SID: 430028388

COMP3406 Assignment 1

Introduction

Currently, most of work in machine learning has focused on supervised learning, which is to learn feature representation from unlabelled input data and classify input data based on its feature. It is reasonable since supervised learning has a wide range of practical implications, especially at this time when Big Data is changing the world.

One of four dimensions of Big Data, referred by IBM's data scientist, is variety. Namely, there are various types of data that we can capture and analysed nowadays. Among that captured information, image data is the hardest form of data to be processed and interpreted. Therefore, algorithms focusing on pattern recognition are of interest to many researchers.

In light of the importance of pattern recognition algorithm, this research aims to implement some basic data processing technique and classification algorithms and yield related experience about how to improve classifiers' robustness and performance.

Methods

The data used in this research is CIFAR dataset, which includes CIFAR-10 and CIFAR-100. Both of them are labelled subsets of 80 million tiny images datasets and of size 32x32x3 (32 wide, 32 high, 3 colour channels). It can be seen that the dimensions of image, even though in this case that image is tiny, are pretty large.

To reduce the computational burden of processing image data, dimension reduction technique is applied and optimal number of dimension would be chose based on the validation set. Principle component analysis (PCA) and Zero component analysis (ZCA) are the two commonly-applied approach in machine-learning community.

In this research, Principle component analysis is applied with the aim of reducing dimension. Mathematically, PCA may use eigenvalue and eigenvector in covariance matrix to calculate and rank the importance of features or apply Singular Value Decomposition (SVD) on covariance matrix to calculate and rank the importance of the features. In this research, PCA based on Eigenvalue decomposition is applied rather than SVD.

Algorithmic Description:

Let $X = \{x_1\ x_2\ x_3, \dots\ , x_n\}$ be a random vectors with observations $x_i \in \mathcal{R}^d$.

(1) Calculate the average μ by $\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$
(2) Calculate the Cov Matrix S by $S = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)(x_i - \mu)^T$
(3) Compute the eigenvalues λ_i and eigenvectors v_i of S by $Sv_i = \lambda_i v_i, i = 1,2 \dots n$
(4) Order the eigenvectors descending by their eigenvalue. The k principal components are the eigenvectors corresponding to the k largest eigenvalue

Similarity metrics

The core classifier built in this research is k-nearest neighbour algorithm; its similarity measure is based on Euclidean distance. In other words, we use the sum of squared difference (SSD) to compare the similarity of two pictures. The less SSD two images have, the more similar they are.

However, we are not about to sum all dimensions' squared difference. As we discussed before, the increase in the dimensions used to compare can pose a heavier computational burden. Instead, we compare the weight vectors of query image and images in database, trying to find the most similar image in the dataset. The SSD between two images I_1 (query image) and I_2 (image from training set) would be:

$$Given\ that\ query\ image\ vector\ U\ \in\ \Re, the\ mean\ image\ vector\ from\ the\ database\ A,$$

$$we\ can\ calculate\ the\ weight\ of\ the\ kth\ eigenvalue\ as: w_k = V_k^T\ (U - A)$$

$$where\ V_k\ is\ the\ coresponding\ eigenvector\ for\ w_k$$

$$D_{ssd}^2 = \sum_{i=1}^{n} I_1(w_1, w_2, \dots, w_n) - I_2(w_1, w_2, \dots, w_n)$$

In the following part of this research, we will conduct some transformation on training data like rotation. In this scenario, SSD should be expressed as:

$$D_{ssd}^2 = \sum_{i=1}^{n} I_1(w_1, w_2, \dots, w_n) - T(I_2(w_1, w_2, \dots, w_n))$$

It is notable that the technique to reduce misclassification in this research is not Parameter Tuning. Even though most current researchers focus on finding the optimal parameters, in this case, the technique we adopted is Data Expansion or Resampling technique in Statistics.

Outcome analysis

In CIFAR-10, with the increasing of size of training size, the accuracy of our classifier is improving as well. The best record (37.7%) for CIFAR-10 dataset is achieved the classifier with 50 eigenvalues and 50000 training rows. And its confusion matrix is as follows:

CIFAR-10
Training set: 50000 testing set: 1000
Components (eigenvalue):50
Accuracy score: 37.7%
[[45 5 10 4 8 1 9 4 14 3]
[8 39 1 0 8 3 7 2 10 11]
[11 1 31 8 17 8 15 5 3 1]
[4 3 10 20 15 16 24 5 3 3]
[8 3 11 8 36 4 12 6 1 1]
[2 0 12 10 14 25 10 7 4 2]
[3 1 22 4 23 6 51 1 0 1]
[5 6 6 6 17 10 8 34 5 5]
[18 4 4 5 6 2 2 1 61 3]
[10 8 6 4 6 1 9 8 22 35]]

Figure 1

However, in CIFAR-100, the accuracy of classifier has seen an obvious drop. The best record (23.3%) for CIFAR-100 dataset is achieved the classifier with 100 eigenvalues and 50000 training rows. And its confusion matrix is as follows:

It can be seen that, our classifier's accuracy is particular low when it comes to classify some particular classed like the forth class and the sixed class in CIFAR-10. We are going to address this and improve accuracy systematically in the following part.

Reason for misclassifications: Weakness of PCA

From the perspective of method self, a very likely source for misclassification is the weakness of Principle component analysis. Since the raw data matrix unavoidably contains noise, there is the risk that model would capture the noise (a.k.a sparse matrix in data matrix) and take it into account. Admittedly, in terms of recovering the original data back, the more eigenvector and eigenvalue are used, the recovered image will more approximate to the original one. However, it is not the case when it comes to classification. To improve classifier's performance, it is necessary to filter out noise in image data as much as possible.

Improvement in robustness: RPCA

Emmanuel (2011) brings about a technique called *Robust Principle Component Analysis* (RPCA) to decompose data matrix into low-rank matrix and sparse matrix. In linear Algebra, the rank of a matrix A is the dimension of the vector space generated by its columns. This can be interpreted as the fact that the structural information implied in matrix can be measured and captured by matrix's rank. Conversely, sparse matrix is the residual noise after extracting low-rank matrix from original data. This can be expressed as optimization problem as follows:

$$\min_{A,E} \ rank\ (A) + \ \lambda\ ||E||\ \ s.t. X = A + E$$

$$where\ A\ stands\ for\ low\ rank\ matrix\ and\ E\ stands\ for\ sparse\ matrix$$

In this research, we implement RPCA and use its low-rank part in output as the input of classifier.

Reason for misclassifications: lack of comparable image

It is also possible that some misclassification is because of distorted image or lack of similar image to make comparisons (especially for the classifier based on similarity comparison like KNN). Besides, one of natural drawbacks of KNN is so-called curse of dimensionality.

Improvement in robustness: data expansion and reconstruction

Efron et al. (1979) firstly suggests that parameter estimation could be refined by a resampling approach, which is Bootstrapping, even in the case of lack of samples. After that, resampling technique had been spread out in the school of Frequentist Statistics. Despite of heavier computational burden it may bring up, data expansion may further improve classifier's performance and robustness. In this research, based on the feature of image data, some image data has been rotated and turn into grayscale in order to improve the classifier's accuracy. We can decide which class of data should be resampled based on confusion matrix below:

CIFAR-10
Training set: 50000 testing set: 1000
Components (eigenvalue):50
Accuracy score: 37.7%
[[45 5 10 4 8 1 9 4 14 3]
[8 39 1 0 8 3 7 2 10 11]
[11 1 31 8 17 8 15 5 3 1]
[4 3 10 20 15 16 24 5 3 3]
[8 3 11 8 36 4 12 6 1 1]
[2 0 12 10 14 25 10 7 4 2]
[3 1 22 4 23 6 51 1 0 1]
[5 6 6 6 17 10 8 34 5 5]
[18 4 4 5 6 2 2 1 61 3]
[10 8 6 4 6 1 9 8 22 35]]

We can see that the fourth class and the sixth class are easy to be misclassified by our algorithm. Therefore, we rotate image data of class 4 & 6 and put them into training set.

Speed-accuracy trade-off

(1) Anomaly within the trade-off of speed and accuracy

One of interesting findings about PCA in this research is that more extracted eigenvalues and eigenvectors does not always leads to an increase in precision. One example is that in CIFAR-10 dataset, we take the first 10000 training data and try to classify the first 1000 query images with different number of number of eigenvalues & vectors. It proves that a growth in dimensions we calculate will not always improve accuracy. This may be caused by the noise captured by classifier. We hold other conditions constant and set number of eigenvalues as 10, 30, 50, 100,200, and 500. Outputted Confusion matrices are shown in *Figure 1*.

Confusion matrix
Training set: 10000
Testing set: 1000
Components (eigenvalue):10
Accuracy score: 25.7%
[[35 3 15 5 13 2 6 3 18 3]
[13 26 3 5 13 1 9 2 11 6]
[11 1 34 4 21 4 15 5 3 2]
[4 2 21 20 14 12 20 5 4 1]
[12 0 11 8 27 4 17 5 6 0]
[6 0 20 11 11 10 16 5 5 2]
[0 1 34 5 26 6 35 3 1 1]
[7 0 12 5 21 8 12 27 7 3]
[16 2 4 0 6 4 7 2 62 3]
[13 12 12 7 10 2 13 5 13 22]]

Confusion matrix
Training set: 10000
Testing set: 1000
Components (eigenvalue):30
Accuracy score: 30.4%
[13 30 2 7 4 2 7 5 6 13]
[10 30 9 17 8 16 5 3 2]
[8 3 18 18 15 9 19 5 3 5]
[7 0 10 10 26 3 16 9 7 2]
[3 3 13 13 9 15 16 4 6 4]
[1 1 26 8 19 7 38 8 2 2]
[5 2 14 11 10 9 9 28 6 8]
[20 3 2 1 5 4 7 2 57 5]
[11 19 2 9 5 2 12 6 18 25]]

Confusion matrix
Training set: 10000
Testing set: 1000
Components (eigenvalue):50
Accuracy score: 31.7%
[[40 4 15 5 11 0 6 3 17 2]
[14 30 1 3 9 1 9 3 7 12]
[7 1 35 6 19 6 17 5 2 2]
[5 3 19 21 10 11 26 4 4 0]
[9 0 11 7 33 4 12 6 6 2]
[5 1 14 13 10 14 14 6 5 4]
[1 1 23 6 28 7 39 5 1 1]
[4 1 10 7 19 9 13 30 5 4]
[21 2 5 0 7 5 7 2 55 2]
[10 16 8 7 11 3 9 6 19 20]]

Figure 2

Confusion matrix
Training set: 10000
Testing set: 1000
Components (eigenvalue):100
Accuracy score: 29.8%
[[35 3 15 5 13 2 6 3 18 3]
[13 26 3 5 13 1 9 2 11 6]
[11 1 34 4 21 4 15 5 3 2]
[4 2 21 20 14 12 20 5 4 1]
[12 0 11 8 27 4 17 5 6 0]
[6 0 20 11 11 10 16 5 5 2]
[0 1 34 5 26 6 35 3 1 1]
[7 0 12 5 21 8 12 27 7 3]
[16 2 4 0 6 4 7 2 62 3]
[13 12 12 7 10 2 13 5 13 22]]

Confusion matrix
Training set: 10000
Testing set: 1000
Components (eigenvalue):200
Accuracy score: 29.0%
[[39 3 14 5 12 2 4 3 18 3]
[14 22 2 2 15 1 13 3 12 5]
[13 1 34 3 20 6 15 2 4 2]
[3 1 22 19 13 13 20 6 5 1]
[12 0 10 11 28 3 13 5 8 0]
[5 0 21 9 12 10 16 5 6 2]
[1 1 35 7 23 6 33 3 2 1]
[8 0 14 5 25 9 9 22 7 3]
[17 2 3 1 6 4 5 3 63 2]
[22 7 14 6 10 2 13 4 11 20]]

Confusion matrix
Training set: 10000
Testing set: 1000
Components (eigenvalue):500
Accuracy score: 28.7%
[[36 3 16 4 12 2 6 2 20 2]
[13 19 3 4 18 3 12 2 12 3]
[11 1 36 3 20 8 14 2 4 1]
[2 1 25 22 15 11 17 3 6 1]
[12 0 11 8 28 3 14 6 8 0]
[5 0 20 12 13 13 11 6 6 0]
[1 1 33 8 27 7 29 2 3 1]
[7 0 13 6 24 10 10 22 8 2]
[16 2 2 2 5 3 5 3 66 2]
[22 6 14 6 13 2 12 5 13 16]]

Figure 1

We can see that the classifier with 50 eigenvalues has achieved the best performance. Also, we know that more eigenvalues means that it takes longer time to compute. This shows that parameter selection is far more important than the time to compute. In most cases, classifier's running time and performance is positively correlated but not always is true.

(2) Optimisation

Another function embedded in classifier is the optimisation of parameters in Robust Principle Component. Optimisation algorithm will stop iteration if its error rate falls into tolerance range. Therefore, the setting of tolerance range would influence classifier's accuracy and time to run. There is a hard trade-off between accuracy and speed, not soft one in the previous example.

Firstly, we only use PCA to extract features and classify 1000 testing images with training set with the size of 10000. Next, we use RPCA to decompose the query images to get low-rank matrix and run PCA on outputted low-rank matrix. The output of two classifiers is as follows (figure 2):

```
Confusion matrix
Training set: 10000  Testing set: 1000
Components (eigenvalue):100
Feature extraction: PCA
Accuracy score: 29.8%
[[35  3 15  5 13  2  6  3 18  3]
 [13 26  3  5 13  1  9  2 11  6]
 [11  1 34  4 21  4 15  5  3  2]
 [ 4  2 21 20 14 12 20  5  4  1]
 [12  0 11  8 27  4 17  5  6  0]
 [ 6  0 20 11 11 10 16  5  5  2]
 [ 0  1 34  5 26  6 35  3  1  1]
 [ 7  0 12  5 21  8 12 27  7  3]
 [16  2  4  0  6  4  7  2 62  3]
 [13 12 12  7 10  2 13  5 13 22]]
```

```
Confusion matrix
Training set: 10000  Testing set: 1000
Components (eigenvalue):100
Feature extraction: PCA + RPCA
Accuracy score: 30.2%
[[35  3 16  5 12  3  6  3 17  3]
 [15 24  3  3 16  1 10  2  9  6]
 [11  1 36  4 21  4 16  3  2  2]
 [ 4  2 20 21 15 11 20  5  4  1]
 [11  0 11  7 29  4 16  5  5  2]
 [ 6  0 19 11 11 11 16  5  4  3]
 [ 0  1 34  5 24  6 37  3  1  1]
 [ 7  0  9  4 21  8 12 28  8  5]
 [18  2  3  0  6  4  7  2 60  4]
 [15 15 12  7  9  3 12  5 10 21]]
```

Figure 3

Also, the time required to run simply PCA classifier is only 16 minute 37 seconds. However, we need around 1 hour to run the combination model. The tolerance range of combination model is set to 5e-5. Totally, 4 times of iterations are required as its precision changes are shown below:

$$error = 1.0$$

$$error = 0.000253587090016$$

$$error = 0.000140970552292$$

$$error = 7.8324802077e-05$$

$$error = 4.93416876593e-05$$

It seems that classifier's accuracy is only improved by a little bit. However, it takes nearly four times time to run classifier. It is not like a trade-off we want to make to improve the accuracy, but it is the case only when we will not re-use the training set.

In this research, we need to look through the whole training set and find the most similar image for the query. The training set does not change from query to query; therefore, we can use the low-rank matrix decomposed from training set repetitively. It is reasonable to run optimisation (embedded in RPCA) in order to improve performance.

Conclusions and future work

Obviously, the classifier we build, which applies PCA, KNN, RPCA, does not perform very satisfactorily (Maximum accuracy: 37.7%). The main limit lies in the way to extract feature and core classifying algorithm. We may implement some more efficient feature-extraction algorithm like K-means in the following research. Also, Convolutional Neural Network (CNN) would also be promising classifiers. This is not because it can extract local features (this makes CNN different from NN) but

also since it has a multiple of parameters, which means it is more flexible and can outperform simple classifier like KNN on large dataset.

Discussion

In the process of conducting this research, a wide range of previous relevant papers on classifying CIFAR datasets have been read. Most of those researches focus on finding the optimal parameters. The various ways to define and find those parameters make me know that data scientists are those who can only run available algorithm and adjust parameters. Creation and innovations is still of great importance. One of instances is that supervised learning and unsupervised learning is not independent from each other. K-means, an algorithm I thought should only apply in non-classification problem, can also be used to extract features and improve classifier. Adam et al. (2011) analyse the performance of single layer network in unsupervised learning feature extraction and their research is the one which enlightens me a lot.

Reference:

C. J. Emmanuel, X. Li, Y. Ma and J. Wright, “Robust principal component analysis?", *Journal of the ACM,* vol.58, num.3, 2011.

B. Efron, “Bootstrap Methods: Another Look at the Jackknife”, *Annals of Statistics*, vol.7, num. 1, 1979.

C. Adam, A.Y. Ng and H. Lee, "An analysis of single-layer networks in unsupervised feature learning." *International conference on artificial intelligence and statistics*, 2011.