MP2: Principal Component Analysis (PCA)

K-Nearest Neighbors (KNN)

ECE 417 Fall 2017

Weicheng Jiang

Yuchen Liang

Zixu Zhang

October 10, 2017

1 Introduction

In this MP, the experimented feature vectors are the ones obtained from principal component analysis (PCA). The results are compared to those obtained from raw image pixels and from random orthogonal vectors (random projection). In the result, PCA feature vectors appear far more effective than either of raw pixels or random projection. The intuitive reason behind the scene is that PCA produces uncorrelated feature vectors that align in the direction of those main patterns of the image, while the raw pixel vectors have high correlation that make the underlying image patterns hard to decompose, and the random projection vectors, though have no correlation, make incorrect guesses of the underlying main patterns. The classifying algorithm used in the MP is k-nearest-neighbor (kNN), which is not exactly a learning algorithm because it memorizes all of the (projected) images. It learns the position of the training images in the projected space and classify the test data as the closest k neighbors in the space.

2 Method

2.1 Principal Component Analysis (PCA)

The file `calc_PCA.m` contains all of the calculation needed for PCA. Specifically, line 4 to 13 describes the process to obtain the PCA basis vectors for the set of images. The mathematical deduction is below.

Given a set of M data vectors, $\mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_M} \in \mathbb{R}^{N\times 1}$, which we refer as observation vectors. Therefore, we can use an unbiased estimator to obtain the sample covariance matrix as:

$$\Sigma = \frac{1}{M-1}\sum_{i=1}^{M}(\mathbf{x_i}. - \bar{\mathbf{x}})(\mathbf{x}_i. - \bar{\mathbf{x}})^T \tag{1}$$

Then, we find the eigen-decomposition of Σ (in order of descending eigenvalues) to find the orthogonal eigenvectors matrix $U \in \mathbb{R}^{6300\times 6300}$, whereby the PCA of each observation vector $\mathbf{x_i}$ as:

$$\mathbf{y_i} = U^T(\mathbf{x_i} - \bar{\mathbf{x}}) \tag{2}$$

However, only the first 6300×80 submatrix of U is valuable for the reason below. In this MP, since there are at most 80 samples each with at most 6300 dimensions, only the first 80 eigenvectors will form a projection space, while the rest 6220 eigenvectors are from the null space.

To enhance efficiency, we utilize the following mathematical trick in our calculation. We first find the Gram

matrix:

$$\Gamma = \frac{1}{M-1}\sum_{i=1}^{M}(\mathbf{x_i}. - \bar{\mathbf{x}})^T(\mathbf{x}_i. - \bar{\mathbf{x}}) \tag{3}$$

The eigen-decomposition of Γ wil give an orthogonal matrix V and corresponding eigenvalue k_i of each eigenvector. From singular value decomposition (SVD), we claim that the normalized data matrix an be decomposed as

$$\tilde{X} = \frac{1}{\sqrt{M-1}}(\mathbf{x_i} - \bar{\mathbf{x}}) = USV^T \tag{4}$$

where, $\tilde{X} \in \mathbb{R}^{6300\times 80}$, $U \in \mathbb{R}^{6300\times 80}$, $S \in \mathbb{R}^{80\times 80}$, $V \in \mathbb{R}^{80\times 80}$, and S is a diagnoal matrix with $s_{ii} = \sqrt{k_i}$. In this way, it is much easier to obtain the left eigenvector matrix of the data matrix as

$$U = \tilde{X}VS^{-1} \tag{5}$$

A further step to reduce the dimensionality, we can take the top N eigenvectors, whose eigenvalues compose 95% of the energy of PCA (sum of eigenvalues). In this case, our PCA projected data $\curvearrowright_i$ can reduce its dimension from $\mathbb{R}^{80\times 1}$ to $\mathbb{R}^{N\times 1}$.

2.2 k-Nearest Neighbor (kNN)

The files `knn.m` and `findlabel.m` contains all of the calculation for the kNN classifier. It classifies the test data as the same label that appears for its nearest k neighbors. Mathematically, given M training sample each with N dimensions, each with label $l_1, l_2, \cdots, l_b$, denote each sample as $\mathbf{x_i}$ of length N, and define our distance for measurement in line 7 as the Euclidean distance squared

$$d_2(\mathbf{x_i}, \mathbf{x_j})^2 = ||\mathbf{x_i} - \mathbf{x_j}||^2 \tag{6}$$

For a test data $\mathbf{x}_{\text{test}}$, we want to find k indexes, say $a_1, a_2, \cdots, a_k$, that give the k smallest $d_2(\mathbf{x}_{\text{test}}, \mathbf{x_i})^2$ value, where $1 \le i \le M$. Then we label $\mathbf{x}_{\text{test}}$ as the label l_m that appear the most times in the label set $\{l(\mathbf{x_{a_1}}), l(\mathbf{x_{a_2}}), \cdots, l(\mathbf{x_{a_k}})\}$, where $l(\mathbf{x_i})$ represents the label of $\mathbf{x_i}$. This operation is done in `findlabel.m`, which is invoked in line 9 to 13 in `knn.m`.

3 Reuslts

The table 1 shows the accuracy for 1-NN search with set of 80 data samples provided, and the table 2 shows the accuracy for 5-NN test.

Table 1: 1-Nearest Neighbor Search Accuracy Table

%	Raw $[70 \times 90]$	Raw $[35 \times 45]$	Raw $[17 \times 22]$	Raw $[7 \times 9]$	PCA 95%	Random
A	100	100	100	100	100	100
B	75	75	80	100	75	75
C	80	85	90	95	80	60
D	100	100	100	100	100	85
Ave	88.75	90	92.5	98.75	88.75	80

Table 2: 5-Nearest Neighbor Search Accuracy Table

%	Raw $[70 \times 90]$	Raw $[35 \times 45]$	Raw $[17 \times 22]$	Raw $[7 \times 9]$	PCA 95%	Random
A	100	100	100	100	100	100
B	80	80	85	90	70	60
C	45	45	45	55	40	45
D	95	90	95	85	95	85
Ave	80	78.75	81.25	82.5	76.25	72.5

The table 3 shows the effects of PCA energy on the accuracy of 1-NN and 5-NN search.

Table 3: 1-NN and 5-NN Search Accuracy Table for PCA with Different Energy

%	1-NN			5-NN		
	PCA 90%	PCA 95%	PCA 98%	PCA 90%	PCA 95%	PCA 98%
A	100	100	100	100	100	100
B	75	75	75	55	70	75
C	80	80	80	35	40	40
D	100	100	100	100	95	95
Ave	88.75	88.75	88.75	72.5	76.25	77.5

The table 4 and 5 shows accuracy of 1-NN and 5-NN search, after 10 trails for random projection matrix.

Table 4: 1-Nearest Neighbor Search Accuracy Table for 10 Random Projection Matrix

%	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7	Run 8	Run 9	Run 10	Ave
A	100	100	95	100	100	100	95	90	100	100	98
B	80	65	85	75	60	75	60	80	75	75	73
C	75	60	70	70	80	75	70	95	85	70	75
D	95	85	90	90	90	95	75	90	80	75	86.5
Ave	87.5	77.5	85	83.75	82.5	86.25	75	88.75	85	80	83.125

Table 5: 5-Nearest Neighbor Search Accuracy Table for 10 Random Projection Matrix

%	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7	Run 8	Run 9	Run 10	Ave
A	90	95	100	100	95	100	85	85	100	100	95
B	70	60	75	75	50	80	60	70	70	55	66.5
C	35	40	25	60	30	60	40	30	35	30	38.5
D	95	75	95	85	90	85	80	85	70	80	84
Ave	72.5	67.5	73.75	80	66.25	81.25	66.25	67.5	68.75	66.25	71

4 Discussion

In the result, different features provide different accuracies. The random projection features have the lowest average accuracies for both 1-NN and 5-NN, because the random guess in the underlying patterns to be projected on can easily go wrong. The reason that the PCA (95% energy) has lower accuracy than raw pixels for both classifiers is that the test data is too similar to several parts of the training data that over-fitting occurs. Within each classifier, the average accuracy decreases as raw pixel size increases, because additional dimensionality in the data make the data points sparse and thus hard to classify (i.e. curse of dimensionality).

Different classifiers also result in different accuracies. For any feature, 1-NN performs better than 5-NN, due to the fact that our data set is so small (only 19-20 samples for a label) that the second through fifth nearest neighbor introduces unnecessary classification errors.

The increase in PCA energy has a positive effect on average accuracies using 5-NN, while it has totally no effect when using 1-NN. This is because the given data set is so similar with such a small size, that the change in PCA energy (i.e. relevant to loss of dimensions) does not change the top closest training data, which is already really close due to over-fitting, but gives a higher chance for other training data of the same face to reach the second through fifth closest positions.