COMP5318 MACHINE LEARNING AND DATA MINING
Assignment 1 Sem 1/2020
Report

Table of Contents

1 Introduction

In this assignment, grayscale images of the size 28x28 were given to be classified into 10 categories:

```
Class| Name
0 T-shirt/Top
1 Trouser
2 Pullover
3 Dress
4 Coat
5 Sandal
6 Shirt
7 Sneaker
8 Bag
9 Ankle boot
```

The fashion dataset consists of two parts: a training set with 30,000 images and a test set with 5,000 images. The training set was later split into training and validation sets for hyperparameter tuning (see Chapter 2.1.4).

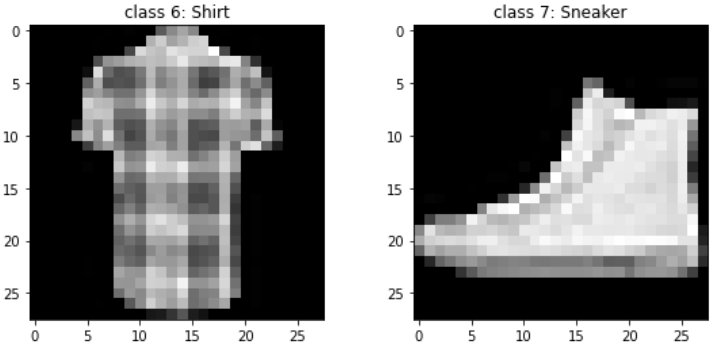

Figure 1-1 Two examples of image respectively from class 6 (left) and class 7 (right), extracted from training data.

The purposes of the study are to first, code two classifiers from scratch using Python and Numpy library, second predict the classes of test data, and third compare the performance (in terms of accuracy and running time) of the classifiers. The classifiers implemented here are K-Nearest Neighbours Classifier and Naïve Bayes Classifier.

Classification is one of the supervised learning approaches in Machine Learning (Murphy 2012, p. 3). It is important to practice and master this approach which is the form of ML most widely used. The goal of classification is generalisation, which we could predict novel inputs. Without the aid of advance packages or libraries in python, we could understand the data pre-processing techniques and application of each classifier algorithm in a deeper and better way.

2 Methods

In this section, the pre-processing techniques and classifier methods algorithms will be discussed.

2.1 Data Pre-processing

We perform Principal Component Analysis to a data and reconstruct the data with first n principal components to retain most of the information while reducing its dimension.

2.1.1 Centering

To center a real matrix **X**, we subtract the mean vector from each row.

$$X_{centered\ i} = X_i - \mu \tag{1}$$

In our study, mean vector μ along columns of the training data is subtracted from the training data and test data respectively to obtained centered training data and centered test data. Figures below show the distribution of datasets before after centering. We can see that the data shifted around 0 after centering (see right column of Figure 2-1).

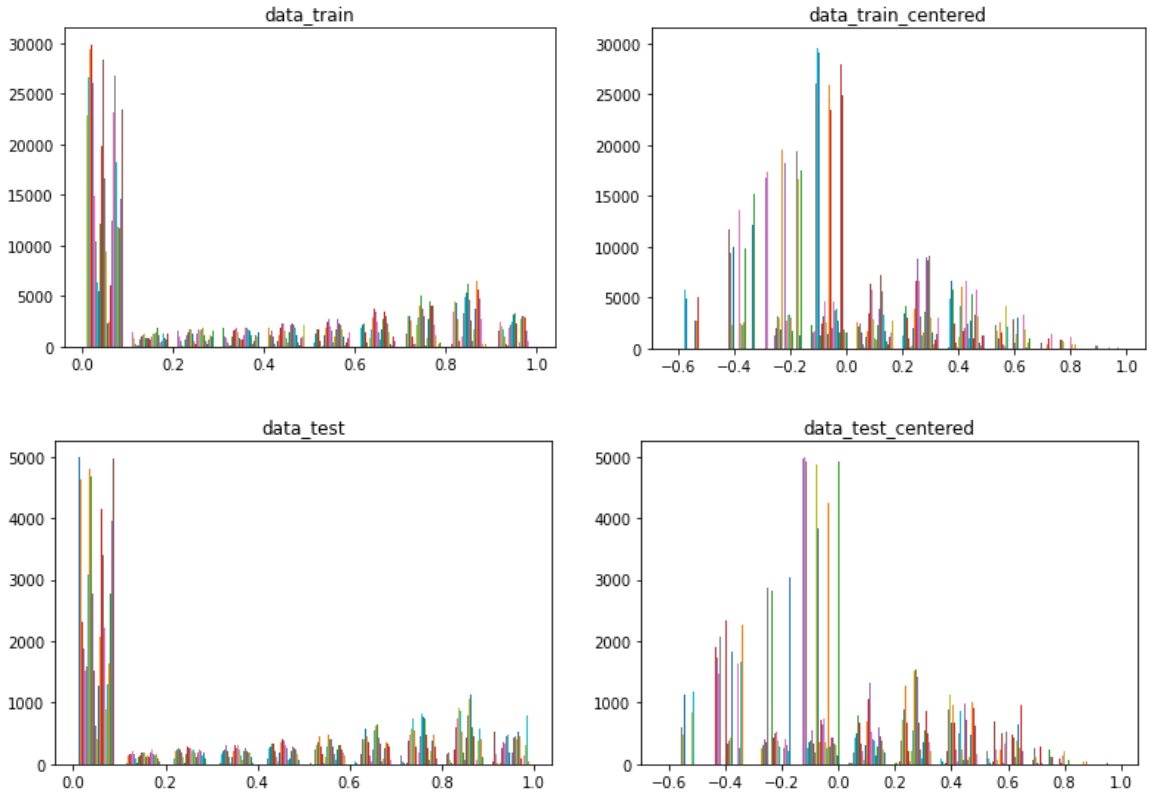

Figure 2-1 Before (left) and after (right) centering training data (above) and test data(bottom).

2.1.2 Principal Component Analysis (PCA)

Principal components of a real matrix **X** (size n x d) are the eigenvectors of $\mathbf{X^TX}$ (size d x d), that explained the variance of the data.

In Principal Component Analysis (PCA), we orthogonally project our data to the most important axes of maximal variance to minimise the reconstruction error(Rajaraman & Ullman 2011, pp. 436-7). For instance, to perform PCA for **X**, we want to first find the covariance matrix of centered **X**, and then its eigenvalues and eigenvectors. The former component found is used to find the explained variance of **X**, while the latter is used to reconstruct the data after dimensionality reduction. In standard practice, other than the steps mentioned, one can also standardise the data and find the eigenvalues and eigenvectors of its correlation matrix (Murphy 2012, pp. 387 - 91).

To find the eigenvalues and eigenvectors of a matrix, we can use Eigen Decomposition (for square matrix) or Singular Value Decomposition (for matrix with any shape).

Eigen Decomposition of $\mathbf{X} = \mathbf{U\Lambda U^T}$,

where **U** is the square $n \times n$ matrix whose ith column is the eigenvector $\boldsymbol{u}_i$ of **X** and **Λ** is the diagonal matrix which has corresponding eigenvalues as its diagonal elements.

Singular Value Decomposition of **X** (size m, n) = $\mathbf{U\sum V^T}$,

where **U** and **V** are column-orthonormal matrices, with size (m, r) and (n, r) respectively, and **∑** is a diagonal matrix with size (r, r), such that r is the rank of **X** (Rajaraman & Ullman 2011, pp. 442-5)

With the relationship of eigenvalues and singular values illustrated and proved by Murphy (2012, p. 392), it is concluded that the eigenvectors of $\mathbf{X^TX}$ are actually **V**, and their corresponding eigenvalues are equals to **∑**.

After obtaining the eigenvalues and eigenvectors, we can find the explained variance of data in % using equation below:

$$explained\ variance_i = \frac{eigenvalues_i}{\sum_i^n eigenvalues} x100 \qquad (2)$$

where n is the number of variables of the data.

As mentioned before, we would want to maximise variance to retain most of the information and reduce reconstruction error. Hence, we select first k of the principal eigenvalues which has high value of sum of explained variance in representing the data. This k value also indicates the final rank of data.

```
[29.07705331 17.72924699  6.01392833  4.93665536  3.8278513   3.4506048
 2.35977445  1.87614082  1.35558285  1.30976276  0.98624744  0.91098136
 0.76765203  0.65798655  0.60750331  0.58897842  0.55580433  0.53109696
 0.46149294  0.45643812  0.43087447  0.4053171   0.3852867   0.37165356
 0.36384276  0.35356986  0.33338035  0.32046338  0.30885648  0.2907938
 0.27789465  0.26788498  0.26331244  0.25521322  0.24738286  0.23658654
 0.23001372  0.22512466  0.21946822  0.21117399  0.20149036  0.19580962
 0.19278139  0.18293842  0.17413398  0.17235527  0.16800805  0.16552647
 0.16119179  0.1554929   0.15369411  0.15113497  0.14736272  0.14320924
 0.13758767  0.13638068  0.13126304  0.12863455  0.12722421  0.12526774
 0.12253255  0.12010108  0.11851827  0.11658314  0.11532513  0.11276207
 0.11130515  0.11036051  0.10608916  0.10543684  0.10316982  0.10083037
 0.09874446  0.09657788  0.09443052  0.09323912  0.09285246  0.09014736
 0.08969416  0.08799006  0.08689481  0.08421529  0.08325295  0.0816475
 0.08103946  0.07996107  0.07846385  0.07718043  0.07643295  0.07618583
 0.07467399  0.07427156  0.07381765  0.07229656  0.07134107  0.07078048
 0.06988577  0.06833023  0.06712822  0.06704883]
```

Figure 2-2 Snippet from codes: first 100 explained variance of training data

From Figure 2-2, we could see the first 100 explained variance of training data, where the values are actually in descending order. If we take all the components that have explained variance greater than and equals to 0.1, we will end up taking 72 principal components which explained 88.95% of data, let us use this k for our first experiment. On the other hand, if we take only those are greater than equals to 1.0, we will have k=10, our second experiment value.

2.1.3 Reconstruction of data

Thus, having known the rank k which we had chosen to represent the data, we can use it to reconstruct the matrix. There are two ways of doing it: reverting PCA or compress the original matrix using Singular Value Decomposition (SVD). Sum of Squared Error (SSE) of each method is calculated by a summation of squared of difference between raw data and reconstructed data.

2.1.3.1 "Reversed" PCA

Let k be the number of eigenvectors we want to use, **X** be the original matrix with size (m x n) and after centering it by substracting mean vector μ from each row, we get centered matrix $\mathbf{X_c}$. The PCA projections matrix with size (m x k), **Z** is given by **XV**, where **V** is the n x k matrix. In order to reconstruct X, we map it back to n dimensions by multiplying (dot product) **Z** with $\mathbf{V^T}$. Thus, we have $\mathbf{X}_{hat} = \mathbf{ZV^T} = \mathbf{XVV^T}$. Finally, we add the mean μ back to obtain the reconstruction matrix:

$$\hat{X}_{PCA} = XVV^T + \mu \quad (3)$$

Note that VV^T is identity matrix if only if all n eigenvectors are used.

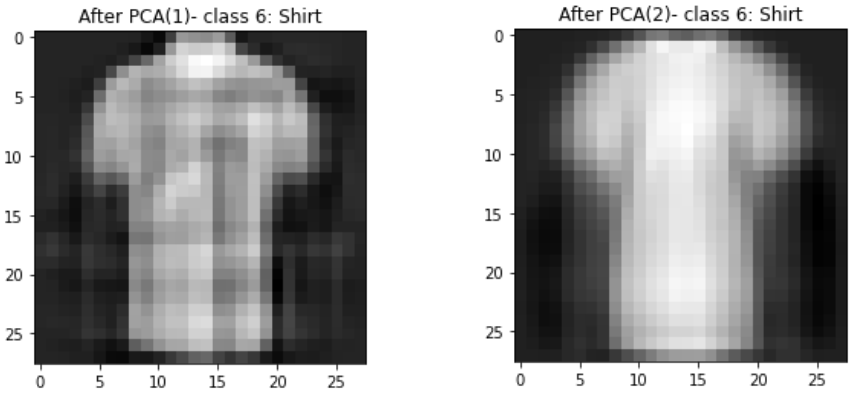

Figure 2-3 Images after dimensionality reduction and reconstructed using PCA: (left) when 72 principal components are reserved, and (right) when only 10 principal components are used.

Figure 2-3 shows the results of reconstructed first index of training data with two different value of k. SSE of two reconstruction of data using PCA is 226354.54 and 575042.52 respectively. It is seen that when more principal components are retained, the SSE is lower, and the image is clearer.

2.1.3.2 Singular Value Decomposition

First, we decompose the original matrix data X using SVD.

X (m, n) = **U** (m x r) **∑** (r x r) $\mathbf{V^T}$ (r x n)

Reconstructing X means we truncate columns of U, rows and columns of ∑, and rows of V^T to k components we want(Rajaraman & Ullman 2011, p. 446). Hence, reconstruction matrix is:

$$\hat{X}_{SVD} = U_{:,1:k}\sum_{1:k,1:k}V^T{}_{1:k,:} \quad (4)$$

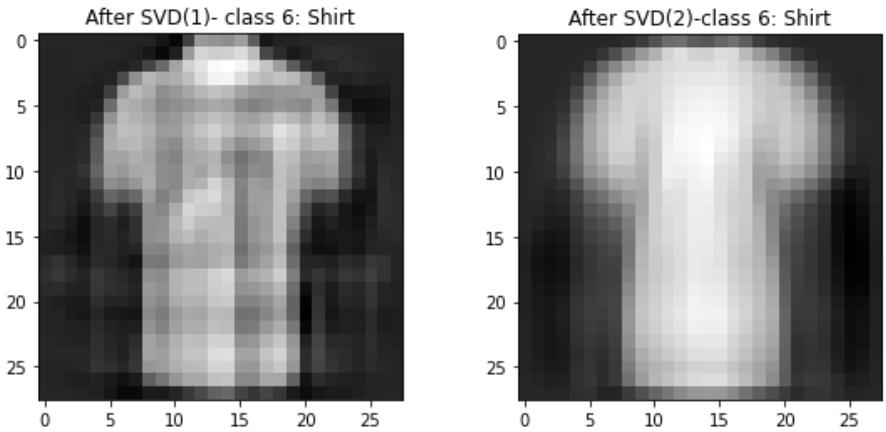

Figure 2-4 Images after dimensionality reduction and reconstructed using SVD: (left) when 72 principal components are reserved, and (right) when only 10 principal components are used.

From Figure 2-4, again we noticed that the results from the ones using PCA. In fact, the SSE of reconstructed data using SVD are slightly higher than that from PCA, where first experiment with 72 components is 226455.20 and second experiment with 10 rank is 577955.75. The performance of both methods is consistent.

Reconstructed training data with different rank using PCA were used in the rest experiment. Same transformation of training data should be apply to test data (Rajaraman & Ullman 2011, pp. 503-4). Hence, test data was processed and reconstructed using PCA method, with two different ranks as well.

2.1.4 Training-validation split for hyperparameter tuning

To validate the training performance, a validation set is needed to tune the hyperparameter. This is done by splitting the reconstructed data matrix (30000, 784) into two portions, training set and validation set. In this study, 80/20-split is used, which means 80% of data is accounted for training (24000, 784) and remaining 20% for validation (6000, 784). To avoid bias, shuffling of data together with the labels is performed before the split.

2.2 Classifier Algorithm

2.2.1 K-Nearest Neighbours (K-NN) Classifier

K-NN Classifier is a non-parametric classifier, which means, the number of parameters grow with the amount of training data (Murphy 2012, p. 16). The concept of this classification method is that, we classify the test point based on the k of the nearest training points.

First, distances of training data to first test point is calculated. Next, the chosen K of the nearest points (shortest distance) are reviewed. The test point is classified as the same group with most frequent occurrence of K neighbours. Finally, process is looped for all test points of the data given.

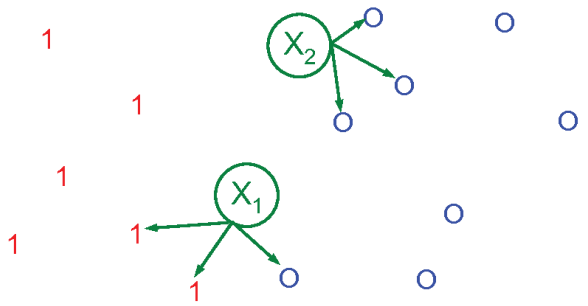

Figure 2-5 Illustration of KNN classification when K=3, adapted from Murphy (2012, p. 16). From the figure, test point X_1 is classified as category 1 and X_2 category O.

In this study, the distances are calculated using Euclidean distance equation (Rajaraman & Ullman 2011, p. 105) as stated below:

$$d(p,q) = \sqrt{\sum_i (p_i - q_i)^2} \tag{5}$$

2.2.1.1 Hyperparameter tuning

Value of K chosen affects the prediction result of our test point label. If K is too small, the classification is sensitive to noise points while on the other hand, we might classify the test point to the wrong category if K is too large, especially for the test point near the boundary of training points of the same category. Hence, the next question would be, what K should we choose? Here comes the hyperparameter K tuning process. By training the data with validation set, we plot a graph of accuracy against different K to visualise the best K to use in classifying test set.

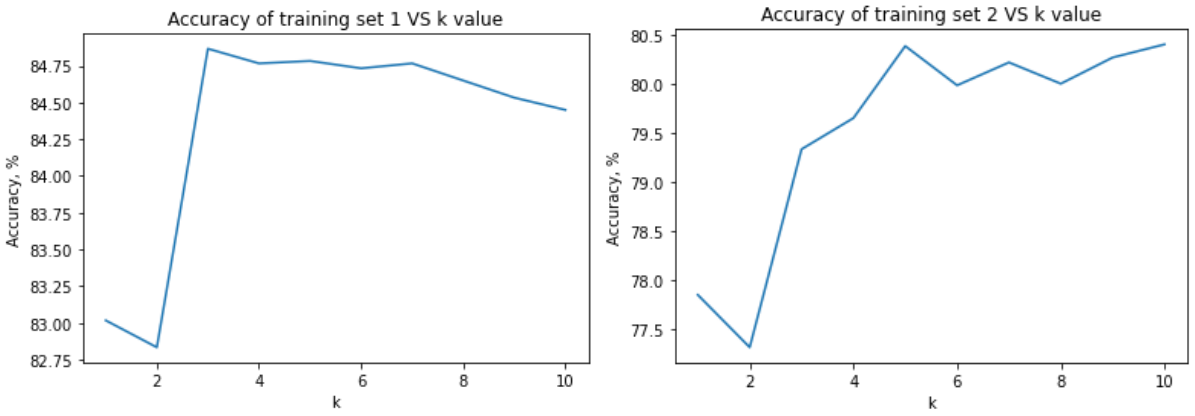

Figure 2-6 Hyperparameter tuning for two training sets: (left) Reconstructed training set using 72 principal components, and (right) Reconstructed training set using 10 principal components.

From Figure 2-6 (left), it is observed that the K that gives the best accuracy is 3 for Experiment 1. For Experiment 2, K=5 and K=10 both give high accuracy in predicting validation set (see Figure 2-6 right). However, K=5 is selected for this case because it costed lower computation time for large data.

2.2.2 Naïve Bayes Classifier

2.2.2.1 Bayes and Naïve Bayes

Based on Bayes' theorem, the posterior probability of i[th] class/category C given n attributes/variables A is computed by:

$$P(C_i|A_1, A_2, \dots, A_n) = \frac{P(A_1, A_2, \dots, A_n|C_i)P(C_i)}{P(A_1, A_2, \dots, A_n)} \quad (6)$$

The aim is to choose the value of C that maximize $P(A_1, A_2, \dots, A_n|C_i)P(C_i)$. However, computing $P(A_1, A_2, \dots, A_n|C_i)$ is tedious and expensive, where we need to find the product of $P(A_1|C_i) \times P(A_2|C_i, A_1) \times \dots \times P(A_n|C_i, A_1, A_2, \dots, A_{n-1})$. Hence, Naïve Bayes Classifier comes into point.

In Naïve Bayes concept, it is assumed that attributes A_n are independence to each other, thus the computation of likelihood probability is now

$$P(A_1, A_2, \dots, A_n|C_i) = P(A_1|C_i) \times P(A_2|C_i) \times \dots \times P(A_n|C_i) \quad (7)$$

There are three main types of Naïve Bayes (NB) Classifier, which are Bernoulli NB, multinomial NB and Gaussian NB, with respect to the type of distribution of data. In our study, Gaussian NB rather than multinomial NB is chosen because the variables of data are of real value or numerical instead of categorical (Murphy 2012, pp. 82-3).

2.2.2.2 Gaussian Naïve Bayes

Implementing Gaussian NB in classifying data is no other but replacing the likelihood equation above to

$$P(x|C_k) = \prod_{j=1}^{d} \frac{1}{\sqrt{2\pi\sigma_{j,k}^2}} \exp\left(-\frac{1}{2\sigma_{j,k}^2}\left(x_j - \mu_{j,k}\right)^2\right) \quad (8)$$

(Murphy 2012, p. 82)

Where x is the test data given that $x_j|C_k$ follows the distribution of $\mathcal{N}(\mu_{j,k}, \sigma_{j,k})$. It is noted that mean $\mu_{j,k}$ and standard deviation $\sigma_{j,k}$are those of training data.

The predicted label of test point is hence $argmax\ p(C_k) \prod p(x|C_k)$.

2.2.2.3 Log-sum trick

When x is a high dimensionality vector like in our case, $p(x_j|C_k)$ is often very small number since $P(x|C_k) = 1$ (Murphy 2012, p. 86). In order to avoid numerical underflow, we could perform the calculation by add log terms to the equation, and thus equation to predict test label becomes $argmax\ p(C_k) \sum p(x|C_k)$, which gives another advantage, as performing a sum is cheaper than multiplication.

2.2.2.4 Smoothing factor

To avoid zero dividing error, we add a smoothing factor, alpha = 0.001 when we calculate the variance for categorised training data.

2.3 Performance Evaluation

The primary evaluation metric for the classifiers is the accuracy, i.e. how many true predictions are there over all predictions? The equation of accuracy is as follow:

$$Accuracy, \% = \frac{number\ of\ correct\ classifications}{total\ number\ of\ test\ examples\ used} \times 100 \quad (9)$$

Other than that, the running time for each experiment is also considered in evaluating performance of a classifier. It is said to be the secondary evaluation metrics.

3 Experiments and Results

3.1 Results

A histogram was plotted based on the result of two classifiers used in three experiments (see Figure 3-1). “KNN” denotes the experiment using K-Nearest Neighbour classifier, and “GNB” for the one using Gaussian Naïve Bayes. The number behind the classifiers indicates the number of experiments: “0” is the experiment using unprocessed or raw data, “1” is the experiment using reconstructed data with 72 principal components, and “2” is the experiment using reconstructed data with 10 principal components.Figure 1-1

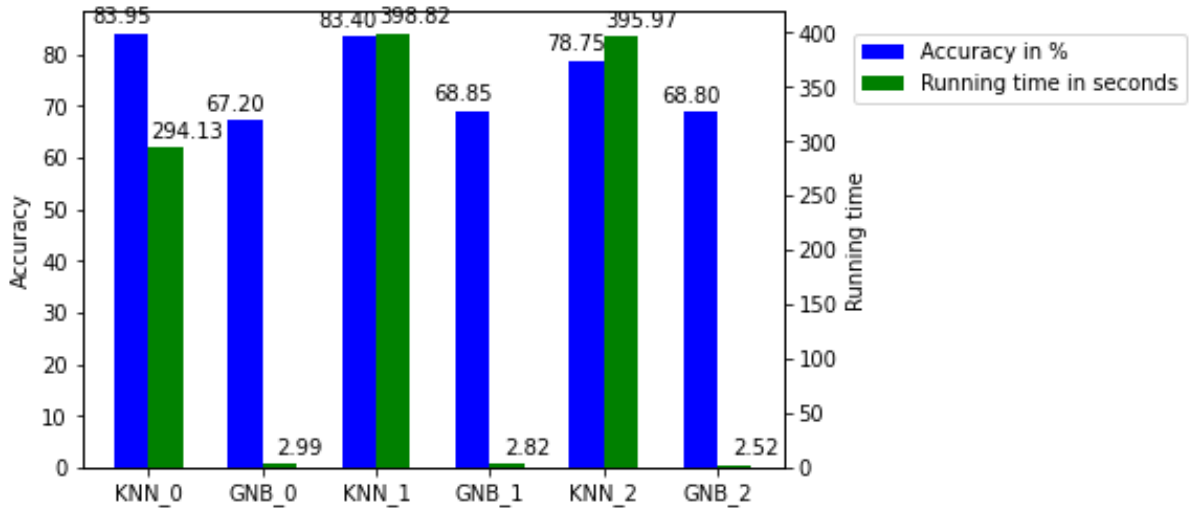

Figure 3-1 Comparison of performance of different experiment using two classifiers.

1) When K-NN classifier was used:
   First, let us compare the result Experiment 0 with Experiment 1 and 2. It is noted that accuracy of prediction was slightly lower when the training data was reconstructed. In fact, the running time for the raw data was around 26% faster than those using PCA-reconstructed data.
   Comparing Experiment 1 and 2, we see that, the prediction was 4.65% more accurate when more principal components were used (83.40% vs 78.75%), and their running time did not differ a lot, in which Experiment 1 took 2.85 seconds longer than Experiment 2.

2) When Gaussian Naïve Bayes was used:
   Comparing Experiment 0 with Experiment 1 and 2, it is observed that when raw data is used to train, the accuracy achieved was lower to those trained using processed data and it also took more time to compute result. In fact, learning using unprocessed data gave the lowest accuracy (67.20%) and longest time (2.99) among experiments on Gaussian NB.
   When Experiment 1 is compared with Experiment 2, we observe that including more components in reconstructing data yielded 0.05% higher accuracy than that when less components were included. Their computing time were similar as well, with

Experiment 1 slower than Experiment 2 0.3 seconds. Overall, effect of number of ranks of reconstructed data on the result is trivial.

3) Comparing 1) and 2):
In general, we can see that results from K-NN classification were better in terms of accuracy than Gaussian NB classification. On the other hand, although the accuracy achieved by Gaussian NB was around 19.2% lower than K-NN in average, the classification model was far more time-efficient than the latter, around 130.6 times faster in average.

The prediction of test data classification by KNN_0 is selected as it has the highest accuracy among all experiments and the time used to compute the output is the lowest among the experiments done using K-NN classifier. The output of Experiment KNN_0 was summarised by the confusion matrix shown in Figure 3-2.

```
[[158.   0.   5.   2.   3.   0.  22.   0.   2.   0.]
 [  0. 183.   0.   1.   0.   0.   0.   0.   0.   0.]
 [  6.   0. 164.   4.  23.   0.   9.   0.   0.   0.]
 [ 11.   0.   4. 169.  12.   0.  11.   0.   0.   0.]
 [  3.   0.  36.  10. 154.   0.  17.   0.   0.   0.]
 [  0.   0.   0.   1.   0. 156.   0.  19.   0.  14.]
 [ 34.   0.  22.   4.  13.   0. 113.   0.   4.   0.]
 [  0.   0.   0.   0.   0.   1.   0. 181.   0.  10.]
 [  0.   0.   4.   1.   1.   0.   4.   2. 215.   0.]
 [  0.   0.   0.   0.   0.   0.   0.   6.   0. 186.]]
```

Figure 3-2 Confusion Matrix of selected classification result, where the rows are the actual class and columns are predicted class.

The True Positive (TP) for each category is the diagonal element of confusion matrix, False Positive (FP) is the sum of values along the column minus TP value, False Negative (FN) is the sum of values along the row minus TP value, and True Negative is the sum of all values minus TP, FP and FN. With these values, precision (p), recall (r), and F-measure (F) of each class were calculated using equations (Fawcett 2006) below:

$$precision, p = \frac{TP}{TP + FP} \tag{10}$$

$$recall, r = \frac{TP}{TP + FN} \tag{11}$$

$$F - measure, F = \frac{2TP}{2TP + FN + FP} \tag{12}$$

The performance evaluation is summarised in

Table 3-1.

Table 3-1 Summary of performance of predicted output

Class	0	1	2	3	4	5	6	7	8	9
TP	158.	183.	164.	169.	154.	156.	113.	181.	215.	186.
FP	34.	1.	42.	38.	66.	34.	77.	11.	12.	6.
FN	54.	0.	71.	23.	52.	1.	63.	27.	6.	24.
TN	1754.	1816.	1723.	1770.	1728.	1809.	1747.	1781.	1767.	1784.
p	0.8229	0.9946	0.7961	0.8164	0.7	0.8211	0.5947	0.9427	0.9471	0.9688
r	0.7453	1.	0.6979	0.8802	0.7476	0.9936	0.6420	0.8702	0.9729	0.8857
F	0.7822	0.9973	0.7438	0.8471	0.7230	0.8991	0.6175	0.905	0.9598	0.9253

3.2 Discussion

Both K-NN and NB Classifiers are generative models while the former is non-parametric and the latter parametric (Murphy 2012, p. 270). K-NN is a supervised lazy learner model with local approximation, whereas NB is an eager classifier which is much faster than the former. Being a parametric model, parameters in NB learner are only the prior probability and conditional likelihood probability, which could be computed directly and no iterations for each test data are needed. On the other hand, in K-NN calculations of distance, sorting, and predicting are needed for each test point, making it to have large computation time when the data is big.

The nature of K-NN in local optimization makes it more accurate in classifying images: it focuses in finding similarity between observations and predicts by major votes; thus, decision boundaries are more complex. NB, on the other hand is a linear classifier and works better when the decision boundary is linear.

From Table 3-1, we can see that the selected method has the best performance in predicting class 1, which the precision, recall, and F-measure are the highest among all the predictions. On the other hand, prediction for class 6 images is relatively poor, with lowest score for precision, recall and F-measure. In fact, F-measure is the harmonic mean of precision and recall.

4 Conclusion

Based on the result, we can conclude that K-Nearest Neighbour Classifier has higher accuracy in classifying novel test input than Naïve Bayes Classifier. Secondly, the greater the number of principal components retained, the more they represented the data, hence the better performance it is in terms of accuracy, especially for K-NN Classifier.

It is suggested to revisit the reconstruction data as the work presented in this study did not work out (accuracy and running time are both lower than unprocessed data matrix) as expected. To improve the accuracy of classifiers, higher explained variance should be adopted

while performing PCA. Lastly, other Machine Learning methods such as the multinomial Logistic Regression is suggested to be implemented in the future and compare the results.

5 Appendix

5.1 Hardware configuration and software environment

The notebook is initially run in Google Colab environment without hardware accelerator such as GPU

vendor_id: GenuineIntel

cpu family: 6

model: 79

model name: Intel(R) Xeon(R) CPU @ 2.20GHz

cpu MHz: 2200.000

address sizes: 46 bits physical, 48 bits virtual

Local machine details

Hardware configuration

Windows edition: Windows 10 Home Single Language

System Processor: Intel(R) Core(TM) i5-4200U

CPU @ 1.60GHz 2.30GHz

Installed memory (RAM): 4.00GB

System type: 64-bit Operating System, x64-based processor

Software environment

version of Python: Python 3.8.1

version of Numpy: 1.18.4

version of h5py: 2.10.0

version of matplotlib: 3.2.1

5.2 How to run the code

How to use the code:

1. The code was arranged in a way that one should run it from the first cell to the last cell.
2. There are sections with “(must be run)” in the section head, these cells are important to output the final decision in predicting test labels i.e. the very last cell in the code, hence, they cannot be skipped.
3. There are also sections with “(can be skipped)” in the section head, if you do not run them, it would not affect the running of the following sections.
4. Under Chapter “2 K-Nearest Neighbour Classifier”, in the section named “Hyperparameter tuning (can be skipped)”, the long process of tuning was commented out, so no worries if you still like to run this section.
5. If you wish to visualise the comparison of different experiments in Chapter “4 Decision”, make sure to run cells in sections “KNN Results > other experiments” and “Gaussian Naïve Bayes”.

6 References

Fawcett, T 2006, 'An introduction to ROC analysis', *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861-74.

Murphy, KP 2012, *Machine Learning: A Probabilistic Perspective*, The MIT Press.

Rajaraman, A & Ullman, JD 2011, *Mining of Massive Datasets*, Cambridge University Press.