Abstract

This report analyses on various aspects of performance of face recognition by several classification, including Principal Component Analysis (PCA), Support Vector Machines (SVM) and SVM by using PCA coefficients. The latter two achieved 98.08% accuracy, while SVM by using PCA coefficients saves four fifths execution time comparing to SVM by its own. We concluded SVM achieves highest accuracy and, though SVM using PCA coefficients is less computationally expensive, its accuracy can never exceed SVM.

Introduction

Nowadays human face recognition has been an important subject in computer vision and machine learning. A few recognition intelligence has been created and proved its effectiveness in the past half century. In this report we will compare and discuss upon various aspects of performance including accuracy, execution time, recognition margin etc. of Principal Component Analysis (PCA) and Support Vector Machine (SVM) on distinguishing human faces.

Data Partition

Here we have a piece of 520 faces as our raw data, each of which is present in a 56\*46 grayscale format. Among those faces, we found that every successive ten faces belong to a single person, referring to Figure 1. For each ten faces belonging to the same person, we took nine of them out as training set and the remaining as test set.

  
Figure 1

Principal components analysis

We first applied Principal components analysis to our training set. In order to obtain the principal components (eigenfaces) among training set, we calculated the eigenvectors and eigenvalues of the data covariance matrix **S**, where

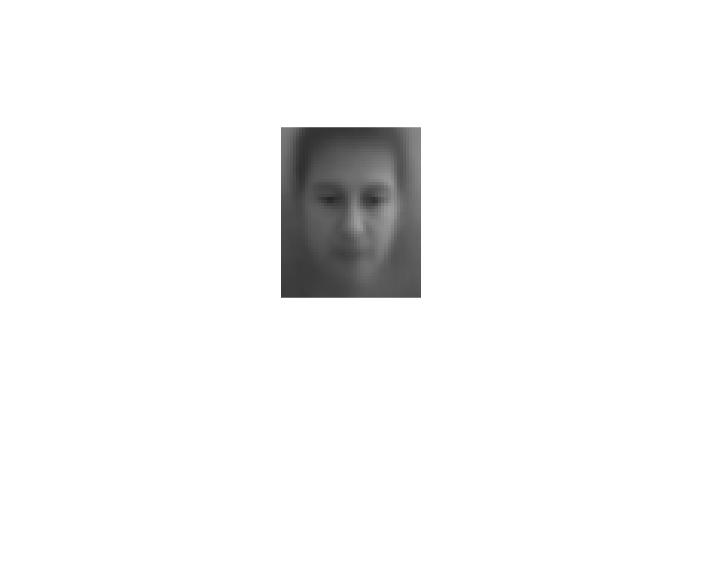
,

N is the number of raw data,

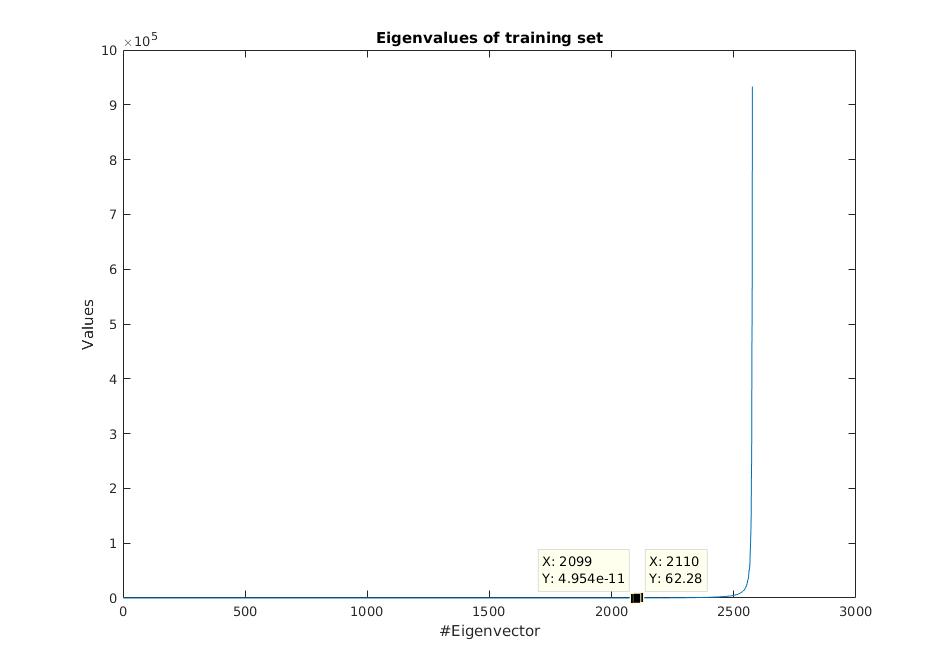
**A** is the difference of each face vector from the mean face.

We can plot the mean face image, as shown in Figure 2.

Data covariance matrix evaluates on the difference between each pair of raw face instance. Then eigenvectors and eigenvalues allow us to extract the features with the most variance since eigenvalue is a measure of strength of respective eigenvector.

  
Figure 2

Finally we got a total number of 2576 eigenvectors and eigenvalues. If we plot those eigenvalues, we have

  
Figure 3

As shown in Figure 3, there is a eigenvalue drop between the 2099th and 2110th eigenvector, from 62.28 down to 4.95e-11, telling us those faces among our data sets behave no much difference on those features represented by the first 2099 eigenvectors. As a result, to save computational power, we only select the largest 467 eigenvectors as our principal components and drop others.

Reduce the covariance matrix dimension

In the previous section we have a covariance matrix whose dimension equals to the number of attributes and thus in our case equals 2576. Computing such a large dimension matrix consumes a lot of execution time. Instead, we can construct the covariance matrix with much smaller size while obtaining the same set of principal components.

More specifically, instead of calculating

,

we compute

and the same eigenfaces can be obtained by

,

Where *u* is the same set of principal components and *v* is the eigenvectors of **S**’.

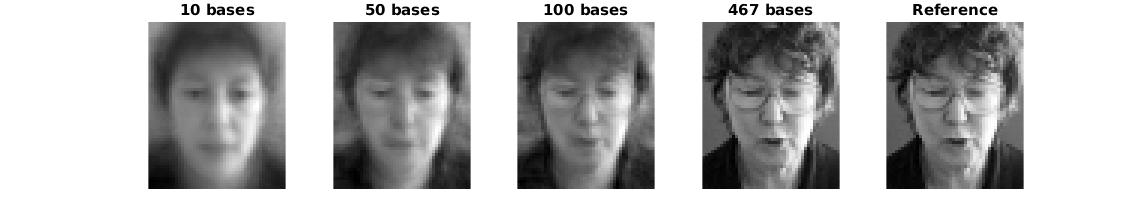
Here we obtained 468 eigenvectors and eigenvalues which are exactly identical as the largest 468 eigenvectors and eigenvalues in last sections. Table 1 summaries the average time consumed on calculating eigenvectors out of ten trials.

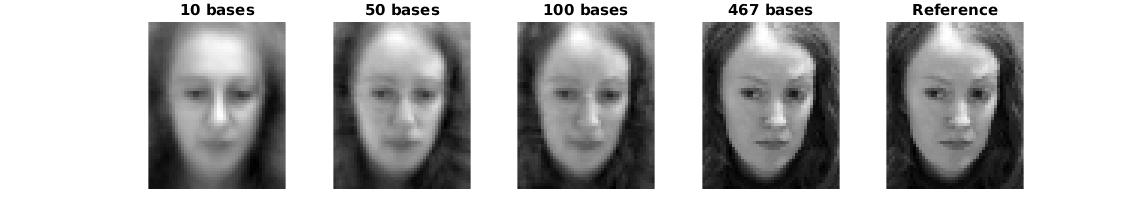
|  |  |
| --- | --- |
| Strategy | Time consumed |
| AA’T | 1.511074 s |
| A’TA | 0.042874 s |

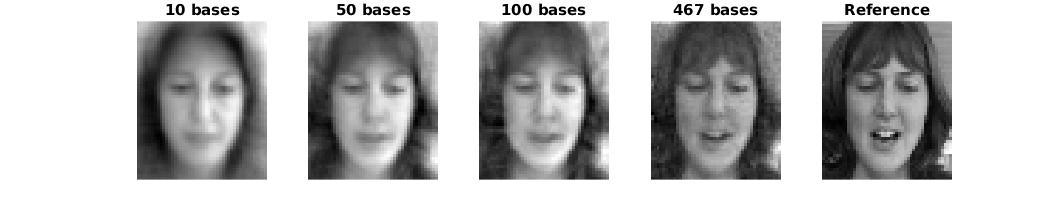
From Table 1 it can be easily deduced that the second strategy is much more timing efficient. However it must be noted that the second strategy still has some inherent limitations that it can only derive a number of eigenvectors equal to the number of data we have. In our case we obtained 468 eigenvectors, which includes all the principal components, from 468 faces in training set. However, consider the case that the number of significant eigenvectors (eigenfaces) exceeds the data count by much, we are suffering from a loss in number of principal components obtained if the second strategy is applied.

Face Image Reconstruction

We now applied face Image reconstruction by using PCA. Here we selected two images from training set and one from test set. Figure 4, 5, 6 shows the respective reconstructed image, by using a different number of bases.

  
Figure 4

  
Figure 5

  
Figure 6 From test set

Here we saw that at 10 bases, the reconstructed image is barely recognizable while at 467 bases, the reconstructed image is very much identical to the original image. One another point to note is that in the third set of reconstruction, the regenerated face from test set has more variation with reference image comparing to other reconstructed ones from training set. Table 2 gives a much more intuitive way measuring how different each reconstructed image from reference image, evaluated by the summing absolute difference at each pixel.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 10 bases | 50 bases | 100 bases | 467 bases |
| Image 1 | 45778.8678597105 | 32084.7713540377 | 24651.2041768910 | 1.45682221841525e-09 |
| Image 2 | 48881.3783559088 | 26309.7164171667 | 18632.4823358340 | 2.12943263022680e-09 |
| Image 3 | 36743.7026307189 | 24455.3554848496 | 21337.6406366512 | 14196.3581145697 |

Among those data present in Table 2, in Image 3 there shows a considerable difference between reconstructed face and reference face comparing to Image 1 and 2. The reason is, we constructed *test set* sample face by projecting into the *training set* principal components. However we can still seem a tendency that the difference decreases with more and more bases we used during reconstruction.

PCA-based Face Recognition

In this section, we applied PCA based face recognition to our test set. To reinterpret our experiment settings, we are having 468 faces as training set and 52 faces as our test set. Our principal components consists 467 pre-computed most significant eigenfaces we found in previous sections. We obtained the PCA coefficients by projecting each test face into the principal components. Each test face is assigned to its nearest target class neighbour.

Through our testing, the recognition process takes an average of 0.215614 seconds out of ten trials. Recognition results behave exact coherence out of these trials and have a constant accuracy of 75%. Figure 7 shows the obtained confusion matrix.

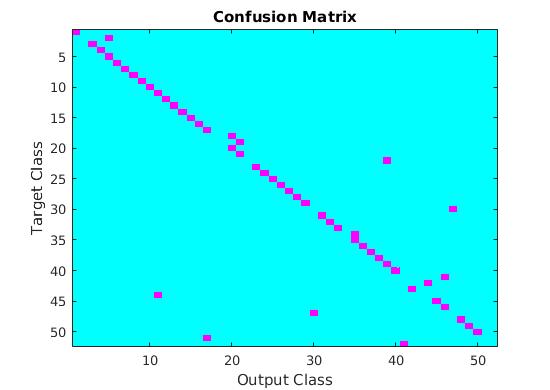
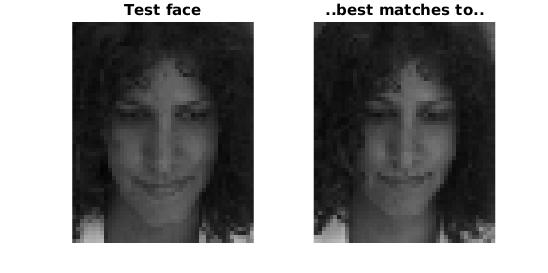
  
Figure 7

Figure 8 and 9 show an example of successful and failed recognition case respectively.

  
Figure 8

Improve the execution speed

  
Figure 9

The same accuracy can be achieved with less execution time by using fewer principal components. As shown in Table x, the accuracy does not drop until the number of eigenfaces used drops below 50, implying that in our case the most significant 50 eigenfaces do a major role in distinguishing between different faces.

|  |  |  |
| --- | --- | --- |
| No of eigenfaces | Accuracy | Execution Time |
| 467 | 0.75 | 0.215614 s |
| 100 | 0.75 | 0.078719 s |
| 50 | 0.75 | 0.066534 s |
| 49 | 0.7308 | 0.066383 s |
| 45 | 0.7308 | 0.065439 s |
| 40 | 0.7115 | 0.064243 s |

SVM Multi-class face recognition

We now moved on to SVM analysis on our training and test data. SVM linearly separates two target classes by mapping data into a higher (or infinite) dimension space. There are two approaches to implement a multi-class SVM, one is to train several One-versus-all (OVA) machine and the other is to train one-versus-one (OVO) machines. We will cover both approaches in the following sections.

To apply SVM to our face data, we followed a traditional procedure listed as following throughout the entire testing process [1].

* Transform data into a LIBSVM compatible format
* Scale the data so that data lies in [-1,1]
* Apply RBF, Linear and polynomial models
* Elaborate on model parameters
* Test

A. One-versus-all machines

To summaries the principal of OVA machines, we trained a total of 52, equal to the number of target classes, machines by the pixel values from training set and each of them will be used to test on the same test set. Each test set instance will be assigned to the class with highest decision values.

Table one summaries the performance of our SVM machine, with default parameters used.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Type | Accuracy | Training Time | Test Time | Total Time | Minimum Margin |
| Linear | 0.9423 | 4.988365 s | 0.435295 s |  | 0.1522 |
| Polynomial | 0.8654 | 4.866701 s | 0.425985 s |  | 0.0035 |
| RBF | 0.8846 | 4.949180 s | 0.416292 s |  | 0.0035 |

Throughout those three kernel types we used in the above, we will then emphasize more on RBF model. There are a few reasons why we would favour RBF models. First RBF model handles better the non-linear relationship between class labels and attributes. In our case it is easy to deduce that the pixel values must have a non-linear relation with class classification. Moreover, linear model can be considered as a special case of RBF model. The same performance by linear model with penalty parameter C can also be achieved by some RBF model with parameter C and gamma.

Underfitting/overfitting on C

The penalty constant C measures the cost imposed on misclassified points. With a large penalty value, a hard margin which tries to place all the training samples into its correct target class is generated and thus could possibly cause over-fit on data model. The table below summaries the performance of SVM with different C parameters.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| C Value of RBF model | Accuracy | Training Time | Test Time | Total Time | Minimum Margin |
| 1 | 0.8846 | 4.949180 s | 0.416292 s |  | 0.0035 |
| 8 | 0.9038 | 5.557688 s | 0.477322 s |  | 4.4653e-04 |
| 32 | 0.9231 | 5.917304 s | 0.484375 s |  | 1.8558e-04 |
| 128 | 0.9615 | 5.928273 s | 0.481910 s |  | 1.3576e-04 |
| 512 | 0.9615 | 5.929451 s | 0.471817 s |  | 1.3576e-04 |

At the beginning, increasing C parameter leads to a higher accuracy, meaning that our first a few model under-fits. With increasing C, the accuracy and minimum margin freeze, telling us that the decision boundary no longer changes, no matter how large the misclassification penalty is.

Same procedure applied on another parameter gamma, where gamma is the free parameter in RBF Gaussian function. Gamma is inversely proportional to the variance and thus small gamma values leads to large variance and more curly decision boundary. The following table demonstrates performance of SVM machine with different gamma values.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Gamma Value of RBF model | C Value of RBF model | Accuracy | Training Time | Test Time | Total Time | Minimum Margin | Comment |
| 0.1 | 32 | 0.5577 | 39.048015 s | 5.032947 s |  | 0.0018 | Under-fit |
| 3.8820e-04 | 32 | 0.9231 | 5.917304 s | 0.484375 s |  | 1.8558e-04 |  |
| 4.8828e-04 | 32 | 0.9615 | 5.551655 s | 0.505110 s |  | 2.0969e-04 | Best |
| 1e-06 | 32 | 0.9038 | 4.952899 s | 0.399528 s |  | 1.0936e-04 | Over-fit |

In this case we observed a clear under-fit, best, over-fit tendency in above table. At first, decreasing gamma value leads to better accuracy and larger margin. Upon reaching best point, where C equals to 32 and gamma equals to 4.8828e-04, rising gamma value squeezes on margin and drops accuracy, showing a typical over-fit behaviour.

best parameters

By applying exponential grid search on parameter C and gamma, we found the best pair is C=32, gamma=4.8828e-04. Note that in Table x, such pair achieved best prediction accuracy among all the trials (96.15%) and exceeds the accuracy by other models (Linear, polynomial).

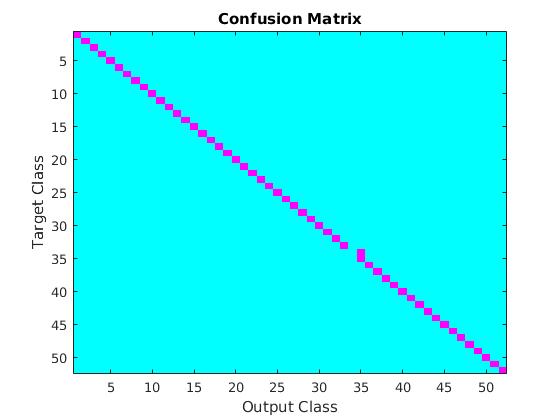
Can it still be improved?

Though an accuracy of 96.15% has exceeded the accuracy of PCA based approach by much, it can still be improved without sacrificing much on execution time. We are aware that in an OVA SVM, the training set is highly unbalanced, i.e. out of fifty-two faces there is only one instance classified into the minor class and all the remaining fifty-one instances belong to the other major class. Discrimination on misclassification penalty leads to a softer margin for major class and harder margin for minor class so as to improve the accuracy of minor class classification. The following tables summaries parameters of our best performance model, which achieves a 98% accuracy.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ratio of class | Base penalty parameter C | Gamma Parameter | Weighting of parameter C on minor class | Weighting of parameter C on major class |
| 51:1 | 32 | 4.8828e-04 | 51 | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Accuracy | Training Time | Test Time | Total Time | Minimum Margin |
| 0.9808 | 5.345750 s | 0.466387 s |  | 1.7396e-04 |

Figure 10, 11 and 12 shows the confusion matrix, failure and success case of face recognition respectively. Note that in each case an image of the most significant support vector, i.e. the support vector with highest coefficient value, from output class is attached. Note that we can observe quite much similarity between test face and support vector not only in success case but also in failure case.

  
Figure 10

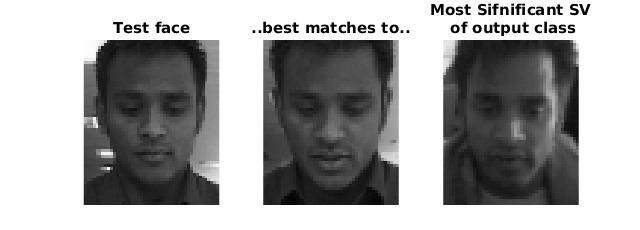
  
Figure 11

  
Figure 12 Wrong case

B. one-versus-one machines

Another approach to apply multi-class SVM is to train models of one class versus each one another class and thus we will have a total of N(N-1)/2 models, where N is the number of target classes. Each model makes a prediction on each test face. The final output class is given by the class with the most vote from the entire model set.

The Table x summaries the best performance of face recognition by using different type of kernels.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Kernel Type | Parameters with best performance | Accuracy | Minimum Margin | Training time | Testing time | Total time |
| Linear | C=1 | 0.9038 | 1.0087 | 2.384590 s | 4.089175 s |  |
| Polynomial | Gamma=1, coef0=0, degree=2 | 0.9231 | 1.2182e+03 | 2.502267 | 4.395384 |  |
| RBF | C=2, gamma= 0.0078125 | 0.8462 | 0.0055 | 2.988897 s | 5.611555 s |  |

In OVO SVM, the linear kernel and polynomial kernel behave far better than RBF kernel in terms of accuracy. The reason is, the number of features (pixels, 2576) exceeds the number of instance (18 instances per training set) by much. One may be aware that if we have a large dimension feature and small number of samples, it will be much easier to linearly separate classes, possibly even without mapping data to a higher dimension space. In our case, without any dimension transformation (Linear Kernel), or with mapping to a slightly higher dimension (Polynomial Kernel with degree 2), the SVM is already able to linearly separate classes quite well, instead of mapping it to an infinite dimension (RBF model).

C. comparison between One-versus-all machines and one-versus-one machines

Through testing we observed that OVA machines have a better prediction accuracy (98.08%) than OVO machines (92.31%). The reason behind can be deduced qualitatively. Consider, in every OVA machine, each test set instance can be classified to one of its two target classes in a model. Each model makes a valid decision. However in OVO machines, for a single test set instance, only N out of N(N+1)/2 models are able to give a valid classification, while others, instead of making supportive decision, produce only distraction since any prediction they make are absolutely wrong. These distractions lower the prediction accuracy.

Multi-class SVM using PCA coefficients

Instead of using pixel as features, it is also approachable to use PCA coefficients we computed in the previous sections as features. The advantage is, dimensions of feature vector drops to around one fourths of previous approach, so as support vectors. Smaller support vectors reduce the time consumed for training and testing data. However due to the fact that PCA inevitably drops some of the features of test variable during reducing data dimension, theoretically SVM using PCA coefficients can never exceed the performance of one using pixels and potentially impair the SVM performance.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Kernel Type | Parameters with best performance | Accuracy | Minimum Margin | Training time | Testing time | Total time |
| Linear | C=16 | 0.9423 | 4.5661e-04 | 0.718480 s | 0.067682 s |  |
| Polynomial | Gamma=0.25, coef0=0,  degree=2 | 0.9423 | 0.0225 | 0.718457 s | 0.072429 s |  |
| RBF | C=32, gamma=0.25 | 0.9808 | 2.4121e-04 | 0.864412 | 0.100247 |  |

Referring to Table x we found that, by using parameters listed above, SVM using PCA coefficients achieves the same performance as the one using pixel values, with only one fifth execution time consumed. Note that for this RBF model, unlike the previous case, discrimination of data set does NOT further improves the accuracy. We concluded the reasons are, first it can never break its theoretical accuracy limitation as we mentioned earlier, and also PCA eliminates some of the aliasing element by removing those features that does not show much variance among training set, and thus reduce the chance of over-fit.

The only failure case is exactly the same one as we presented in previous section. Please refer to Figure 10 ,11 and 12 for confusion matrix, success and failure cases.

Conclusion

In this report we have discussed three approaches of face recognition, PCA with NN classifier, SVM and SVM using PCA coefficients as features. PCA with NN classifier gives the worst accuracy while the others have a much higher accuracy (98.08%). Although SVM and SVM using PCA coefficients have the same accuracy in THIS case, it is obvious that SVM is more versatile and robust, but also more computationally demanding, as SVM using PCA coefficients drops some of the raw information and theoretically can never exceed the former.

Reference

[1] <https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf>

[2] <http://stats.stackexchange.com/questions/46022/libsvm-cost-weights-for-unbalanced-data-doesnt-work>