ECE 1395 Problem Set 5

Bryan Hess: 4259226

ps5-0.)

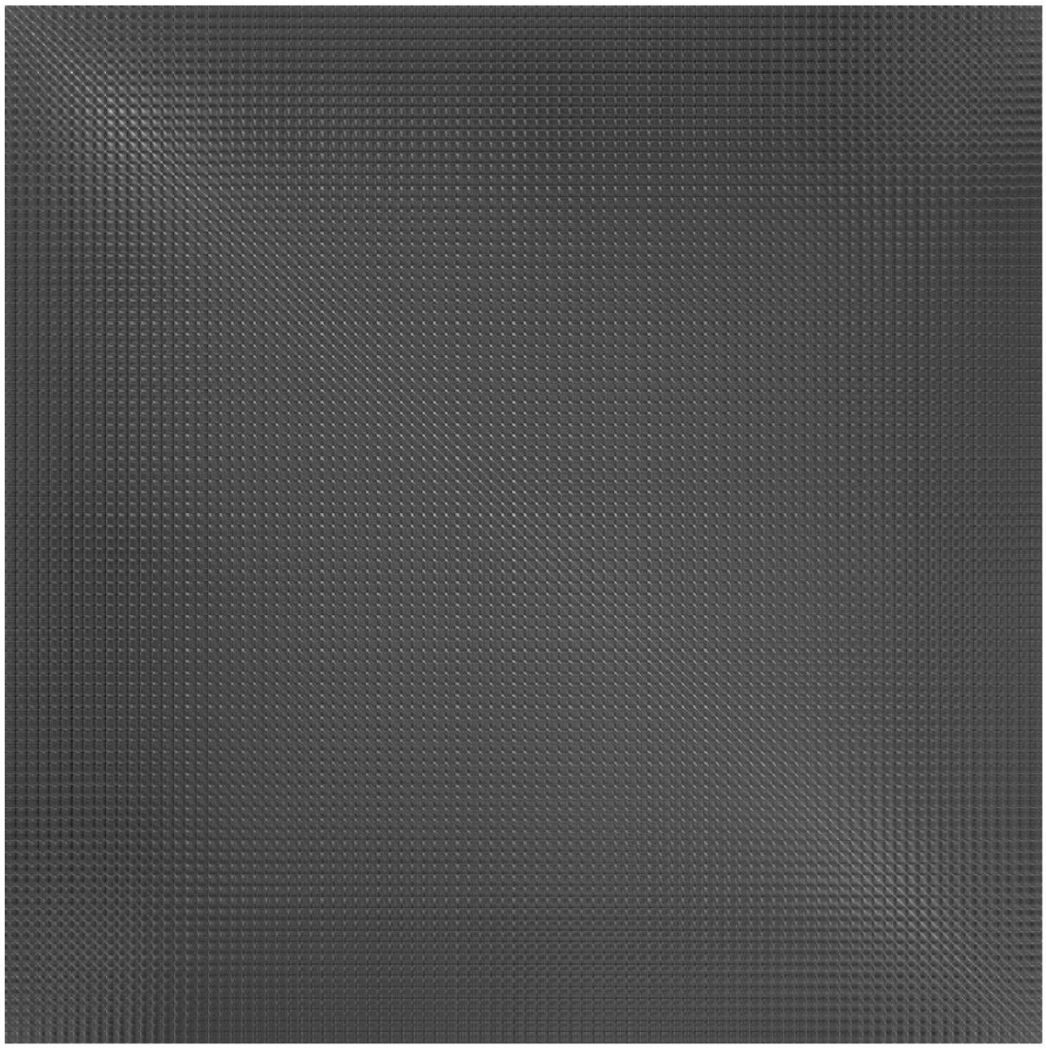
First training image selected. Most of the time this will be the face 1-1, unless 1-1 is selected to be a test face.

ps5-1-a.)

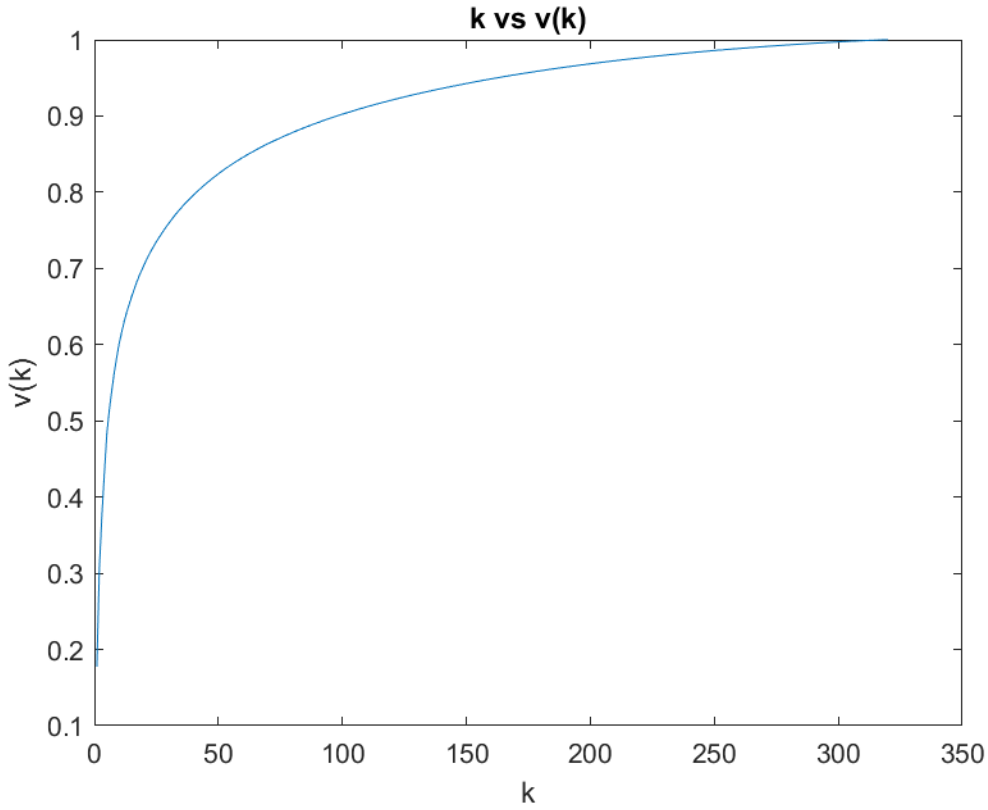
ps5-1-b.)

This image is the average face of the dataset. Given each face, this is the amalgomized combination of each data point. The mean face is important to extract as we need it in order to obtain the eigenvalues.

ps5-1-c.)

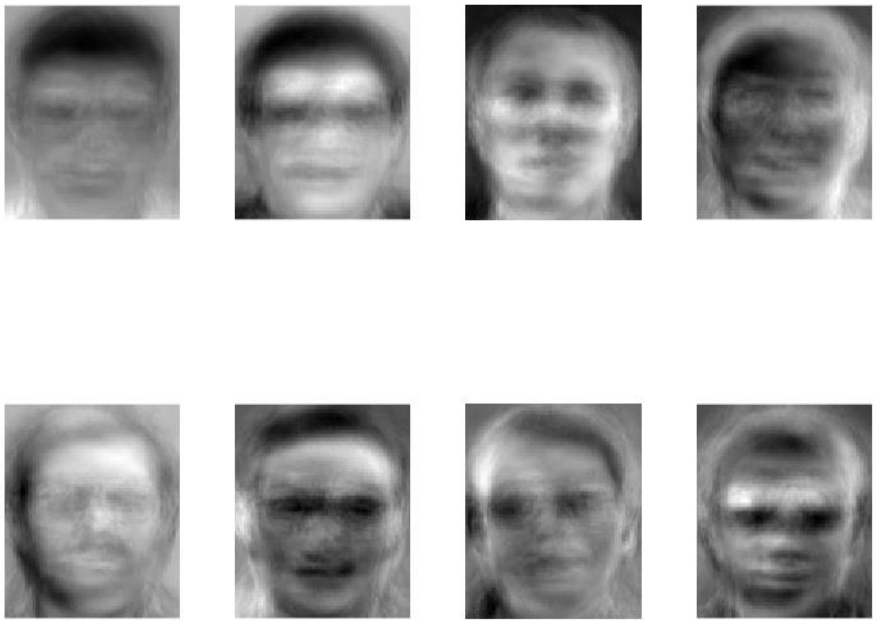


ps5-1-d.)



There are 162 k values that capture 95% of the variance.

ps5-1-e.)



```
>> ps5
The size of U is:
       10304         162
```

The eigenfaces are the principal components (or PCs) of a distribution of faces, also can be described as the eigenvectors of the covariance matrix of the set of face images. These are 8 faces pulled as an example. They are quite eerie!

ps5-2-a.)

```
Size of W_training
   320   162
```

ps5-2-b.)

```
Size of W_testing
    80   162
```

ps5-3-a.)

k Value	Accuracy
1	0.9875
3	0.975
5	0.925
7	0.875
9	0.7875
11	0.775

Utilizing KNN, we have accuracy deminish as more Ks are introduced. This makes sence as we are generalizing the data more and more as each additional K is introduced.

ps5-3-b.)

This part of the code I had issues with getting fitcsvm to work properly. I attached my attempted code below as well as the results of the opperations.


```
%%3.b SOMETHING DOES NOT ALLOW FOR THIS PART OF THE CODE TO PROPERLY
%%COMPUTE ACCURACIES
svm_accLin = zeros(1, 3);
svm_accPoly = zeros(1, 3);
svm_accRBF = zeros(1, 3);

for i=1:40
  svmLabels_training = zeros(320, 1);
  for j=1:8
    svmLabels_training((i-1)*8+j) = i;
  end
  svm_classLin{i} = fitcsvm(W_training, svmLabels_training, 'ClassNames', [false true], 'Standardize', true, 'KernelFunction', 'Linear', 'BoxConstraint', 1);
  svm_classPoly{i} = fitcsvm(W_training, svmLabels_training, 'ClassNames', [false true], 'Standardize', true, 'KernelFunction', 'polynomial', 'PolynomialOrder', 3);
  svm_classRBF{i} = fitcsvm(W_training, svmLabels_training, 'ClassNames', [false true], 'Standardize', true, 'KernelFunction', 'RBF');
end

svm_predLin = zeros(80, 1);
svm_predPoly = zeros(80, 1);
svm_predRBF = zeros(80, 1);

for i=1:80
  top1 = 0;
  top2 = 0;
  top3 = 0;
  for j=1:40
    [label, score] = predict(svm_classLin{j}, W_testing(i, :));
    if(score(1) > top1)
      svm_predLin(i) = j;
    end
  end
  if(labels_testing(i) == svm_predLin(i))
    svm_accLin(1) = svm_accLin(1) + 1;
  end
    [label, score] = predict(svm_classPoly{j}, W_testing(i, :));
    if(score(1) > top2)
      svm_predPoly(i) = j;
    end
```

```
  if(labels_testing(i) == svm_predPoly(i))
    svm_accPoly(l) = svm_accPoly(l) + 1;
  end
    [label, score] = predict(svm_classRBF{j}, W_testing(i, :));
    if(score(l) > top3)
      svm_predRBF(i) = j;
    end

  if(labels_testing(i) == svm_predRBF(i))
    svm_accRBF(l) = svm_accRBF(l) + 1;
  end
end

svm_accLin(l) = svm_accLin(l) / 80;
svm_accPoly(l) = svm_accPoly(l) / 80;
svm_accRBF(l) = svm_accRBF(l) / 80;

%%svmTable = table(svm_class, svm_acc,'VariableNames', {'SVM_class' 'Accuracy'});
%%disp(svmTable)
```

I know in general SVM take cares of outliers better. If training data is much larger than number of features), KNN is better. SVM outperforms KNN when there are large features and less training data.