Face Recognition Using PCA, LDA and LPP

1. Introduction

Face recognition is a very easy task for human being. However, it is a rather hard task for computer to classify faces correctly. Many face recognition techniques have been developed over the past few decades, and they can be divided into two groups based on how they represent faces [1]:

1. Appearance-based, which applies holistic texture features to either whole –face or specific regions in face image.
2. Feature-based, which uses geometric facial features and geometric relationships.

Among many techniques which are appearance-based, two popular techniques for this purpose are Principal Component Analysis and Linear Discriminant Analysis. PCA is a dimension reduction method which projects n dimensional data onto k dimensional subspace where k is smaller than n, and the k dimensional subspace is defined by leading eigenvectors of the data's covariance matrix. In 1991, Turk and Pentland achieved face recognition by projecting face images onto a feature space called "eigenfaces" through PCA [2]. LDA, on the other hand, is a supervised learning algorithm. The aim of LDA is to find a linear combination of features which separate different classes of objects. In other words, LDA is supposed to find a subspace on which objects in different classes are far away from each other while requiring objects in the same class are close to each other. In 1997, Belhumeur, Hespanha and Kriegman applied the linear discriminant analysis into face recognition [3].

However, both PCA and LDA only effectively see the Euclidean Structure. They fail to discover the underlying structure which might be a nonlinear submanifold [4]. In 2005, He, Yan, Hu, Niyogi and Zhang proposed a new approach which considers the manifold structure for face analysis through Locality Preserving Projections (LPP) [4]. They used LPP to find the face subspaces which they called as Laplacianfaces to achieve face recognition.

In this project, the above three methods have been successfully implemented. After the system is trained by the training data, the feature space "eigenfaces" through PCA, the feature space "fisherfaces" through LDA and the feature space "laplacianfaces" through LPP are found using respective methods. Later in this report, W is used to represent the obtained feature space. Once W is obtained, training faces are projected to subspace defined by W to construct FaceDB. When an unknown face is needed to recognize, this test face is firstly projected onto subspace W. Afterward, the program finds the K nearest neighbors of the projected data in FaceDB. Finally, the class label is assigned to the test face according to the majority vote among the neighbors. This classification algorithm is known as K-nearest neighbor.

The rest of this report is organized as follows: Section 2, 3, 4 describes PCA, LDA and LPP respectively. KNN algorithm is talked about in Section 5. In Section 6, the whole process of face recognition is discussed. Section 7 presents the design and architecture of this system.

2. Principal Component Analysis (PCA)

Let $X = \{x_1, x_2, \dots, x_n\}$ be the matrix containing n face images. Note that each image matrix has been converted into a vector. For example, a $m * n$ matrix is converted into a vector with $m * n$ rows. The subspace "eigenfaces" could be obtained by following the below steps:

1. Computer the mean μ
$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$
2. Compute the Covariance Matrix S
$$S = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)(x_i - \mu)^T$$
3. Compute the eigenvectors v_i and eigenvalues λ_i
$$Sv_i = \lambda_i v_i$$
4. Order the eigenvectors descending by their eigenvalues and select k eigenvectors corresponding to k largest eigenvalues

The above k eigenvectors form the subspace W, and it is called as "eigenfaces". There is still a computational problem left. For images in ORL database, each image is 92×112 meaning that there are 10304 dimensions. Once such faces are applied to calculate the covariance matrix S, S will end up with a 10304×10304 matrix which is really huge and almost could not be processed on normal laptops. Here we first define Y:

$$Y = X - U$$

where $U = \{\mu, \mu, \dots, \mu\}$. By this way, the original data set is transformed into a data set with zero mean. In order to solve the problem just mentioned, the eigenvectors and eigenvalues of $Y^T Y$ is first calculated:

$$Y^T Y v_i = \lambda_i v_i$$

Then the original eigenvectors are calculated by a left multiplication of the data matrix Y:

$$YY^T(Yv_i) = \lambda_i(Yv_i)$$

The last step is to normalize the eigenvectors just calculated. Finally, the k eigenvectors corresponding to k largest eigenvalues are retrieved and form the subspace W. Also, the column vectors of W are the so-called eigenfaces.

Normally, the number of training samples is far less than the dimensions of each face. As a result, the above method is needed to avoid the computational problem mentioned before.

3. Linear Discriminant Analysis (LDA)

LDA is a supervised learning algorithm meaning that the class labels of training set will be used in the training process. Let X be the matrix containing training face images and X_i be the matrix containing face images belonging to class i.

$$X = \{X_1, X_2, \dots, X_c\}$$

$$X_i = \{x_1, x_2, \dots, x_n\}$$

The scatter matrices S_B and S_w are calculated as follows:

$$S_B = \sum_{i=1}^{c} N_i(\mu_i - \mu)(\mu_i - \mu)^T$$

$$S_w = \sum_{i=1}^{c} \sum_{x_j \epsilon X_i} (x_j - \mu_i)(x_j - \mu_i)^T$$

where N_i represents the number of training samples belonging to class i, μ_i represents the mean of training samples belonging to class label i and μ represents the mean of the total samples.

In order to find a subspace where S_B is maximized and S_w is minimized, the following general eigenvalue problem is needed to be solved.

$$S_B v_i = \lambda_i S_w v_i$$

However, one is confronted with the difficulty that the within-class scatter matrix S_w is always singular. In order to solve this problem, it is suggested that PCA first projects the image set to a lower dimensional space so that the resulting within-class matrix S_w is no longer singular [3]. After this process, the general eigenvalue problem becomes:

$$S_w^{-1} S_B v_i = \lambda_i v_i$$

By solving the above eigenvalue problem, W_{fld} is obtained which defines a subspace. Please note that there are at most $c - 1$ nonzero generalized eigenvalues for the above formula. Also, note that PCA can directly reduce the dimension of the feature space to $N - c$ because the rank of S_w is at most $N - c$. Here N is the number of samples while c is the number of classes.

By combing the transformation matrix of PCA, the below transformation matrix for LDA is obtained:

$$W = W_{pca} W_{fld}$$

The column vectors of W are the so-called fisherfaces.

4. Locality Preserving Projections (LPP)

The steps to perform LPP are as follows:

1. **PCA projection**. The training image set is first projected into the PCA subspace by throwing away several components with small eigenvalues. The reason of PCA projection is to ensure XDX^T is not singular which is helpful to solve a generalized eigenvector problem later on.
2. **Constructing the nearest-neighbor graph**. Let G denote a graph with n nodes. The ith node corresponds to the face image x_i. An edge will be established between

x_i and x_j if x_i and x_j are "close" enough. Here k-nearest neighbor is adopted to check whether two nodes are close to each other or not. For instance, edges will be set between x_i and its k neighbors found by KNN.

3. **Choosing the weights**. Here a simple variation for weighting the edges is adopted. If node i and j are connected, put
$$S_{ij} = 1$$
Otherwise, put $S_{ij} = 0$. Please note that there is also other way to weight the edges which should more accurate.
4. **Eigenmap**. Compute the eigenvectors and eigenvalues for the generalized eigenvector problem:
$$XLX^T w = \lambda XDX^T w$$
where X is the training image set, D is a diagonal matrix whose entries are column sums of S and $L = D - S$.

Once the generalized eigenvalue problem in step 4 is solved, W_{LPP} is obtained. By combing the transformation matrix of PCA, the below transformation matrix for LPP is obtained:

$$W = W_{pca} W_{LPP}$$

The column vectors of W are the so-called Laplacianfaces.

5. K-Nearest Neighbor Classification Algorithm

The k-nearest neighbor algorithm is a method for classifying objects based on closest training objects: an object is classified according to a majority vote of its neighbors. In this project, KNN plays an important role in recognition phase. The steps of performing KNN are as follows:

1. **Find K nearest neighbors.** For a specific test face, it is first projected onto the subspace defined by W, and W is calculated by any methods mentioned above such as PCA, LDA and LPP. Afterward, an array with size K is created to keep the K nearest neighbors, and the way to achieve this goal is to go through the whole training set and update the array if there is a training face whose distance is smaller than the largest distance in the original array. At the end of this process, the K nearest neighbors are found. Also, please notice that the training set has already been projected onto the same subspace defined by W.
2. **Classify according to weights.** After the K nearest neighbors are found, a HashMap is created to keep the pairs <label, weight> by going through all the neighbors. If the HashMap happens to meet a new label, <label, 1 / distance> is directly added into the map. Otherwise, an updated version which adds the original weight with 1 / distance is put into the map. After the HashMap is correctly constructed, this program will go through the whole HashMap and find the label associated with the largest weight. This class label will be the label recognized by this system.

Last but not least, three different metric methods such as cosine similarity, L1 distance and Euclidean distance are also implemented in this project. In Section 8, comparisons between different metric methods will also be discussed.

6. Face Recognition Methodology

After introducing PCA, LDA, LPP and KNN, now it is the time to combine them together to build the face recognition system. The methodology used in this project can be divided into the following three parts:

1. Training

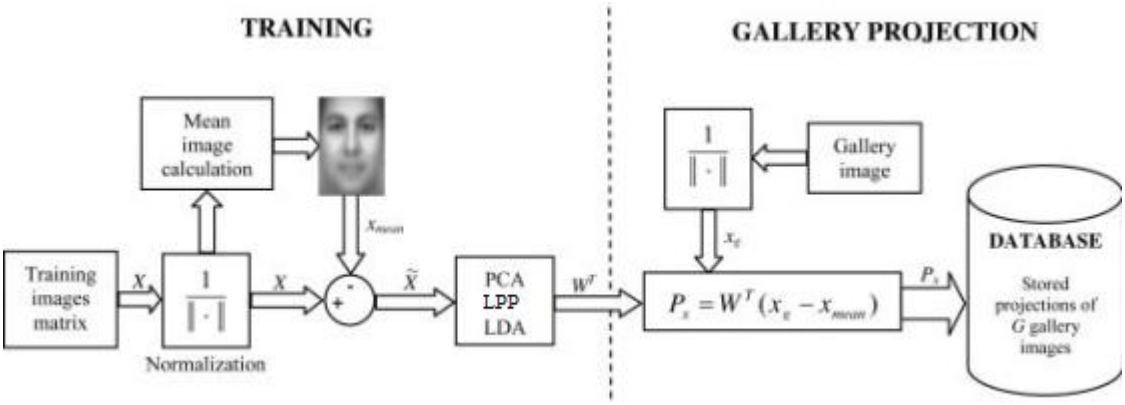


Figure 1. An illustration of the training process [1]

As Figure 1 shows, there are two major tasks: training and gallery projection. For the training part, the training face set is normalized followed by subtracting the mean face. The projection matrix W is then calculated using PCA, LPP or LDA. For the gallery projection, faces with known labels are projected onto the subspace defined by W, and their projections are stored in the database for the next step.

2. Recognizing

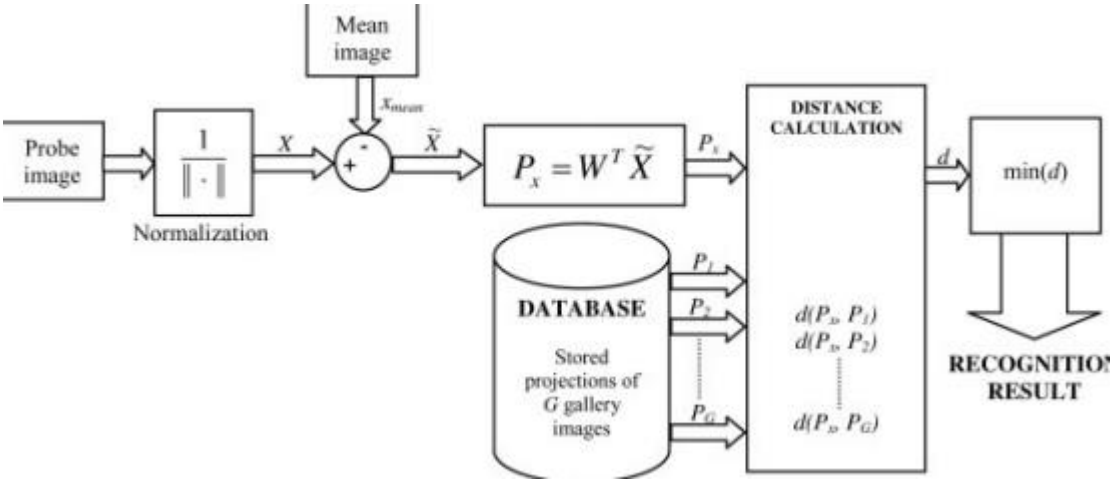


Figure 2. An illustration of the recognizing process [1]

Now it is the time to recognize an unknown face. As Figure 2 illustrates, the probe image is normalized followed by subtracting the mean face of the original training set. Next it is projected onto subspace W, and the projected data P_x is calculated. The last step is find P_x's K nearest neighbors among the database constructed in step 2.

Finally, the label of the probe face is predicted according to the majority vote among the K nearest neighbors.

This is a general description about the face recognition system built in this project. More details will be discussed about in Section 7 Implementation.

7. Implementation

This project is purely written in Java, and it relies on the Java Matrix Package (JAMA), which is an open-source library, for performing matrix operations such as transpose and inverse. Furthermore, this project is designed in an object-oriented manner consisting of the following components:

- Distance Methods

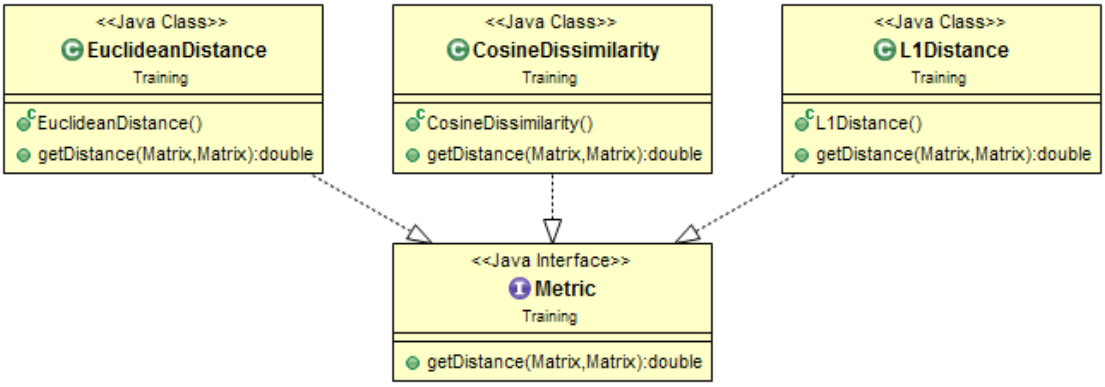


Figure 3. Class diagram which illustrates the metric component

The Metric interface defines the signature of a method called "getDistance", and this method aims to calculate the distance between two matrixes. Classes which implements Metric interface such as EuclideanDistance, CosineDissimilarity and L1Distance have to give different implementations to method "getDistance". The formulas which shows how each method is implemented are listed below:

- EculideanDistance

$$d(\boldsymbol{p}, \boldsymbol{q}) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}$$

- CosineDissimilarity

$$\cos(\boldsymbol{p}, \boldsymbol{q}) = \frac{\boldsymbol{p}^T \boldsymbol{q}}{|\boldsymbol{p}||\boldsymbol{q}|}$$

Because the above formula calculates the similarity between $\boldsymbol{p}$ and $\boldsymbol{q}$, $1/cos(\boldsymbol{p}, \boldsymbol{q})$ is returned in order to be unified with Euclidean distance and L1 distance. The reason why we do this is because that distance should be larger when the cosine similarity becomes smaller. The case when cosine similarity equals to 0 is also taken into consideration. The maximum value defined in double type of Java is returned in this case.

- L1 Distance

$$d(p,q) = \sum_{i=1}^{n} |p_i - q_i|$$

- K-nearest Neighbor

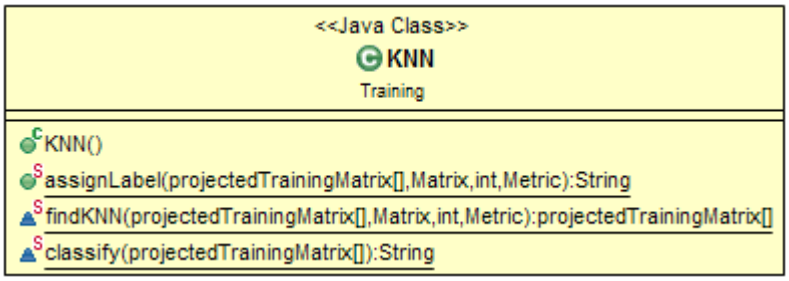


Figure 4. Class diagram illustrating the KNN class

Referring to Section 5, it is obvious to see what this class does. The method "findKNN" finds the K nearest neighbors of a test face using metric designated by the user, and the method "classify" returns a label according to the majority vote of the K nearest neighbors.

- File Manager

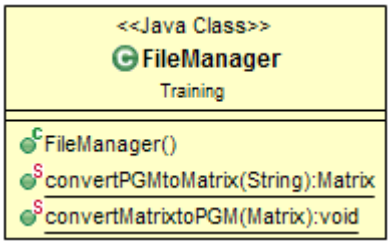


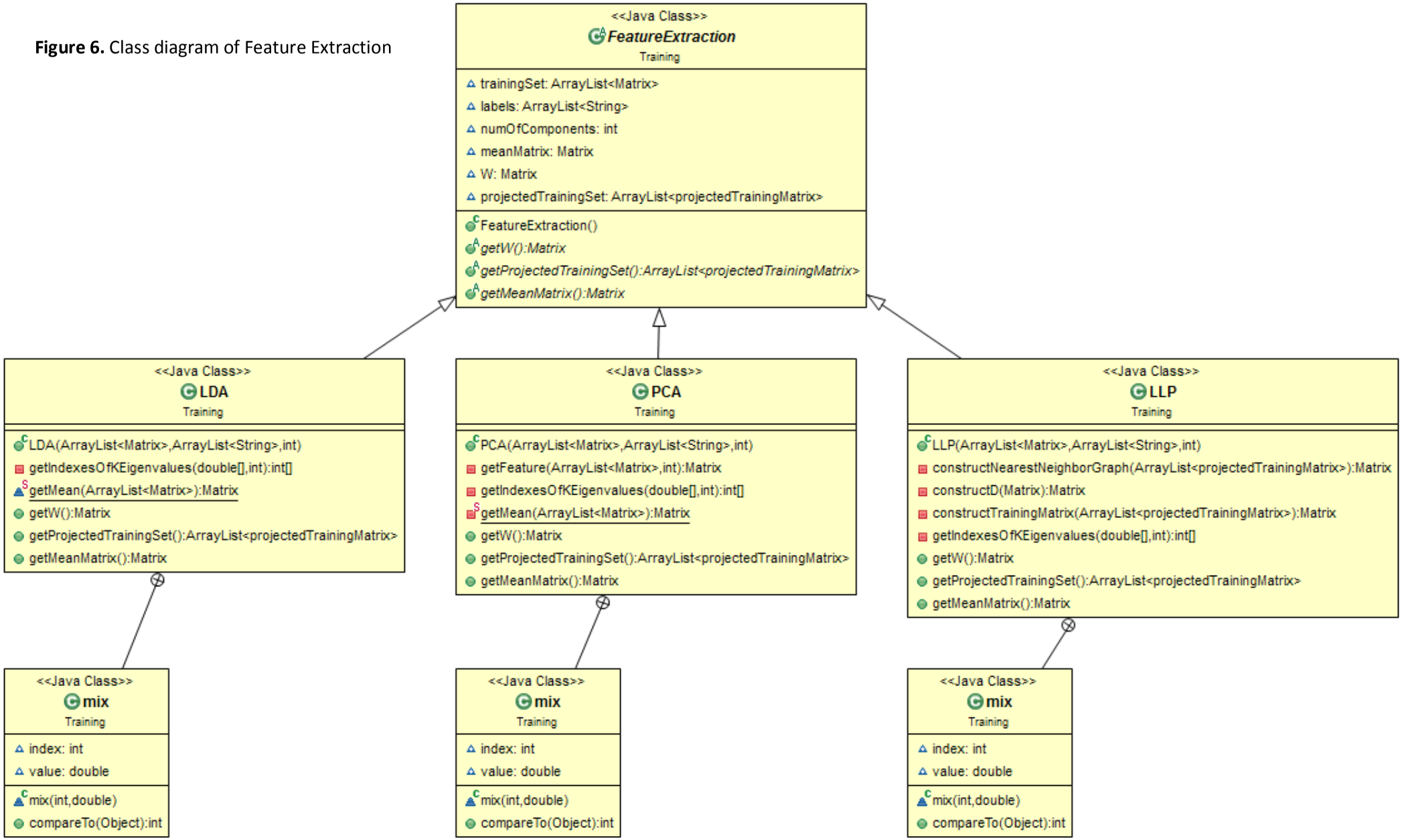
Figure 5. Class diagram of FileManager class

Face images are stored in image files. In order to read face images into this program, this FileManager class is needed to convert PGM image files into matrixes. Furthermore, another method converting matrixes into image files is also needed. For instance, we want to see those eigenfaces, fisherfaces and laplacianfaces visually. In this case, the transformation matrix W will have to be transformed into image files.

- Feature Extraction

This component consists of three different methods to extract features of faces: PCA, LDA and LPP. Because the three methods are both appearance-based, an abstract class FeatureExtraction is designed to encapsulate some common properties of PCA, LDA and LPP. The advantage of doing so is that the type of FeatureExtraction could be determined at runtime which leads to compact structures and codes. Since the details about PCA, LDA and LPP have already been discussed in above sections, there is no need to repeat them again. Below is the class diagram for this component.

Figure 6. Class diagram of Feature Extraction



References

[1] Delac, K., Grgic, M., & Grgic, S. (2005). Independent comparative study of PCA, ICA, and LDA on the FERET data set. *International Journal of Imaging Systems and Technology*, *15*(5), 252-260.

[2] Turk, M., & Pentland, A. (1991). Eigenfaces for recognition. *Journal of cognitive neuroscience*, *3*(1), 71-86.

[3] Belhumeur, P. N., Hespanha, J. P., & Kriegman, D. J. (1997). Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *19*(7), 711-720.

[4] He, X., Yan, S., Hu, Y., Niyogi, P., & Zhang, H. J. (2005). Face recognition using laplacianfaces. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *27*(3), 328-340.