Learning Human Activities from Body-Worn Motion and Magnetic Sensor Data

Project Report: Introduction to Machine Learning, Spring '19

Akhil Wadhwa (aw3509)[1] **and Suyash Sule (ss11524)**[1]

[1]First Year Master of Science Students, NYU Tandon School of Engineering

ABSTRACT

Wearable electronics are a promising technology that is taking over the consumer markets with an estimated valuation reaching over USD 20 billion. Motion and inertia based body-worn sensors form an important class of devices that are making this technology viable with a wide range of applications in robotics, medical diagnosis, gaming and virtual reality, telemedicine among others. In this project, we use different machine learning techniques like logistic regression, support vector machines and principle component analysis to detect the activity being performed by a human subject using sensor data. Contrary to what has been attempted in literature, we employ a simple pre-processing pipeline to achieve good classification accuracy and provide signal processing analysis to explain why it works.

1 Dataset Description

The 'Daily and Sports Activities Data Set' is borrowed from the University of California, Irvine - Machine Learning Repository. It consists of well organized motion and inertial sensor data corresponding to 19 different activities like walking, standing, sitting, playing basketball, exercising on stepper, rowing, etc. The sensor data for each of these 19 activities has been measured for 8 different persons, 4 males and 4 females, each performing the activity for 5 minutes.

The sensors involved in capturing the motion data are embedded in three kinds of devices: accelerometers, gyroscopes and magnetometers, each of which records readings along three orthogonal axes: x, y and z. Accelerometers measure the linear acceleration and hence, the translational motion in the body. Gyroscopes, on the other hand, measure the angular velocity of a body, thus capturing the rotation of the body and magnetometers measure the Earth's magnetic field strength giving the alignment of the body.

There are 5 of these sensor units, each installed with all 3 of these devices, planted on the torso(T), right arm (RA), left arm (LA), right leg (RL) and left leg (LL) of each of the human subjects. This results in a total of 3 (per device) $\times$ 3 (per axis) $\times$ 5 (per body part) $=$ 45 readings. The subjects perform the activities for a total of 5 minutes. The sensors record measurements at a frequency of 25 Hz, thus resulting in a total of $25 \times 5 \times 60 = 7500$ reading per activity per person and this holds for each of the 45 sensors. For ease of storage and reading, these 7500 readings are saved in 60 text files called segments, each with 125 samples.

2 Problem Formulation

This is a classification problem as we wish to train and develop a model which when fed with the sensor readings as input, spits out 1 of the 19 activities as output, essentially detecting what activity the subject is performing. The dataset used in this problem is very well organized into readings corresponding to

different activities, persons and time samples against the different sensors recording data. Thus, the data matrix can be viewed as having time samples as rows and sensor label (or features) as columns. We thus have 45 features and we wish to classify based on the 45-length feature vectors. The data can be represented as follows, where $a = 1,\ldots,19$ are the activities and $p = 1,\ldots,8$ is the person performing it.

$$\bar{d}^{(a)(p)}[n] = d_j^{(a)(p)}[n], \quad j = 1,\ldots,45$$

However, this raw data cannot be directly used to train the model as the data samples involved are time series and a sensor reading cannot directly be used to tell what the person is doing. For example, while playing basketball, the subject could be initially standing waiting for the ball, which makes the activity indistinguishable from other activities like standing or even certain instances of jumping merely from the sensor readings. Similarly, the sensor data at the start of rowing can be very similar to that of sitting, thus muddling the classification of these two activities. Therefore, the first set of data processing must involve converting these time series into values that capture what happens overall in an activity, or at least over a significantly long time window.

Classification: Assign class $k \in \{1,\ldots,19\}$ for each processed vector $\bar{s} = g(\bar{d}^{(a)(p)}[n])$

3 Pre-processing of the Time Series Data

3.1 Approach followed in the References Papers

The research papers cited in the references follow a complex set of operations for feature extraction which lack physical meaning. For each of the 5s long segment, they find the mean, minimum, maximum, skewness and kurtosis for each of the 45 sensors. They also compute and save the 5 largest DFT coefficients, corresponding frequency values, and 11 auto-correlation samples. This results in a total of $(5+5+5+11)\times 45 = 1170$ features. Normalization and subsequent principle component analysis (PCA) for dimensionality reduction brings down this number to 30.

$$\text{mean: } \mu_x = \frac{1}{125}\sum_{i=0}^{124} x_i$$

$$\text{skewness: } s_x = \frac{1}{125\sigma^3}\sum_{i=0}^{124}(x_i-\mu_x)^3$$

$$\text{kurtosis: } \kappa_x = \frac{1}{125\sigma^4}\sum_{i=0}^{124}(x_i-\mu_x)^4$$

$$\text{DFT: } X(k) = \sum_{i=0}^{124} x_i e^{-2\pi jki/125}, k = 0,\ldots,124$$

$$\text{Autocorrelation: } R_{xx}(\tau) = \frac{1}{125-\tau}\sum_{i=0}^{124-\tau} x_i x_{i+\tau}, \tau = 0,\ldots,124$$

The advantage of this pipeline is that it computes the first, second, third and fourth order moments, along with the ordered statistics min and max which helps capture most of the statistical behavior of the series in the time window. Along with the prominent frequency response, it is able to assign features

that effectively capture patterns in the data as seen in the proportion of variance (PoV) using principal component analysis (PCA).

However, the downside of this approach is that the extracted features no longer hold any physical meaning, making it hard to attribute the performance of this model to any physical phenomenon and thus develop intuition. Moreover, the pipeline is complex, can be computationally expensive, and the features before PCA are still functions of the 5s time windows.

3.2 Our Approach

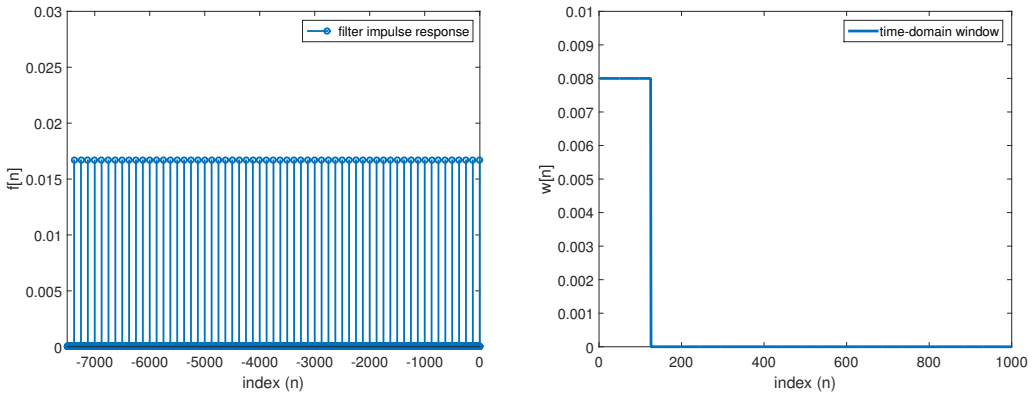
Since the time dependency of the data samples poses the major obstacle in learning, we tried to simplify the pipeline and observe how the classification accuracy varies from that of the aforementioned pipeline. We decided to stick to the original 45 sensor readings as features and process the data vectors to capture what happen over different instances of time. We employ a simple strategy of finding the mean of data samples located at a periodic spacing of 125 samples, so that we convert the 7500-samples time series to a sequence of 125 samples, wherein each sample captures exactly one sample from each of the 60 5s-window segments.

Let $d_j[n]$ be the data matrix for a person performing a particular activity, where $j = 1,\ldots,45$ is the index for features and $n = 0,\ldots,7500-1$ is the sample number. Let $s_j[n],\ \ n = 0,\ldots,124$ be the sequence after processing as per described above, so that:

$$s_j[n] = \frac{1}{60}\sum_{i=0}^{59} d_j[n+125i] \quad n = 0,\ldots 124 \text{ and } \quad \forall j$$

This way, each $s_j[n]$ captures what happens in each of the 5s-time window. We are going to drop the index j for further analysis as it will hold for all the features.

3.3 Analysis of the Data Processing



(a) Filter $f[n]$ **(b)** Time window $w[n]$

This operation of $d[n] \mapsto s[n]$ can be viewed as a filtering operation using a linear time invariant (LTI) anti-causal digital filter, followed by a time-windowing operation. Let $f[n]$ be the filter and $w[n]$ be the time window, defined as:

$$f[n] = \frac{1}{60}\sum_{i=0}^{59} \delta[n+125i]$$

where $\delta[n]$ is the Kronecker delta function. Thus, $f[n]$ has a value of $\frac{1}{60}$ at $n = 0, -125, \ldots, -125 * 59$ and 0 everywhere else. Thus,

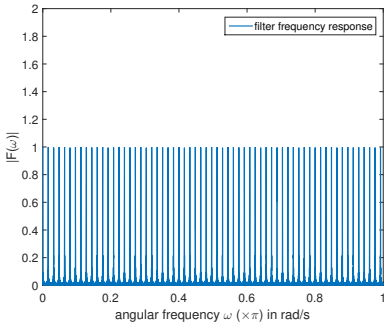
$$s[n] = d[n] * f[n] = \sum_{\tau} d[\tau] f[n - \tau]$$

Note that $s[0], \ldots, s[124]$ gives us the desired sequence. However, the above filtering also results in other unwanted samples, which can be discarded using a time-domain windowing operation using $w[n]$:

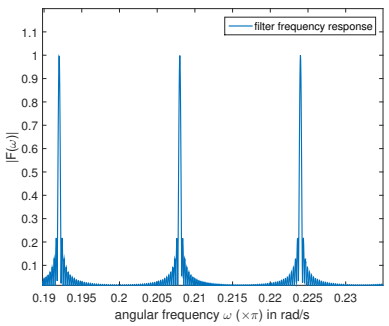
$$w[n] = \begin{cases} 1 & \text{if } 0 \leq n \leq 124 \\ 0 & \text{otherwise} \end{cases}$$

Element-wise multiplication $s[n] \odot w[n]$ gives us the final sequence that is used for training and testing the model.

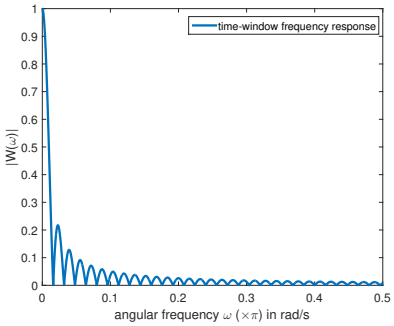
3.4 Frequency Domain Analysis



(a) $|F(\omega)|$



(b) $|F(\omega)|$ closer view



(c) $|W(\omega)|$

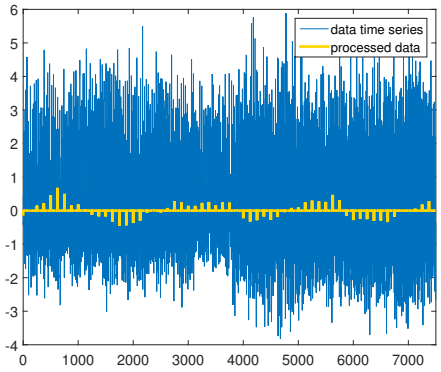
The filter $f[n]$ can be viewed as an upsampled version of the standard moving average filter of length 60. The frequency response of a moving average filter is the digital sinc function, thus the the frequency response of $f[n]$, say $F(\omega)$ will be a compressed version of the digital sinc, which looks like a series on 60 peaks in the range $[0, \pi)$. In the frequency domain, the data signal will multiply with $F(\omega)$, resulting in the selection of periodic frequency components $\omega = 0, \frac{\pi}{60}, \ldots, \frac{59\pi}{60}$. This helps in capturing the harmonics that define motions like the swinging of arms during exercising, swift movement of legs during running compared to walking, or even the leaps taken by the body during jumping or playing basketball. This explains why the simple yet effective operation that we employed gives good training and validation accuracy.

On the other hand, the time-domain windowing will correspond to convolution with the window frequency response $W(\omega)$.

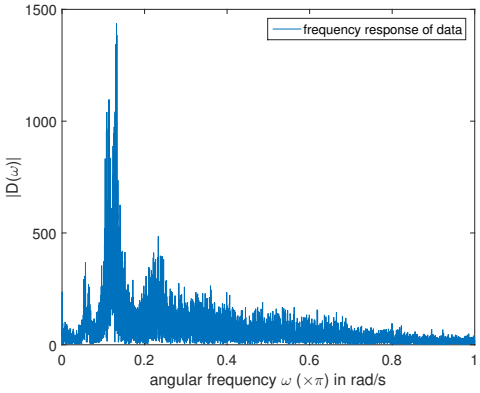
3.5 Effect on Data

The following figures show the data time series for walking in a parking lot on the right arm x-acceleration sensor, the processed sequence upsampled to equal the length of original time series, and the frequency response of the data before processing.The frequency response shows the prominent harmonics at $0.15\pi \equiv \frac{0.15\pi}{2\pi} \times 25 \approx 1.9Hz$ frequency of the swinging arm.

The next two figures show how the frequency response of the data changes when filtered with $f[n]$ and then time-windowed with $w[n]$. It can be observed that the processing captures the harmonics, thus

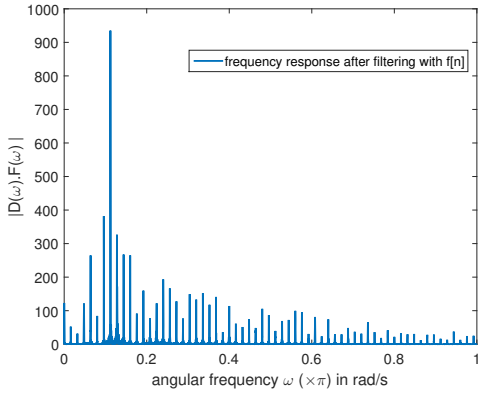


(a) $d_{10}[n]$ for A9

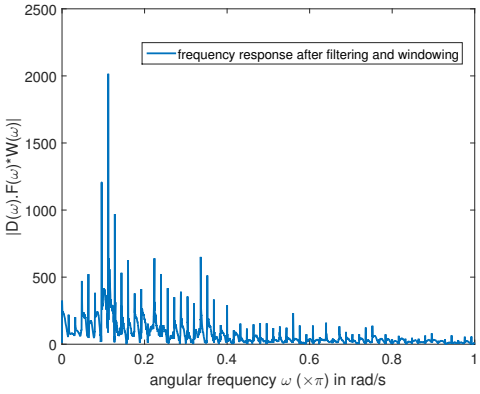


(b) $|D(\omega)|$ before processing

retaining useful information of the time series while eliminating time dependency of data at the same time. This explains why this simple yet effective data processing pipeline works and gives very good classification accuracy.



(a) $|D(\omega) \odot F(\omega)|$ frequency response after filtering



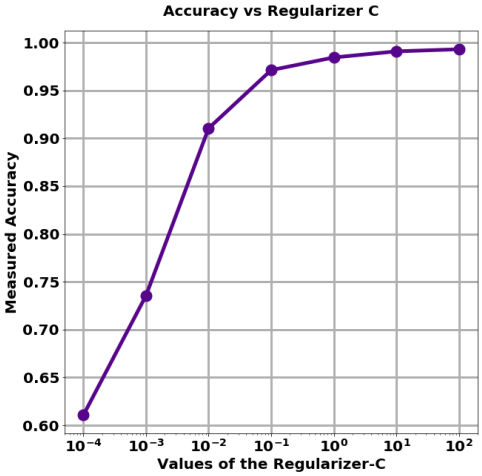
(b) $|D(\omega) \odot F(\omega) * W(\omega)|$ frequency response after filtering and windowing

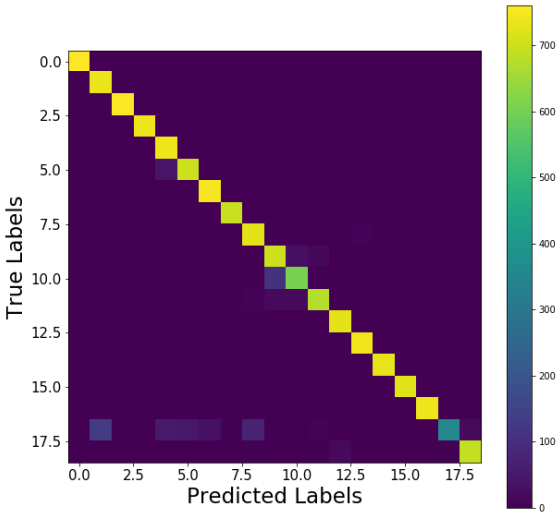
4 Logistic Regression

As discussed above, we have a total of 45 classes for classification and about 19,000 data point for testing and training. So, we start with logistic regression as our first classifier.

As shown in the Python notebook, we are using normalized data, which has a zero mean and a unit variance for logistic regression. We fit the data on the training split and then predict the labels for the test split using predict routine. Then we compare the accuracy by comparing the labels. For logistic regression, we got an approximate accuracy of 97%.

Subsequently, printing the confusion matrix, we see that almost half of the classes were predicted with 100% accuracy. Most errors were found in class 18 with about 250 data point misclassified. Then, in order to choose the optimal inverse regularization factor (C), we plot the accuracy metric with respect to C ranging from 10^{-4} to 10^{2}.





5 Principle Component Analysis and Visualization

Next, we apply principal component analysis to try and reduce the dimensionality of the data, while capturing most of the information in the original data. Again, to start with, we define the PCA transform and then fit it on the normalized data and print the singular values. Next, we plot the portion of variation (PoV) as a function of number principal components. In order to find the PoV we directly use routine 'cumsum'.

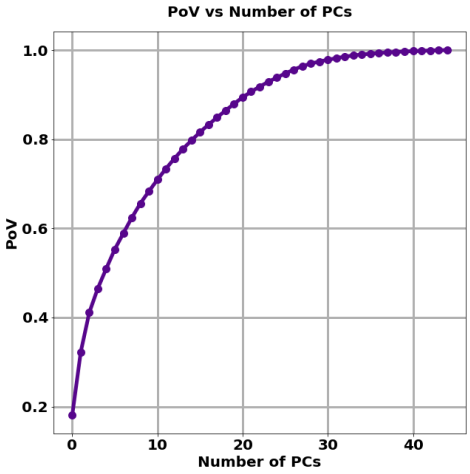
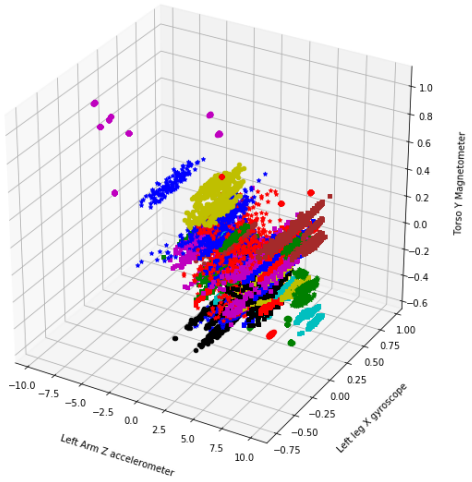


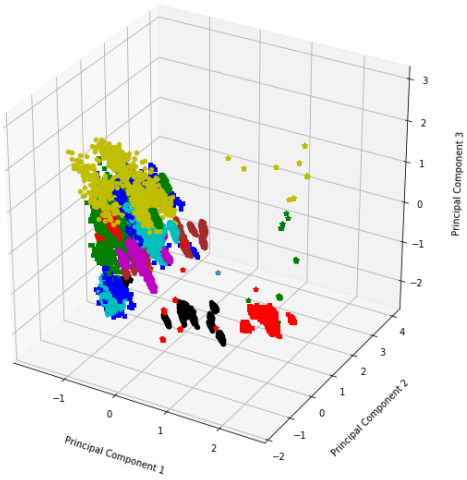
Figure 6. PoV vs number of principal components

From the plot, one can see that, it is an increasing plot with every principal component added until it saturates at about 35. As we have learnt in class, for some datasets, we experienced a very high shoot in the PoV for a much smaller number of dimensions, indicating that a few values of PC can capture almost all of the information in the original data. However, in our case, one can observe a much slower increase in PoV with the number of principal components. This shows that we cannot effectively reduce the dimensionality without loss of information.

Just to show how our data is distributed, we take measurements on all the three axes from three different sensors fitted on different parts of the body and plot all the data points. We can observe that our data is not very well separated in 3 dimensions. A similar trend is observed on plotting the data points along the first 3 principal components. The data still does not look separable, which makes sense given that the data points are 45-dimensional and the PoV plot shows that 3 PCs capture only about 40% of the information.



(a) Scatter plot using 3 features



(b) Scatter plot using 3 PCs

6 Support Vector Machines

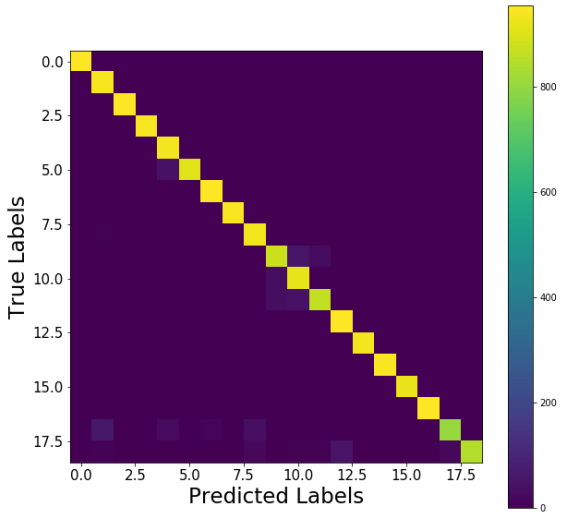
Before fitting a support vector machines classifier to our data, we split our data for test and train. Then, we remove the mean from each of the data sets and divide by the standard deviation, in order to normalize the data. Again, we form a model, fit the model with the training data and then predict the labels using the test data. Consecutively, we find the accuracy by comparing the predicted labels with the original. In our case, we trained our model on 1000 points and used the rest for testing. We get an accuracy of about 97%.

Then, we go on to printing the confusion matrix. From the confusion matrix, one can extract information about the miss-classified labels. In our case, many of the class 12 labels were miss-classified to belong to class 10 and class 11. If we have a close look at what these classes represent, we can make out that these classes are very similar in many aspects. For example: walking on a flat surface and on a slightly inclined plain are very similar.

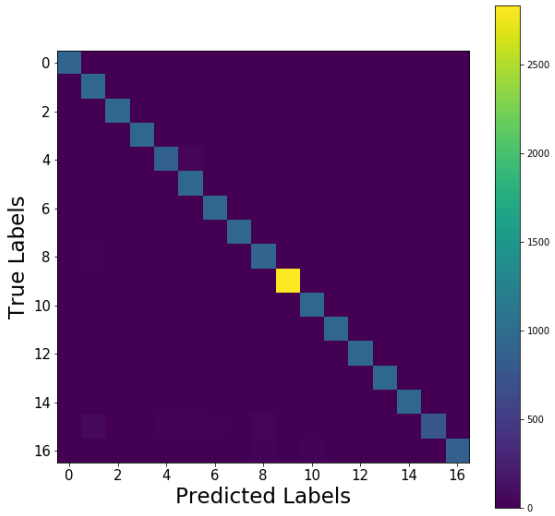
So, in order to improve our accuracy further, we combine such similar classes and then again fit an SVM model and measure the accuracy. We can see that our accuracy increased significantly from about 97% to 98.5%. We consecutively print the confusion matrix of this new data with 17 classes instead of the original 19 classes. Next up, in order observe the behavior of our data on the number of data points being used for training, we do an analysis of how the accuracy will vary with different number of training data points ranging from just a 1000 to 10,000 and the rest being used to test and measure the accuracy. We see a sharp increase in the accuracy when we move from 1000 training data points to 2000 with about 2% increase in the accuracy.

Finally, we try to find the best parameters for our data, and we use the GridSearch method in order to accomplish that. We form a pipeline for form a scalar and an SVC object. We test the data on 5 values of C: $10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}$ and 4 values of γ: $10^{-2}, 10^{-1}, 10^{0}, 10^{1}$.

Because of an enormous amount of data and about 20 combination of C and γ possible, the GridSearchCV routine was taking a long time fitting the data. For the sake of simplicity, we have shuffled the data and have applied the procedure to the first 10,000 data points and computed the best score and the

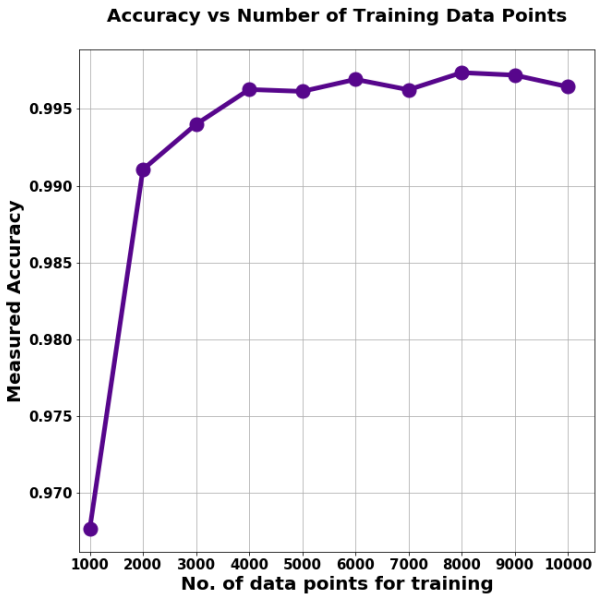


(a) Confusion matrix with SVC

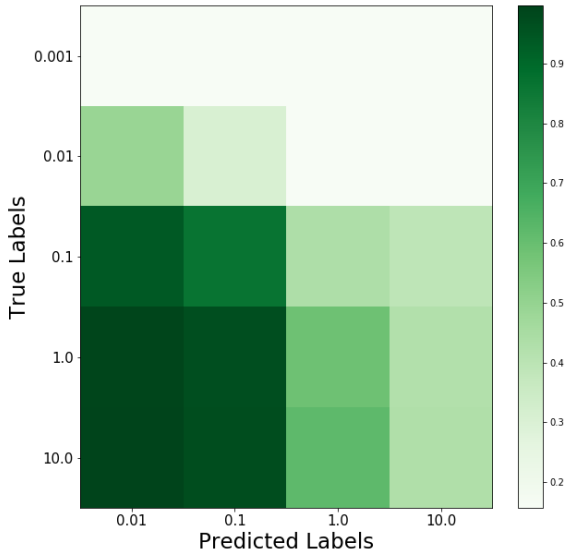


(b) Confusion matrix after combining classes 10, 11, 12

best parameters. We can see that the best C comes out to be 100 and γ to be 0.01 which helps the score to reach 99.7%.



(a) Accuracy versus number of training data points



(b) Grid Search

7 Conclusions

The average accuracy values for different techniques and parameter values are as follows (*CC* stands for combine classes):

Technique	Relevant Parameters and Details	Average Accuracy
Logistic Regression	C=0.1	97.1
Logistic Regression	C=100	99.3
Support Vector Machines (SVM)	$C = 1$, linear kernel	97.1
Support Vector Machines (SVM)	$C = 1$, linear kernel, *CC* 10, 11, 12	98.5
Support Vector Machines (SVM)	$C = 10, \gamma = 0.01$	99.7

For comparison, following are the average accuracy values achieved by some of the techniques followed in the reference papers[1]:

Technique	Relevant Parameters and Details	Average Accuracy
Least Squares Method (LSM)	$l-2$ norm minimization	89.4
Artifical Neural Network (ANN)	1 hidden layer with 12 neurons	86.9
Bayesian Decision Making (BDM)	MAP estimation	99.1
Support Vector Machines (SVM)	RBF kernel: $\gamma = 4$	98.6
κ-Nearest Neighbors	$\kappa = 7$	98.2

Please refer to the .ipynb Jupyter Notebook for the code and outputs.

References

1. K. Altun, B. Barshan, and O. Tunçel, "Comparative study on classifying human activities with miniature inertial and magnetic sensors," Pattern Recognition, 43(10):3605-3620, October 2010.

2. B. Barshan and M. C. Yüksek, "Recognizing daily and sports activities in two open source machine learning environments using body-worn sensor units," The Computer Journal, 57(11):1649–1667, November 2014.

3. K. Altun and B. Barshan, "Human activity recognition using inertial/magnetic sensor units," Proceedings First International Workshop on Human Behavior Understanding (in conjunction with the 20th Int. Conf. on Pattern Recognition), 22 August 2010, Istanbul, Turkey, A. A. Salah, T. Gevers, N. Sebe, A. Vinciarelli (editors), HBU 2010, LNCS 6219, pp.38-51, Springer: Berlin, Heidelberg, 2010.