Lab 12 - Machine Learning in Android

Mobile phone sensors provide raw data, e.g. GPS coordinates. Machine learning is used to link the sensed data with higher level concepts, e.g. whether a person is at home or work. In this lab we will build upon the periodic accelerometer sensing lab, and recognise certain phone movements using a simple Naive Bayesian classifier.

Periodic accelerometer sensing

To shortcut the development process, you can download code for periodic accelerometer sensing from https://github.com/vpejovic/AccSensingDemo
The code relies on `SensingAlarmReceiver`, which is based on the `AlarmManager` class. This class calls `AccSenseService`, an `IntentService` for sensing the accelerometer, every five seconds. The service then calculates the mean, variance and the mean crossing rate (MCR) of the accelerometer data, and sends them, via an Intent that is broadcast locally. Currently, this data is being received in `MainActivity`, via `AccBroadcastReceiver`.

The above code is just a starting point, you should check if you understand everything in it (please raise your hand if you don't!), and ensure that it runs on your phone (it might require some tweaking).

Recognising different movement patterns

Different phone movements, for example, a circular motion, jerking, flipping, result in different accelerometer values' recordings. Our hypothesis is that different movements will result in different values of accelerometer mean, variance and MCR. We will use a third-party machine learning library to manage the movement pattern classifier.

Add the Android machine learning library to your project, by putting the following line in the app's gradle file:

```
compile 'si.uni_lj.fri.lrss.machinelearningtoolkit:mltoolkit:1.2'
```

The machine learning library lets you create classifiers, train them with labelled data, and then query them to infer high level information from the sensed raw data. The complete code for the library can be found here: https://github.com/vpejovic/MachineLearningToolkit/

Instantiating a classifier

A classifier is instantiated through the `MachineLearningManager`,. The manager requires a context:

```
MachineLearningManager mManager =
MachineLearningManager.getMLManager(ApplicationContext.getContext());
```

A classifier is defined by its `Signature`, i.e. the features it connects, and the indicator of the class feature (the one we are trying to predict). Features can be numeric (i.e. expressed through real numbers, can be ordered), or nominal (i.e. representing different categories, cannot be ordered). An example of a numeric feature is our accelerometer mean value. An example of a nominal feature is our class, a particular movement (e.g. "horizontal", "vertical", "still", "moving", etc.).

The machine learning library supports a few different classifier types, and exposes certain configuration options. We will keep it simple and instantiate a default Naive Bayesian classifier.

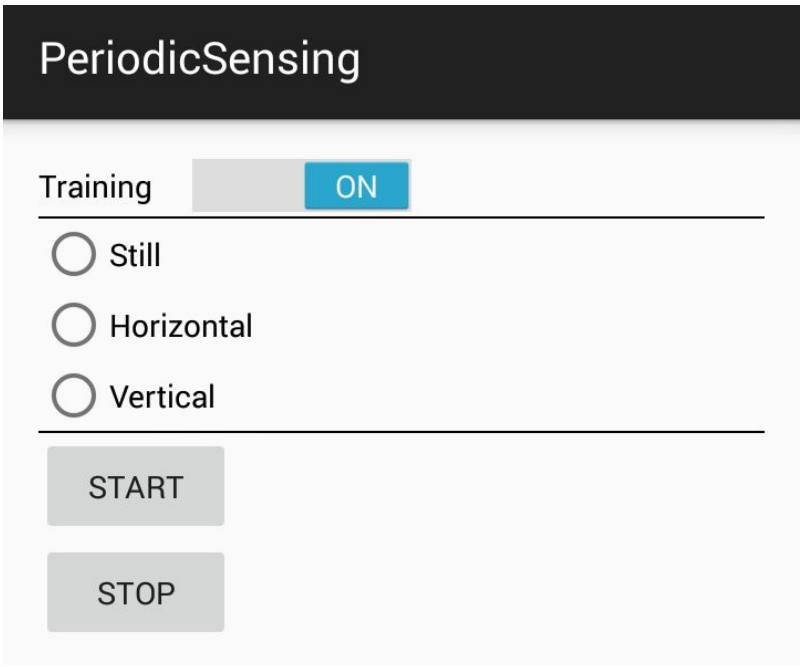
The following code instantiates a Naive Bayesian with one nominal feature and one numeric class feature with two values. Using the code below as template, instantiate your own classifier that takes the three accelerometer features, and predicts one nominal feature with the movements you want to predict:

```
Feature accMean = new FeatureNumeric("accMean");
ArrayList<String> classValues = new ArrayList<String>();
classValues.add("horizontal");
interValues.add("vertical");
Feature movement = new FeatureNominal("movement", classValues);
ArrayList<Feature> features = new ArrayList<Feature>();
features.add(accMean);
features.add(movement);

Signature signature = new Signature(features, features.size()-1);

mManager.addClassifier(Constants.TYPE_NAIVE_BAYES, signature,
                       new ClassifierConfig(),"movementClassifier");
```

Where to instantiate your classifier? You can do that in the MainActivity's `onCreate` method. The instantiation itself is not processing-heavy. On the other hand, classifier training and querying might be, and should be done on a separate thread.

Training a classifier

Now, your classifier doesn't know much, so let's train it. First, we need to provide a label for the sensed movement. Decide which movements you want to recognise and modify the activity_main.xml to let the user signal whether she's training the classifier, and if so, which particular movement is she performing at the moment. A sample interface is given in the figure.



Next, in MainActivity, in `AccBroadcastReceiver's recordAccData` method check if the user is training the classifier (the corresponding switch is turned on), and if so, which movement is being performed (check which radio button is selected).

Training a classifier can be processing heavy, so it should be performed in a separate thread. Create a new class `TrainClassifierService`, make it extend an IntentService, and in its `onHandleIntent` train the classifier. Here is an example of training a classifier with a class value `label` (String), and a mean accelerometer value `mean` (double):

```
ArrayList<Value> instanceValues = new ArrayList<Value>();
Value meanValue = new Value(mean, Value.NUMERIC_VALUE);
Value classValue = new Value(label, Value.NOMINAL_VALUE);
instanceValues.add(classValue);
Instance instance = new Instance(instanceValues);

ArrayList<Instance> instances = new ArrayList<Instance>();
instances.add(instance);
MachineLearningManager mManager =

MachineLearningManager.getMLManager(ApplicationContext.getContext());
Classifier c = mManager.getClassifier("movementClassifier");
c.train(instances);
```

To train a classifier in a separate service you need to pass the information on the mean, variance, MCR and the label to the service in an Intent from the `MainService`. This is not much different from the way you call `AccSenseService` when you do periodic sensing.

Finally, you can check the state of your classifier by calling:
`c.printClassifierInfo()`

Movement inference

Ensure that you train your classifier with different phone movements. Then, switch the training off on the main screen, and let's try to infer the movement - is it horizontal, vertical, still (or any category you want to recognise). Since we might be moving the phone rapidly, just showing a `Toast` with the inference label might not be readable. Instead, change the background colour of the parent `LinearLayout` of the `MainActivity` (the ID of the `View` is `container`) depending on the movement (e.g. let it be red if you move horizontally, blue if you move vertically, etc.).

The classification is done via `classifier.classify()` method, that requires an Instance consisting of features -- mean, variance, MCR -- which we are trying to find a label for:

```
Classifier c = mManager.getClassifier("movementClassifier");
Value inference = c.classify(instance);
```

`Value.getValue().toString()` gives you the string value of the inference. Depending on the inference, change the background of the screen.

Where to do the inference? Ideally, you should do it in a separate `IntentService`. However, you might want to try doing it in the `MainActivity` first (it will be easier to change the background colour from there). If your application crashes, move to a different thread.

Happy coding!