**EX4 – SVM Spam vs Ham**

%% ==================== Part 1: Email Preprocessing ====================

fprintf('\nPreprocessing sample email (emailSample1.txt)\n');

file\_contents = readFile('emailSample1.txt');

word\_indices = processEmail(file\_contents);

fprintf('Word Indices: \n');

fprintf(' %d', word\_indices);

fprintf('\n\n');

fprintf('Program paused. Press enter to continue.\n');

pause;

%% ==================== Part 2: Feature Extraction ====================

fprintf('\nExtracting features from sample email (emailSample1.txt)\n');

file\_contents = readFile('emailSample1.txt');

word\_indices = processEmail(file\_contents);

features = emailFeatures(word\_indices);

fprintf('Length of feature vector: %d\n', length(features));

fprintf('Number of non-zero entries: %d\n', sum(features > 0));

%% =========== Part 3: Train Linear SVM for Spam Classification ========

load('spamTrain.mat'); %Pre-processed Training Emails in Matrix Representation. Procedure same as implemented in Part 1 and 2

fprintf('\nTraining Linear SVM (Spam Classification)\n')

fprintf('(this may take 1 to 2 minutes) ...\n')

C = 0.1;

model = svmtrain(y, X, '-s 0 -t 0 -c 0.1');

[p,accuracy,dummy] = svmpredict(y, X, model);

fprintf('Training Accuracy: %f\n', accuracy);

%% =================== Part 3: Test Spam Classification ================

load('spamTest.mat'); % Pre-processed Testing Emails in Matrix Representation. Procedure same as implemented in Part 1 and 2

fprintf('\nEvaluating the trained Linear SVM on a test set ...\n')

[p,accuracy,dummy] = svmpredict(ytest, Xtest, model);

fprintf('Test Accuracy: %f\n', accuracy);

pause;

%% =================== Part 4: Try Your Own Emails =====================

filename = 'emailSample1.txt';

file\_contents = readFile(filename);

word\_indices = processEmail(file\_contents);

x = emailFeatures(word\_indices);

y = rand();

p = svmpredict(y,x',model);

fprintf('Spam Classification: %d\n', p);

fprintf('(1 indicates spam, 0 indicates not spam)\n\n');

function word\_indices = **processEmail**(email\_contents)

vocabList = getVocabList();%Words having >100 occurrence in Training set

word\_indices = [];

email\_contents = lower(email\_contents); %Lower Case

email\_contents = regexprep(email\_contents, '<[^<>]+>', ' '); %HTML

email\_contents = regexprep(email\_contents, '[0-9]+', 'number'); %Number

email\_contents = regexprep(email\_contents, ...

'(http|https)://[^\s]\*', 'httpaddr'); %URL

email\_contents = regexprep(email\_contents, '[^\s]+@[^\s]+', 'emailaddr'); %Email Address

email\_contents = regexprep(email\_contents, '[$]+', 'dollar');

fprintf('\n==== Processed Email ====\n\n');

l = 0;

while ~isempty(email\_contents)

[str, email\_contents] = ...

strtok(email\_contents, ...

[' @$/#.-:&\*+=[]?!(){},''">\_<;%' char(10) char(13)]);

str = regexprep(str, '[^a-zA-Z0-9]', '');

try str = porterStemmer(strtrim(str)); %Porter Stemmer -> Root words

catch str = ''; continue;

end;

if length(str) < 1

continue;

end

idx = strmatch(str,vocabList,'exact');

if size(idx,1)~=0

word\_indices = [word\_indices ; idx];

end

if (l + length(str) + 1) > 78

fprintf('\n');

l = 0;

end

fprintf('%s ', str);

l = l + length(str) + 1;

end

end

function x = **emailFeatures**(word\_indices)

%EMAILFEATURES takes in a word\_indices vector and produces a feature vector

%from the word indices

n = 1899;

x = zeros(n, 1);

x(word\_indices) = 1;

end

**EX4 – Parameter Selection SVM**

%% ========== Training SVM with RBF Kernel (Dataset 3) ==========

% Finding Optimal parameter values for C and sigma

load('ex6data3.mat');

% Try different SVM Parameters here

[C, sigma] = dataset3Params(X, y, Xval, yval);

disp(C);

disp(sigma);

function [C, sigma] = dataset3Params(X, y, Xval, yval)

%EX6PARAMS returns your choice of C and sigma for Part 3 of the exercise

%where you select the optimal (C, sigma) learning parameters to use for SVM

%with RBF kernel

C = 1; sigma = 0.3; min\_error = 999; min\_c = 0; min\_sigma = 0;

C\_List = [0.01, 0.03, 0.1, 0.3, 1, 3 , 10, 30];

sigma\_List = [0.01, 0.03, 0.1, 0.3, 1, 3 , 10, 30];

%C\_List = [0.1,0.13,0.16,0.19,0.2,0.22,0.25];

%sigma\_List = [0.06,0.063,0.066,0.07,0.073,0.076,0.08];

for i=1:length(C\_List)

for j=1:length(sigma\_List)

Options = ['-s 0 -t 3 -g', ' ', num2str(sigma\_List(i)),' -c ',num2str(C\_List(i))];

model= svmtrain(y, X, Options);

predictions = svmpredict(yval, Xval, model);

err = mean(double(predictions ~= yval));

if err<min\_error

min\_error = err;

min\_c = C\_List(i);

min\_sigma = sigma\_List(j);

end

end

end

C = min\_c;

sigma = min\_sigma;

end

function sim = **gaussianKernel**(x1, x2, sigma)

%RBFKERNEL returns a radial basis function kernel between x1 and x2

x1 = x1(:); x2 = x2(:);

sim = exp(-(x1-x2)'\*(x1-x2)/(2\*sigma.^2));

end