# Machine Learning for Android Motion: Work Plan

The person responsible for each task will be listed in () next to the task.

**Plan for Development of Machine Learning Algorithms:**

1. Obtain existing labeled accelerometer data for training and become familiar with the format of the data. (Doris)
2. Dimensionality reduction and feature reduction during pre-processing. (Doris)
   Consult Sherrill et al 2005 for the type of features that are useful for motion classification. Evaluate the importance of every possible feature and select 5-10 features as the final list of features on which the learning is performed. Project the features onto 2D space and make scatterplots showing the distribution of accelerometer data associated with different activities to analyze the informativeness of features. Divide time series into segments of equal length (possibly 1-3 seconds in duration) and pre-process the data with feature extraction for training.
3. Implement multiclass SVM in Matlab for performance analysis. (Doris)
   Implement both one-vs-one and one-vs-all SVMs and compare the performance of these 2 multiclass SVMs on accelerometer data classification based on the ROC curves.
4. Implement backpropagation in Matlab for performance analysis. (Wenqi)
   Implement backpropagation with different neural network architectures and compare their performance on accelerometer data classification based on the ROC curves.
5. Explore the applicability of genetic algorithms to motion classification. (Wenqi and Alex)
   Implement genetic algorithms in Matlab and do performance analysis on the algorithm if we find that genetic algorithms are suitable for the learning task.
6. Develop and implement clustering algorithm for classification of time series in Matlab. (Dai Wei and Wenqi)
   Consult Sherrill et al 2005 for the type of clustering algorithms suitable for motion classification and devise original clustering algorithms for implementation. Use ROC curves for performance analysis.
7. Explore and recommend other online learning algorithms applicable to classification of accelerometer data for implementation and performance analysis. (Dai Wei)
8. Evaluate performance of all learning algorithms and choose the best performing algorithm. (Doris)
   Take into consideration factors like the ROC curve for each algorithm but also the complexity and the amount of computation resource required. We plan on doing learning in on the Cloud and sending the trained classifiers back to the user after training to avoid exhausting the limited computation resource on the Android phone; therefore, the specification on the complexity of the learning algorithm is not too stringent.
9. Implement the best performing online learning algorithm in Java for incorporation into the Android App. (Alex and Robert)
10. Test the classification algorithm on Android phones and collect user feedback. (Dai Wei and Alex)
    Deploy a simplified version of the Android application containing just the motion classification component and collect feedback on runtime, resource utilization, and accuracy of classification.

**Plan for Development of DynaMap, the Android application:**

1. Where am I? (Alex)
    - To get the map up in the first place. We have a Google map API key.
    - We need to be able to display our own location on the map.
    - We need to be able to draw arbitrary lines on the map.
    - Record GPS data online.
    - Send GPS data to the app engine.
    - Download GPS data from the app engine.
    - Load false data into the app engine, and view it on the map.
    - Record GPS data offline
    - Prepare offline GPS data and send when appropriate
2. Who am I friends with? (Dai Wei and Robert)
    - To add friends, you have to know their email address. We ask them when they are next online. Email?
    - Friend somehow is able to accept. We'll link up it up for them on the app engine side.
    - Obtain friend list from the app engine.
    - Be able to display friends on the map. And can supposedly ask for their position/data.
    - Put fake friends in the app engine database. Download them and display them.
    - Unfriend. Requires app engine side checks for database consistency and the fact that no non-friend is asking for information.
3. What is everyone doing? (Doris and Wenqi)
    - Collect accelerometer data.
    - Hopefully we can assume that there is some sort of rudimentary classification system. Obtain classifcations and display them as plaintext on map.
    - Send classifications to database as required.
    - Download classifications and show a bunch of activity dots on the map. (because it's paired with location info) instead of plaintext, use icons or color!
    - And now we upload a bunch of fake data again and see what it looks like.
    - Make sure you can access friends' data too.