HUMAN ACTIVITY RECOGNITION FROM ACCELEROMETER AND GYROSCOPE DATA

Jafet Morales

University of Texas at San Antonio
8/8/2013
HUMAN ACTIVITY RECOGNITION (HAR)

- My goal is to recognize low level activities, such as:
  - walking
  - running
  - jumping jacks
  - lateral drills
  - jab cross punching
  - Michael Jackson’s moonwalk
ACCELEROMETER BASED CLASSIFICATION

- Most accelerometer based HAR done with supervised learning algorithms [1]
  - Class labels for training feature vectors are known beforehand
  - As opposed to unsupervised learning, where only the number of classes is known
**Equipment**

**Fig. 1.** A motion tracking device that can measure acceleration and angular velocity
InvenSense MPU-6050

- 6-axis motion tracking device
  - Accelerometer and gyro sensor
- 4x4x.9mm
- 16-bit output that can span 4 different ranges selected by user
- Can add an additional 1-axis (i.e., digital compass) or 3-axis sensor through I2C
- Onboard Digital Motion Processor (DMP) can process signals from all sensors
• MPU-6050 is hosted by another device
• Hosting device provides us with samples at 50Hz sampling rate
OUR METHOD - PREPROCESSING

- We calculate PCA transform on frames of accelerometer and gyroscope data
- After calculating PCA, the coordinate system is changed
  1. The principal component will have the most variance
  2. The next component will have the maximum variance possible while being orthogonal to the principal component
  3. The last component will be orthogonal to these two components, and point in the direction given by the right hand rule
AFTER PCA

- Jogging frames

- Acceleration components 1 and 2 (m/s^2)

- Angular velocity components 1 and 2 (rad/s)
**Signals for feature extraction**

*Table 3.* List of signals from which to extract features.

<table>
<thead>
<tr>
<th>Signal type</th>
<th>Axis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Acceleration</strong></td>
<td>along <em>Principal</em> axis</td>
</tr>
<tr>
<td></td>
<td>along <em>Second</em> axis</td>
</tr>
<tr>
<td></td>
<td>along <em>Third</em> axis</td>
</tr>
<tr>
<td><strong>Angular speed</strong></td>
<td>about <em>Principal</em> axis</td>
</tr>
<tr>
<td></td>
<td>about <em>Second</em> axis</td>
</tr>
<tr>
<td></td>
<td>about <em>Third</em> axis</td>
</tr>
</tbody>
</table>
**Original Feature Set**

- Use features from papers [2][3]
- And introduced some new features
- From all of those features, only a few were selected to be used in the system
- The process by which we select an optimum set of features is called **feature selection**
**Greedy Backwards Search for Feature Selection**

- Preselect a set of features
- Iteratively
  - Remove one feature at a time
  - The one that maximizes a goodness metric after it is deleted
- Stop when accuracy cannot be increased anymore or there is only one feature left
THE "GOODNESS METRIC"

- A correlation-based feature selection method has been used [4]
  - Correlation between each feature and class is maximized in the final feature set
  - Correlation among features is minimized in the final feature set
- Goodness metric is [5]

\[ Merit_S = \frac{k\bar{r}_{cf}}{\sqrt{k + k(k - 1)\bar{r}_{ff}}} \]  \hspace{1cm} (1)

where \( k \) is the size of the subset, \( \bar{r}_{ff} \) is the mean for Pearson’s correlation between each feature-feature pair and \( \bar{r}_{cf} \) is the mean for Pearson’s correlation between each feature-class pair.
Some features are discrete
- Classes are categorical
- We need to discretize all features and classes to be able to calculate the goodness metric [4]
- We then use a quantity that works with discrete variables instead of Pearson’s cross correlation $r$
  - mutual information or information gain
- First define **entropy information**

$$H(Y) = - \sum_{y \in Y} p(Y) \log_2 p(Y)$$

where $Y$ is a discrete variable and $p$ is its pdf
Conditional entropy

\[ H(Y|X) = - \sum_{x \in X} p(X) \sum_{y \in Y} p(Y|X) \log_2 p(Y|X) \]

is the expected amount of information in \( Y \) when \( X \) is known.
Information gain or mutual information

\[ \text{inf. gain}(X,Y) = H(Y) - H(Y|X) \]

- Is the reduction in the information entropy of \( Y \) when \( X \) is known
- If it is easy to predict \( Y \) by looking at \( X \) (i.e., each \( X \) maps to a single \( Y \)), then \( H(Y|X) \) will be low and inf. gain (mutual info) will be high. Also,

\[ \text{inf. gain}(X,Y) = \text{inf. gain}(Y,X) \]
<table>
<thead>
<tr>
<th>Sensor</th>
<th>Axis</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accel.</td>
<td>Mean acceleration (1,2,3)</td>
<td>$\mu_a$</td>
</tr>
<tr>
<td>Accel.</td>
<td>Acceleration signals Correlation (1,3)</td>
<td>[ \frac{1}{N} \sum_{n=0}^{N} (a_1[n] - \mu_1)(a_3[n] - \mu_3) ] [ \sigma_1 \sigma_3 ]</td>
</tr>
<tr>
<td>Accel.</td>
<td>Standard deviation (1)</td>
<td>$\sigma_{a_1}$</td>
</tr>
<tr>
<td>Accel.</td>
<td>Signal Magnitude Area (1)</td>
<td>[ \frac{1}{N} \sum_{n=0}^{N}</td>
</tr>
<tr>
<td>Accel.</td>
<td>Power (1)</td>
<td>[ \frac{1}{N} \sum_{n=0}^{N} a_1^2[n] ]</td>
</tr>
<tr>
<td>Accel.</td>
<td>Power (3)</td>
<td>[ \frac{1}{N} \sum_{n=0}^{N} a_3^2[n] ]</td>
</tr>
<tr>
<td>Accel.</td>
<td>Power contained in [8.1Hz to 16.1Hz] (2)</td>
<td>Power [8.1Hz to 16.1Hz]</td>
</tr>
<tr>
<td>Accel.</td>
<td>Entropy (1)</td>
<td>$H(a_1[n])$</td>
</tr>
<tr>
<td>Accel.</td>
<td>Entropy (2)</td>
<td>$H(a_2[n])$</td>
</tr>
<tr>
<td>Accel.</td>
<td>Entropy (3)</td>
<td>$H(a_3[n])$</td>
</tr>
<tr>
<td>Accel.</td>
<td>Repetitions per second (1)</td>
<td>Rep/s</td>
</tr>
<tr>
<td>Gyro</td>
<td>Mean angular speed (1,2,3)</td>
<td>$\mu_\omega$</td>
</tr>
<tr>
<td>Gyro</td>
<td>Correlation (1,2)</td>
<td>[ \frac{1}{N} \sum_{n=0}^{N} (\omega_1[n] - \mu_1)(\omega_2[n] - \mu_2) ] [ \sigma_1 \sigma_2 ]</td>
</tr>
<tr>
<td>Gyro</td>
<td>Standard deviation (1)</td>
<td>$\sigma_{\omega_1}$</td>
</tr>
<tr>
<td>Gyro</td>
<td>Signal Magnitude Area (2)</td>
<td>[ \frac{1}{N} \sum_{n=0}^{N}</td>
</tr>
<tr>
<td>Gyro</td>
<td>Power contained in [10Hz to 20Hz] (3)</td>
<td>Power [10Hz-20Hz]</td>
</tr>
<tr>
<td>Gyro</td>
<td>Entropy (1)</td>
<td>$H(\omega_1[n])$</td>
</tr>
</tbody>
</table>
NORMALIZING MUTUAL INFORMATION

- Let’s say we have a feature $Y$ and a feature $X$ that is highly correlated with it.
- Then $H(Y|X)$ will be zero.
- In such case the $\text{inf. gain}(Y,X) = H(Y)$.
- This means that for a uniform distribution, the more categories a feature has, the higher the info. gain it will get when compared to the classes or to any other feature.
- So we normalize $\text{inf. gain}$ to always get a value in the range $[0,1]$

$$\text{sym. uncertainty}(X,Y) = 2 - \frac{\text{inf. gain}(X,Y)}{H(X) + H(Y)}$$

- Then we substitute sym. uncertainty into (1)
CLASSIFICATION STAGE

- Tried 2 classifiers
  - kNN
  - Multilayer perceptron
**kNN**

1. Find $k$ nearest training sequences in feature space using Euclidean distance

2. For each class (i.e., walking, running) add the number of cases that fall within the $k$ nearest neighbors for both classifiers

4. Select the class with the most votes
MULTILAYER PERCEPTRON

- 1 hidden layer with 12 units
  - Obtained by using rule of thumb:
    - \# hidden units = round(\# attributes + \# classes) / 2 = round((18+5)/2)=12

\[
\begin{align*}
& x_1 & w_{x1,1} & f_1 & w_{1,19} & f_{19} & w_{19,31} & f_{31} & y_1 \\
& x_2 & & f_2 & & f_{20} & & f_{32} & y_2 \\
& \vdots & & \vdots & & \vdots & & \vdots & \vdots \\
& x_{18} & w_{x18,18} & f_{18} & w_{18,30} & f_{30} & w_{30,35} & f_{35} & y_5
\end{align*}
\]
Multilayer Perceptron Training

- MLP was trained by using backpropagation algorithm
- This algorithm uses gradient descent to train the weights in the network
  - In other words, modify each weight in the direction that will diminish the error at the output of the layer
    \[ \mathbf{v} = \mathbf{v}_{old} - \mu \nabla \text{Error}(\mathbf{v}_{old}) \]
  - Gradient is made of partial derivatives, and in case of weights, each partial derivative corresponds to a weight
So to modify each weight individually, we use the partial derivative of an error function with respect to that weight [6][7]:

\[
\frac{\partial E_p}{\partial w_{ij}} = \frac{\partial E_p}{\partial a_j} \cdot \frac{\partial a_j}{\partial w_{ij}}
\]

\[
E_p = \frac{1}{2} \sum_{j=\text{first node in current layer}}^{\text{last node in current layer}} (t_j - a_j)^2
\]

\[
\frac{\partial E_p}{\partial a_j} = -(t_j - a_j)
\]

the only unknown parameter that impedes gradient descent!!!

\[
\frac{\partial a_j}{\partial w_{ij}} = \frac{\partial a_j}{\partial e_j} \cdot \frac{\partial e_j}{\partial w_{ij}} = \frac{\partial f_j(e_j)}{\partial e_j} \cdot \frac{\partial \sum w_{ij} a_i}{\partial w_{ij}} = \frac{\partial f_j(e_j)}{\partial e_j} a_i
\]

\[
i = \text{input node}
\]

\[
j = \text{output node}
\]

\[
t_j = \text{desired output at node } j
\]

\[
a_j = \text{output of node } j = f_j(e_j)
\]

\[
e_j = \sum_{i=\text{first node in input layer}}^{\text{last node in input layer}} w_{ij} a_i
\]
we may not know $t_j - a_j$ because we do not know $t_j$ except at the outermost layer, but we can guess it

- The guess for $t_j - a_j$ at each node is calculated by backpropagating the error at the output to the neurons in previous layers.
- This is done by reversing the arrows in the network and using $t_{(one \ of \ last \ layer \ nodes)} - a_{(one \ of \ last \ layer \ nodes)}$ as inputs.

Then we use $\delta_j = \text{error propagated back to node } j$ as a substitute for $t_j - a_j$.

It turns out even if you reach a peak for $t_j - a_j$, you will be approaching a valley for the last layer.
REAL-TIME SETUP

- Feature extraction
  - kNN Vote(s)
  - MP Vote(s)
- Transformation
  - LPF
  - max
- motion tracking signals
- class
EVALUATION OF RESULTS

- Train and test on single subject
- Train on several subjects and test on one subject
## Results

| Simulation of method in [5] | | | |
|-----------------------------|------------------|------------------|
| Single subject              | Unknown subject  | |
| C4.5                        | kNN              | C4.5             |
| 82.2                        | 93.1             | 56               |
| 63.2                        |                  |                  |

| Simulation of method in [7] | | | |
|-----------------------------|------------------|------------------|
| Single subject              | Unknown subject  | |
| C4.5                        | MP               | C4.5             |
| 77.1                        | 96.2             | 68               |
| 66.4                        |                  |                  |

| Simulation of proposed method | | | |
|-----------------------------|------------------|------------------|
| Single subject              | Unknown subject  | |
| KNN                         | MP               | KNN              |
| 100                         | 100              | 98.4             |
| 99.2                        |                  |                  |
Table 3. Confusion table for proposed method

<table>
<thead>
<tr>
<th></th>
<th>Still</th>
<th>Walk</th>
<th>Jog</th>
<th>Jump jack</th>
<th>Squat</th>
</tr>
</thead>
<tbody>
<tr>
<td>Still</td>
<td>25</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Walk</td>
<td>0</td>
<td>25</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Jog</td>
<td>0</td>
<td>0</td>
<td>25</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Jump jack</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>25</td>
<td>0</td>
</tr>
<tr>
<td>Squat</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>24</td>
</tr>
</tbody>
</table>
CONCLUSION

- The proposed algorithm allows for highly accurate human activity recognition without imposing any constraints on the user, except for the requirement to place the smartphone in his front right pocket.
REFERENCES


