Table of Contents


```
% 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']);
```

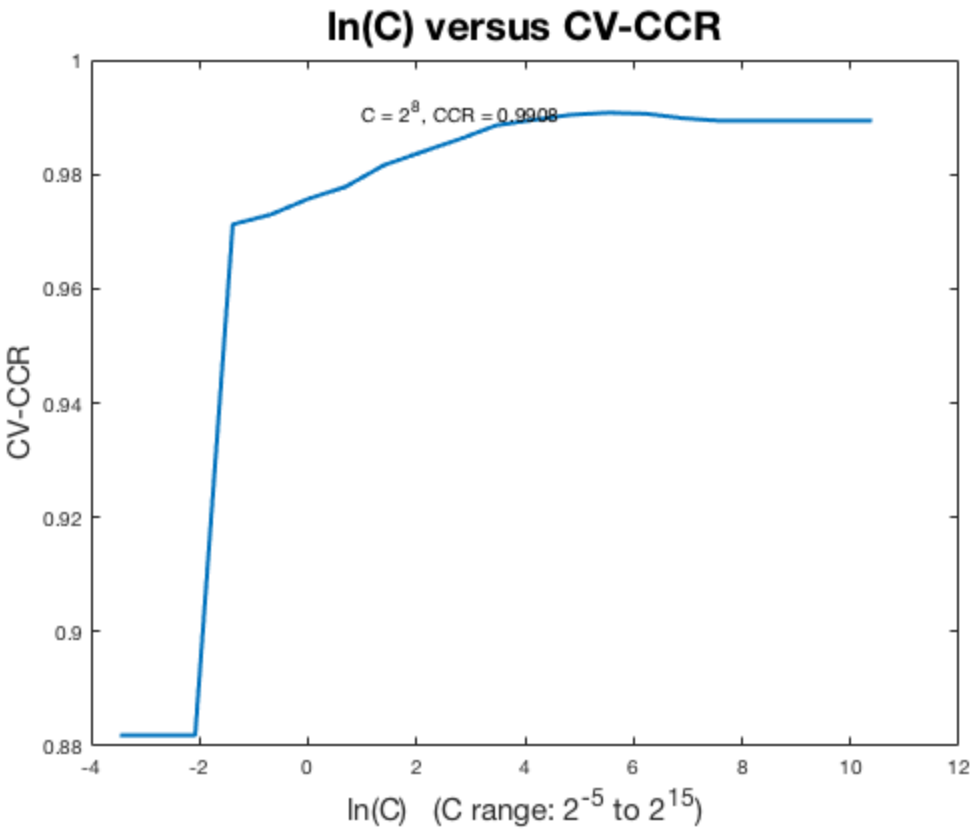
C is 2^8 and corresponding CV-CCR value is 0.9908*

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.

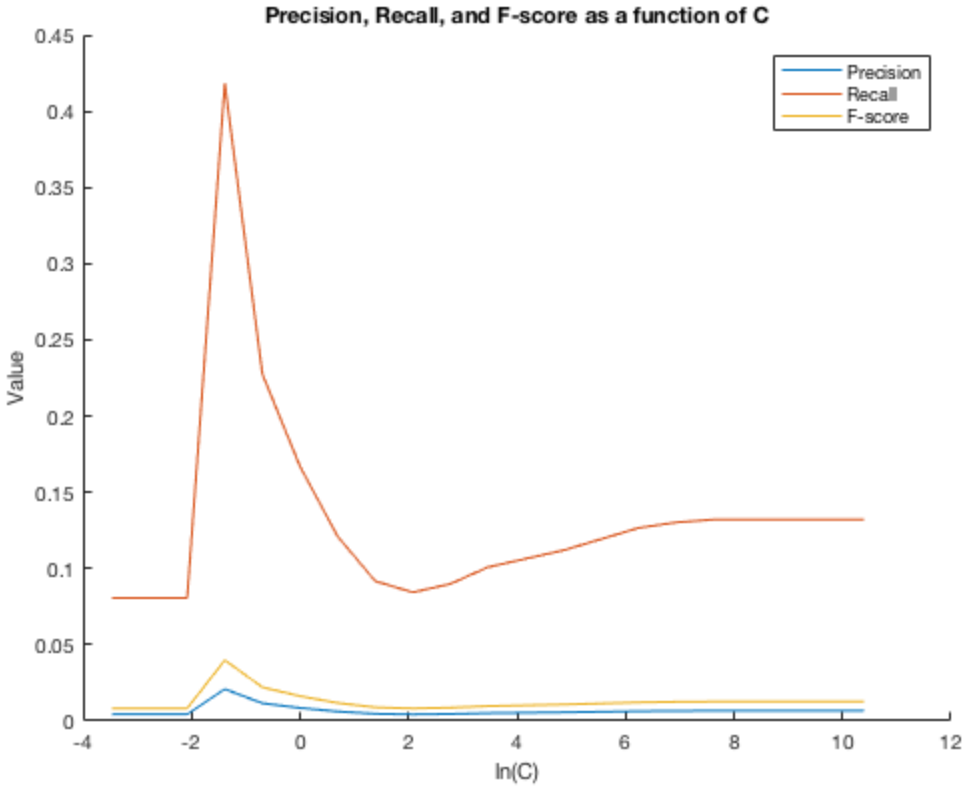


Part (d) : Performance metrics

```
% (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