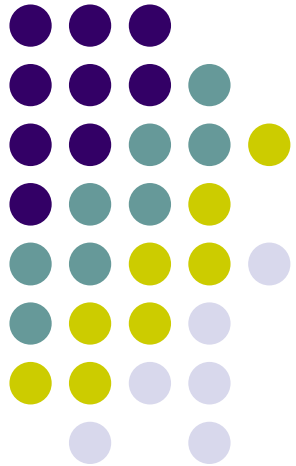
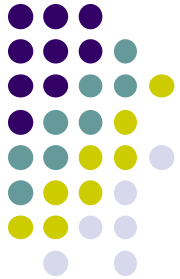
CS 4518 Mobile and Ubiquitous Computing

Lecture 13: Machine Learning for Ubiquitous Computing

Emmanuel Agu

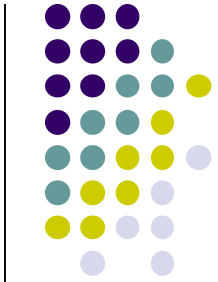


Reminder: 1 Slide of Final Project

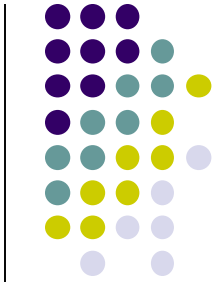


- 1-slide from group today, extended till tomorrow Tuesday (2/7):
  - 2/40 of final project grade
- Propose mobile/ubiquitous computing app, solves WPI problem
- Slide should contain 3 bullets
  1. **Problem you intend to work on**
     - Solve WPI/societal problem (e.g. walking safe at night)
     - Use at least location, 1 sensor or camera
     - If games, must gamify solution to real world problem
  2. **Why this problem is important**
     - E.g. 37% of WPI students feel unsafe walking home
  3. **Summary of envisioned mobile app (?) solution**
     1. E.g. Mobile app automatically texts users friends when they get home at night
- Can bounce ideas of me (email, or in person)
- Can change idea any time

Rubric: Grading Considerations

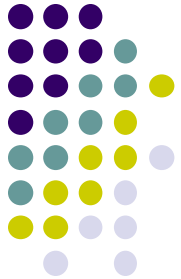


- **Problem (30/100)**
  - How much is the problem a real problem (e.g. not contrived)
  - Is this really a good problem that is a good fit to solve with mobile/ubiquitous computing? (e.g. are there better approaches?)
- **Importance (30/100)**
  - How useful would it be if this problem is solved?
  - What is the potential impact on the community (e.g. WPI students) (e.g. how much money? Time? Productivity.. Would be saved?)
  - What is the evidence of the importance? (E.g. quote a statistic)
- **Proposed Solution (40/100)**
  - How good/clever is the solution?
  - How sophisticated and how many are the mobile/ubiquitous computing components (high level) proposed? (e.g. location, geofencing, activity recognition, face recognition, machine learning, etc)



Intuitive Introduction to Machine Learning for Ubiquitous Computing

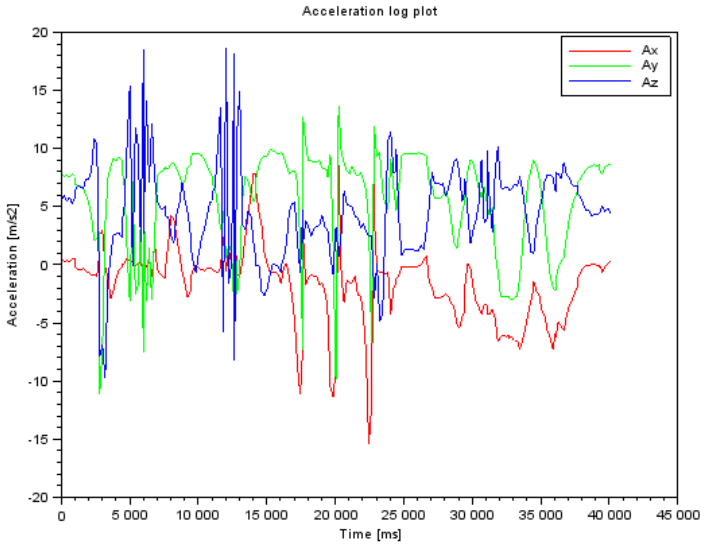
My Goals in this Section



- If you know machine learning
  - Set off light bulb
  - Projects involving ML?
- If you don’t know machine learning
  - Get general idea, how it’s used
- Knowledge will also make papers easier to read/understand

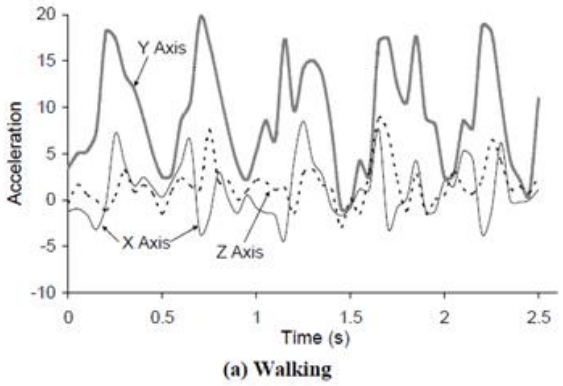
Recall: Activity Recognition

- Want app to detect when user is performing any of the following 6 activities
  - Walking,
  - Jogging,
  - Ascending stairs,
  - Descending stairs,
  - Sitting,
  - Standing



Recall: Activity Recognition Overview

Gather Accelerometer data



(a) Walking

Machine Learning Classifier

Classify Accelerometer data

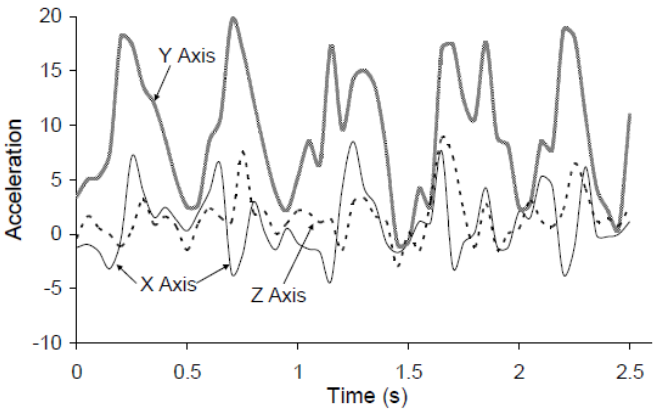
Walking

Running

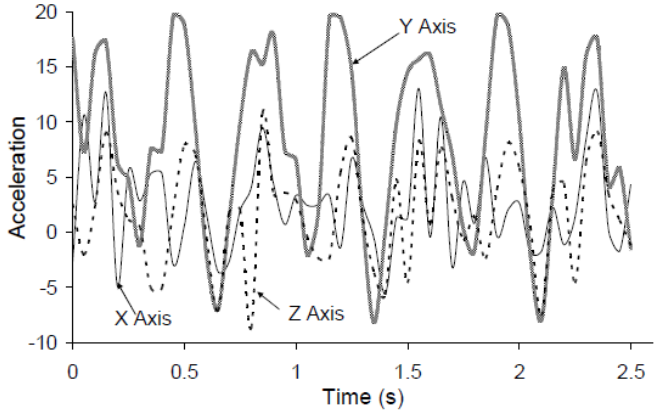
Climbing Stairs

Recall: Example Accelerometer Data for Activities

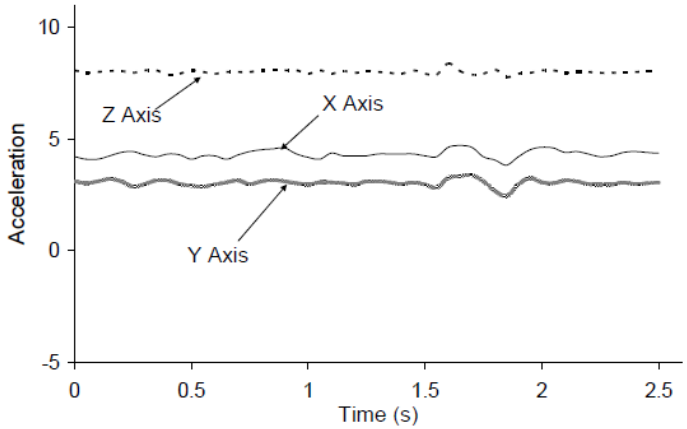
Different user activities generate different accelerometer patterns



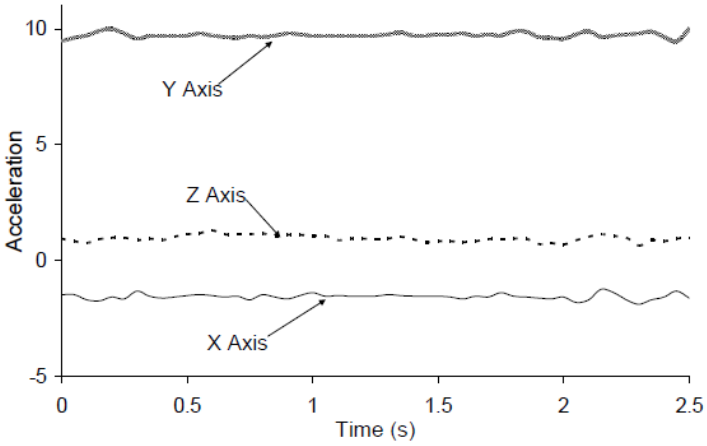
(a) Walking



(b) Jogging



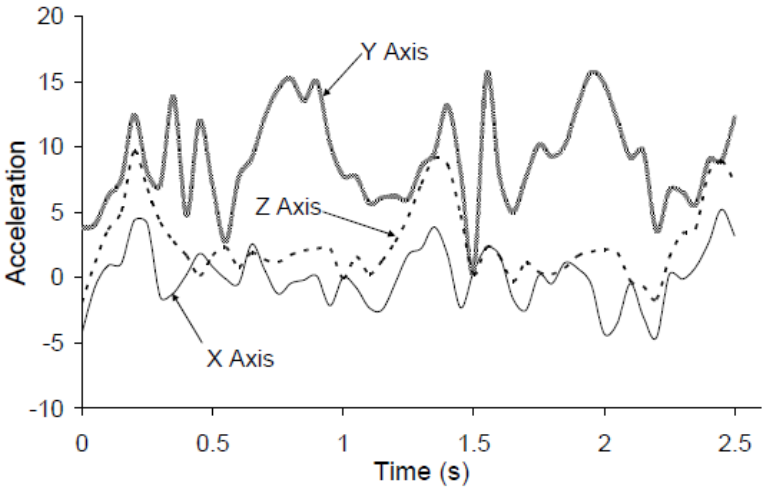
(e) Sitting



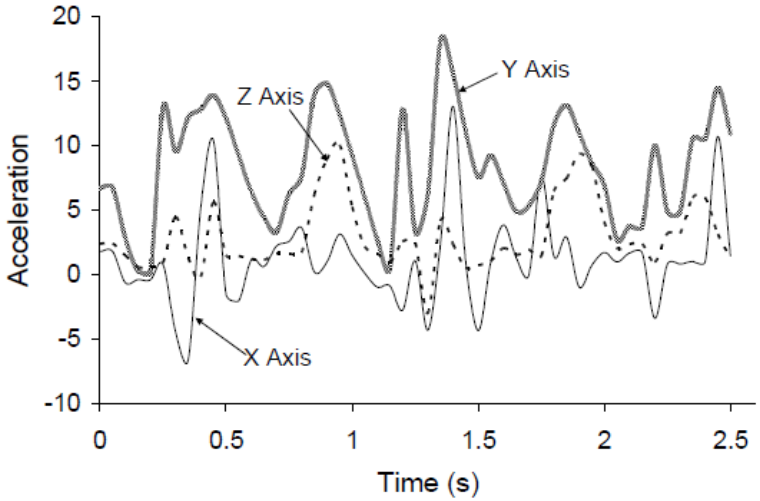
(f) Standing

Recall: Example Accelerometer Data for Activities

Different user activities generate different accelerometer patterns



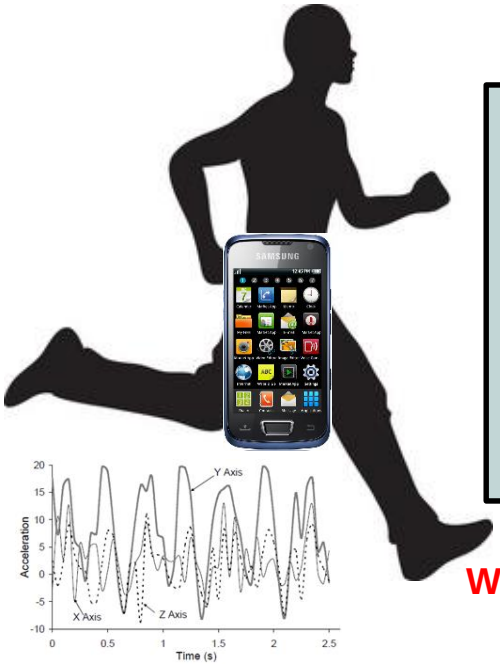
(c) Ascending Stairs

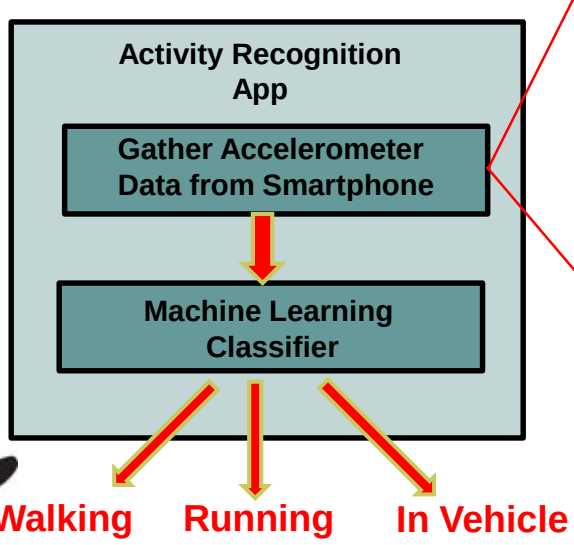


(d) Descending Stairs

DIY Activity Recognition (AR) Android App

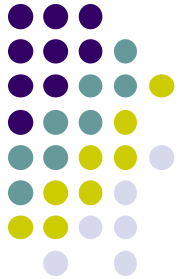
- As user performs an activity, AR app on user's smartphone
  1. Gathers accelerometer data
  2. Uses **machine learning classifier** to determine what activity (running, jumping, etc) accelerometer pattern corresponds to
- **Classifier:** Machine learning algorithm that guesses what activity **class** accelerometer sample corresponds to






```
msensor = (mSensorManager)
    getSystemService(Context.SENSOR_SERVICE)

....

Public void onSensorChanged(SensorEvent event){

....

}
```

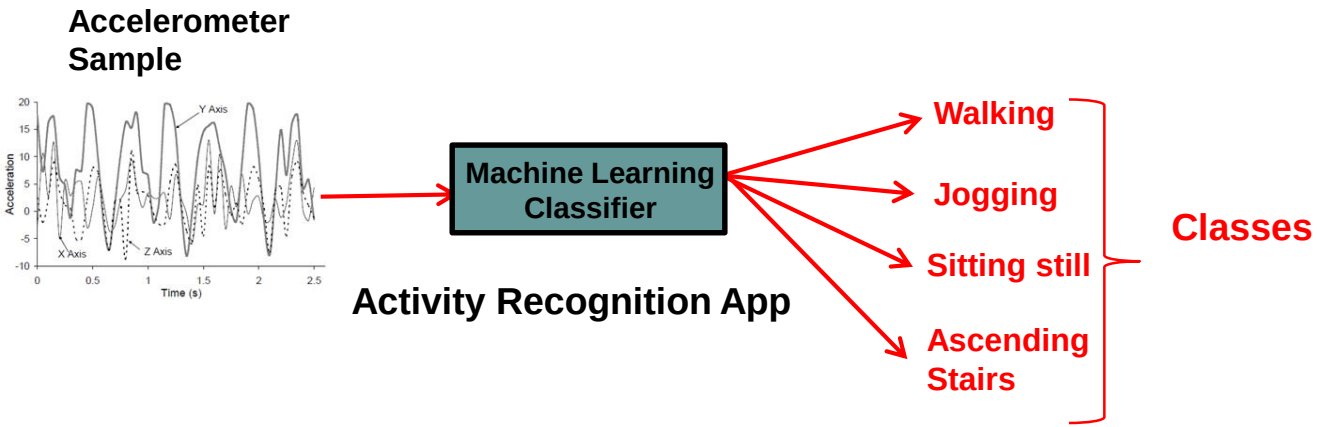
Next: Machine learning Classification

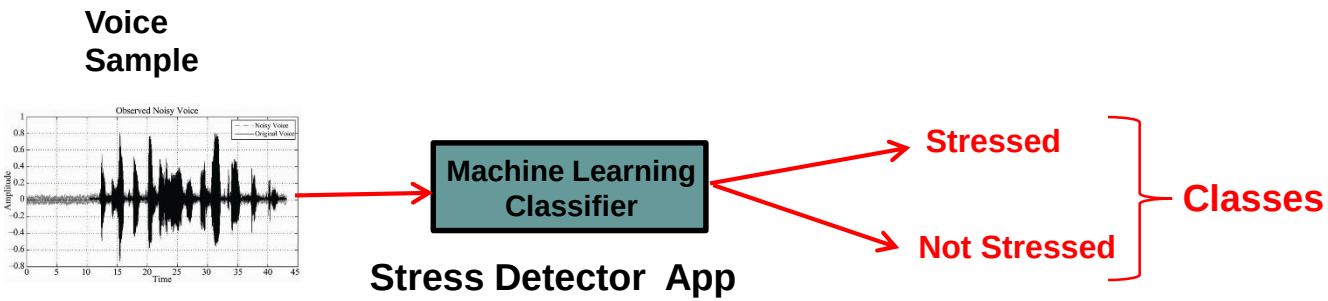


Classification for Ubiquitous Computing

Classification

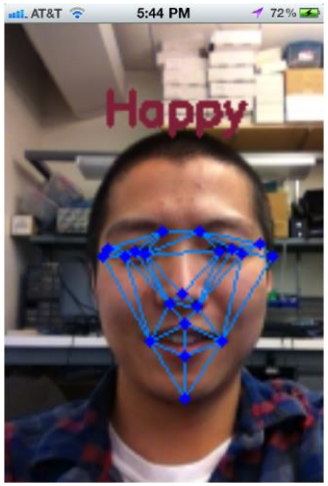
- **Classification** is type of machine learning used a lot in Ubicomp
- Classification? determine which **class** a sample belongs to. Examples:





Classification

Image showing
Facial Expression



Machine Learning Classifier

- Anger
- Disgust
- Fear
- Happy
- Neutral
- Sadness
- Surprise

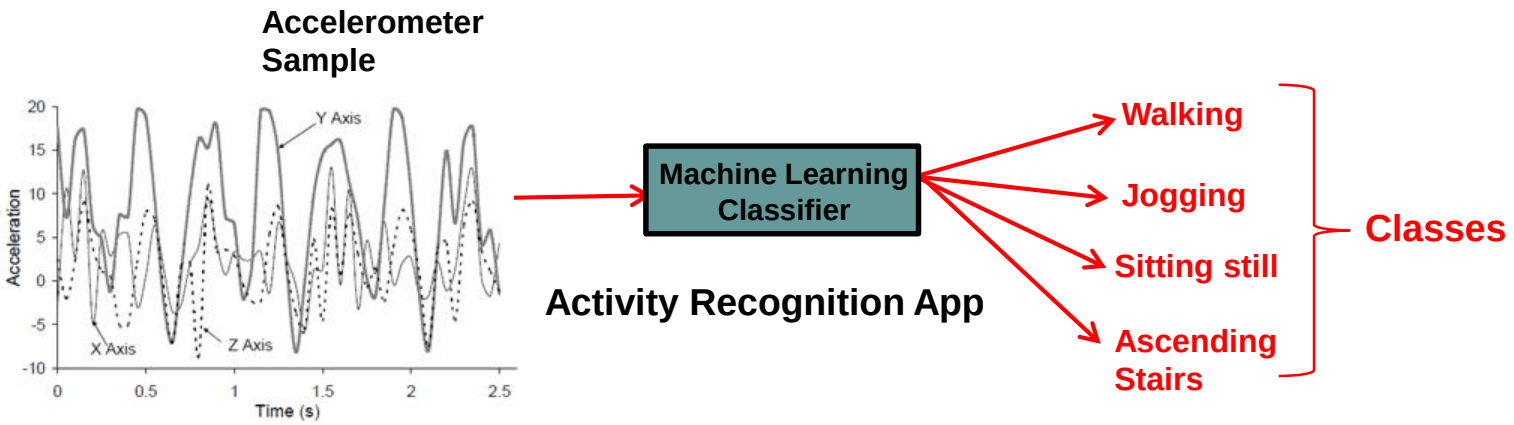
Classes

Facial Interpretation
App

Classifier

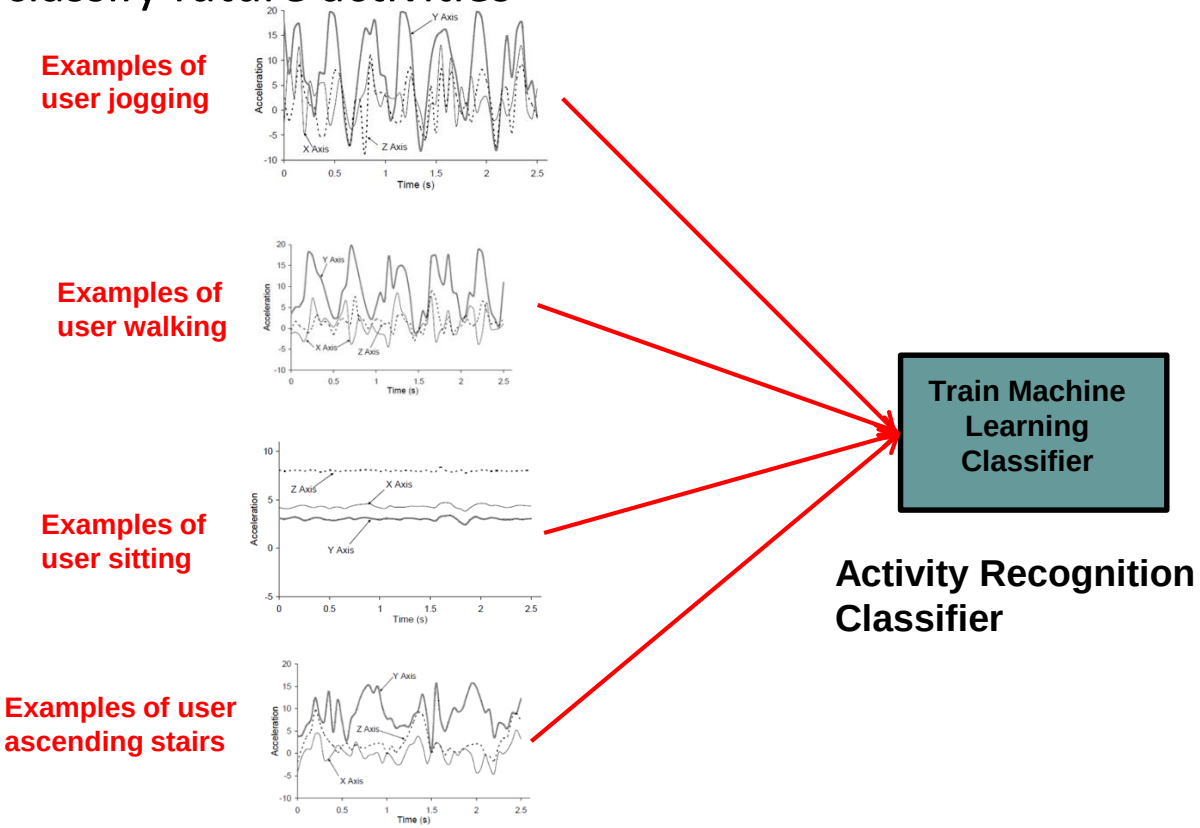
- Analyzes new sample, guesses corresponding class
- Intuitively, can think of classifier as set of rules for classification. E.g.
- Example rules for classifying accelerometer signal in Activity Recognition

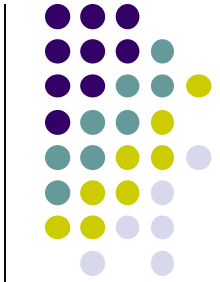
```
If ((Accelerometer peak value > 12 m/s)
      and (Accelerometer average value < 6 m/s)){
            Activity = "Jogging";
}
```



Training a Classifier

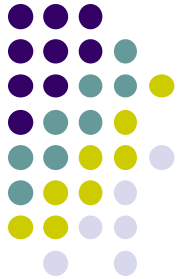
- Created using example-based approach (called training)
- **Training a classifier:** Examples of each class => generate rules to categorize new samples
- **E.g:** Analyze 30+ Examples (from 30 subjects) of accelerometer signal for each activity type (walking, jogging, sitting, ascending stairs) => generate rules (classifier) to classify future activities





Training a Classifier: Steps

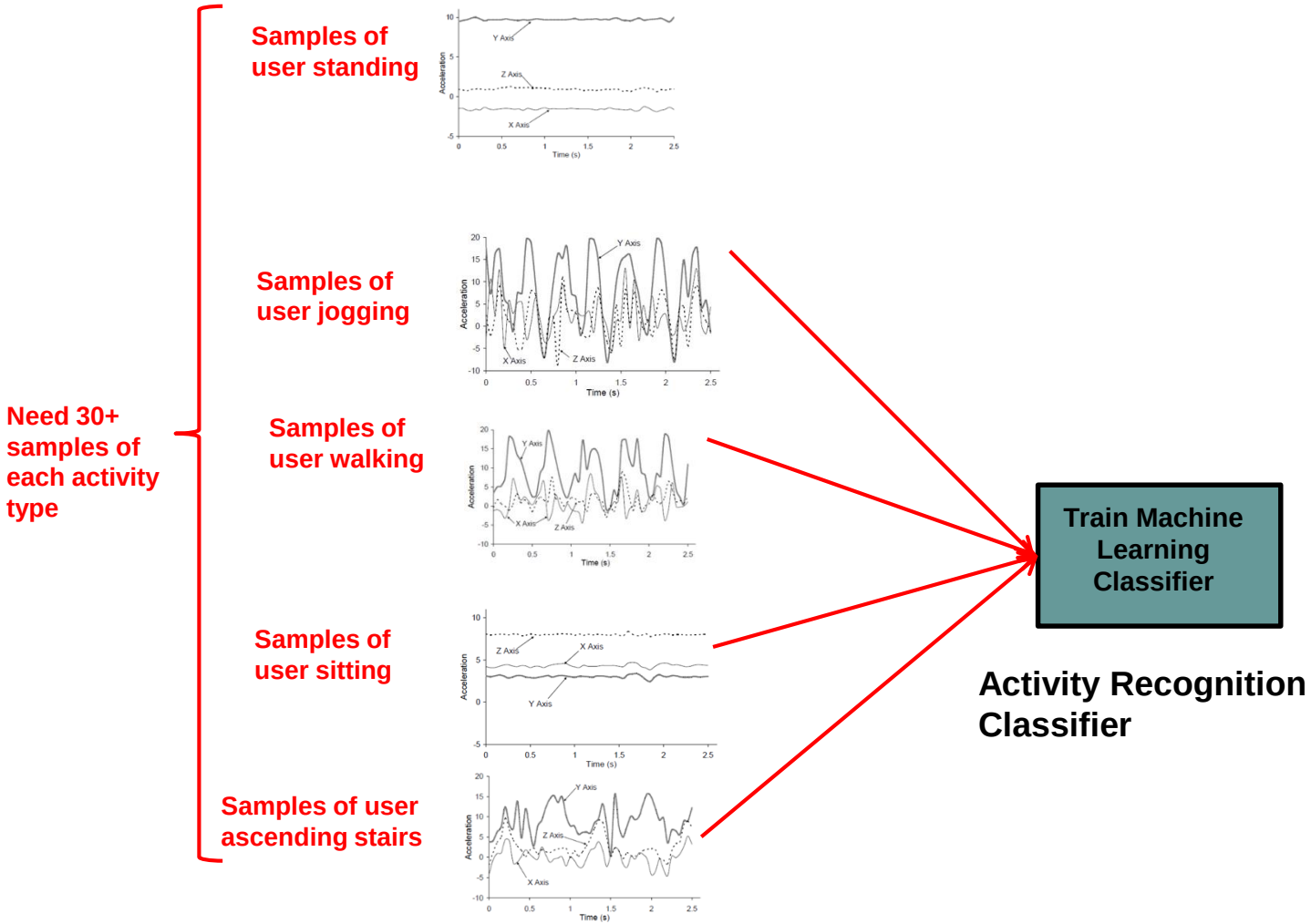
Steps for Training a Classifier



1. Gather data samples + label them
2. Import accelerometer samples into classification library (e.g. Weka, MATLAB)
3. Pre-processing (segmentation, smoothing, etc)
4. Extract features
5. Train classifier
6. Export classification model as JAR file
7. Import into Android app

Step 1: Gather Sample data + Label them

- Need many samples of accelerometer data corresponding to each activity type (jogging, walking, sitting, ascending stairs, etc)



Step 1: Gather Sample data + Label them

- Run a study to gather sample accelerometer data for each activity class
  - Recruit 30+ subjects
  - Run program that gathers accelerometer sensor data on subject's phone
  - Make subjects perform each activity (walking, jogging, sitting, etc)
  - Collect accelerometer data while they perform each activity (walking, jogging, sitting, etc)
  - Label data. i.e. tag each accelerometer sample with the corresponding activity
- Now have 30 examples of each activity

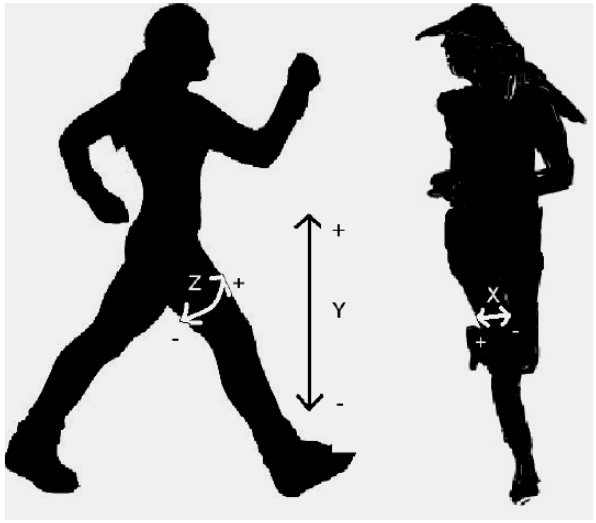
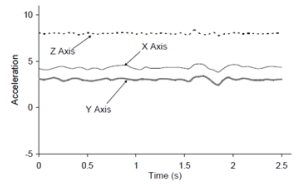
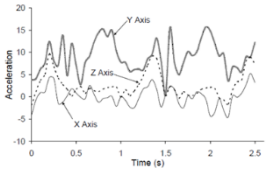


Figure 1: Axes of Motion Relative to User

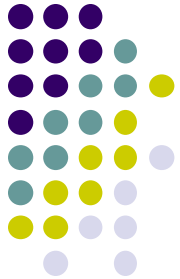
30+ Samples of user sitting



30+ Samples of user ascending stairs



Step 1: Gather Sample data + Label them
Program to Gather Accelerometer Data



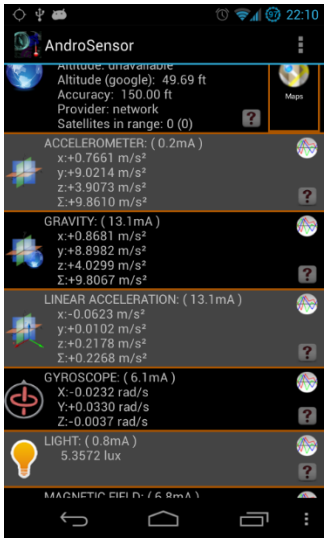
- **Option 1:** Can write sensor program app that gathers accelerometer data while user is doing each of 6 activities (1 at a time)

```
msensor = (mSensorManager)
    getSystemService(Context.SENSOR_SERVICE)

....

Public void onSensorChanged(SensorEvent event){

....

}
```

Step 1: Gather Sample data + Label them
Program to Gather Accelerometer Data

- **Option 2:** Use 3rd party app to gather accelerometer
  - 2 popular ones: **Funf** and **AndroSensor**
  - Just download app,
    - Select sensors to log (e.g. accelerometer)
    - Continuously gathers sensor data in background
- FUNF app from MIT
  - Accelerometer readings
  - Phone calls
  - SMS messages, etc
- AndroSensor

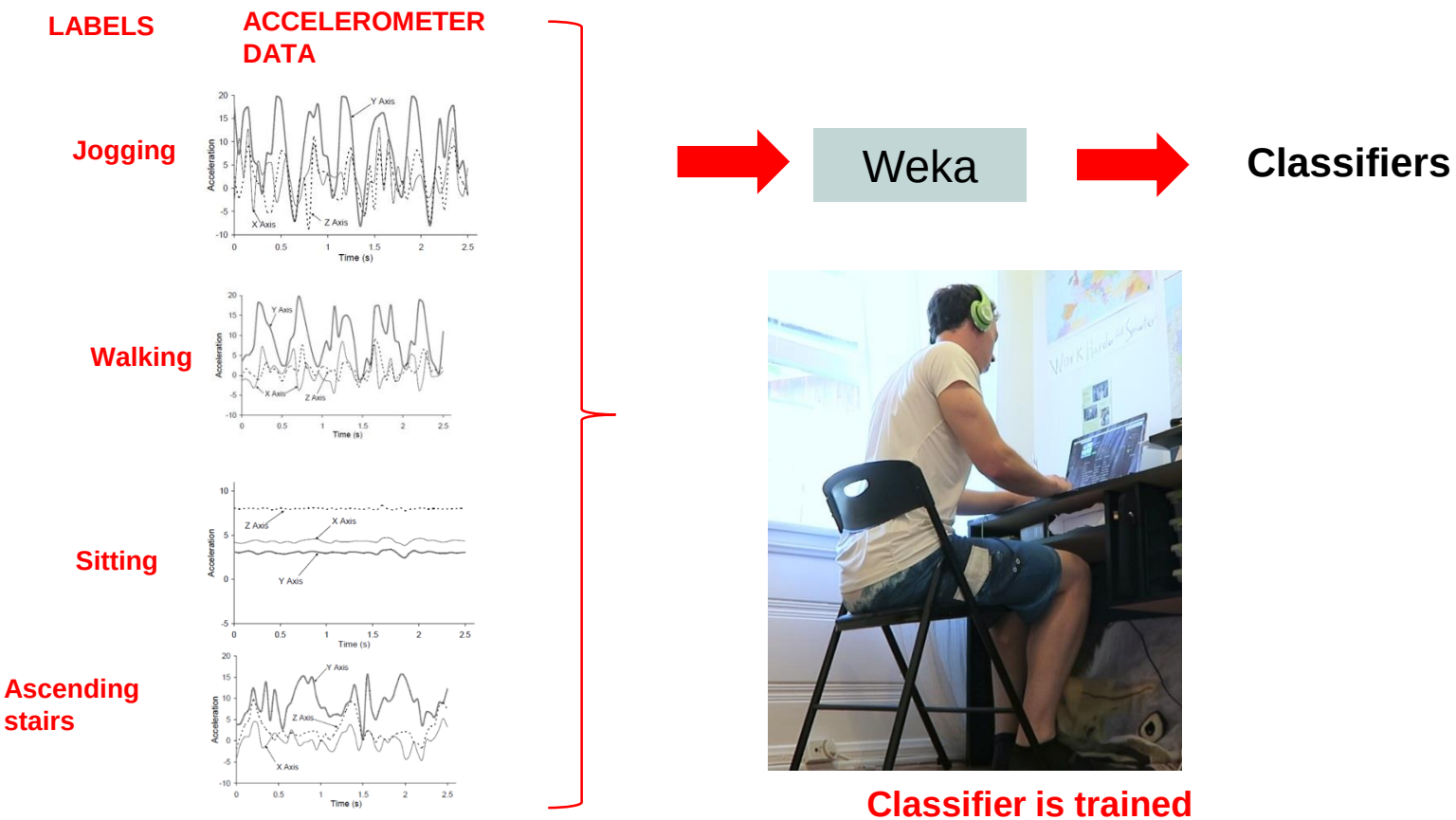
Funf



AndroSensor

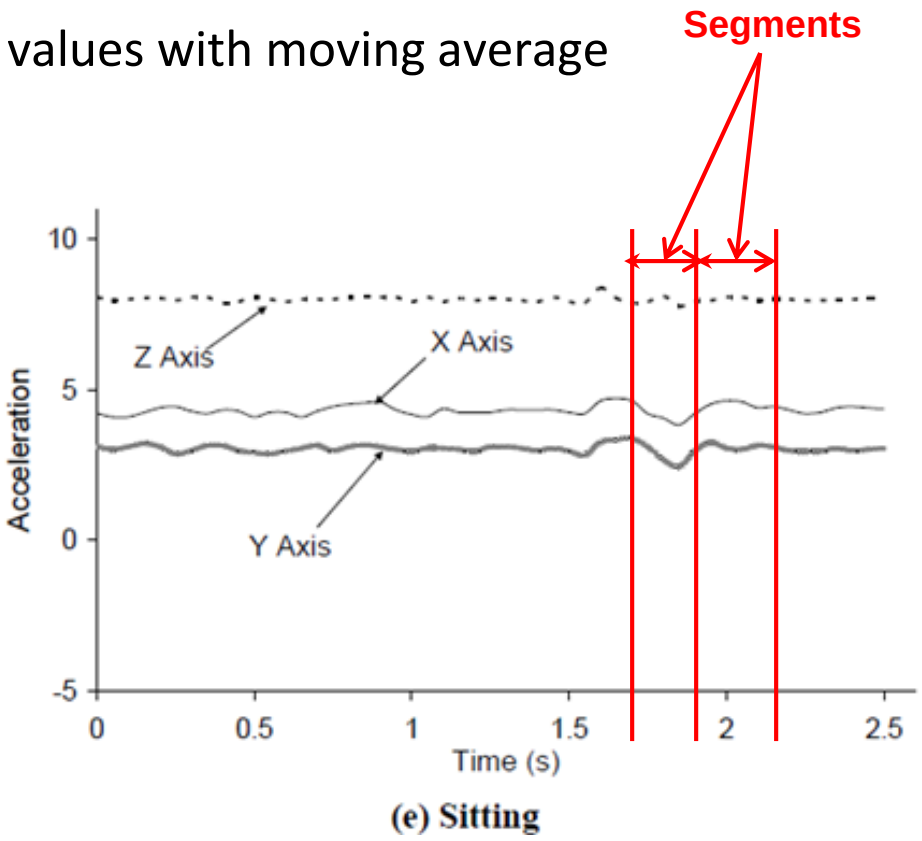
Step 2: Import accelerometer samples into classification library (e.g. Weka, MATLAB)

- Import accelerometer data (labelled with corresponding activity) into Weka (or other Machine learning Framework)



Step 3: Pre-processing (segmentation, smoothing, etc) Segment Data (Windows)

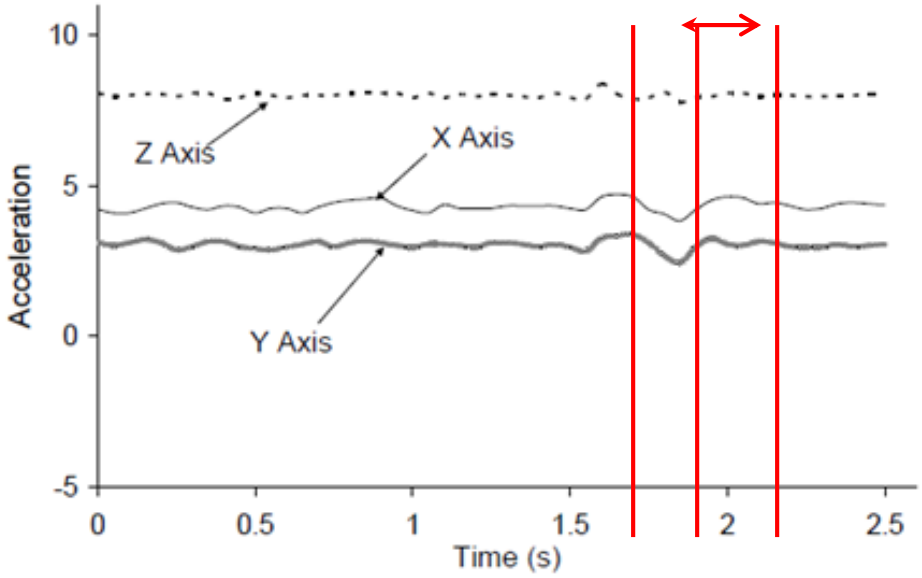
- Pre-processing data (in Weka) may include segmentation, smoothing, etc
  - **Segment:** Divide 60 seconds of raw time-series data divided into chunks(e.g. 10 seconds)
  - **Smoothing:** Replace groups of values with moving average



(e) Sitting

Step 4: Compute (Extract) Features

- For each segment (batch of accelerometer values) compute features (in Weka)
- **Features:** Functions computed on accelerometer data, captures important accelerometer characteristics
- **Examples:** min-max of values, largest magnitude within segment, standard deviation

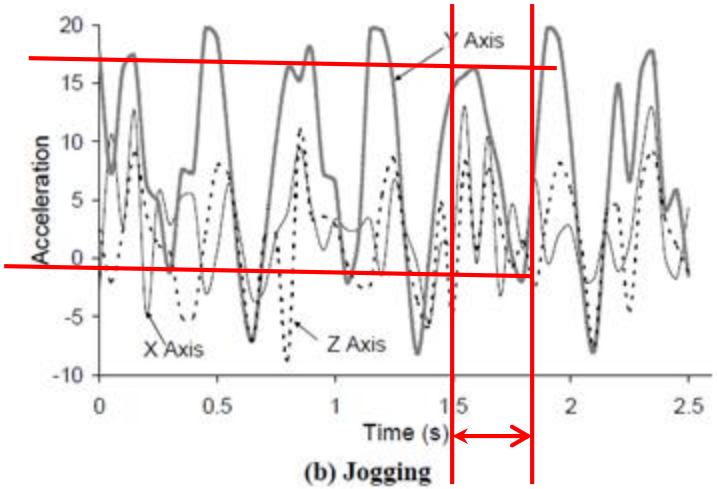


(e) Sitting

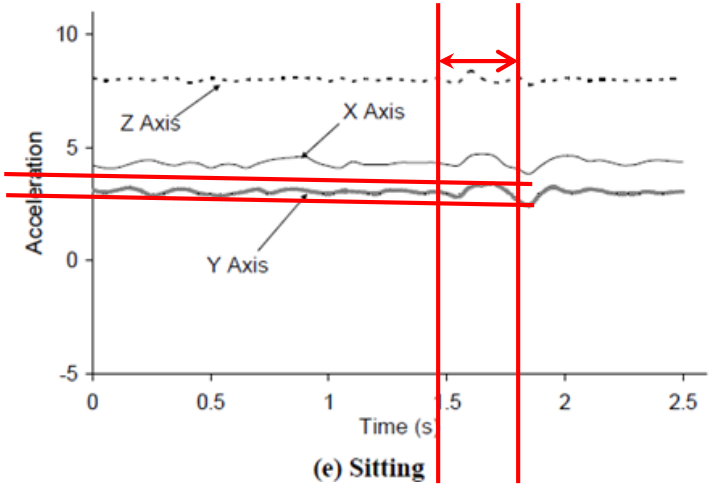
Step 4: Compute (Extract) Features

- **Important:** Ideally, values of features different for each activity type
- **E.g:** Min-max range feature

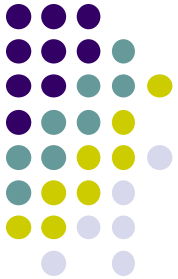
Large min-max for jogging



Small min-max for sitting



Step 4: Compute (Extract) Features

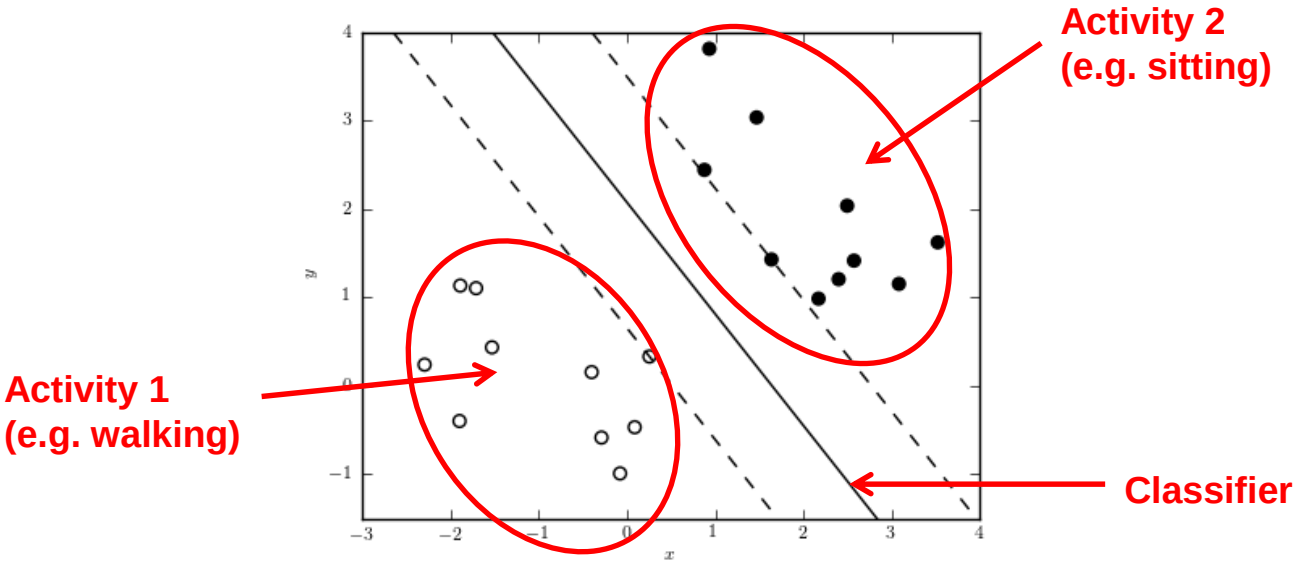


Calculate many different features

- Average[3]: Average acceleration (for each axis)
- Standard Deviation[3]: Standard deviation (for each axis)
- Average Absolute Difference[3]: Average absolute difference between the value of each of the 200 readings within the ED and the mean value over those 200 values (for each axis)
- Average Resultant Acceleration[1]: Average of the square roots of the sum of the values of each axis squared $\sqrt{(x_i^2 + y_i^2 + z_i^2)}$ over the ED
- Time Between Peaks[3]: Time in milliseconds between peaks in the sinusoidal waves associated with most activities (for each axis)
- Binned Distribution[30]: We determine the range of values for each axis (maximum – minimum), divide this range into 10 equal sized bins, and then record what fraction of the 200 values fell within each of the bins.

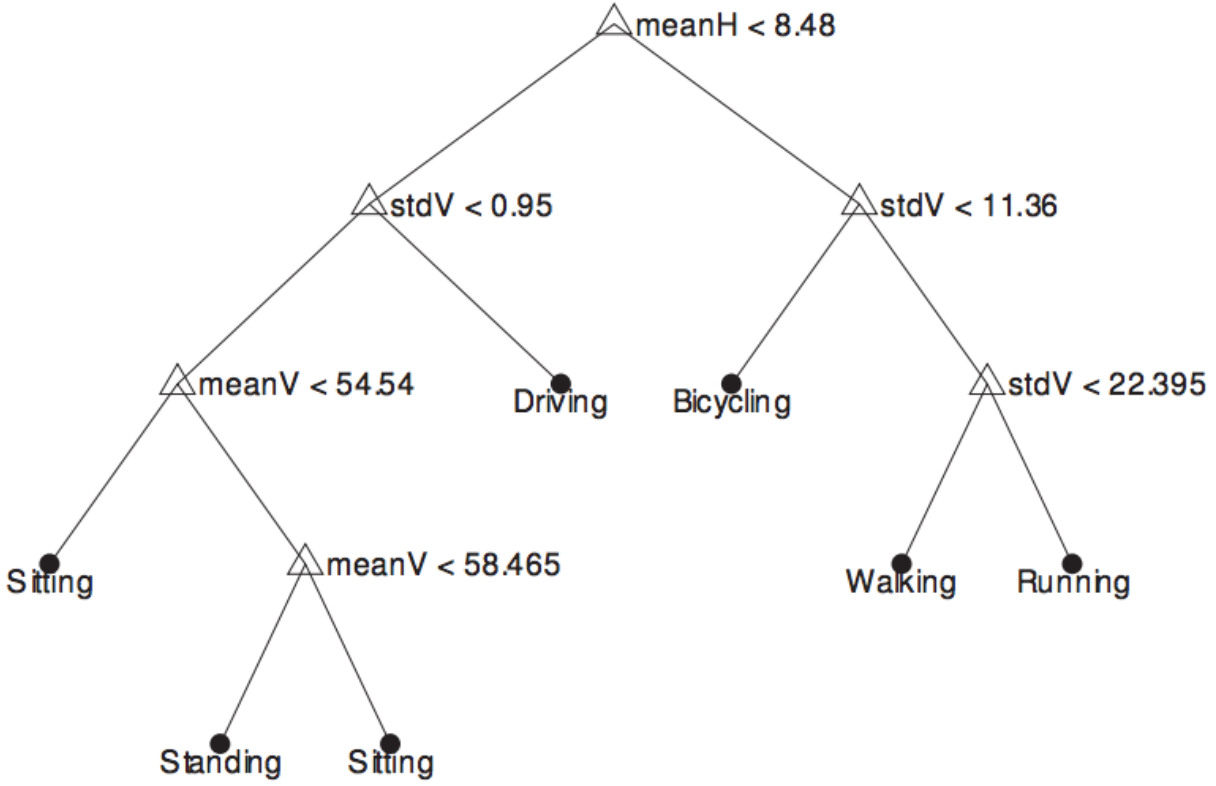
Step 5: Train classifier

- Features are just numbers
- Different values for different activities
- **Training classifier:** figures out feature values corresponding to each activity
- Weka already programmed with different classification algorithms (SVM, Decision Trees, Naïve Bayes, Random Forest, J48, logistic regression, SMO, etc)
- Try different classification algorithms, compare accuracy
- **SVM example**

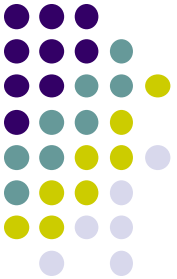


Step 5: Train classifier

- **Example:** Decision Tree Classifier
- Feature values compared against learned thresholds at each node



Step 5: Train classifier
Compare Accuracy of Classifier Algorithms



- Weka also reports accuracy of each classifier type

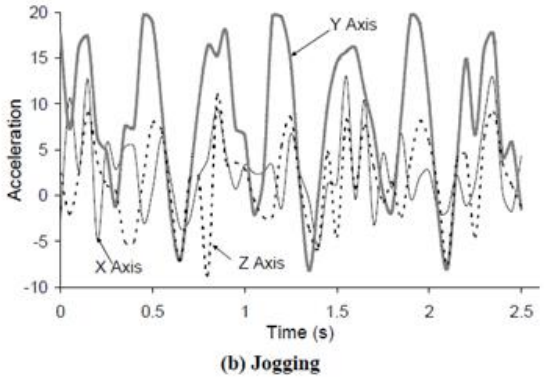
Table 2: Accuracies of Activity Recognition

	% of Records Correctly Predicted			
	J48	**Logistic Regression**	**Multilayer Perceptron**	**Straw Man**
Walking	89.9	**93.6**	91.7	37.2
Jogging	96.5	98.0	**98.3**	29.2
Upstairs	59.3	27.5	**61.5**	12.2
Downstairs	**55.5**	12.3	44.3	10.0
Sitting	**95.7**	92.2	95.0	6.4
Standing	**93.3**	87.0	91.9	5.0
Overall	85.1	78.1	**91.7**	37.2

Pick most accurate classification algorithm for all classes

Step 6: Export classification model as JAR file
Step 7: Import into Android app

- Export classification model (most accurate classifier) as Java JAR file
- Import JAR file into Android app
- In app write Android code to
  - Gather accelerometer data, segment, extract feature, classify using classifier in JAR file
- Classifies new accelerometer patterns while user is performing activity => Guess (infer) what activity

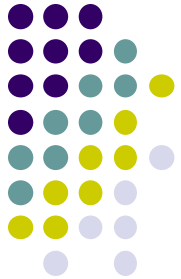


(b) Jogging

New accelerometer Sample in real time

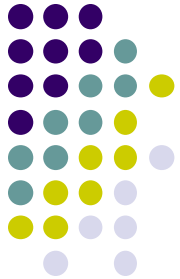
Classifier in Android app

Activity (e.g. Jogging)



Context Sensing

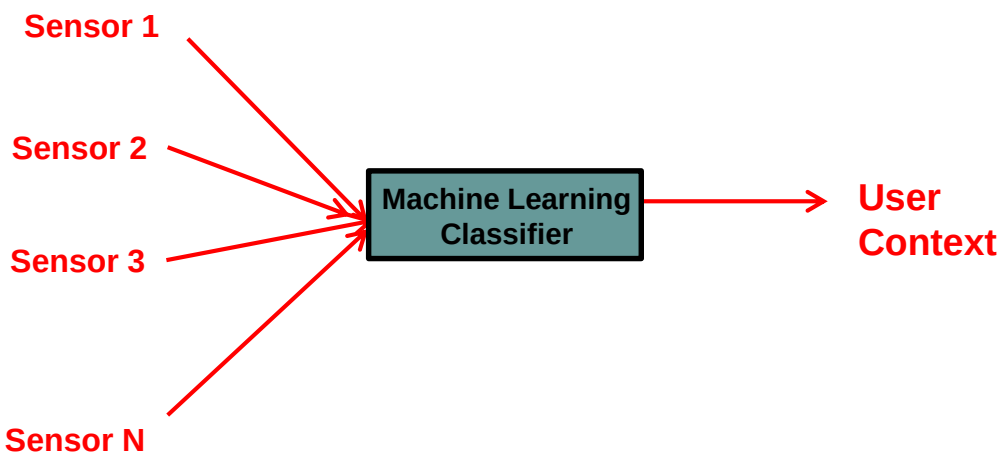
Recall: Ubicomp Senses User's Context



- Context?
  - *Human:* motion, mood, identity, gesture
  - *Environment:* temperature, sound, humidity, location
  - *Computing Resources:* Hard disk space, memory, bandwidth
  - *Ubicomp example:*
    - *Assistant senses:* Temperature outside is 10F (environment sensing) + Human plans to go work (schedule)
    - *Ubicomp assistant advises:* Dress warm!
- Sensed **environment + Human + Computer resources** = ***Context***
- *Context-Aware* applications adapt their behavior to context

Context Sensing

- Activity Recognition uses data from only accelerometer (1 sensor)
- Can combine multiple sensors, use machine learning to sense **user context**
- More later



References

- Jennifer R. Kwapisz, Gary M. Weiss, and Samuel A. Moore, Activity recognition using cell phone accelerometers, SIGKDD Explor. Newsl. 12, 2 (March 2011), 74-82.
- Deepak Ganesan, Activity Recognition, Physiological Sensing Class, UMASS Amherst