Introduction

The aim of the study is to optimise and build multiple classifiers to classify greyscale images into one of 10 classes and select the best classifier.
The classes are listed below:

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

This study is important as we will apply what we have learnt throughout our course on data processing and building classification models. This is to further strengthen our understanding in this field by experiencing how to efficiently build and optimise our classifiers and analyse results to yield the most accurate classifier.

Methods

2.1 Data Pre-processing

The dataset has he dimensions (30000, 784). This is a very large dataset thus would require large amount of computational power to run on a classifier. The first data pre-processing is dimension reduction by Principal Component Analysis (PCA). This is to reduce the dataset into a workable dataset without a large loss of information. The first step of Principal Component Analysis is to centre our data via zero mean. This is done by subtracting each observation in the training data by the mean of the training data.

Next, a co-variance matrix is formed to hold the co-variance between each feature. From there, the eigenvalues and eiganvectors of the training dataset are retrieved from the co-variance matrix by the linear algebra library. In order to find the amount of components to reduce, explained variance of each feature is computed. This is calculated as the proportion each features eigenvalue holds in the data. As this is a percentage it is multiplied by 100. The variances of all features are sorted and accumulated.

The method in this assignment to choose the number of components is on 95% cumulative variance explained. This yields 187 components, in other words 187 features. The eiganvector is reduce to hold 187 components, transposed and projected onto our training data. The training data now has the dimensions (30000,187). To keep consistency with our dimension reduction, the same projection matrix is used onto the test data.

Once dimension reduction is completed, the next step in data pre-processing is splitting our training data into a set of train and validation sets. In this assignment i have adopted a 70:30 ratio for train and validation respectively.

2.2 Classifier Methods Algorithms

The classifier method algorithms used are k-Nearest Neighbour (kNN), Gaussian Naive Bayes and Multinominal Logistic Regression.

2.2.i kNN

k-Nearest Neighbour used is based on a squared distance metric and fine-tuned hyperparameters to find number of k-neighbours. The algorithm calculates each record in the test data to every record in the training data to measure similarities between samples. The distances is then sorted by ascending order to find its nearest neighbors. Depending on k, the neighbour that occurs the most out of k distances is selected. [9]

2.2.ii Naive Bayes

The Naive Bayes in the assignment is based on calculating the Gaussian probability of each feature in a class. We have 30,000 number of observations thus can assume normality via the central limit theorem. An assumption is made that there is strong independence between features. Firstly, for each class the mean and variance is calculated for each feature that belongs in the corresponding class. The prior probability is calculated by find the proportion of each class in the data. The probability of each feature in a class is calculated by the formula [8] listed below.

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right) \tag{1}$$

the posterior probability is then calculated for each class by multiplying along each features probability given the corresponding class and then by the probability of the class. Formula given below [8].

$$\hat{y} = \text{argmax}_{k \in 1,2,...K}\Big(p(C_k) \prod_{j=1}^{D} p(x_j|C_k)\Big) \in \{C_k\}_{k=1}^{K} \tag{2}$$

To find the prediction, the class with the highest posterior probability is chosen.

2.2.iii Multi-nominal Logistic Regression

Lastly, Multi-nominal Logistic Regression is selected. Rather than sigmoid function in logistic regression, a soft-max function is used. The softmax function [8] is below:

$$h_w(x) = p(y = c|x; w_1, ..., w_c) = \frac{\exp(w_c^T x)}{\sum_{c=1}^{C} \exp(w_c^T x)} \tag{3}$$

Using gradient descent we aim to minimize loss. Gradient descent formula [8] given below.

$$w_j \leftarrow w_j - \alpha(\sum_{i=1}^{n}(h_w(x^i) - y^i)x^i + \lambda w_j) \tag{4}$$

The classifier then updates the weights by subtracting the gradient multiplied by the learning rate.

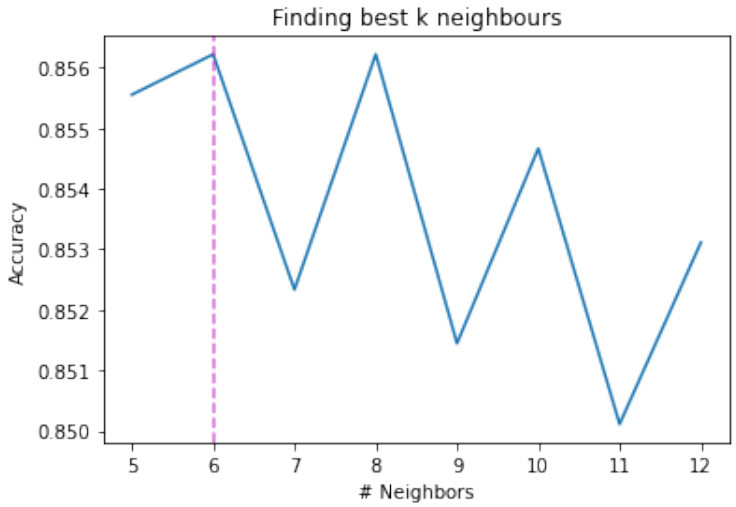
Experiments and Results

3.1 Experiment

3.1.i kNN

Fine-tuning was done in kNN to find the number of neighbours that would yield the highest accuracy from our train data. This is shown below.

Figure 1: Fine-tune to find optimal k



3.2 Result Comparisons

Below are the confusion matrix across all classifiers on the test data.

Figure 2: Confusion Matrix for kNN [5]

Predicted / Actual	0.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0	9.0	All
0	162	0	5	4	3	1	12	0	5	0	192
1	0	182	1	1	0	0	0	0	0	0	184
2	5	1	161	4	20	0	15	0	0	0	206
3	8	1	4	170	13	0	10	0	1	0	207
4	1	0	29	13	160	0	17	0	0	0	220
5	0	0	0	0	0	163	0	17	0	10	190
6	36	1	24	4	14	0	107	0	4	0	190
7	0	0	0	0	0	3	0	182	0	7	192
8	0	0	2	1	2	0	2	0	219	1	227
9	0	0	0	0	0	1	0	9	0	182	192
All	212	185	226	197	212	168	163	208	229	200	2000

Figure 3: Confusion Matrix for Naive Bayes [5]

Predicted / Actual	0	1	2	3	4	5	6	7	8	9	All
0	127	2	3	13	2	3	18	0	24	0	192
1	0	165	1	8	0	1	2	0	7	0	184
2	3	0	125	1	23	0	38	0	16	0	206
3	13	2	1	159	6	3	9	0	14	0	207
4	1	1	21	10	137	0	29	0	21	0	220
5	0	0	2	0	0	129	5	45	9	0	190
6	20	0	14	13	13	2	101	0	27	0	190
7	0	0	0	0	0	21	0	160	0	11	192
8	7	0	4	1	1	7	12	8	186	1	227
9	0	0	0	0	0	5	4	10	7	166	192
All	171	170	171	205	182	171	218	223	311	178	2000

Figure 4: Confusion Matrix for Multi-nominal Logistic Regression [5]

Predicted / Actual	0	1	2	3	4	5	6	7	8	9	All
0	153	3	5	4	2	0	23	0	2	0	192
1	0	181	0	3	0	0	0	0	0	0	184
2	4	3	148	4	22	0	24	1	0	0	206
3	11	7	3	167	4	1	12	0	2	0	207
4	1	2	31	13	152	0	21	0	0	0	220
5	0	0	0	0	0	179	0	9	1	1	190
6	33	1	20	9	24	0	100	0	3	0	190
7	0	0	0	0	0	12	0	170	0	10	192
8	0	0	0	3	1	3	5	2	212	1	227
9	0	0	0	0	0	3	0	6	0	183	192
All	202	197	207	203	205	198	185	188	220	195	2000

Figure 5: Accuracy across classifiers

	Train Accuracy	Test Accuracy
kNN	0.856222	0.8440
Naive Bayes	0.753333	0.7275
Multi-Logistic	0.851889	0.8225

3.3 Performance Review

After fine-tuning kNN it is shown in figure 1 that based on train data the best number of neighbours selected for kNN is 6. This means that the kNN algorithm will compute similarities between test sample with every training sample to its nearest 6 neighbours.

Gaussian Naive Bayes algorithm has a significantly lower accuracy compared to the other two algorithms. This is because, Gaussian Naive Bayes low performance can be attributed to the fact that the datasets features not strongly independent to each other. The algorithm is heavily dependent on the fact the features have no correlation amongst each other. Thus, it is the wrong assumption to use the Bayes Theorem formula to calculate the posterior probability, leading to a very low score.

Multi-nominal logistic regression results are sub par due to the fact that fine tune hyperparamters was not conducted. Adding in regularisation, making the learning rate dynamic, and running the regression in batches or stochastic may optimise the classifier to produce higher accuracy,

From above we can see that k-Nearest Neighbour yields the highest accuracy amongst the classifiers.

3.4 Future work and Improvement

Instead of slicing the data by first 70% for train and remaining 30% for validate, it would of been more appropriate to randomly select the train. This may of led to bias in our selection of train data leading sub opitmal hyperparameters

Finding kNN hyperparameters is very intensive, taking x minutes. validation samples. As this may of led to incorrectly fine tuning the parameters on a biased set of data.

kNN would have better run time performance if found a more efficient way to calculate distance between test data and training data. At the moment taking O(m*n), where m number of test records and n the number of training records.

Below is the computational time for each classifier.

Classifier	Fine-Tune	Training	Test
kNN	23min 22s	2min 51s	53.5 s
NaiveBayes	-	33.2 s	18.5 s
Multi-Logistic	-	18.9 s	18.5 s

Computational performance may also be due to hardware. This assignment was built on
MacBook Pro (Retina, 13-inch, Mid 2014)
Processor: 2.8 GHz Intel Core i5
Memory: 8 GB 1600 MHz DDR3
Graphics: Intel Iris 1536 MB
OS: macOS Mojave Version 10.14.6

Conclusion

In conclusion, k-Nearest Neighbour is the best classifier compared to Gaussian Naive Bayes and Multi-nominal Logistic Regression. However, it is also the most computational intensive classifier amongst them all. Gaussian Naive Bayes is a poor classifier due to is reliance on the assumption that features are independent. Multi-nominal Logistic Regression has potential to become a better classifier with parametric tuning and dynamic learning rate.

References

[1] juanpa arrivillaga. *What does axis = 0 do in Numpy's sum function?* 2017. URL: https://stackoverflow.com/questions/40200070/what-does-axis-0-do-in-numpys-sum-function (visited on 10/10/2020).

[2] Rohith Gandhi. *Naive Bayes Classifier*. 2018. URL: https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c (visited on 10/05/2020).

[3] Daniel Jurafsky and James H Martin. "Logistic Regression". In: *Speech and Language Processing* (Oct. 2019). DOI: 10.1075/ps.5.3.02chi.audio.2f.

[4] SSebastian Raschka. *What is Softmax Regression and How is it Related to Logistic Regression?* 2019. URL: https://www.kdnuggets.com/2016/07/softmax-regression-related-logistic-regression.html (visited on 10/18/2020).

[5] scls. *How to write a confusion matrix in Python?* 2010. URL: https://stackoverflow.com/questions/2148543/how-to-write-a-confusion-matrix-in-python (visited on 10/08/2020).

[6] Aman Sharma. *MULTINOMINAL LOGISTIC REGRESSION*. 2019. URL: https://towardsdatascience.com/ml-from-scratch-multinomial-logistic-regression-6dda9cbacf9d (visited on 10/13/2020).

[7] Saravanan Thirumuruganathan. *A Detailed Introduction to K-Nearest Neighbour*. 2010. URL: https://saravananthirumuruganathan.wordpress.com/2010/05/17/a-detailed-introduction-to-k-nearest-neighbor-knn-algorithm/ (visited on 10/17/2020).

[8] Nguyen Hoang Tran. *Lab codes in COMP5318 - Machine Learning and Data Mining*. Oct. 2020.

[9] Nguyen Hoang Tran. *Lecture notes in COMP5318 - Machine Learning and Data Mining*. Oct. 2020.