Algorithm Selection

This report summarises my findings on how the values of hyper-parameters in three different classification algorithms affect their accuracy when applied to a bank of images of clothes. The library Scikit Learn was used to implement these algorithms.

The K-nearest neighbour algorithm predicts the class of a sample by observing the classes of a number (k) of neighbouring samples. It is easy to visualise, and can be implemented for non-binary problems. It doesn't require a training step, and runs in a relatively short amount of time. For these reasons, it was chosen to classify the fashion-mnist dataset. K-nn has one hyperparameter: the number of neighbours used to make predictions. A number too small can lead to overfitting, but too large can lead to over-generalisation.

The Random Forest classification method makes predictions according to the outcome of a number of decision trees. It is known to perform well with high dimensionality and has a low risk of overfitting to the training data, which is why it was selected as a classification algorithm. The hyperparameters I chose to test were the number of trees, which increases precision of fitting, and randomness when bootstrapping the samples used when building trees.

Support Vector Machines classify data by finding the margins between classes. It also performs well in high-dimensional spaces, and the choice of kernels allows us to fit the SVM more closely to the data. The hyperparameters I chose to test were the kernel choice, to find which one was best fit for this data, and the regularisation parameter (called "C" in scikit-learn), which is directly proportional to how closely the algorithm fits to the training data.

Methodology

I chose which parameters to test by searching through the parameters built into the scikit-learn functions, and relating them back to the content of the course. By referring to the course material and looking at the default value suggested by scikit-learn, I inferred which scale to test on. For example in K-nn, I knew the optimal number of neighbours would likely be inferior to 10, otherwise the algorithm would be over-regularised. For the number of trees to implement in Random Forest, the default value is 100 in scikit learn, so I knew to test on a wider scale.

I found subdivising the data was not necessary for K-nn and Random Forest, however I chose to implement Principal Component Analysis with 95% of retained variance (having normalised the data beforehand) to reduce the computing cost of my SVM algorithm. At first, I attempted to perform SVM with the original dataset. This took hours, and did not make for optimal testing conditions. With PCA however, the computing time was significantly reduced.

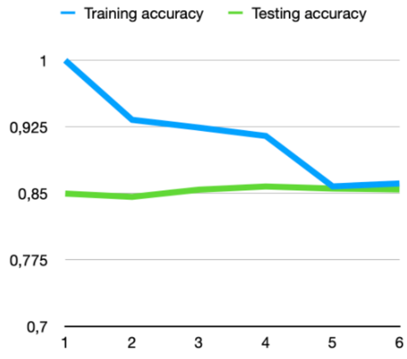
Results

For K-nn - best accuracy: 0.8577. Obtained with 4 neighbours.

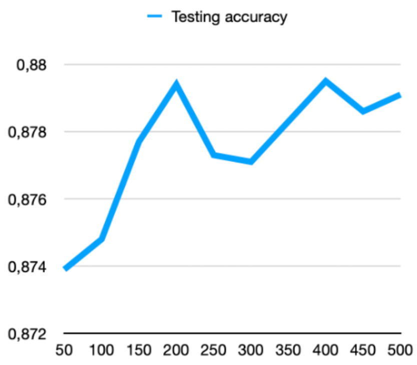
For Random Forest - best accuracy: 0.8795. Obtained with 400 trees and no randomisation.

For SVM - best accuracy: 0.9. Obtained with the “rbf” kernel and a regularisation factor of 13

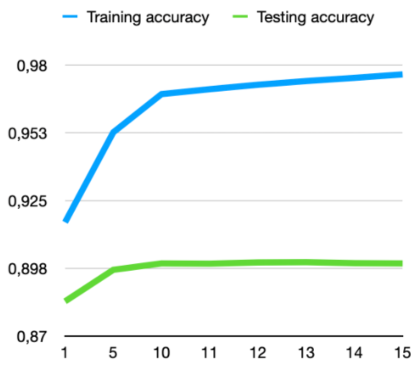
Figure 1: accuracy for different hyperparameters



(a) Accuracy per Number of Neighbours in K-nn



(b) Accuracy per Number of Trees in Random Forest



(c) Accuracy for Values of the Regularisation Factor in SVM

When observing the confusion matrix, I noticed "Shirts" (label 6) was the least successfully classified label for all three algorithms, sometimes getting classified as "T-Shirt", "Pullover" or "Coat". This makes sense as those items looks similar; on the other hand, labels like "Trousers" and "Sneakers" were more successfully classified.SVM performed the best out of the 3 for all the labels, with K-nn and Random Forest performing slightly worse in most of the labels.

Conclusion

For K-nn, in figure 1(a), we can clearly see the model overfitting to the training data with up to 5 neighbours. This was to be expected, especially if the classes are well defined. Although there is a slight increase in accuracy leading up to 4 neighbours, followed by a slight decrease, the variation is minor . The best accuracy obtained with K-nn is not great, since it doesn’t actually train according to the data, and possibly because of outlier sensitivity, as well as classes overlapping.

For Random Forest, I chose not to represent the randomising factor, as it didn't seem to positively affect the accuracy (the values stagnated, slightly decreasing from their "non random" counterpart at times). In figure 1(b), I showed only the testing accuracy, as the Random Forest algorithm isn't supposed to overfit to the data. We can also see that, although the number of trees does have some impact on the accuracy, that impact is negligeable. Besides, increasing the number of trees has a strong impact on runtime, so the price for this slight increase in accuracy is high. The best accuracy obtained with Random Forest is a slight improvement from K-nn, however it's not ideal. Perhaps it is due to some overfitting to the training data if the noise level is high. It's also worth noting that more trees being generated took significantly more computation time, so the optimal value being at least 200 trees comes at a price.

For SVM, I first tested each kernel. I selected the "rbf" kernel because it was clearly more accurate. From those tests, I concluded that kernels are an important factor for accuracy, as the best one allows the algorithm to fit to the data closely. I then tested the regularisation factor. As shown in figure 1(c), the regularisation increases fitting of the algorithm for the training data. It also slightly improves the accuracy of the classifications for the testing data. Although the accuracy of classification for the training data seems to keep improving, the accuracy for the testing data peaks at a value of 13 for the regularisation factor and seems to stagnate afterwards.

SVM had the best accuracy of the three classification algorithms. This is probably due to kernels, with which can be performed the "kernel trick”. This helps split the data into classes over multiple dimensions while being computationally efficient. Furthermore, coupled with PCA, SVM was relatively fast. For those reasons, SVM is my preferred classification algorithm of the three cases studied in this assignment. Had I not implemented PCA, I would have selected Random Forest as my favorite algorithm because it had a much shorter runtime than SVM on its own (for a reasonably low amount of trees), and has the second best accuracy.