Ziyu Chen
Dec 16th, 2019
Project Advisor: Danhong Song

Image Classification in Machine Learning

Abstract

Image classification is the process of taking an image as input and giving a class or a probability of a class as output. It's a trending topic including face recognition (input a human face picture, for example, and recognize who he/she is), object classification (input a picture of an object and recognize what it is), and many other applications in various regions. In this project, I mainly looked at using various statistical methods in achieving these goals.

Principle component analysis, an unsupervised learning method, is widely used in handling the data, reducing dimensions of the variables involved, and improving time efficiency. By applying PCA in face recognition, we have an algorithm finding Eigenfaces and can express all the face image matrices in a linear combination of them. Then when PCA is used with CNN, a computer science technique used to solve complicated classification problems, it shows that PCA could increase the speed of training, decreasing the time for the first training epoch from 37 seconds by CNN alone to 19 seconds by PCA and CNN, while not losing significant variation of the original data. The results of classification using, Random Forest Classifier, K-nearest neighbors, logistic regression, Support vector machine, CNN, and CNN after PCA were also compared.

Introduction

Artificial Intelligence is trending nowadays, and it has been used in different areas to let a machine, or computer, to mimic human abilities. Machine learning, as a specific subset of AI, trains the machine how to learn from data. Various algorithms have been developed to accomplish this goal. Statistical model and theories are usually applied to help a machine analyzing the data. We refer these as statistical methods used in machine learning. This project is then a secondary research on the statistical applications in image classification.

It is quite notable that Principal Component Analysis (PCA) is introduced in many recent papers as a powerful predictor dimension reduction technique to reduce model complexity and to improve training efficiency. It uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. Regarding its application in image classification, it reduces the dimension of an input image. In face recognition, the goal is to recognize an unknown face, matching it with a known face in the database; we will train a model by finding the principal components of the distribution of faces. This method is usually referred as *eigenface*, which was developed by Sirovich and Kirby (1987) and used by Matthew Turk and Alex Pentland in face classification [1]. In object classification, the goal is to classify an unknown image of objects, putting it to one of the pre-defined categories; we will usually apply PCA to reduce the dimension and then use the results as input for another classification technique, such as convolutional neural network (CNN). I will illustrate the steps in detail in the following paragraphs.

Background

Before explaining the algorithm of PCA in face recognition, I will firstly illustrate how an image is stored in and processed by a computer. This would be the basics of training a machine learning model for images.

Each image, in a computer, is composed by many small squares called "pixels", which is just the color information. For a black and white image, the range for one pixel is usually 0-255, depending on the coding system, where 0 stands for pure white and 255 stands for pure black. We call this the greyscale as all the colors between white and black look like grey. However, face images are usually obtained as colored pictures by today's technique. All the different colors, considered by the computer, are composed from three base colors – red, green, and blue (RGB). Each of the three colors is stored as a number from 0 to 255. Then we have three layers of color information for each pixel. The visual output of a pixel would be a mixture of three colors. Because of the fact that color information does not help identifying the edges and other features, we will convert color to greyscale before training the model. The simplest approach is to obtain the average of the three RGB scores, while studies show that different algorithms for converting color to greyscale could result in different model prediction accuracy. In this project, we will take greyscale pictures as our inputs.

Now that we knew that images are composed by pixels. A latest iPhone 11 Pro can take a picture with 12 million pixels, which indicates there are 4000 pixels in length and 3000 pixels in width. If we treat each pixel as a random variable, we will have too many parameters to deal with. In real scenario, we will compress nearby pixels to reduce the dimension, and then use the much smaller images as the training data. So, in this project, we choose the *Labeled Faces in the Wild* (LFW) [2] as our data set, which provides 400 black and white face images with a dimension of 64*64 pixels. These images are localized and scaled to a common size to help the machine extract common features and distinguish the differences. We divide our sample data to a training set of 300 face images and a testing set of 100 face images.

The PCA application in Eigenface

After the preprocessing showed above, we obtain a face image matrix with dimension N*N, where N = 64 pixels in our case. We denote the matrix as $A = \begin{bmatrix} \square & \cdots & \square \\ \square & \vdots & \square \\ \square & \cdots & \square \end{bmatrix}^{N*N}$. Firstly, we will convert the matrix A to a vector $\boldsymbol{x} = \begin{bmatrix} \square \\ \square \\ \square \\ \square \end{bmatrix}^{N^2*1}$. We can retrieve the image matrix back from the vector easily.

Suppose we have T training images, where T = 300 in our cases, then we put all the face vectors together in an aggregated image matrix $X = [\boldsymbol{x}_1 \quad \boldsymbol{x}_2 \quad \cdots \quad \boldsymbol{x}_T] = \begin{bmatrix} \square & \cdots & \square \\ \square & \vdots & \square \\ \square & \cdots & \square \end{bmatrix}^{N^2*T}$. Matrix X has all the information of the training faces. To concentrate on the differences of faces,

we will compute the mean face and subtract it from each column vector in X to get a new matrix, which I call a normalized image matrix (NIM). The mean face is obtained by simply finding the average face:

$$\boldsymbol{\mu} = \frac{1}{T}\begin{bmatrix} \sum_{i=1}^{T} x_{i1} \\ \sum_{i=1}^{T} x_{i2} \\ \vdots \\ \sum_{i=1}^{T} x_{N^2 1} \end{bmatrix}^{N^2 * 1}.$$

Then, NIM is denoted as

$$M = [\boldsymbol{m}_1 \quad \boldsymbol{m}_2 \quad \cdots \quad \boldsymbol{m}_T]^{N^2 * T}, where\ \boldsymbol{m}_i = \boldsymbol{x}_i - \boldsymbol{\mu}.$$

So, the training model will focus on the deviations of each face from the mean face.

According to linear algebra, we know that we can find bases for NIM by finding eigenvectors of the covariance matrix of M. Also, the first eigenvector is corresponding to the largest eigenvalue. In this way, each eigenvector is obtained in the direction of maximum variation. So, our goal is to find the eigenvectors of the covariance matrix of M. Since NIM is normalized, the means of dimensions in M are 0. Thus, we can get the covariance matrix

$$C = MM^T = \frac{1}{T}\sum_{i=1}^{T} \boldsymbol{m}_i \boldsymbol{m}_i{}^T = [\,]^{N^2 * N^2},$$

which is usually very large, i.e. when n=64, the dimension of C would be 4096*4096. It would be neither practical nor time-efficient to do such a huge computation. In practice, we do not need all the N^2 eigenvectors to cover the whole variation. According to the principle of linear algebra, we will only need the first few eigenvectors of C to span the original space. So, we can take a shortcut.

Usually, we would find the singular value decomposition (SVD) of M, $M = U\Sigma V^*$, where U is the matrix composed by eigenvectors of covariance matrix C of M and U should have a dimension of N^2* N^2. As a mathematical trick, we would consider computing the eigenvectors of $M^T M$ instead of MM^T, so we are dealing with a dimension of T*T instead of N^2* N^2. It is supported by the fact that MM^T and $M^T M$ have the same eigenvalues and their eigenvectors are related. A simple proof is given as: suppose $\boldsymbol{v}$ is an eigenvector of $A^T A$, then $A^T A\boldsymbol{v} = \lambda\boldsymbol{v}$. Multiplying A on both sides, we will get $AA^T A\boldsymbol{v} = \lambda A\boldsymbol{v}$, which indicates that $A\boldsymbol{v}$ is an eigenvector of AA^T. We can find a total number of T eigenvectors for $M^T M$. And since the first few eigenvectors contain the most variation of data, to obtain 90% of the variation in the data, we only need p eigenvectors, where p<T<<N^2. We can achieve the number p by keep adding the corresponding eigenvalues until it reaches 90% of the sum of the eigenvalues. Then we choose the first p eigenvectors of $M^T M$. Putting these eigenvectors to U_p, we obtain the new bases matrix $U_p = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1p} \\ \vdots & \vdots & \vdots & \vdots \\ u_{N^2 1} & u_{N^2 2} & \cdots & u_{N^2 p} \end{bmatrix}^{N^2 * p} = [\boldsymbol{u}_1 \quad \boldsymbol{u}_2 \quad \cdots \quad \boldsymbol{u}_p]$. These eigenvectors can be thought of as a set of features that together characterize the variation between face images. We call these $\boldsymbol{u}_j$ eigenfaces, where $1 \le j \le p$. Each $\boldsymbol{u}_j$ actually looks like a face and contains information of different important features on faces. That is probably the reason why its founder gave it the name eigenface. The first 30 eigenfaces in our training model is shown in figure 1.

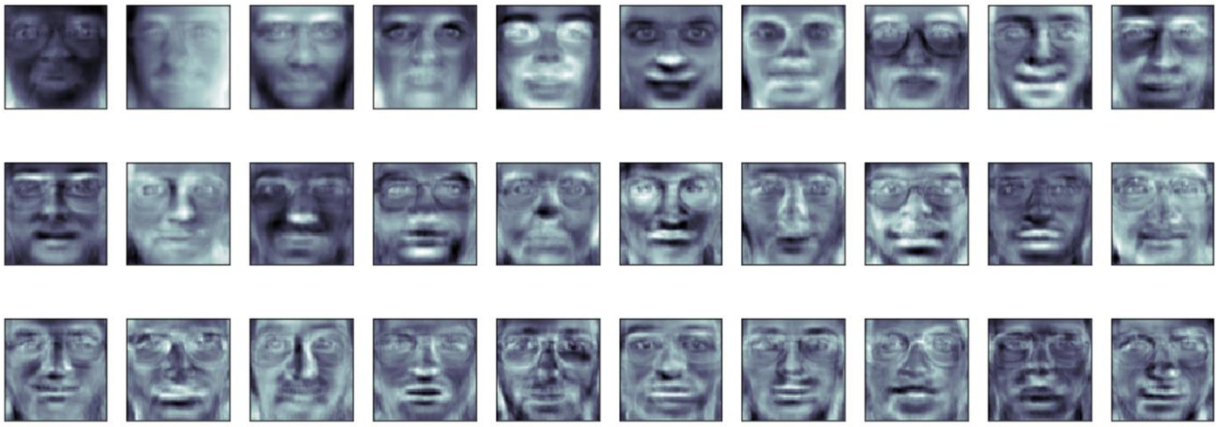

Figure 1

Next, we will compute the principal components of M, which is the NIM projected on a new coordinate given by the new basis U_p. The projection of NIM onto eigenspace is given as $M^* = U_p^T M = \begin{bmatrix} \square & \cdots & \square \\ \square & \vdots & \square \\ \square & \cdots & \square \end{bmatrix}^{p*T} = [\boldsymbol{w}_1 \quad \boldsymbol{w}_2 \quad \cdots \quad \boldsymbol{w}_T]$, where $\boldsymbol{w}_i = \begin{bmatrix} w_{1i} \\ w_{2i} \\ \vdots \\ w_{pi} \end{bmatrix}$. We denote each column as $\boldsymbol{w}_i$ because we will further refer to it as a weight vector. Next, we will express the original NIM with the new coordinates. A simple proof is given as:

$$\boldsymbol{w}_i = U_p^T \cdot \boldsymbol{m}_i \; and \; then \; \boldsymbol{m}_i = U_p \boldsymbol{w}_i \; since \; U_p U_p^T = I$$

Finally, we can express our original image vector

$\boldsymbol{x}_i = \boldsymbol{\mu} + \boldsymbol{m}_i = \boldsymbol{\mu} + w_{1i}\boldsymbol{u}_1 + w_{2i}\boldsymbol{u}_2 + \cdots + w_{pi}\boldsymbol{u}_p = \boldsymbol{\mu} + \sum_{\boldsymbol{j=1}}^{\boldsymbol{p}} w_{\boldsymbol{ji}}\boldsymbol{u}_{\boldsymbol{j}}$, $where \; w_{ji} = \boldsymbol{u}_j{}^T \boldsymbol{m}_i$.

In this way, we express every face as the mean face plus a linear combination of eigenfaces. By comparing the weight vector corresponding to each face, we can compare whether two faces are alike.

To test an unknown face, we will follow these procedures:

1. Convert the new input matrix A^* to a face vector $\boldsymbol{x}^*$

2. Normalize the vector by subtracting the mean face, $\boldsymbol{m} = \boldsymbol{x}^* - \boldsymbol{\mu}$

3. Project the normalized face vector $\boldsymbol{m}$ onto the eigenspace given by U_p, $\widehat{\boldsymbol{m}} = \sum_{\boldsymbol{i=1}}^{\boldsymbol{p}} w_{\boldsymbol{i}}\boldsymbol{u}_{\boldsymbol{i}}$ ($w_{\boldsymbol{i}} = \boldsymbol{u}_{\boldsymbol{i}}{}^{\boldsymbol{T}}\boldsymbol{m}$)

4. Obtain the weight vector for the test image, $\boldsymbol{w} = \begin{bmatrix} w_{\boldsymbol{1}} \\ w_{\boldsymbol{2}} \\ \vdots \\ w_{\boldsymbol{p}} \end{bmatrix}$

5. Calculate the distance between the weight vector of test image and all the weight vectors in the training database and choose the index l that gives the smallest distance.

$$d = min_l \|\boldsymbol{w} - \boldsymbol{w}^l\|$$

6. Compare the distance to a threshold value to decide whether this distance is small enough to be interpreted as two faces are alike. If $d < threshold$, we conclude that the test image A^* is similar to the l^{th} image in the training database. Otherwise, we believe that these two images are from different persons.

Now we have illustrated the complete algorithm of PCA applied in finding eigenfaces and achieving principle components as weight vector. To design a practical system for face recognition, there is always tradeoffs between generality, accuracy, and efficiency. In practice, the designers tend to combine PCA with other methods to develop a better machine learning algorithm. For example, support vector machine (SVM) and linear discriminant analysis (LDA) are two popular choices. In this project, I compared the prediction accuracy given by the models used SVM alone, PCA reduced data then SVM, LDA alone, and PCA reduced data then LDA (figure 2). It is shown that in LDA approach, using PCA reduced 150 eigenfaces as parameters will give us 2% lower prediction accuracy than using original 4096 pixels as parameters. In the next part, I will compare the time efficiency of using PCA with not using it.

Method	Prediction Accuracy
SVM	0.84
PCA+SVM	0.94
LDA	0.99
PCA+LDA	0.97

Table 1

The PCA application in Object Classification

In this part, I mainly focused on the algorithms used to classify an object from several classes. I learnt the popular neural network approach, especially convolutional neural network (CNN). And then applying PCA to the dataset to make a comparison.

I choose the well-known CIFAR-10 dataset which contains 60000 pictures of objects in one of 10 classes. Each picture is composed of 32*32*3 pixels since it is a colored image. Thus, our predictors would have 3072 parameters. To substantially decrease this number, we continue to use the linear algebra concept mentioned above, that, for example, to achieve 90% of the original data variation, we will only require 99 components. Then we use these 99 components as the input layers for CNN. It takes 19 seconds for the first epoch of CNN to run. As a comparison, we also apply CNN directly to the 3072-dimensional data, and it takes 37 seconds for the first epoch to run. The time efficiency is improved by nearly 50% when using PCA as a first step to reduce feature dimension. We also compared their prediction accuracy with the results given by some famous statistical algorithms, such as random forest, KNN, logistic regression, and SVM (table 2). These famous statistical algorithms do not provide a reliable prediction model. CNN gives a much better result. Using CNN after PCA has a prediction accuracy only 2% lower than using PCA alone, while the training time is 50% less than using PCA alone. We believe this tradeoff is worthy when designing a more practical algorithm.

Method	Prediction Accuracy
Random Forest	0.3174
K-Nearest Neighbors	0.3808
Logistic Regression	0.4006
SVM	0.5523
CNN	0.5500
PCA+CNN	0.4800

Table 2

Conclusion

Image classification, as a sub-topic under machine learning, has various scenarios to deal with. Our study shows that statistical theories can be applied efficiently in some scenarios, i.e. in the face recognition problem; while in the object classification problem, a deep learning algorithm such as CNN could give a much better result. It is also a notable fact that using PCA to reduce feature dimension is sometimes followed by other computer science approach. We believe the future study in machine learning may blur the boundary between statistics and computer science; embedding ideas from both subjects can lead to a better model.

Footnotes

[1] Matthew A. Turk, Alex P. Pentland. Face Recognition Using Eigenfaces. *"Face recognition using eigenfaces," Proceedings*. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Maui, HI, USA, 1991, pp. 586-591. doi: 10.1109/CVPR.1991.139758

[2] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. *Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments. University of Massachusetts, Amherst, Technical Report 07-49*, October 2007.

Reference

A. A. M. Hagar, M. A. M. Alshewimy and M. T. F. Saidahmed, "A new object recognition framework based on PCA, LDA, and K-NN," *2016 11th International Conference on Computer Engineering & Systems (ICCES)*, Cairo, 2016, pp. 141-146. doi: 10.1109/ICCES.2016.7821990

J. Lu and K. N. Plataniotis, "On conversion from color to gray-scale images for face detection," *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, Miami, FL, 2009, pp. 114-119. doi: 10.1109/CVPRW.2009.5204297

L. Zhining, G. Xiaozhuo, Z. Quan and X. Taizhong, "Combining Statistics-Based and CNN-Based Information for Sentence Classification," *2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI)*, San Jose, CA, 2016, pp. 1012-1018. doi: 10.1109/ICTAI.2016.0156

Ming Li, Baozong Yuan, *2D-LDA: A statistical linear discriminant analysis for image matrix*, Pattern Recognition Letters,Volume 26, Issue 5, 2005, Pages 527-532,ISSN 0167-8655, https://doi.org/10.1016/j.patrec.2004.09.007.

Stanford University lectures: *CS231n: Convolutional Neural Networks for Visual Recognition*. http://cs231n.stanford.edu/index.html