UNIVERSITY OF VICTORIA
Department of Electrical and Computer Engineering
ECE 403/503 Optimization for Machine Learning
LABORATORY REPORT

Experiment No: 1

Title: Handwritten Digits Recognition Using PCA

Date of Experiment: 09/26/2019

Report Submitted on:10/07/2019

To: TA

Laboratory Group No.:

Name(s): Kurt Elliott & Raj Deepak S. N.

Objectives:

The objective of this laboratory experiment is to learn PCA as a technique for pattern recognition and apply it to the HWDR problem.

Introduction:

Handwritten digit recognition (HWDR) is a classic machine learning (ML) problem as its is used in varied fields of work. The machine should be able to understand and recognize different digits in an accurate and efficient manner. To do so, it requires a training dataset and to which it can apply the knowledge gained in HWDR. PCA works on the basis of singular variable decomposition, where the dimension of a matrix is reduced.

Results:

The following figures show the MATLAB script used in this lab as well as plotted results with a brief description of how the program is implemented. The program shown in figure 1 was the code used in this lab. It consists of three primary sections. The first section of the code is used to load the training, testing, and verification data-sets into variables in MATLAB and reshapes the first rows of the digits 1 thus 9 to verify the data is the correct set by plotting(figure 1).

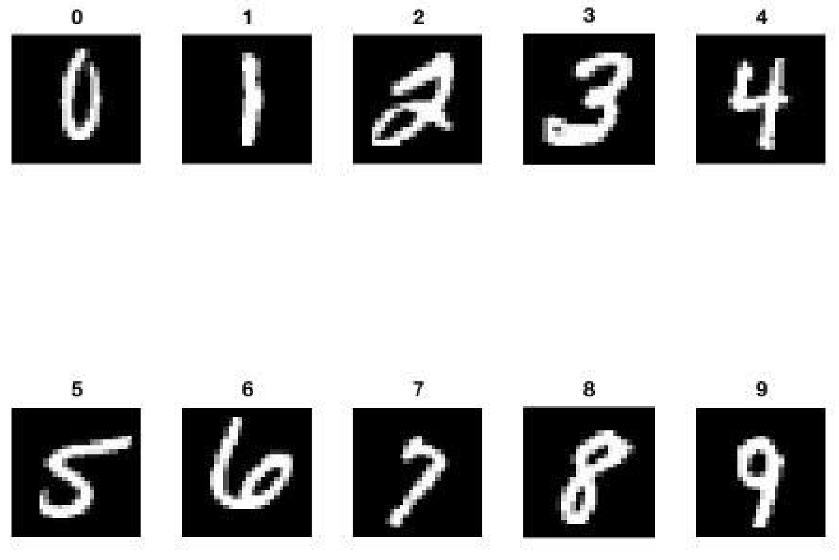


Figure 1 - Numbers 0 to 9

The next section consists of the principal component analysis(PCA) algorithm, which has six steps:

0. Initial variables for:
    - size of image row(784)
    - number of classes(10)
    - number of samples(1600)
    - number of rank approximation(29)
1. Loop through the training data and calculate the centerized mean and covariance of each of the 10 classes in the dataset, then put them in a matrix.
2. Calculate the eigenvectors of the covariance matrix.
3. Calculate the principal components.
4. Calculate principal components analysis approximation
5. Calculate error between PCA approximation and test dataset
6. Identify the target classes of PCA approximation by which class it has the lowest error too.

The final section of the MATLAB code is used to compute the CPU execution time of the PCA approximation and calculates the number of mislabelled data points and plot the data(figure 2).

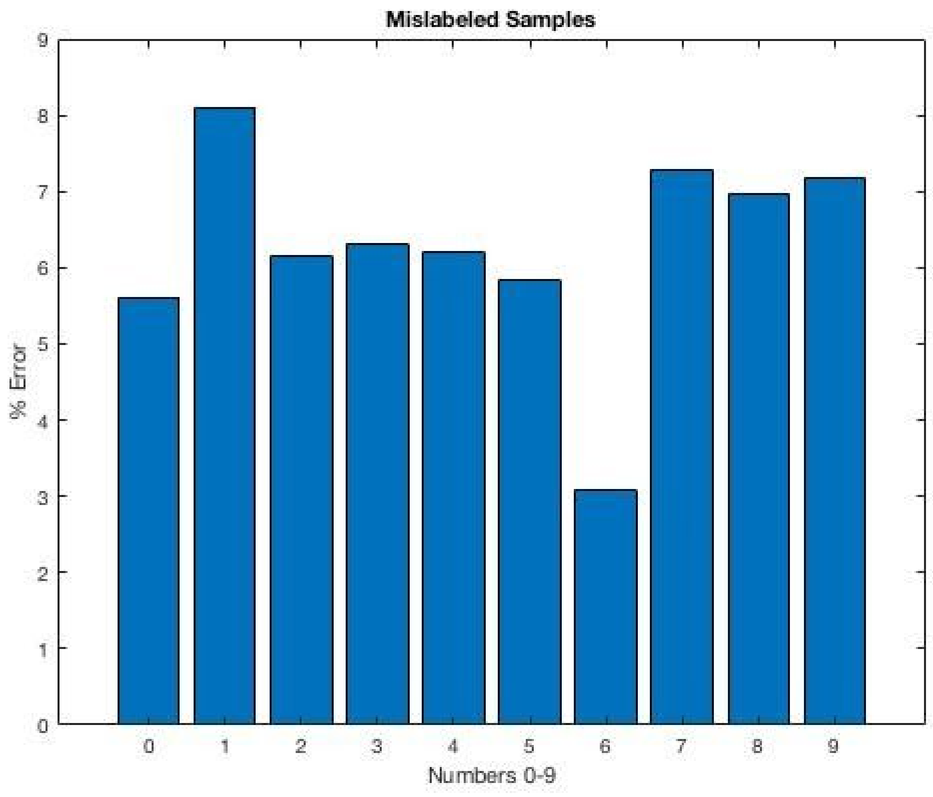


Figure 2 - Classification error for each digit

The following was the script used to execute the MATLAB code:

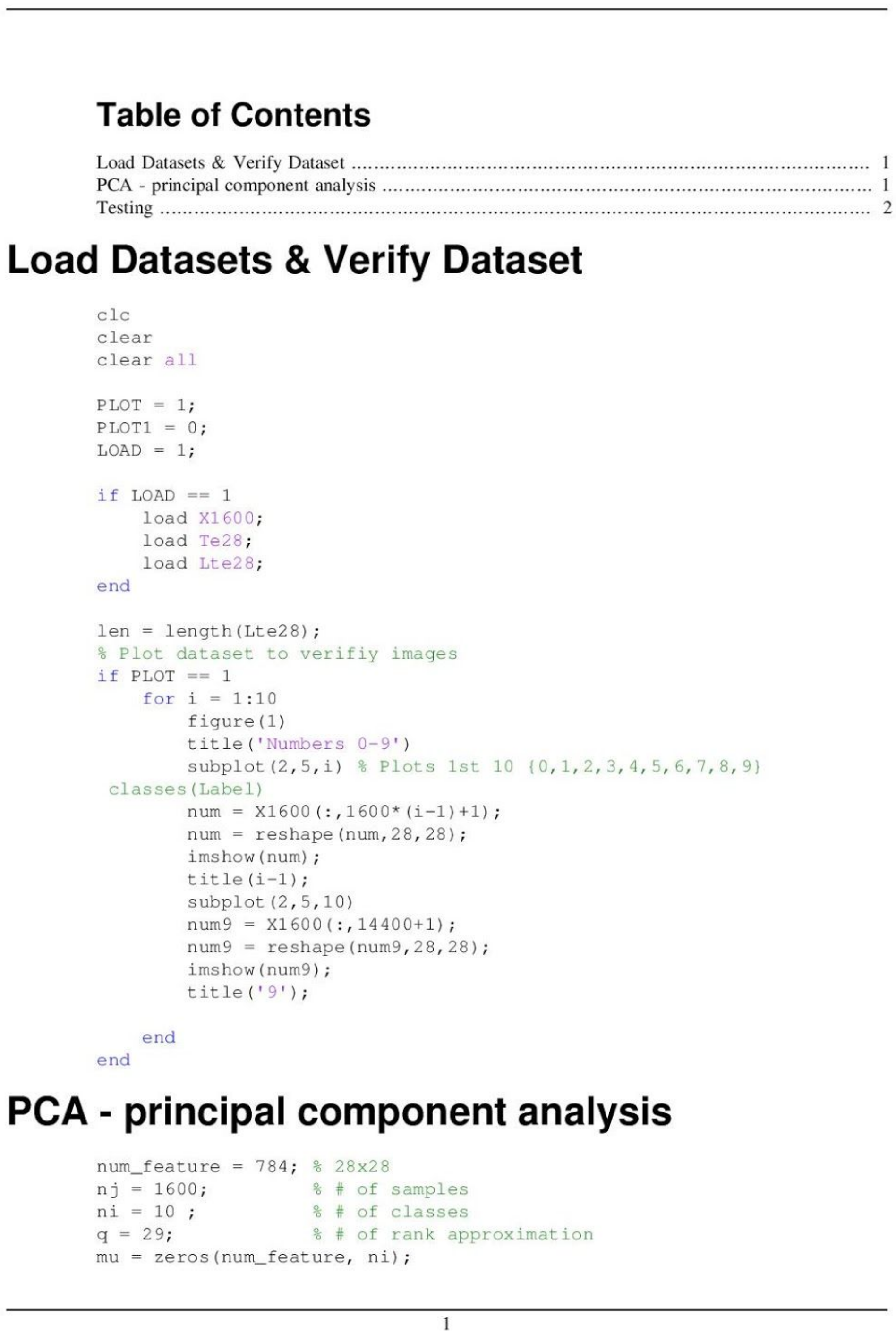


Table of Contents

Load Datasets & Verify Dataset

```
clc
clear
clear all

PLOT = 1;
PLOT1 = 0;
LOAD = 1;

if LOAD == 1
    load X1600;
    load Te28;
    load Lte28;
end

len = length(Lte28);
% Plot dataset to verifiy images
if PLOT == 1
    for i = 1:10
        figure(1)
        title('Numbers 0-9')
        subplot(2,5,i) % Plots 1st 10 {0,1,2,3,4,5,6,7,8,9}
 classes(Label)
        num = X1600(:,1600*(i-1)+1);
        num = reshape(num,28,28);
        imshow(num);
        title(i-1);
        subplot(2,5,10)
        num9 = X1600(:,14400+1);
        num9 = reshape(num9,28,28);
        imshow(num9);
        title('9');

    end
end
```

PCA - principal component analysis

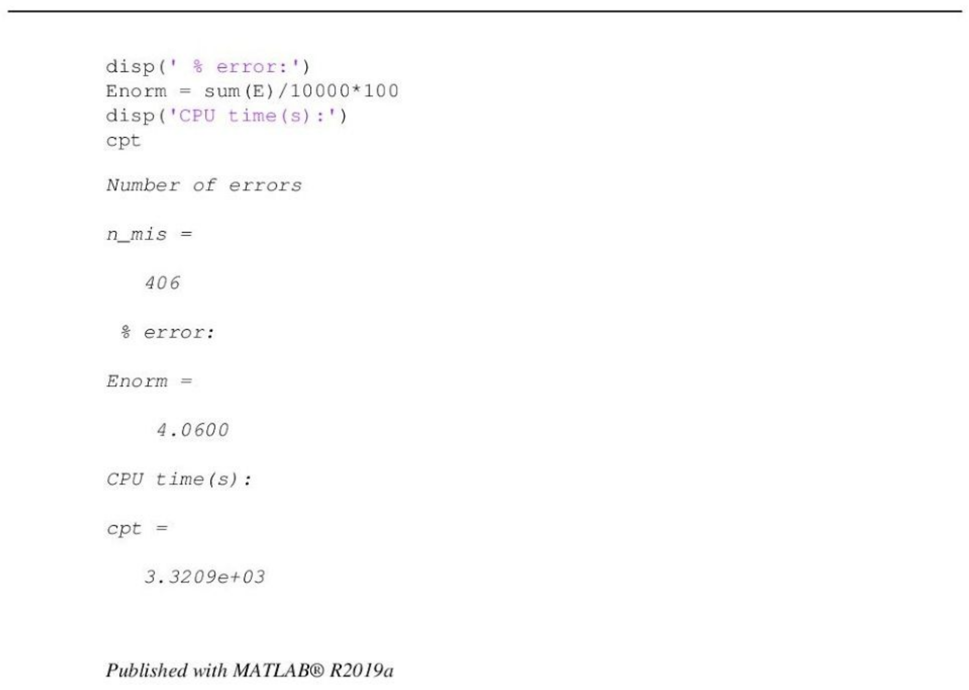
```
num_feature = 784; % 28x28
nj = 1600;         % # of samples
ni = 10 ;          % # of classes
q = 29;            % # of rank approximation
mu = zeros(num_feature, ni);
```

1

```
U = zeros(num_feature, q*ni);

for i = 1:ni
    % STEP-1 Calculate mean-j & covariance-j matrices
    Ai = X1600(:,(i-1)*nj+1:i*nj);
    mu_j = mean(Ai,2);
    Ah = Ai - mu_j*ones(1,nj);
    Cov = (Ah*Ah');

    % STEP-2 Calculate the q eigenvectors of covariance-j matrices
    [Uq,~] = eigs(Cov,q);
    U(:,(i-1)*q+1:i*q) = Uq;
    mu(:,i) = mu_j;
end
```

Testing

```
t0 = cputime;
Predicted_Label = zeros(len,1);

for j = 1:len
    At = Te28(:,j);
    e = zeros(1,ni);
    for i = 1:ni
        % STEP-3 Calculate principal components
        Cov2 = At - mu(:,i);
        fj = U(:,(i-1)*q+1:i*q).'*Cov2;

        % STEP-4 Calculate PCA approximation
        Aj = U(:,(i-1)*q+1:i*q)*fj + mu(:,i);

        % STEP-5 Calculate error
        e(i) = norm(At-Aj);
    end
    % STEP-6  Identify the target class Aj for data point x.
    [~,MinIdx] = min(e);
    Predicted_Label(j) = MinIdx - 1;
end

cpt = cputime - 1;
E = (Lte28 ~= Predicted_Label);

if PLOT1 == 1
    figure(3)
    bar(e);
    set(gca,'xticklabel',{'0','1','2','3','4','5','6','7','8','9'});
    xlabel('Numbers 0-9');
    ylabel('% Error');
    title('Mislabeled Samples');
end

disp('Number of errors')
n_mis = sum(E)
```



```
disp(' % error:')
Enorm = sum(E)/10000*100
disp('CPU time(s):')
cpt

Number of errors

n_mis =

   406

 % error:

Enorm =

    4.0600

CPU time(s):

cpt =

   3.3209e+03
```

Published with MATLAB® R2019a

Discussion:

This experiment involved using a PCA algorithm to classify handwritten digits from 0 through 9. The algorithm performed the best when rank approximation was set to 29, which provided 4.06% error or 406 mislabeled numbers in the dataset when tested on the 10,000 test samples set and had a CPU execution time of 3.3209e+03s. Also figure 2 shows the classification % error of each digit. Over all the algorithm performed quite well for not having to modify and be tuned. To achieve better results one could clean the sample data set more to discard any outlier or run another algorithm such a k-mean over the classified data to provide better grouping of each digit.

Conclusions:

PCA algorithm has been used to recognize handwritten digits from a sample of 10,000 images, 28 x 28 pixels each. When the rank is set to the optimum value (29), the number of mislabeled digits takes the least possible value (406) after an execution time of 3.3209e+03s.