Pattern Recognition

CA2 Report

GUO ZONGYI

A0206639Y

CONTENT

1. Principle Component Analysis

Main Idea of PCA

PCA is a statistical procedure that 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. This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components. The resulting vectors (each being a linear combination of the variables and containing *n* observations) are an uncorrelated orthogonal basis set. PCA is sensitive to the relative scaling of the original variables.

Variance is represented as:

$$var[z] = E((z-\bar{z})^2) = \frac{1}{n}\sum_{1}^{n}(a^T x - a^T \bar{x})^2 = a^T S a$$

$$where\ S = \frac{1}{n}\sum_{i=1}^{n}(x-\bar{x})(x-\bar{x})^T$$

The goal is to find ‘a’, to maximize variance. Then, let it go through a Lagrange multiplier and calculate the partial derivative to find ‘a’.

$$\frac{\partial}{\partial a}L = Sa - \lambda a = 0$$

$$(S - \lambda I)a = 0$$

Thus, S and λ are eigenvalues and eigenvectors of a.

PCA for Feature Extraction, Visualization and Classification

Feature Extraction and Visualization:

(1) When reduce the dimensionality to 2:

When use the 500 pictures from the PIE dataset to do visualization, the result is shown below:

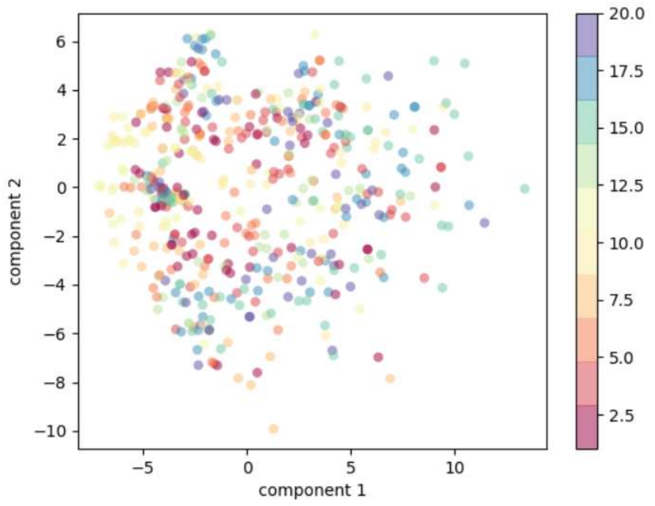


Fig.1 2D visualization of PCA, with PIE dataset

When use the 510 pictures from the PIE and my own photos to do visualization, the result is shown below:

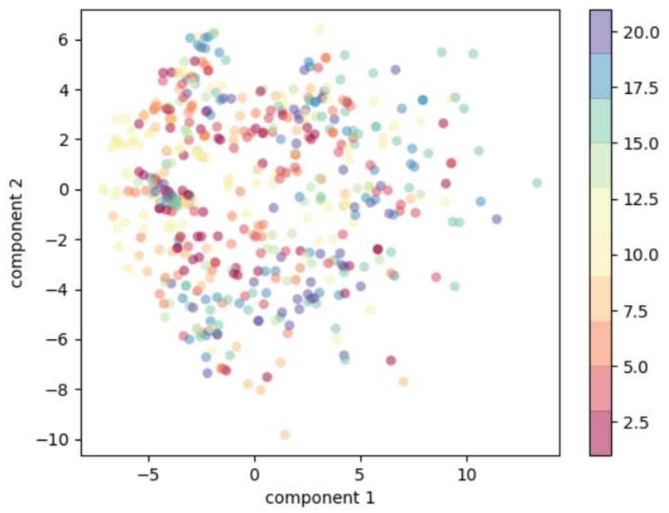


Fig.2 2D visualization of PCA, with PIE and own pic.

In figures, different labels represent different classes.

(2) When reduce the dimensionality to 3:

When use the 500 pictures from the PIE dataset to do visualization, the result is shown below:

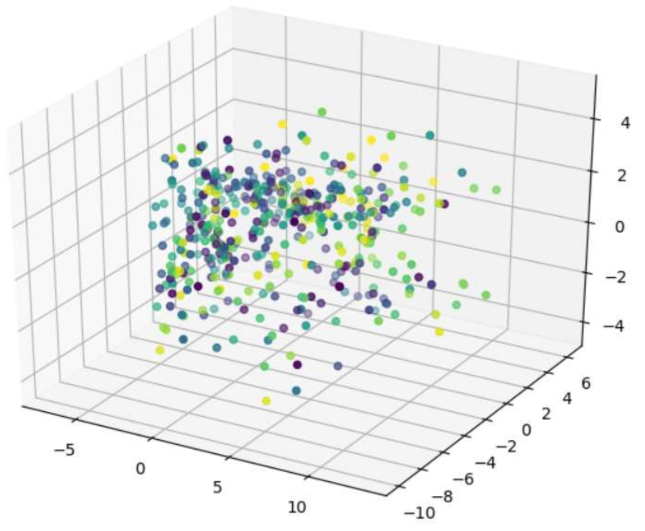


Fig.3 3D visualization of PCA, with PIE

When use the 510 pictures from the PIE dataset and my own to do visualization, the result is shown below:

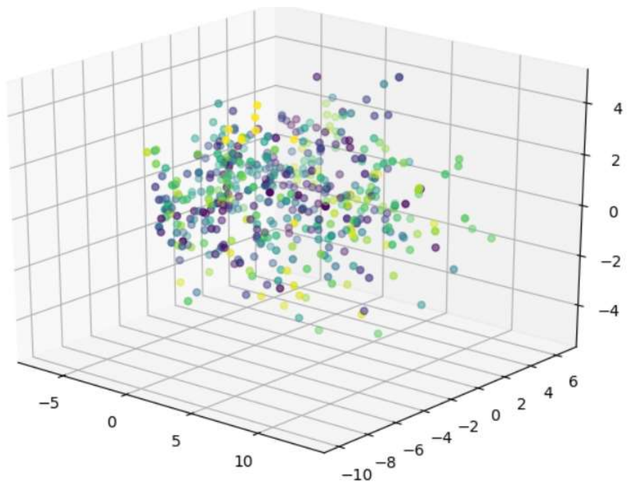


Fig.4 3D visualization of PCA, with PIE and own pic.

(3) Plot eigenfaces:

Use the largest three principle components to form the eigenfaces, which are shown below.

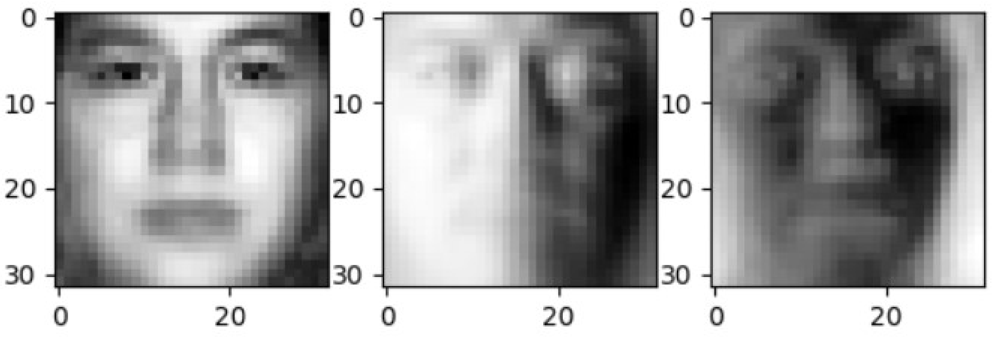


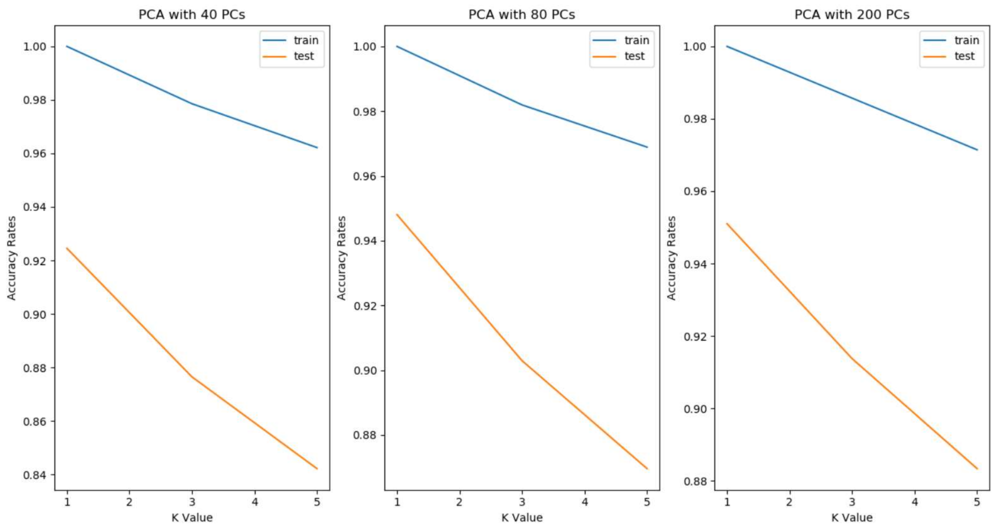
Fig.5 Eigenfaces

PCA Classification, based on KNN

After dimension reduction to 40, 80, and 200, use K-Nearest Neighbor method to do classification. The k-Nearest Neighbors algorithm (k-NN) is a non-parametric method used for both classification and regression. In both cases, the input consists of the k closest training examples in the feature space. The output depends on whether k-NN is used for classification or regression. 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 (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor. Training set includes 119 figures from 1 person and 20 people in total. The rest of dataset is used as test set. Intuitively, pictures with higher dimension, will have better classification accuracy.

(1) Classification Performance Using PIE Dataset

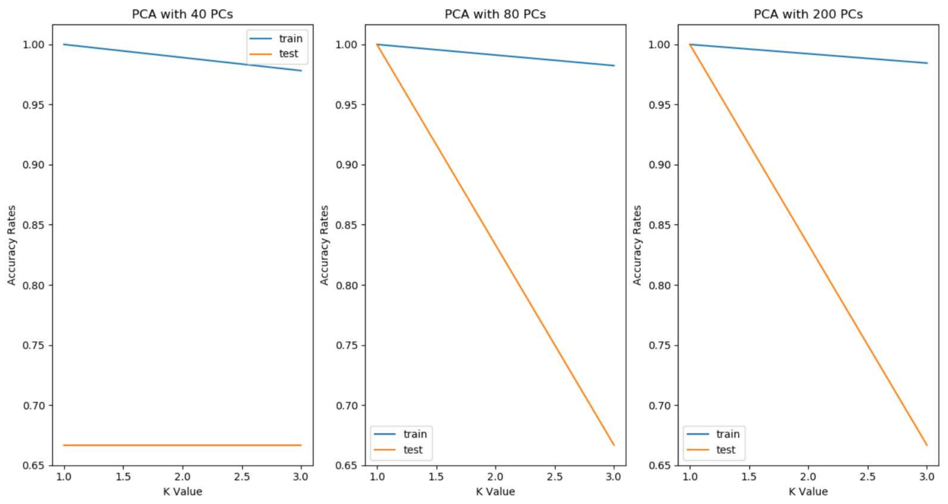
K value is set as 1,3,5. Performance is analyzed with different principle components, considering different K values.



Accuracy		Number of PCs		
		40	80	200
K Value	1	92.5%	94.8%	95.0%
	3	86.7%	90.3%	91.4%
	5	84.2%	86.9%	88.3%

Fig.6 Accuracy Format of PIE Dataset

(2) Classification Performance Using PIE and My Own Pictures



Accuracy		Number of PCs		
		40	80	200
K Value	1	66.7%	100%	100%
	3	66.7%	66.7%	66.7%

Fig.7 Accuracy Format of Own Pictures

Summary

In visualization part, when the dimension is reduced to 2 or 3, the points are entangled with each other, which makes it difficult to discriminate each class. The accuracy for 2 or 3 principle components should be relatively poor. The eigenfaces, formed by the largest 3 principle components, show the common characteristic of the whole 510 pictures. Intuitively, the eigenface formed by the most important principle component, is much clear and implies more detail than other two faces.

In classification, when using PIE dataset, the accuracy in different conditions differ with each other. As we can see, a constant K value with the increasing of number of principle components, the accuracy also increases. With the increasing of K values and constant number of PCs, the accuracy decreases. The results of both practice and theory are consistent with each other. When classification using my own data, the graph shows that in higher dimensions (80 and 200), it can accurately classify the pictures of my face. On the other conditions, it may not have good performance, due to the lack of the training examples (only have 7 pictures to train, others classes have more than 100 pictures).

2. Linear Discriminant Analysis

Main Idea of LDA

Linear discriminant analysis (LDA) is a generalization of Fisher's linear discriminant, a method used in Machine Learning to find a linear combination of features that characterizes or separates two or more classes of objects or events. The resulting combination may be used as a linear classifier or, more commonly, for dimension reduction before later classification.

Note that LDA is a supervised learning algorithm, and cannot be used in regression problems, which differs from PCA.

The goal of LDA is maximize the between-class scatter, and minimize the within-class scatter.

The important matrix in LDA is shown below:

(1) Class-specific mean vector: $\mu_i = \frac{1}{n_i}\sum x$, n_i is the size of class C_i

(2) Class-specific covariance matrix: $S_i = \frac{1}{n_i}\sum(x-\mu_i)(x-\mu_i)^T$

(3) Total mean vector: $\mu = \frac{1}{N}\sum x$, N is the number of all samples

(4) Within-class scatter: $S_w = \sum_{i=1}^{C} S_i = \sum_{i=1}^{C} P_i S_i$, C is the class number

(5) Between-class scatter: $S_B = \sum_{i=1}^{C} P_i(\mu_i-\mu)(\mu_i-\mu)^T$

After simplify the Fisher criterion, we obtain the equation:

$$(S_w^{-1}S_B - \lambda I_d)w = 0$$

Thus, we only need to find the first largest eigenvector, eigenvalue of $S_w^{-1}S_B$.

LDA for Feature Extraction, Visualization and Classification

LDA for Feature Extraction and Visualization

(1) When reduce the dimensionality to 2:

When use the 500 pictures from the PIE dataset to do visualization, the result is shown below:

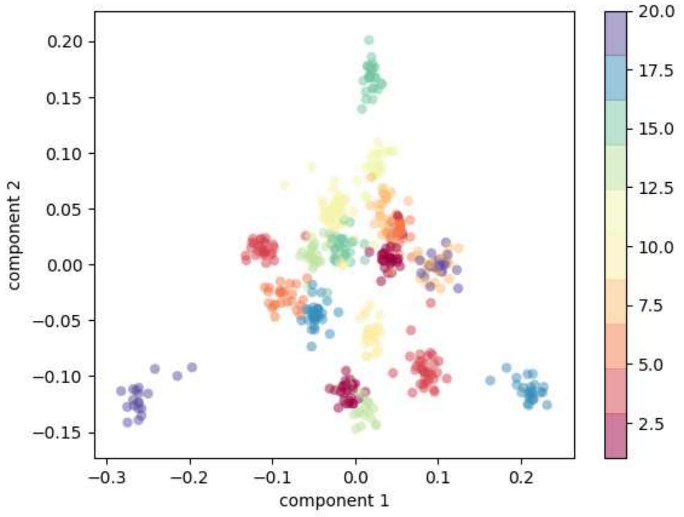


Fig.8 2D visualization of LDA, with PIE

When use the 510 pictures from the PIE dataset to do visualization, the result is shown below:

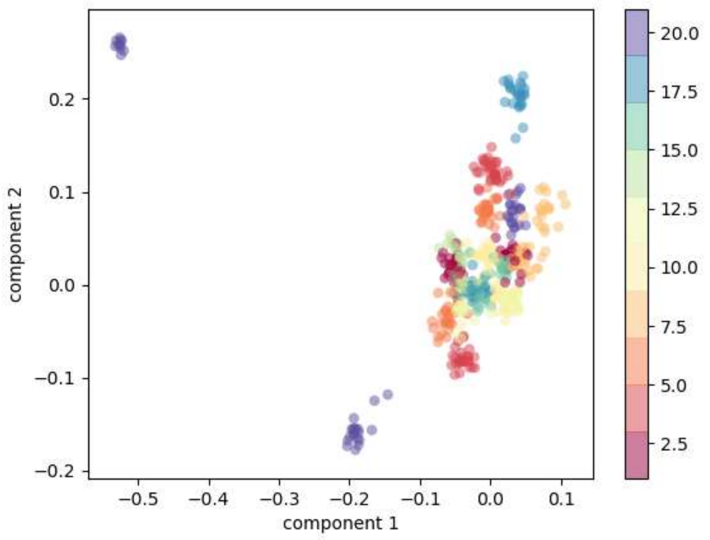


Fig.9 2D visualization of LDA, with PIE and own pic.

The class in the upper left corner is mine, which means my class is very different from the others.

(2) When reduce the dimensionality to 3:

When use the 500 pictures from the PIE dataset to do visualization, the result is shown below:

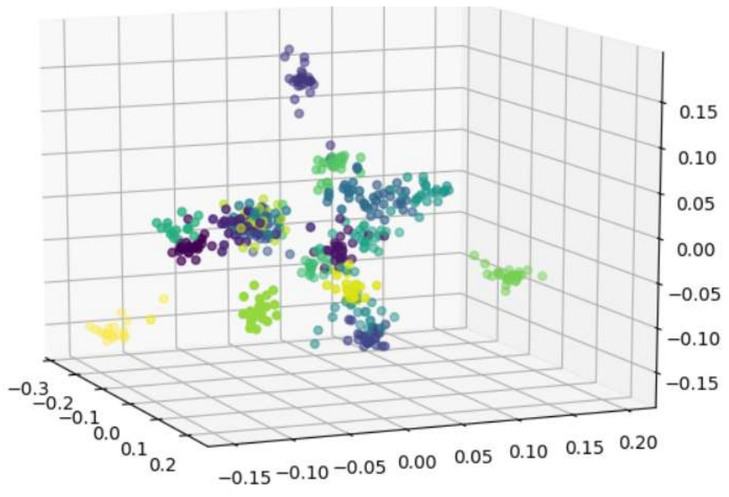


Fig.10 3D visualization of LDA, with PIE and own pic.

When use the 510 pictures from the PIE dataset to do visualization, the result is shown below:

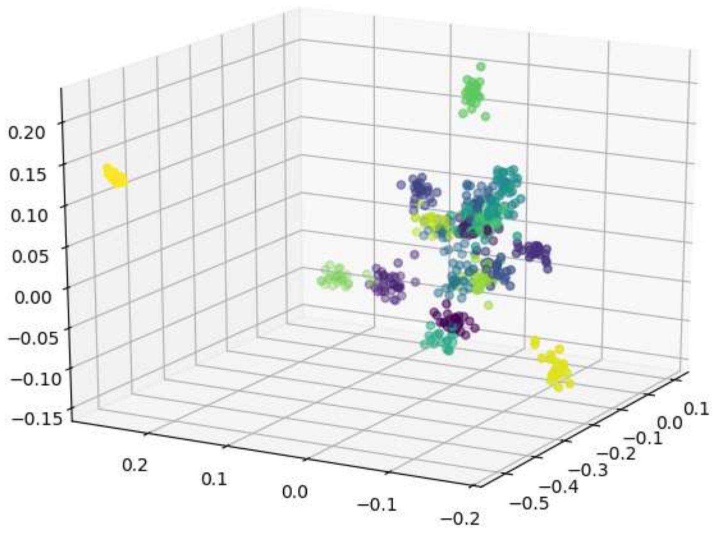
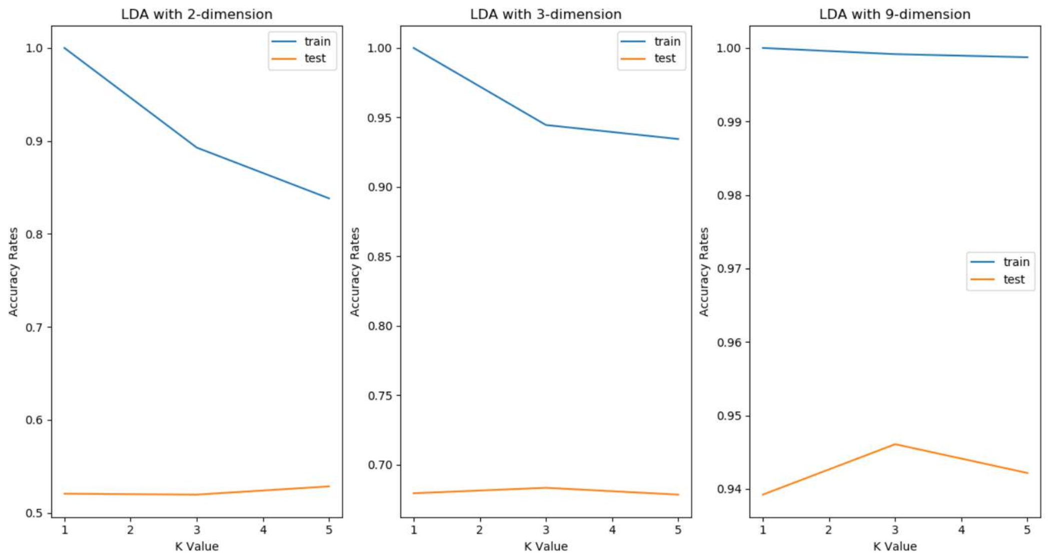


Fig.11 2D visualization of LDA, with PIE and own pic.

Similarly, the class in the upper left corner belong to me, which means it differs from the other classes a lot.

LDA for Classification, based on KNN

(1) Classification Performance Using PIE Dataset



Test Accuracy		Reduced Dimension		
		2	3	9
K Value	1	52.1%	67.9%	93.9%
	3	51.9%	68.3%	94.6%
	5	52.8%	67.8%	94.2%

Fig.12 Accuracy Format of PIE Dataset

(2) Classification Performance Using PIE and Own Dataset

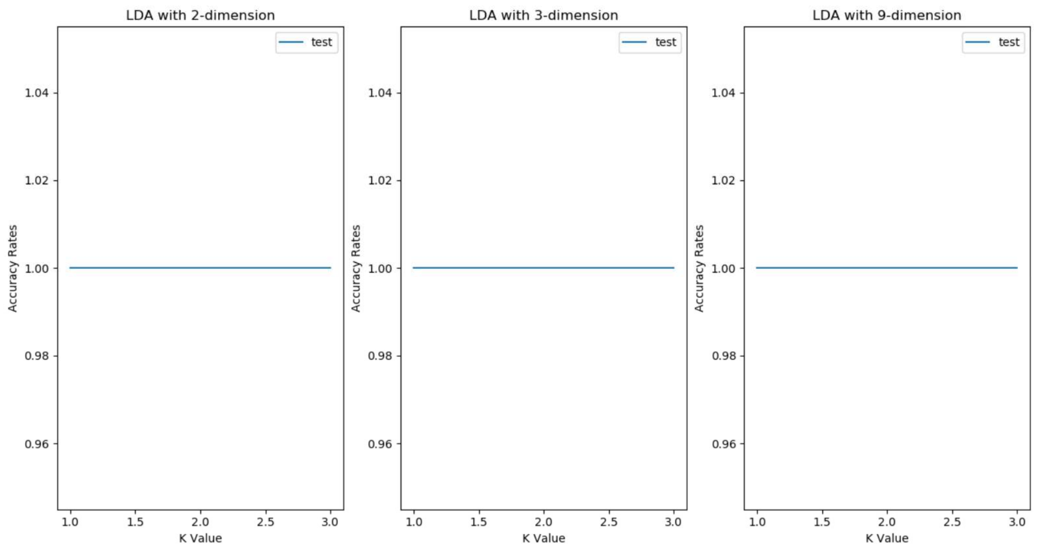


Fig.13 Accuracy Format of PIE Dataset and own pic.

The accuracy is always 100%, because of the big difference between own pictures and PIE dataset.

Summary

When regarding to classification using PIE dataset, the changes with the dimension dramatically. With the increasing of dimension, the accuracy increases from 52% to 94%, which means the higher the dimension, the higher the performance. But, noted that, the dimension should not go beyond (C-1). In this case, that is 19 or 20, which is the basic characteristic of LDA. Thus, in this case, choosing the dimension of 9, maybe the good choice for classification. The reason why it has poor performance when only use 2 or 3-dimension, it is because these dimensions are not enough to denote all the important details from the training set. When increase the dimension, the reduced matrix then can include more details so that accuracy can be guaranteed.

When test on my own faces, from the visualization, it’s easy to find that the projection of my own picture is far away from the PIE dataset. Thus, the result proves this that the KNN classifier can easily classify my own photo. Therefore, the accuracy is 100%, no matter of the dimension, already reduced.

3. Support Vector Machine

Main Idea of SVM

In machine learning, support-vector machines are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier (although methods such as Platt scaling exist to use SVM in a probabilistic classification setting). An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on the side of the gap on which they fall.

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

When consider non-separable cases, we use ‘slack variables’. The goal is:

$$min_{w,b,\varepsilon}\frac{1}{2}W'W + C\sum \varepsilon_i$$

$$Subject\ to\ Y_i(W'X_i + b) \geq 1 - \varepsilon_i\ for\ all\ i$$

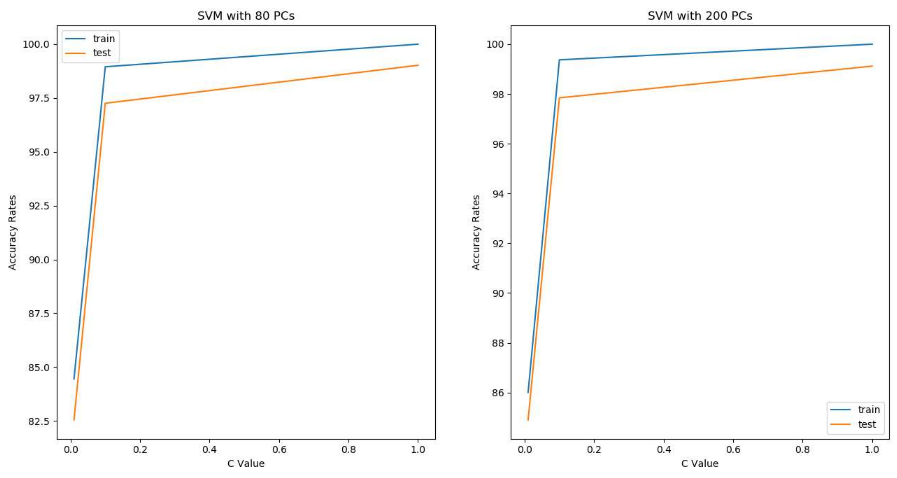
$$\varepsilon_i \geq 0\ for\ all\ i$$

This is called soft-margin SVM.

SVM for Classification

(1) Classification using PIE dataset, processed by PCA:

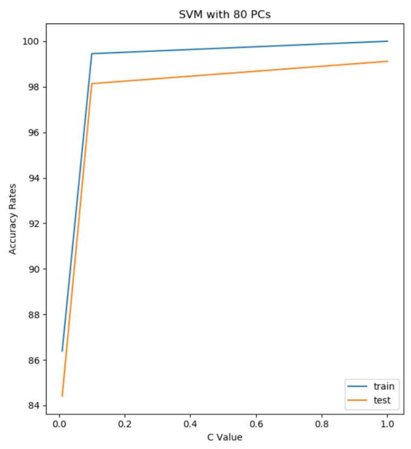
Classification result is shown below.



Test Accuracy		Number of PCs	
		80	200
C Value	0.01	82.5%	84.9%
	0.1	97.3%	97.8%
	1	99.0%	99.1%

Fig.14 Accuracy Format of PIE Dataset

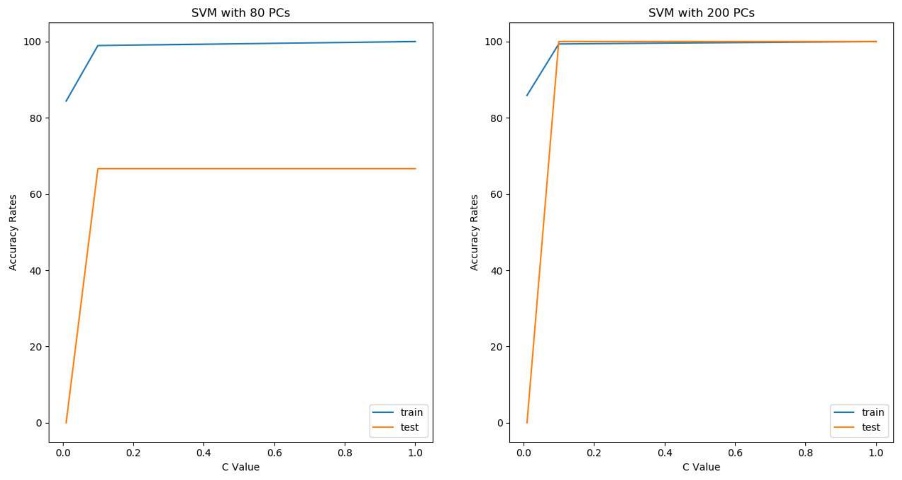
(2) Classification using raw images



Test Accuracy		Raw Image
		Dimension = 1024
C Value	0.01	84.4%
	0.1	98.1%
	1	99.1%

Fig.15 Accuracy Format of Raw Dataset

(3) Classification using PIE and my pictures, processed by PCA:



Test Accuracy		Number of PCs	
		80	200
C Value	0.01	0%	0%
	0.1	66.7%	100%
	1	66.7%	100%

Fig.16 Accuracy Format of PIE and Mine Dataset

Summary

From the graphs shown above, we can reach some conclusions. When using PIE dataset to do classification, with the increase of C value, the accuracy also increases, which is coordinate to theory. Decrease C value, which means it has a higher tolerance of mal-classification. Thus, it results to poorer performance. Besides, similar to previous cases, with the increasing of dimensions, the accuracy also gets higher, which is because larger dimensions will preserve more details from the training data.

If we use the raw images to be the input, which means we do not use PCA to reduce dimensions, the performance is very close to that of 200 PCs, using PCA. This means 200 PCs can already preserve the whole important information from the raw images. Thus, it's better to use PCA to reduce the dimension from 1024 to 200, which can preserve computation burden and reduce complexity.

When use SVM to classify my own pictures, as the result of lack of training examples, the accuracy is very poor when C equals to 0.01. As the increasing of C values, the accuracy also increases. Besides, the performance with 200 PCs also beats that with 80 PCs, because of more details preserved.

4. Convolutional Neural Network

Main Idea of CNN

CNNs are regularized versions of multi-perceptrons. Multilayer perceptrons usually mean fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "fully-connectedness" of these networks makes them prone to overfitting data. Typical ways of regularization include adding some form of magnitude measurement of weights to the loss function. However, CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns. Therefore, on the scale of connectedness and complexity, CNNs are on the lower extreme.

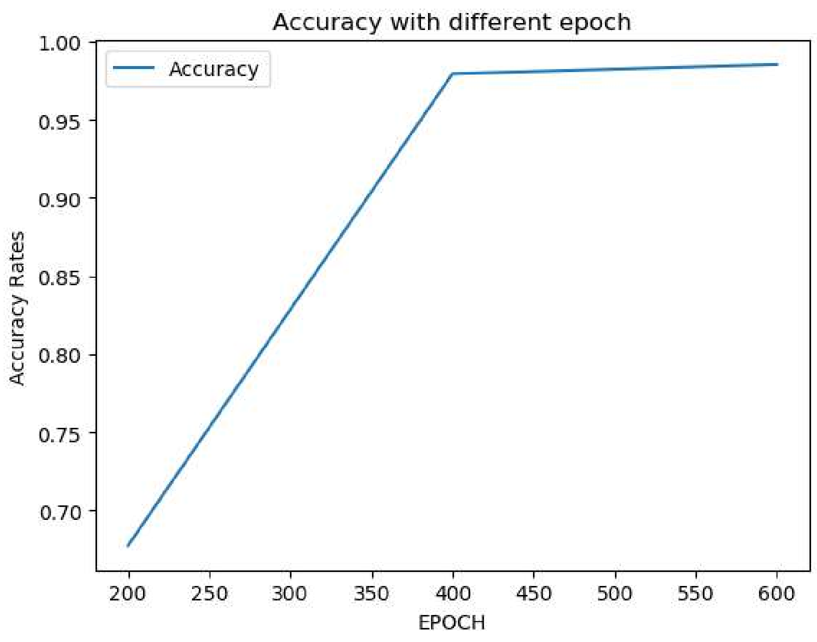
CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns the filters that in traditional algorithms were hand-engineered. This independence from prior knowledge and human effort in feature design is a major advantage.

CNN is one typical Deep Learning model, which consists not only fully-connected layers, but also convolutional layers, padding and pooling layers. Convolutional layers can extract topological properties from an image.

CNN for Classification

Baseline classification analysis

This is based on the given hyperparameters, given in the CA2 instruction file. Accuracy is shown in the following graphs.



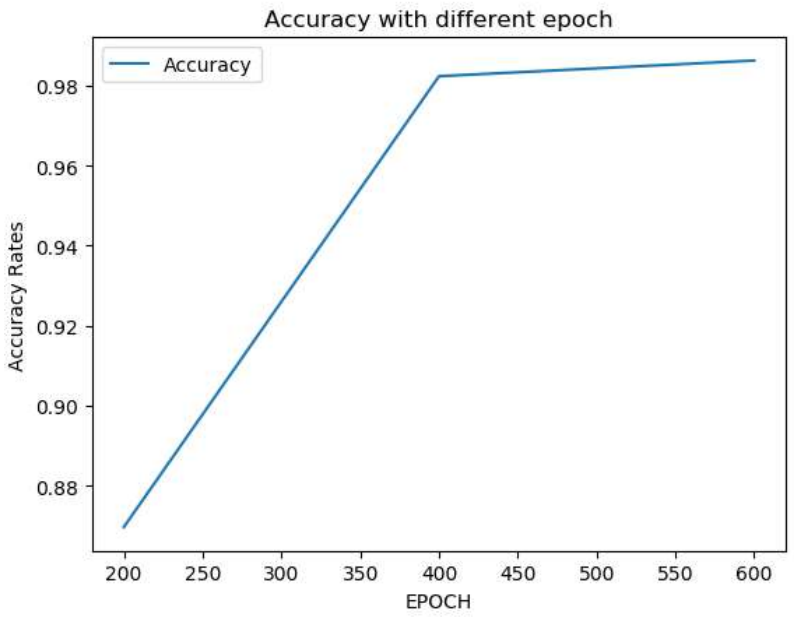
EPOCH	Test Accuracy
200	67.7%
400	97.9%
600	98.5%

Fig.17 Performance of baseline CNN

Classification performance analysis using different CNN models

① Change number of neurons in fully-connected layer

Through changing the number of neurons in fully-connected layer from 500 to 1000, analyze the accuracy. The other parameters are not changed this time.



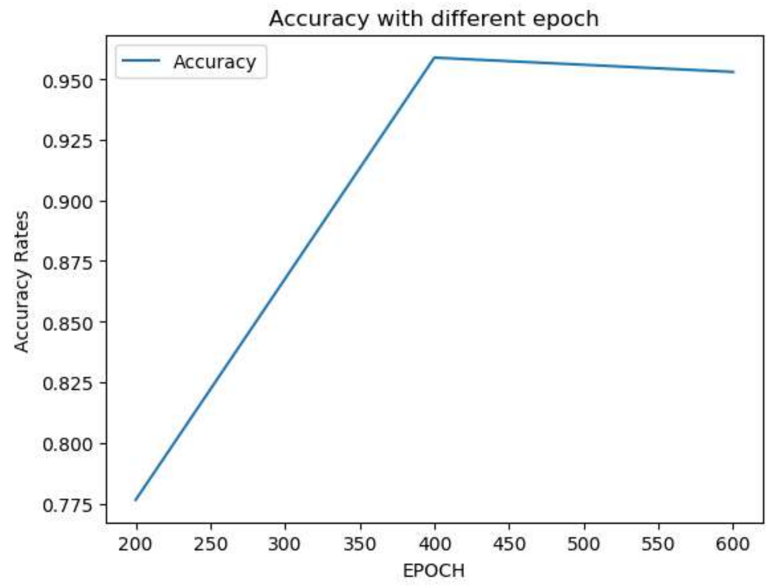
EPOCH	Test Accuracy
200	86.9%
400	98.2%
600	98.6%

Fig.18 Performance of 1000 neurons in FC layer

Performance is higher than baseline model, considering all of the different number of epochs. Thus, this optimization is useful.

② Add activation functions after each convolutional layer

This time I adds two more activation functions, which are also ReLU functions, after two convolutional layers. Performance is shown below.



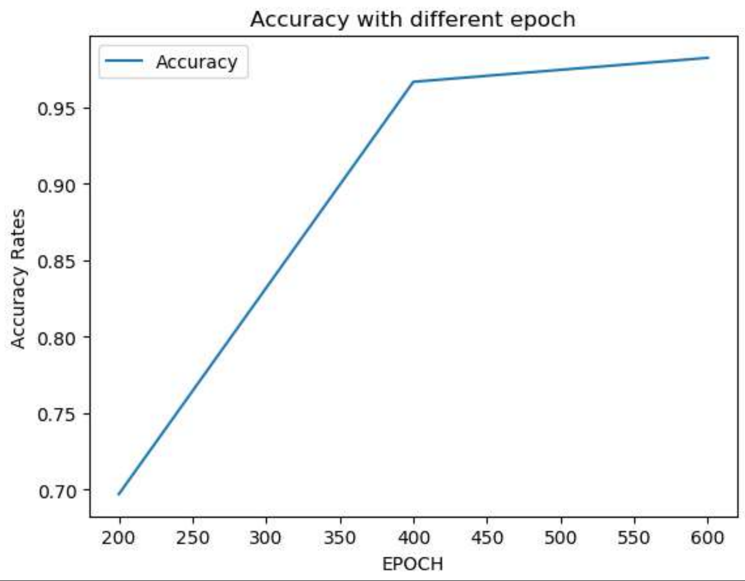
EPOCH	Test Accuracy
200	77.6%
400	95.8%
600	95.3%

Fig.19 Performance of adding more ReLU functions

The performance is poorer than before.

③ Add one more fully-connected layer, with 500 neurons.

The performance is shown below.



EPOCH	Test Accuracy

200	69.7%
400	96.7%
600	98.2%

Fig.20 Performance of adding more ReLU functions

The result is poorer than precious case. Thus, this optimization may not be used.

Summary

In the CNN model, it consists 2 convolutional layers, each layer followed by a max pooling layer. Besides, it also has one fully-connected layer before the output. It also has one activation function to obtain the ability to do non-linear classification. The number should be figured out of the input of fully-connected layer. After two pooling layers, the dimension of each picture is reduced to 5*5, mainly because of halving the dimension, when it goes through the pooling layer. Thus, the input dimension should be 50*5*5. Besides, the learning rate is 0.01. Cross-Entropy loss function is chosen and optimizer is Adam optimizer. Moreover, batch size equals to 64 and shuffle is used. Finally, 'dataloader' function from Pytorch is used to automatically load data.

With increasing the hyperparameter, number of iterations, the accuracy increases from 68% to 98%, which shows a very powerful classification ability. Final classification result of the baseline CNN model (using the given conditions), is about 98.5%. Then, several changes have been made to the baseline model to check whether it can reach higher performance. Firstly, number of neurons in fully-connected layer is changed to 1000, from 500. The performance, shown above, implies it has higher accuracy in all of the different epochs. Thus, the optimization is useful. Besides, two more activation functions are added after two convolutional layers. The performance finally ends up to be poorer than before. Thirdly, one more fully-connected layer is added, with 500 neurons. This time, the accuracy is poorer than previous case. Besides, several other cases, like changing the number of neurons in a particular layer, have been experimented. However, these cases cannot perform better than previous case. Thus, they are not presented in this report. After changing different architectures of CNN model, finally, I would choose the model, which has 1000 nodes in fully-connected layer.

In short, CNN model is very powerful. It can reach more than 98% test accuracy. Comparing with other methods above mentioned, it one of the methods has the highest accuracy.