UNIVERSITY OF VICTORIA

Department of Electrical and Computer Engineering

ECE 403/503 Optimization for Machine Learning

LABORATORY REPORT

Experiment No: 01

Title: Handwritten Digits Recognition Using PCA

Date of Experiment: 28 May, 2019

Report Submitted on: 4 June, 2019

To: Mr. J. Zhan

Laboratory Group No.: B01 T

Name(s): Alvi Mahadi (V00912845)

1 Introduction and Objectives

Research for handwritten digit recognition (HWDR) by machine learning (ML) techniques has stayed active in the past several decades. The sustained interest in HWDR is primarily due to its broad applications in bank check processing, postal mail sorting, automatic address reading, and mail routing, etc. In these applications, both accuracy and speed of digit recognition are critical indicators of system performance. In a machine learning setting, we are given a training data set consisting of a number of samples of handwritten digits, each belongs to one of the ten classes $D_j, j = 0, 1, ..., 9$ where class D_j collects all data samples labeled as digit j. The HWDR problem seeks to develop an approach to utilizing these known data classes to train a multi- category classifier so as to recognize handwritten digits outside the training data. The primary challenge arising from the HWDR problem lies in the fact that handwritten digits (within the same digit class) vary widely in terms of shape, line width, and style, even when they are properly centralized and normalized in size. Classification of multi-category data can be accomplished by ML methods that are originally intended for classifying binary-class data using the so-called one-versus-the-rest approach. Alternatively, there are ML techniques that deal with multi-category data directly. A representative method of this type is based on principal component analysis (PCA). The objective of this laboratory experiment is to learn PCA as a technique for pattern recognition and apply it to the HWDR problem.

2 Implementation Steps and Results

2.1 Implementation Steps

We strictly followed the implementation steps stated in the laboratory manual. Initially we had the computation of μ a little wrong leading us to a wrong result on the accuracy. The computation of μ was corrected by debugging in the lab time to get the desired result.

2.2 Code

2.2.1 PCA Extraction(pca.m)

```
function [mew, U] = pca(f)
        mew = double.empty;
        U = double.empty;
        chunk = 1600;
        for i = 1:chunk:16000
                Xj = f(:, i:i+chunk-1);
                mew = [mew mean((Xj')) '];
                A = Xj - mew(:, int16(i / 1600) + 1);
                [Uj, \~{}] = eigs(((A * A') / 1600), 29);
                U = [U Uj];
        end
end
```

2.2.2 Classification(classify.m)

```
function [correct, incorrect] = classify(T, l, U, mew)
        correct = 0;
        incorrect = 0;
        chunk = 1000;
        for k = 1:chunk:10000
                tic
                for i = k:k+chunk-1
                        x = T(:, i);
                        ej = double.empty;
                        for j=1:10
                                Uq_j = U(:, (j-1)*29+1:j*29);
                                mew_j = mew(:, j);
                                Fj = Uq_j' * (x - mew_j);
                                x_head = Uq_j * Fj + mew_j;
                                ej = [ej norm(x - x_head)];
                        end
                        [~, class] = min(ej);
                        if (l(i) == class-1)
                                correct = correct + 1;
                        else
                                incorrect = incorrect + 1;
                        end
                end
                fprintf(''CPU time of %d - %d: %f\n", k, i, toc);
        end
end
```

2.2.3 Main(main.m)

```
clc;
clear all;
close all;
f = load('/home/alvi/Documents/courses/ece503/labs/1/data/X1600.mat');
t = load('/home/alvi/Documents/courses/ece503/labs/1/data/Te28.mat');
l = load('/home/alvi/Documents/courses/ece503/labs/1/data/Lte28.mat');
f = f.X1600;
t = t.Te28;
l = l.Lte28;
[mew, U] = pca(f);
[correct, incorrect] = classify(t, l, U, mew);
fprintf(''Accuracy: Correct: %d, Incorrect: %d\n", correct, incorrect);
```

2.3 Result

```
CPU time of 1 - 1000: 0.572029
CPU time of 1001 - 2000: 0.592969
CPU time of 2001 - 3000: 0.492402
CPU time of 3001 - 4000: 0.501142
CPU time of 4001 - 5000: 0.492034
CPU time of 5001 - 6000: 0.491773
CPU time of 6001 - 7000: 0.492966
CPU time of 7001 - 8000: 0.499808
CPU time of 8001 - 9000: 0.494592
CPU time of 9001 - 10000: 0.492901
Accuracy: Correct: 9594, Incorrect: 406
```

3 Discussion

Initially the training dataset structure is $D = (x_n, y_n), n = 1, 2, ..., N$ with N = 16,000, where, for each digit x_n , a label is chosen from $y_n in 0, 1, ..., 9$ to match what x_n represents. Originally each

x_n is a gray-scale digital image of 28×28 pixels, with components in the range [0, 1] with 0 and 1 denoting most white and most black pixels, respectively. In this experiment, each x_n has been converted into a column vector of dimension d = 784 by stacking matrix x_n column by column. In this way, each digit from train dataset can be regarded as a "point" in the 784-dimensional Euclidean space. If we put the entire training data together, column by column, to form a matrix $X = [x_1 x2...xm]$, then m = 16,000 and X has a size of 784×60000. After employing PCA to the training dataset, we get μ with dimension of 784×10 and the principal components of the dataset is reduced to 784×290. This is a significant reduction of the dimension of the dataset. The result is also very encouraging. Every chunk of 1,000 image takes on an average of 0.5 sec to classify. Also the accuracy rate is quite high detecting 9,594 digit correctly and 406 incorrect classification.

4 Conclusion

From our analysis we can see that PCA and Euclidean distance work well for the recognition of the hand digit characters. But the PCA was initially introduced for dimension reduction. We can see from our experiment that the dimension of our train dataset in reduced a great deal using PCA and it contributes to the high speed execution of our program resulting in under 1 sec per 1,000 characters. This result justifies our choice of using PCA as the feature extraction technique. There are a lot to be done in term of classifying the dataset. The accuracy of our analysis can be higher with some other classification techniques such as Support Vector Machine(SVM).