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

% clear variables/console and suppress warnings
clear; clc; tic;
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...

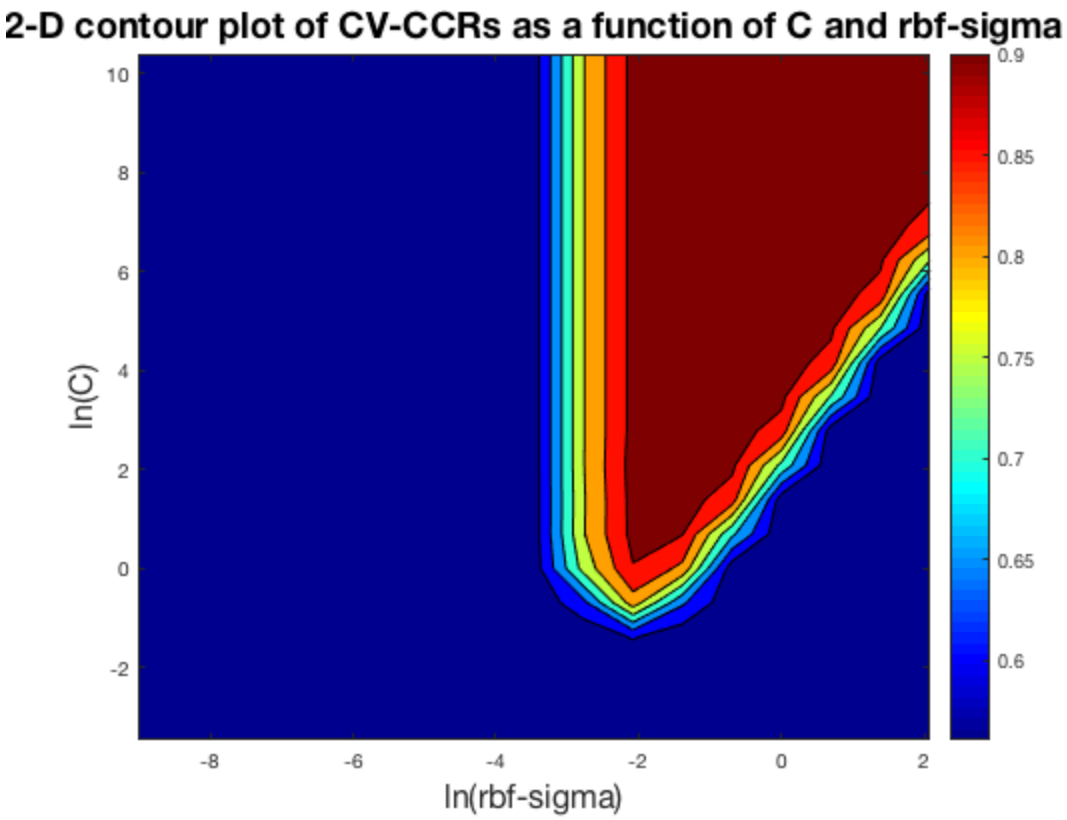
part (b) : binary SVM with RBF kernel

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

% exhaustive grid-search with 5-fold cross-validation for both
% boxconstraint (cost) parameter and rbf_sigma parameter
fprintf('Beginning part (b)...\n\n');
K = 5;
CV = cvpartition(twoTrainLabel, 'KFold', K);
ccrs = zeros(CV.NumTestSets,1);
cRange = -5:15; % boxconstraint exponents
sRange = -13:3; % rbf-sigma exponents
CV_CCRs = zeros(length(cRange),length(sRange));
h = waitbar(0,'Cross-validating C and rbf-sigma...', ...
    'Name','Part (b)');
it = 0;
% loop through boxconstraint values
for i=1:length(cRange)
    waitbar(i/length(cRange));
    C = 2^cRange(i);
    % loop through rbf-sigma values
    for k=1:length(sRange)
        rbf_sigma = 2^sRange(k);
        % loop through k-folds
        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', 'rbf', ...
                'rbf_sigma', rbf_sigma, '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,k) = mean(ccrs);
        it = it+1;
        %fprintf(['Iteration %3d:   C = 2^%3d, rbf-sigma = 2^%3d,
 ', ...
        %    'CV-CCR = %0.4f\n'], it, cRange(i),
 sRange(k),CV_CCRs(i,k));
    end
end
close(h);
toc

% (i)
% plot 2-D contour of CV-CCRs as a function of C and rbf-sigma
colormap('jet');
contourf(log(2.^sRange),log(2.^cRange),CV_CCRs)
title('2-D contour plot of CV-CCRs as a function of C and rbf-
sigma', ...
    'FontSize',18);
xlabel('ln(rbf-sigma)','FontSize',16);
ylabel('ln(C)','FontSize',16);
colorbar;

% (ii)
% report best (boxconstraint, rbf-sigma) pair
bestCCR = max(CV_CCRs(:));
[cInd,sInd] = find(CV_CCRs==bestCCR);
fprintf('\nBest CV-CCR: %0.4f\n\n', bestCCR);
fprintf('Corresponding (boxconstraint, rbf-sigma) pair(s):\n');
fprintf('(2^%d, 2^%d)\n', cRange(cInd), sRange(sInd));

% (iii)
% use best pair(s) to train on entire training set and test on test
 set
fprintf('\nBest (boxconstraint, rbf-sigma) pair(s) on test set:\n');
testCCRs = zeros(length(cInd));
for i=1:length(cInd)
SVMStruct = svmtrain(twoTrainData, ...
    twoTrainLabel, 'kernel_function', 'rbf', ...
    'rbf_sigma', 2^sRange(sInd(i)), 'boxconstraint', ...
    (2^cRange(cInd(i)))*ones(length(twoTrainLabel),1),'autoscale', ...
    'false', 'kernelcachelimit', 20000);
yPredictions = svmclassify(SVMStruct, twoTestData);
CCR = sum(yPredictions == twoTestLabel)/length(twoTestLabel);
fprintf('(2^%d, 2^%d) CCR: %0.4f\n', ...
    cRange(cInd(i)), sRange(sInd(i)), CCR);
testCCRs(i) = CCR;
end
```

```
fprintf(['\nBest CCR from above on entire test set (classes 1 & 20):
 ', ...
    '%0.4f\n\n'], max(testCCRs(:)));
fprintf(['Overall, the RBF kernel seems to perform very close to,
\n', ...
    'but usually slightly better than the linear kernel for classes
\n', ...
    '1 and 20 on this particular training/test set.\n\n']);

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

```
Beginning part (b)...

Elapsed time is 394.932509 seconds.

Best CV-CCR: 0.9229

Corresponding (boxconstraint, rbf-sigma) pair(s):
(2^3, 2^-2)

Best (boxconstraint, rbf-sigma) pair(s) on test set:
(2^3, 2^-2) CCR: 0.8067

Best CCR from above on entire test set (classes 1 & 20): 0.8067

Overall, the RBF kernel seems to perform very close to,
but usually slightly better than the linear kernel for classes
1 and 20 on this particular training/test set.
```



Published with MATLAB® R2017a