Classification of Body-Rocking Behavior

Ziad Ali
Electrical Engineering
North Carolina State University
Raleigh, US
zaali@ncsu.edu

Anusha Manur
Computer Science
North Carolina State University
Raleigh, US
amanur@ncsu.edu

Shashank Shekhar
Computer Science
North Carolina State University
Raleigh, US
sshekha4@ncsu.edu

I. Methodology

Our goal was to determine an optimal non-RNN-based machine learning framework that could be used to detect body-rocking behavior from blind subjects. The data was obtained with inertial sensors on the subjects' wrists and arms using both accelerometers and gyroscopes. In the future, we will compare the accuracy of the optimal framework from this study with that of an RNN to determine how much more accurate time-based machine learning models are at classifying time series data.

The data from each sensor was provided in the following format: for each sample, six data points were collected. The first three points correspond to the accelerometer measurements in the x, y, and z directions, while the second three points correspond to the gyroscope measurements in the x, y, and z directions. The sampling rate of both devices was 50 Hz.

To train our models, we used a sliding window approach and extracted features over a time period of the recorded data, rather than using each individual data point. We first computed the mean of each column of the dataset, then computed the covariance matrix (3x3) and recorded all of the unique values (ignored symmetrical data). We then computed the skewness of each column as well as the kurtosis. Lastly, we recorded the maximum frequency response of each column. We performed these computations for the accelerometer and gyroscope data for both the arm and wrist sensor, and then concatenated all of the features to obtain our input vector for the model. To collect multiple input vectors for training, we computed these features over a window of a specified length, then shifted the window by a specified slide value, and recalculated the features iteratively.

The three models we chose to investigate were the random forest model, the k-nearest neighbors model, and the multi-layer perceptron model. A random forest is an ensemble learning method that averages the classification results of a set of decision trees. A decision tree is a classification model that breaks down a set of input features into a tree-like structure such that each feature corresponds to a layer of the tree and different combinations of input features result in different paths along the tree and thus correspond to different classifications. While individual decision trees are not adequate estimators, random forests aggregate multiple decision trees to obtain more robust results [1] The RandomForestClassifier from the Ensemble module of the Scikit-learn machine learning library was used to build the model.

The k-nearest neighbors (kNN) algorithm classifies data points based on a similarity measure. Nearest-neighbor classifiers are based on learning by analogy, that is, by comparing test data with similar training data. In kNN classification, an object is assigned to the class most common among its k "nearest" neighbors. To find the k closest neighbors, a distance metric like Euclidean or Manhattan is used [2]. The kNeighborsClassifier from the Neighbors module of the Scikit-learn machine learning library was used to build the model.

A multilayer perceptron (MLP) is a type of artificial neural network composed of non-linear activation functions. MLPs are composed of input layers, an arbitrary number of hidden layers, and an output layer that makes a decision or prediction regarding the input. Training involves adjusting the parameters, or the weights and biases of the model, through backpropagation in order to minimize error [3]. The MLPClassifier from the Neural Network module of the Scikit-learn machine learning library was used to build the model.

We trained each possible classifier over a range of hyper-parameters and validated our classifier-hyper-parameter combinations using a validation set. The optimal classifier was then applied to the test data to generate final predictions.

TABLE I
Model Hyper-Parameter Values

Classifier	Hyper-Parameter	Values
Random Forest	Trees	100,150,200,250,300,350,400
KNN	Neighbors	50,100,150,200,300,400,500
MLP	Layers	2,3,5,10,20
All	Window Length	100,150,200,250,300,350,400
All	Slide Length	10,25,50

II. Model Training and Hyper-parameter Selection

For each model, we evaluated the performance over a range of hyper-parameters corresponding to window length values of 100-400 samples (step: 50) and slide lengths of 10, 25, and 50 samples. This corresponds to window lengths of 2-8 seconds, and slide lengths of 0.2-1 seconds. We used 5 sessions of data to train and one session to validate.

For the random forest models, the hyper-parameter we tuned was the total number of trees used in the model (100-400, step: 50). For the kNN algorithm, the hyper-parameter we tuned was the number of neighbors used in classification (50-200, step: 50, then 200-500, step: 100). For the MLP classifier, the hyper-parameter we tuned was the number of hidden layers (2, 3, 5, 10, and 20). Table I summarizes our hyper-parameter options for the different models.

III. Evaluation

To choose the final model for each classifier, we varied the aforementioned hyper-parameters and obtained the following results.

From Fig. 1 - 3, we see that the random forest model with number of estimators (trees) = 150, window length = 300 and slide = 50 gave us the best results. For this model, we achieved a precision of 82.06%, a recall of 94.06%, an F1 score of 87.65%, and an accuracy of 95.60% on our validation set.

From Fig. 4 - 6, we see that the KNN model with number of neighbors = 50, window length = 250, and slide = 25 gave us the best results. For this model, we achieved a precision of 47.56%, a recall of 77.11%, an F1 score of 58.83%, and an accuracy of 82.08% on our validation set.

From Fig. 7 - 9, we see that the MLP model with hidden layers = 2, window length = 300, and slide = 50 gave us the best results. For this model, we achieved a precision of 75.53%, a recall of 73.07%, an F1 score of 74.28%, and an accuracy of 91.60% on our validation set.

Based on these results, we chose to use a random forest model with 150 estimators, a window length of 300 (6 s), and a slide length of 50 (1 s). We trained this model over our entire training set to generate predictions for the test sessions, which will be evaluated in a future study. An example of a segment of the random forest predictions vs. the ground truth validation set is shown in Fig. 10.

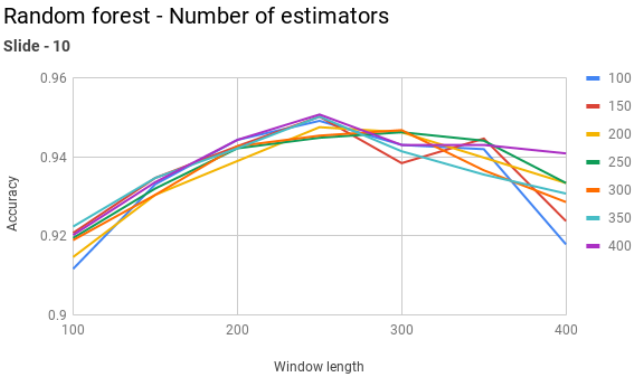

Fig. 1. Accuracy of random forest model on validation set trained over a range of window length values (100-400) and for a variable number of estimators (100-400) with a window slide length of 10.

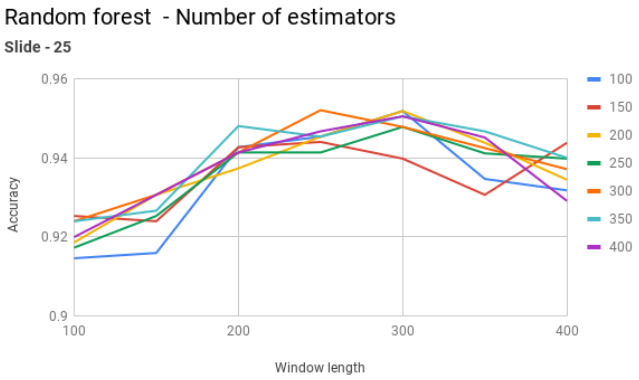

Fig. 2. Accuracy of random forest model on validation set trained over a range of window length values (100-400) and for a variable number of estimators (100-400) with a window slide length of 25.

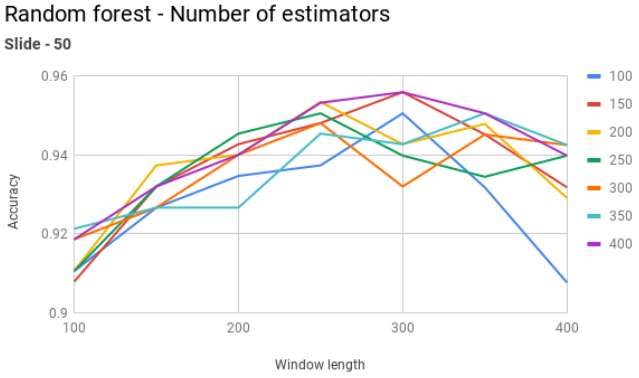

Fig. 3. Accuracy of random forest model on validation set trained over a range of window length values (100-400) and for a variable number of estimators (100-400) with a window slide length of 50.

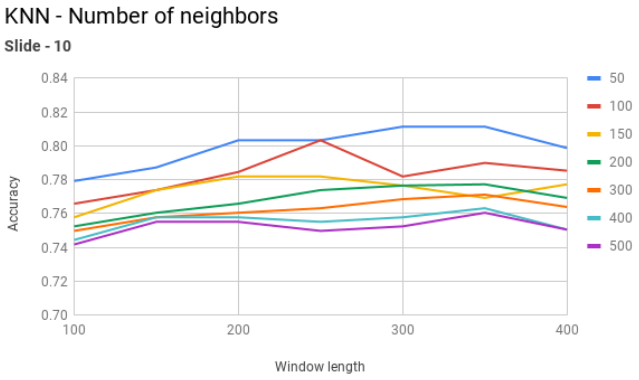

Fig. 4. Accuracy of KNN model on validation set trained over a range of window length values (100-400) and for a variable number of neighbors (50-500) with a window slide length of 10.

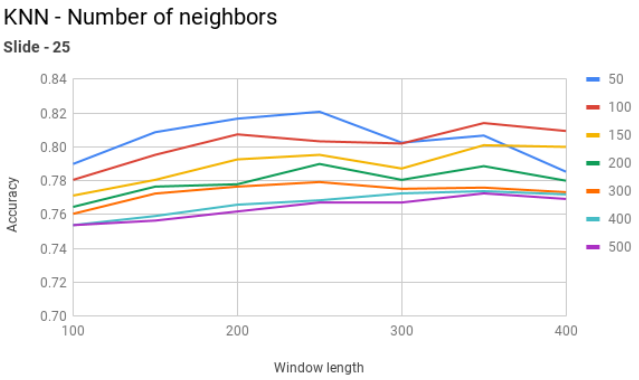

Fig. 5. Accuracy of KNN model on validation set trained over a range of window length values (100-400) and for a variable number of neighbors (50-500) with a window slide length of 25.

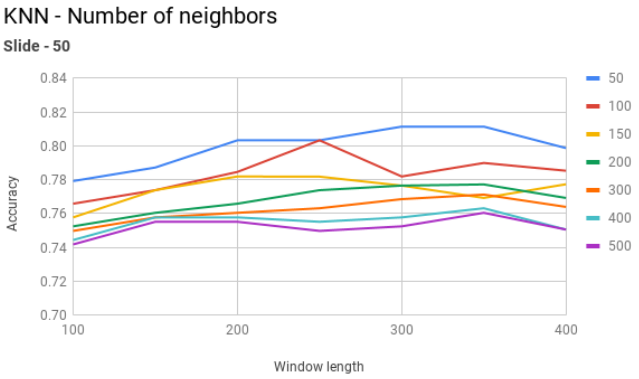

Fig. 6. Accuracy of KNN model on validation set trained over a range of window length values (100-400) and for a variable number of neighbors (50-500) with a window slide length of 50.

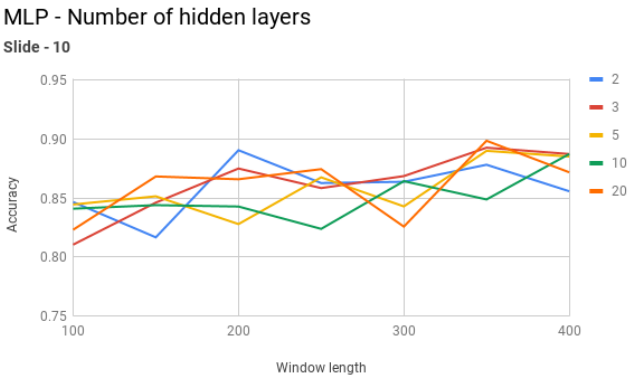

Fig. 7. Accuracy of MLP model on validation set trained over a range of window length values (100-400) and for a variable number of hidden layers (2,3,5,10,20) with a window slide length of 10.

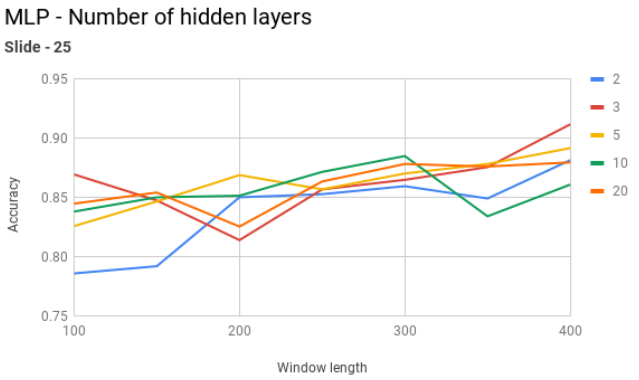

Fig. 8. Accuracy of MLP model on validation set trained over a range of window length values (100-400) and for a variable number of hidden layers (2,3,5,10,20) with a window slide length of 25.

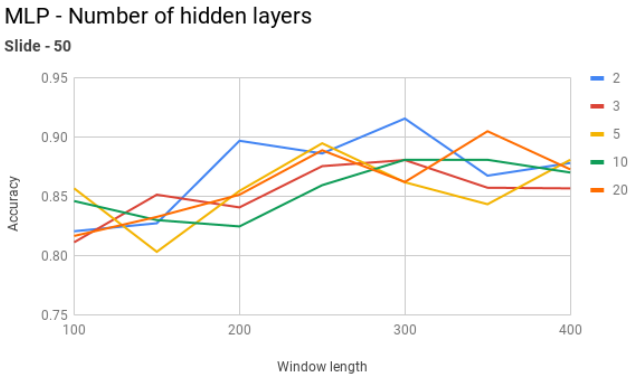

Fig. 9. Accuracy of MLP model on validation set trained over a range of window length values (100-400) and for a variable number of hidden layers (2,3,5,10,20) with a window slide length of 50.

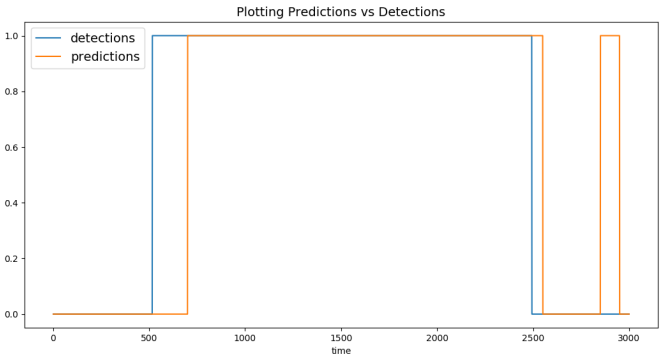

Fig. 10. Predictions and the given detection values over a range of the validation set. The predicted values are orange and the true values are blue.

REFERENCES

[1] A. Liaw and M. Wiener, Classification and Regression by randomForest, R News, Dec. 2002.

[2] T. M. Cover and P. E. Hart, Nearest Neighbor Pattern Classification, IEEE Transactions on Information Theory, Jan. 1967.

[3] M.W. Gardner and S.R. Dorling, Artificial neural networks (the multilayer perceptron) - a review of applications in the atmospheric sciences, Atmospheric Environment, Aug. 1998.