^14, CV-CCR: 0.9042

C = 2^15, CV-CCR: 0.9042

C* is 2^6 and corresponding CCR value is 0.9077
```

C seems to range from 2^6 to 2^9 on different runs, with the most common value seen from repeated trials being 2^7. The CV-CCR starts at 0.5589 (with C^-5) and stays there until C reaches 2^3, upon which time the CV-CCR begins to rapidly increase. It peaks at approximately C = 2^7, then drops very slightly and levels off.*

```
CCR on entire test data for classes 1 & 20: 0.8102
```



Published with MATLAB® R2017a