HW6: SVM Document Classification

Austin Welch

EC503

April 3, 2017

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

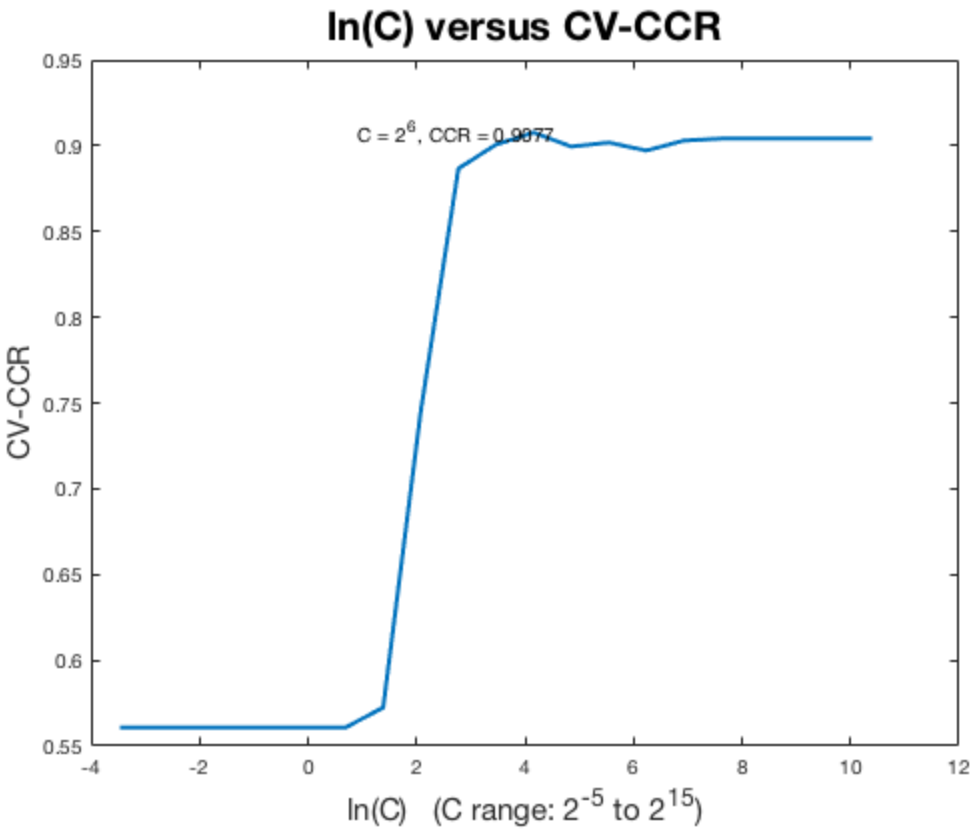
C = 2^14, CV-CCR: 0.9042

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

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
% 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...

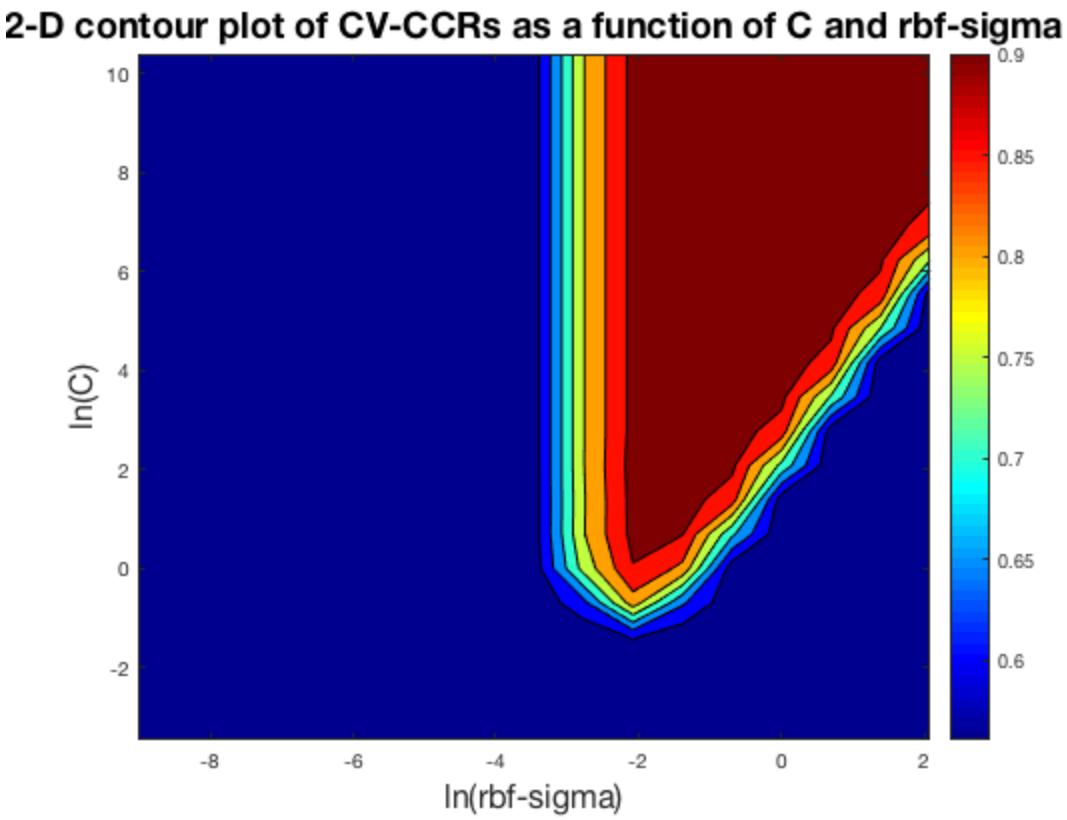
part (b) : binary SVM with RBF kernel

```
% 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

Table of Contents

..... 1
Setup 1
Part (c) : SVM with linear kernel, one class vs. all 2
Determine best CV_CCR and boxconstraint and run full test data with C* 4
Part (d) : Performance metrics 6


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

Setup

```
% 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 (c) : SVM with linear kernel, one class vs. all

```
% set all non-17 class labels to 0
trainlabel(trainlabel~=17)=0;
testlabel(testlabel~=17)=0;

% 5-fold cross-validation for boxconstraint (cost) parameter
fprintf('Running parts (c),(d)...\n\n');
K = 5;
CV = cvpartition(trainlabel, '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 (c)');
% performance metrics for each C
avgPrecisions = zeros(1,length(cRange));
avgRecalls = zeros(1,length(cRange));
avgFscores = zeros(1,length(cRange));
topRecall = 0; topFscore = 0;
% loop through boxconstraint values
for i=1:length(cRange)
    waitbar(i/length(cRange));
    C = 2^cRange(i);
    % performance metrics for each fold
    confMats = cell(1,K);
```

```
    precisions = zeros(1,K);
    recalls = zeros(1,K);
    fscores = zeros(1,K);
    % loop through CV partitions
    for j = 1:CV.NumTestSets
        trIdx = CV.training(j);
        teIdx = CV.test(j);
        SVMStruct = svmtrain(Mtrain(trIdx,:), ...
            trainlabel(trIdx), 'kernel_function', 'linear', ...
            'boxconstraint',C, 'autoscale', ...
            'false', 'kernelcachelimit', 20000);
        yPredictions = svmclassify(SVMStruct, Mtrain(teIdx,:));
        ccrs(j) = sum(yPredictions == trainlabel(teIdx))/
CV.TestSize(j);

        % transpose and swap columns (to arrange as in lecture slides)
        confMats{j} = confusionmat(trainlabel(teIdx),yPredictions)';
        confMats{j}(:,[1,2])=confMats{j}(:,[2,1]);

        % performance metrics for each fold
        recalls(j) = confMats{j}(1,1)/sum(confMats{j}(:,1));
        precisions(j) = confMats{j}(1,1)/sum(confMats{j}(1,:));
        fscores(j) = (2*recalls(j)*precisions(j)) / ...
            (recalls(j)+precisions(j));
    end
    % CV-CCR for current C
    CV_CCRs(i) = mean(ccrs);
    fprintf('C = 2^%d, CV-CCR: %0.4f\n\n', cRange(i), CV_CCRs(i));
    % average performance metrics for each C
    avgPrecisions(i) = mean(precisions);
    avgRecalls(i) = mean(recalls);
    avgFscores(i) = mean(fscores);
    % save best confusion matrices for top average recall and F-score
    if avgRecalls(i) > topRecall
        bestRecallConfs = confMats;
    end
    if avgFscores(i) > topFscore
        bestFscoreConfs = confMats;
    end
end
close(h);
```

Running parts (c),(d)...

C = 2^-5, CV-CCR: 0.8819

C = 2^-4, CV-CCR: 0.8819

C = 2^-3, CV-CCR: 0.8819

C = 2^-2, CV-CCR: 0.9712

C = 2^-1, CV-CCR: 0.9729

```
C = 2^0, CV-CCR: 0.9757

C = 2^1, CV-CCR: 0.9778

C = 2^2, CV-CCR: 0.9815

C = 2^3, CV-CCR: 0.9838

C = 2^4, CV-CCR: 0.9861

C = 2^5, CV-CCR: 0.9886

C = 2^6, CV-CCR: 0.9895

C = 2^7, CV-CCR: 0.9904

C = 2^8, CV-CCR: 0.9908

C = 2^9, CV-CCR: 0.9906

C = 2^10, CV-CCR: 0.9898

C = 2^11, CV-CCR: 0.9894

C = 2^12, CV-CCR: 0.9894

C = 2^13, CV-CCR: 0.9894

C = 2^14, CV-CCR: 0.9894

C = 2^15, CV-CCR: 0.9894
```

Determine best CV_CCR and boxconstraint and run full test data with C*

```
% find C*
[bestCCR, bestCIndex] = max(CV_CCRs);
bestC = cRange(bestCIndex);
fprintf('C* is 2^%d and corresponding CV-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);
```

```
% Now that I have C*, train on all training data
SVMStruct = svmtrain(Mtrain, trainlabel, 'kernel_function', ...
    'linear','boxconstraint',2^(bestC), ...
    'autoscale','false', 'kernelcachelimit', 20000);

% Then test on all test data and report CCR
yPredictions = svmclassify(SVMStruct, Mtest);
CCR = sum(yPredictions==testlabel)/length(testlabel);
fprintf('\nC* CCR on entire test data for 17 vs. all: %0.4f\n', CCR);

% confusion matrix
conf = confusionmat(testlabel,yPredictions)';
conf(:,[1,2]) = conf(:,[2,1]);
printmat(conf, 'confusion matrix','y_hat=17 y_hat=0','y=17 y=0');

% report observations
fprintf(['Because of the highly unbalanced nature of the data,\n', ...
    'the classifier can just choose the non-17 set nearly every time
\n',...
    'and obtain a high CCR. If the classifier chose the non-17 class
\n',...
    'every single time it would still produce over a 0.95 CCR.\n\n']);
```

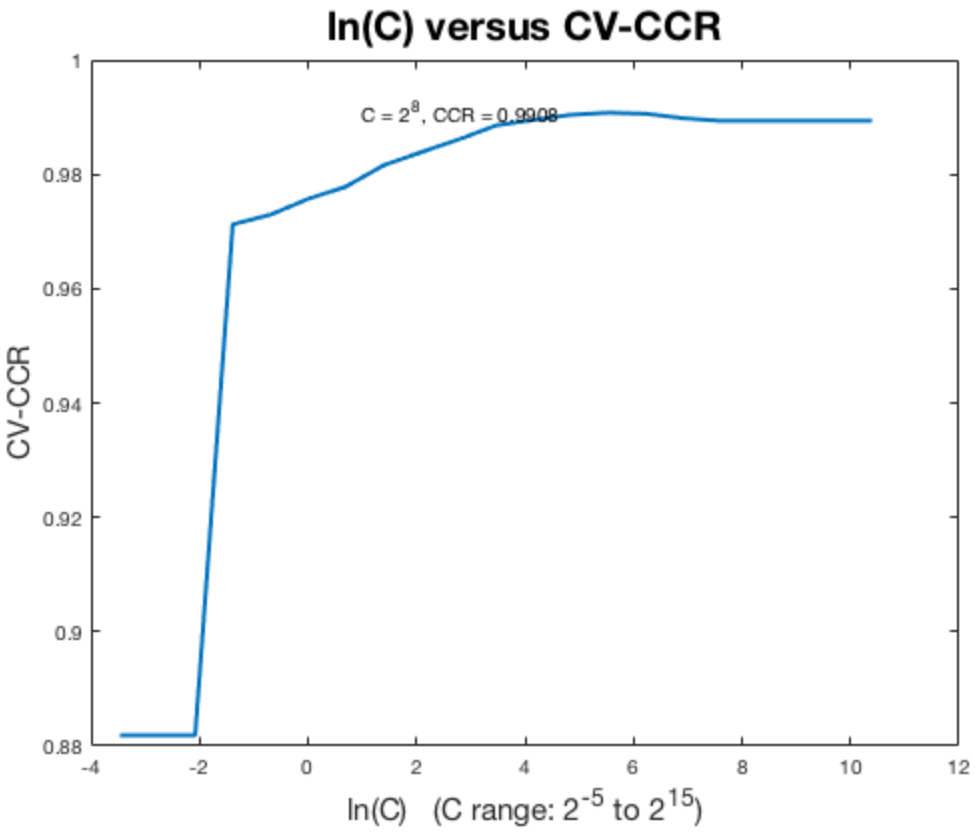
C is 2^8 and corresponding CV-CCR value is 0.9908*

C CCR on entire test data for 17 vs. all: 0.9712*

confusion matrix =

	y=17	y=0
y_hat=17	116.00000	7041.00000
y_hat=0	248.00000	99.00000

Because of the highly unbalanced nature of the data,
the classifier can just choose the non-17 set nearly every time
and obtain a high CCR. If the classifier chose the non-17 class
every single time it would still produce over a 0.95 CCR.

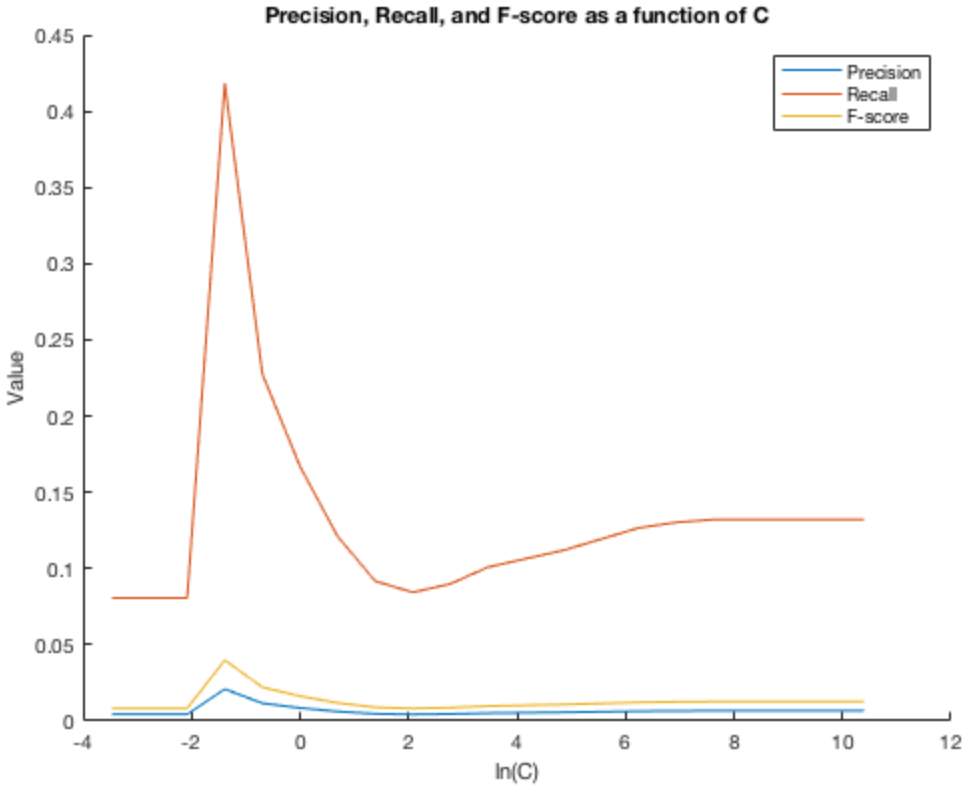


Part (d) : Performance metrics

```
% (i) : plot CV precision, recall, and F-score as functions of C
figure(2);
hold on;
plot(log(2.^cRange), avgPrecisions);
plot(log(2.^cRange), avgRecalls);
plot(log(2.^cRange), avgFscores);
title('Precision, Recall, and F-score as a function of C');
xlabel('ln(C)');
ylabel('Value');
legend('Precision','Recall','F-score');

% (ii) : best values of C in terms of both recall and F-score
[bestRecall, recallInd] = max(avgRecalls);
[bestFscore, fscoreInd] = max(avgFscores);
recallBestC = cRange(recallInd);
fscoreBestC = cRange(fscoreInd);
fprintf('Best C in terms of recall: 2^%d (recall=%0.4f)\n', ...
    recallBestC, bestRecall);
fprintf('Best C in terms of F-score: 2^%d (F-score=%0.4f)\n', ...
    fscoreBestC, bestFscore);

% corresponding confusion matrices
```

```
fprintf('\n5-fold confusion mats corresponding to best mean recall:
\n');
for i=1:K
    disp(bestRecallConfs{i});
end

fprintf('5-fold confusion mats corresponding to best mean F-score:
\n');
for i=1:K
    disp(bestFscoreConfs{i});
end

% show elapsed time and play sound alert when completed
toc
load handel
sound(y,Fs)
```

```
Best C in terms of recall: 2^-2 (recall=0.4183)
Best C in terms of F-score: 2^-2 (F-score=0.0399)

5-fold confusion mats corresponding to best mean recall:
          10        2139
          99           5

          19        2133
          90          12

          18        2140
          91           5

          16        2129
          93          16

           9        2135
         100           9

5-fold confusion mats corresponding to best mean F-score:
          10        2139
          99           5

          19        2133
          90          12

          18        2140
          91           5

          16        2129
          93          16

           9        2135
         100           9

Elapsed time is 785.026721 seconds.
```



Published with MATLAB® R2017a

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

Setup

```
% 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 (e) : One-versus-one OVO multi-class classification linear kernel

```
fprintf('\nStarting part (e)...\n\n');

% all combinations and count
allPairs = combnk(1:20,2);
mChoose2 = nchoosek(20,2);

% train m(m-1)/2=190 binary SVMs for all class pairs
tic
allSVMs = cell(1,mChoose2);
fprintf('Training all binary SVM pairs with linear kernel...\n\n');
h = waitbar(0, 'Training all binary SVM pairs...','Name','Part (e)');
for p=1:mChoose2
    waitbar(p/mChoose2);
    % select pair
    pair = allPairs(p,:);
    trainDataPair = sparse(Mtrain((trainlabel==pair(1) | ...
        trainlabel==pair(2)),:));
    trainLabelPair = trainlabel(trainlabel==pair(1) |
 trainlabel==pair(2));
    % train pair
    %fprintf('training pair %3d/%d: (%d,
%d)\n',p,mChoose2,pair(1),pair(2));
    SVMStruct = svmtrain(trainDataPair, trainLabelPair, ...
        'autoscale','false', 'kernelcachelimit', 20000);
    allSVMs{p} = SVMStruct;
end
close(h);
trainingTime = toc;
fprintf('Total training time: %0.2f seconds\n\n', trainingTime);


% test on all binary SVM pairs
tic
```

```
allPredictions = zeros(length(testlabel),mChoose2);
fprintf('Testing all binary SVM pairs...\n\n');
h = waitbar(0, 'Testing all binary SVM pairs...','Name','Part (e)');
for i=1:mChoose2
    waitbar(i/mChoose2);
    %pair = allPairs(i,:);
    %fprintf('testing pair %3d/%d: (%d,
%d)\n',i,mChoose2,pair(1),pair(2));
    allPredictions(:,i) = svmclassify(allSVMs{i}, Mtest);
end
close(h);
testTime = toc;
fprintf('Total test time: %0.2f seconds\n\n', testTime);

% majority vote
yPredictions = mode(allPredictions,2);
% overall CCR
CCR = sum(yPredictions==testlabel)/length(testlabel);
fprintf('Overall CCR: %0.4f\n\n', CCR);

% confusion matrix of test set
conf = confusionmat(testlabel,yPredictions)';
disp(conf);
testLabelTotals = accumarray(testlabel(:),1);

% double check that confusion matrix columns sum to label totals
fprintf('\n\nconfusion matrix column totals:\n');
disp(sum(conf))
fprintf('test data label totals:\n');
disp(testLabelTotals')
fprintf('conf mat totals - test label totals:\n');
disp(sum(conf)-testLabelTotals')
fprintf('Seems to be missing one in classification for doc #19...\n
\n');

% determine most commonly classified label
[~,maxInd] = max(sum(conf,2));
mostCommonDoc = classes(maxInd);
fprintf('Most commonly classified document label: %s (label #%d)\n
\n', ...
    char(mostCommonDoc),maxInd);
```

```
Starting part (e)...

Training all binary SVM pairs with linear kernel...

Total training time: 46.87 seconds

Testing all binary SVM pairs...

Total test time: 61.11 seconds

Overall CCR: 0.3083
```

```
 Columns 1 through 13

   68     0     0     0     0     0     0     2     1     4     0
1     0
    0    46    10     7     0     9     1     1     0     1     0
1     1
    0     4   121     8     2    14     0     1     0     0     0
1     0
    0     1    15   100    20     0    30     0     0     0     0
0     1
    0     1     4     1    68     0     5     0     0     0     0
1     2
    0     0     7     0     0    28     0     0     0     0     0
0     0
    0     1     1     2     2     0    72     2     1     2     1
0     1
    0     0     0     0     0     0     3    58     3     0     0
0     0
    0     0     0     0     0     0     0     0   110     0     0
0     0
    0     0     1     1     0     0     0     2     1   222    44
0     0
    0     0     0     1     1     0     0     0     0     7   183
0     0
    1     0     1     0     0     3     1     0     0     0     0
85     2
  174   331   215   269   282   331   261   297   253   134   155
206   378
    9     1     2     1     3     0     3    10    12    10     3
7     2
    2     1     1     0     1     4     1     1     0     0     1
0     2
   25     0     0     0     0     0     0     0     0     0     1
0     0
   37     3    13     2     4     1     5    21    16    17    11
93     4
    0     0     0     0     0     0     0     0     0     0     0
0     0
    0     0     0     0     0     0     0     0     0     0     0
0     0
    2     0     0     0     0     0     0     0     0     0     0
0     0

 Columns 14 through 20

    3     0     6     1     7     1    17
    0     4     0     0     0     0     0
    0     0     1     0     0     0     0
    0     0     0     0     0     0     0
    0     1     0     0     0     0     0
    1     0     0     0     0     0     0
    2     0     0     2     0     0     0
    1     0     0     0     0     0     0
```

```
     0     0     0     0     0     0     0
     1     1     0     0     3     1     0
     0     0     0     0     0     0     0
     0     0     0     1     0     0     0
   252   214   242    43   203    94   118
   123     6     6     3     8    14    11
     0   147     2     0     0     0     0
     2     0   106     0     0     0    27
     7    19    35   314   115   166    65
     0     0     0     0    39     0     0
     0     0     0     0     0    33     0
     1     0     0     0     1     0    13


confusion matrix column totals:
  Columns 1 through 13

   318   389   391   392   383   390   382   395   397   397   399
 395   393

  Columns 14 through 20

   393   392   398   364   376   309   251

test data label totals:
  Columns 1 through 13

   318   389   391   392   383   390   382   395   397   397   399
 395   393

  Columns 14 through 20

   393   392   398   364   376   310   251

conf mat totals - test label totals:
  Columns 1 through 13

     0     0     0     0     0     0     0     0     0     0     0
 0     0

  Columns 14 through 20

     0     0     0     0     0    -1     0

Seems to be missing one in classification for doc #19...

Most commonly classified document label: sci.electronics (label #13)
```

Published with MATLAB® R2017a

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

Setup

```
% 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 (f) : One-versus-one OVO multi-class classification rbf kernel

```
fprintf('\nStarting part (f)...\n\n');

% all combinations and count
allPairs = combnk(1:20,2);
mChoose2 = nchoosek(20,2);

% train m(m-1)/2=190 binary SVMs for all class pairs
tic
allSVMs = cell(1,mChoose2);
fprintf('Training all binary SVM pairs with rbf kernel...\n\n');
h = waitbar(0, 'Training all binary SVM pairs...','Name','Part (e)');
for p=1:mChoose2
    waitbar(p/mChoose2);
    % select pair
    pair = allPairs(p,:);
    trainDataPair = sparse(Mtrain((trainlabel==pair(1) | ...
        trainlabel==pair(2)),:));
    trainLabelPair = trainlabel(trainlabel==pair(1) |
 trainlabel==pair(2));
    % train pair
    %fprintf('training pair %3d/%d: (%d,
%d)\n',p,mChoose2,pair(1),pair(2));
    SVMStruct = svmtrain(trainDataPair, trainLabelPair, ...
        'kernel_function','rbf','autoscale','false', ...
        'kernelcachelimit', 20000);
    allSVMs{p} = SVMStruct;
end
close(h);
trainingTime = toc;
fprintf('Total training time: %0.2f seconds\n\n', trainingTime);


% test on all binary SVM pairs
```

```
tic
allPredictions = zeros(length(testlabel),mChoose2);
fprintf('Testing all binary SVM pairs...\n\n');
h = waitbar(0, 'Testing all binary SVM pairs...','Name','Part (f)');
for i=1:mChoose2
    waitbar(i/mChoose2);
    %pair = allPairs(i,:);
    %fprintf('testing pair %3d/%d: (%d,
%d)\n',i,mChoose2,pair(1),pair(2));
    allPredictions(:,i) = svmclassify(allSVMs{i}, Mtest);
end
close(h);
testTime = toc;
fprintf('Total test time: %0.2f seconds\n\n', testTime);

% majority vote
yPredictions = mode(allPredictions,2);
% overall CCR
CCR = sum(yPredictions==testlabel)/length(testlabel);
fprintf('Overall CCR: %0.4f\n\n', CCR);

% confusion matrix of test set
conf = confusionmat(testlabel,yPredictions)';
disp(conf);
testLabelTotals = accumarray(testlabel(:),1);

% double check that confusion matrix columns sum to label totals
fprintf('\n\nconfusion matrix column totals:\n');
disp(sum(conf))
fprintf('test data label totals:\n');
disp(testLabelTotals')
fprintf('conf mat totals - test label totals:\n');
disp(sum(conf)-testLabelTotals')
fprintf('Seems to be missing one in classification for doc #19...\n
\n');

% determine most commonly classified label
[~,maxInd] = max(sum(conf,2));
mostCommonDoc = classes(maxInd);
fprintf('Most commonly classified document label: %s (label #%d)\n
\n', ...
    char(mostCommonDoc),maxInd);
```

```
Starting part (f)...

Training all binary SVM pairs with rbf kernel...

Total training time: 56.37 seconds

Testing all binary SVM pairs...

Total test time: 155.93 seconds
```

```
Overall CCR: 0.3141

  Columns 1 through 13

   56     0     0     0     0     0     0     1     1     2     0
0     0
    2   136    38    18     4    30     2     1     0     0     0
6     4
    0     4   131     7     2    15     0     1     0     0     0
1     0
    0     2    23   133    30     2    41     0     0     0     0
0     3
    0     1     3     0    71     0     9     0     0     0     0
0     3
    0     0     7     0     1    82     0     1     0     0     0
0     0
    0     1     0     2     2     3   110     1     1     2     1
0     1
    0     0     0     0     0     0     5    56     1     0     0
0     0
    0     0     0     0     0     0     0     0   118     0     0
0     0
    0     0     1     1     0     0     1     1     0   107    13
0     0
    0     0     0     1     1     0     0     0     0    15   179
0     0
    1     1     1     1     0     4     1     0     0     0     0
87     2
    6   106    58   140   132   120   115    67    18    27    19
19   188
   88    88    71    53    76    76    45   113   127   122    54
73   108
    1     2     1     2     1     5     2     0     0     0     1
0     0
   20     0     0     0     0     0     0     0     0     0     1
0     0
  142    48    57    34    63    53    51   153   131   122   131
209    84
    0     0     0     0     0     0     0     0     0     0     0
0     0
    0     0     0     0     0     0     0     0     0     0     0
0     0
    2     0     0     0     0     0     0     0     0     0     0
0     0

  Columns 14 through 20

    2     0    13     0     3     1    12
    1     3     2     1     0     1     2
    0     0     1     0     0     0     0
    0     0     0     0     0     0     0
    0     1     0     0     0     0     0
    1     0     0     0     0     0     0
    4     0     0     2     0     0     0
```

```
     0     0     0     1     0     0     0
     0     0     0     0     0     0     0
     1     0     0     0     2     0     0
     0     0     0     0     0     0     0
     0     0     0     1     0     0     0
    37    15    25     5     5     4     5
   327   162   110    12    48    37    62
     0    95     1     0     0     0     0
     2     0    76     0     0     0    23
    18   116   170   342   294   239   135
     0     0     0     0    24     0     0
     0     0     0     0     0    27     0
     0     0     0     0     0     0    12


confusion matrix column totals:
  Columns 1 through 13

   318   389   391   392   383   390   382   395   397   397   399
 395   393

  Columns 14 through 20

   393   392   398   364   376   309   251

test data label totals:
  Columns 1 through 13

   318   389   391   392   383   390   382   395   397   397   399
 395   393

  Columns 14 through 20

   393   392   398   364   376   310   251

conf mat totals - test label totals:
  Columns 1 through 13

     0     0     0     0     0     0     0     0     0     0     0
 0     0

  Columns 14 through 20

     0     0     0     0     0    -1     0

Seems to be missing one in classification for doc #19...

Most commonly classified document label: talk.politics.guns (label
 #17)
```

Published with MATLAB® R2017a