Fashion data set analysis

Sabrina Yue Wang

December 31, 2017

1 Introduction and plan

The implementation utilises the sklearn python package which has inbuilt classifiers, those of which are tested are support vector machines and random forest models. The important thing to note that comparison between different classifiers is not as important as the preprocessing of the data to maximise the differences between the classes for easier distinguishability. Initially, the matrices for all the training datasets were naively put into the sklearn.svm.NuSVR classifier took almost a day. This is unrealistic and thus the main part of the code looks at a simple linear variable reduction analysis.

2 PCA and LDA Implementation and results

Sebastian Raschka provides simple examples of Principle component analysis (PCA) [1] and Linear disciminant analysis (LDA) [2] from which the code is adapted from. PCA is used to map the data onto a different subspace, often with lower dimensions. The effectiveness of this method depends on the magnitude ratios between the eigenvalues of the scatter matrix, a method of approximating the covariance matrix which is a generalisation of the variance for higher dimensions. If the largest eigenvalues are much larger than the rest, then it is 'safer' to neglect the smaller eigenvalues (and their corresponding eigenvectors). Once the eigenvalues and vectors are determined the training set can be mapped onto the new subspace using the eigenvectors. This reduction in complexity as well as method of separating properties amongst the total dataset increases the speed and accuracy of classifiers implemented on the transformed input data. Multiple Discriminant analysis is a similar method of separating features, except it is more concerned with the variance between different classes rather than the whole dataset. Both methods are implemented (see fashion.py).

The analyse function in fashion.py returns the score when the trained estimator (final one settled on was sklearn.ensemble.ExtraTreesClassifier()) is applied on the test data (which is transformed like the PCA and LDA testing sets and then scaled). Finally, the effects of the number of largest eigenvalues used on the classifier is investigated through a scan of the k's used in the PCA and LDA. The best possible resulting score from the analysis came out to be ≈ 0.71, when the kdim=kLDA=11 (NB they were not varied individually as it would take too long so this is estimate of best values. Also the number 12 can fluctuate between 9 and 12 as these values produce similar results). The whole program should run in approximately 6 minutes and the plots for the best choice of the number of eigenvalues used for PCA/LDA is shown in Figure 2, whereas the terminal outputs will be something similar to Figure 2:

Notes on the code that is commented out:

- lines 67-166 show that the confusion matrix (using svm.NuSVR) after almost a day of running turned out very inaccurate (please don't run it)
- Lines 201-205 and 356-360 are commented out because instead of taking the mean of each row of the input matrices before applying PCA and LDA, the vector of eigenvalues are calculated and used in the analysis instead, which gave worse prediction scores.
- Lines 292-298 is a quick plot showing the visual differences between the first 2 classes in the dictionary after PCA and LDA is applied.
- Lines 317-338 uses a KFold split (*n_splits* is self chosen) such that the cross validation method reveals how the classifier does against the data (i.e. the performance of the chosen classifier). It takes a few (2-3) minutes to run, outputting a score and confusion matrix. Note that the number of splits (used in KFold) can slightly change the outcome of how the perform seems to be. Too large of a split means that the number of combinations of each group is less, so avoid that. However, in the end when the fitting is done for the dataset this is neglected.
- lines 375-380 plots the average vector values of the first image of every class, showing they are quite different. Lines 381-393 plots the first N pictures of a few classes, showing large variation within the classes and thus there is a lot of overlap.

3 Other classification methods and future improvements

The random forest model is used for the classification as it is faster in speed compared to support vector machine models. In order to further improve the classifier, one can use other non-linear classification methods with non linear kernels, for example using deep learning methods with different layers (e.g. neural networks which sklearn does have inbuilt but sadly I didn't spend enough time studying it in the break :( ) as they have methods of feature extraction (e.g. some pictures may have sharper changes in curvature) that can help increase the distinguishability of class features. Finally for the preprocessing phase, there are fractal methods which make use of fractal characteristics (e.g. fractal

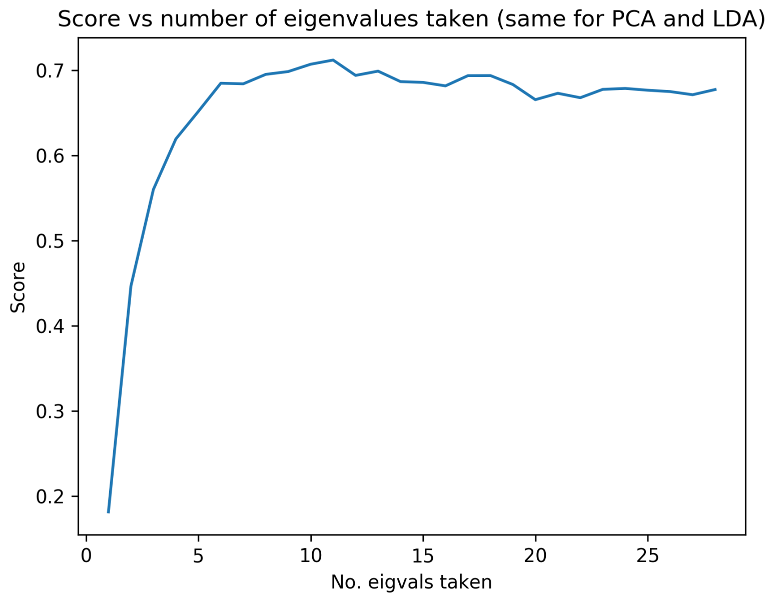
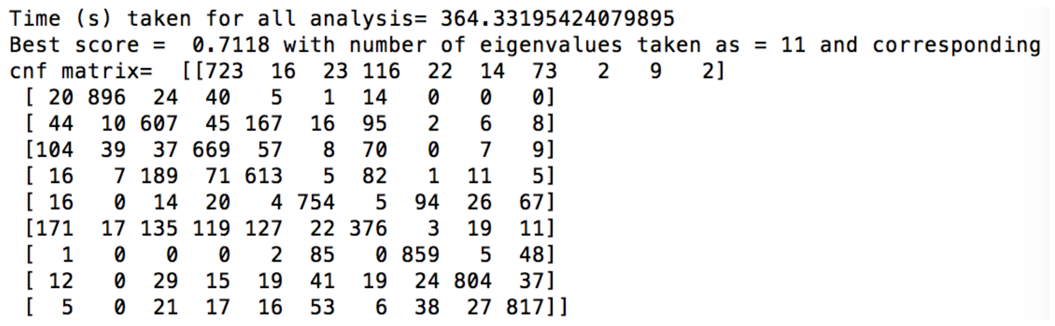


Figure 1: Python file fashion.py output figure: showing that if the number of eigenvalues used for PCA and LDA are taken to be the same then the best value is around 9-12 (usually 12)



```
Time (s) taken for all analysis= 364.33195424079895
Best score =  0.7118 with number of eigenvalues taken as = 11 and corresponding
cnf matrix=  [[723  16  23 116  22  14  73   2   9   2]
 [ 20 896  24  40   5   1  14   0   0   0]
 [ 44  10 607  45 167  16  95   2   6   8]
 [104  39  37 669  57   8  70   0   7   9]
 [ 16   7 189  71 613   5  82   1  11   5]
 [ 16   0  14  20   4 754   5  94  26  67]
 [171  17 135 119 127  22 376   3  19  11]
 [  1   0   0   0   2  85   0 859   5  48]
 [ 12   0  29  15  19  41  19  24 804  37]
 [  5   0  21  17  16  53   6  38  27 817]]
```

Figure 2: Terminal output when kdim=kLDA=12, using the ExtraTreesClassifier, showing the score and corresponding confusion matrix when predicting on the xtest and ytest datasets.

dimension) of the data and thus lead to picking out characteristics of data sets that are similar/hard to distinguish using the naked eye. However, this method often will enhance characteristics both between and within classes, improving and damaging the effectiveness of preprocessing respectively. Furthermore fractal methods usually are applied to data that has a lot of details such as market fluctuations and natural phenomena, which is not really applicable here but may be interesting to try out and see which effect dominates.

References

[1] S. Raschka, "Implementing a Principal Component Analysis (PCA)." http://sebastianraschka.com/Articles/2014_pca_step_by_step.html#drop_labels. [Online; accessed 31-December-2017].

[2] S. Raschka, "Linear Discriminant Analysis." http://sebastianraschka.com/Articles/2014_python_lda.html. [Online; accessed 31-December-2017].

NB: The url does not seem to work when I use bibtex so please either google Sebastian Raschka and the title of the page or copy the link by hand. Sorry for the inconvenience.