Silvia Ionescu
10-28-2016

Learning from Data - Assignment 6

In this assignment we classified text documents in the 20newsgroup dataset using SVM classifier. The train and test data sets contain information about the document id, word id, and word count. The train and test label sets contained labels of the training and testing documents. The vocabulary list contains all the words that can possibly appear in the documents.

Part 6.1a

Trained a binary SVM classifier using the linear kernel to separate between class1 (alt.atheism) and class20 (talk.religion.misc). I pulled out from the training and testing datasets only the documents that were classified as class 1 or 20. Removed all the stop words from the original documents and calculated the word frequency vector for each document according to Eq. 1.

$$x_{w,j} = \frac{n_{w,j}}{n_j} \qquad (1)$$

Where $x_{w,j}$ is the word frequency vector, $n_{w,j}$ is the the number of times a word appears in document j, and n_j is the total number of words in the document.
Parameter C, ”boxconstraint”, was determined via cross validation, by creating a 5-fold cross-validation partitioning by splitting the training data into 5 equal disjoint parts uniformly at random. The CV-CCR was calculated for different values of C by first training on 4 out of 5 folds and testing on the remaining one. This was repeated 5 times with a different split as a test set and the average CCR was calculated across the 5 splits.

i) The CV-CCR as a function of C values ranging from 2^{-5} to 2^{15} is shown below:

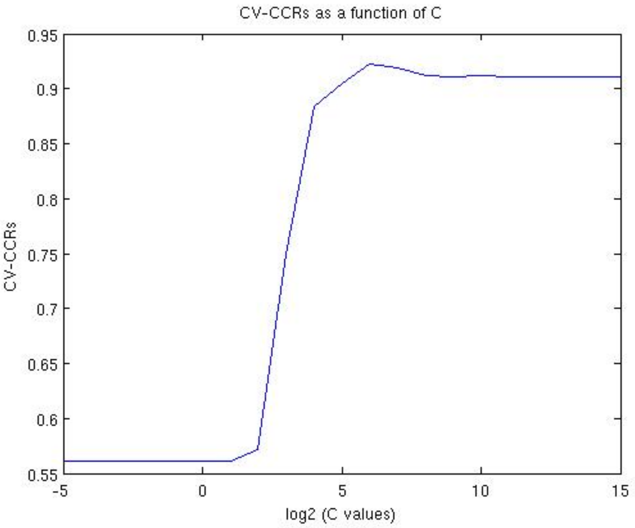

Fig. 1: CV-CCR as a function of C values

ii) The value of C that achieves the best CCR is:

C = 2^6 = 64

iii) Used the above C value to train the linear SVM classifier on all the data in the training set, and tested it on the test dataset.
CCR = 0.8225

Part 6.1b

Trained a binary SVM classifier by using RBF kernel to distinguish between class 1 and class 20. The goal is to determine the best pair of C and RBF_sigma values by splitting the training set into 5 partitions and calculate CV-CCR. As in part a, C was taken in the range of 10^{-5}, …, 10^{15} and RBF_sigma was taken between 10^{-13}, …, 10^{3}.

i) CV-CCR as a function of C and RBF_sigma values is shown below

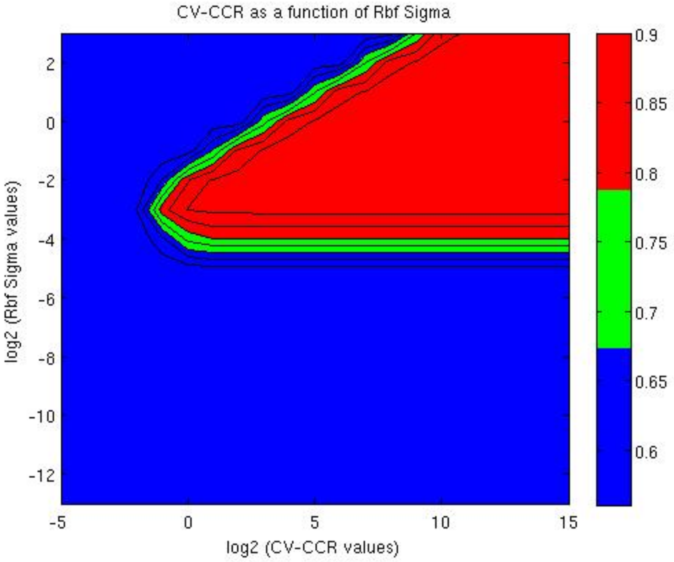

Fig. 2: CV-CCRs as a function of C and RBF_sigma values.

ii) The best pair is:
C_max = 2^9 = 512
Rbf_sigma_max = 2^0 = 1

iii) The CCR for the entire training and testing datasets using the best pair of C and rbf_sigma values is:
CCR = 0.7856

Part 6.1c

Train a binary SVM using a linear kernel by separating the training set into class # 17(positive samples) and not class #17 (negative samples). The problem is that the number of positive samples are less then the number of negative samples. If we follow the standard method $C_+ = C_- = C$, then the training objective function is dominated by the negative class samples. In order to mitigate for this problem different penalties C_+ and C_- can be set for different classes. The matlab function svmtrain does that for us if we C to be a scalar. The kernel was linear and autoscale was set to false.

i) The CV-CCR as a function of C values ranging from 2^{-5} to 2^{15} is shown below:

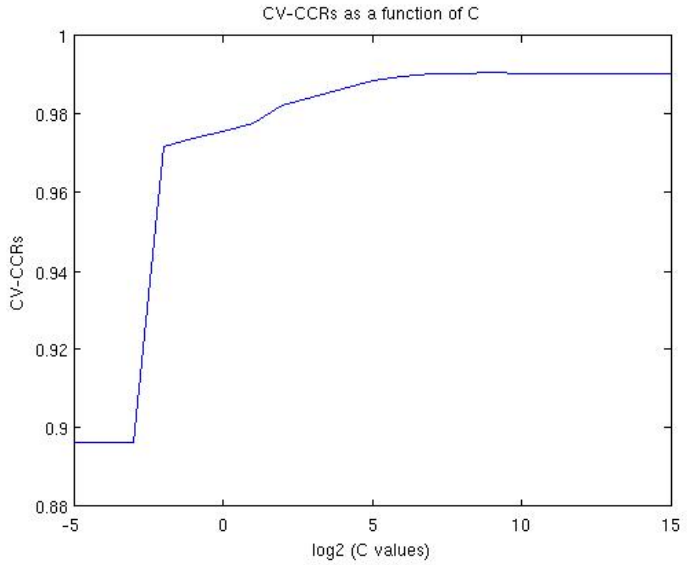

Fig. 3: CV-CCR as a function of C values

ii) The value of C that achieves the best CCR is: C = 512

iii) Used the above C value to train the linear SVM classifier on all the data in the training set, and tested it on the test dataset.

CCR = 0.9714

The confusion matrix of the test data is shown below:

	Truth	
Decision	7045.0	96.0
	119.0	245.0

Table1: Confusion matrix for all training data and test data.

We can observe that the number of predicted labels in one class is a lot bigger then the number of predicted labels in the other class. The confusion matrix shows the imbalance of the dataset.

Part 6.1d

The cross-validation precision, recall, and F-score were plotted as a function of C and the graph is shown below.

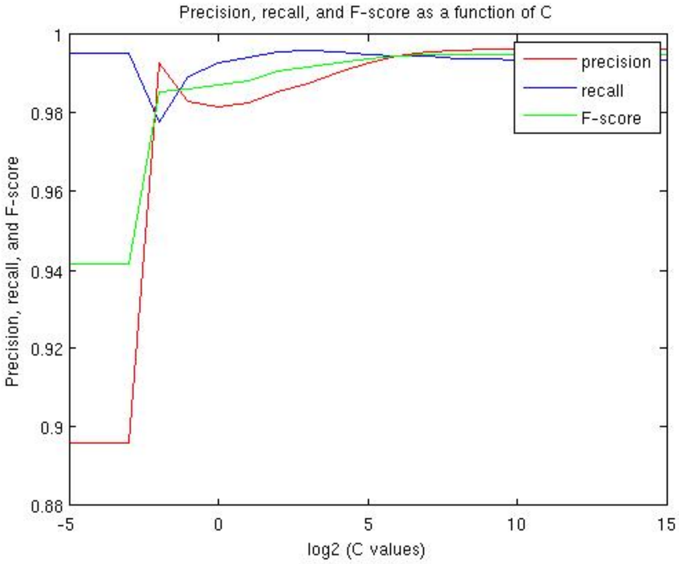

Fig. 4: Precision, recall, and F-score as a function of C.

ii) The best value of C in terms of recall is C = 8 and the best value in terms of F-score is C = 512.

The confusion matrix for the recall C value is shown below:

	Truth	
Decision	6961.0	180.0
	66.0	298.0

The confusion matrix for the F-score C value is:

	Truth	
Decision	7045.0	96.0
	119.0	245.0

Part 6.1e – OVO multi-class classification – linear kernel

For this part I build a multi-class SVM classifier by training m(m-1)/2 binary SVM's for all the class pairs. For testing I applied all m(m-1)/2 binary SVM's to the test sample and predicted the label by taking the label with most votes. Used a linear kernel with default Matlab regularization parameter C.

i) CCR = 0.3087

ii) Training time: 53.57
Test time = 50.48

iii) Confusion matrix

	Truth																			
Decision	68	0	0	0	0	0	0	2	1	4	0	1	0	3	0	6	1	7	1	17
	0	46	10	7	0	9	1	1	0	1	0	1	1	0	4	0	0	0	0	0
	0	4	121	8	2	14	0	1	0	0	0	1	0	0	0	1	0	0	0	0
	0	1	15	100	20	0	30	0	0	0	0	0	1	0	0	0	0	0	0	0
	0	1	4	1	68	0	5	0	0	0	0	1	2	0	1	0	0	0	0	0
	0	0	7	0	0	28	0	0	0	0	0	0	0	1	0	0	0	0	0	0
	0	1	1	2	2	0	72	2	1	2	1	0	1	2	0	0	2	0	0	0
	0	0	0	0	0	0	3	58	3	0	0	0	0	1	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	110	0	0	0	0	0	0	0	0	0	0	0
	0	0	1	1	0	0	0	2	1	222	44	0	0	1	1	0	0	3	1	0
	0	0	0	1	1	0	0	0	0	7	186	0	0	0	0	0	0	0	0	0
	1	0	1	0	0	3	1	0	0	0	0	85	2	0	0	0	1	0	0	0
	174	331	215	269	282	331	261	297	253	134	152	206	378	252	214	242	43	203	95	118
	9	1	2	1	3	0	3	10	12	10	3	7	2	123	6	6	3	8	14	11
	2	1	1	0	1	4	1	1	0	0	1	0	2	0	147	2	0	0	0	0
	25	0	0	0	0	0	0	0	0	0	1	0	0	2	0	106	0	0	0	27
	37	3	13	2	4	1	5	21	16	17	11	93	4	7	19	35	314	115	166	65
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	39	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	33	0
	2	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	13

Part 6.1f – OVO multi-class classification – RBF kernel

Applied OVO as in part e, but this time used RBF kernel.

i) CCR = 0.3139
ii) Training time = 61.02
Test time = 100.73

iii) Confusion matrix of the test set is shown below:

	Truth																			
Decision	56	0	0	0	0	0	0	1	1	2	0	0	0	2	0	13	0	3	1	12
	2	136	38	18	4	30	2	1	0	0	0	6	4	1	3	2	1	0	1	2
	0	4	131	7	2	15	0	1	0	0	0	1	0	0	0	1	0	0	0	0
	0	2	23	133	30	2	41	0	0	0	0	0	3	0	0	0	0	0	0	0
	0	1	3	0	71	0	9	0	0	0	0	0	3	0	1	0	0	0	0	0
	0	0	7	0	1	82	0	1	0	0	0	0	0	1	0	0	0	0	0	0
	0	1	0	2	2	3	110	1	1	2	1	0	1	4	0	0	2	0	0	0
	0	0	0	0	0	0	5	56	1	0	0	0	0	0	0	0	1	0	0	0
	0	0	0	0	0	0	0	0	118	0	0	0	0	0	0	0	0	0	0	0
	0	0	1	1	0	0	1	1	0	106	13	0	0	1	0	0	0	2	0	0
	0	0	0	1	1	0	0	0	0	16	179	0	0	0	0	0	0	0	0	0
	1	1	1	1	0	4	1	0	0	0	0	87	2	0	0	0	1	0	0	0
	6	106	58	140	132	120	115	67	18	27	19	19	188	37	15	25	5	5	4	5
	88	88	71	53	76	76	45	113	127	122	54	73	108	327	162	110	12	48	37	62
	1	2	1	2	1	5	2	0	0	0	1	0	0	0	95	1	0	0	0	0
	20	0	0	0	0	0	0	0	0	0	1	0	0	2	0	76	0	0	0	23
	142	48	57	34	63	53	51	153	131	122	131	209	84	18	116	170	342	294	240	135
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	24	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	27	0
	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12

```
% Assignemnt 6_1a
% Train a binary SVM classifier using a linear kernel
% Silvia Ionescu

clear; close all; clc;
```

```
% train and test datasets
train_data = load('train.data');
train_label = load('train.label');

test_data = load('test.data');
test_label = load('test.label');

vocabulary = textread('vocabulary.txt', '%s');
stoplist = textread('stoplist.txt', '%s');


% find the index for the stoplist words in vocabulary
stoplist = unique(stoplist);
p = 0;
for i =1: size(stoplist, 1);
    index = strmatch(stoplist(i), vocabulary, 'exact');
    if ~isempty(index)
        p = p + 1;
        remove_list(p,1) = index;
    end
end

% set the words in stoplist to zero frequency
for i = 1: length(remove_list)
    train_loc = find(train_data(:,2) == remove_list(i));
    train_data(train_loc, 3) = 0;

    test_loc = find(test_data(:,2) == remove_list(i));
    test_data(test_loc, 3) = 0;
end

% documets id that are part of class1 and 20
training_class1 = find(train_label == 1);
training_class20 = find(train_label == 20);
index_class1_20 = [training_class1; training_class20];

% labels for class1 and class20
train_label1_20 = train_label(index_class1_20);

train_word_frequency = zeros(length(index_class1_20), length(vocabulary));
for i = 1:length(index_class1_20)
    b = find(train_data(:,1) == index_class1_20(i));
    samples = train_data(b,:);

    % total number of words in the document
    words_per_doc = sum(samples(:,3),1);
    for j = 1: size(samples,1)
        train_word_frequency(i,samples(j,2)) = samples(j,3)/words_per_doc;
    end
end

% test for class1 and class20
test_class1 = find(test_label == 1);
test_class20 = find(test_label == 20);
index_test_class1_20 = [test_class1; test_class20] ;
```

```
% labels for class1 and class20
test_label1_20 = test_label(index_test_class1_20);

test_word_frequency = zeros(length(index_test_class1_20),
length(vocabulary));
for k = 1:length(index_test_class1_20)
    test_data_index = find(test_data(:,1)== index_test_class1_20(k));
    test_samples = train_data(test_data_index,:);

    % total number of words in the document
    test_words_per_doc = sum(test_samples(:,3),1);
    for l = 1:size(test_samples)
        test_word_frequency(k,test_samples(l,2)) =
test_samples(l,3)/test_words_per_doc;
    end
end


vocab_index = 1:length(vocabulary);
train_index_diff = setdiff(vocab_index, remove_list);

% remove stoplist words from train_data
train_word_frequency_new = train_word_frequency(:,train_index_diff);

% remove stoplist words from test_data
test_word_frequency_new = test_word_frequency(:,train_index_diff);


s = RandStream('mt19937ar','Seed',1);

random_train = randperm(s,size(train_word_frequency_new,1));
split1 = train_word_frequency_new(random_train(1:171),:);
split2 = train_word_frequency_new(random_train(172:342),:);
split3 = train_word_frequency_new(random_train(343:513),:);
split4 = train_word_frequency_new(random_train(514:684),:);
split5 = train_word_frequency_new(random_train(685:856),:);

label_split1 = train_label1_20(random_train(1:171),:);
label_split2 = train_label1_20(random_train(172:342),:);
label_split3 = train_label1_20(random_train(343:513),:);
label_split4 = train_label1_20(random_train(514:684),:);
label_split5 = train_label1_20(random_train(685:856),:);

C = [2^-5, 2^-4, 2^-3, 2^-2, 2^-1, 1, 2, 2^2, 2^3, 2^4, 2^5, 2^6, 2^7, 2^8,
2^9, 2^10, 2^11, 2^12, 2^13, 2^14, 2^15];
C = C';
for i = 1:length(C)
    disp(i);
    % test on split 5
    train1 = [split1; split2; split3; split4];
    label1 = [label_split1; label_split2; label_split3; label_split4];
    SVMStruct1 = svmtrain(train1, label1,'boxconstraint',
C(i)*ones(size(train1,1),1), 'autoscale', 'false');
    Group1 = svmclassify(SVMStruct1,split5);
    C1 = confusionmat(label_split5,Group1);
    CCR1 = sum(diag(C1))/ sum(sum(C1));

    % test on split 4
    train2 = [split1; split2; split3; split5];
    label2 = [label_split1;label_split2; label_split3; label_split5];
    SVMStruct2 = svmtrain(train2,
label2,'boxconstraint',C(i)*ones(size(train2,1),1), 'autoscale', 'false');
    Group2 = svmclassify(SVMStruct2,split4);
    C2 = confusionmat(label_split4,Group2);
    CCR2 = sum(diag(C2))/ sum(sum(C2));

    % test on split 3
    train3 = [split1; split2; split4; split5];
    label3 = [label_split1;label_split2; label_split4; label_split5];
    SVMStruct3 = svmtrain(train3, label3,'boxconstraint',
C(i)*ones(size(train3,1),1), 'autoscale', 'false');
    Group3 = svmclassify(SVMStruct3,split3);
    C3 = confusionmat(label_split3,Group3);
    CCR3 = sum(diag(C3))/ sum(sum(C3));

    % test on split 2
    train4 = [split1; split3; split4; split5];
    label4 = [label_split1; label_split3; label_split4; label_split5];
    SVMStruct4 = svmtrain(train4, label4,'boxconstraint',
C(i)*ones(size(train4,1),1), 'autoscale', 'false');
    Group4 = svmclassify(SVMStruct4,split2);
    C4 = confusionmat(label_split2,Group4);
    CCR4 = sum(diag(C4))/ sum(sum(C4));

    % test on split 1
    train5 = [split2; split3; split4; split5];
    label5 = [label_split2;label_split3; label_split4; label_split5];
    SVMStruct5 = svmtrain(train5, label5,'boxconstraint',
C(i)*ones(size(train5,1),1), 'autoscale', 'false');
    Group5 = svmclassify(SVMStruct5,split1);
    C5 = confusionmat(label_split1,Group5);
    CCR5 = sum(diag(C5))/ sum(sum(C5));

    CCR(i) = (CCR1+CCR2+CCR3+CCR4+CCR5)/5;
end

figure(1)
plot(log2(C), CCR);
title('CV-CCRs as a function of C');
xlabel('log2 (C values)');
ylabel('CV-CCRs');

[CCR_max, CCR_max_index] = max(CCR);
C_max = C(CCR_max_index);

SVMStruct_all = svmtrain(train_word_frequency_new,
train_label1_20,'boxconstraint',
C_max*ones(size(train_word_frequency_new,1),1), 'autoscale', 'false');
Group_all = svmclassify(SVMStruct_all,test_word_frequency_new);
```

```
confusion_matrix = confusionmat(test_label1_20,Group_all);
CCR_all = sum(diag(confusion_matrix))/ sum(sum(confusion_matrix));
```

```
% Assignemnt 6_1b
% Train a binary SVM classifier using a RBF kernel
% Silvia Ionescu

clear; close all; clc;

% train and test datasets
train_data = load('train.data');
train_label = load('train.label');

test_data = load('test.data');
```

```
test_label = load('test.label');

vocabulary = textread('vocabulary.txt', '%s');
stoplist = textread('stoplist.txt', '%s');

% find the index for the stoplist words in vocabulary
stoplist = unique(stoplist);
p = 0;
for i =1: size(stoplist, 1);
    index = strmatch(stoplist(i), vocabulary, 'exact');
    if ~isempty(index)
        p = p + 1;
        remove_list(p,1) = index;
    end
end

% set the words in stoplist to zero frequency
for i = 1: length(remove_list)
    train_loc = find(train_data(:,2) == remove_list(i));
    train_data(train_loc, 3) = 0;

    test_loc = find(test_data(:,2) == remove_list(i));
    test_data(test_loc, 3) = 0;
end


% documets id that are part of class1 and 20
training_class1 = find(train_label == 1);
training_class20 = find(train_label == 20);
index_class1_20 = [training_class1; training_class20];

% labels for class1 and class20
train_label1_20 = train_label(index_class1_20);

train_word_frequency = zeros(length(index_class1_20), length(vocabulary));
for n = 1:length(index_class1_20)
    b = find(train_data(:,1) == index_class1_20(n));
    samples = train_data(b,:);

    % total number of words in the document
    words_per_doc = sum(samples(:,3),1);
    for j = 1: size(samples,1)
        train_word_frequency(n,samples(j,2)) = samples(j,3)/words_per_doc;
    end
end

% test for class1 and class20
test_class1 = find(test_label == 1);
test_class20 = find(test_label == 20);
index_test_class1_20 = [test_class1; test_class20] ;

% labels for class1 and class20
test_label1_20 = test_label(index_test_class1_20);

test_word_frequency = zeros(length(index_test_class1_20),
length(vocabulary));
for k = 1:length(index_test_class1_20)
    test_data_index = find(test_data(:,1)== index_test_class1_20(k));
    test_samples = train_data(test_data_index,:);

    % total number of words in the document
    test_words_per_doc = sum(test_samples(:,3),1);
    for l = 1:size(test_samples)
        test_word_frequency(k,test_samples(l,2)) =
test_samples(l,3)/test_words_per_doc;
    end
end

vocab_index = 1:length(vocabulary);
train_index_diff = setdiff(vocab_index, remove_list);

% remove stoplist words from train_data
train_word_frequency_new = train_word_frequency(:,train_index_diff);

% remove stoplist words from test_data
test_word_frequency_new = test_word_frequency(:,train_index_diff);

s = RandStream('mt19937ar','Seed',1);

random_train = randperm(s,size(train_word_frequency_new,1));
split1 = train_word_frequency_new(random_train(1:171),:);
split2 = train_word_frequency_new(random_train(172:342),:);
split3 = train_word_frequency_new(random_train(343:513),:);
split4 = train_word_frequency_new(random_train(514:684),:);
split5 = train_word_frequency_new(random_train(685:856),:);

label_split1 = train_label1_20(random_train(1:171),:);
label_split2 = train_label1_20(random_train(172:342),:);
label_split3 = train_label1_20(random_train(343:513),:);
label_split4 = train_label1_20(random_train(514:684),:);
label_split5 = train_label1_20(random_train(685:856),:);

C = [2^-5, 2^-4, 2^-3, 2^-2, 2^-1, 2^0, 2^1, 2^2, 2^3, 2^4, 2^5, 2^6, 2^7,
2^8, 2^9, 2^10, 2^11, 2^12, 2^13, 2^14, 2^15];
rbf_sigma = [2^-13, 2^-12, 2^-11, 2^-10, 2^-9, 2^-8, 2^-7, 2^-6, 2^-5, 2^-4,
2^-3, 2^-2, 2^-1, 2^0, 2^1, 2^2, 2^3];

for n = 1:length(C)
    for m = 1:length(rbf_sigma)
        disp(n);
        % test on split 5
        train1 = [split1; split2; split3; split4];
        label1 = [label_split1; label_split2; label_split3; label_split4];
        SVMStruct1 = svmtrain(train1, label1,'boxconstraint',
C(n)*ones(size(train1,1),1),'kernel_function','rbf','rbf_sigma',
rbf_sigma(m), 'autoscale', 'false');
        Group1 = svmclassify(SVMStruct1,split5);
        C1 = confusionmat(label_split5,Group1);
        CCR1 = sum(diag(C1))/ sum(sum(C1));

        % test on split 4
        train2 = [split1; split2; split3; split5];
        label2 = [label_split1;label_split2; label_split3; label_split5];
        SVMStruct2 = svmtrain(train2,
label2,'boxconstraint',C(n)*ones(size(train2,1),1),'kernel_function','rbf','r
bf_sigma', rbf_sigma(m), 'autoscale', 'false');
        Group2 = svmclassify(SVMStruct2,split4);
        C2 = confusionmat(label_split4,Group2);
        CCR2 = sum(diag(C2))/ sum(sum(C2));

        % test on split 3
        train3 = [split1; split2; split4; split5];
        label3 = [label_split1;label_split2; label_split4; label_split5];
        SVMStruct3 = svmtrain(train3, label3,'boxconstraint',
C(n)*ones(size(train3,1),1),'kernel_function','rbf','rbf_sigma',
rbf_sigma(m), 'autoscale', 'false');
        Group3 = svmclassify(SVMStruct3,split3);
        C3 = confusionmat(label_split3,Group3);
        CCR3 = sum(diag(C3))/ sum(sum(C3));

        % test on split 2
        train4 = [split1; split3; split4; split5];
        label4 = [label_split1; label_split3; label_split4; label_split5];
        SVMStruct4 = svmtrain(train4, label4,'boxconstraint',
C(n)*ones(size(train4,1),1),'kernel_function','rbf','rbf_sigma',
rbf_sigma(m), 'autoscale', 'false');
        Group4 = svmclassify(SVMStruct4,split2);
        C4 = confusionmat(label_split2,Group4);
        CCR4 = sum(diag(C4))/ sum(sum(C4));

        % test on split 1
        train5 = [split2; split3; split4; split5];
        label5 = [label_split2;label_split3; label_split4; label_split5];
        SVMStruct5 = svmtrain(train5, label5,'boxconstraint',
C(n)*ones(size(train5,1),1),'kernel_function','rbf','rbf_sigma',
rbf_sigma(m), 'autoscale', 'false');
        Group5 = svmclassify(SVMStruct5,split1);
        C5 = confusionmat(label_split1,Group5);
        CCR5 = sum(diag(C5))/ sum(sum(C5));

CCR(n,m) = (CCR1+CCR2+CCR3+CCR4+CCR5)/5;
    end
end

figure(1)
CCR = CCR';
contourf(log2(C), log2(rbf_sigma),CCR);
map = [ 0 0 1; 0 1 0;1 0 0];
colormap(map);
%set(gca,'xscale','log', 'yscale','log');
colorbar;
xlabel('log2 (CV-CCR values)');
ylabel('log2 (Rbf Sigma values)');
title('CV-CCR as a function of Rbf Sigma');


[num, idx] = max(CCR(:));
```

```
[x, y] = ind2sub(size(CCR),idx);

C_max = C(x);
rbf_sigma_max = rbf_sigma(y);

SVMStruct_all = svmtrain(train_word_frequency_new,
train_label1_20,'boxconstraint',
C_max*ones(size(train_word_frequency_new,1),1),'kernel_function','rbf','rbf_s
igma', rbf_sigma_max, 'autoscale', 'false');
Group_all = svmclassify(SVMStruct_all,test_word_frequency_new);

confusion_matrix = confusionmat(test_label1_20,Group_all);
CCR_all = sum(diag(confusion_matrix))/ sum(sum(confusion_matrix));



% Assignemnt 6_1cd
% Train a binary SVM classifier using a linear kernel and C is scalar
% in order to mitigate the unbalenced dataset.
% Calculate CV-CCR as in part a
% Calculate CV precision, recall, and F-score

% Silvia Ionescu
clear; close all; clc;

% train and test datasets
train_data = load('train.data');
train_label = load('train.label');

test_data = load('test.data');
test_label = load('test.label');

vocabulary = textread('vocabulary.txt', '%s');
stoplist = textread('stoplist.txt', '%s');

% find the index for the stoplist words in vocabulary
stoplist = unique(stoplist);
p = 0;
for i =1: size(stoplist, 1);
    index = strmatch(stoplist(i), vocabulary, 'exact');
    if ~isempty(index)
        p = p + 1;
        remove_list(p,1) = index;
    end
end

% set the words in stoplist to zero frequency
for i = 1: length(remove_list)
    train_loc = find(train_data(:,2) == remove_list(i));
    train_data(train_loc, 3) = 0;

    test_loc = find(test_data(:,2) == remove_list(i));
    test_data(test_loc, 3) = 0;
end
```

```
% training - documets id that are part of class 17
training_class17 = find(train_label == 17);
training_classNot17 = setdiff(1:length(train_label), training_class17);

% labels for class 17
train_label17 = train_label(training_class17);
train_labelNot17 = train_label(training_classNot17);

% label the classes that are not 17 equal to class 1
train_labelNot17(:,1) = 1;

% calculate the word frequency for class 17
train_word_frequency17 = zeros(length(training_class17), length(vocabulary));
for i = 1:length(training_class17)
    b = find(train_data(:,1) == training_class17(i));
    samples = train_data(b,:);

    % total number of words in the document
    words_per_doc = sum(samples(:,3),1);
    for j = 1: size(samples,1)
        train_word_frequency17(i,samples(j,2)) = samples(j,3)/words_per_doc;
    end
end


% calculate the word frequency for class 1
train_word_frequencyNot17 = zeros(length(training_classNot17),
length(vocabulary));
for i = 1:length(training_classNot17)
    d = find(train_data(:,1) == training_classNot17(i));
    samples_Not17 = train_data(d,:);

    % total number of words in the document
    words_per_doc = sum(samples_Not17(:,3),1);
    for j = 1: size(samples_Not17,1)
        train_word_frequencyNot17(i, samples_Not17(j,2)) =
samples_Not17(j,3)/words_per_doc;
    end
end

% conbine class 17 and not 17
train_word_frequency = [train_word_frequency17; train_word_frequencyNot17];
train_label_binary = [train_label17; train_labelNot17];


%%%%% Test %%%%

% test - documets id that are part of class 17
 test_class17 = find(test_label == 17);
 test_classNot17 = setdiff(1:length(test_label), test_class17);

% test - labels for class 17
test_label17 = test_label(test_class17);
test_labelNot17 = test_label(test_classNot17);
```

```
% label the classes that are not 17 equal to class 1
test_labelNot17(:,1) = 1;


% test - calculate the word frequency for class 17
test_word_frequency17 = zeros(length(test_class17), length(vocabulary));
for i = 1:length(test_class17)
    b = find(test_data(:,1) == test_class17(i));
    samples = test_data(b,:);

    % total number of words in the document
    words_per_doc = sum(samples(:,3),1);
    for j = 1: size(samples,1)
        test_word_frequency17(i,samples(j,2)) = samples(j,3)/words_per_doc;
    end
end


% test - calculate the word frequency for class 1
test_word_frequencyNot17 = zeros(length(test_classNot17),
length(vocabulary));
for i = 1:length(test_classNot17)
    d = find(test_data(:,1) == test_classNot17(i));
    samples_Not17 = test_data(d,:);

    % total number of words in the document
    words_per_doc = sum(samples_Not17(:,3),1);
    for j = 1: size(samples_Not17,1)
        test_word_frequencyNot17(i, samples_Not17(j,2)) =
samples_Not17(j,3)/words_per_doc;
    end
end

% conbine class 17 and not 17
test_word_frequency = [test_word_frequency17; test_word_frequencyNot17];
test_label_binary = [test_label17; test_labelNot17];



vocab_index = 1:length(vocabulary);
train_index_diff = setdiff(vocab_index, remove_list);

% remove stoplist words from train_data
train_word_frequency = train_word_frequency(:,train_index_diff);
clear train_word_frequency17 train_word_frequencyNot17 test_word_frequency17
test_word_frequencyNot17 train_data test_data train_label test_label

% remove stoplist words from test_data
test_word_frequency = test_word_frequency(:,train_index_diff);

clear remove_list samples samples_Not17 stoplist test_label17 test_labelNot17
train_index_diff train_label17 train_labelNot17
clear vocabulary vocab_index training_class17 training_classNot17
test_class17 test_classNot17
```

```
s = RandStream('mt19937ar','Seed',1);

random_train = randperm(s,size(train_word_frequency,1));

split1 = train_word_frequency(random_train(1:2253),:);
split2 = train_word_frequency(random_train(2254:4507),:);
split3 = train_word_frequency(random_train(4508:6761),:);
split4 = train_word_frequency(random_train(6762:9015),:);
split5 = train_word_frequency(random_train(9016:11269),:);


label_split1 = train_label_binary(random_train(1:2253),:);
label_split2 = train_label_binary(random_train(2254:4507),:);
label_split3 = train_label_binary(random_train(4508:6761),:);
label_split4 = train_label_binary(random_train(6762:9015),:);
label_split5 = train_label_binary(random_train(9016:11269),:);

C = [2^-5, 2^-4, 2^-3, 2^-2, 2^-1, 2^0, 2^1, 2^2, 2^3, 2^4, 2^5, 2^6, 2^7,
2^8, 2^9, 2^10, 2^11, 2^12, 2^13, 2^14, 2^15];


for i = 1:length(C)
    a = tic
    disp(i);
    % test on split 5
    train1 = [split1; split2; split3; split4];
    label1 = [label_split1; label_split2; label_split3; label_split4];
    SVMStruct1 = svmtrain(train1, label1,'boxconstraint',
C(i),'kernelcachelimit', 500000, 'autoscale', 'false');
    Group1 = svmclassify(SVMStruct1,split5);
    C1 = confusionmat(label_split5,Group1);
    CCR1 = sum(diag(C1))/ sum(sum(C1));

    C1 = C1';
    % calculate precision TP/np_hat
    P1 = C1(1,1)/ (C1(1,1) + C1(1,2));

    % calculate recal TP/np
    R1 = C1(1,1)/ (C1(1,1) + C1(2,1));
    % calculate F-score
    F1 = 2*(P1*R1)/(P1 + R1);

    clear train1 label1 SVMStruct1 Group1 C1

    % test on split 4
    train2 = [split1; split2; split3; split5];
    label2 = [label_split1;label_split2; label_split3; label_split5];
    SVMStruct2 = svmtrain(train2, label2,'boxconstraint',C(i),
'kernelcachelimit', 500000,'autoscale', 'false');
    Group2 = svmclassify(SVMStruct2, split4);
    C2 = confusionmat(label_split4,Group2);
    CCR2 = sum(diag(C2))/ sum(sum(C2));

    C2 = C2';
    % calculate precision TP/np_hat and recall
    P2 = C2(1,1)/ (C2(1,1) + C2(1,2));
    R2 = C2(1,1)/ (C2(1,1) + C2(2,1));
    F2 = 2*(P2*R2)/(P2 + R2);

    clear train2 label2 SVMStruct2 Group2 C2

    % test on split 3
    train3 = [split1; split2; split4; split5];
    label3 = [label_split1;label_split2; label_split4; label_split5];
    SVMStruct3 = svmtrain(train3, label3,'boxconstraint',
C(i),'kernelcachelimit', 500000, 'autoscale', 'false');
    Group3 = svmclassify(SVMStruct3, split3);
    C3 = confusionmat(label_split3,Group3);
    CCR3 = sum(diag(C3))/ sum(sum(C3));

    C3 = C3';
     % calculate precision TP/np_hat
    P3 = C3(1,1)/ (C3(1,1) + C3(1,2));
    R3 = C3(1,1)/ (C3(1,1) + C3(2,1));
    F3 = 2*(P3*R3)/(P3 + R3);

    clear train3 label3 SVMStruct3 Group3 C3

    % test on split 2
    train4 = [split1; split3; split4; split5];
    label4 = [label_split1; label_split3; label_split4; label_split5];
    SVMStruct4 = svmtrain(train4, label4,'boxconstraint',
C(i),'kernelcachelimit', 500000, 'autoscale', 'false');
    Group4 = svmclassify(SVMStruct4,split2);
    C4 = confusionmat(label_split2,Group4);
    CCR4 = sum(diag(C4))/ sum(sum(C4));

    C4 = C4';
    % calculate precision TP/np_hat
    P4 = C4(1,1)/ (C4(1,1) + C4(1,2));
    R4 = C4(1,1)/ (C4(1,1) + C4(2,1));
    F4 = 2*(P4*R4)/(P4 + R4);

    clear train4 label4 SVMStruct4 Group4 C4

    % test on split 1
    train5 = [split2; split3; split4; split5];
    label5 = [label_split2;label_split3; label_split4; label_split5];
    SVMStruct5 = svmtrain(train5, label5,'boxconstraint',
C(i),'kernelcachelimit', 500000, 'autoscale', 'false');
    Group5 = svmclassify(SVMStruct5, split1);
    C5 = confusionmat(label_split1,Group5);
    CCR5 = sum(diag(C5))/ sum(sum(C5));

    C5 = C5';
     % calculate precision TP/np_hat
    P5 = C5(1,1)/ (C5(1,1) + C5(1,2));
    R5 = C5(1,1)/ (C5(1,1) + C5(2,1));
    F5 = 2*(P5*R5)/(P5 + R5);

    clear train5 label5  SVMStruct5 Group5 C5
    CCR(i) = (CCR1+CCR2+CCR3+CCR4+CCR5)/5
    R(i) = (P1 + P2 + P3 + P4 + P5)/5
    P(i) = (R1 + R2 + R3 + R4 + R5)/5
    F(i) = (F1 + F2 + F3 + F4 + F5)/5
    toc(a)
end

figure(1)
plot(log2(C), CCR);
title('CV-CCRs as a function of C');
xlabel('log2 (C values)');
ylabel('CV-CCRs');

figure(2)
plot(log2(C), P,'r');
title('Precision, recall, and F-score as a function of C');
xlabel('log2 (C values)');
ylabel('Precision, recall, and F-score');
hold on;
plot(log2(C), R,'b');
hold on;
plot(log2(C), F, 'g');
legend('precision', 'recall', 'F-score');
hold off;


[CCR_max, CCR_max_index] = max(CCR);
C_max = C(CCR_max_index);

SVMStruct_all = svmtrain(train_word_frequency,
train_label_binary,'boxconstraint', C_max,'kernelcachelimit', 500000,
'autoscale', 'false');
Group_all = svmclassify(SVMStruct_all, test_word_frequency);

confusion_matrix = confusionmat(test_label_binary,Group_all)
CCR_all = sum(diag(confusion_matrix))/ sum(sum(confusion_matrix));

[F_max, F_max_index] = max(F);
C_F_max = C(F_max_index);

% confusion matrix for F
SVMStruct_F = svmtrain(train_word_frequency,
train_label_binary,'boxconstraint', C_F_max,'kernelcachelimit', 500000,
'autoscale', 'false');
Group_F = svmclassify(SVMStruct_F, test_word_frequency);

confusion_matrix_F = confusionmat(test_label_binary,Group_F)
CCR_F = sum(diag(confusion_matrix_F))/ sum(sum(confusion_matrix_F));

% confusion matrix for R
[R_max, R_max_index] = max(R);
C_R_max = C(R_max_index);

SVMStruct_R = svmtrain(train_word_frequency,
train_label_binary,'boxconstraint', C_R_max,'kernelcachelimit', 500000,
'autoscale', 'false');
Group_R = svmclassify(SVMStruct_R, test_word_frequency);

confusion_matrix_R = confusionmat(test_label_binary,Group_R)
CCR_R = sum(diag(confusion_matrix_R))/ sum(sum(confusion_matrix_R));

save('hw6_1cd_results', 'P', 'R', 'F', 'CCR','C', 'confusion_matrix',
'CCR_all', 'confusion_matrix_F', 'confusion_matrix_R', 'C_R_max', 'C_F_max')
```

```
% Assignemnt 6_1e
% One-vs-one (OVO) multiclass classification with linear kernel
% in order to mitigate the unbalenced dataset.
% Calculate overall CCR
% Calculate test and training time

clear; close all; clc;

% train and test datasets
train_data = load('train.data');
train_label = load('train.label');

vocabulary = textread('vocabulary.txt', '%s');
stoplist = textread('stoplist.txt', '%s');

test_data = load('test.data');
test_label = load('test.label');

% find the index for the stoplist words in vocabulary
stoplist = unique(stoplist);
p = 0;
for i =1: size(stoplist, 1);
    index = strmatch(stoplist(i), vocabulary, 'exact');
    if ~isempty(index)
```

```
        p = p + 1;
        remove_list(p,1) = index;
    end
end

% set the words in stoplist to zero frequency
for i = 1: length(remove_list)
    train_loc = find(train_data(:,2) == remove_list(i));
    train_data(train_loc, 3) = 0;

    test_loc = find(test_data(:,2) == remove_list(i));
    test_data(test_loc, 3) = 0;
end

% find the word frequency of the training set
train_word_frequency = zeros(length(train_label), length(vocabulary));
for i = 1:length(train_label)
    disp(i)
    b = find(train_data(:,1) == i);
    samples = train_data(b,:);

    % total number of words in the document
    words_per_doc = sum(samples(:,3),1);
    for j = 1: size(samples,1)
        train_word_frequency(i,samples(j,2)) = samples(j,3)/words_per_doc;
    end
end
train_word_frequency = sparse(train_word_frequency);
clear train_data stoplist train_loc vocabulary



% find the word frequency ov the test set
test_word_frequency = zeros(length(test_label), 61188);
for i = 1:length(test_label)
    disp(i)
    b = find(test_data(:,1) == i);
    samples = test_data(b,:);

    % total number of words in the document
    words_per_doc = sum(samples(:,3),1);
    for j = 1: size(samples,1)
        test_word_frequency(i,samples(j,2)) = samples(j,3)/words_per_doc;
    end
end
test_word_frequency = sparse(test_word_frequency);
clear test_data b


vocab_index = 1:61188;
vocab_index_diff = setdiff(vocab_index, remove_list);

% remove stoplist words from train_data
train_word_frequency = train_word_frequency(:,vocab_index_diff);
```

```
% remove stoplist words from test_data
test_word_frequency = test_word_frequency(:,vocab_index_diff);

clear vocabulary stoplist

train_time = 0;
test_time = 0;
t = 0;
for i = 1:20

    disp(i)
    % class i training data and labels
    class_a_index = find(train_label == i);
    class_a = train_word_frequency(class_a_index,:);
    class_label_a = train_label(class_a_index);

    for j = (i + 1):20
        t = t + 1;
        disp('t:')
        disp(t)
        % class j training data and labels
        class_b_index = find(train_label == j);
        class_b = train_word_frequency(class_b_index,:);
        class_label_b = train_label(class_b_index);

        train_data_binary = [class_a; class_b];
        train_label_binary = [class_label_a; class_label_b];

        tic
        SVMStruct = svmtrain(train_data_binary,
train_label_binary,'kernelcachelimit', 1000000, 'autoscale', 'false');
        train_time = train_time + toc;

        tic
        Group(:,t) = svmclassify(SVMStruct, test_word_frequency);
        test_time = test_time + toc;

    end

end



label = mode(Group,2);
confusion_matrix = confusionmat(label,test_label)
CCR = sum(diag(confusion_matrix))/ sum(sum(confusion_matrix))

save('hw6_1e_results.mat', 'confusion_matrix', 'CCR')
```

```
% Assignemnt 6_1f
% One-vs-one (OVO) multiclass classification with rbf kernel
% in order to mitigate the unbalenced dataset.
% Calculate overall CCR
% Calculate test and training time
clear; close all; clc;

% train and test datasets
train_data = load('train.data');
train_label = load('train.label');

vocabulary = textread('vocabulary.txt', '%s');
stoplist = textread('stoplist.txt', '%s');

test_data = load('test.data');
test_label = load('test.label');

% find the index for the stoplist words in vocabulary
stoplist = unique(stoplist);
p = 0;
for i =1: size(stoplist, 1);
    index = strmatch(stoplist(i), vocabulary, 'exact');
    if ~isempty(index)
        p = p + 1;
        remove_list(p,1) = index;
    end
end

% set the words in stoplist to zero frequency
```

```
for i = 1: length(remove_list)
    train_loc = find(train_data(:,2) == remove_list(i));
    train_data(train_loc, 3) = 0;

    test_loc = find(test_data(:,2) == remove_list(i));
    test_data(test_loc, 3) = 0;
end

% find the word frequency of the training set
train_word_frequency = zeros(length(train_label), length(vocabulary));
for i = 1:length(train_label)
    disp(i)
    b = find(train_data(:,1) == i);
    samples = train_data(b,:);

    % total number of words in the document
    words_per_doc = sum(samples(:,3),1);
    for j = 1: size(samples,1)
        train_word_frequency(i,samples(j,2)) = samples(j,3)/words_per_doc;
    end
end
train_word_frequency = sparse(train_word_frequency);
clear train_data stoplist train_loc vocabulary



% find the word frequency ov the test set
test_word_frequency = zeros(length(test_label), 61188);
for i = 1:length(test_label)
    disp(i)
    b = find(test_data(:,1) == i);
    samples = test_data(b,:);

    % total number of words in the document
    words_per_doc = sum(samples(:,3),1);
    for j = 1: size(samples,1)
        test_word_frequency(i,samples(j,2)) = samples(j,3)/words_per_doc;
    end
end
test_word_frequency = sparse(test_word_frequency);
clear test_data b


vocab_index = 1:61188;
vocab_index_diff = setdiff(vocab_index, remove_list);

% remove stoplist words from train_data
train_word_frequency = train_word_frequency(:,vocab_index_diff);


% remove stoplist words from test_data
test_word_frequency = test_word_frequency(:,vocab_index_diff);

clear vocabulary stoplist
```

```
train_time = 0;
test_time = 0;
t = 0;
for i = 1:20

    disp(i)
    % class i training data and labels
    class_a_index = find(train_label == i);
    class_a = train_word_frequency(class_a_index,:);
    class_label_a = train_label(class_a_index);

    for j = (i + 1):20
        t = t + 1;
        disp('t:')
        disp(t)
        % class j training data and labels
        class_b_index = find(train_label == j);
        class_b = train_word_frequency(class_b_index,:);
        class_label_b = train_label(class_b_index);

        train_data_binary = [class_a; class_b];
        train_label_binary = [class_label_a; class_label_b];

        tic
        SVMStruct = svmtrain(train_data_binary,
train_label_binary,'kernel_function','rbf','kernelcachelimit', 1000000,
'autoscale', 'false');
        train_time = train_time + toc;

        tic
        Group(:,t) = svmclassify(SVMStruct, test_word_frequency);
        test_time = test_time + toc;

    end

end


label = mode(Group,2);
confusion_matrix = confusionmat(label,test_label)
CCR = sum(diag(confusion_matrix))/ sum(sum(confusion_matrix))

save('hw6_1f_results.mat', 'confusion_matrix', 'CCR', 'train_time',
'test_time')
```