CS-423 Assingment-2 Report

1. PCA (Principal Component Analysis)

1. Description

The fundamental reasons for a PCA are the examination of information to recognize examples and discovering examples to diminish the elements of the dataset with negligible loss of data.
our ideal result of the PCA is to extend an element space (our dataset comprising of n d-dimensional elements) onto a smaller subspace that speaks of our information well. A potential application would be a pattern classification task, where we need to lessen the computational expenses and the mistake of parameter estimation by decreasing the quantity of measurements of our element space by removing a subspace that depicts our information best.

2. Steps

i. Compute the d-dimensional mean vector

$$\bar{x}_j = \frac{1}{N}\sum_{i=1}^{N} x_{ij}, \quad j = 1, \ldots, K.$$

ii. Compute covariance matrix

$$cov(X,Y) = \frac{1}{n-1}\sum_{i=1}^{n} (Xi - \bar{x})(Yi - \bar{y})$$

iii. Compute eigenvectors and corresponding eigenvalues
   1. Let A be a square matrix, ν a vector and λ a scalar that satisfies Aν = λν, then λ is called eigenvalue associated with eigenvector ν of A.

$$\det(A - \lambda I) = 0$$

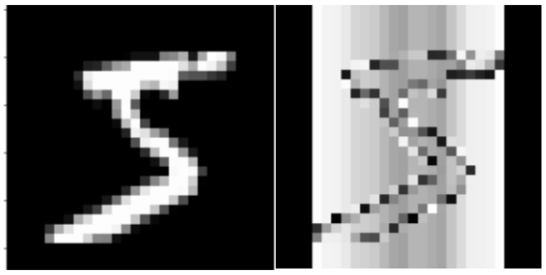
iv. Sort the eigenvectors by decreasing eigenvalues choose k eigenvectors with the largest eigenvalues to form a d * k dimensional matrix W

v. Use this W to form to transform samples onto a new subspace. In the equation below W is eigenvectors, x is data and y is the new coordinates.

$$y = W' \times x$$

3. In our case

i. Taking mean of an image [2d array, along columns]



ii. This image is (28 * 28) pixels taking covariance of this matrix results in a 28 * 28 covariance matrix
iii. Use this covariance matrix to find eigen values and eigen vectors
iv. Choose largest eigenvalues to sort eigenvectors and use best ones
v. Project the data onto the new subspace to find new coordinates

2. K Nearest Neighbors Classification

1. Description

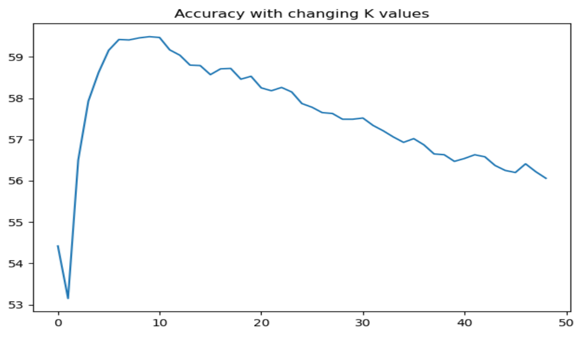
In k nearest neighbors' classification, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors. KNN uses Euclidian distance to calculate k neighbors.

2. Steps

i. Determine parameter K
ii. Calculate the Euclidian distance between the instance and all training samples
iii. Sort the distance and determine k neighbors
iv. Gather the category of the k neighbors
v. According to the majority vote put the instance in that category

3. Our example

i. Accuracy of MNIST dataset with changing values of K



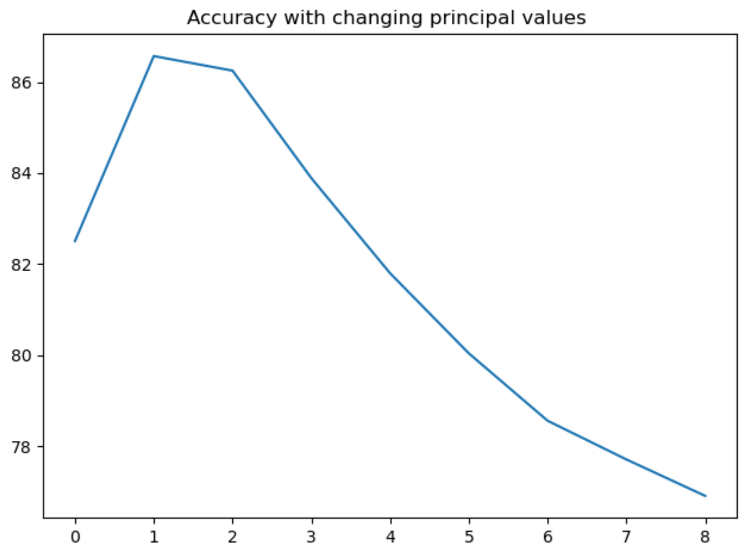
3. MNIST Dataset

1. Description

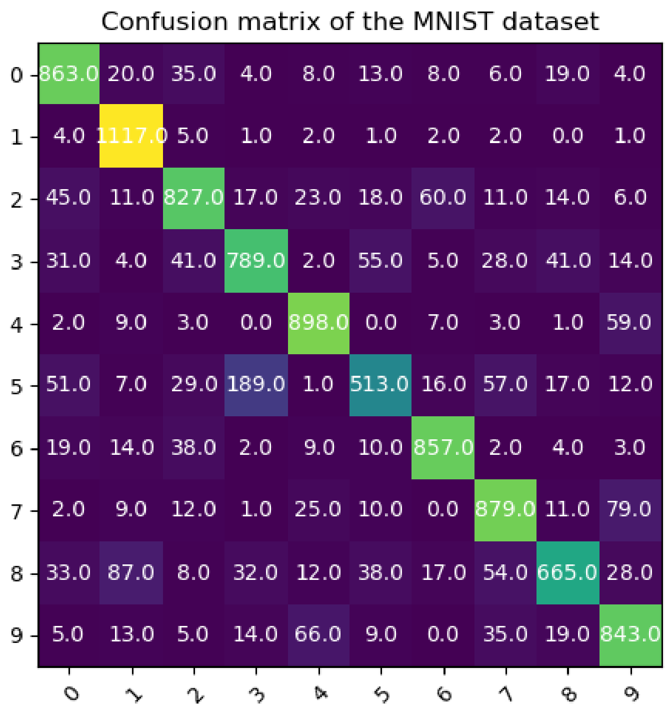
The MNIST database of handwritten digits, has a training set of 60,000 examples, and a test set of 10,000 examples. The digits have been size-normalized and centered in a fixed-size image.

2. Tests

i. **Changing number of principle components correlation to accuracy (index = 0 => 1 principal component)**

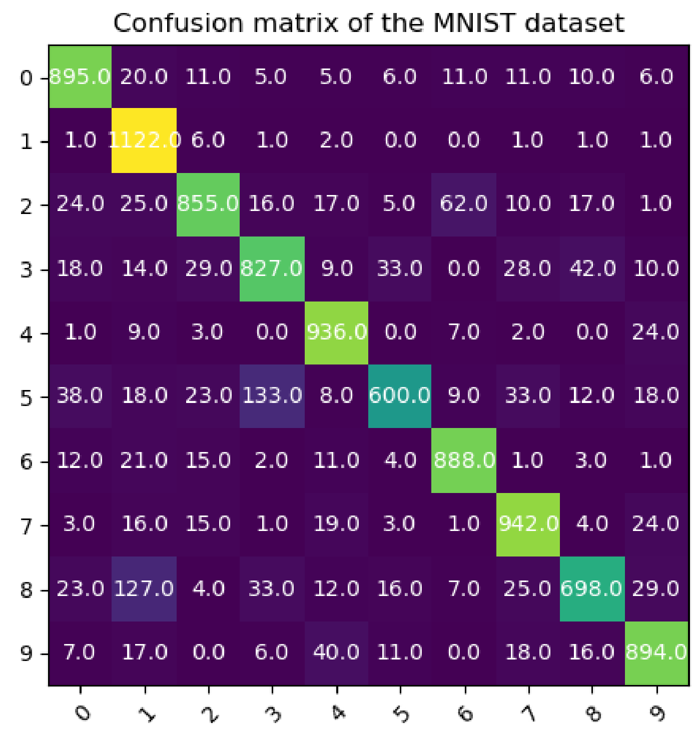


ii. **Confusion matrix (k=10, principal component=1)**


Confusion matrix of the MNIST dataset

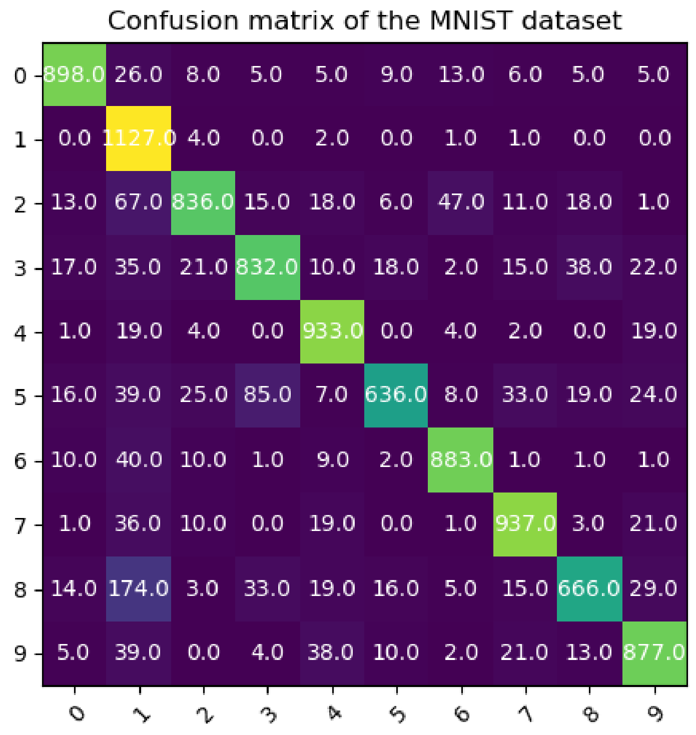
	0	1	2	3	4	5	6	7	8	9
0	863.0	20.0	35.0	4.0	8.0	13.0	8.0	6.0	19.0	4.0
1	4.0	1117.0	5.0	1.0	2.0	1.0	2.0	2.0	0.0	1.0
2	45.0	11.0	827.0	17.0	23.0	18.0	60.0	11.0	14.0	6.0
3	31.0	4.0	41.0	789.0	2.0	55.0	5.0	28.0	41.0	14.0
4	2.0	9.0	3.0	0.0	898.0	0.0	7.0	3.0	1.0	59.0
5	51.0	7.0	29.0	189.0	1.0	513.0	16.0	57.0	17.0	12.0
6	19.0	14.0	38.0	2.0	9.0	10.0	857.0	2.0	4.0	3.0
7	2.0	9.0	12.0	1.0	25.0	10.0	0.0	879.0	11.0	79.0
8	33.0	87.0	8.0	32.0	12.0	38.0	17.0	54.0	665.0	28.0
9	5.0	13.0	5.0	14.0	66.0	9.0	0.0	35.0	19.0	843.0

iii. **Confusion Matrix(k=10, Principal components= 2) Max accuracy achieved (%86.57)**


Confusion matrix of the MNIST dataset

	0	1	2	3	4	5	6	7	8	9
0	895.0	20.0	11.0	5.0	5.0	6.0	11.0	11.0	10.0	6.0
1	1.0	1122.0	6.0	1.0	2.0	0.0	0.0	1.0	1.0	1.0
2	24.0	25.0	855.0	16.0	17.0	5.0	62.0	10.0	17.0	1.0
3	18.0	14.0	29.0	827.0	9.0	33.0	0.0	28.0	42.0	10.0
4	1.0	9.0	3.0	0.0	936.0	0.0	7.0	2.0	0.0	24.0
5	38.0	18.0	23.0	133.0	8.0	600.0	9.0	33.0	12.0	18.0
6	12.0	21.0	15.0	2.0	11.0	4.0	888.0	1.0	3.0	1.0
7	3.0	16.0	15.0	1.0	19.0	3.0	1.0	942.0	4.0	24.0
8	23.0	127.0	4.0	33.0	12.0	16.0	7.0	25.0	698.0	29.0
9	7.0	17.0	0.0	6.0	40.0	11.0	0.0	18.0	16.0	894.0

iv. **Confusion Matrix(k=10, Principal components= 3)**


Confusion matrix of the MNIST dataset

	0	1	2	3	4	5	6	7	8	9
0	898.0	26.0	8.0	5.0	5.0	9.0	13.0	6.0	5.0	5.0
1	0.0	1127.0	4.0	0.0	2.0	0.0	1.0	1.0	0.0	0.0
2	13.0	67.0	836.0	15.0	18.0	6.0	47.0	11.0	18.0	1.0
3	17.0	35.0	21.0	832.0	10.0	18.0	2.0	15.0	38.0	22.0
4	1.0	19.0	4.0	0.0	933.0	0.0	4.0	2.0	0.0	19.0
5	16.0	39.0	25.0	85.0	7.0	636.0	8.0	33.0	19.0	24.0
6	10.0	40.0	10.0	1.0	9.0	2.0	883.0	1.0	1.0	1.0
7	1.0	36.0	10.0	0.0	19.0	0.0	1.0	937.0	3.0	21.0
8	14.0	174.0	3.0	33.0	19.0	16.0	5.0	15.0	666.0	29.0
9	5.0	39.0	0.0	4.0	38.0	10.0	2.0	21.0	13.0	877.0

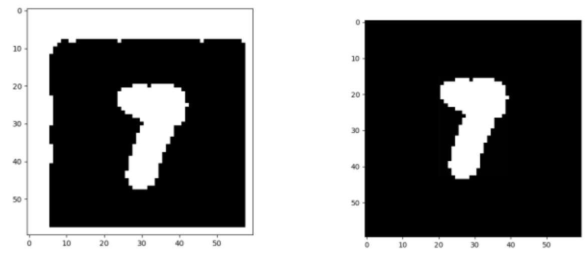
4. Sudoku Dataset

1. Process

i. Get sudoku grid (from assignment 1)

ii. Extract 81 cells

iii. Preprocess cells to erase Hough lines which confuses digit recognition. Example input output image as follows



iv. Resize cells to (28, 28) to standardize to MNIST dataset image resolution

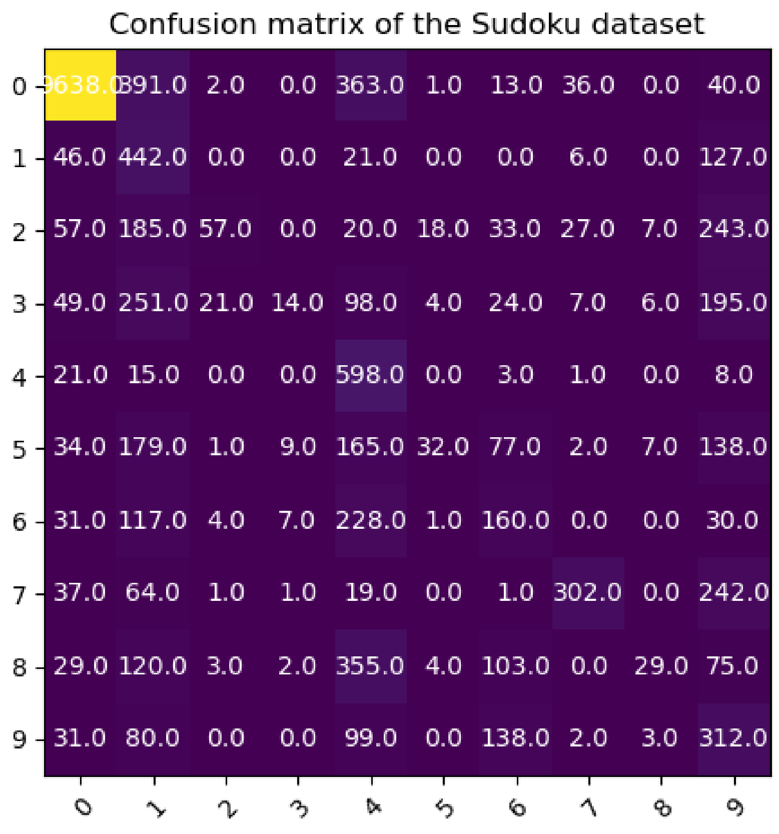
v. Use the algorithm trained on MNIST to and k nearest neighbor classifier evaluate digit

vi. If the white cells are less than threshold (50 pixels) evaluate as empty (which is defined as 0)

vii. Reshape results (9, 9) to compare with actual data

2. Confusion Matrix of Sudoku Dataset

i.



Confusion matrix of the Sudoku dataset

	0	1	2	3	4	5	6	7	8	9
0	9638.0	391.0	2.0	0.0	363.0	1.0	13.0	36.0	0.0	40.0
1	46.0	442.0	0.0	0.0	21.0	0.0	0.0	6.0	0.0	127.0
2	57.0	185.0	57.0	0.0	20.0	18.0	33.0	27.0	7.0	243.0
3	49.0	251.0	21.0	14.0	98.0	4.0	24.0	7.0	6.0	195.0
4	21.0	15.0	0.0	0.0	598.0	0.0	3.0	1.0	0.0	8.0
5	34.0	179.0	1.0	9.0	165.0	32.0	77.0	2.0	7.0	138.0
6	31.0	117.0	4.0	7.0	228.0	1.0	160.0	0.0	0.0	30.0
7	37.0	64.0	1.0	1.0	19.0	0.0	1.0	302.0	0.0	242.0
8	29.0	120.0	3.0	2.0	355.0	4.0	103.0	0.0	29.0	75.0
9	31.0	80.0	0.0	0.0	99.0	0.0	138.0	2.0	3.0	312.0