HW 6 Computer Problem

Problem1: Implementing the PCA algorithm

Principle components algorithm for entire data set

```
% Preparing training data: Rows are number of samples, columns
% are features or dimension of training data
data = transpose(reshape (imageTrain, [28*28,5000]));

%Find the mean
data_mean = mean(data, 1); % taking means of features
data_mean = data_mean(ones(1,size(data,1)),:);

% Find covariant
data_cov = ((data-data_mean)'*(data-data_mean))/(size(data,1)-1);

% Get diagonal and arrage from largest to smallest
[prin_com, eig_val] = eig(data_cov);
eig_val = diag(eig_val);
[hold_val, idx] = sort(eig_val, 1, 'descend');

eig_val = eig_val(idx,1); % the top 10 eig values locate in the first 10 columns
prin_com= prin_com(:,idx); % top 10 PC locates in the first 10 columns
```

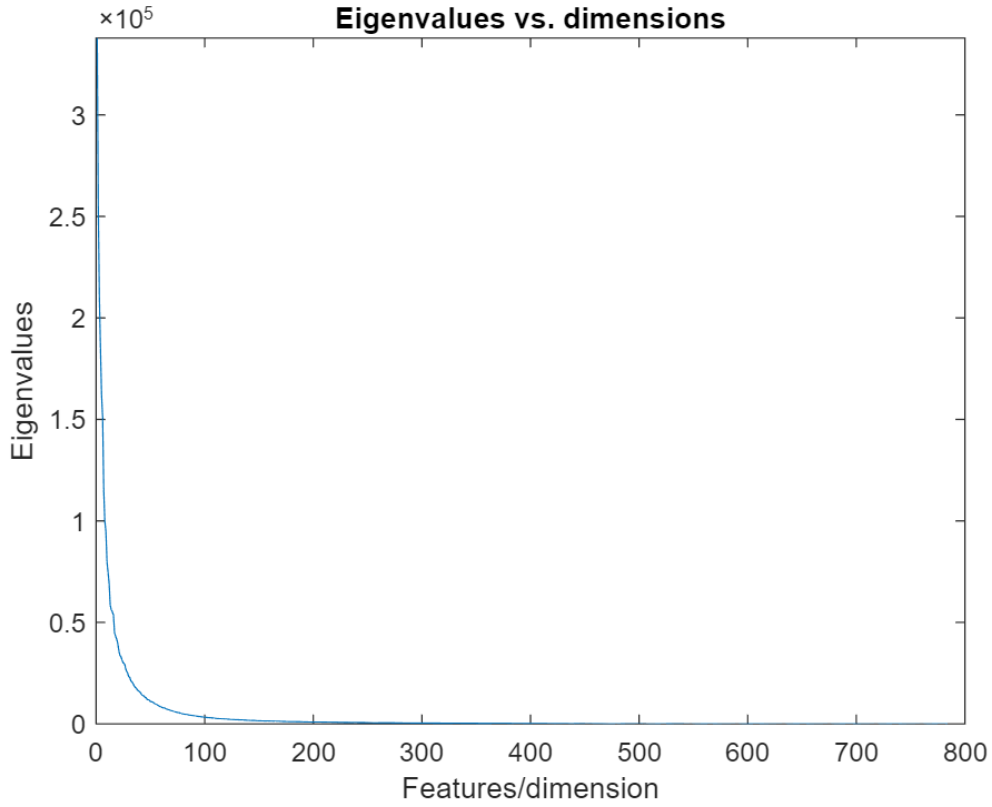
Displaying the top 10 principle component for entire data set

```
% extract the top 10 PC
top_10_PC = prin_com(:,1:10);

for i = 1:10
    subplot(2,5,i),axis image
    imshow(reshape(normalize(top_10_PC(:,i)), [28,28]));
end
```

Displaying the eigenvalues plot for entire data set

```
figure
plot (eig_val);
xlabel('Features/dimension');
ylabel("Eigenvalues")
title("Eigenvalues vs. dimensions")
ylim([0 inf])
```



Extracting data of digit 5

```
% keep in mind that each row contains an image
index5  = find(labelTrain == 5);
data_digit5 = data(index5, :);
[digit5_sam_num, feature_num] = size(data_digit5);
```

Apply PCA to digit 5 data

```
data_mean_5 = mean(data_digit5, 1); % taking means of features
data_mean_5 = data_mean_5(ones(1,size(data_digit5,1)),:);

% Find covariant
data_cov_5 = ((data_digit5-data_mean_5)'*(data_digit5-data_mean_5))/(size(data_digit5,1)-1);

% Get diagonal and arrage from largest to smallest
[prin_com_5, eig_val_5] = eig(data_cov_5);
eig_val_5 = diag(eig_val_5);
[hold_val_5, idx_5] = sort(eig_val_5, 1, 'descend');

eig_val_5 = eig_val_5(idx_5,1);
prin_com_5= prin_com_5(:,idx_5);
```

Displaying the 10 principle components for digit 5

```
% extract the top 10 PC of digit 5
top_10_PC_5= prin_com_5(:,1:10);
for i = 1:10
    subplot(2,5,i),axis image
    imshow(reshape(normalize(top_10_PC_5(:,i)), [28,28]));
end
```

Problem 2: Classification using the PCA subspace on the imageTest dataset

Part a)

According to the eigenvalues plot above it seems that dimensions 100 is the most efficient for classfication because the eigenvalues after that begin to trail off into zero which means that these dimensions don't add much to the full description of the data. That is, the majority of the data can be described with about 100 dimensions or features, and the rest can be deemed irrelevant information.

Part b) Calculate the total error rate using different subspaces

```
data_test = transpose(reshape (imageTest, [28*28,500]));
subspace=  [5, 10, 20, 30, 40, 60, 90, 130, 180, 250, 350];
error = zeros(1,length(subspace));
```

Calling the Bayes algorithm from HW 3

```
for i = 1: length(subspace)

    p = subspace(1,i);
    reduced_feat = prin_com(:, 1:p);
```

Reconstructing the testing and training data to the desired dimensions by finding the projection onto the subspaces.

```
    test_recons = (data_test*reduced_feat)';
    train_recons = (data*reduced_feat)';

    error(1,i) = Bayes(train_recons, test_recons, labelTrain, labelTest, p);
end
```

Displaying the error rates

```
subspace
```

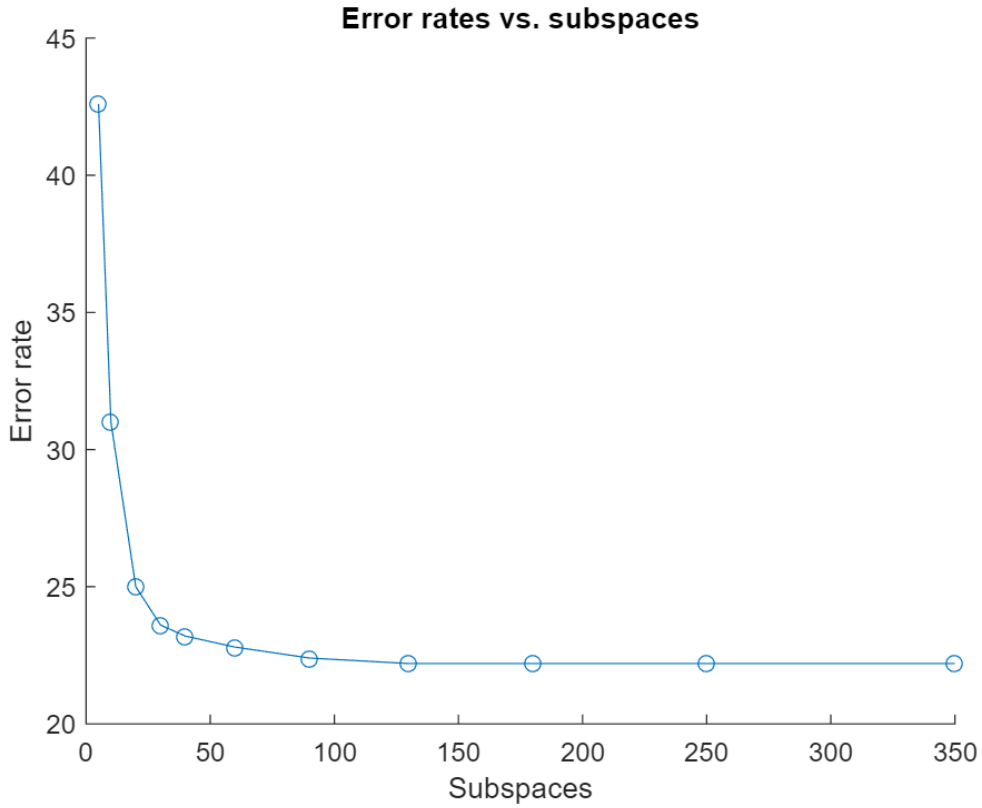
```
subspace = 1×11
     5    10    20    30    40    60    90   130   180   250   350
```

```
error
```

```
error = 1×11
   42.6000   31.0000   25.0000   23.6000   23.2000   22.8000   22.4000   22.2000 ···
```

Plotting the error rates

```
figure
scatter(subspace,error)
hold on
line(subspace,error)
title ('Error rates vs. subspaces');
ylabel('Error rate');
xlabel ('Subspaces');
```



Analysis: The error rate was high inititally but eventually tappers off and does not change much after subspace 100.

Part c) Compare your results with the final error rate obtained in problem set 3

The total error rate in HW 3 was 22.2%, and the error rate using Principle Component Algorithm is 22.2% as well; we can conclude that the error rate did not change at all and that put PCA is as efficient as our Bayes algorithm from HW 3. The PCA algorithm's purpose is to reduce the dimensions/features of data set so that any irrelevant features that don't contribute significantly to the data's spread are omitted. From the error plot above, we see that only about 100 features are significant in defining the data set, the inclusion or exclusion of any other subspaces does not change the result of classification. Thus, the observation from part a) based on the eigenvalues plot was correct, the best subspace dimension for classification is 100.

Below is the Bayes algorithm from HW 3, with some alterations:

```
function [total_error] = Bayes(train_recons, test_recons, labelTrain, labelTest, p)

class = 10;
mean_vector_matrix = zeros(p,class);

for i = 0: 9
    mean_list = train_recons(:,find(labelTrain == i));
    mean_vector_matrix(:,i+1) = mean(mean_list, 2);
end
```

```
sigma = eye (p,p); % sigma is equal to sigma inverse
det_sigma = det(sigma);
log_value = p*log(2*pi); % dimension = 784
log_PY = log(1/class); % N = 10 classes from 0 to 9
bayes_list = zeros (length(test_recons),class); % will be updated to contain the Bayes
% probabilities
bayes_index = zeros(length(test_recons),class); % will be updated to contain the indexes
% of Bayes probability, useful in sorting

for test_index = 1: length(labelTest)
 for class_index = 1 : class
 part_1 = (test_recons(:, test_index) - mean_vector_matrix(:, class_index))';
 part_2 = (test_recons(:, test_index) - mean_vector_matrix(:, class_index ));

 bayes_list(test_index, class_index) = (-1/2)*part_1*(inv(sigma))* part_2 -...
 (1/2)*log_value - (1/2)*log(det_sigma)+log_PY;
 % bayes_list now contains probability for all 10 classes
 end
 [bayes_list(test_index,:), bayes_index(test_index, :)] = ...
 sort(bayes_list(test_index, :), 'descend');
 % sorts the Bayes probability from largest to smallest to locate the
 % the maximum number and the index of that maximum number

end
predictions = bayes_index(:,1)-1;
total_error = size (find(predictions~=labelTest),1)/size(labelTest,1)*100;
end
```