CS 4518 Mobile and Ubiquitous Computing

Lecture 9: Other Android Mobile & Ubicomp Components, Project Proposal, Machine Learning

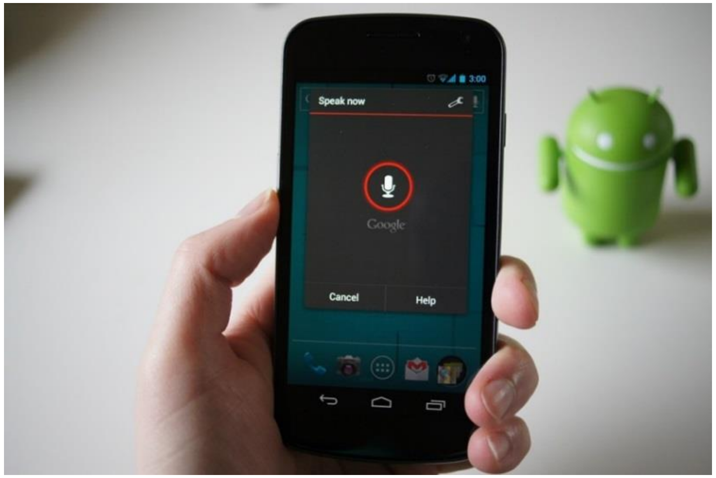
Emmanuel Agu

What other Android APIs may be useful for Mobile/ubicomp?

Speaking to Android

http://developer.android.com/reference/android/speech/SpeechRecognizer.html
https://developers.google.com/voice-actions/

- **Speech recognition:**
  - Accept inputs as speech (instead of typing) e.g. dragon dictate app?
  - Note: Requires internet access
- Two forms
  1. **Speech-to-text**
     - Convert user's speech to text. E.g. display voicemails in text
  2. **Voice Actions:** Voice commands to smartphone (e.g. search for, order pizza)

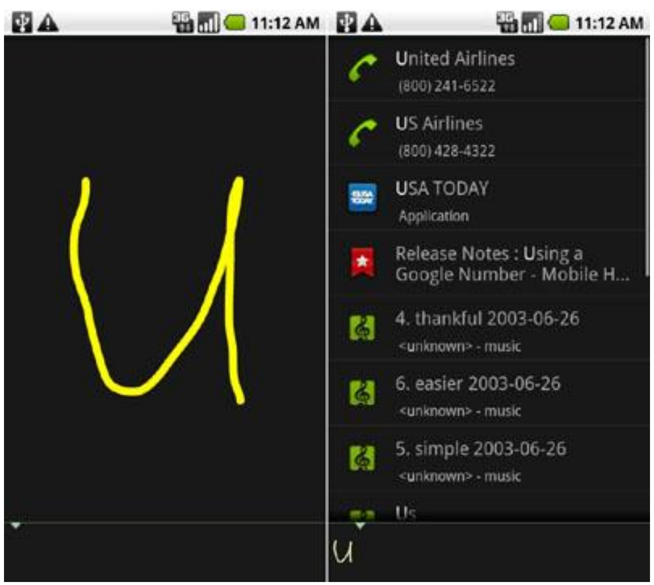


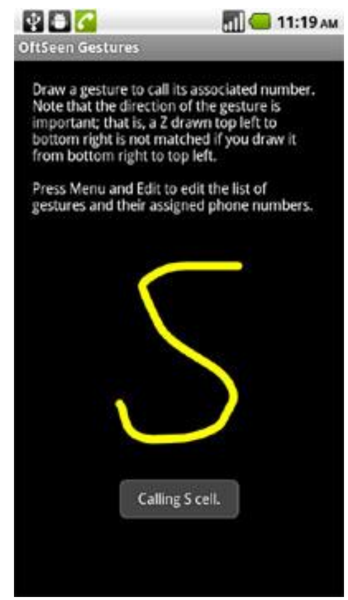
Speech to text

Gestures

https://developer.android.com/training/gestures/index.html
http://www.computerworld.com/article/2469024/web-apps/android-gestures--3-cool-ways-to-control-your-phone.html

- **Gesture:** Hand-drawn shape on the screen
- Example uses:
  - Search your phone, contacts, etc by handwriting onto screen
  - Speed dial by handwriting first letters of contact's name
  - Multi-touch, pinching





More MediaPlayer & RenderScript

http://developer.android.com/guide/topics/renderscript/compute.html

- MediaRecorder is used to **record** audio
  - Manipulate raw audio from microphone/audio hardware, PCM buffers
    - E.g. if you want to do audio signal processing, speaker recognition, etc
    - **Example:** process user's speech, detect emotion, nervousness?
  - Can playback recorded audio using MediaPlayer

- **RenderScript**
  - High level language for GPGPU
  - Use Phone's Graphics Processing Unit (GPU) for computational tasks
  - Very few lines of code = run GPU code
  - Useful for heavy duty tasks. E.g. image, video processing

Wireless Communication

http://developer.android.com/guide/topics/connectivity/bluetooth.html
http://developer.android.com/reference/android/net/wifi/package-summary.html

- Bluetooth
  - Discover, connect to nearby bluetooth devices
  - Communicating over Bluetooth
  - Exchange data with other devices

- WiFi
  - Scan for WiFi hotspots
  - Monitor WiFi connectivity, Signal Strength (RSSI)
  - Do peer-to-peer (mobile device to mobile device) data transfers

Wireless Communication

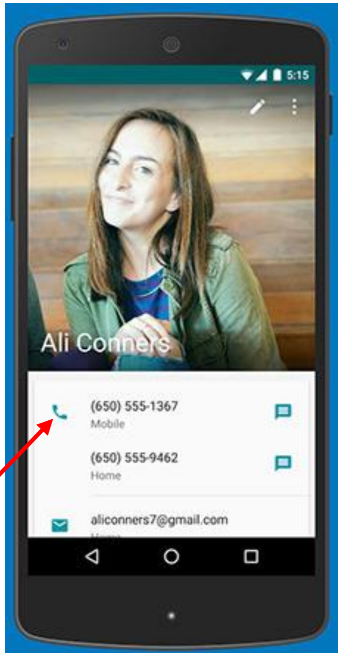
http://developer.android.com/guide/topics/connectivity/nfc/index.html

- NFC:
  - Contactless, transfer small amounts of data over short distances
  - **Applications:** Share spotify playlists, Google wallet
  - **Android Pay**
    - Store debit, credit card on phone
    - Pay by tapping terminal

Telephony and SMS

http://developer.android.com/reference/android/telephony/package-summary.html
http://developer.android.com/reference/android/telephony/SmsManager.html

- **Telephony:**
  - Initiate phone calls from within app
  - Access dialer app, etc
- **SMS:**
  - Send/Receive SMS/MMS from app
  - Handle incoming SMS/MMS in app



Dialer

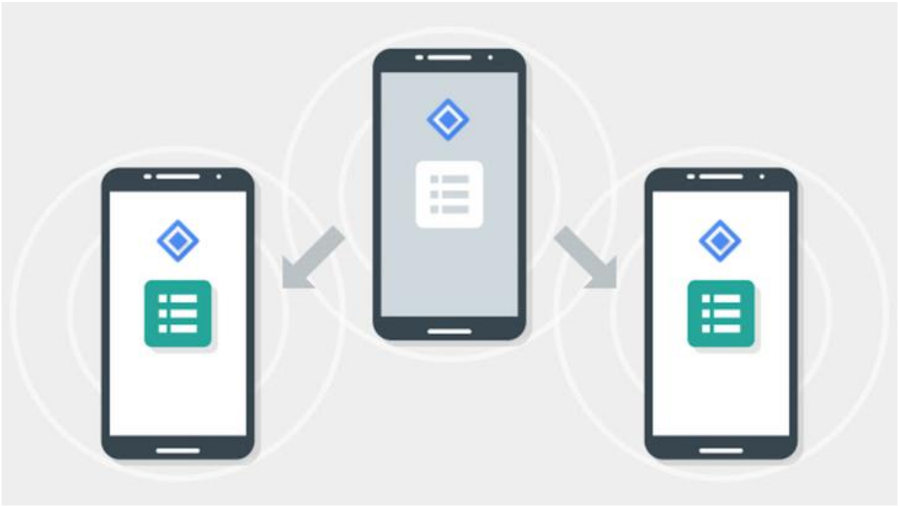
SMS

Google Play Services: Nearby Connections API

https://developers.google.com/nearby/connections/overview

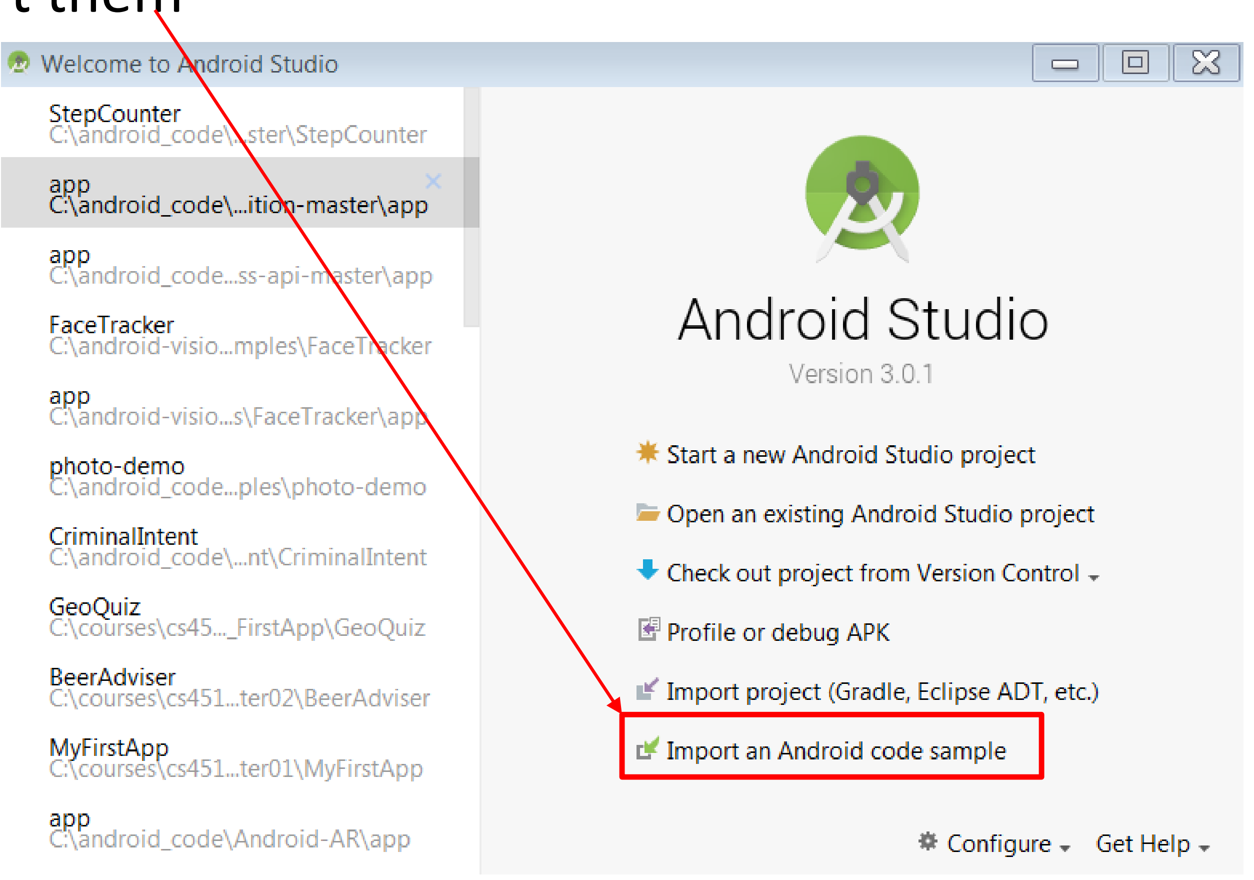
- Peer-to-peer networking API, allows devices communicate over a LAN
- Allows one device to serve as host, advertise
- Other devices can discover host, connect, disconnect
- **Use case:** Multiplayer gaming, shared virtual whiteboard
- Good tutorial by Paul Trebilcox-Ruiz

https://code.tutsplus.com/tutorials/google-play-services-using-the-nearby-connections-api--cms-24534?_ga=2.245472388.1231785259.1517367257-742912955.1516999489



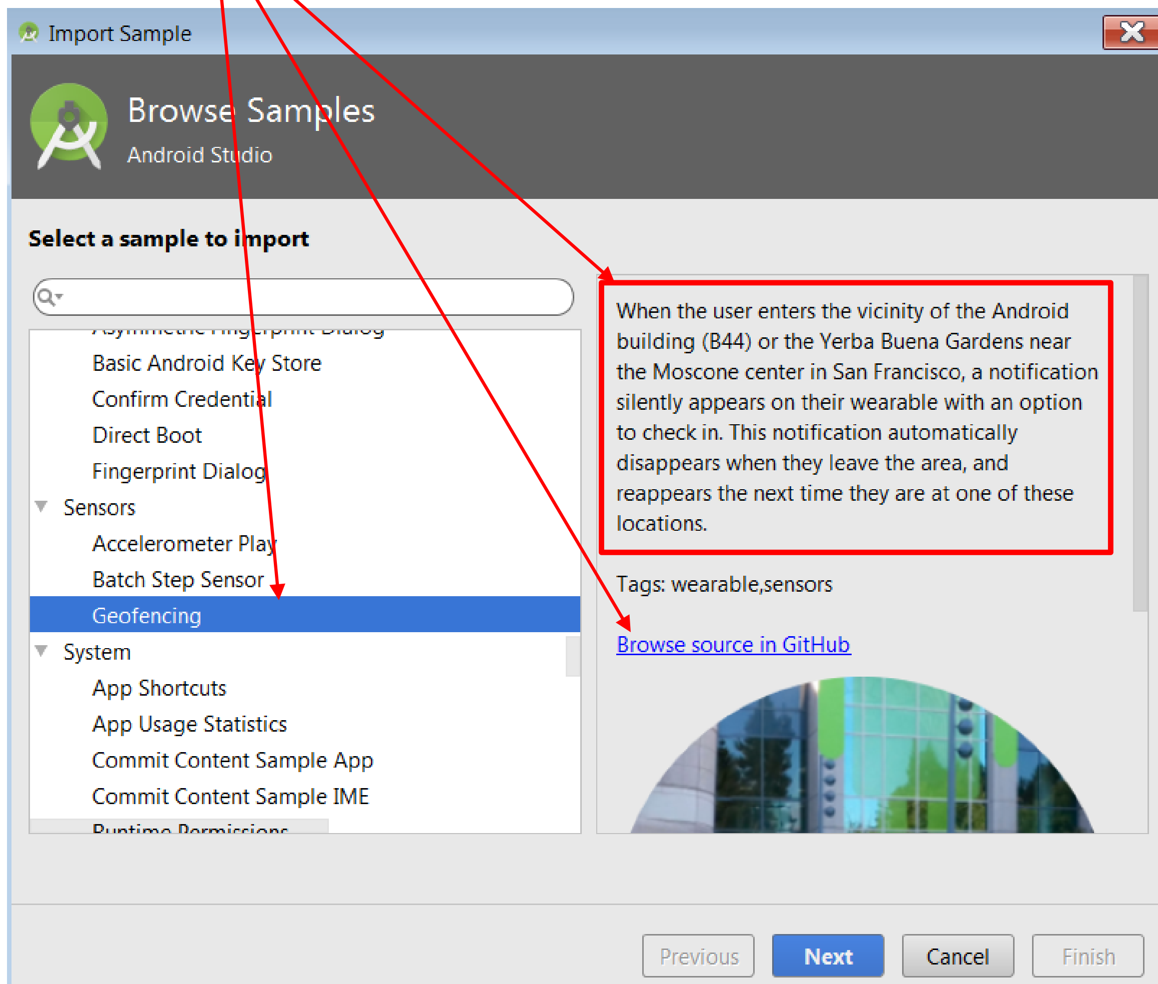
Google Android Samples

- Android Studio comes with many sample programs
- Just need to import them



Google Android Samples

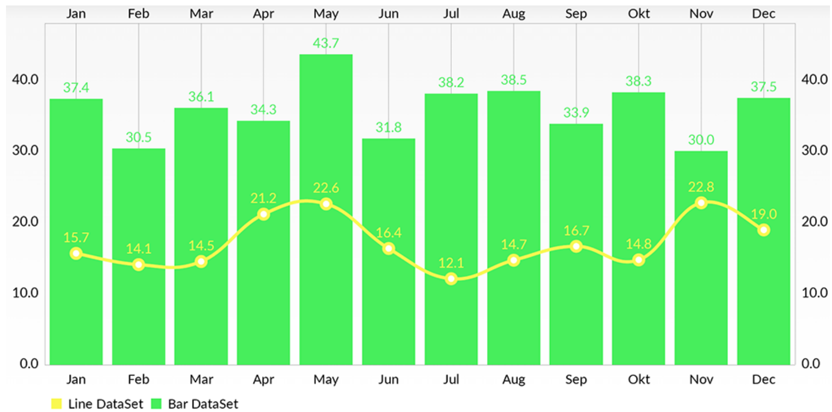
- Can click on any sample, read overview
- Source code available on github



Other 3rd Party Stuff

http://web.cs.wpi.edu/~emmanuel/courses/ubicomp_projects_links.html
https://developer.qualcomm.com/software/trepn-power-profiler

- **MPAndroid:** Add charts to your app

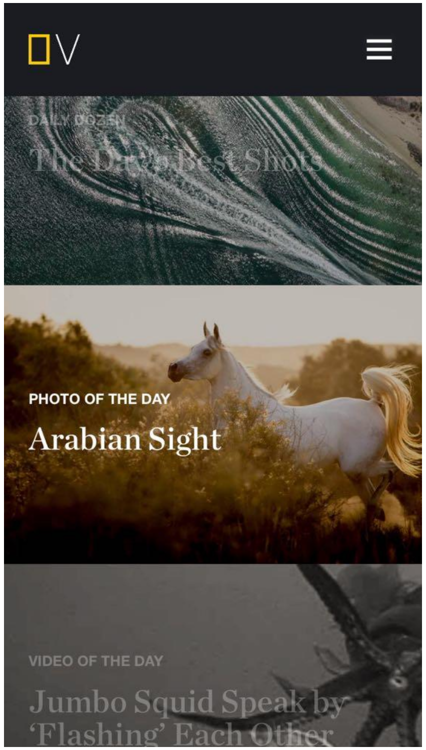


- **Trepn:** Profile power usage and utilization of your app (CPU, GPU, WiFi, etc)
  - By Qualcomm

Other 3rd Party Stuff

http://web.cs.wpi.edu/~emmanuel/courses/ubicomp_projects_links.html

- **Programmable Web APIs:** 3rd party web content (e.g RESTful APIs) you can pull into your app with few lines of code
  - **Weather:** Weather channel, yahoo weather
  - **Shared interests:** Pinterest
  - **Events:** Evently, Eventful, Events.com
  - **Photos:** flickr, Tumblr
  - **Videos:** Youtube
  - **Traffic info:** Mapquest traffic, Yahoo traffic
- **E.g. National Geographic:** picture of the day



Final Project Proposal

Final Project Proposal

- While finishing up project 3, also brainstorm on final project
- Mon Feb 12, all groups 5-min pitch mobile/ubicomp app, solves WPI problem or Machine learning
- Proposals should include:
  1. **Problem you intend to work on**
     - Solve WPI/societal problem (e.g. walking safe at night)
     - Encouraged to use mobile/ubicomp components. See difficulty table
     - 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. **Related Work:** What prior solutions have been proposed for this problem (apps but also academic papers)

Final Project Proposal

4. **Summary of envisioned mobile app (?) solution**
   - E.g. Mobile app automatically texts users friends when they get home at night

5. **Implementation plan:**
   - Mobile/ubiquitous computing components (high level) to be used
   - Project Timeline

6. **Evaluation plan**
   - User studies, performance analysis, etc

- Can bounce ideas of me (email, or in person)
- Can change idea any time

Rubric: Grading Considerations

- **Problem (10/100)**
  - 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?)
  - 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)
- **Related Work (10/100)**
  - What else as been done to solve this problem previously
- **Proposed Solution (10/100)**
  - How good/clever/interesting 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)

Rubric: Grading Considerations

- **Implementation Plan + Timeline (10/100)**
  - Clear plans to realize your design/methodology
  - Android modules/3rd party software used
  - Software architecture,
  - Preliminary screenshots (or sketches of UI), or study design + timeline
- **Evaluation Plan (10/100)**
  - How will you evaluate your project.
  - E.g. small user studies for apps
  - Machine learning cross validation, etc
- 50 more points allotted for your slides + presentation

Final Project: Proposal Vs Final Submission

Final Project Proposal Vs Final Submission

- Introduction
- Related Work
- Approach/methodology
- Implementation
- **Project timeline**

- Evaluation/Results
- Discussion
- Conclusion
- Future Work

Proposal

Final Talk Slides
Final Paper

Note: No timeline
In final paper

The Rest of the Class

The Rest of this class

- **Part 1: Course and Android Introduction**
  - Introduce mobile computing, ubiquitous Computing, Android,
  - Basics of Android programming, UI, Android Lifecycle
- **Part 2: Mobile and ubicomp Android programming**
  - mobile Android components (location, Google Places, maps, geofencing)
  - Ubicomp Android components (camera, face detection, activity recognition, etc)
- **Part 3: Mobile Computing/Ubicomp Research**
  - Machine learning (classification) in ubicomp
  - Ubicomp research (smartphone sensing examples, human mood detection, etc) using machine learning
  - Mobile computing research (app usage studies, energy consumption, etc)

Smartphone Sensing

Smartphone Sensors

- Typical smartphone sensors today
  - accelerometer, compass, GPS, microphone, camera, proximity
- Use machine learning to classify sensor data

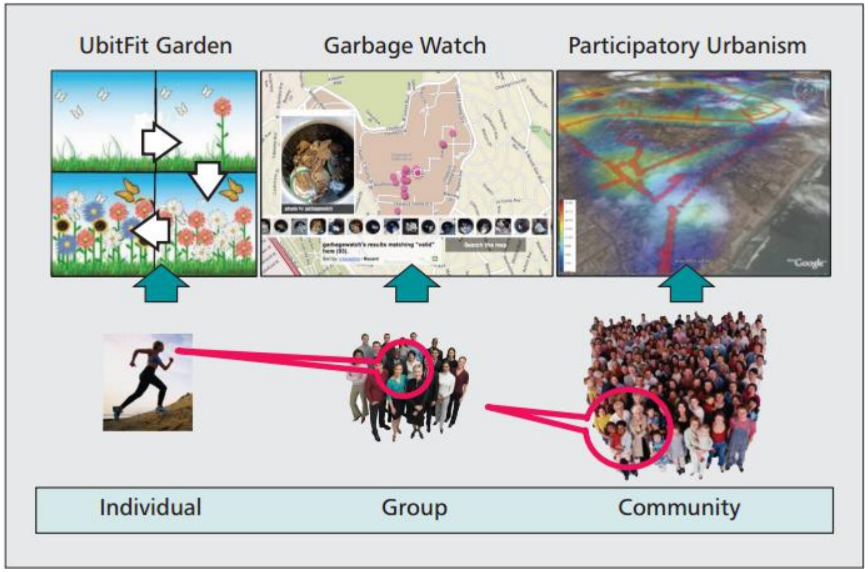
Future sensors?

- Heart rate monitor,
- Activity sensor,
- Pollution sensor,
- etc

Mobile CrowdSensing

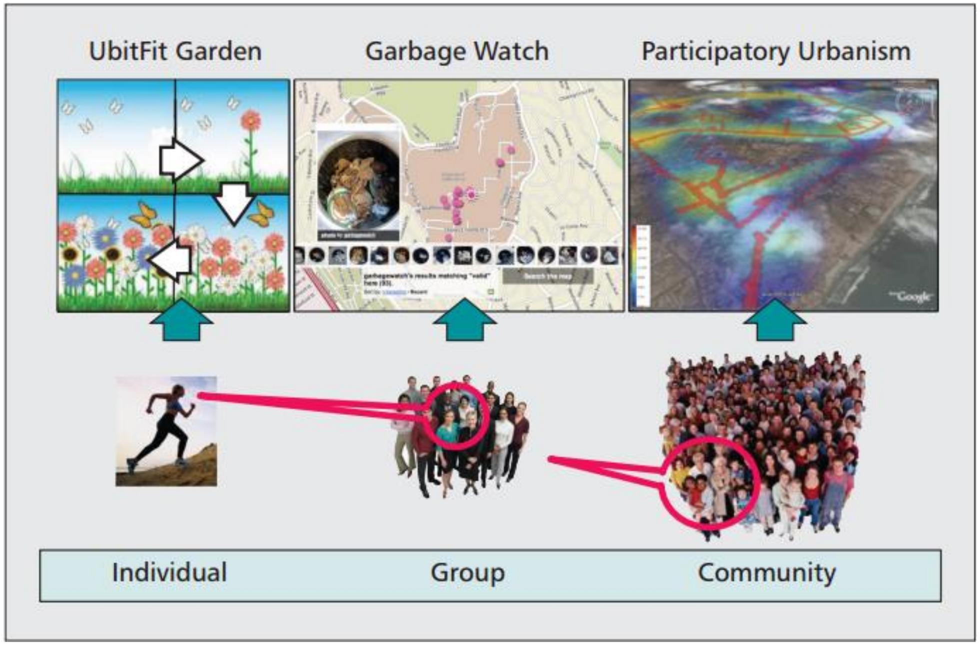
- **Mobile CrowdSensing:** Sense collectively
- **Personal sensing:** phenomena pertain to individual
  - E.g: activity detection and logging for health monitoring

- **Group:** friends, co-workers, neighborhood
  - E.g. GarbageWatch recycling reports, neighborhood surveillance



Mobile CrowdSensing

- **Community sensing (mobile crowdsensing):**
  - Large-scale phenomena monitoring
  - Many people contribute their individual readings
  - **Examples:** Traffic congestion, air pollution, spread of disease, migration pattern of birds, city noise maps



Mobile Crowd Sensing Types

- Many people cooperate, share sensed values
- 2 types:
  1. ***Participatory Sensing:*** User enters sensed values (**active** involvement)
     - E.g. Comparative shopping: Compare price of toothpaste at CVS vs Walmart
  2. ***Opportunistic Sensing:*** Mobile device automatically senses values (**passive** involvement)
     - E.g. Waze crowdsourced traffic

Sense What?

- **Environmental:** pollution, water levels in a creek
- **Transportation:** traffic conditions, road conditions, available parking
- **City infrastructure:** malfunctioning hydrants and traffic signs
- **Social:** photoblogging, share bike route quality, petrol price watch
- **Health and well-being:**
  - Share exercise data (amount, frequency, schedule),
  - share eating habits and pictures of food

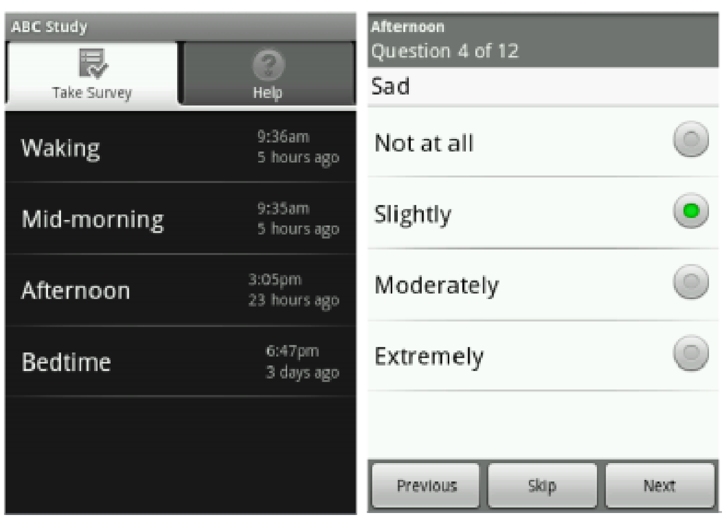
Smartphone Sensing Examples

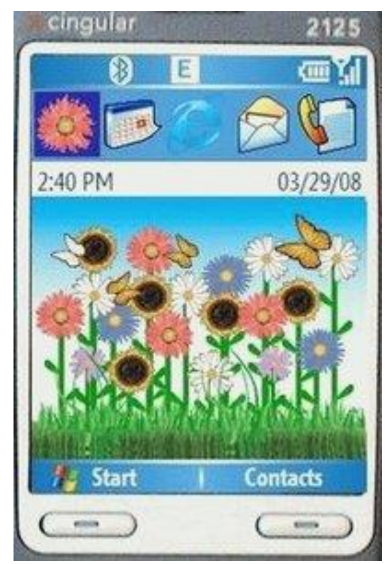
Personal Sensing

- Personal monitoring
- Focusing on user's daily life, physical activity (*Khan et al.*)
- Basically like Fitbit on your phone

Other Examples of Personal Participatory Sensing

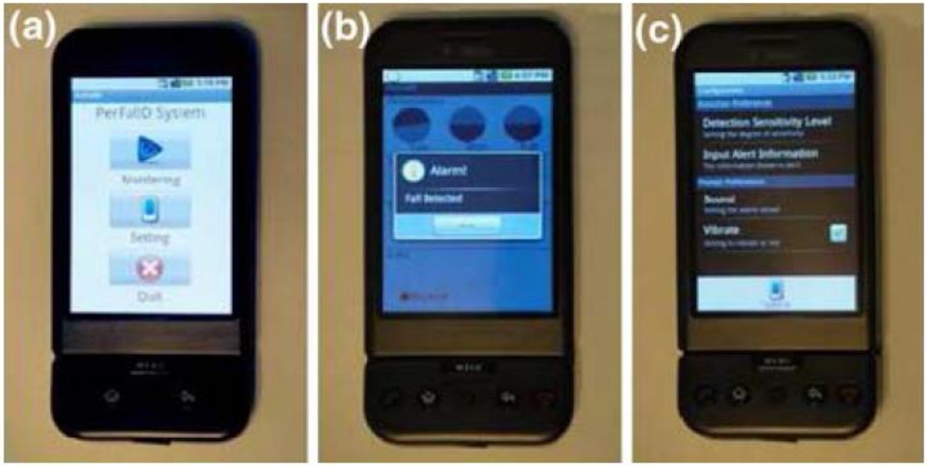
- AndWellness
  - “Personal data collection system”
  - Active user-triggered experiences and surveys
  - Passive recording using sensors
- UbiFit Garden
  - Uses smartphone sensors , real-time statistical modeling, and a personal, mobile display to encourage regular physical activity





Personal Opportunistic Sensing

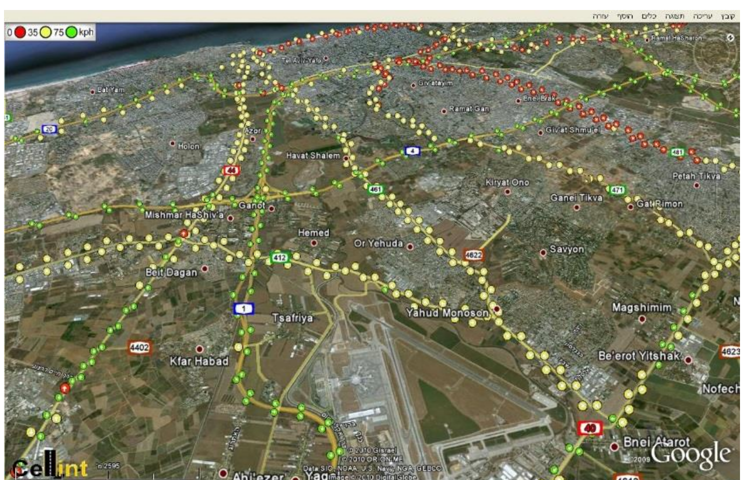
- PerFalld
  - How It Works
    - Detects if someone falls using sensor
    - Starts a timer if it detects that someone fell
    - If individual does not stop timer before it ends, emergency contacts are called

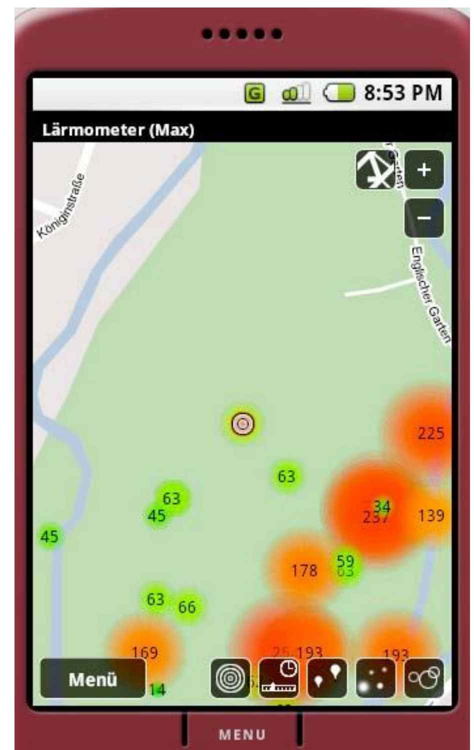


User interfaces in PerFallD: (**a**) bright, large virtual buttons on operating screen (**b**) clear alert window (**c**) simple, non-confusing preference screen

Public Sensing

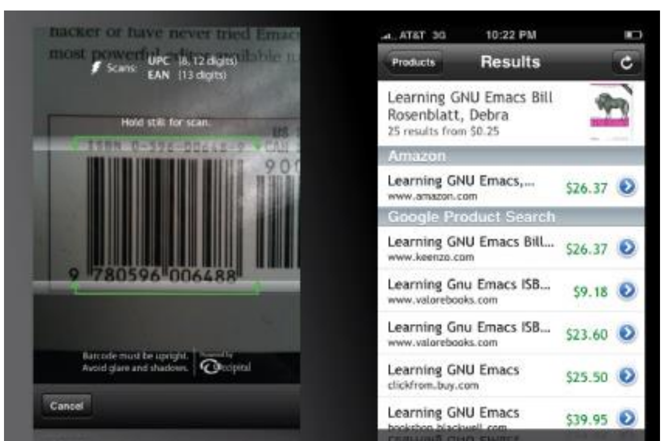
- Data is shared with everyone for public good
- Traffic
- Environmental
  - Noise levels
  - Air pollution

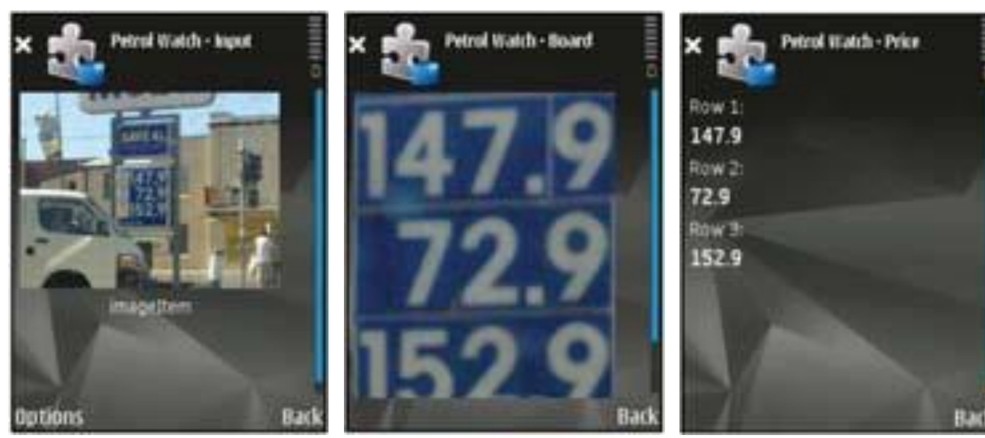




Public Participatory Sensing

- **LiveCompare**
  - User-created database of UPCs and prices
  - GPS and cell tower info used to find nearby stores
- **PetrolWatch**
  - Turns phone into fully automated dash-cam
  - Uses GPS to know when gas station is near



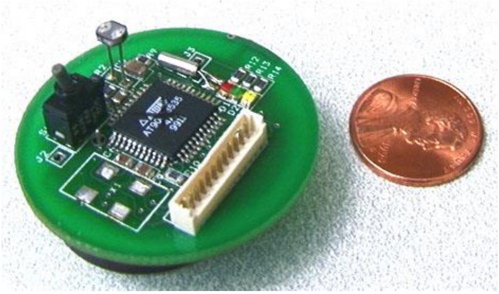


Public Participatory Sensing

- **Pothole Monitor**
  - Combines GPS and accelerometer
- **Party Thermometer**
  - Asks you questions about parties
  - Detects parties through GPS and microphone

Smartphone Sensing vs Dedicated Sensors

VS



Sensing with Smartphones vs Dedicated Sensors

- **More resources:** Smartphones have much more processing and communication power

- **Easy deployment:** Millions of smartphones already owned by people
  - Instead of installing sensors in road, we detect traffic congestion using smartphones carried by drivers

- **Time-varying data:** population of mobile devices, type of sensor data, accuracy changes often due to user mobility and differences between smartphones

Sensing with Smartphones vs Dedicated Sensors

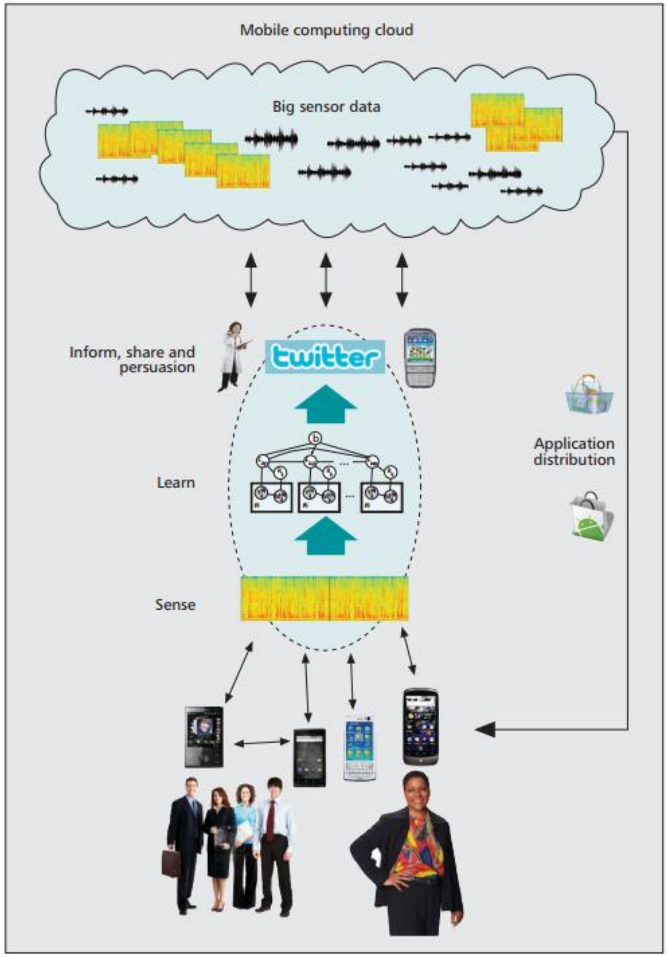
- **Reuse of few general-purpose sensors:** While sensor networks use dedicated sensors, smartphones reuse relatively few sensors for wide-range of applications
  - E.g. Accelerometers used in transportation mode identification, pothole detection, human activity pattern recognition, etc

- **Human involvement:** humans who carry smartphones can be involved in data collection (e.g. taking pictures)
  - Human in the loop can collect complex data
  - Incentives must be given to humans

Smartphone Sensing Architecture

Smartphone Sensing Architecture

- Paradigm proposed by Lane *et al*
- **Sense:** Phones collect sensor data
- **Learn:** Information is extracted from sensor data by applying **machine learning and data mining** techniques
- **Inform, share and persuasion:** inform user of results, share with group/community or persuade them to change their behavior
  - **Inform:** Notify users of accidents (Waze)
  - **Share:** Notify friends of fitness goals (MyFitnessPal)
  - **Persuasion:** avoid speed traps (Waze)



References

1. ***A Survey of Mobile Phone Sensing.*** Nicholas D. Lane, Emiliano Miluzzo, Hong Lu, Daniel Peebles, Tanzeem Choudhury, Andrew T. Campbell, In IEEE Communications Magazine, September 2010
2. ***Mobile Phone Sensing Systems: A Survey,*** Khan, W.; Xiang, Y.; Aalsalem, M.; Arshad, Q.; , Communications Surveys & Tutorials, IEEE , vol.PP, no.99, pp.1-26

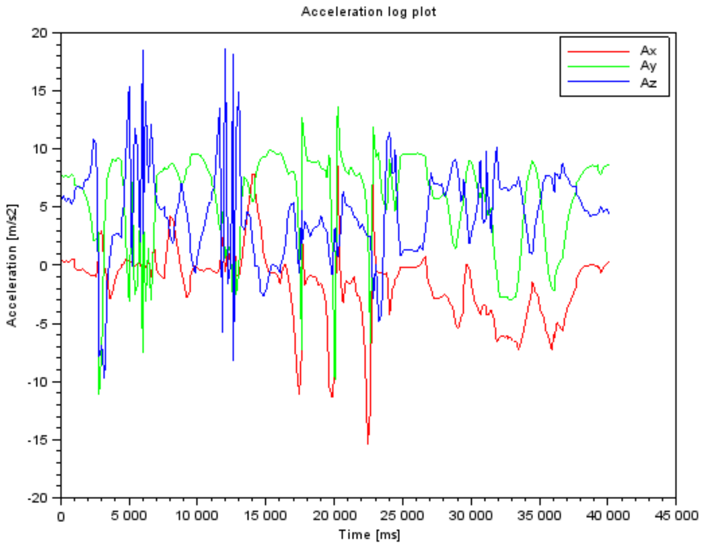
Intuitive Introduction to Machine Learning for Ubiquitous Computing

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 research papers easier to read/understand

Recall: Activity Recognition

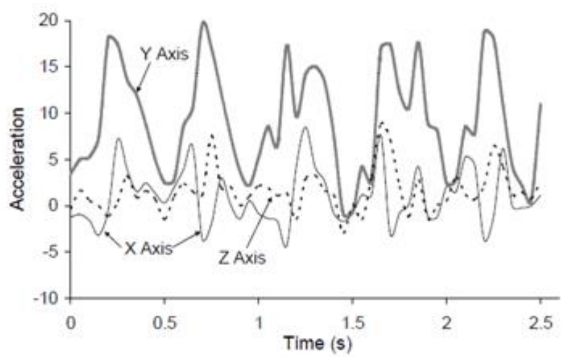
- Want app to detect when user is performing any of the following 6 activities
  - Walking,
  - Jogging,
  - Ascending stairs,
  - Descending stairs,
  - Sitting,
  - Standing



- I will use Activity Recognition as concrete example

Recall: Activity Recognition Overview

Gather Accelerometer data



(a) Walking

Machine Learning Classifier

Classify Accelerometer data

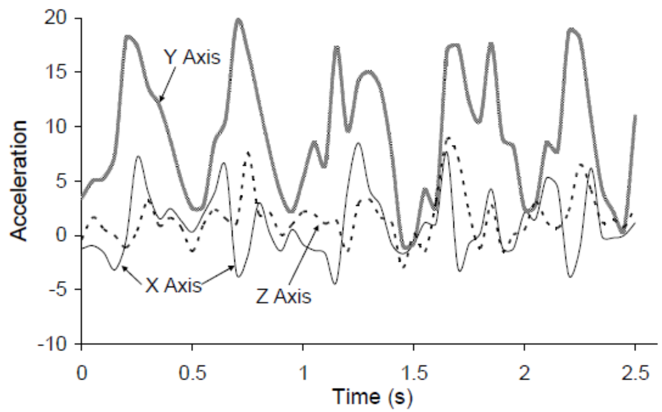
Walking

Running

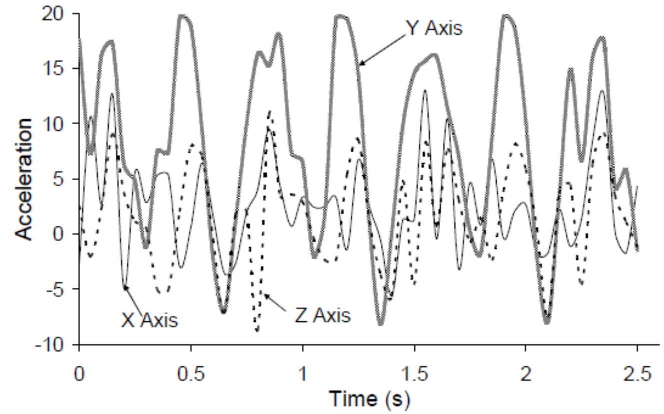
Climbing Stairs

Recall: Example Accelerometer Data for Activities

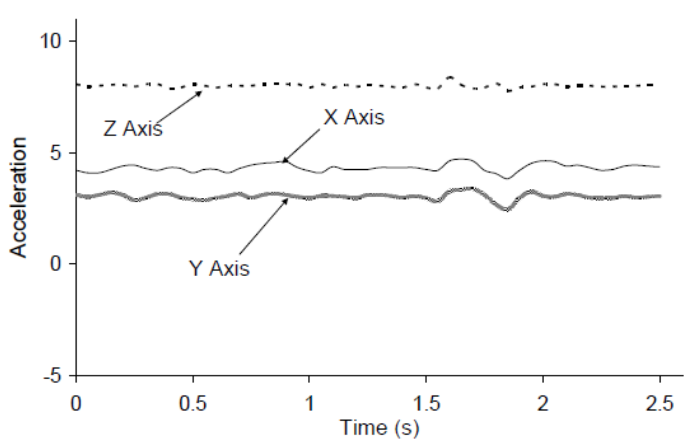
Different user activities generate different accelerometer patterns



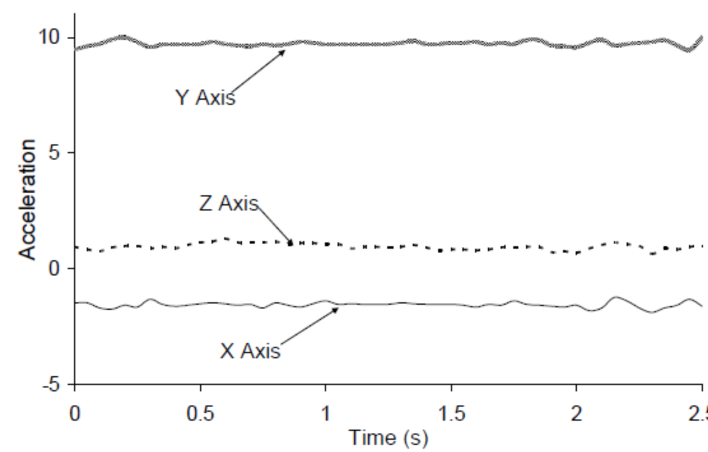
(a) Walking



(b) Jogging



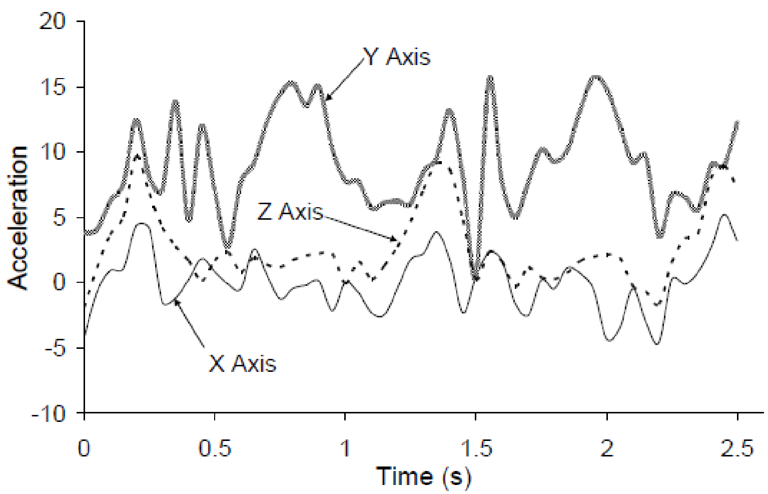
(e) Sitting



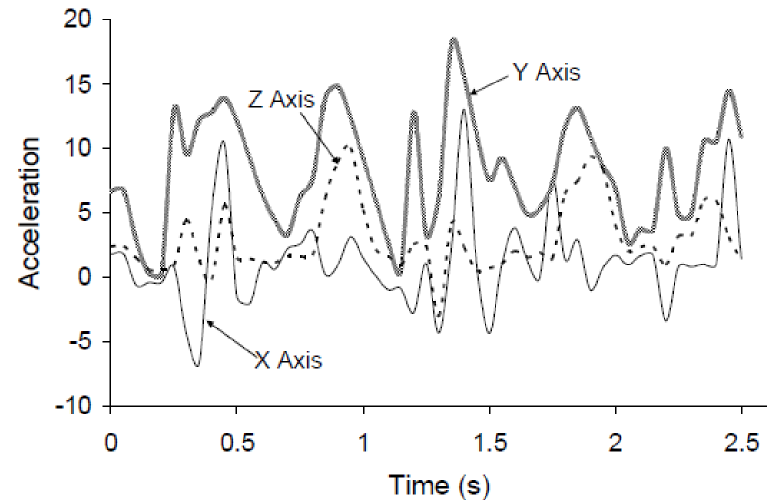
(f) Standing

Recall: Example Accelerometer Data for Activities

Different user activities generate different accelerometer patterns



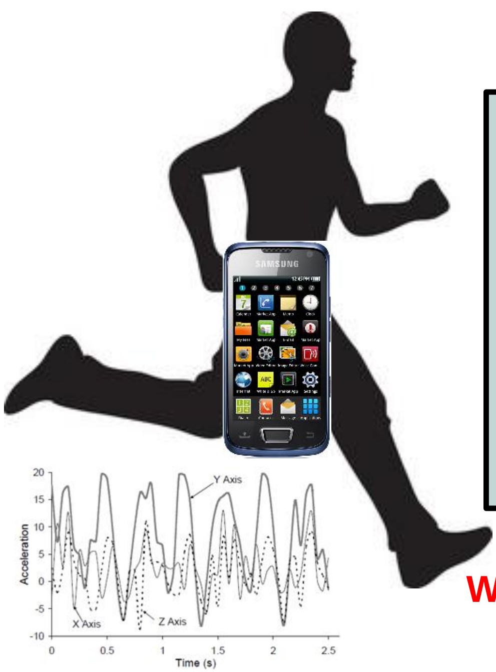
(c) Ascending Stairs

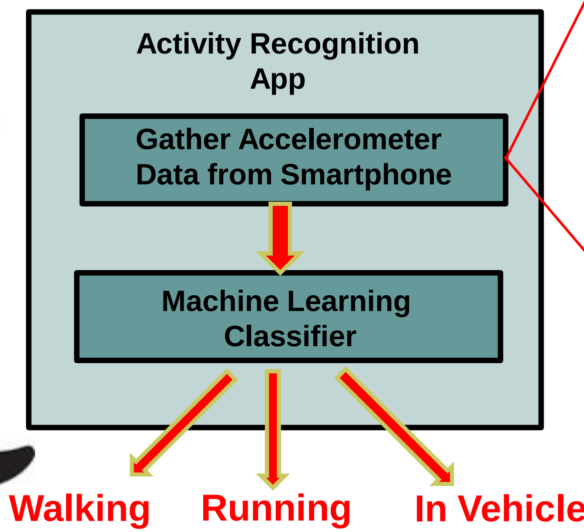


(d) Descending Stairs

Activity Recognition (AR) App: How it works

- 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





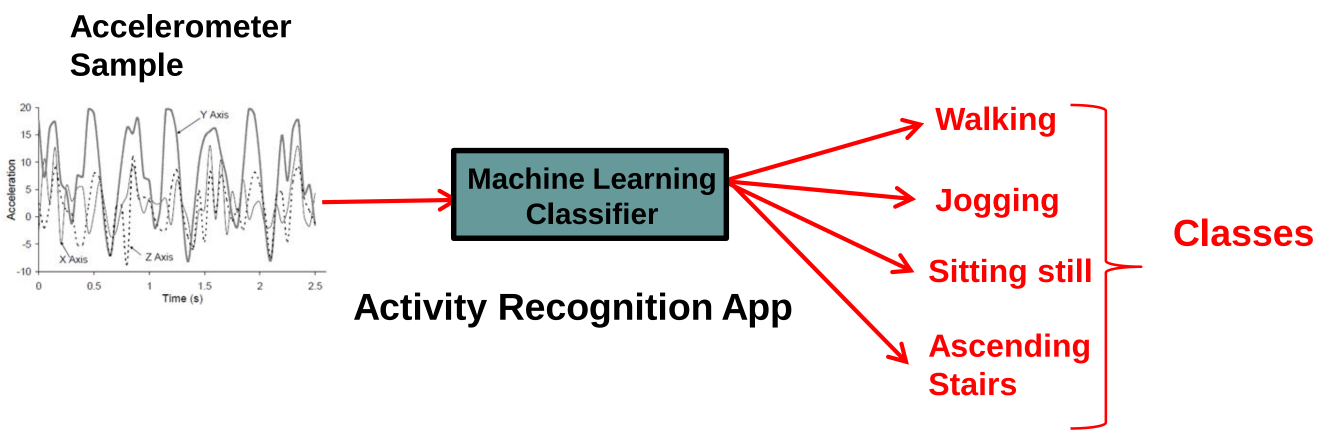

```
msensor = (mSensorManager)
   getSystemService(Context.SENSOR_SERVICE)

....

Public void onSensorChanged(SensorEvent event){

....

}
```

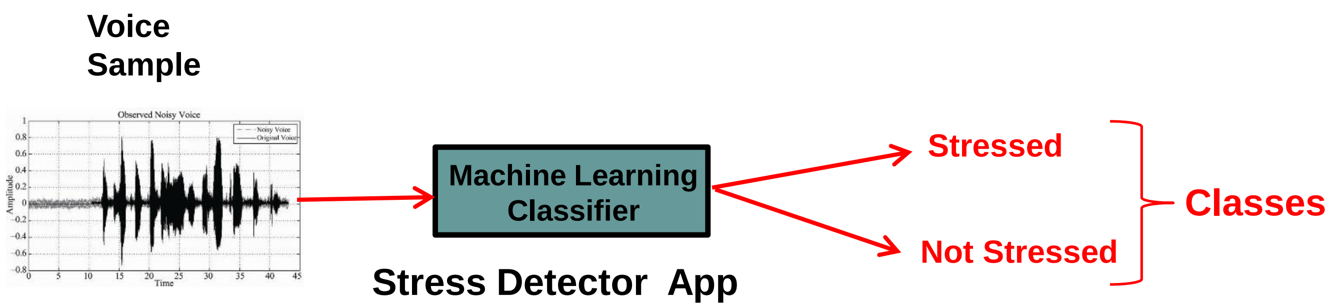
Next: Machine learning Classification

Classification for Ubiquitous Computing

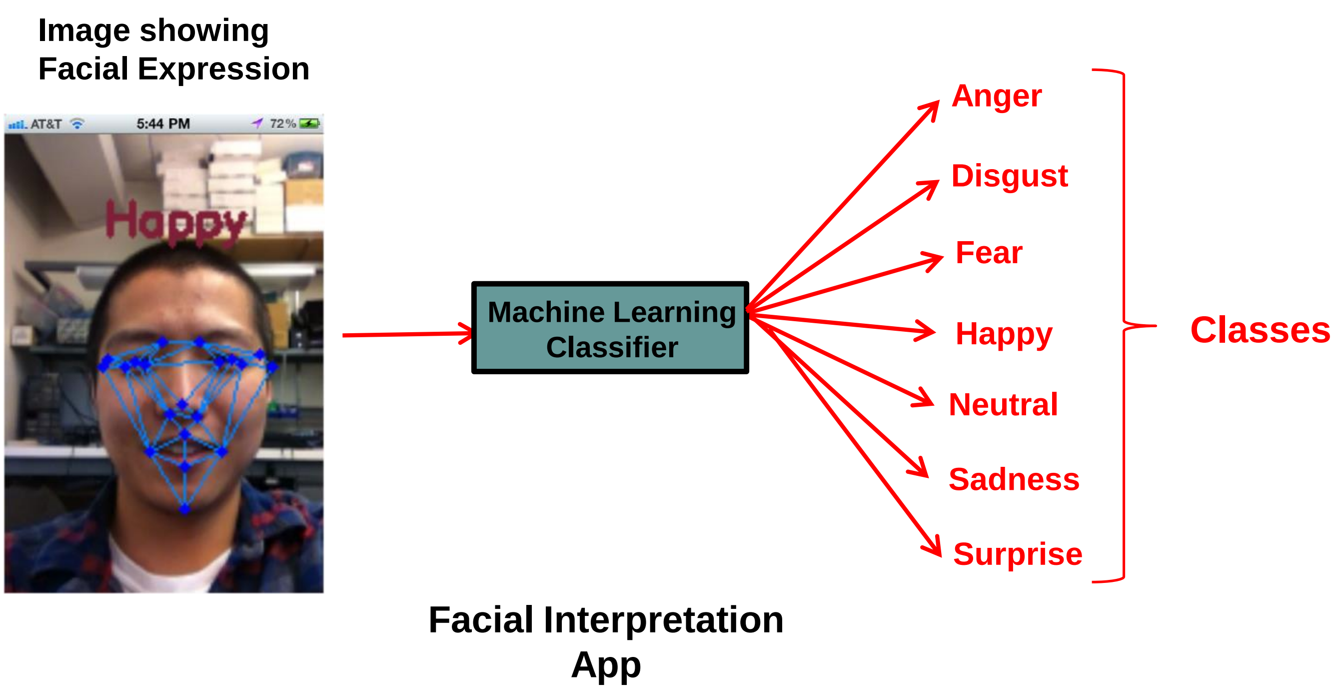
Classification

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





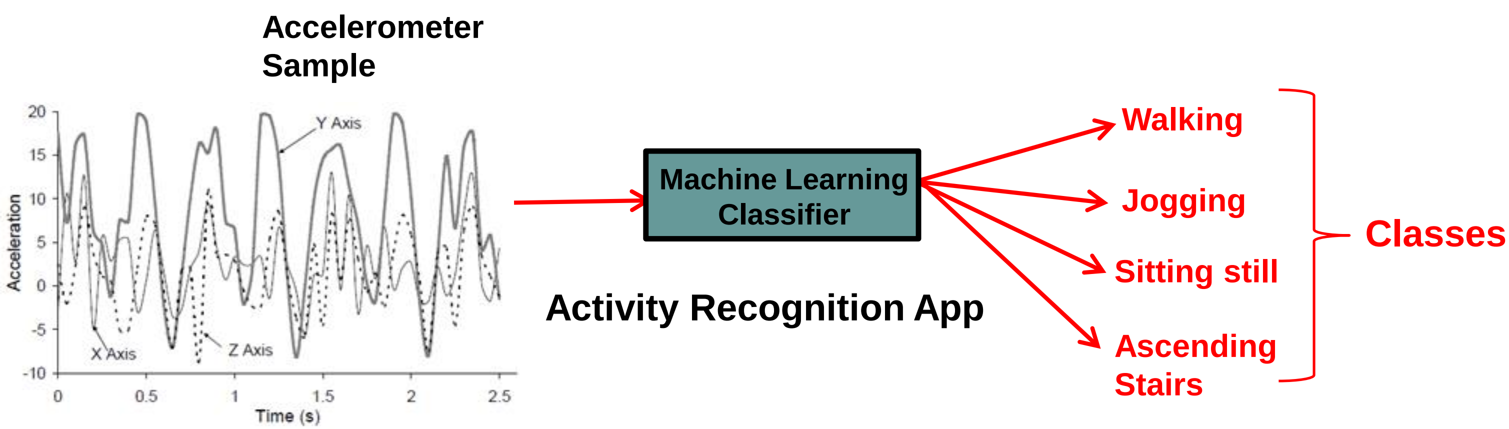
Classification



Classifier

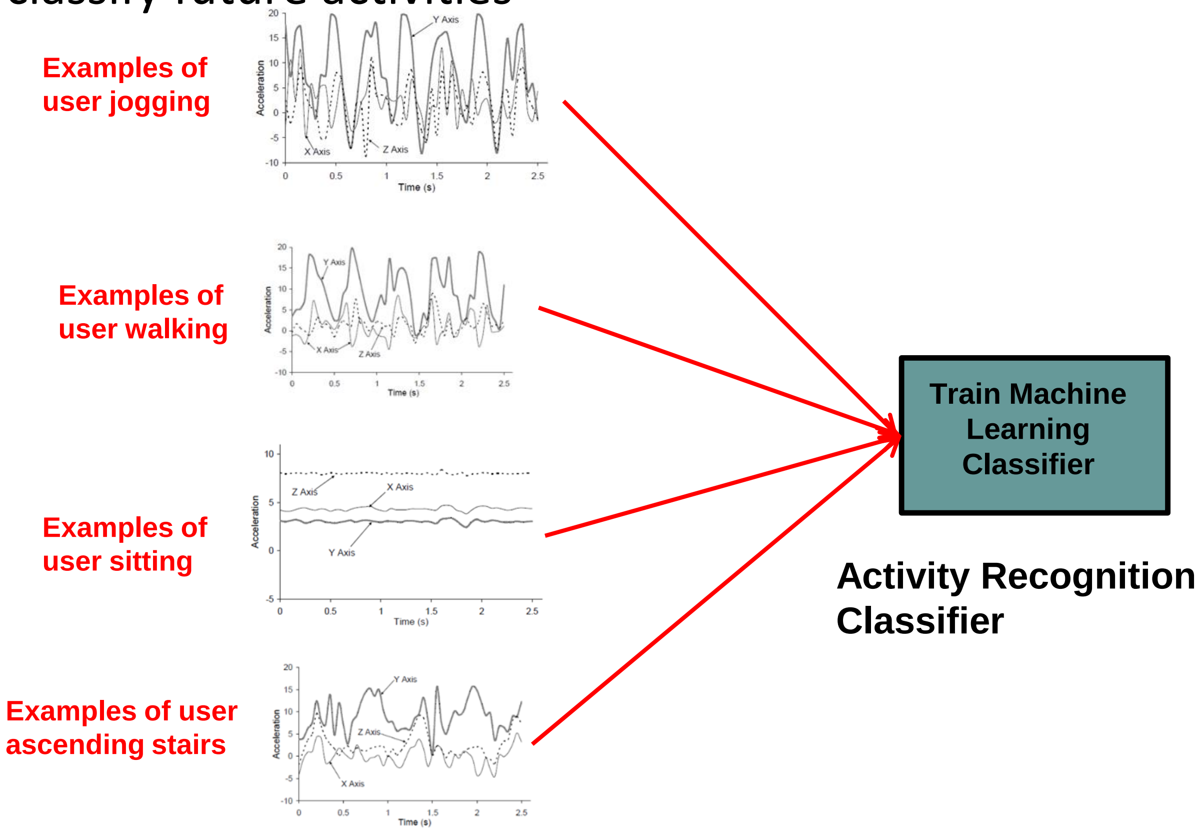
- Analyzes new sample, guesses corresponding class
- Intuitively, can think of classifier as set of rules for classification. E.g.
- Example rules for classifying accelerometer signal in Activity Recognition

```
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:** Given examples of each target class => generate rules to categorize new samples
- **E.g:** Analyze example data from 30 subjects of accelerometer signal for each activity type (walking, jogging, sitting, ascending stairs) => generate rules (classifier) to classify future activities



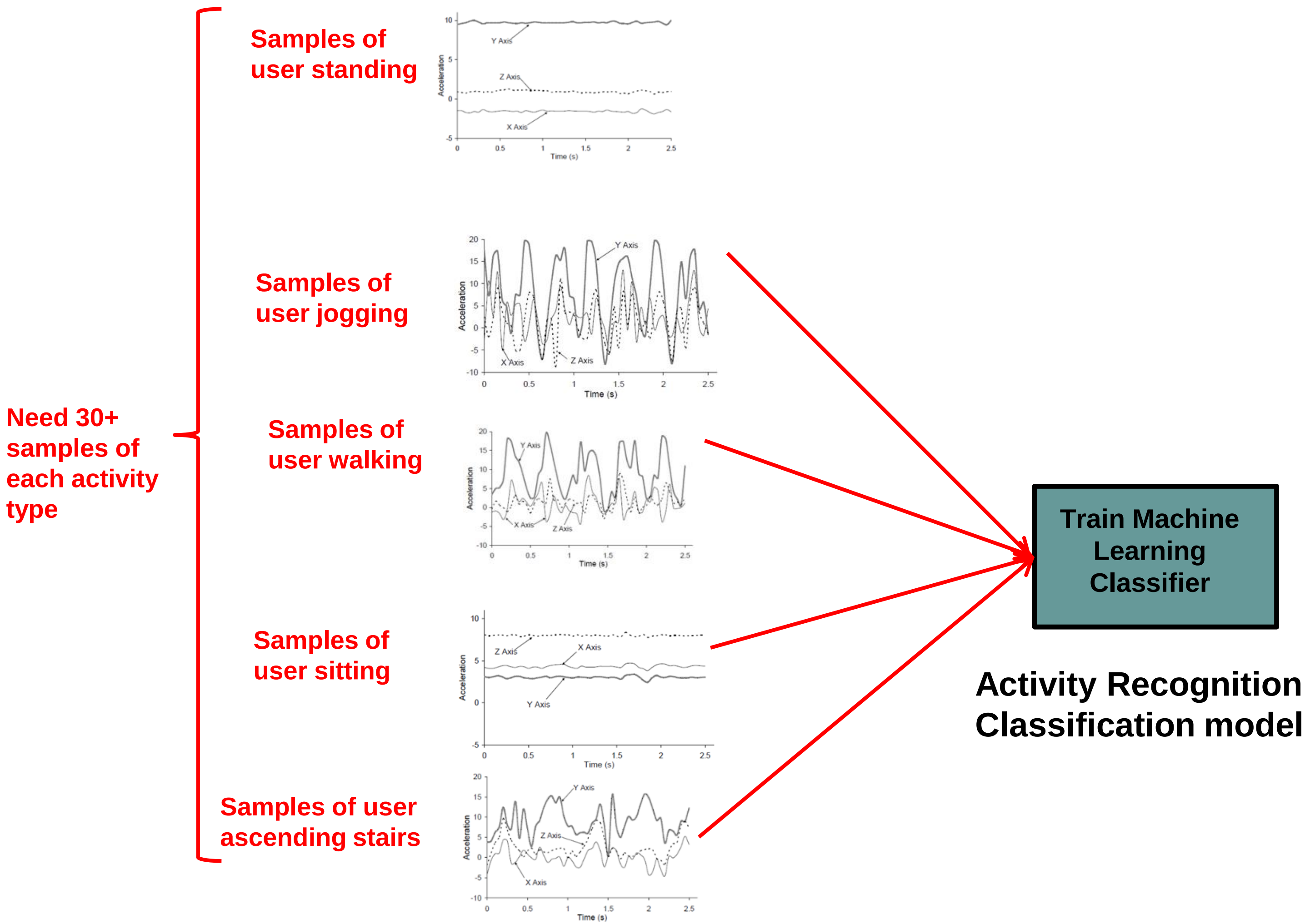
Training a Classifier: Steps

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

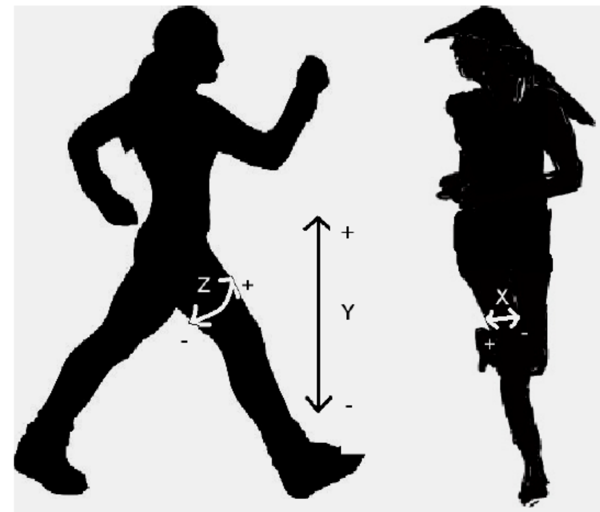
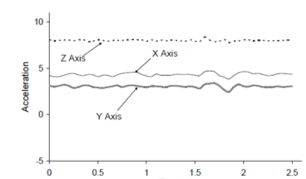
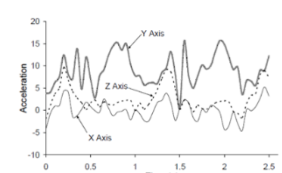


Figure 1: Axes of Motion Relative to User

30+ Samples of user sitting



30+ Samples of user ascending stairs



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

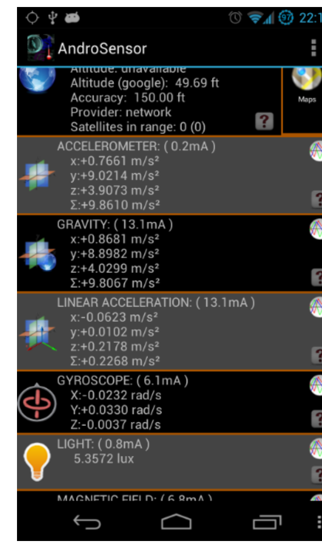
- **Option 1:** Can write sensor program app that gathers accelerometer data while user is doing each of 6 activities (1 at a time)

```
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, MATLAB (or other Machine learning Framework)

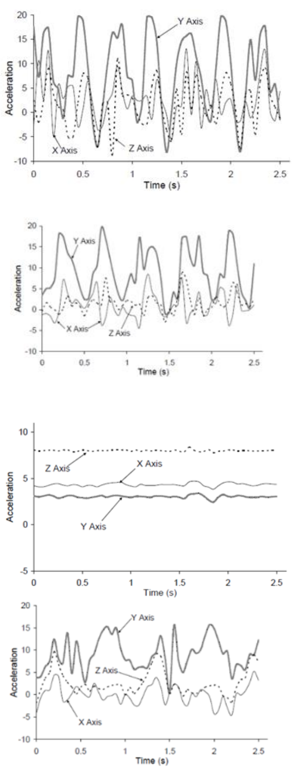
LABELS | **ACCELEROMETER DATA**

Jogging

Walking

Sitting

Ascending stairs

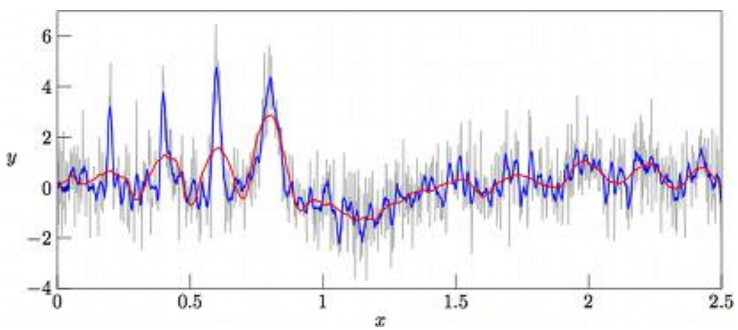


→ **Weka** → **Classifiers**

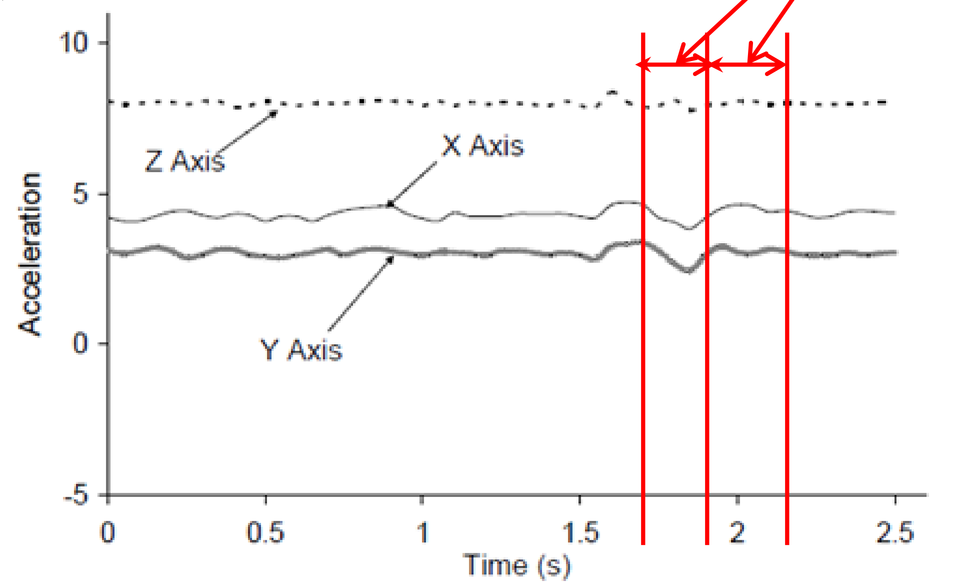
Classifier is trained offline

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

- Pre-processing data (in Weka, or MATLAB) may include segmentation, smoothing, etc
  - **Smoothing:** Replace batches of values with their moving average
    - Reduce choppiness
  - **Segment:** Divide 60 seconds of raw time-series data divided into chunks(e.g. 5 seconds)
    - Note: 5 seconds of accelerometer data could be 100s of readings



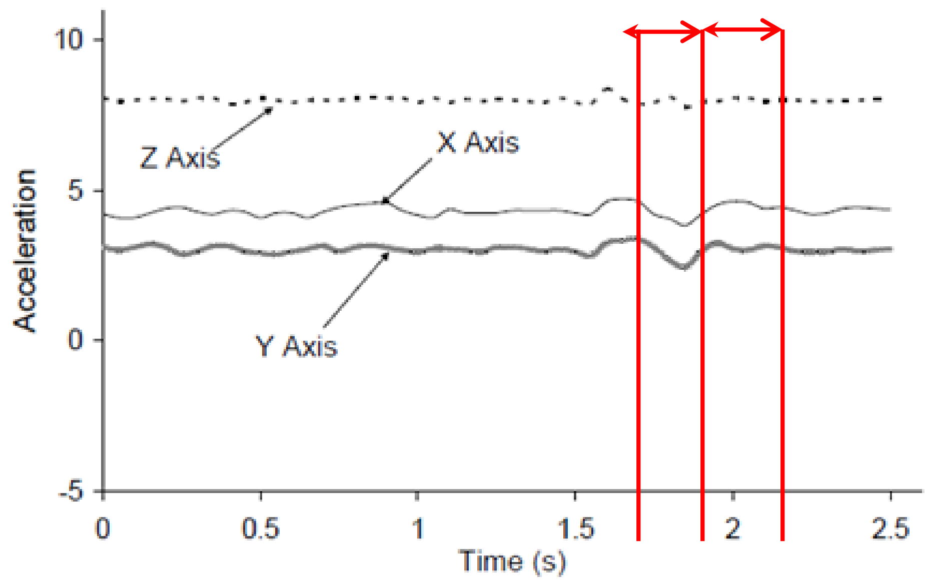
Smoothing



(e) Sitting

Step 4: Compute (Extract) Features

- For each 5-second segment (batch of accelerometer values) compute features (in Weka, MATLAB, etc)
- **Features:** Functions computed on accelerometer data, captures important accelerometer characteristics
- **Examples:** min-max of values, largest magnitude within segment, standard deviation

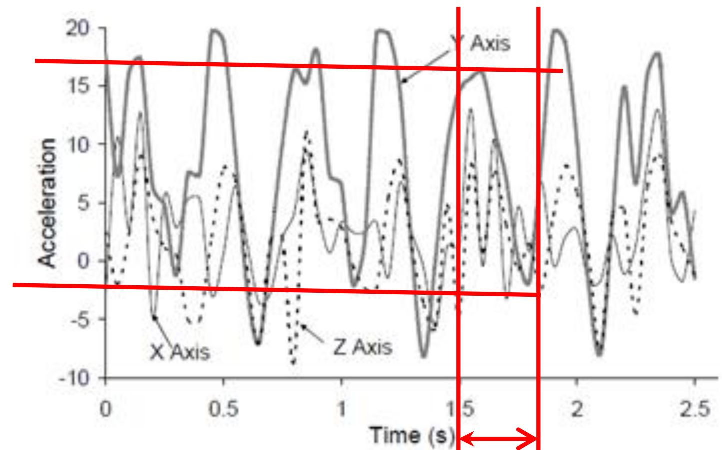


(e) Sitting

Step 4: Compute (Extract) Features

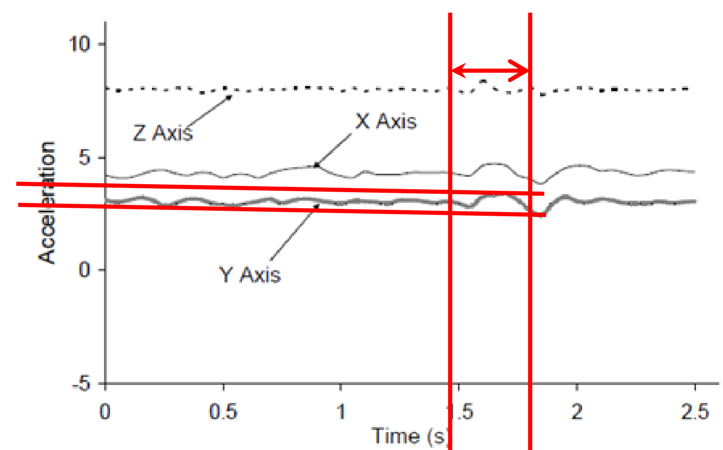
- **Important:** Ideally, values of features calculated should be different for, distinguish each activity type
- **E.g:** Min-max range feature

Large min-max for jogging



(b) Jogging

Small min-max for sitting



(e) Sitting

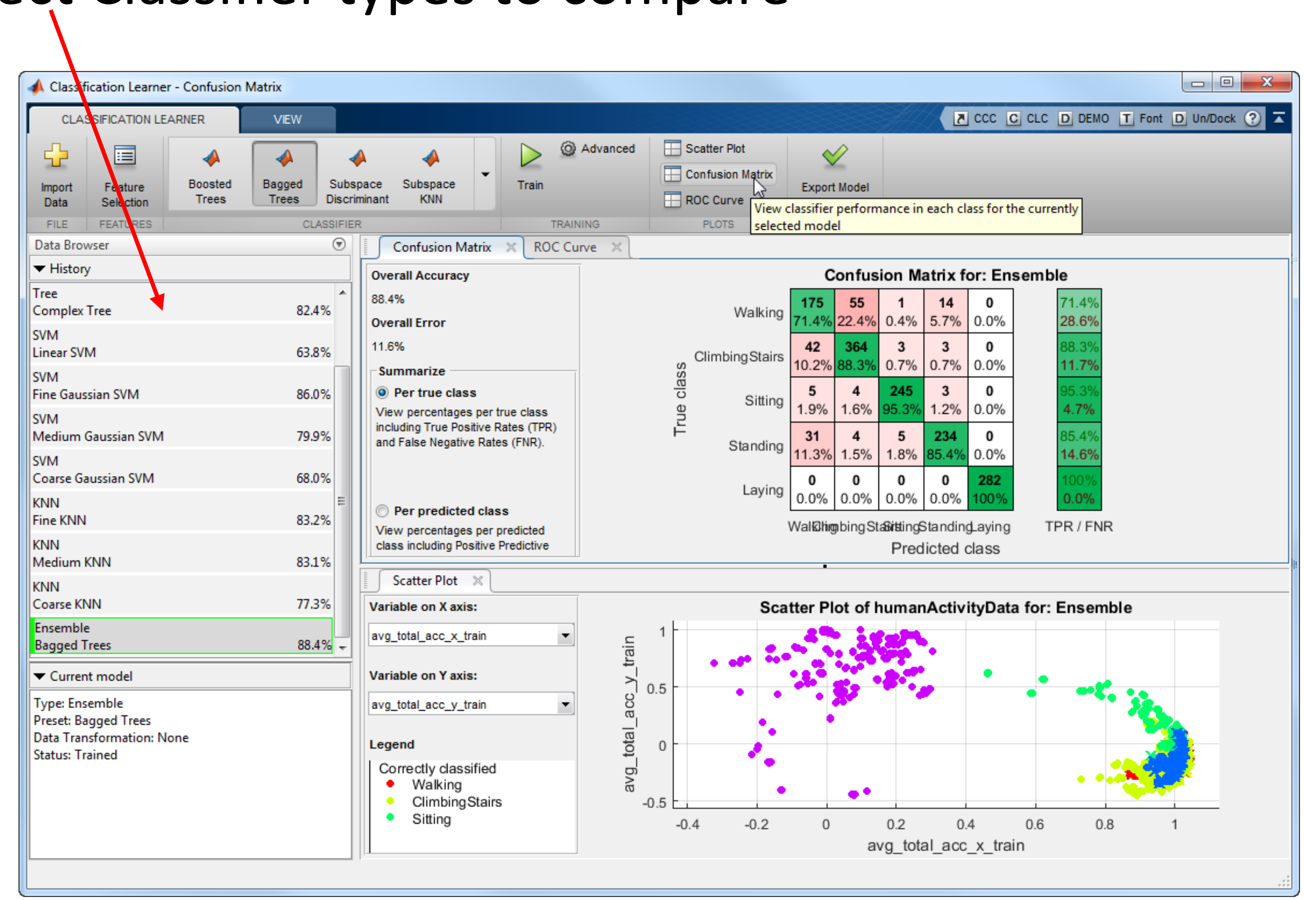
Step 4: Compute (Extract) Features

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.

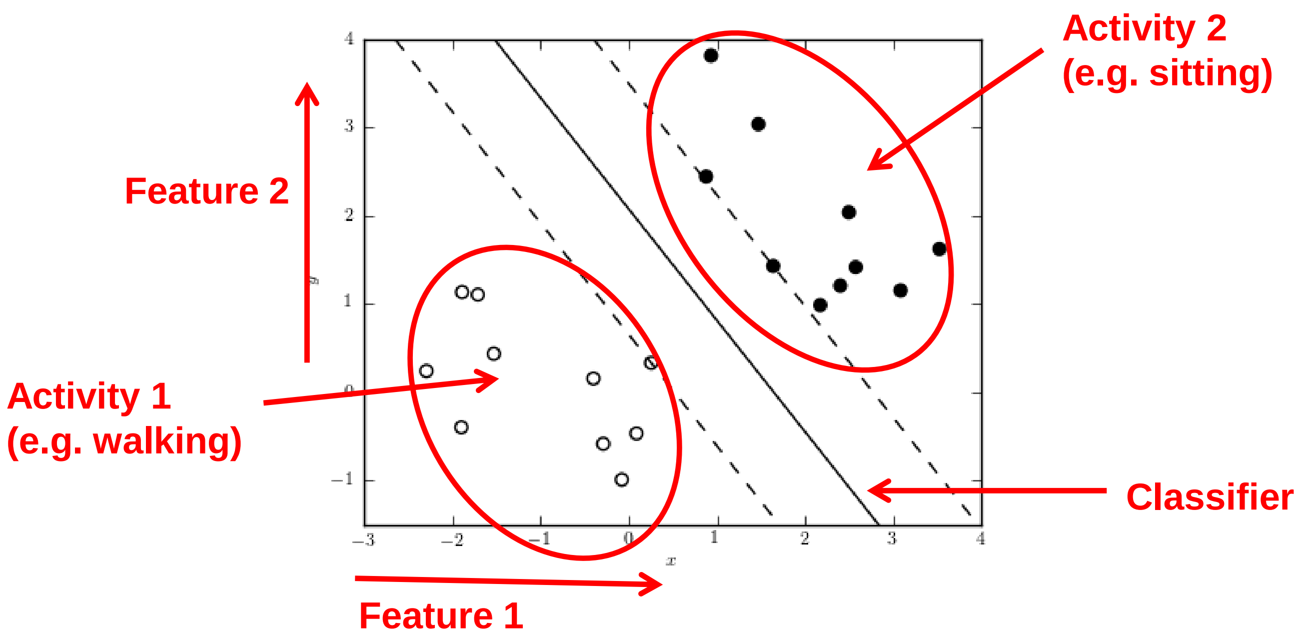
Step 5: Train Classifier
MATLAB Classification Learner App

- Import accelerometer data into MATLAB
  - Can do feature extraction in MATLAB
- Select Classifier types to compare



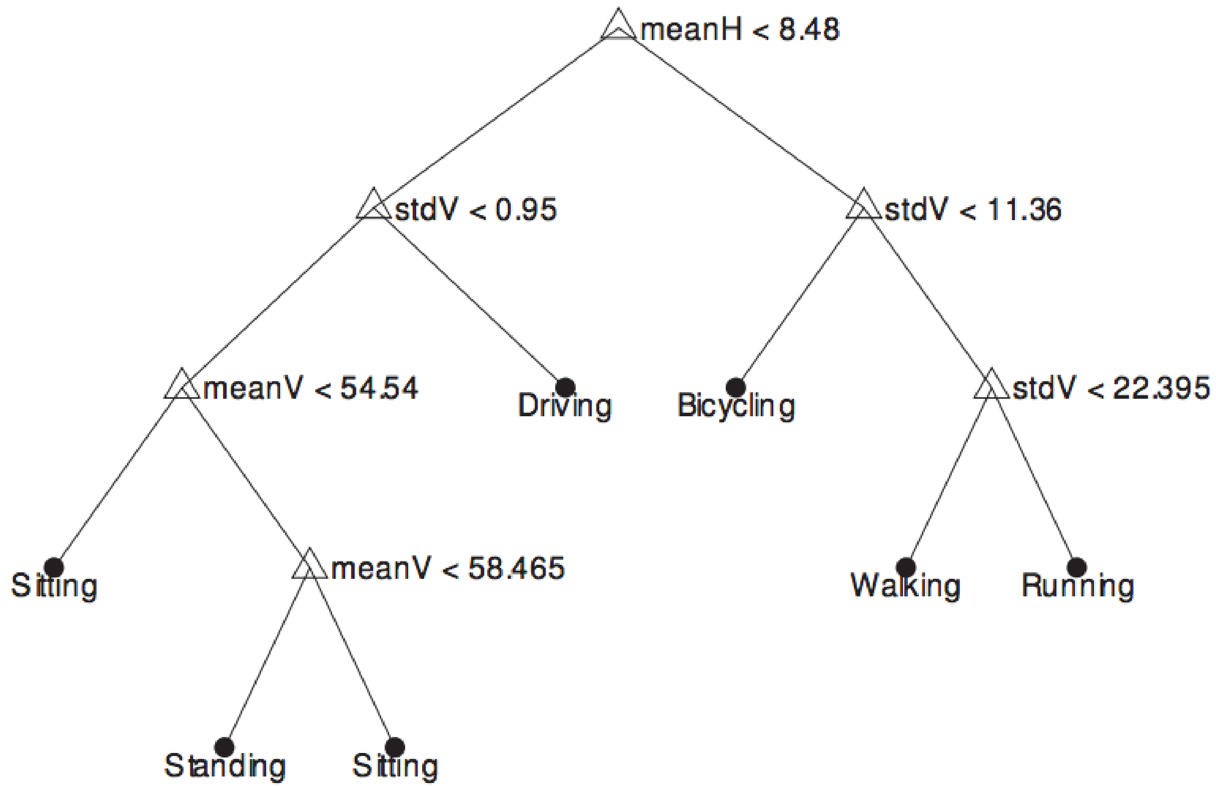
Step 5: Train classifier

- Features are just numbers
- Different values for different activities
- **Training classifier:** figures out feature values corresponding to each activity
- Weka, MATLAB already programmed with different classification algorithms (SVM, Naïve Bayes, Random Forest, J48, logistic regression, SMO, etc)
- Try different ones, compare accuracy
- Points in diagram are feature values in multi-dimensional space. 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, MATLAB also reports accuracy of each classifier type

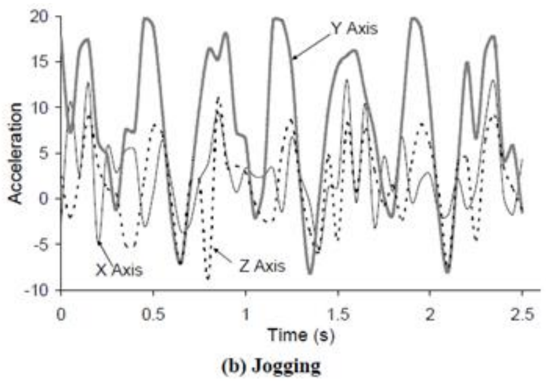
Table 2: Accuracies of Activity Recognition

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

Compare, pick most accurate classification algorithm

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

- Export classification model (most accurate classifier type + data threshold values) 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



New accelerometer Sample in real time

Classifier in Android app

Activity (e.g. Jogging)

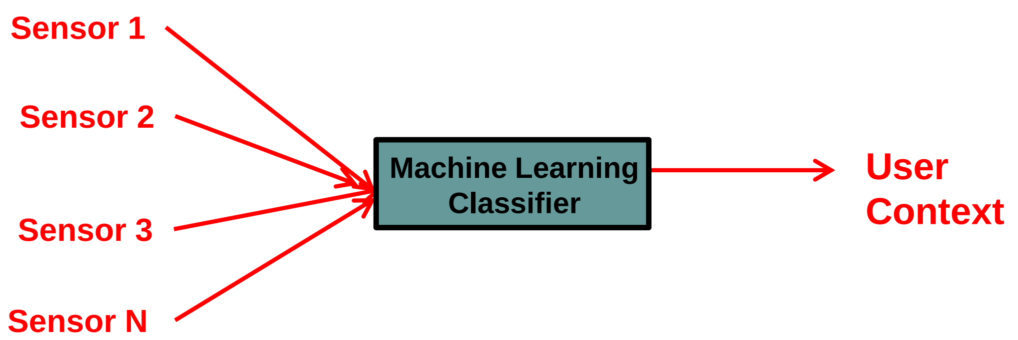
Context Sensing

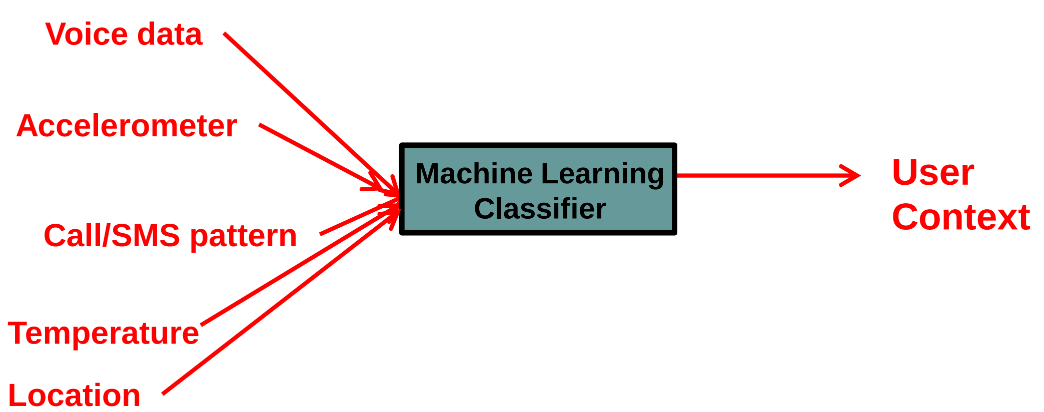
Recall: Ubicomp Senses User's Context

- 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 learn **user context** that occur to various outcomes (e.g. user's emotion)
- More later

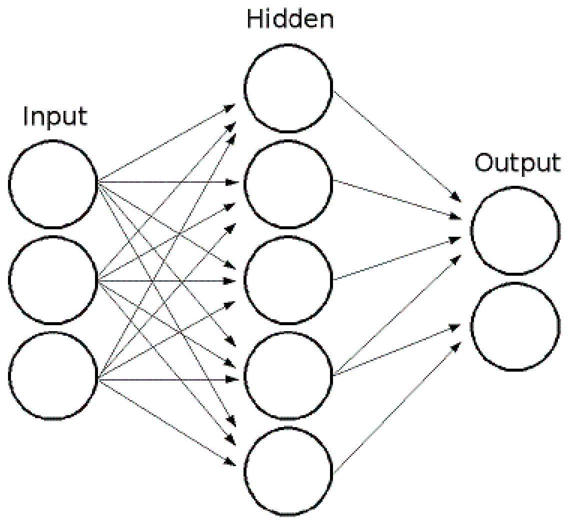




Deep Learning

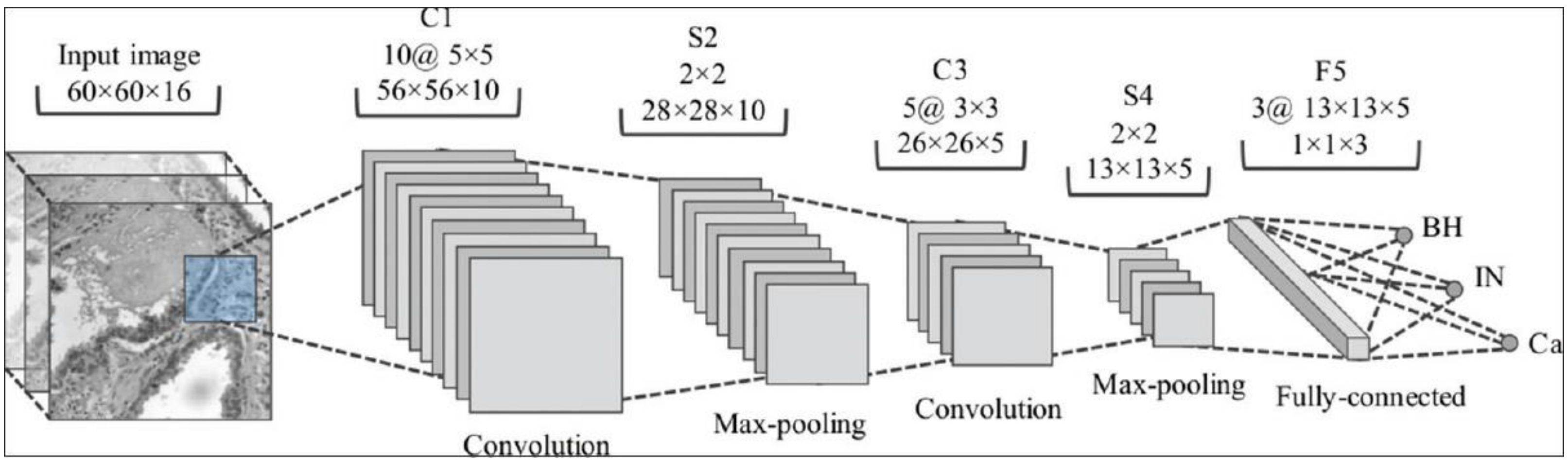
Deep Learning

- Network of nodes, connectivity weights learned from data
- Learns best weights to classify inputs (x) into outputs y
- Can think about it as curve fitting
- Generally more accurate if more data is available
- Requires lots of computational power to train



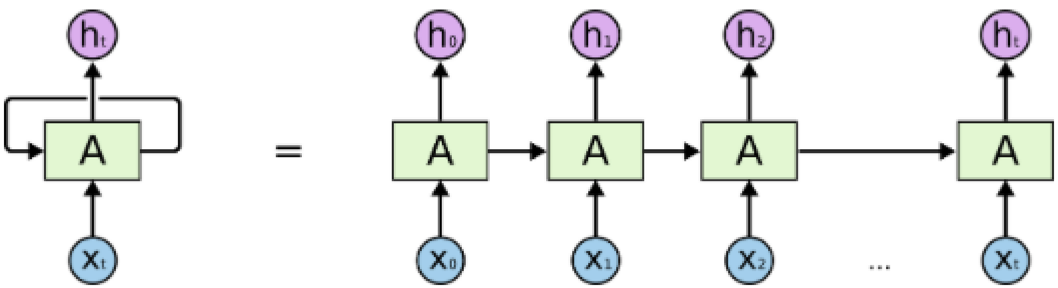
Convolutional Neural Networks (CNNs)

- Different types of neural networks good for different things
- Convolutional Neural Networks good for classifying images
- E.g. Is there a cat in an input picture?



Recurrent Neural Networks (RNNs)

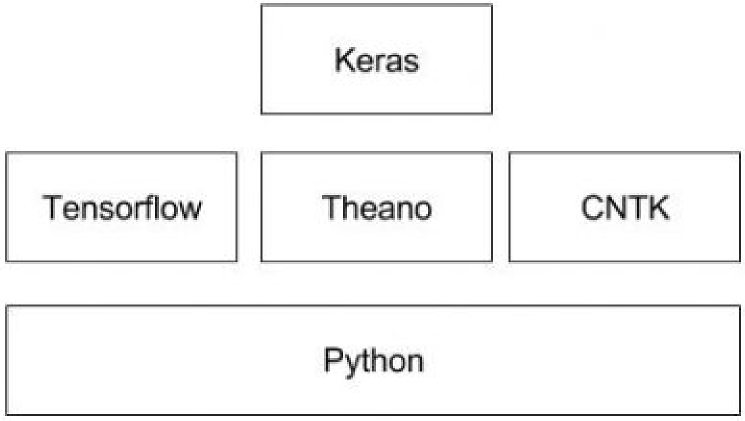
- Good at classifying sequential data
- E.g. Speech translation
- E.g. translate german sentence to English



Programming/Mobile Support for Neural Networks

https://developer.android.com/ndk/guides/neuralnetworks/index.html

- Many python libraries
- Enable training neural networks in a few lines of code
  - Keras
  - PyTorch
  - ScikitLearn
- Training neural networks on Smartphone still tough
- New in Android 8.1: Android Neural Networks API allows inference (test) of pre-trained neural networks on smartphone
- Keras also has some mobile support



References

- Google Mobile Vision API, https://developers.google.com/vision/
- Camera "Taking Photos Simply" Tutorials, http://developer.android.com/training/camera/photobasics.html
- Busy Coder's guide to Android version 6.3
- CS 65/165 slides, Dartmouth College, Spring 2014
- CS 371M slides, U of Texas Austin, Spring 2014