ECE 542- Project 3a - Body-Rocking Behaviour Recognition

Srinivas Gopalakrishnan[1] Rajaram Sivaramakrishnan[2] and Priyankha Bhalasubbramanian [3]

I. METHODOLOGY

In this project we have to devise a Machine Learning model for Body-Rocking Behaviour. We have obtained the data of wearable sensors on arm and wrist for 1-2 hours with which they have determined whether there is Body Rocking.

A. FEATURE EXTRACTION

With the given data of accelerometers and gyroscopes for different sessions and detections, we can determine whether there is the body rocking motion by using basic Machine Learning Algorithms for the training. The first step (pre-processing) in our approach is to extract the features from the IMU signals generated from accelerometer and gyroscopes from the 2 devices located on the arm and wrist of the subjects. For every session, the data is such that there are 3 readings of accelerometer and gyroscopes (corresponding to the 3 axes), each from both the devices installed on arm and wrist. These readings are taken in the intervals of $2e^{-2}$ secs. So, essentially we group the 3 (x,y,z) readings of accelerometer/gyroscope for a window of rows in the Session and extract temporal and frequency based features. Some of the temporal features include mean, variances and covariances, skewness and Kurtosis. The frequency based features involve computing the maximum frequency response and we use FFT.
While extracting features we observe that we extract 21 features (scalar values) for every window corresponding to accelerometer/gyroscope. This way we extract 84 features for every window, when we combine the accelerometer and gyrosope IMU signals generated from the device on arm and wrist. We repeat this step by sliding the window (using a pre-defined stride length which will be discussed in the later sections of the report) and we also ensure that we cover all the rows in every session.

B. POPULATING RESPONSE VARIABLE

For the response variable, we need to do a transformation such that for every window, only one y value (detection) is extracted. There are 2 approaches to address this. One approach is to count the number of 1s (observed body rocking behavior) and the number of 0s (body rocking behavior not observed) and choose the label that has highest frequency, similar to a majority voting. The other approach is to, take the middle label of every window as the response to that particular window. For example, for the first window of 1 to 150, the 75th label will be considered. In this project we have implemented the second approach.

C. CLASSIFIER

After extracting the input features and transforming the response variable, the task now is to decide a classifier to construct a model. Few easily implementable algorithms include Decision Tree classifier, Random Forest classifier and Support Vector Machine(SVM). For this project we have chosen to use the SVM for the training model. The idea of SVMs is to constrain the capacity of a learned hyperplane function in order to maximize its generalization properties. Moreover, the work done by Ulf Groekathfer et al., in the paper titled Automated Detection of Stereotypical Motor Movements in Autism Spectrum Disorder Using Recurrence Quantification Analysis shows that SVM achieves a good classification accuracy when compared to classical Decision tree methods and Random Forests. SVMs incorporate a hyperparameter C by which value miss-classifications are penalized during training.

II. MODEL TRAINING AND HYPER-PARAMETERS TUNING

We chose to train the model with SVM (support Vector Machines). There are lot of hyper-parameters options available for our project. We have mainly restricted ourselves to the Stride length for feature extraction, the penalty factor for SVM, and the kernel. In this project we have fixed the time interval considered for 1 data to be 3 seconds(150 accelerometer and gyroscope measures).

A. VALIDATION SETS

For the training purposes, we have totally 6 sessions of accelerometer and gyroscope data for both arm and wrist movements. With these 6 sessions, our approach for tuning the hyper-parameters is to randomly select 2 sessions as our validation set and use the others for training. For this purpose, we shuffled all the sessions and picked 2 sessions for validation and used the remaining sessions for training. This ensures that we don't use 2 continuous sessions for validation which will give a biased result. After shuffling we use the sessions 5 and 13 for validation and remaining (in shuffled order) for training. This has been summarized as
So our data is split as

- Training Set = Session01, Session07, Session06, and Session12
- Validation Set = Session05, and Session13

[1] Srinivas Gopalakrishnan is a student of the Masters program in the ECE Department at North Carolina State University
[2] Rajaram Sivaramakrishnan is a student of the Masters program in the ECE Department at North Carolina State University
[3] Priyankha Bhalasubbramanian is a student of the Masters program in the CSE Department in North Carolina State University

B. STRIDE LENGTH AND PENALTY FACTOR

Stride Length is the measure of the data points to be skipped before considering the next time interval. If the stride length is 75, our first data point is for values 0 - 150 and next is from 75 - 225. We have considered 3 different stride lengths to tune the model **25, 50, and 75**.

Penalty Parameter 'C' is a crucial hyper-parameter for SVM, as it gives SVM optimization how much you want to avoid misclassifying each training example. For this project we have considered the values for penalty parameter as **1,10, and 100** for tuning.

For these values, the model was fit and the Validation accuracy is got as shown in Figure 1 and Figure 2.

```
In [7]: runfile('C:/Users/rthiruv/Desktop/NN project3/sample.py', wdir='C:/
Users/rthiruv/Desktop/NN project3')
75 1 0.8954195379002837
75 10 0.9027158492095663
75 100 0.9019051479529794
50 1 0.8872972972972973
50 10 0.8981081081081081
50 100 0.8991891891891892
25 1 0.8933639681037978
25 10 0.901878632247601
25 100 0.9093120691985404
```

Fig. 1. Accuracy Score for Different Stride Lengths and Penalty

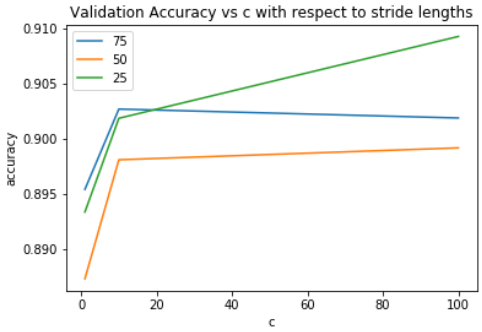

Fig. 2. Validation Accuracy vs Penalty

C. KERNEL

For the SVM optimizer we have a Kernel option, we can use linear, polynomial or a gaussian(rbf) for simplicity. When we used linear or polynomial kernel, we observed that the time complexity for the model training was very large and the rbf kernel gave us the most optimum result.

D. OBSERVATIONS

From the tuning of the hyperparameters, our model was decided as

- Time Interval = 3 seconds
- Stride Length = 25
- Penalty Parameter = 100
- Kenel = rbf kernel
- gamma = scale

We have observed that these values give the maximum validation accuracy for the Validation sets. The observed prediction and ground truth detection data for Session05 and Session13 are as shown in Figure 3 and Figure 4 respectively.

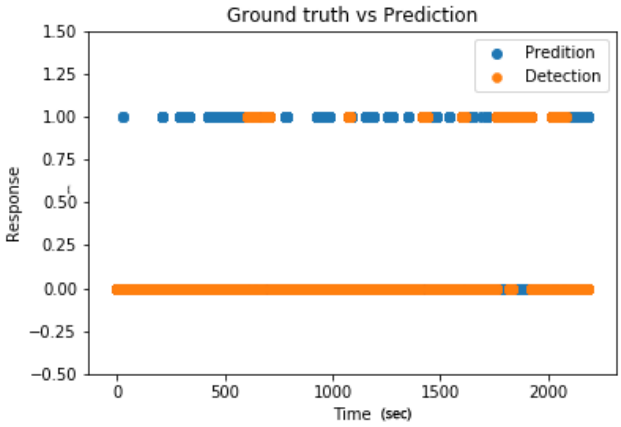

Fig. 3. Ground Truth vs Prediction for Session05

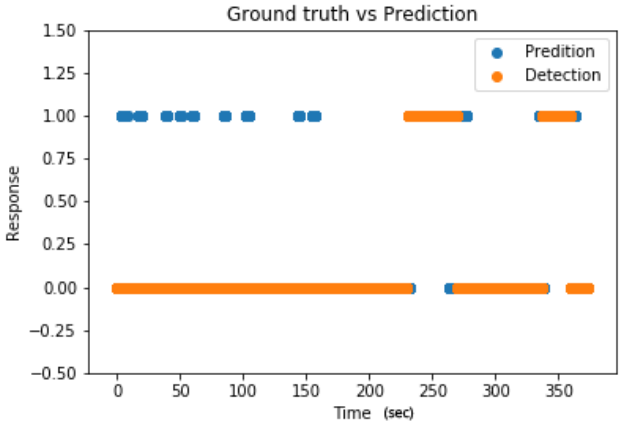

Fig. 4. Ground Truth vs Prediction for Session13

III. EVALUATION

A. EVALUATION METRICS

During the validation phase for hyper-parameter tuning we considered the data from Sessions 5 and 13 as validation set. So, without loss of generality, we consider the same two sets to estimate the performance of our model using error metrics like Accuracy, Precision, Recall and F1 measure.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Recall = \frac{TP}{TP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall}$$

where,

TP = True Positives, i.e., number of positive(1) observations

predicted as positive (1)
TN = True Negatives, i.e., number of negative(0) observations predicted as negative (0)
FP = False Positive, i.e., number of negative(0) observations predicted as positive (1)
FN = False Negatives, i.e., number of positive(1) observations predicted as negative (0)
Using these measures we observe the following results:

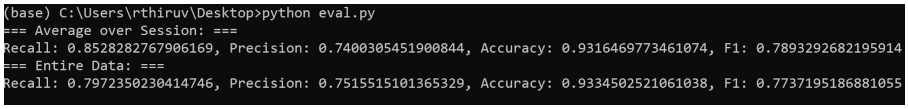

Fig. 5. Evaluation Result

The performance metrics under Average over Session correspond to the results obtained from Session 5 and 13 in Figure 5.

B. PREDICTIONS FOR TEST DATA

Finally, we train our model using all the sessions 1, 5, 6, 7, 12 and 13. Using this trained model we predict the detections for the test set consisting of Sessions 2, 3, 15 and 16. For the purpose of generating our predictions for every row of the aforementioned test sessions, we generate the features similar to how we did while training, with window of 3 seconds (150 rows), but the only difference is that the stride length was 1. So, our aim is to generate a row of features for every row of input data (except the first 150). For the first 3 seconds (150 rows) we have assumed that the predictions are 0 (no observed body rocking behavior), because the general pattern after inspecting all the detections of training sessions is that the subject does not show the behavior for almost the first 20-30 seconds. Hence, it is fair to assume that the predictions for the first 3 seconds of test set is 0. For the remaining rows we feed our model with the extracted features, generate the prediction and append it to out prediction file for that session. This is repeated for all sessions and we successfully generate our prediction results.