CS 423

Sudoku Digit Detection Algorithm Report

Burak Deniz S010031

Introduction:

In this assignment, there are 200 pictures that consist of the sudoku puzzles that are taken from various newspapers and their data where each consists of 9 X 9 matrix and represents the digits inside the sudoku puzzle. The goal of this assignment is to recognize the digits inside each sudoku box and construct a 9 X 9 matrix to compare against the data of the picture and calculate the accuracy of the algorithm. Technologies available for completing these tasks are
Python 3.x, OpenCV, NumPy, MNIST dataset, and the default packages of python.

Problem Statement:

There are a total of 200 pictures of sudoku puzzles in the dataset. These pictures have different sharpness levels, they are taken from different angles, they have different sizes. The goal is to find each digit inside the sudoku puzzle and compare these digits against the sudoku data to construct an accuracy percentage.

Proposed Solution:

To solve the digit detection problem I used the MNIST dataset and a PCA function that I built and the K-nearest neighbor classification method. In this function, I will explain how the PCA function and K-nearest neighbor classification method works from a mathematical point of view.

The Definition of the PCA:

Given a collection of points in two, three, or higher-dimensional space, a “best fitting “ line can be defined as one that minimizes the average squared distance from a point to the line. The next best-fitting line can be similarly chosen from directions perpendicular to the first. Repeating this process yields an orthogonal basis in which different individual dimensions of the data are uncorrelated. These basis vectors are called principal components and several related procedures principal component analysis

In order to calculate the principal components and apply principal component analysis to any given data, we apply the following steps to the data that we have.

STEP 1 - Normalize the Data: Subtract the mean from each of the data dimensions. This is done by taking the mean of the data set and subtracting that mean from each of the data dimensions. Subtracting the mean makes variance and covariance calculation easier by simplifying their equations.

STEP 2 - Calculate the covariance matrix: After normalizing the data I calculated the covariance matrix to calculate the covariance matrix I used the following formula

$$\mathrm{K}_{\mathbf{XX}} = \operatorname{cov}[\mathbf{X}, \mathbf{X}] = \mathrm{E}[(\mathbf{X} - \mu_{\mathbf{X}})(\mathbf{X} - \mu_{\mathbf{X}})^{\mathrm{T}}] = \mathrm{E}[\mathbf{X}\mathbf{X}^{T}] - \mu_{\mathbf{X}}{\mu_{\mathbf{X}}}^{T}$$

With this formula, created a square matrix giving the covariance between each pair of elements of a given vector. When applied to our MNIST train data set consist of 60 thousand 28 X 28 (60K X 784) pictures of handwritten numbers the covariance matrix is shaped as a 784 X 784 matrix.

STEP 3 - Calculate the eigenvectors and corresponding eigenvalues of the covariance matrix: After creating a covariance matrix we have to calculate the eigenvalues and eigenvector to calculate these two I used the following definition and the formula of the eigenvector and eigenvalue.

If T is a linear transformation from a vector space V over a field F into itself and v is a nonzero vector in V, then v is an eigenvector of T if T(v) is a scalar multiple of v. This can be written as :

$$T(\mathbf{v}) = \lambda\mathbf{v},$$

Where λ is a scalar in F, know as the eigenvalue associated with v.

STEP 4 - Reduce the dimensionality and form a feature vector: After calculating the eigenvalues and their corresponding eigenvectors we sort eigenvectors according to their eigenvalues, highest to lowest. This gives us the components in order of significance.
After this, we take N many eigenvectors until variance is higher than 0.95.

In the example of the MNIST train set which is 60000X 784 matrix to get a variance that is higher than 0.95, we have to take 154 eigenvectors which means that feature vector is 154 X 784.

STEP 5 - Deriving the new
After obtaining the feature data we create a RowFeatureVector which consists of eigenvectors in the rows of the matrix where the most significant eigenvector at the top.

Then we create RowZeroMeanData which is the mean adjusted data transposed

After calculating both RowZeroMeanData and RowFeatureVector we find the final data with the following formula

FinalData = RowFeatureVector x RowZeroMeanData

When these 5 steps are applied to any data we get the PCA

Usage of the PCA in the algorithm: In the algorithm, we put the train-dataset from MNIST to the PCA function to create a matrix consist of feature vectors this means that the algorithm will return the result of step 4. After obtaining the matrix we apply the final step to both train-dataset and test-dataset with the same feature matrix to project them into the same space. After applying the final step to both data set we have 2 separate final data which is :

train_images : 60000 X 154
test_images : 10000 X 154

After the application of these step we use the K- nearest neighbor classification method to classify the test images in the following steps, I will explain how the K-nearest neighbor method works from a mathematical point of view.

KNN: In pattern recognition, the k-nearest neighbor algorithm is a non-parametric method used for classification. In k-NN classification, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors

Usage of KNN in the algorithm:

After obtaining the reduced dimensions data sets from PCA which are:

train_images : 60000 X 154
test_images : 10000 X 154

I added classes(labels) to train images as 155’th collum which makes train images data set’s dimensions to 60000 X 155. After that, we take one row from test_images and add zero as 155’th collum which makes the dimensions of the row we took 1 X 155. As a next step, we

calculate the euclidian distances between the row we took from test_images and each row of the train_images with the following formula.

$$d(\mathbf{p}, \mathbf{q}) = d(\mathbf{q}, \mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

Where q is the point from tran_image row and p is the point from test_image row. After calculating the distance between test_image row and each row of the train_image we choose the k nearest neighbors of the test_image row which means that we take k of the closest train_images rows to test_image row. After obtaining k nearest neighbors we take the most repeated neighbor's class and assign that class with the test_row we took at the beginning. We repeat this process for each row in test_images.

After obtaining the classes from the k-nearest neighbor algorithm we compare the classes we obtained with the true classes of the test dataset of MNIST. You can see the comparison results in the result section

After evaluation of the MNIST is done we continue with the sudoku digit detection part of the assignment.

In this part, the first task is to obtain each box in the sudoku grid from the sudoku images. To do that I modified the functions from assignment one to give me each box of sudoku grid as two points top left and bottom right. After obtaining each box algorithm traverse over each box and does the following

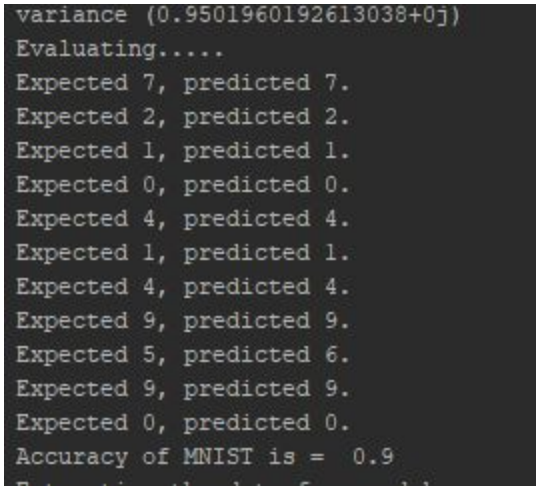
1 - Crop that part of the image which we obtain a box from the sudoku image
2- Put the box image to adaptive thresholding and bitwise not so that we obtain a black background and a white-colored digit if it contains one
3 - Resize the box image to 28 X 28 so we can project the box image and train_image into the same space
4- Take the mean of the pixels in the image if it is lower then 60 that means that box is most probably empty and append 0 to the result matrix
5- If the mean is more then 60 resize the image in to shape 1,784
6 - Apply the common PCA which we obtained from PCA function to obtain a 1 X 154 data of the image
7 - Predict its classification with KNN as we did to one row of test_images
8 - Append the class obtained from KNN to result-matrix

When this process applied to each rectangle of the sudoku we obtain a result matrix (9 X 9) we take the transpose of this matrix because our rectangles are ordered from top to bottom first and the transpose of the result matrix gives us the correct order.

After obtaining the result matrix of one image we compare the result matrix to data files of the sudoku images and calculate the accuracy. We apply this process to every image in the images folder.
You can see the comparison result and the confusion matrix of the first image in the result section as well as cumulative accuracy.

Results:

MNIST evaluation:
For the first ten images in the test_images, you can see the result and the confusion matrix below.



On the expected side of the images, numbers are taken from the original labels of the test images. On the predicted side of the images, numbers are taken from prediction through KNN

Confusion Matrix	True Positive	True Negative
Predicted Positive	9	0
Predicted Negative	1	0

Accuracy Rates:

The accuracy rate of the first 100 data: 0. 92
The accuracy rate of the first 1000 data: 0.918
The accuracy rate of all the data: 0.90

These accuracy rates also show us the success of the PCA function which is around 90 percent.

Sudoku Digit Detection Algorithm Comparison:3

Data From the image1.dat file:

[0 0 0 7 0 0 0 8 0]
 [0 9 0 0 0 3 1 0 0]
 [0 0 6 8 0 5 0 7 0]
 [0 2 0 6 0 0 0 4 9]
 [0 0 0 2 0 0 0 5 0]
 [0 0 8 0 4 0 0 0 7]
 [0 0 0 9 0 0 0 3 0]
 [3 7 0 0 0 0 0 0 6]
 [1 0 5 0 0 4 0 0 0]

Data From digit recognition for image1:

[0 0 0 1 0 1 0 2 0]
 [0 6 0 0 0 1 6 0 0]
 [0 0 6 6 0 6 0 2 0]
 [0 2 0 6 0 7 0 5 4]
 [0 0 0 2 0 0 0 5 0]
 [0 0 7 5 5 5 5 0 5]
 [0 0 0 5 0 0 0 3 0]
 [9 7 0 0 0 0 0 0 3]
 [0 0 5 0 0 7 0 0 0]

Accuracy of image 1 = 71.6

Confusion Matrix Image 1	True Positive	True Negative
Predicted Positive	25	5
Predicted Negative	2	49

Cumulative Accuracy of the sudoku digit detection algorithm: 51.1