Contemporary Computer Science III – Machine Learning

Tom Robson - hzwr87

Experimental Approach

The algorithms are implemented in OpenCV, using code from the Github Repository provided [1], and can be run in machineLearning.py on the provided test and training splits of the datasets. The statistics were calculated using the Scikit Learn library [3] in separate files, machineLearningStatistics.py and grid.py, also supplied.

The algorithms selected for implementation are K-Nearest Neighbour and Support Vector Machines. K-fold cross validation was used for the experiments on both algorithms to ensure that the whole dataset is included for both training and testing. $K = 10$ was the value chosen to ensure uniform testing across all parameters for both algorithms, as this has previously been shown to be a good value. [2]

The decimal precision provided in the tests carried out for this report are constrained by the values that some of the Scikit Learn functions, such as classification_report(), which was used to generate the precision, recall and F1 values given below. It is also important to note that in the confusion matrices, some false classifications did occur for classes with a large number of occurrences, but due to normalisation, it appears as if they did not occur.

K-Nearest Neighbour (KNN)

The main parameter that can be changed in the KNN algorithm is K, or the number of neighbours considered. This was varied from 1 to 30 across all 10 folds, and the average precision, recall and F1 score across all 12 classes are given in Table 1, as well as the accuracy of classification across the entire dataset. The accuracy has also been graphed in Figure 1. The best values found for K from the experiments run are 1 and 3, with an accuracy of 95.62% and 95.59% respectively. The results of these per class is shown in Table 2, as well as the corresponding confusion matrices in Figure 2.

K	Precision	Recall	F1-Score	Accuracy
1	0.96	0.96	0.96	0.96
2	0.95	0.94	0.94	0.94
3	0.96	0.96	0.96	0.96
4	0.95	0.95	0.95	0.95
5	0.95	0.95	0.95	0.95
6	0.95	0.95	0.95	0.95
7	0.95	0.95	0.95	0.95
8	0.95	0.95	0.95	0.95
9	0.95	0.95	0.95	0.95
10	0.95	0.95	0.95	0.95
11	0.95	0.95	0.95	0.95
12	0.95	0.95	0.94	0.95
13	0.95	0.95	0.95	0.95
14	0.94	0.94	0.94	0.94
15	0.94	0.94	0.94	0.94
16	0.94	0.94	0.94	0.94
17	0.94	0.94	0.94	0.94
18	0.94	0.94	0.94	0.94
19	0.94	0.94	0.94	0.94
20	0.94	0.94	0.94	0.94
21	0.94	0.94	0.94	0.94
22	0.94	0.94	0.94	0.94
23	0.94	0.94	0.94	0.94
24	0.94	0.94	0.94	0.94
25	0.94	0.94	0.94	0.94
26	0.94	0.94	0.94	0.94
27	0.94	0.94	0.94	0.94
28	0.93	0.93	0.93	0.93
29	0.93	0.93	0.93	0.93
30	0.93	0.93	0.93	0.93

Table 1: KNN with varying K

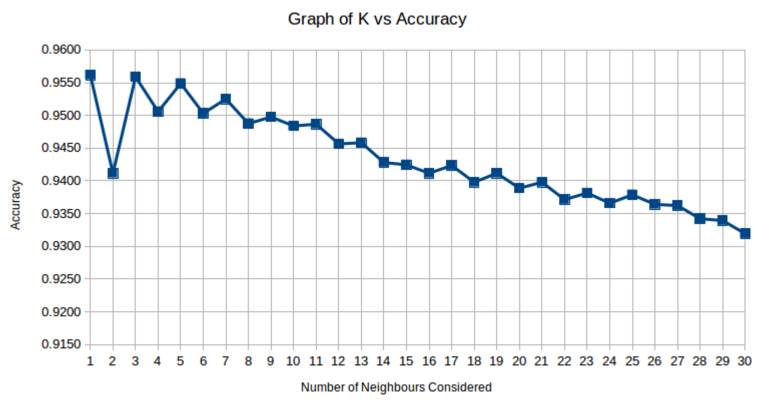
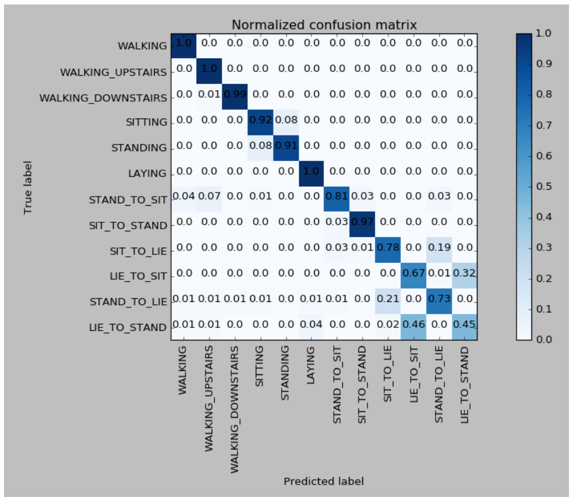


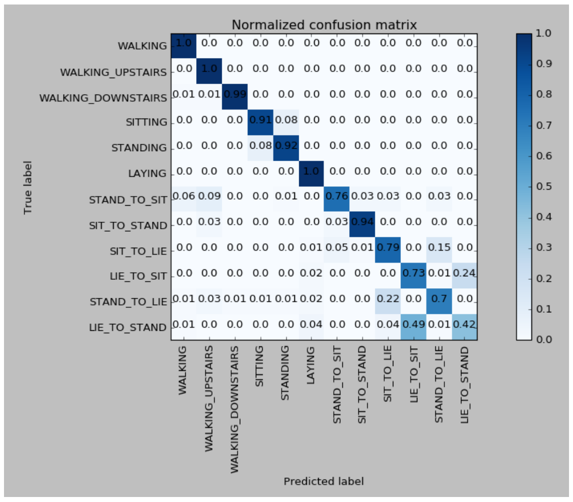
Figure 1: Graph of Results

Class	K=1			K=3		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score
Walking	0.99	1.00	1.00	0.99	1.00	0.99
Walking Upstairs	0.99	1.00	0.99	0.98	1.00	0.99
Walking Downstairs	1.00	0.99	0.99	0.99	0.99	0.99
Sitting	0.91	0.92	0.91	0.93	0.91	0.92
Standing	0.93	0.92	0.92	0.92	0.93	0.93
Laying	1.00	1.00	1.00	0.99	1.00	1.00
Stand to Sit	0.86	0.84	0.85	0.85	0.74	0.79
Sit to Stand	0.86	0.94	0.90	0.86	0.94	0.90
Sit to Lie	0.74	0.81	0.78	0.73	0.82	0.77
Lie to Sit	0.61	0.69	0.65	0.59	0.72	0.65
Stand to Lie	0.81	0.73	0.77	0.84	0.71	0.77
Lie to Stand	0.60	0.48	0.53	0.60	0.39	0.47
Average	0.96	0.96	0.96	0.96	0.96	0.96

Table 2: Tables of Class Specific Results



(a) K=1



(b) K=3

Figure 2: Confusion Matrices

Support Vector Machines (SVM)

In the SVM algorithm, there are many more parameters that can be changed. The most important of these is the kernel used, and each of these has different parameters associated with it. To evaluate these, a grid searching approach was employed, again using methods from Scikit Learn. A comparison of the Linear, Polynomial and RBF kernels on the optimal parameters found for each is included below in Table 3. A more thorough grid search was performed, but has been omitted here due to the shear volume of data generated. For all appropriate kernels, C was tested between 1 and 100000, as it was found to only have a great effect when changing on this scale. The same was true of γ, with this being tested between $10^{-1}$and 10^{-5} on the RBF and Polynomial kernels. Degree behaves differently for the Polynomial Kernel, and this was varied between 1 and 10. Many of the tests for each kernel yielded the same accuracy value, but only 1 set of parameters that generated the highest accuracy value to 3 decimal places is provided here.

Linear Kernel		RBF Kernel			Poly Kernel			
C	Accuracy	C	γ	Accuracy	C	γ	Degree	Accuracy
10	0.974	100	0.01	0.982	100	0.01	6	0.981

Table 3: Optimal Parameters and Accuracy values for each Kernel

As seen in Table 3, the best performing kernel was the RBF kernel. The precision, recall and F1 score per class with these settings are included in Table 4, as well as its confusion matrix in Figure 3.

Class	RBF Kernel		
	Precision	Recall	F1-Score
Walking	1.00	1.00	1.00
Walking Upstairs	1.00	1.00	1.00
Walking Downstairs	1.00	1.00	1.00
Sitting	0.98	0.98	0.98
Standing	0.98	0.98	0.98
Laying	1.00	1.00	1.00
Stand to Sit	0.94	0.87	0.90
Sit to Stand	0.91	0.91	0.91
Sit to Lie	0.78	0.78	0.78
Lie to Sit	0.75	0.73	0.74
Stand to Lie	0.81	0.81	0.81
Lie to Stand	0.73	0.70	0.72
Average	0.98	0.98	0.98

Table 4: Tables of Class Specific Results

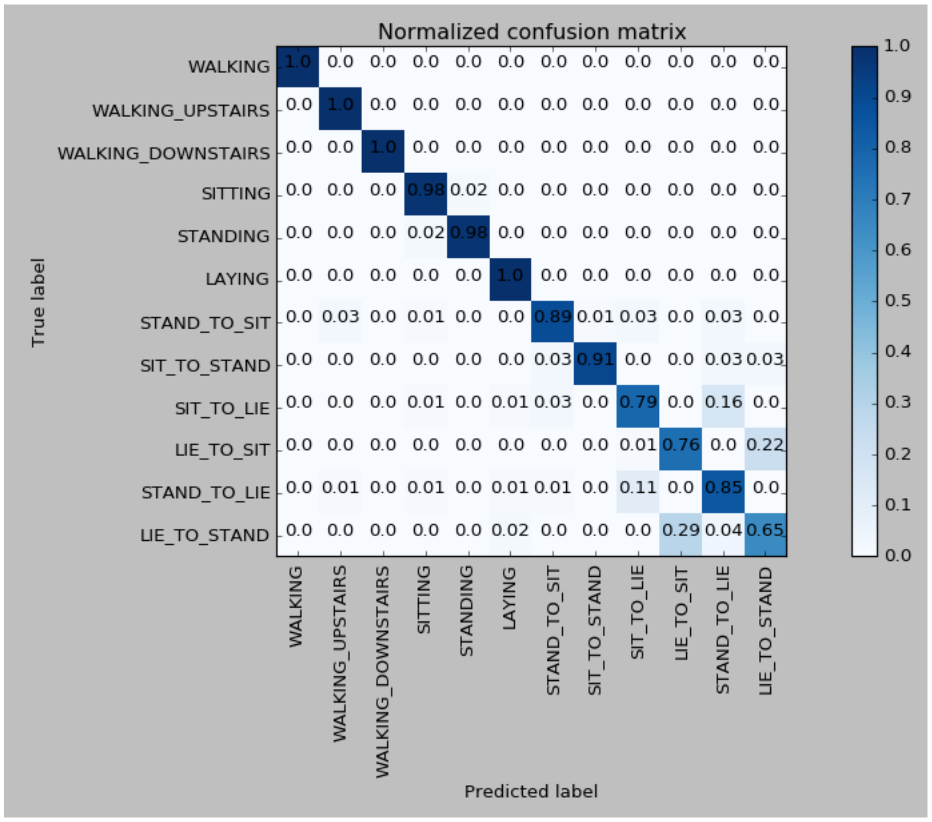


Figure 3: Confusion Matrix for SVM with RBF Kernel

Conclusions from Experimentation

We can see from these tests that the best method to use for this dataset would be SVM with the RBF kernel, with an accuracy rate of 98.2%. This is one of the slower kernels to run, but the increase in accuracy makes up for this. This is a classic example of the "No Free Lunch" theorem.

Ethical Issues

As these results are accurate, with an optimal value of 98.2% for SVM, these algorithms can reasonably be used to monitor behaviour, but not in critical situations. This level of accuracy is acceptable for this dataset and would be useful and practical for an application like an exercise monitoring system. In a system like this, the few false classifications generated by the algorithm will not be particularly important. However, if a system like this was used to monitor the health of elderly or infirm people, for example, an incorrect classification could be the difference between someone getting the help that they needed or not. While this could increase the coverage that people in this situation need, the fact that false classifications can occur introduce interesting ethical complications, such as whether a machine should really be responsible to making decisions about a person's health, and who is to blame if a mistake occurs. Even if the system is correct in the vast majority of cases, the unpredictability of the consequences of false classifications are a concern for many.

References

[1] Toby Breckon. python-examples-ml. *https://github.com/tobybreckon/python-examples-ml/*, 2016.

[2] Ron Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145, 1995.

[3] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.