Classification

In the previous exercise you implemented the activity recognition chain up to feature extraction, now you will have to assign the computed feature vectors to one of several activity classes. This is done by a K-Nearest-Neighbor (KNN) Classifier, provided as a ready-to-use Java package by the Opportunity Project (see https://www.pervasive.jku.at/Research/Projects/?key=918 if you want to know more about Opportunity). Adding the KNN-Classifier to your application is the easy part of this exercise; the hard part will be defining the activities you want to recognize and finding the best features and parameters to do just that.

1) KNN-Classifier

Create a new Project called Classification and copy all your files from the previous exercise. Download 03 KNN Classifier.zip from the department homepage and extract it into your source folder. It contains the required package at.jku.pervasive.android.classifier.

Classes:
- Classifier
  Abstract classifier superclass.
  
  ```java
  public abstract class Classifier {
    public Classifier(Context context)
    public abstract int classify(float[] featureVector);
    public abstract void trainIncremental(float[] featureVector, int label);
    public void loadState(String filename) throws IOException
    public void saveState(String filename) throws IOException
  }
  ```

  - ClassifierState
    Represents the current state of a classifier, including any trained data.

  - KNNClassifier extends Classifier
    K-Nearest-Neighbor (KNN) Classifier implementation.

  - KNNClassifierState extends ClassifierState
    Represents the current configuration and all training data of a KNN classifier.

  - FeatureSpaceMonitor
    This class provides methods to analyze sets of feature vectors, e.g. find similarity of class clusters.

Exceptions:
- ClassifierSizeMismatchException
  Thrown whenever two vectors or lists that should have identical size have different sizes.
2) Activities (10 points)

It is time to think about the activities that you want to recognize with your application. Append a report (doc, pdf, txt, …) to this exercise in which you describe the following:

- Think of an everyday situation, e.g. driving a car, working in an office, studying at home, cooking....
- Describe at least 3 activities that are common in this situation. Think about what kind of movements are involved and how you could sense them with your mobile phone. Take into account that your activities should lead to different movement or orientation of the mobile phone, or it will be difficult to distinguish them. You may place the mobile phone on any body part as long as it is somewhat realistic: e.g. there are phone cases to wear your phone on the hip or on the arm.
- You have to use the accelerometer sensor, but are not limited to just this one sensor: You may include any other sensors in the recognition process, e.g. the GPS location or the wireless networks in range.
- Design your feature vector: Which features might be useful to recognize your activities, e.g. accelerometer mean or variance, magnetic field, etc. All these features have to be assembled into a single feature vector (a float array) that can be passed to the classifier.
- Try to estimate good values for the parameters in the activity recognition process: minimum sensor sample rate, window size, window jump size, classifier parameters K and threshold (check slides if you are not sure what these are).

3) Recognition (15 points)

Include the KNN Classifier in your application and implement any additional features you need. You may customize the files from previous exercises and change whatever you want, there are no more limitations.

Your application has to provide two different modes, a training mode and a live mode.

- In training mode your application is training the classifier with collected samples. The training process depends on your application, an example what this could look like is in appendix A.
- In live mode the application continuously collects sensor data, performs classification and displays the result. Simple text output is sufficient.

To keep the application running while your phone is locked (e.g. because you want to wear it in your pocket), you have to acquire a wake lock. See android.os.PowerManager. You also have to add the permission android.permission.WAKE_LOCK to your AndroidManifest.xml.

Experiment with the variable parameters (sample rate, window size, classifier K, …) to get optimal results. Append the results of your experiments to the report.

Guidelines:

- Zip your whole project folder and upload it.
- Your source code has to include comments, to explain things that are not obvious anyway (please do not add comments like "assigning a value to xyz").
- Always add a summary of what you did in the class description at the beginning of the file.
Appendix A: Sample Screenshots

These screenshots show both live and training mode. Your application might have very different requirements and can look totally different, as long as there is a training and a live mode.