Mobile Computing Lab 2

Isaaq Khader, Devon Delgado, Beza Desta

Task 1: Validation Set

The train/test split was generated using some simple array masking with numpy. Using np.random.permutation(max_val), one can generate a randomly permuted array with elements from [0, max_val). We decided on a 15% size of the validation set and a 85% size of the training set. For 140 samples per activity, 21 is roughly 15%, so the first 21 values from that permutation are used as a mask for getting the validation set and the rest of the values from the permutation are used as training. This was done iteratively per activity. Simple os and glob functions were used to move the specified training and validation data files in their respective locations.

Task 2: Statistical Threshold Classifier

The statistical threshold classifier was constructed using the summarize sensor trace function we created for our previous lab. By calculating the means, variance and standard deviation of the sample files using the training data we created, we were able to, in combination with attributes we identified to be important indicators of each activity in the previous lab, recognize the distinguishing feature of each activity. Similar to the authors of the CenceMe research paper, we also had difficulty differentiating between standing and sitting. Hence we had additional conditions to make sure that we were not confusing the two activities, standing and sitting, as well as other activities with the two. We accounted for reasonable deviations by ensuring that the calculated mean and variance of the sensed activity is within a certain range of the distinguishing mean or variance of each activity.

Average Precision	83%
Average Latency	2.48 seconds
Accuracy score	78.5%

Average (per activity)	
Jog	95%
Stand	67%
Sit	81%
Stretch	81%
Overhead	71%
Twist	76%

Mobile Computing Lab 2
Isaaq Khader, Devon Delgado, Beza Desta

Precision (per activity)	
Jog	57%
Stand	89%
Sit	80%
Stretch	71%
Overhead	100%
Twist	100%

Task 3: Shallow Learning Classifier

We chose a random forest model to determine the activities. It works by using the training data we created in Task 1 and having the model learn using specific features we deemed noteworthy and differentiating. We chose 13 features: the mean of the left and right controller y-angular velocity, y-position, headset y-position and variance of the headset y-angular velocity, y-velocity, left and right controller velocities in all directions. With these, we were able to achieve a significant accuracy and precision: both yielding 99.2%.

We chose a random forest model as they can perform classification tasks and produce good predictions with large datasets at an efficient level. By doing Google searches, we found that random forest models tend to be more accurate than other classification algorithms. Moreover, the model does not overfit data by using multiple trees but instead gives accurate and precise results, as seen by our results.

Average Precision	99.2%
Average Latency	2.9 seconds

Average (per activity)	
Jog	100%
Stand	100%
Sit	95.2%
Stretch	100%

Overhead	100%
Twist	100%

Precision (per activity)	
Jog	100%
Stand	100%
Sit	100%
Stretch	95.5%
Overhead	100%
Twist	100%

Task 4: Real-Time Detection

Unfortunately, we were unable to test our C# program on doing real time analysis for different activities, but we did fill out the file with our algorithm from Task 2. Essentially, we fill an array up with data from the last three seconds, get means and variances from that set of data, and perform real time analysis with the thresholds we decided on in Task 2. The hard-coded values are from our CSV files that calculate the means and variances for each feature from the training data.

From the experiences in Lab1 and Lab2, real-time VR sensing data could work towards a new 2FA that incorporates biometrics. Studies have been able to identify individual people with a very high accuracy based on just movement data alone. A paper that Prof. Lopes and and Prof. Zheng explorea biometrics from muscle stimulation. If this can become more fine tuned, then we believe that this is a promising and useful aspect of VR data for security. This could lock potentially sensitive data if the movement data seems off or have people sign into their profiles based on movements alone, i.e. challenges.

Contributions

Isaaq: I helped with Task 1 by fixing a bug we had where it would overlap files in both folders. In Task 2, I found a function to help us locate specific columns from the CSV files we used for statistical thresh holding. From there, we all worked together on using different features and values to make predictions. In Task 3, I worked with Devon to create the random forest tree using the training data we created in Task 1. I did the write up for Task 3 mentioning the significance of a random forest tree along with the statistical data required. I also coded the accuracy and precision scores for both Task 2 and Task 3. For Task 4, I drove the coding bits where we coded how we saw Task 4 playing out. I also added some segments to Task 1 and 4 where needed.

Devon: I made a train/validation split for the .csv files. I also worked on finding specific statistics for each attribute that are significant for thresholding and also training the Random Forest model. I also did some research for some nifty pandas/numpy functions that could help automate analysis better. I figured out how to train the Random Forest model after reading the sklearn docs and how to predict a label for an incoming sample. Collectively, we all coded it up together. I wrote up Task 1 for the report and half of Task 4 for the report.

Beza: I helped code up along with everyone on task 2 and task 3. Specifically helping with threshold features for task 2 and helping setup features for the shallow learning models in task 3. Additionally, I helped code up algorithms to calculate the precision and accuracy scores of our classifiers using sklearn. I also helped write up code for task 4 and how we envisioned the algorithm to be working in a VR setting. I also wrote up the lab report for task 2.