Report for COMP5318 Assignment 1

Yingbin Mo-- SID510202271

Songling Cheng-- SID500540969

Introduction

1. The aim of the study

This topic is a comprehensive examination of the various types of classification algorithms and basic steps in supervised machine learning. The aim of the study is to build several different types of classifiers with different combinations of parameters, and select the best classifier of them to categorize 5000 grayscale images of the size 28 x 28 into 10 corresponding categories. Therefore, the two main processes of the study are data pre-processing and hyperparameters tuning. Specifically, Principal Component Analysis (PCA) was used to perform dimensionality reduction on the dataset, while Grid Search and Cross Validation (GridSearchCV) are used to find the hyperparameters with the highest accuracy on the validation set among all the combinations.

2. The significance of the study

The dataset of the study is based on the Fashion MNIST dataset, which is provided by the research department of a German fashion technology company (Zalando) and is planned to replace the MNIST dataset directly ("GitHub - zalandoresearch/fashion-mnist: A MNIST-like fashion product database. Benchmark", 2021). In contrast to the MNIST dataset, the Fashion MNIST dataset is less about abstract symbols and more about tangible human necessities. Even classifiers with high recognition rates in the MNIST dataset perform less well in the Fashion MNIST dataset, which means that the Fashion MNIST dataset is more challenging, prompting analysts to make classifiers that are more realistic and meaningful.

Methods

1. Data pre-processing

Since the dataset is composed of 28*28 pixel images, each example has 784 dimensions. Without dimensionality reduction of the dataset, subsequent steps (e.g. tuning parameters and training the model) will take a lot of computational time. Therefore, we will adopt Principal Component Analysis (PCA) to pre-process the data set.

Before performing PCA, the dataset needs to be normalized, in order to make the machine learning model more suitable for the actual situation. Specifically, because all features are assumed to be equally weighted in PCA, if one feature has a particularly large value, which means it weights heavily in the overall error calculation (Yamamoto, Nakayama & Tsugawa, 2021). In this case, features with relatively small value are likely to be ignored, resulting in a large amount of missing information. Since the given data set already has values between 0 and 1, PCA can be performed directly.

Principal component analysis (PCA), also known as feature projection, is an unsupervised linear transformation technique that is primarily used for feature extraction and dimensionality reduction. PCA identifies the inherent patterns in the data based on correlations between features. The core

idea is to find the direction of maximum variance in high-dimensional data and project it onto a new subspace of the same or fewer dimensions as the original data ("A Comparison of Principal Component Analysis (PCA) and Linear Dicriminant Analysis (LDA) For Dimensionality Reduction", 2020).

According to the figure of explained variance vs number of dimensions, when the explained variance is greater than 95%, the growing trend of the curve tends to flatten out. This means that even if the remaining components are ignored, the error will not be significant. Therefore 95% of the components were selected, data dimension reduced from 784 to 188.

2. classifier methods algorithms

In this assignment, we import the 'GridSearchCV' class and 'cross_val_score' class from the 'sklearn.model_selection' module. All classifiers will be tuned for parameters in a 10-fold cross validation. The output includes best parameters, best estimator, best cross-validation score with best estimator, test set (first 2000 rows) score with best estimator and total running time.

a) K-Nearest Neighbour

K-Nearest Neighbours (KNN) is a supervised pattern classification learning algorithm. It assumes that similar things exist in very close proximity. In other words, similar things are very close to each other ("Machine Learning Basics with the K-Nearest Neighbors Algorithm", 2021). Therefore, it can find which class a new input (test value) belongs to when nearest neighbours are selected and the distance between them is calculated.

We import the 'KNeighborsClassifier' class from the 'sklearn.neighbors' module. Specifically, 'n_neighbors' is the number of neighbours used in the query. 'p' represents power parameters of Minkowski indicator.

b) Logistic Regression

Logistic regression is used to describe data, show the relationship between features and then calculate the probability of some outcome (Joy, 2020). It can be used for both regression and classification tasks (more common). There are three basic types of logistic regression, binary logistic regression, multinomial logistic regression and ordinal logistic regression

We import the 'LogisticRegression' class from the 'sklearn.linear_model' module. Specifically, 'C' represents the reciprocal of regularization coefficient λ (smaller value indicates a stronger regularisation). As for 'multi_class', the 'ovr' option represents one-vs-rest (OvR), while the 'multinomial' option represents many-vs-many (MvM). They do not differ in any way on binary logistic regression, but in multivariate logistic regression.

c) Support Vector Machine

Support Vector Machine (SVM) is a binary classification model, whose basic model is defined as a linear classifier with maximum interval on the feature space. Its learning strategy is interval maximisation, which can eventually be translated into the solution of a convex quadratic

programming problem (Mantovani, Rossi, Alcobaça, Vanschoren & de Carvalho, 2019). It is therefore valuable in mapping a linearly indistinguishable feature space to a more high-dimensional space.

We import the 'SVC' class from the 'sklearn.svm import' module. Specifically, 'C' represents the regularisation parameter, the strength of which is inversely proportional to its regularisation. 'kernel' refers to the kernel type used in the algorithm.

Experiments and Results

1. Comparing results

Algorithms	Best Hyperparameter	Test Accuracy	Running Time
K Neighbors Classifier	n_neighbors: 7 p: 1	84.05%	1h 32min 7s
Logistic Regression	C: 1, multi_class: multinomial	84.10%	27min
SVC	C: 10 kernel: rbf	88.90%	1h 2min 27s

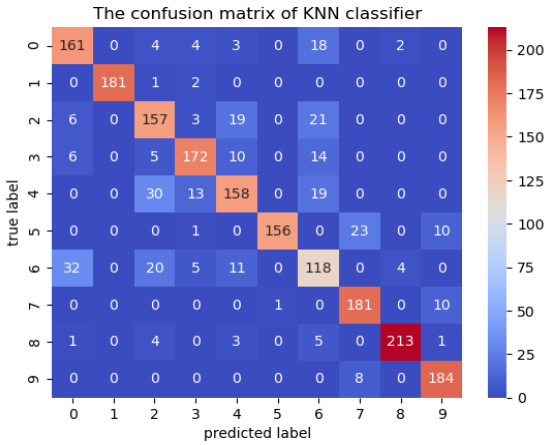
As for test accuracy, it can be seen that SVC has the highest accuracy rate of nearly 90%, while the remaining two algorithms performed poorly, with an accuracy rate of less than 85%. In respect of running time, Logistic Regression has the shortest running time, which is 1/2 that of SVC and is 1/3 that of K Neighbors Classifier.

2. Analysis result

a) K-Nearest Neighbour

According to the confusion matrix of KNN classifier, we can find that a total of 72 images were incorrectly classified as other categories in class 6, with nearly half of these errors coming from class 0. Therefore, the classifier identifies shirt(class 6) with a accuracy of 61%, the probability of correctly identifying a shirt among clothes is 62%.

This is not a good classifier. We believe that this situation can be attributed to the fact that the attributes are equally weighted. The distance of the sample is calculated based on all the attributes. Of these attributes, some are strongly correlated with the classification, some are weakly correlated and some (probably most) are not. This would mislead the classification process if the similarity of the samples was calculated according to the same weights of all the attributes.

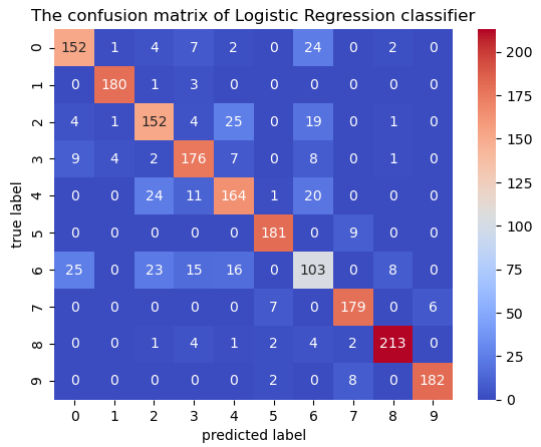


	precision	recall	f1-score	support
0	0.78	0.84	0.81	192
1	1.00	0.98	0.99	184
2	0.71	0.76	0.74	206
3	0.86	0.83	0.85	207
4	0.77	0.72	0.75	220
5	0.99	0.82	0.90	190
6	0.61	0.62	0.61	190
7	0.85	0.94	0.90	192
8	0.97	0.94	0.96	227
9	0.90	0.96	0.93	192
accuracy			0.84	2000
macro avg	0.84	0.84	0.84	2000
weighted avg	0.84	0.84	0.84	2000

Moreover, for high-dimensional samples or large sample sets, the time and space complexity is high and the time cost is O (mn), where 'm' is the spatial feature dimension of the vector space model and 'n' is the training sample set size.

b) Logistic Regression

According to the confusion matrix of ii. Logistic Regression classifier, we can find that a total of 87 images were incorrectly classified as other categories in class 6, with similar percentage of class 0, 2, 3 and 4. Therefore, the classifier identifies shirt(class 6) with a accuracy of 58%, the probability of correctly identifying a shirt among clothes is 54%.

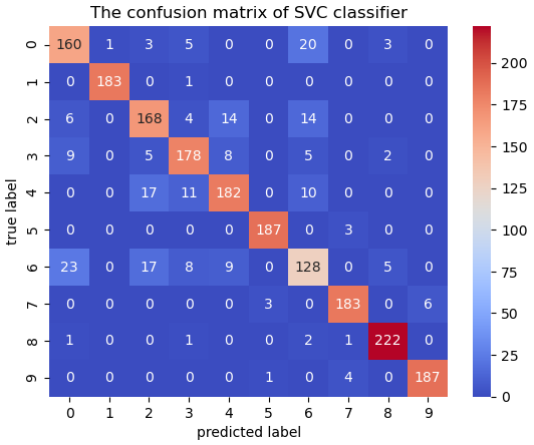


	precision	recall	f1-score	support
0	0.80	0.79	0.80	192
1	0.97	0.98	0.97	184
2	0.73	0.74	0.74	206
3	0.80	0.85	0.82	207
4	0.76	0.75	0.75	220
5	0.94	0.95	0.95	190
6	0.58	0.54	0.56	190
7	0.90	0.93	0.92	192
8	0.95	0.94	0.94	227
9	0.97	0.95	0.96	192
accuracy			0.84	2000
macro avg	0.84	0.84	0.84	2000
weighted avg	0.84	0.84	0.84	2000

From the results, we know that the performance of logistic regression is not very good when the features is large. Also, because it is essentially a linear classifier, it does not handle correlation between features well. The other reason is that the model representation is very simple and it is difficult to fit the true distribution of the data.

c) Support Vector Machine

Compared to the other two classifiers, SVC performs best in classifying the sixth class, and therefore has a significant improvement in overall accuracy. The model's overall ability (f1-score) exceeds 95% in several classes, with the lowest category (class 6) approaching 70%.



	precision	recall	f1-score	support
0	0.80	0.83	0.82	192
1	0.99	0.99	0.99	184
2	0.80	0.82	0.81	206
3	0.86	0.86	0.86	207
4	0.85	0.83	0.84	220
5	0.98	0.98	0.98	190
6	0.72	0.67	0.69	190
7	0.96	0.95	0.96	192
8	0.96	0.98	0.97	227
9	0.97	0.97	0.97	192
accuracy			0.89	2000
macro avg	0.89	0.89	0.89	2000
weighted avg	0.89	0.89	0.89	2000

The results of the scores show that it is effective in solving the classification problem for high-dimensional features, and still works well when the feature dimension is larger than the number of samples. One reason for this is that it uses only a portion of the support vector to make hyperplane decisions, without relying on the full data. In addition, a large number of kernel functions are available, allowing flexibility in solving various non-linear classification and regression problems.

Conclusion

Through the above analysis, we determined that the SVC algorithm would be our finally best classifier. However, due to time and capacity constraints, we did not make further improvements to the K-Nearest Neighbor and Logistic Regression method. For K-Nearest Neighbor, we can add weights to the neighbors to improve the model. Neighbors with a small distance from the target sample have a large weight, and vice versa. Thus, the proximity is taken into account to avoid misclassification due to one sample being too large. For logistic regression, the most effective method is still to adjust the regularization coefficients in the model to avoid overfitting or underfitting the training model, as the accuracy decreases in either case.

References

A Comparison of Principal Component Analysis (PCA) and Linear Dicriminant Analysis (LDA) For Dimensionality Reduction. (2020). *Journal Of Environmental Science, Computer Science And Engineering & Technology, 9*(1). doi: 10.24214/jecet.b.9.1.00107

GitHub - zalandoresearch/fashion-mnist: A MNIST-like fashion product database. Benchmark. (2021). Retrieved 24 September 2021, from https://github.com/zalandoresearch/fashion-mnist

Joy. (2020). What Business Strategy Does and what Management Accounting is Pursuing: A Logistic Regression Analysis. *International Journal Of Economics And Business Administration, VIII*(Issue 1), 124-133. doi: 10.35808/ijeba/413

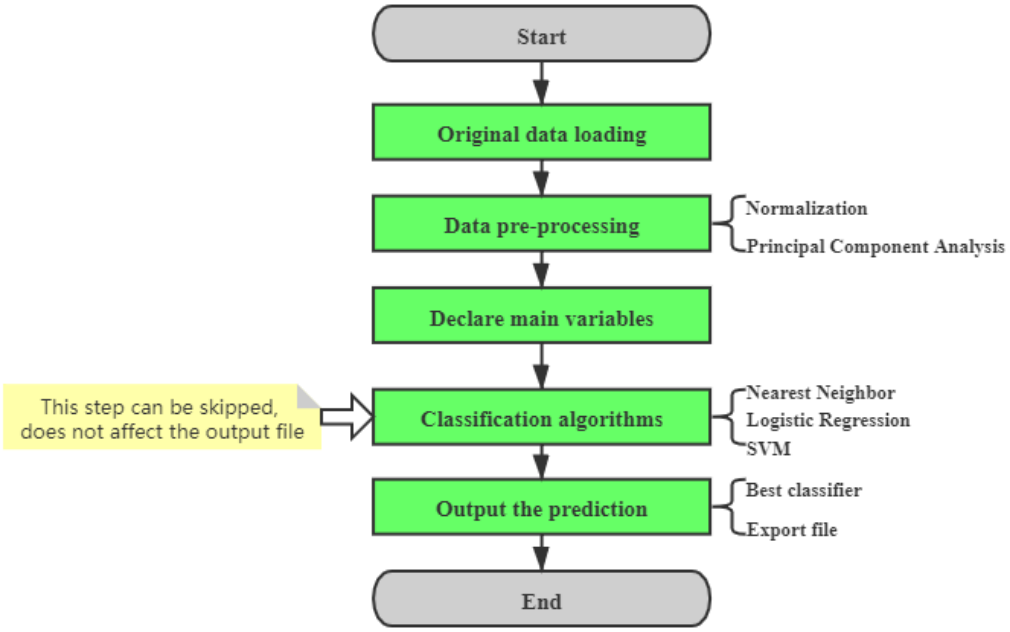
Machine Learning Basics with the K-Nearest Neighbors Algorithm. (2021). Retrieved 24 September 2021, from https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761

Mantovani, R., Rossi, A., Alcobaça, E., Vanschoren, J., & de Carvalho, A. (2019). A meta-learning recommender system for hyperparameter tuning: Predicting when tuning improves SVM classifiers. *Information Sciences, 501*, 193-221. doi: 10.1016/j.ins.2019.06.005

Yamamoto, H., Nakayama, Y., & Tsugawa, H. (2021). OS-PCA: Orthogonal Smoothed Principal Component Analysis Applied to Metabolome Data. *Metabolites*, *11*(3), 149. doi: 10.3390/metabo11030149

Appendix

1. Instructions



2. GridSearchCV run results

a) K-Nearest Neighbour

```
%%time
# Fine-tune hyper-parameters for KNeighborsClassifier
from sklearn.neighbors import KNeighborsClassifier

param_grid = {'n_neighbors': [3, 5, 7], # Create the parameter grid
              'p': [1, 2]}
print("Parameter grid: {}".format(param_grid))

grid_search = GridSearchCV(KNeighborsClassifier(), param_grid, cv=10,return_train_score=True)
# Use GridSearcCV on the training set
grid_search.fit(X_train, y_train)

print("Best cross-validation score: {:.4f}".format(grid_search.best_score_))
print("Best parameters: {}".format(grid_search.best_params_))
print("Best estimator: {}".format(grid_search.best_estimator_))
print("Test set (first 2000 rows) score: {:.4f}".format(grid_search.score(X_test, y_test)))
# Accuracy on test set of the model with selected best parameters
```

```
Parameter grid: {'n_neighbors': [3, 5, 7], 'p': [1, 2]}
Best cross-validation score: 0.8570
Best parameters: {'n_neighbors': 7, 'p': 1}
Best estimator: KNeighborsClassifier(n_neighbors=7, p=1)
Test set (first 2000 rows) score: 0.8405
Wall time: 1h 32min 7s
```

b) Logistic Regression

```
%%time
# Fine-tune hyper-parameters for Logistic Regression
from sklearn.linear_model import LogisticRegression

param_grid = {'C':[1,10],
              'multi_class': ['ovr', 'multinomial']}# Create the parameter grid
print("Parameter grid: {}".format(param_grid))

grid_search = GridSearchCV(LogisticRegression(random_state=14, max_iter =5000),
                           param_grid, cv=10, return_train_score=True)
# Use GridSearcCV on the training set
grid_search.fit(X_train, y_train)

print("Best cross-validation score: {:.4f}".format(grid_search.best_score_))
print("Best parameters: {}".format(grid_search.best_params_))
print("Best estimator: {}".format(grid_search.best_estimator_))
print("Test set (first 2000 rows) score: {:.4f}".format(grid_search.score(X_test, y_test)))
# Accuracy on test set of the model with selected best parameters
```

```
Parameter grid: {'C': [1, 10], 'multi_class': ['ovr', 'multinomial']}
Best cross-validation score: 0.8540
Best parameters: {'C': 1, 'multi_class': 'multinomial'}
Best estimator: LogisticRegression(C=1, max_iter=5000, multi_class='multinomial',
                   random_state=14)
Test set (first 2000 rows) score: 0.8410
Wall time: 27min
```

c) Support Vector Machine

```
%%time
# Fine-tune hyper-parameters for SVC
from sklearn.svm import SVC

param_grid = {'C':[10, 100],
              'kernel': ['poly', 'rbf']}# Create the parameter grid
print("Parameter grid: {}".format(param_grid))

grid_search = GridSearchCV(SVC(random_state=14), param_grid, cv=10, return_train_score=True)
# Use GridSearcCV on the training set
grid_search.fit(X_train, y_train)

print("Best cross-validation score: {:.4f}".format(grid_search.best_score_))
print("Best parameters: {}".format(grid_search.best_params_))
print("Best estimator: {}".format(grid_search.best_estimator_))
print("Test set (first 2000 rows) score: {:.4f}".format(grid_search.score(X_test, y_test)))
# Accuracy on test set of the model with selected best parameters
```

```
Parameter grid: {'C': [10, 100], 'kernel': ['poly', 'rbf']}
Best cross-validation score: 0.9004
Best parameters: {'C': 10, 'kernel': 'rbf'}
Best estimator: SVC(C=10, random_state=14)
Test set (first 2000 rows) score: 0.8890
Wall time: 1h 2min 27s
```

3. Operating environment

CPU: Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz 1.80 GHz
RAM: 8.00 GB (7.89 GB Available)
GPU: NVIDIA GeForce MX250
Software: Jupyter Notebook 6.3.0; Python 3.8.8