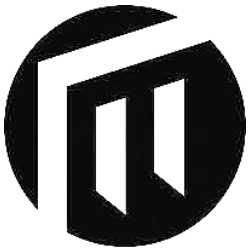

DA274A: Internet of Things and People

Hand Gesture Recognition

Lab 3

Shahram Jalaliniya

November 2017

Lab 3: Hand gesture recognition

Activity and gesture recognition is one of the most popular topics in Pervasive Computing. In short, activity recognition denotes the work involved in trying to infer human activity from sensory data. The goal of this lab is to learn how to use weka for processing the data you have collected in Lab 2. In Lab 2 you have collected sensor data in two files: 1) training data and 2) test data. The training data includes at least 20 instances of each gesture while the test data includes about 5 instances. Each instance of the data (i.e. each row in the .csv file) is a window of 121 values (6 values of Acc and Gyr **X** 20 samples **+** 1 label). In this lab, you will build your classifier model in Weka. You will compare the performance and accuracy of different classifiers (K Nearest Neighbor, Decision tree, Multilayer Perceptron, etc.) through cross-fold validation and also holdout method (using test data). Also in this lab you apply come of the preprocessing techniques that you learned in lecture 3 for improving the performance of the classifier.

Step1: Classification using Weka

1- Before the classification step make sure that your training and test data collected in Lab 2 is complete and correct. All rows of the .csv file should have 121 values and the last value should be the label. If there are a lot of missing data in the rows, repeat the data collection by performing each gesture 25 times. Use 20 instances for training data and 5 instances for test data. Remember to use the new version of the processing code to collect the data. There was a bug in the previous version that led to counting wrong number of gestures while recording. You can find the new version from below link: https://www.dropbox.com/s/zziak2z6bf1cqak/imu_Lab_New_2.pde?dl=0

2- Download and install Weka software (https://sourceforge.net/projects/weka/)

3- Run Weka and open the Explorer tab. Open the train.csv file in the preprocessing panel.

4- Try to classify the data instances (for gesture_label class) using the 10-fold cross validation option with different classifiers such as J48, BayesNet, Neural Network (MultilayerPerceptron), KStar, Logistic, etc. (If you have problems at this step, refer to the Weka tutorial.) Interpret the results. The result will be similar to what you see in Figure 1.

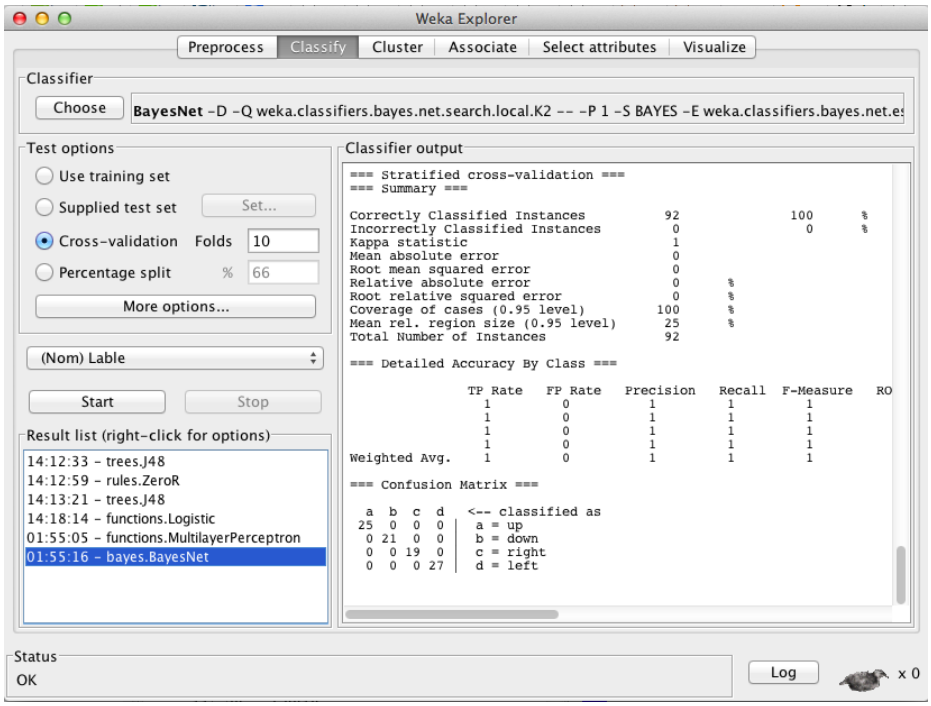

Figure 1: An example output of the classification in Weka

5- **Mandatory task 1:** Please write your answers to these questions on a paper and hand in the paper to the teachers in the lab.

   a. Which classifiers give more accurate results?
   b. Which ones are faster?
   c. Visualize the generated decision tree after using J48. Which axis is chosen as the first branching node?
   d. How does the confusion matrix look?
   e. Which gestures are classified more correctly than others?
   f. Which gestures are more confused than others?

6- Now, try to classify the test data in the same way by choosing the *'supplied test set'* and opening the test.csv file. Make sure the gesture_label is chosen as the class. If you are getting error message, you might need to convert the .csv files to .arff in Weka. Because sometimes the order of the attributes changes in the .arff file generated by Weka. The order of the attributes of the training and test datasets needs to be the same. Again, interpret the output.

7- **Mandatory task 2:** Please write your answers to these questions on a paper and hand in the paper to the teachers in the lab.

   a. Which classifier gives the best results on the test data?

b. Are the results far from the cross-validation output? Why?

Step2: Preprocessing to improve the classification

1- If the performance of the gesture recognition is not high you can improve it by using some preprocessing techniques such as smoothing and normalization. Open the train.csv and test.csv files in MS Excel. Now smooth the data using a moving average with the size of 5 as you did it in Lab 2.

$$\frac{x_t + x_{t-1} + x_{t-(n-1)}}{n}$$

If you don't know how to enter the formula in MS Excel, look at this video: https://www.youtube.com/watch?v=cVqyGpK2RWI

2- Use the min-max technique that you learned in Lecture 3 to normalize the data. Remember to use a similar min and max for both train and test datasets. You can ask teachers for help if you do not know how to define formula in MS Excel (v' is the normalized value and v is the old value).

$$v' = \frac{v - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A$$

If you don't know how to enter the formula in MS Excel, look at this video: https://www.youtube.com/watch?v=oHrPh727JIU

3- **Mandatory task 3:** Now repeat the classification steps in Weka with the normalized and smoothed data. Use the best classifier you found in the Step 1 and compare the results of cross-fold validation and performance of the classifier on the test data with the results from Step 1. Do you see any significant difference? Why?

Step3: Use Weka to detect gestures in real-time

You can add the Weka.jar file as an external library to your java code and use it for building classifiers and classification samples in real-time. You can learn from (https://weka.wikispaces.com/Use+WEKA+in+your+Java+code) how to use Weka in your java code. You can find a simple example of using Weka in a java code to label the test data

here (https://www.dropbox.com/s/hgx1y9ciqzo6525/wekaClassifier.java?dl=0). Remember to add the weka.jar file (https://www.dropbox.com/s/lqv35as8ul3egbh/weka.jar?dl=0) as an external library to your IDE. For live gesture recognition in your project, you should first provide the training dataset and build the classifier. Then you should build a data instance in real-time including 121 columns (20 instances x 6 values = 120 sensor data + 1 label) when you get the data from your wearable sensor. The label should be an arbitrary dummy value. The classifier chooses the closest label based on the reported sensor data. You can add this weka.jar file (https://www.dropbox.com/s/wz484vzdof7t9g2/weka.jar?dl=0) to your Android code because apparently not all versions of weka library work in Android.

In the processing code that you have used in Lab 2 for collecting training and test datasets you can change “logFile” to false, “trainFile” to false, and “gesture_rec” to true. You also need to define the path of your train.arff file on your computer instead of the current “trainPath” in the code. Remember to convert the train.csv file (before preprocessing) to train.arff file using Weka. Now run the program and try live gesture recognition.

Tip: If you have problem with the processing code, try to install processing version 2.2.1 and change “size(820, 600, P2D);” in the code to “ size(g_winW, g_winH, P2D);”

Mandatory task 4: In live gesture recognition, repeat each gesture 5 times and write down the detected gestures. Create a confusion matrix for 6 gestures and 5 repetitions. Calculate the accuracy and recall of the live gesture recognition. How is the performance of live gesture recognition compared to the results from steps 1 and 2? Why?

In the processing code, I used wekazing library which is a limited version of Weka. The classifier used in the code is Logistic classifier. A schematic view of the live gesture recognition process is illustrated in Figure 2.

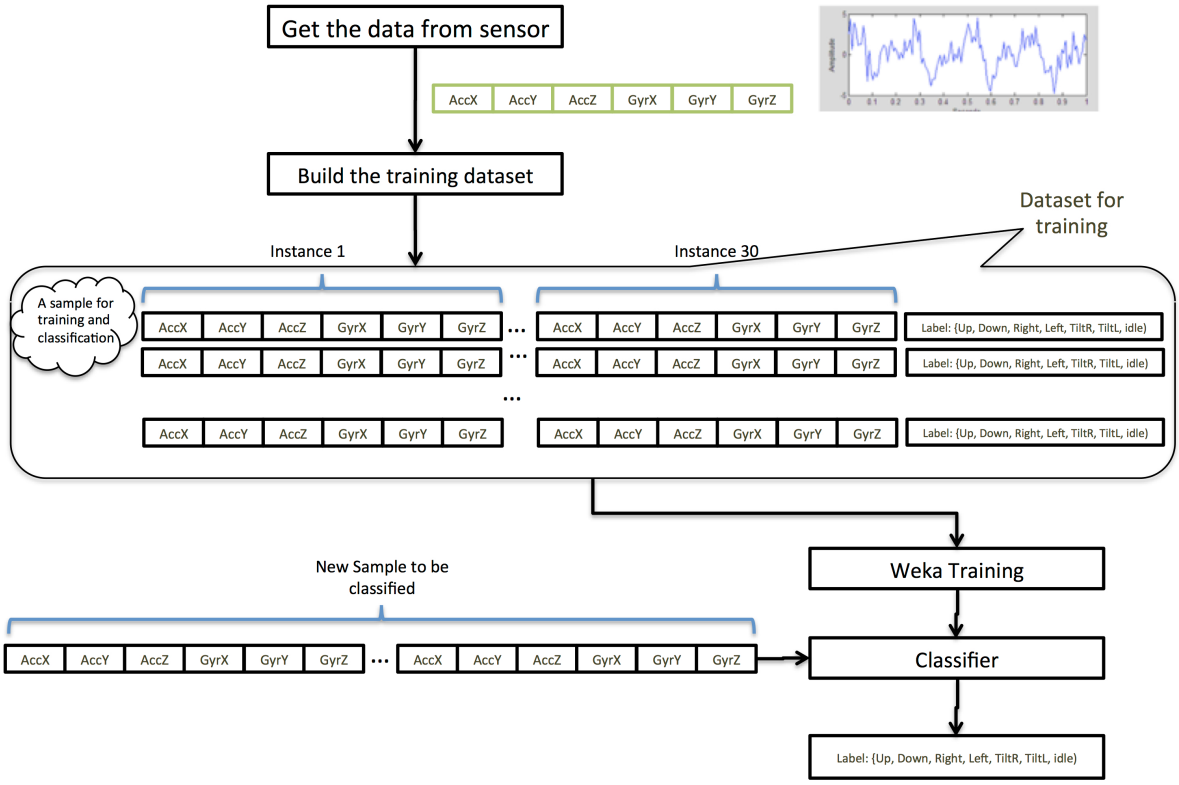

Figure 2: A schematic view of the live gesture recognition process