```
clear
clc
close all

load('Final_Math285_WS.mat')
%% Load the training data and test data i

load data_batch_1

train_data_1 = data;
train_labels_1 = labels;

load data_batch_2

train_data_2 = data;
train_labels_2 = labels;

load data_batch_3

train_data_3 = data;
train_labels_3 = labels;

load data_batch_4

train_data_4 = data;
train_labels_4 = labels;

load data_batch_5

train_data_5 = data;
train_labels_5 = labels;

load test_batch

TestImages = data;
TestLabels = labels;


load data_batch_1.mat
size(data)

Objects = find(train_labels_1 == 1 | train_labels_1 == 9);
Data = data(Objects,:);

for i = 40:54;
    R=Data(i,1:1024);
    G=Data(i,1025:2048);
    B=Data(i,2049:3072);
    A(:,:,1)=reshape(R,32,32);
    A(:,:,2)=reshape(G,32,32);
    A(:,:,3)=reshape(B,32,32);
    subplot(3,5,i-39)
    imshow(A)
    camroll(270)
end

%% Concatenate and convert the datasets

% plain kNN and try to find out the best K and test error
```

```
TrainImages = cat(1, train_data_1, train_data_2, train_data_3, train_data_4, train_data_5);

TrainLabels = cat(1, train_labels_1, train_labels_2, train_labels_3, train_labels_4,
train_labels_5);

TrainImages = double(TrainImages);
save TrainImages TrainImages


TrainLabels = double(TrainLabels);
save TrainLabels TrainLabels

TestImages = double(TestImages);
save TestImages TestImages

TestLabels = double(TestLabels);
save TestLabels TestLabels


kstart = 1;
kmax = 10;
testError_plain_kNN = zeros(kmax,1);

tic
for k = 1:kmax
    k
    mdlC = fitcknn(TrainImages, TrainLabels,'NumNeighbors', k);
    predC = predict(mdlC, TestImages);
    testError_plain_kNN(k) = sum(TestLabels~=predC)/numel(TestLabels);
end
runtime_plain_kNN = toc/60; saveFinal_Math285_WS.mat;

% Final_plot_1_a: plain kNN:

figure;
hold on;
plot(1:10, testError_plain_kNN,'-o', 'markersize', 10, 'linewidth', 2)
set(gca,'XTickLabel',1:10)
a = [cellstr(num2str(get(gca,'ytick')'*100))];
pct = char(ones(size(a,1),1)*'%');
new_yticks = [char(a),pct];
set(gca,'yticklabel',new_yticks)
title({'Plain kNN Test Error:','k = 1:10'}, 'fontsize', 15)
xlabel('k', 'fontsize', 14)
ylabel('Errors (%)', 'fontsize', 14)
[~,I] = min(testError_plain_kNN(:));
[I_row_plain_kNN, I_col_plain_kNN] = ind2sub(size(testError_plain_kNN),I);
xmin_plain_kNN = I_row_plain_kNN;
ymin_plain_kNN = min(testError_plain_kNN(:));
strmin = ['Minimum Test Error= ',num2str(ymin_plain_kNN*100),'%'];
plot(xmin_plain_kNN, ymin_plain_kNN,'^', 'MarkerSize', 12,'Color', 'red');
text(xmin_plain_kNN,ymin_plain_kNN,strmin,'Color','red','FontSize',↙
10,'VerticalAlignment','top', 'HorizontalAlignment', 'left');
saveas(gcf, 'Final_Plot_1_a','pdf');

%% plain local kMeans w/o projection to find out the optimal k and test error
IntClass = 0;
EndClass = 9;
NumClasses = 10;
```

```
ClassCenter = zeros(size(TestImages,1), NumClasses, size(TestImages, 2));
Dist = zeros(size(TestImages,1),length(unique(TrainLabels)));
TestError_LocalKmeans = zeros(10,1);


tic
for k= kstart:kmax
    k
    for c= IntClass:EndClass
        c
        TrainImages_Class = find(TrainLabels == c);
        LocalKnn = knnsearch(TrainImages(TrainImages_Class,:), TestImages,'k', k);
        LocalKnn = TrainImages_Class(LocalKnn);
        TrainImages_Reduced = reshape(TrainImages(LocalKnn,:)', size(TestImages,2), size
(TestImages,1), k);
        ClassCenter(:,c+1,:)=squeeze(mean(TrainImages_Reduced, 3))';
    end
    for t=kstart:size(TestImages,1)
        t
         localTest = knnsearch (squeeze(ClassCenter(t,:,:)), TestImages(t,:))-1;
         TestError_LocalKmeans(k)= (TestError_LocalKmeans(k)+(TestLabels(t)~= localTest));

    end
end
runtime_plain_local_kNN = toc/60; saveFinal_Math285_WS.mat;
TestError_LocalKmeans = TestError_LocalKmeans/10000;

figure;
hold on;
plot(1:10, TestError_LocalKmeans,'-o', 'markersize', 10, 'linewidth', 2)
set(gca,'XTickLabel',1:10)
a = [cellstr(num2str(get(gca,'ytick')'*100))];
pct = char(ones(size(a,1),1)*'%');
new_yticks = [char(a),pct];
set(gca,'yticklabel',new_yticks)
title({'Local kMeans Test Error:','k = 1:10'}, 'fontsize', 15)
xlabel('k', 'fontsize', 14)
ylabel('Errors (%)', 'fontsize', 14)
[~,I] = min(TestError_LocalKmeans(:));
[I_row_LocalKmeans, I_col_LocalKmeans] = ind2sub(size(TestError_LocalKmeans),I);
xmin_LocalKmeans = I_row_LocalKmeans;
ymin_LocalKmeans = min(TestError_LocalKmeans(:));
strmin = ['Minimum Test Error= ',num2str(ymin_LocalKmeans*100),'%'];
plot(xmin_LocalKmeans, ymin_LocalKmeans,'^', 'MarkerSize', 12,'Color', 'red');
text(xmin_LocalKmeans,ymin_LocalKmeans,strmin,'Color','red','FontSize',
10,'VerticalAlignment','top', 'HorizontalAlignment', 'right');
saveas(gcf, 'Final_Plot_2_a','pdf');



%% PCA + LDA


X = TrainImages;


m = mean(X,1); %
X_tilde = X - repmat(m, size(X,1), 1);% centered data
[U,S,V] = svd(X_tilde,'econ');
```

```
s = diag(S); % vector of singular values

figure;
plot(s, '.', 'markersize', 12);
title('singular values', 'fontsize', 14)
grid on

cum_s = cumsum(s.^2); % s(1)^2, s(1)^2+s(2)^1, s(1)^2+s(2)^1+s(3)^2, etc
cum_s = cum_s/cum_s(end);

figure;
plot(cum_s, '.', 'markersize', 12);
title('explained variance', 'fontsize', 14)
grid on

s = find(cum_s>0.95, 1, 'first');

s = find(cum_s>0.975, 1, 'first');% reduced dimension
Y = X_tilde*V(:,1:144);
Y = Y';
figure; plot3(Y(:,1), Y(:,2), Y(:,3),'.')

index = 1
X = reshape(Y(:, index), 13, 13, []);% 28 x 28 x 100
DisplayImageCollection(X);


% test error for our LDA

s2 = (55:382);

testErrorLDA = zeros(length(s2),kstart);

tic
for j = 1:length(s2)
    j
    pca_train = TrainImages_Tilde*V(:,1:s2(j));
    pca_test = TestImages_Tilde*V(:,1:s2(j));
    lda_model = fitcdiscr(pca_train, TrainLabels,'DiscrimType', 'linear');
    pred = predict(lda_model, pca_test);
    testErrorLDA(j) = sum(TestLabels~=pred)/numel(TestLabels);
end
run_time_PCA_LDA_test_error = toc/60; saveFinal_Math285_WS.mat;

figure;
hold on;
plot(s2, testErrorLDA, '-o', 'markersize', 5, 'linewidth', 2)
a = [cellstr(num2str(get(gca,'ytick')'*100))];
pct = char(ones(size(a,1),1)*'%');
new_yticks = [char(a),pct];
set(gca,'yticklabel',new_yticks)
title({'LDA Test Error:', ['s=',num2str(s2(1)),':',num2str(s2(end))]}, 'fontsize', 30)
xlabel('s value', 'fontsize', 14)
ylabel('Errors (%)', 'fontsize', 14)
indexmin_lda = find(min(testErrorLDA)== testErrorLDA);
xmin_pca_lda = s2(indexmin_lda(1));
ymin_pca_lda = testErrorLDA(indexmin_lda(1));
strmin = ['Minimum Test Error= ',num2str(ymin_pca_lda *100),'%'];
```

```
plot(xmin_pca_lda, ymin_pca_lda ,'o','Color', 'red', 'MarkerSize', 10);
text(xmin_pca_lda,ymin_pca_lda ,strmin,'Color','red','FontSize',↙
14,'VerticalAlignment','bottom','HorizontalAlignment','right');
saveas(gcf, 'Final_Plot_3_a','pdf');

% pca 367

save Math285_Chen_WS.mat

%% PCA+QDA

testErrorQDA = zeros(length(s2),kstart);

tic
for j = 1:length(s2)
    j
    pca_train = TrainImages_Tilde*V(:,1:s2(j));
    pca_test = TestImages_Tilde*V(:,1:s2(j));
    qda_model = fitcdiscr(pca_train, TrainLabels,'DiscrimType', 'quadratic');
    pred = predict(qda_model, pca_test);
    testErrorQDA(j) = sum(TestLabels~=pred)/numel(TestLabels);
end
run_time_PCA_QDA_test_error = toc/60; saveFinal_Math285_WS.mat;


figure;
hold on;
plot(s2, testErrorQDA, '-o', 'markersize', 5, 'linewidth', 2)
title({'PCA + QDA Test Error:', ['s=',num2str(s2(1)),':',num2str(s2(end))]}, 'fontsize', 15)
xlabel('s value', 'fontsize', 14)
ylabel('Errors (%)', 'fontsize', 14)
a = [cellstr(num2str(get(gca,'ytick')'*100))];
pct = char(ones(size(a,1),1)*'%');
new_yticks = [char(a),pct];
set(gca,'yticklabel',new_yticks)
indexmin_qda = find(min(testErrorQDA)== testErrorQDA);
xmin_pca_qda = s2(indexmin_qda);
ymin_pca_qda = testErrorQDA(indexmin_qda);
strmin = ['Minimum Test Error= ',num2str(ymin_pca_qda*100),'%'];
plot(xmin_pca_qda, ymin_pca_qda,'o','Color', 'red', 'MarkerSize', 10);
text(xmin_pca_qda,ymin_pca_qda,strmin,'Color','red','FontSize',14,'VerticalAlignment','bottom',↙
'HorizontalAlignment','right');
saveas(gcf, 'Final_Plot_4_a','pdf');

pca_train = TrainImages_Tilde*V(:,1:s2(indexmin_qda));
pca_test = TestImages_Tilde*V(:,1:s2(indexmin_qda));
qda_model = fitcdiscr(pca_train, TrainLabels,'DiscrimType', 'quadratic');
pred_pca_qda_s365 = predict(qda_model, pca_test);

load('batches.meta.mat', 'label_names')
Conf_Mat = confusionmat(TestLabels,pred_pca_qda_s365);
mat = Conf_Mat/1000;          %# A 5-by-5 matrix of random values from 0 to 1
imagesc(mat);            %# Create a colored plot of the matrix values
colormap(flipud(pink)); %# Change the colormap to gray (so higher values are
                        %#   black and lower values are white)

textStrings = num2str(mat(:),'%0.2f');  %# Create strings from the matrix values
textStrings = strtrim(cellstr(textStrings)); %# Remove any space padding
[x,y] = meshgrid(1:10);  %# Create x and y coordinates for the strings
```

```
hStrings = text(x(:),y(:),textStrings(:),..      %# Plot the strings
                'HorizontalAlignment','center');
midValue = mean(get(gca,'CLim'));  %# Get the middle value of the color range
textColors = repmat(mat(:) > midValue,1,3); %# Choose white or black for the
                                            %#   text color of the strings so
                                            %#   they can be easily seen over
                                            %#   the background color
set(hStrings,{'Color'},num2cell(textColors,2)); %# Change the text colors
title({'Confusion Matrix for PCA(s=365) + QDA}, 'fontsize', 15)
set(gca,'XTick',1:10,...                         %# Change the axes tick marks
        'XTickLabel',label_names,...  %#   and tick labels
        'YTick',1:10,...
        'YTickLabel',label_names,...
        'TickLength',[0 0]);
set(gca,'XTickLabel','')
set(gca(), 'XTickLabel', label_names);
rotateXLabels(gca(), 45 );
saveas(gcf, 'Final_Plot_4_b','pdf');


%% PCA + NB

testErrorNB = zeros(length(s2),kstart);

tic
for j = 1:length(s2)
j
pca_train = TrainImages_Tilde*V(:,1:s2(j));
pca_test = TestImages_Tilde*V(:,1:s2(j));
NB_model = fitcnb(pca_train, TrainLabels,'Distribution', 'normal');
pred = predict(NB_model, pca_test);
testErrorNB(j) = sum(TestLabels~=pred)/numel(TestLabels);
end
run_time_PCA_NB_test_error = toc/60;saveFinal_Math285_WS.mat;



figure;
hold on;
plot(s2, testErrorNB, '-o', 'markersize', 5, 'linewidth', 2)
title({'PCA + Naive Bayes Test Error:', ['s=',num2str(s2(1)),':',num2str(s2(end))]},↙
'fontsize', 15)
xlabel('s value', 'fontsize', 14)
ylabel('Errors (%)', 'fontsize', 14)
a = [cellstr(num2str(get(gca,'ytick')'*100))];
pct = char(ones(size(a,1),1)*'%');
new_yticks = [char(a),pct];
set(gca,'yticklabel',new_yticks)
indexmin_cos = find(min(testErrorNB)== testErrorNB);
xmin = s2(indexmin_cos);
ymin = testErrorNB(indexmin_cos);
strmin = ['Minimum Test Error= ',num2str(ymin)];
plot(xmin, ymin, 'o','Color', 'red', 'MarkerSize', 10);
text(xmin,ymin,strmin,'Color','red','FontSize',14,'VerticalAlignment','bottom');
saveas(gcf, 'Final_Plot_5_a','pdf');


%% One vs. one LR
```

```
% PCA 367 training and test images
center = mean(TrainImages,1);
TrainImages_Tilde = TrainImages - repmat(center, size(TrainImages,1), 1);% centered training↙
data
TestImages_Tilde = TestImages - repmat(center, size(TestImages,1),1);% centered test data
[U,S,V] = svd(TrainImages_Tilde,'econ');

pca_train = TrainImages_Tilde*V(:,1:367);
pca_test = TestImages_Tilde*V(:,1:367);

Labels_num = 0:9;
pair = nchoosek(Labels_num, 2);
p = zeros(size(pair,1), size(TestImages,1),length(s2));

tic
for k = 1:length(s2)
    k
    pca_train = TrainImages_Tilde*V(:,1:s2(k));
    pca_test = TestImages_Tilde*V(:,1:s2(k));
    for i = 1:size(pair,1)
    i
        Train_Class = find(TrainLabels == pair(i,1) | TrainLabels == pair(i,2) );
        TrainImages_Sel = pca_train(Train_Class,:);
        TrainLabels_Sel = TrainLabels(Train_Class,:);

        glm = fitglm(TrainImages_Sel, categorical(TrainLabels_Sel),'linear', 'distr',↙
'binomial');
        p(i,:,k) = predict(glm, pca_test);
        for j = 1:size(p,2)
        j
        if p(i,j,k) > 1/2
        p(i,j,k) = pair(i,2);
        else
        p(i,j,k) = pair(i,1);
        end
    end
    end
end
run_time_OVO_LR_test_error = toc/60; saveFinal_Math285_WS.mat;


Mode_p = zeros(size(TestImages,1), length(s2));


for i = 1:length(s2)
     Mode_p(:,i) = mode(p(:,:,i))';
end

testError_one_vs_one = zeros(length(s2),1);

for i = 1:length(s2)
    testError_one_vs_one(i) = sum(TestLabels~=Mode_p(:,i))/numel(TestLabels);
end

save Final_Math285_WS.mat;

figure;
hold on;
plot(s2, testError_one_vs_one,'-o', 'markersize', 5, 'linewidth', 2)
```

```
title({'One vs. One', 'Logistic Regression Test Error:', 's=55 to 382'}, 'fontsize', 30)
xlabel('s value', 'fontsize', 24)
ylabel('Errors', 'fontsize', 24)
% set(gca,'XTick',50:10:154)
indexmin_cos = find(min(testError_one_vs_one)== testError_one_vs_one);
xmin = s2(indexmin_cos);
ymin = testError_one_vs_one(indexmin_cos);
strmin = ['Minimum Test Error= ',num2str(ymin)];
plot(xmin, ymin, 'o','Color', 'red', 'MarkerSize', 10);
text(xmin,ymin,strmin,'Color','red','FontSize',14,'VerticalAlignment','bottom');
saveas(gcf, 'Final_Plot_6_a','pdf');

% xmin = 189; indexmin_cos = 185

conmat_one_vs_one = confusionmat(TestLabels, Mode_p(:,135));

plotconfusion(TestLabels, Mode_p(:,135))

%% One vs. all LR

p = zeros(size(TestImages,1),kmax, length(s2));


TrainLabels_ova = zeros(size(TrainLabels,1), 1);


for k = 1:length(s2)
    k
    pca_train = TrainImages_Tilde*V(:,1:s2(k));
    pca_test = TestImages_Tilde*V(:,1:s2(k));
for i = 1:10
    i
    for j = 1:size(TrainLabels,1)
        j
        if TrainLabels(j) == i-1
        TrainLabels_ova(j) = 1;
        else
        TrainLabels_ova(j) = -1;
        end
    end
    glm = fitglm(pca_train, categorical(TrainLabels_ova),'linear', 'distr', 'binomial');
    p(:,i,k) = predict(glm, pca_test);
end
end


[~,I] = max(transpose(p(:,:,150)));

testerrors = sum(I'~=TestLabels)/numel(TestLabels)

p(:,1,1)

%% Random Forest
Trees = [100, 300, 500, 700];

testError_RF = zeros(length(Trees), 1);

tic
for i = 1:4;
```

```
    i
    TB = TreeBagger(Trees(i), TrainImages, TrainLabels,'oobpred', 'on');
    pred = predict(TB, TestImages);
    pred = cellfun(@str2num, pred);
    testError_RF(i) = sum(pred~=TestLabels)/numel(TestLabels);
end
run_time_RF_test_error = toc/60; saveFinal_Math285_WS.mat;

order = 1:4;

fHand = figure;
aHand = axes('parent', fHand);
hold(aHand, 'on')
colors = hsv(numel(testError_RF));
for i = 1:numel(testError_RF)
    bar(i, testError_RF(i),'parent', aHand, 'facecolor', colors(i,:));
end
a = [cellstr(num2str(get(gca,'ytick')'*100))];
pct = char(ones(size(a,1),1)*'%');
new_yticks = [char(a),pct];
set(gca,'yticklabel',new_yticks)
set(gca, 'XTick', 1:numel(testError_RF), 'XTickLabel', Trees)
title({'Test Error for Random Forests'}, 'fontsize', 15)
xlabel('# of Trees', 'fontsize', 14)
ylabel('Errors (%)', 'fontsize', 14)
for i1=1:numel(testError_RF)
    text(order(i1),testError_RF(i1),[num2str(testError_RF(i1)*100),'%'],...
               'HorizontalAlignment','center',...
               'VerticalAlignment','bottom', 'color', 'red')
end
saveas(gcf, 'Final_Plot_7_a','pdf');

TB = TreeBagger(Trees(3), TrainImages, TrainLabels,'oobpred', 'on');
pred = predict(TB, TestImages);
pred_tree_five_hundred = cellfun(@str2num, pred);

Conf_Mat = confusionmat(TestLabels,pred_tree_five_hundred);
mat = Conf_Mat/1000;          %# A 5-by-5 matrix of random values from 0 to 1
imagesc(mat);            %# Create a colored plot of the matrix values
colormap(flipud(pink)); %# Change the colormap to gray (so higher values are
                        %#   black and lower values are white)

textStrings = num2str(mat(:),'%0.2f');  %# Create strings from the matrix values
textStrings = strtrim(cellstr(textStrings)); %# Remove any space padding
[x,y] = meshgrid(1:10);   %# Create x and y coordinates for the strings
hStrings = text(x(:),y(:),textStrings(:),...      %# Plot the strings
                'HorizontalAlignment','center');
midValue = mean(get(gca,'CLim'));  %# Get the middle value of the color range
textColors = repmat(mat(:) > midValue,1,3); %# Choose white or black for the
                                            %#   text color of the strings so
                                            %#   they can be easily seen over
                                            %#   the background color
set(hStrings,{'Color'},num2cell(textColors,2)); %# Change the text colors
title({'Confusion Matrix for Random Forests with tree = 500'}, 'fontsize', 15)
set(gca,'XTick',1:10,...                         %# Change the axes tick marks
        'XTickLabel',label_names,...  %#   and tick labels
        'YTick',1:10,...
        'YTickLabel',label_names,...
        'TickLength',[0 0]);
```

```
set(gca,'XTickLabel','')
set(gca(), 'XTickLabel', label_names);
rotateXLabels(gca(), 45 );
saveas(gcf, 'Final_Plot_7_b','pdf');
%% svm gaussian kernel one vs. one

rng(6)
k = 9;
n = 600;
num_class = 10;

% To set the gaussian kernel parameter, sigma: %%%%%%%%%%%%%%%%%%%%
[random_images, random_idx] = datasample(pca_train, n);
random_labels = TrainLabels(random_idx,:);

sum_distance = zeros(num_class,1);

for i = 0:9;
    i
    DigitsTrain = pca_train(TrainLabels == i,:);
    DigitsRandom = random_images(random_labels == i,:);
    [~, Distantce] = knnsearch(DigitsTrain, DigitsRandom,'k', k);
    sum_distance(i+1) = sum(Distantce(:,k));
end

SIGMA = sum(sum_distance)/n;



pca_train = TrainImages_Tilde*V(:,1:367);
pca_test = TestImages_Tilde*V(:,1:367);


power = (-4:5);
C = zeros(length(power),1);

for p = 1:length(power)
    C(p) = 2^power(p);
end


s3 = [find(cum_s>0.80, 1,'first'),find(cum_s>0.85, 1,'first'),find(cum_s>0.90, 1,'first'),↙
find(cum_s>0.95, 1,'first'),find(cum_s>0.9750, 1,'first')];

testError_svm_gaussian_ovo = zeros(length(C), length(s3));

tic
for k = 1:length(s3)
    k
    pca_train = TrainImages_Tilde*V(:,1:s3(k));
    pca_test = TestImages_Tilde*V(:,1:s3(k));
parfor i = 1:length(C);
    i
    temp = templateSVM('BoxConstraint', C(i), 'KernelFunction', 'gaussian', 'KernelScale',↙
SIGMA);
    Mdl = fitcecoc(pca_train, TrainLabels,'Coding', 'onevsone', 'learner',temp);
    p = predict(Mdl, pca_test);
    testError_svm_gaussian_ovo(i,k) = sum(TestLabels~=p)/numel(TestLabels);
end
```

```
end
run_time_svm_gaussian_ovo = toc/60; saveFinal_Math285_WS.mat

figure;
hold on;
plot(1:10, testError_svm_gaussian_ovo,'-o', 'markersize', 10, 'linewidth', 2)
set(gca,'XTickLabel',{'2^{-4}', '2^{-3}', '2^{-2}', '2^{-1}', '2^0', '2^1', '2^2', '2^3',↙
'2^4', '2^5'})
a = [cellstr(num2str(get(gca,'ytick')'*100))];
pct = char(ones(size(a,1),1)*'%');
new_yticks = [char(a),pct];
set(gca,'yticklabel',new_yticks)
title({'Test Errors for','PCA + SVM Gaussian Kernel + One vs. One:',['s=',num2str(s3(1)),',',↙
num2str(s3(2)),',', num2str(s3(3)),',', num2str(s3(4)),',', num2str(s3(end))]}, 'fontsize', 15)
xlabel('C', 'fontsize', 14)
ylabel('Errors (%)', 'fontsize', 14)
h_legend = legend( '33','55','99', '217', '382', 'Location', 'eastoutside');
title(h_legend, 'PCA Dimension', 'fontsize',10);
[~,I] = min(testError_svm_gaussian_ovo(:));
[I_row_svm_guassian_ovo, I_col_svm_gaussian_ovo] = ind2sub(size(testError_svm_gaussian_ovo),I);
xmin_svm_gaussian_ovo = I_row_svm_guassian_ovo;
ymin_svm_gaussian_ovo = min(testError_svm_gaussian_ovo(:));
strmin = ['Minimum Test Error= ',num2str(ymin_svm_gaussian_ovo*100),'%'];
plot(xmin_svm_gaussian_ovo, ymin_svm_gaussian_ovo,'^', 'MarkerSize', 12,'Color', 'red');
text(xmin_svm_gaussian_ovo,ymin_svm_gaussian_ovo,strmin,'Color','red','FontSize',↙
10,'VerticalAlignment','top', 'HorizontalAlignment', 'right');
saveas(gcf, 'Final_Plot_8_a','pdf');

%% SVM Gaussian Kernel one vs. all

testError_svm_gaussian_ova = zeros(length(C), length(s3));


tic
for k = 1:length(s3)
    k
    pca_train = TrainImages_Tilde*V(:,1:s3(k));
    pca_test = TestImages_Tilde*V(:,1:s3(k));
parfor i = 1:length(C);
    i
    temp = templateSVM('BoxConstraint', C(i), 'KernelFunction', 'gaussian', 'KernelScale',↙
SIGMA);
    Mdl = fitcecoc(pca_train, TrainLabels,'Coding', 'onevsall', 'learner',temp);
    p = predict(Mdl, pca_test);
    testError_svm_gaussian_ova(i,k) = sum(TestLabels~=p)/numel(TestLabels);
end
end
run_time_svm_gaussian_ova = toc/60; saveFinal_Math285_WS.mat


figure;
hold on;
plot(1:10, testError_svm_gaussian_ova,'-o', 'markersize', 10, 'linewidth', 2)
set(gca,'XTickLabel',{'2^{-4}', '2^{-3}', '2^{-2}', '2^{-1}', '2^0', '2^1', '2^2', '2^3',↙
'2^4', '2^5'})
a = [cellstr(num2str(get(gca,'ytick')'*100))];
pct = char(ones(size(a,1),1)*'%');
new_yticks = [char(a),pct];
set(gca,'yticklabel',new_yticks)
```

```
title({'Test Errors for','PCA + SVM Gaussian Kernel + One vs. All:',['s=',num2str(s3(1)),',',↙
num2str(s3(2)),',', num2str(s3(3)),',', num2str(s3(4)),',', num2str(s3(end))]}, 'fontsize', 15)
xlabel('C', 'fontsize', 14)
ylabel('Errors (%)', 'fontsize', 14)
h_legend = legend( '33','55','99', '217', '382', 'Location', 'eastoutside');
title(h_legend, 'PCA Dimension', 'fontsize',10);
[~,I] = min(testError_svm_gaussian_ova(:));
[I_row_svm_guassian_ova, I_col_svm_gaussian_ova] = ind2sub(size(testError_svm_gaussian_ova),I);
xmin_svm_gaussian_ova = I_row_svm_guassian_ova;
ymin_svm_gaussian_ova = min(testError_svm_gaussian_ova(:));
strmin = ['Minimum Test Error= ',num2str(ymin_svm_gaussian_ova*100),'%'];
plot(xmin_svm_gaussian_ova, ymin_svm_gaussian_ova,'^', 'MarkerSize', 12,'Color', 'red');
text(xmin_svm_gaussian_ova,ymin_svm_gaussian_ova,strmin,'Color','red','FontSize',↙
10,'VerticalAlignment','top', 'HorizontalAlignment', 'right');
saveas(gcf, 'Final_Plot_9_a','pdf');

pca_train = TrainImages_Tilde*V(:,1:s3(I_col_svm_gaussian_ova));
pca_test = TestImages_Tilde*V(:,1:s3(I_col_svm_gaussian_ova));
temp = templateSVM('BoxConstraint', C(I_row_svm_guassian_ova),'KernelFunction', 'gaussian',↙
'KernelScale',SIGMA);
Mdl = fitcecoc(pca_train, TrainLabels,'Coding', 'onevsall', 'learner',temp);
p = predict(Mdl, pca_test);

p_con_pca_svm = p;
Conf_Mat = confusionmat(TestLabels,p_con_pca_svm);
mat = Conf_Mat/1000;          %# A 5-by-5 matrix of random values from 0 to 1
imagesc(mat);            %# Create a colored plot of the matrix values
colormap(flipud(pink)); %# Change the colormap to gray (so higher values are
                        %#   black and lower values are white)

textStrings = num2str(mat(:),'%0.2f');  %# Create strings from the matrix values
textStrings = strtrim(cellstr(textStrings)); %# Remove any space padding
[x,y] = meshgrid(1:10);  %# Create x and y coordinates for the strings
hStrings = text(x(:),y(:),textStrings(:),...      %# Plot the strings
                'HorizontalAlignment','center');
midValue = mean(get(gca,'CLim'));  %# Get the middle value of the color range
textColors = repmat(mat(:) > midValue,1,3); %# Choose white or black for the
                                            %#   text color of the strings so
                                            %#   they can be easily seen over
                                            %#   the background color
set(hStrings,{'Color'},num2cell(textColors,2)); %# Change the text colors
title({'Confusion Matrix for','PCA(s=217) + SVM Gaussian Kernel(C=4) + One vs. All'},↙
'fontsize', 15)
set(gca,'XTick',1:10,...                         %# Change the axes tick marks
        'XTickLabel',label_names,...  %#   and tick labels
        'YTick',1:10,...
        'YTickLabel',label_names,...
        'TickLength',[0 0]);
set(gca,'XTickLabel','')
set(gca(), 'XTickLabel', label_names);
rotateXLabels(gca(), 45 );
saveas(gcf, 'Final_Plot_9_b','pdf');

%% PCA + Local kMeans
s3 = [find(cum_s>0.80, 1, 'first'),find(cum_s>0.85, 1, 'first'),find(cum_s>0.90, 1, 'first'),↙
find(cum_s>0.95, 1, 'first'),find(cum_s>0.9750, 1, 'first')];

Dist = zeros(size(TestImages,1),length(unique(TrainLabels)));
```

```
testError_pca_local_kMeans = zeros(kmax,length(s3));
tic
for l = 1:length(s3)
    l
    pca_train = TrainImages_Tilde*V(:,1:s3(l));
    pca_test = TestImages_Tilde*V(:,1:s3(l));
for k = 2:11;
    k
    for j = 0:9;
        j
        DigitsTrain = pca_train(TrainLabels == j,:);

        IDX = knnsearch(DigitsTrain,pca_test,'K',k);

        for i = 1:size(pca_test,1);
            i
            m = DigitsTrain(IDX(i,:),:);

            c = mean(m);

            Dist(i,j+1) = norm(pca_test(i,:) - c);

        end

    end

    Dist;

    [M,I] = min(Dist,[],2);

    newLabels = I-1;

    testError_pca_local_kMeans(k-1,l) = sum(TestLabels~=newLabels)/numel(TestLabels);
end
end;
runtime_pca_local_kMeans = toc/60; saveFinal_Math285_WS.mat;


figure;
hold on;
plot(1:10, testError_pca_local_kMeans,'-o', 'markersize', 10, 'linewidth', 2)
set(gca,'XTickLabel',2:11)
a = [cellstr(num2str(get(gca,'ytick')'*100))];
pct = char(ones(size(a,1),1)*'%');
new_yticks = [char(a),pct];
set(gca,'yticklabel',new_yticks)
title({'Test Errors for','PCA + Local kMeans:',['s=',num2str(s3(1)),',', num2str(s3(2)),',',↙
num2str(s3(3)),',', num2str(s3(4)),',', num2str(s3(end))]}, 'fontsize', 15)
xlabel('C', 'fontsize', 14)
ylabel('Errors (%)', 'fontsize', 14)
h_legend = legend( '33','55','99', '217', '382', 'Location', 'eastoutside');
title(h_legend, 'PCA Dimension', 'fontsize',10);
[~,I] = min(testError_pca_local_kMeans(:));
[I_row_pca_local_kmeans, I_col_pca_local_kmeans] = ind2sub(size(testError_pca_local_kMeans),I);
xmin_pca_local_kmeans = I_row_pca_local_kmeans;
ymin_pca_local_kmeans = min(testError_pca_local_kMeans(:));
strmin = ['Minimum Test Error= ',num2str(ymin_pca_local_kmeans*100),'%'];
plot(xmin_pca_local_kmeans, ymin_pca_local_kmeans,'^', 'MarkerSize', 12,'Color', 'red');
text(xmin_pca_local_kmeans,ymin_pca_local_kmeans,strmin,'Color','red','FontSize',↙
```

```
10,'VerticalAlignment','top', 'HorizontalAlignment', 'right');
saveas(gcf, 'Final_Plot_10_a','pdf');

% pca + local kmeans confusion matrix


Dist = zeros(size(TestImages,1),length(unique(TrainLabels)));

testError_pca_local_kMeans_con = zeros(kmax,1);
tic

pca_train = TrainImages_Tilde*V(:,1:s3(I_col_pca_local_kmeans));
pca_test = TestImages_Tilde*V(:,1:s3(I_col_pca_local_kmeans));
for k = 2:11;
    k
    for j = 0:9;
        j
        DigitsTrain = pca_train(TrainLabels == j,:);

        IDX = knnsearch(DigitsTrain,pca_test,'K',k);

        for i = 1:size(pca_test,1);
            i
            m = DigitsTrain(IDX(i,:),:);

            c = mean(m);

            Dist(i,j+1) = norm(pca_test(i,:) - c);

        end

    end

    Dist;

    [M,I] = min(Dist,[],2);

    newLabels = I-1;

    testError_pca_local_kMeans_con(k-1,l) = sum(TestLabels~=newLabels)/numel(TestLabels);
end
runtime_pca_local_kMeans_con = toc/60; saveFinal_Math285_WS.mat;

Conf_Mat = confusionmat(TestLabels,newLabels);
mat = Conf_Mat/1000;          %# A 5-by-5 matrix of random values from 0 to 1
imagesc(mat);            %# Create a colored plot of the matrix values
colormap(flipud(pink)); %# Change the colormap to gray (so higher values are
                         %#   black and lower values are white)

textStrings = num2str(mat(:),'%0.2f');  %# Create strings from the matrix values
textStrings = strtrim(cellstr(textStrings)); %# Remove any space padding
[x,y] = meshgrid(1:10);  %# Create x and y coordinates for the strings
hStrings = text(x(:),y(:),textStrings(:),...      %# Plot the strings
                'HorizontalAlignment','center');
midValue = mean(get(gca,'CLim'));  %# Get the middle value of the color range
textColors = repmat(mat(:) > midValue,1,3); %# Choose white or black for the
                                            %#   text color of the strings so
                                            %#   they can be easily seen over
                                            %#   the background color
```

```
set(hStrings,{'Color'},num2cell(textColors,2)); %# Change the text colors
title({'Confusion Matrix for','PCA(s=217) + Local kMeans (K=11)}, 'fontsize', 15)
set(gca,'XTick',1:10,...                         %# Change the axes tick marks
        'XTickLabel',label_names,...  %#   and tick labels
        'YTick',1:10,...
        'YTickLabel',label_names,...
        'TickLength',[0 0]);
set(gca,'XTickLabel','')
set(gca(), 'XTickLabel', label_names);
rotateXLabels(gca(), 45 );
saveas(gcf, 'Final_Plot_10_b','pdf');
```