Department of Electrical and Computer Engineering

Rutgers University – College of Engineering

New Brunswick, NJ

Course: 14:332:472

Robotic & Computer Vision Project 4

12/06/2016

Eun-Sol Kim

133000680

%run('/Users/e.kim4/Documents/MATLAB/vlfeat-0.9.20/toolbox/vl\_setup')

%vl\_version verbose

## 1: Bag Of Words

clear;

close all;

clc;

imgDir = '/Users/e.kim4/Documents/MATLAB/RCV\_project4/vision\_dataset';

imds = imageDatastore(imgDir, 'IncludeSubfolders', true, 'LabelSource', 'foldernames');

%split into testing vs train images: split is 5,5

[testImages,trainImages] = splitEachLabel(imds,5);

%label the trainImages 1~10

trainlabels = grp2idx(trainImages.Labels);

sizelabels = size(trainlabels);

concatDesTrain = [];

for i = 1:sizelabels(1)

train{i} = imread(trainImages.Files{i});

test{i} = imread(testImages.Files{i});

%read each train images % test images

if size(train{i},3) ~= 3

train{i} = imresize(train{i},[300,300]);

elseif size(train{i},3) ~= 3

test{i} = imresize(test{i},[300,300]);

else

train{i} = rgb2gray(imresize(train{i},[300,300]));

test{i} = rgb2gray(imresize(test{i},[300,300]));

end

%format them as single (just to run it on vl\_sift)

%singleTrain{i} = single(train{i});

%singleTest{i} = single(test{i});

%For a subset of all the interstpoints in training image, clustering

%the descriptors using k-means clustering:

%output as k-visual words with each word has an associated 128x1

%centroids

%%%USING VL\_SIFT: find interest points and descriptor for testing and train images

%[interestTrain{i}, desTrain{i}] = vl\_sift(singleTrain{i});

%[interestTest{i}, desTest{i}] = vl\_sift(singleTest{i});

%USING MATLAB BUILT IN FUNCTION

interestTrain{i} = detectSURFFeatures(train{i});

%getOnly 150 strongest interestpoint to compute

interestTrain{i} = interestTrain{i}.selectStrongest(100);

[desTrain{i}, validPTrain{i}] = extractFeatures(train{i}, interestTrain{i},'SURFSize',128);

interestTest{i} = detectSURFFeatures(test{i});

interestTest{i} = interestTest{i}.selectStrongest(100);

[desTest{i}, validPTest{i}] = extractFeatures(test{i}, interestTest{i},'SURFSize',128);

%put all the descriptors in one matrix for testing and train images

concatDesTrain = [concatDesTrain desTrain{i}'];

%concatDesTest = [concatDesTest desTest{i}];

end;

## For K=300 clustering

K = 300;

%so in here, assignments = closest centroids for each of Descriptor.

[centroid, assignTrain] = vl\_kmeans(double(concatDesTrain), K);

%seperate each assigned descriptors into each classes.

for i = 1:sizelabels(1)

[closestToCentroidsTrain{i}, distanceTrain{i}] = knnsearch(centroid',desTrain{i});

end

for i = 1:sizelabels(1)

[closestToCentroidsTest{i}, distanceTest{i}] = knnsearch(centroid',desTest{i});

end

concatFVTrain = [];

for i = 1:sizelabels(1)

featureVectorTrain{i} = zeros(1,K);

featureVectorTest{i} = zeros(1,K);

%can be either size of Test or Train

sizeDescript = size(closestToCentroidsTest{i});

for j=1:sizeDescript(1)

featureVectorTrain{i}(1,closestToCentroidsTrain{i}(j,1)) = featureVectorTrain{i}(1,closestToCentroidsTrain{i}(j,1)) + 1;

featureVectorTest{i}(1,closestToCentroidsTest{i}(j,1)) = featureVectorTest{i}(1,closestToCentroidsTest{i}(j,1)) + 1;

end

% concatFVTrain's each row contains histogram of eachImages

concatFVTrain(i,:) = [featureVectorTrain{i}];

concatFVTest(i,:) = [featureVectorTest{i}];

end

%%show similar histograms

figure(1)

for i = 1:4

if i <= 2

subplot(2,2,i)

bar(featureVectorTrain{45+i});

%featureVectorTest{trackTrain} = dummyTrain.Values;

hold on;

axis([0 K+1 0 15])

title(sprintf('Similarity: class %d Train image %d: Zebra',10,45+i));

hold off;

else

subplot(2,2,i)

bar(featureVectorTest{45+i});

%featureVectorTest{trackTest} = dummyTest.Values;

hold on;

axis([0 K+1 0 15])

title(sprintf('Similarity: class %d Test image %d: Zebra',10,45+i));

hold off;

end

end

%%show different histogram

figure(2)

for i = 1:4

if i <= 2

subplot(2,2,i)

bar(featureVectorTrain{45+i});

%featureVectorTest{trackTrain} = dummyTrain.Values;

hold on;

axis([0 K+1 0 15])

title(sprintf('Similarity: class %d Train image %d: Zebra',10,45+i));

hold off;

else

subplot(2,2,i)

bar(featureVectorTest{1+i});

%featureVectorTest{trackTest} = dummyTest.Values;

hold on;

axis([0 K+1 0 15])

title(sprintf('Similarity: class %d Test image %d: baseball',1,1+i));

hold off;

end

end

%%TrainHistogram vs TestHistogram

%compared a set of concatTrainHistogram with TestingHistogram(1).

[concatFVTestLabel, concatFVTestDistance] = knnsearch(concatFVTest,concatFVTrain);

for i = 1:size(concatFVTestLabel,1)

if concatFVTestLabel(i,1) >= 1 && concatFVTestLabel(i,1) < 6

concatFVTestLabel(i) = 1;

elseif concatFVTestLabel(i,1) >= 6 && concatFVTestLabel(i,1) < 11

concatFVTestLabel(i) = 2;

elseif concatFVTestLabel(i,1) >= 11 && concatFVTestLabel(i,1) < 16

concatFVTestLabel(i) = 3;

elseif concatFVTestLabel(i,1) >= 16 && concatFVTestLabel(i,1) < 21

concatFVTestLabel(i) = 4;

elseif concatFVTestLabel(i,1) >= 21 && concatFVTestLabel(i,1) < 26

concatFVTestLabel(i) = 5;

elseif concatFVTestLabel(i,1) >= 26 && concatFVTestLabel(i,1) < 31

concatFVTestLabel(i) = 6;

elseif concatFVTestLabel(i,1) >= 31 && concatFVTestLabel(i,1) < 36

concatFVTestLabel(i) = 7;

elseif concatFVTestLabel(i,1) >= 36 && concatFVTestLabel(i,1) < 41

concatFVTestLabel(i) = 8;

elseif concatFVTestLabel(i,1) >= 41 && concatFVTestLabel(i,1) < 46

concatFVTestLabel(i) = 9;

else

concatFVTestLabel(i) = 10;

end

end

%%ERROR CHECKING

errorclass = zeros(10,1);

for i = 1:size(concatFVTestLabel,1)

if i>=1 && i <6 && concatFVTestLabel(i) ~= 1

errorclass(1) = errorclass(1) + 1/5;

elseif i >= 6 && i < 11 && concatFVTestLabel(i) ~= 2

errorclass(2) = errorclass(2) + 1/5;

elseif i >= 11 && i < 16 && concatFVTestLabel(i) ~= 3

errorclass(3) = errorclass(3) + 1/5;

elseif i >= 16 && i < 21 && concatFVTestLabel(i) ~= 4

errorclass(4) = errorclass(4) + 1/5;

elseif i >= 21 && i < 26 && concatFVTestLabel(i) ~= 5

errorclass(5) = errorclass(5) + 1/5;

elseif i >= 26 && i < 31 && concatFVTestLabel(i) ~= 6

errorclass(6) = errorclass(6) + 1/5;

elseif i >= 31 && i < 36 && concatFVTestLabel(i) ~= 7

errorclass(7) = errorclass(7) + 1/5;

elseif i >= 36 && i < 41 && concatFVTestLabel(i) ~= 8

errorclass(8) = errorclass(8) + 1/5;

elseif i >= 41 && i < 46 && concatFVTestLabel(i) ~= 9

errorclass(9) = errorclass(9) + 1/5;

elseif i >= 46 && i < 51 && concatFVTestLabel(i) ~= 10

errorclass(10) = errorclass(10) + 1/5;

end

end

%%confusion matrix

%idk if the confusionmatrix is correct

stats = confusionmatStats(trainlabels,concatFVTestLabel);

%plotting the interest points

%showing the class butterfly(3),carplate(4),watch(9)

colors = distinguishable\_colors(50);

figure(11)

for i = 1:4

subplot(2,2,i)

imshow(train{10+i})

hold on;

plot(interestTrain{10+i});

title(sprintf('trainImage: butterfly%d with interest points',i))

hold off;

end

figure(12)

for i = 1:4

subplot(2,2,i)

imshow(train{15+i})

hold on;

plot(interestTrain{15+i});

title(sprintf('trainImage: carplate%d with interest points',i))

hold off;

end

figure(13)

for i = 1:4

subplot(2,2,i)

imshow(train{40+i})

hold on;

plot(interestTrain{40+i});

title(sprintf('trainImage: carplate%d with interest points',i))

hold off;

end

%plot(interestTrain{1}.Location(11,1),interestTrain{1}.Location(11,2),'\*');

%plot(interestTrain{1}.Location(244,1),interestTrain{1}.Location(258,2),'\*')

## For K=200 clustering

K200 = 200;

%so in here, assignments = closest centroids for each of Descriptor.

[centroid200, assignTrain200] = vl\_kmeans(double(concatDesTrain), K200);

%seperate each assigned descriptors into each classes.

for i = 1:sizelabels(1)

[closestToCentroidsTrain200{i}, distanceTrain200{i}] = knnsearch(centroid200',desTrain{i});

end

for i = 1:sizelabels(1)

[closestToCentroidsTest200{i}, distanceTest200{i}] = knnsearch(centroid200',desTest{i});

end

concatFVTrain200 = [];

for i = 1:sizelabels(1)

featureVectorTrain200{i} = zeros(1,K200);

featureVectorTest200{i} = zeros(1,K200);

%can be either size of Test or Train

sizeDescript200 = size(closestToCentroidsTest200{i});

for j=1:sizeDescript200(1)

featureVectorTrain200{i}(1,closestToCentroidsTrain200{i}(j,1)) = featureVectorTrain200{i}(1,closestToCentroidsTrain200{i}(j,1)) + 1;

featureVectorTest200{i}(1,closestToCentroidsTest200{i}(j,1)) = featureVectorTest200{i}(1,closestToCentroidsTest200{i}(j,1)) + 1;

end

% concatFVTrain's each row contains histogram of eachImages

concatFVTrain200(i,:) = [featureVectorTrain200{i}];

concatFVTest200(i,:) = [featureVectorTest200{i}];

end

%%TrainHistogram vs TestHistogram

%compared a set of concatTrainHistogram with TestingHistogram(1).

[concatFVTestLabel200, concatFVTestDistance200] = knnsearch(concatFVTest200,concatFVTrain200);

for i = 1:size(concatFVTestLabel200,1)

if concatFVTestLabel200(i,1) >= 1 && concatFVTestLabel200(i,1) < 6

concatFVTestLabel200(i) = 1;

elseif concatFVTestLabel200(i,1) >= 6 && concatFVTestLabel200(i,1) < 11

concatFVTestLabel200(i) = 2;

elseif concatFVTestLabel200(i,1) >= 11 && concatFVTestLabel200(i,1) < 16

concatFVTestLabel200(i) = 3;

elseif concatFVTestLabel200(i,1) >= 16 && concatFVTestLabel200(i,1) < 21

concatFVTestLabel200(i) = 4;

elseif concatFVTestLabel200(i,1) >= 21 && concatFVTestLabel200(i,1) < 26

concatFVTestLabel200(i) = 5;

elseif concatFVTestLabel200(i,1) >= 26 && concatFVTestLabel200(i,1) < 31

concatFVTestLabel200(i) = 6;

elseif concatFVTestLabel200(i,1) >= 31 && concatFVTestLabel200(i,1) < 36

concatFVTestLabel200(i) = 7;

elseif concatFVTestLabel200(i,1) >= 36 && concatFVTestLabel200(i,1) < 41

concatFVTestLabel200(i) = 8;

elseif concatFVTestLabel200(i,1) >= 41 && concatFVTestLabel200(i,1) < 46

concatFVTestLabel200(i) = 9;

else

concatFVTestLabel200(i) = 10;

end

end

%%ERROR CHECKING

errorclass200 = zeros(10,1);

for i = 1:size(concatFVTestLabel200,1)

if i>=1 && i <6 && concatFVTestLabel200(i) ~= 1

errorclass200(1) = errorclass200(1) + 1/5;

elseif i >= 6 && i < 11 && concatFVTestLabel200(i) ~= 2

errorclass200(2) = errorclass200(2) + 1/5;

elseif i >= 11 && i < 16 && concatFVTestLabel200(i) ~= 3

errorclass200(3) = errorclass200(3) + 1/5;

elseif i >= 16 && i < 21 && concatFVTestLabel200(i) ~= 4

errorclass200(4) = errorclass200(4) + 1/5;

elseif i >= 21 && i < 26 && concatFVTestLabel200(i) ~= 5

errorclass200(5) = errorclass200(5) + 1/5;

elseif i >= 26 && i < 31 && concatFVTestLabel200(i) ~= 6

errorclass200(6) = errorclass200(6) + 1/5;

elseif i >= 31 && i < 36 && concatFVTestLabel200(i) ~= 7

errorclass200(7) = errorclass200(7) + 1/5;

elseif i >= 36 && i < 41 && concatFVTestLabel200(i) ~= 8

errorclass200(8) = errorclass200(8) + 1/5;

elseif i >= 41 && i < 46 && concatFVTestLabel200(i) ~= 9

errorclass200(9) = errorclass200(9) + 1/5;

elseif i >= 46 && i < 51 && concatFVTestLabel200(i) ~= 10

errorclass200(10) = errorclass200(10) + 1/5;

end

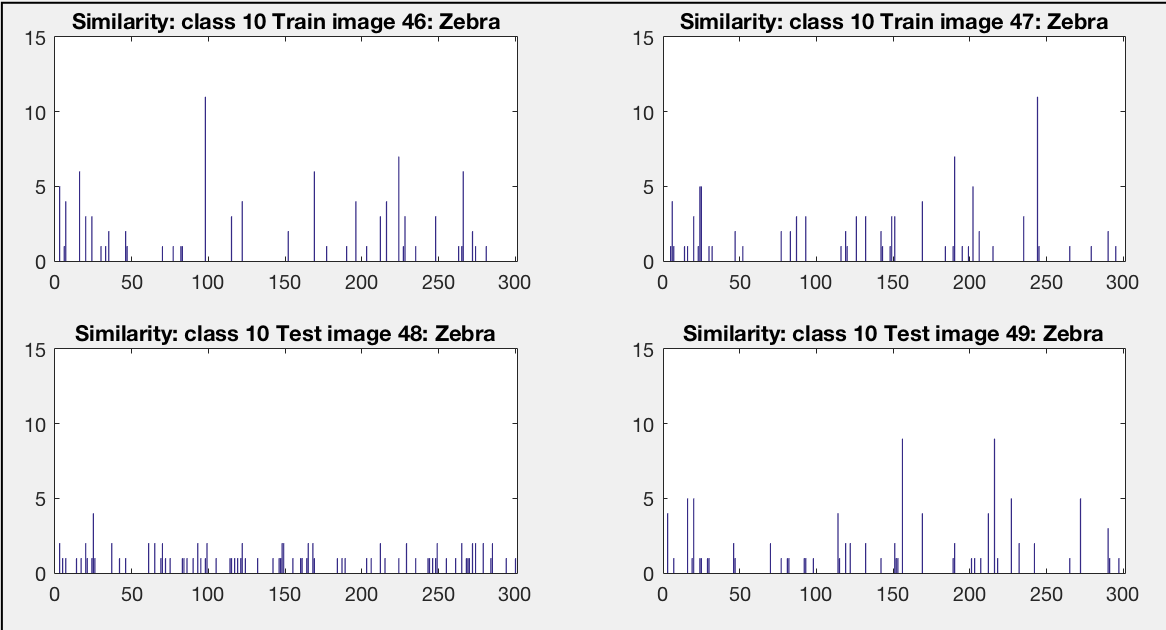
end

%confusion matrix K=200

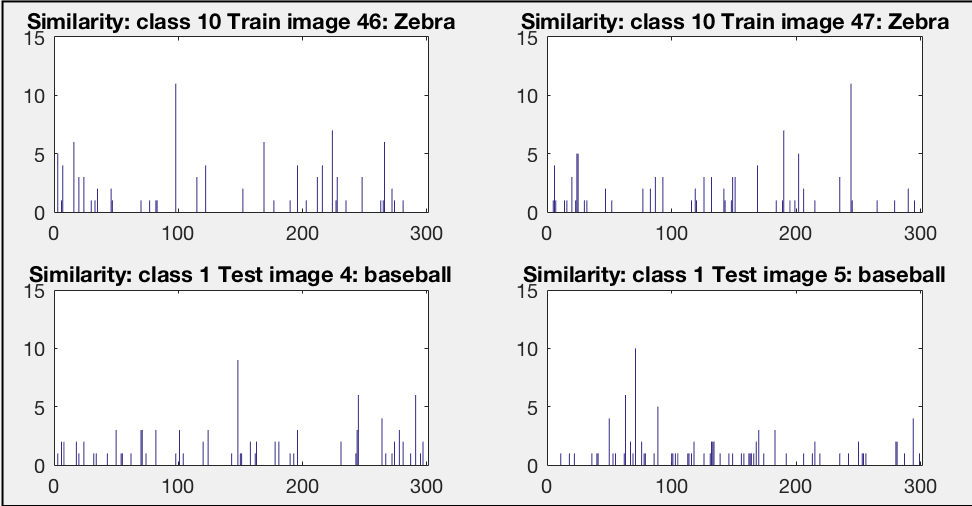
%idk if the confusionmatrix is correct

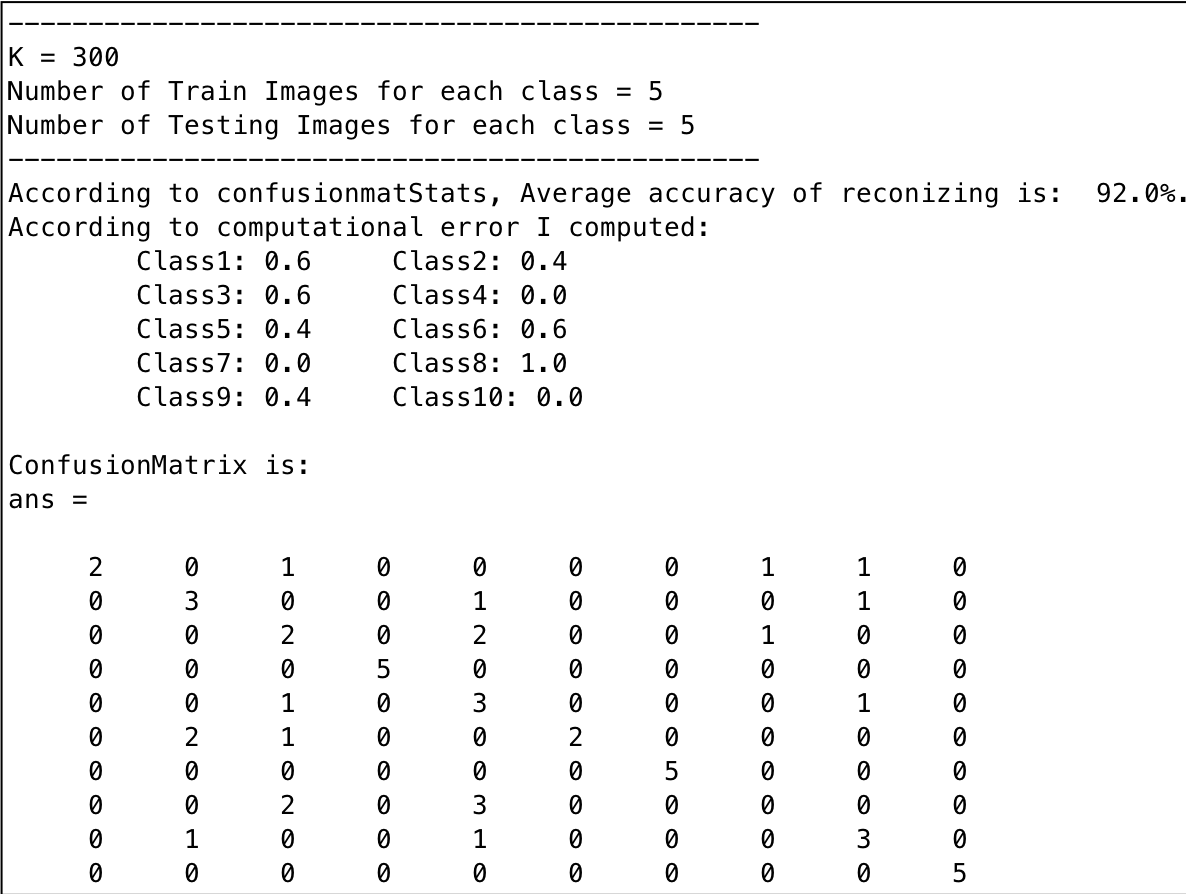
stats200 = confusionmatStats(trainlabels,concatFVTestLabel200);

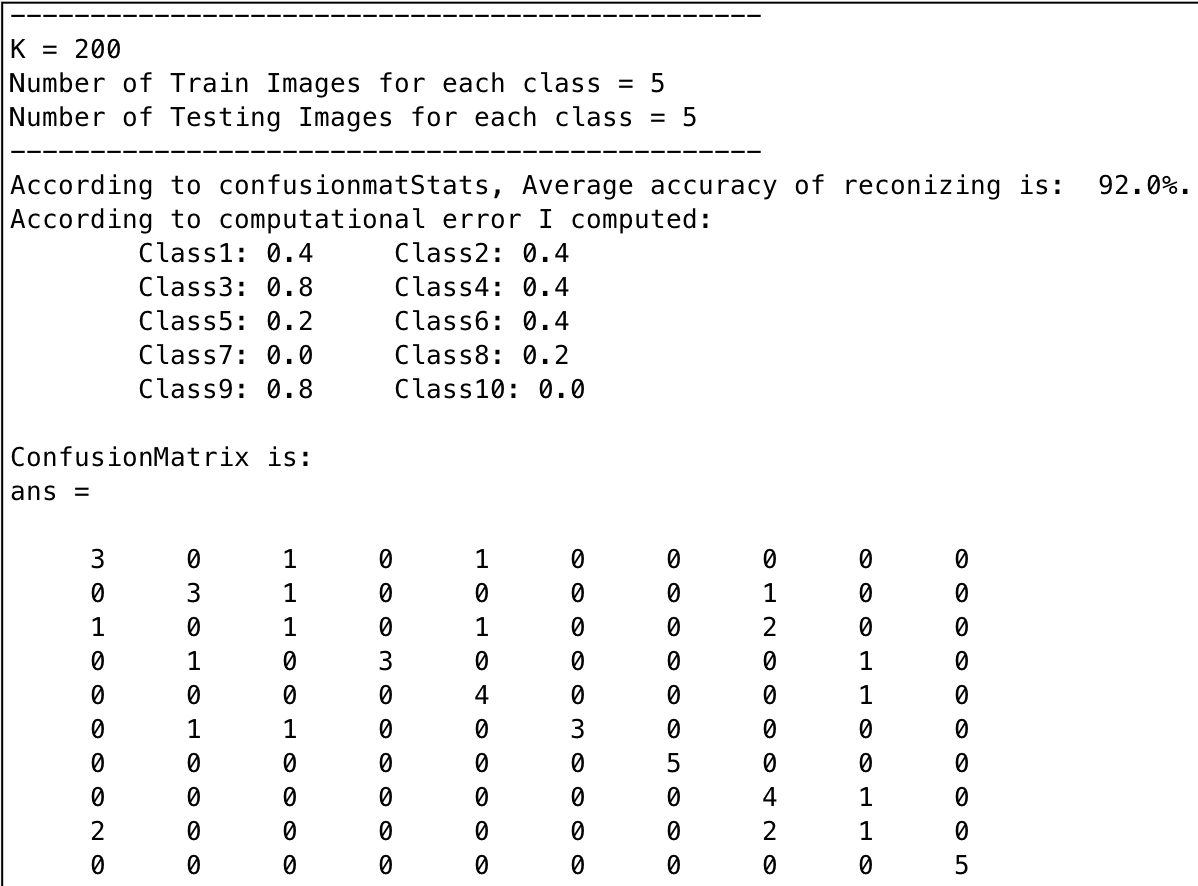
* For histograms, I chose to draw with bar graph because it seems more accurate.



* These title should be ‘Differences’ instead of ‘Similarity’







* Classes: Baseball, Bicycle, Butterfly, Carplate, Flower, Keyboard(piano), Leopard, Pizza, Watch, Zebra
* By computing Bag of Words with two different K values (I actually computed different K values too but too much to show in here), I learned that as K values increases, there are significantly better chance of reducing error for class 4 and class 9. Inversely, there were significantly worse chance of reducing error for class 8. Since all the images are not selected from all the interest Points; rather, picked out 100 strongest points, there might be loss of information such that not stable.
* The pictures with interestpoints <100 will not work with this code because I, purposely, chose pictures with large interestpoints.

# **2: PCA recognition**

%eigs

clear;

close all;

clc;

imgDir = '/Users/e.kim4/Documents/MATLAB/RCV\_project4/vision\_dataset2';

imds = imageDatastore(imgDir, 'IncludeSubfolders', true, 'LabelSource', 'foldernames');

[testFace,trainFace] = splitEachLabel(imds,5);

trainlabels = grp2idx(trainFace.Labels);

sizelabels = size(trainlabels);

for i = 1:sizelabels(1)

% each img is N=243 x P=320 M = numberofimages = 10 per class

% 100 for whole classes

trainF{i} = imread(trainFace.Files{i});

testF{i} = imread(testFace.Files{i});

if size(trainF{i},3)== 3

trainF{i} = rgb2gray(imread(trainFace.Files{i}));

elseif size(testF{i},3) ==3

testF{i} = rgb2gray(imread(testFace.Files{i}));

end

if i == 1

% has (N\*P)x M dimension

trainA = zeros(prod(size(trainF{1})),sizelabels(1));

testA = zeros(prod(size(testF{1})),sizelabels(1));

trainMean = zeros(size(trainF{1}));

%testMean = zeros(size(testF{1}));

end

%avgface has NxP dimension

trainMean = trainMean + double(trainF{i});

%testMean = testMean + double(trainF{i});

trainA(:,i) = trainF{i}(:);

testA(:,i) = testF{i}(:);

end;

%%compute averageface to show; averageface for computation

trainMean = mean(trainA,2);

%testMean = mean(testA,2);

%%compute the differences of original image - Mean face

%removing all common face features that the faces share together

%so that each face is left with each unique features

for i = 1:size(trainA,2)

%(N\*P) x M

%subtract each column with averageface

trainA\_Mean(:,i) = trainA(:,i) - trainMean;

testA\_Mean(:,i) = testA(:,i) - trainMean;

end

%%compute covariance Matrix & get eigenvector and eigenvalues

%originalC = A\_Mean \* transpose(A\_Mean);

%better to reduce the dimensionality to reduce noise and

%number of computation

trainReducedC = transpose(trainA\_Mean)\*trainA\_Mean;

testReducedC = transpose(testA\_Mean)\*testA\_Mean;

%%Choosing K using SVD

%such that K<=M and represent whole training set

%columns of U = eigenvectors

%S = eigenvalues

[trainU,trainS,trainV] = svd(trainReducedC);

[testU,testS,testV] = svd(testReducedC);

figure(1)

subplot(1,2,1)

plot(trainS);

subplot(1,2,2)

plot(testS);

hold off;

title('Decay of Eigenvalues')

xlabel('NM')

ylabel('eignvalues')

%can choose K here.

%find rank R = K

trainK = rank(trainReducedC);

testK = rank(testReducedC);

%%get eigenvector

trainEVecReduced = trainU(:,1:trainK);

testEVecReduced = testU(:,1:testK);

%%convert reduced dimensional K eigenvectors to origianl dimensionality

%u\_i = A\*v\_i;

trainEvecOriginal(:,1:trainK) = trainA\_Mean\*trainEVecReduced(:,1:trainK);

testEvecOriginal(:,1:testK) = testA\_Mean\*testEVecReduced(:,1:testK);

%%normalized each column

%trainEvecOriginal=normc(trainEvecOriginal);

%testEvecOriginal=normc(testEvecOriginal);

%%finding coefficient a by dotproduct

trainCoef = (trainA\_Mean'\*trainEvecOriginal);

testCoef = (testA\_Mean'\*testEvecOriginal);

%normalized each column

trainCoef = normc(trainCoef);

testCoef = normc(testCoef);

%%reconstruc with eigenfaces.

%originalFace = avgfaceForComp + a(1)\*eigvecOriginal(:,1)'+...+a(114)\*eigvecOriginal(114);

testFace = [];

for i=1:trainK

if i == 1

testFace = repmat(trainMean',sizelabels(1),1) + testCoef(:,i)\*testEvecOriginal(:,i)';

end

testFace = testFace + testCoef(:,i)\*testEvecOriginal(:,i)';

end

figure(1)

subplot(5,2,1)

imshow(uint8(reshape(testFace(1,:),[size(testF{1})])));

title(sprintf('K = %d: trainImage 1:’,testK))

subplot(5,2,2)

imshow(uint8(reshape(testFace(6,:),[size(testF{1})])));

title(sprintf('K = %d: trainImage 6:',testK))

subplot(5,2,3)

imshow(uint8(reshape(testFace(11,:),[size(testF{1})])));

title(sprintf('K = %d: trainImage 11:',testK))

subplot(5,2,4)

imshow(uint8(reshape(testFace(16,:),[size(testF{1})])));

title(sprintf('K = %d: trainImage 16:',testK))

subplot(5,2,5)

imshow(uint8(reshape(testFace(21,:),[size(testF{1})])));

title(sprintf('K = %d: trainImage 21:',testK))

subplot(5,2,6)

imshow(uint8(reshape(testFace(26,:),[size(testF{1})])));

title(sprintf('K = %d: trainImage 26:',testK))

subplot(5,2,7)

imshow(uint8(reshape(testFace(31,:),[size(testF{1})])));

title(sprintf('K = %d: trainImage 31:',testK))

subplot(5,2,8)

imshow(uint8(reshape(testFace(36,:),[size(testF{1})])));

title(sprintf('K = %d: trainImage 36:',testK))

subplot(5,2,9)

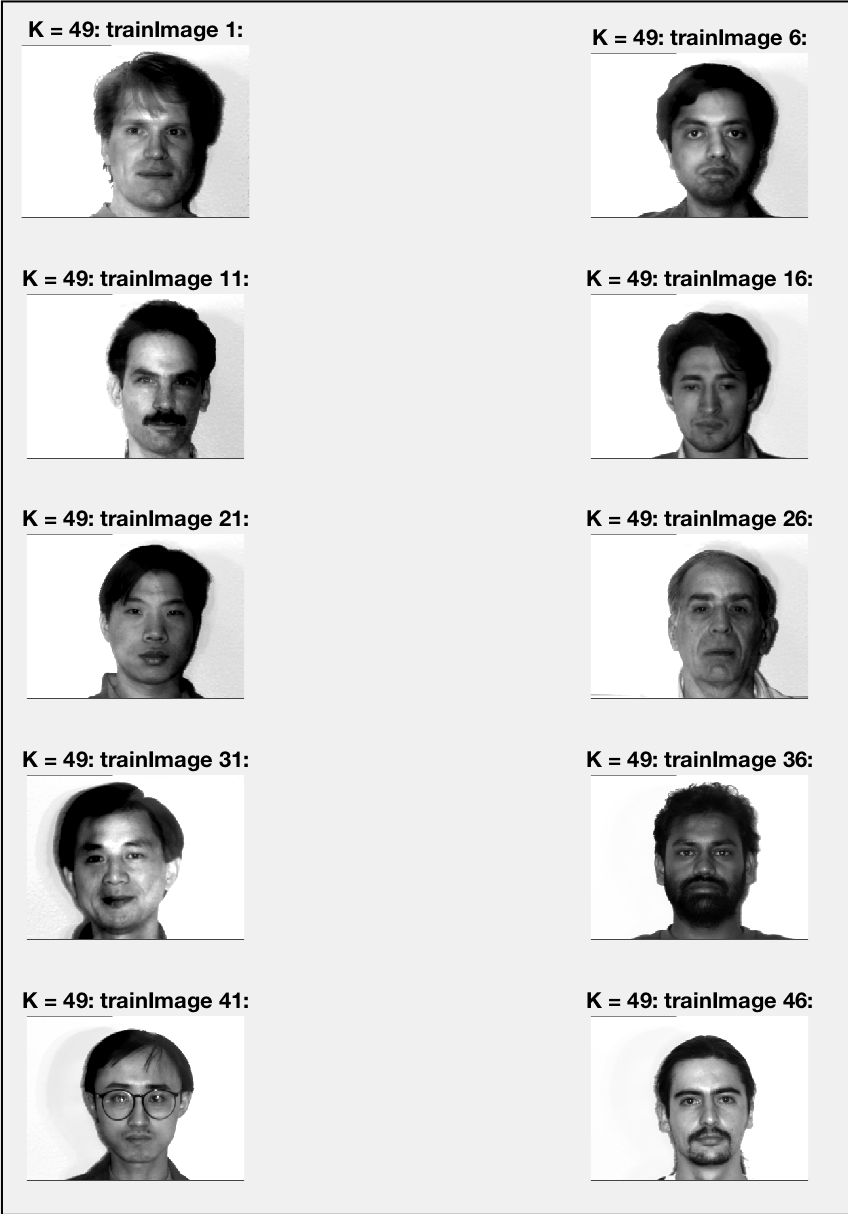
imshow(uint8(reshape(testFace(41,:),[size(testF{1})])));

title(sprintf('K = %d: trainImage 41:',testK))

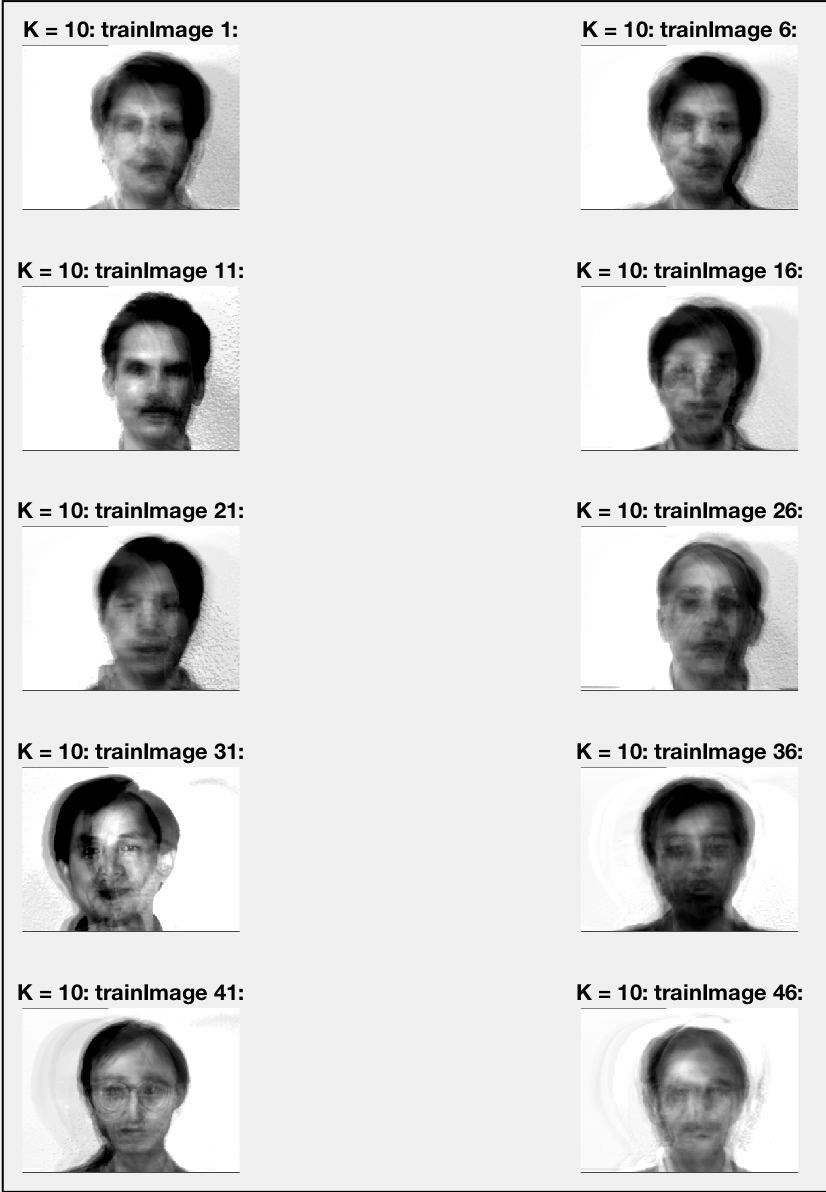
subplot(5,2,10)

imshow(uint8(reshape(testFace(46,:),[size(testF{1})])));

title(sprintf('K = %d: trainImage 46:',testK))







* As the K decreases, I learned that the image construction will more close to mean\_faces because there are not much of coefficient to represent each faces.

#3

Part1.1 Convolution

1. H’’ = H – 2, W’’ = W-(W’-1)
2. Bars

Part1.1.2: Convolution by a filter bank

1. K=3 for this example because there are 3 filters in the convolutions.

Part1.2: Non-Linear Activation

1. Does simple linear transformation over input data.
2. Laplacian: Horizontal, ReLu: Vertical

Part2:Backpropagation

1. Maybe it is derivative of different parameter
2. Yes: for……dx\_numerical(I,j,k,n) = (yp-y)/deta;

Part 2.2:Backpropagation

1. Because its doing projective derivative of p and vl\_nnconv does derivative with dy.

Part3.1

1. Convolution operator refines the images
2. The zero padding would disappearfor all cnn that is layer 2~5
3. Because it is not separated by ReLUs
4. 1) 6 layers

2) support: 3 for Conv, 1 for Relu;

NUM FIT CHANnel: 32 for conv, n/a for ReLU

1. The size (in pixel) of the local image region that affects a particular element in a feature map. Larger receptive field size might be preferable and can be obtained with larger layers.
2. Yes.
3. It is much slower
4. Slower
5. better