UCLA

180D Motion Classification Project
Final Presentation

Corbett Cappon, Anthony Nguyen, Jincheng Zhou, Zhe Wan

Dec.12, 2013

Design Overview: Objectives Summary

- Problem
  - There is a need for classifying human motion in clinical trials, energy expenditure monitoring, and user activity management
  - Advancements in motion sensor technology make it an increasingly usable tool for classifying human motion
- Objective
  - Accurately and cheaply classify human motions using state of the art motion sensor technology.

Design Overview: Approach Summary

- Detailed Problem Statement
  - Design Constraints:
    - Our motion classification needs to be completed using data from only two motion sensors, measured using bluetooth wireless communication.
  - Challenges to be detailed in System Implementation and Testing sections
- Detailed Approach
  - It's difficult to predict in advance which parameters of our motion sensor will best differentiate the different types of motion. As a result, our general methodology is to methodically measure the human motion in question, and use the sensor fusion toolkit to compare all feature effectiveness, and find the features that work best for classification.

Background

- Background integrated into System Implementation and Testing sections

format_data.m

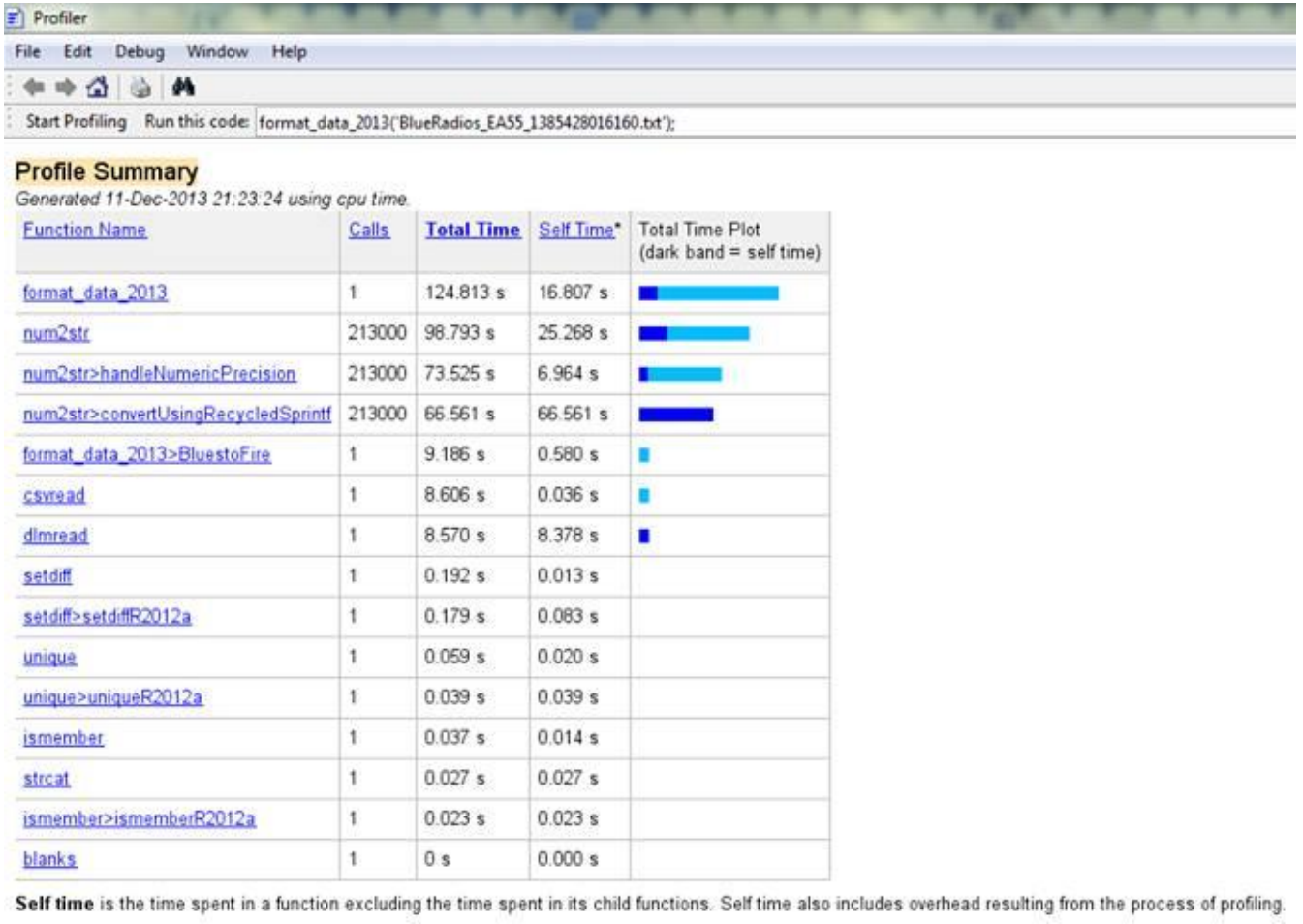
- Provided at beginning of quarter
- Our sensors are newer than the toolbox, so the toolbox cannot understand the data from the sensors

Why make a new data converter?

- Current version was not broken
- Observation – converter took more and more time larger input files
- Why does this happen?

```
>> format_data_2013('BlueRadios_E8AB_1382661824440.txt')
Importing BlueRadios data
BlueRadios data has been converted to Firefly format
Now exporting to file
>>
```

format_data



Profile Summary

Generated 11-Dec-2013 21:23:24 using cpu time.

Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
format_data_2013	1	124.813 s	16.807 s	
num2str	213000	98.793 s	25.268 s	
num2str>handleNumericPrecision	213000	73.525 s	6.964 s	
num2str>convertUsingRecycledSprintf	213000	66.561 s	66.561 s	
format_data_2013>BluestoFire	1	9.186 s	0.580 s	
csvread	1	8.606 s	0.036 s	
dlmread	1	8.570 s	8.378 s	
setdiff	1	0.192 s	0.013 s	
setdiff>setdiffR2012a	1	0.179 s	0.083 s	
unique	1	0.059 s	0.020 s	
unique>uniqueR2012a	1	0.039 s	0.039 s	
ismember	1	0.037 s	0.014 s	
strcat	1	0.027 s	0.027 s	
ismember>ismemberR2012a	1	0.023 s	0.023 s	
blanks	1	0 s	0.000 s	

Self time is the time spent in a function excluding the time spent in its child functions. Self time also includes overhead resulting from the process of profiling.

Algorithmic Time

- Original format_data is constrained by the file read which runs:

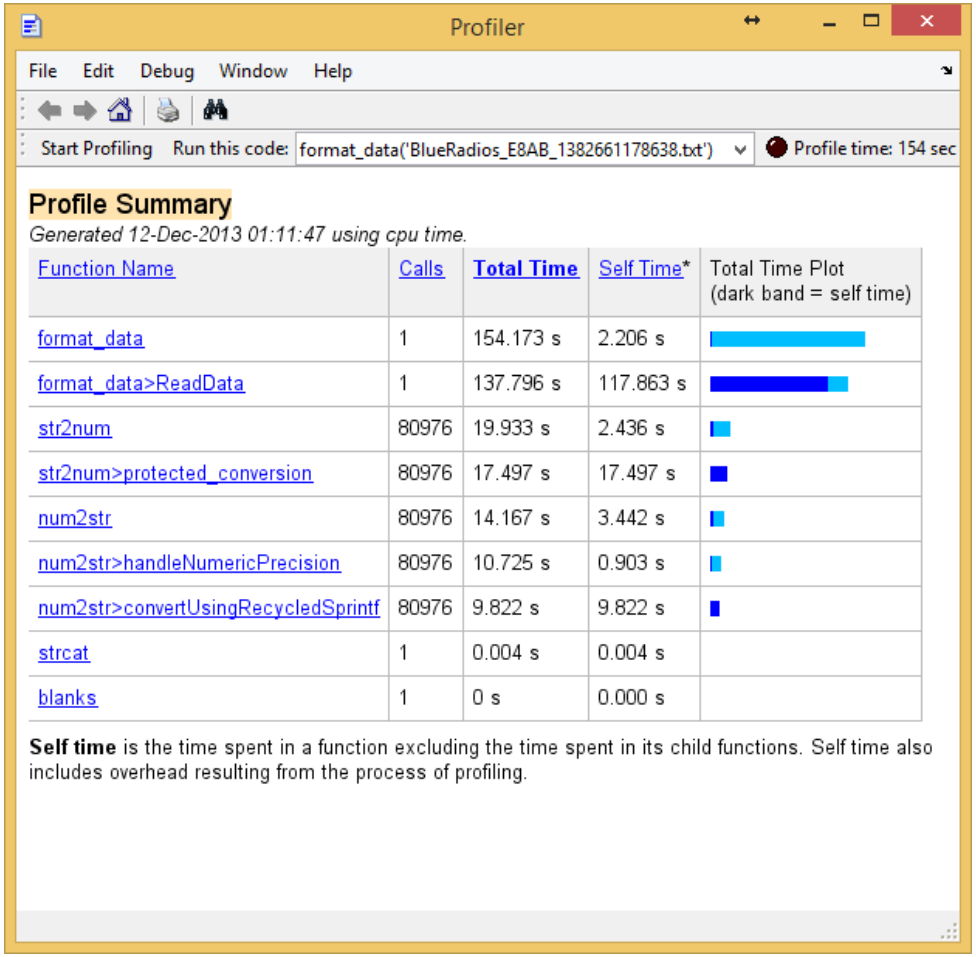
$$\sum_{k=1}^{n} k = \frac{n(n+1)}{2},$$

times for a file of k lines

format_data_2013.m

- Uses Matlab specific function for file import, not C functions
- Does not involve creation of a new array for each additional line
- Just how much faster is it?

format_data_2013



Profile Summary

Generated 12-Dec-2013 01:11:47 using cpu time.

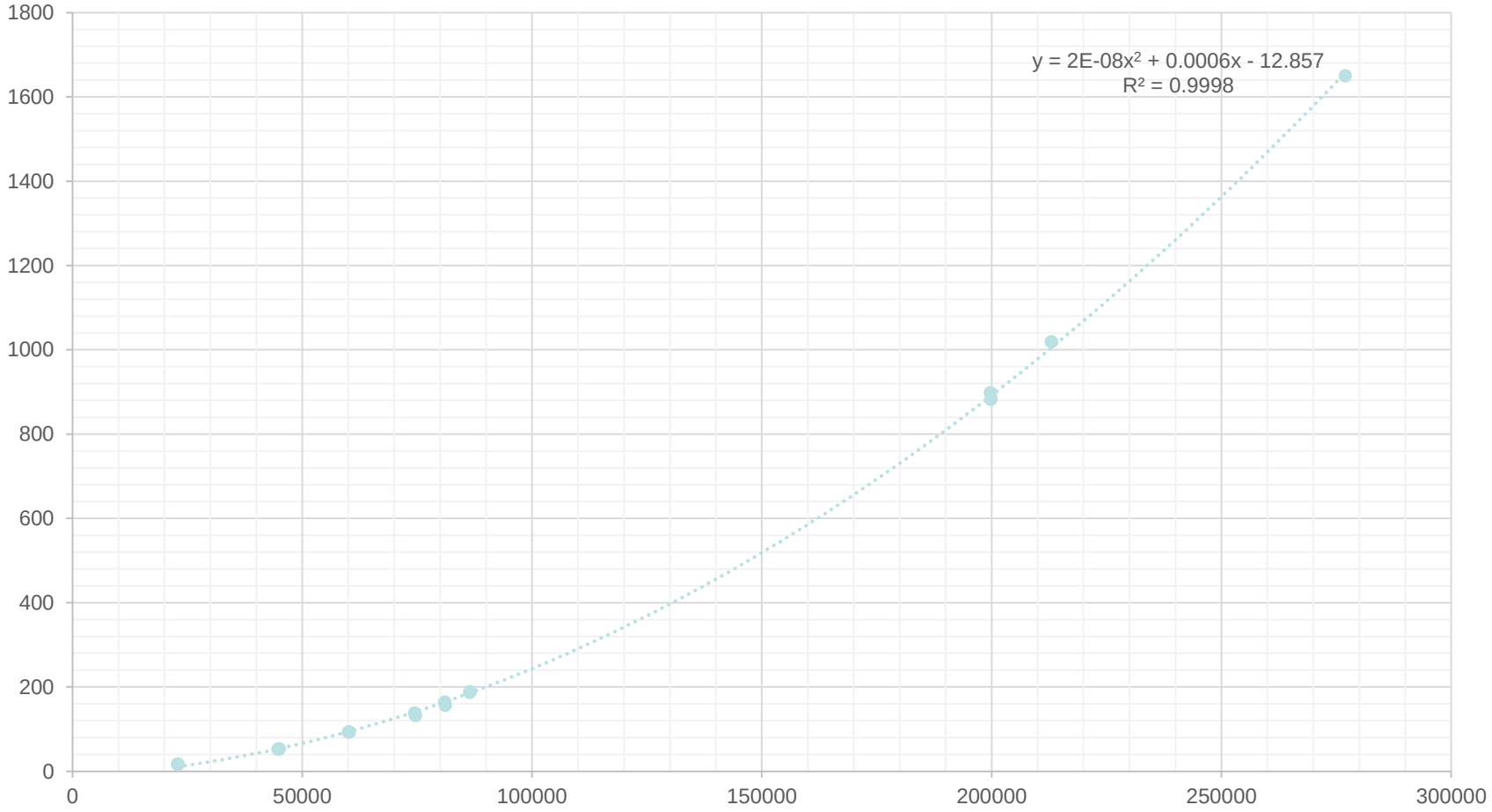
Function Name	Calls	**Total Time**	Self Time*	Total Time Plot (dark band = self time)
format_data	1	154.173 s	2.206 s	
format_data>ReadData	1	137.796 s	117.863 s	
str2num	80976	19.933 s	2.436 s	
str2num>protected_conversion	80976	17.497 s	17.497 s	
num2str	80976	14.167 s	3.442 s	
num2str>handleNumericPrecision	80976	10.725 s	0.903 s	
num2str>convertUsingRecycledSprintf	80976	9.822 s	9.822 s	
strcat	1	0.004 s	0.004 s	
blanks	1	0 s	0.000 s	

Self time is the time spent in a function excluding the time spent in its child functions. Self time also includes overhead resulting from the process of profiling.

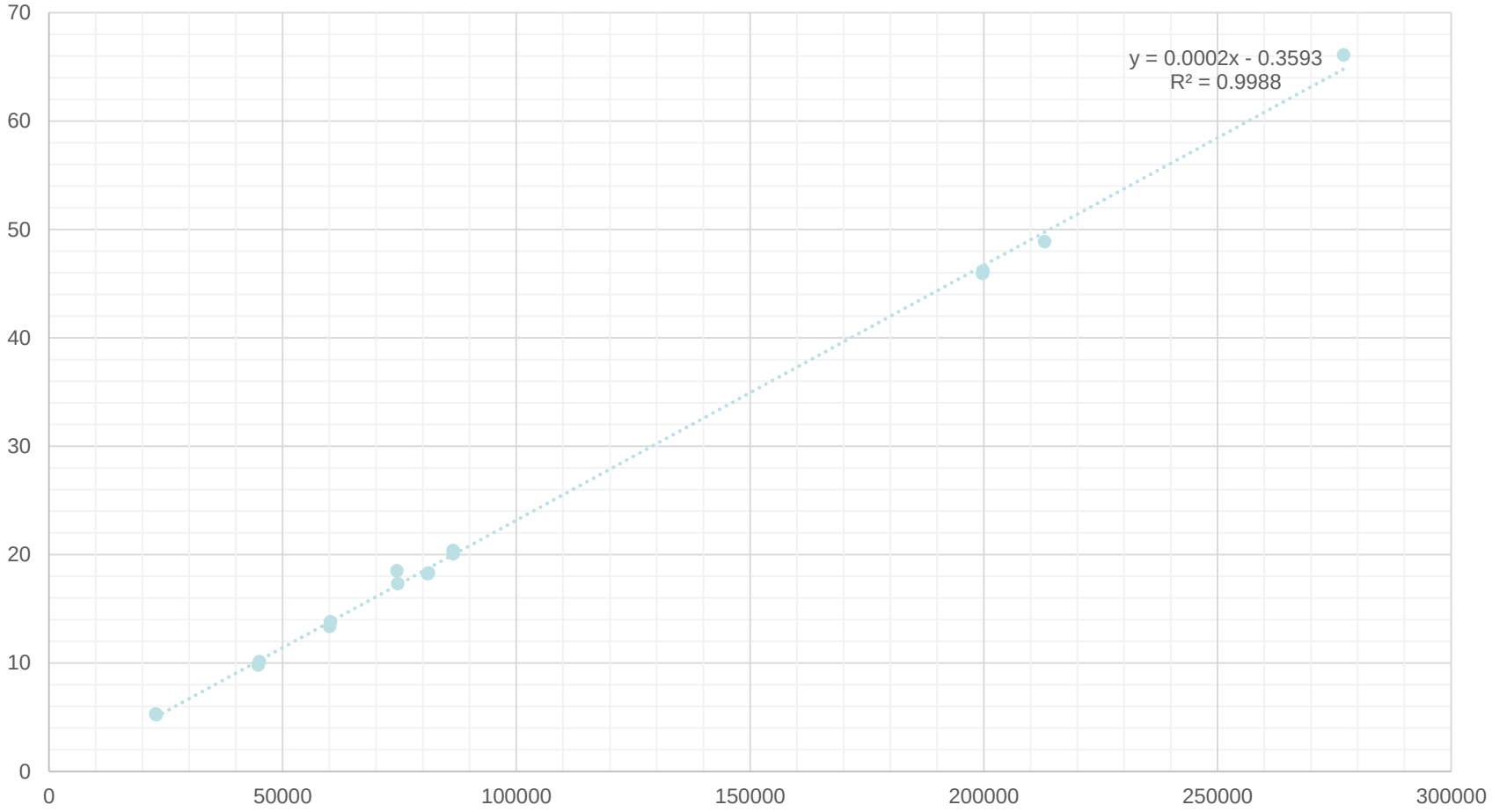
Empirically measured runtimes

		Running Time	
Filesize	# of lines	format_data.m (seconds)	format_data_2013.m (seconds)
(2,063,944 bytes)	22851	16.982	5.285
(1,994,581 bytes)	22943	16.711	5.224
(3,990,815 bytes)	44748	52.534	9.803
(4,057,787 bytes)	44966	53.493	10.129
(5,374,936 bytes)	60040	93.433	13.36
(5,416,544 bytes)	60239	93.41	13.808
(6,708,418 bytes)	74459	138.17	18.509
(6,655,485 bytes)	74644	132.847	17.303
(7,169,115 bytes)	80977	164.173	18.222
(7,126,623 bytes)	81152	156.504	18.304
(7,761,494 bytes)	86471	187.755	20.066
(7,797,077 bytes)	86490	189.032	20.373
(17,906,771 bytes)	199818	882.963	46.179
(18,088,959 bytes)	199726	897.837	45.928
(24,702,347 bytes)	276963	1650.192	66.083
(19,067,061 bytes)	213001	1018.99	48.875

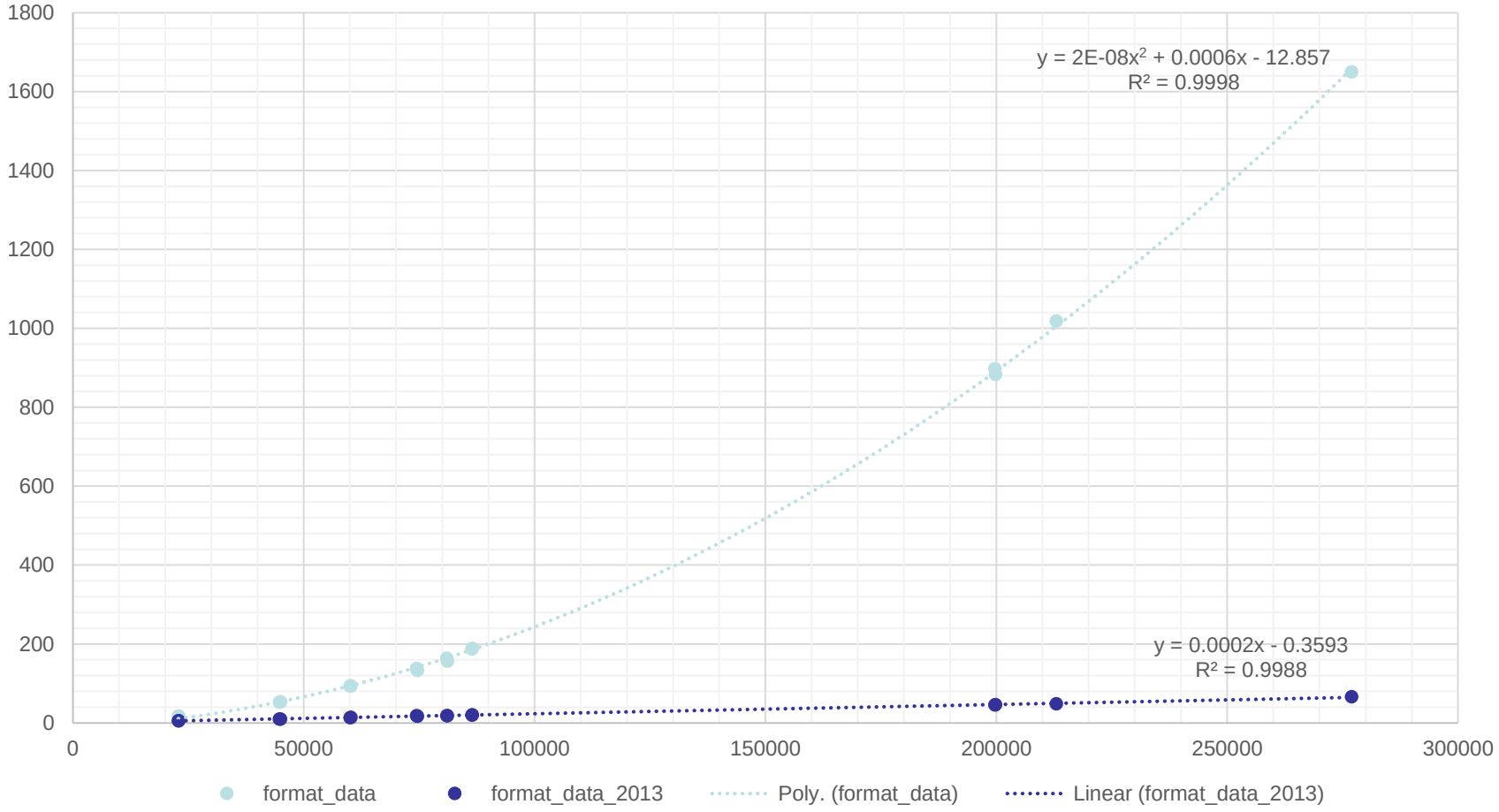
Number of Lines Vs Runtime format_data



Number of Lines Vs Runtime format_data_2013

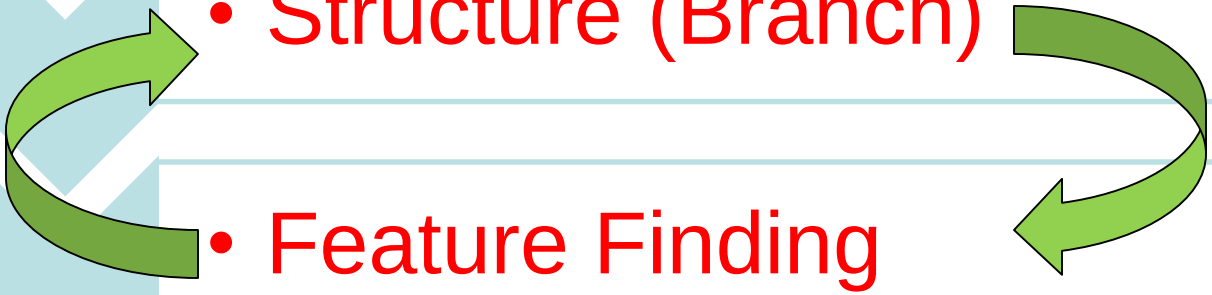


Number of Lines Vs Runtime Both

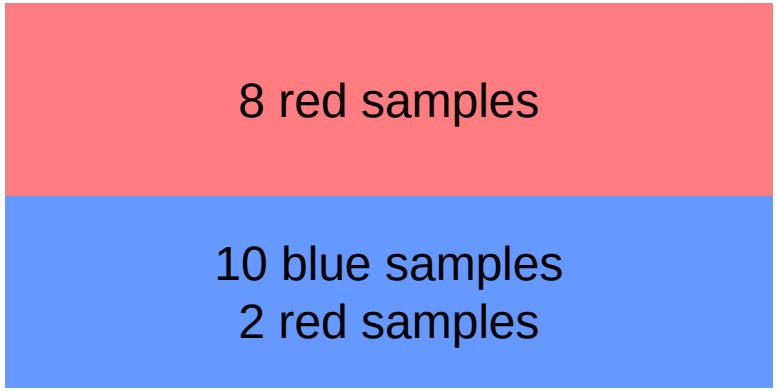


System Implementation

- Data Collection, Merge, Align, Label
- Structure (Branch)
- Feature Finding
- Testing

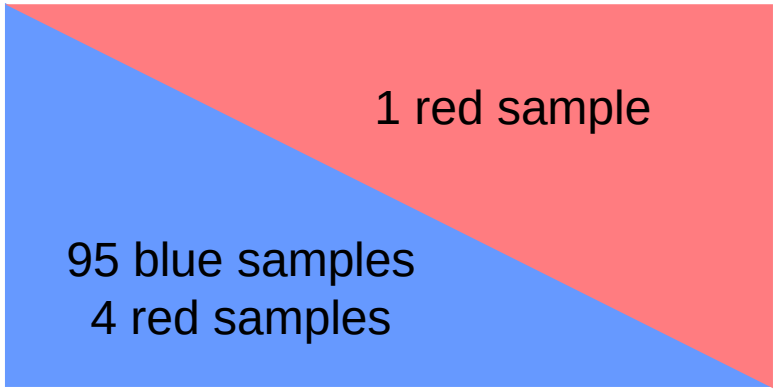


90%>96%?



Classification #1
Total Accuracy is (20-2)/20=90%

Accuracy for red is 80%
Accuracy for blue is 100%



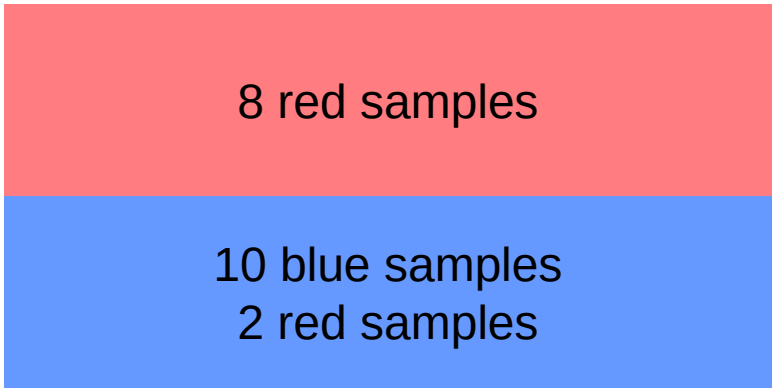
Classification #2
Total Accuracy is (100-4)/100=96%

Accuracy for red is 20%
Accuracy for blue is 100%
Potential failure on distinguishing red out from blue

“Accuracy” in feature finding might not be accurate

- Relevant to the proportion of the testing data
- Thus this accuracy is not representative for the feature finding
- Not practical to demand a dataset with perfect balance in proportion
- The accuracy should be related to the size of the data for each activity
- ROC curve method, but with respect to the features instead of a “threshold”

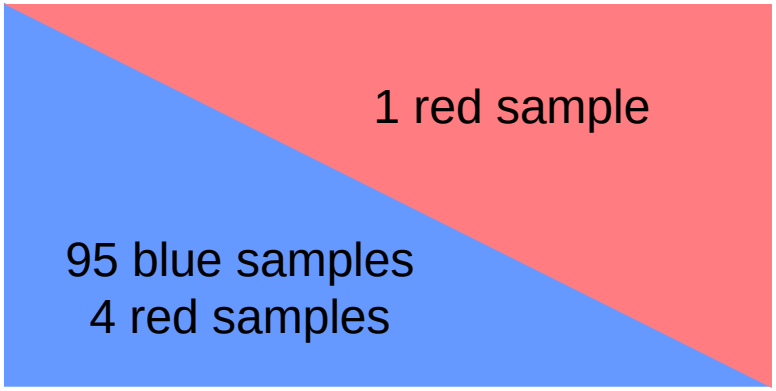
90%>96%!



Classification #1

True Positive: 80%	**False Positive: 0%**
False Negative: 20%	True Negative: 100%

Unbiased Accuracy = (TP+TN)/2= 90%

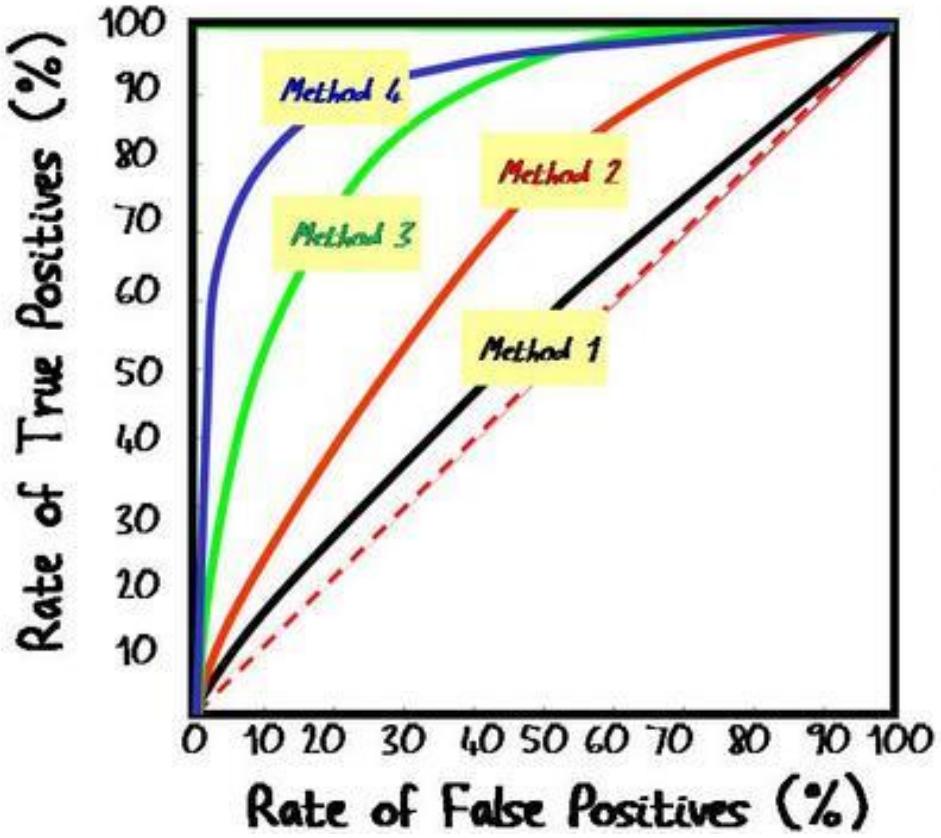


Classification #2

True Positive: 20%	**False Positive: 0%**
False Negative: 80%	True Negative: 100%

Unbiased Accuracy = (TP+TN)/2= 60%

ROC CURVE EXAMPLES



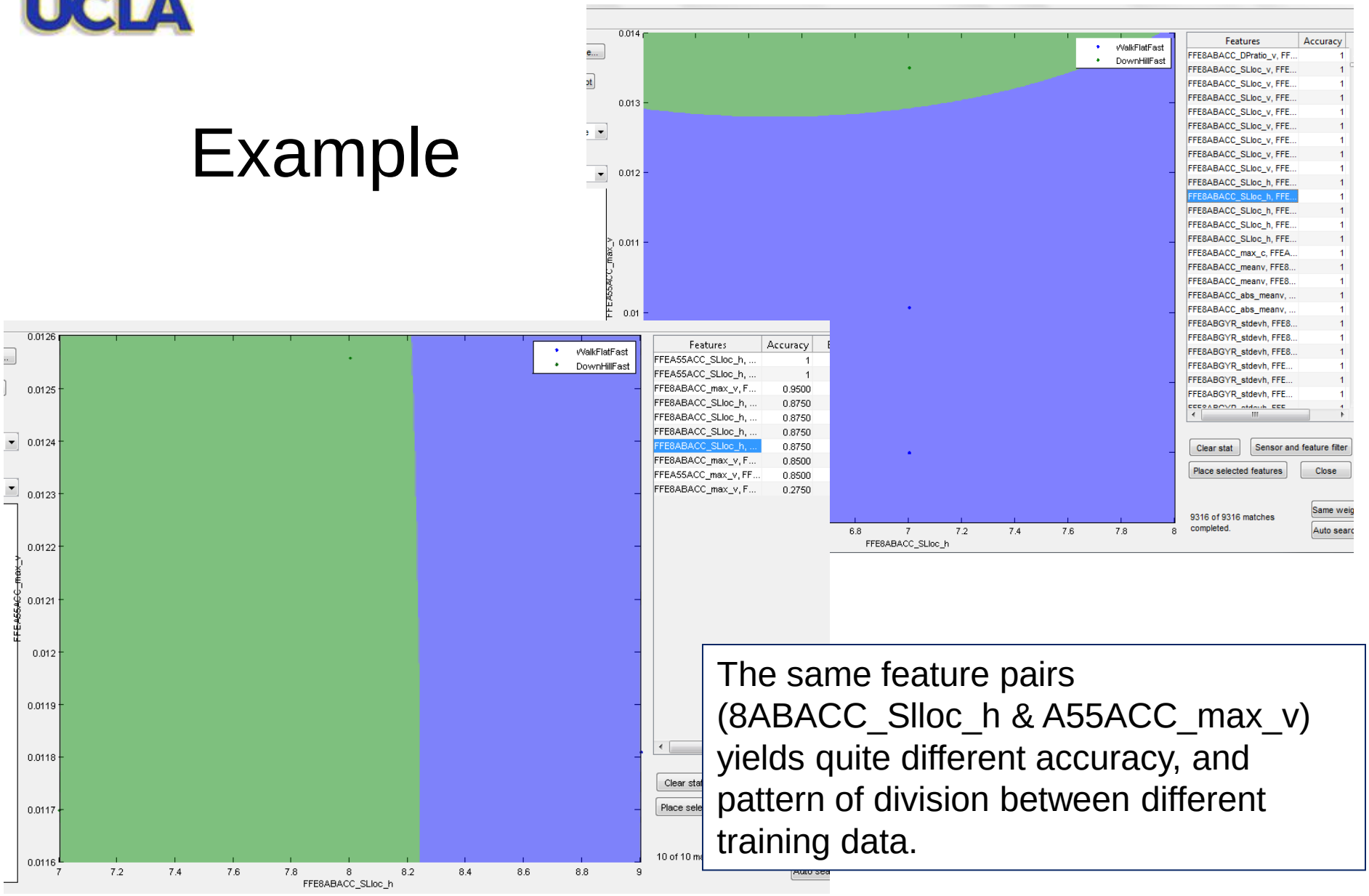
- The best classification has the largest area under the curve.
- Very sensitive to errors in the "gold standard" classification.

http://csb.stanford.edu/class/public/lectures/Lecture6/Lecture6/Data_Visualization/images/Roc_Curve_Examples.jpg

Discrepancy across datasets

- Discrepancy across datasets is significant
- Need to be take care of by training across datasets
- A general “cloud” training data might be helpful for mass application which has:
  - The authentic good features for the action
  - Limited training data for specific user
  - Finding the “problem” of users of interest

Example



The same feature pairs (8ABACC_Slloc_h & A55ACC_max_v) yields quite different accuracy, and pattern of division between different training data.

Making the tree

- With limited training data which is not easy to get 100% or even 95%, the tree making is crucial considering the issue of the “accuracy”.
- Actions can be reasonably divided in multiple ways
  - Orientation
  - Speed
  - physical pattern
- Tricky case: going up the stair two steps at a time
- “Better” division should be in the larger branches
- Examine the tree from both feature finding and the testing outcome

Tree Making Strategies(Level 5)

- Three +1 Division basis:
  - Speed (Fast and Slow)
  - Orientation (Up and Down)
  - Motion Pattern (Walk Flat, Stair and Hill)
  - Separating One motion at a time
    - Limitation on building such tree

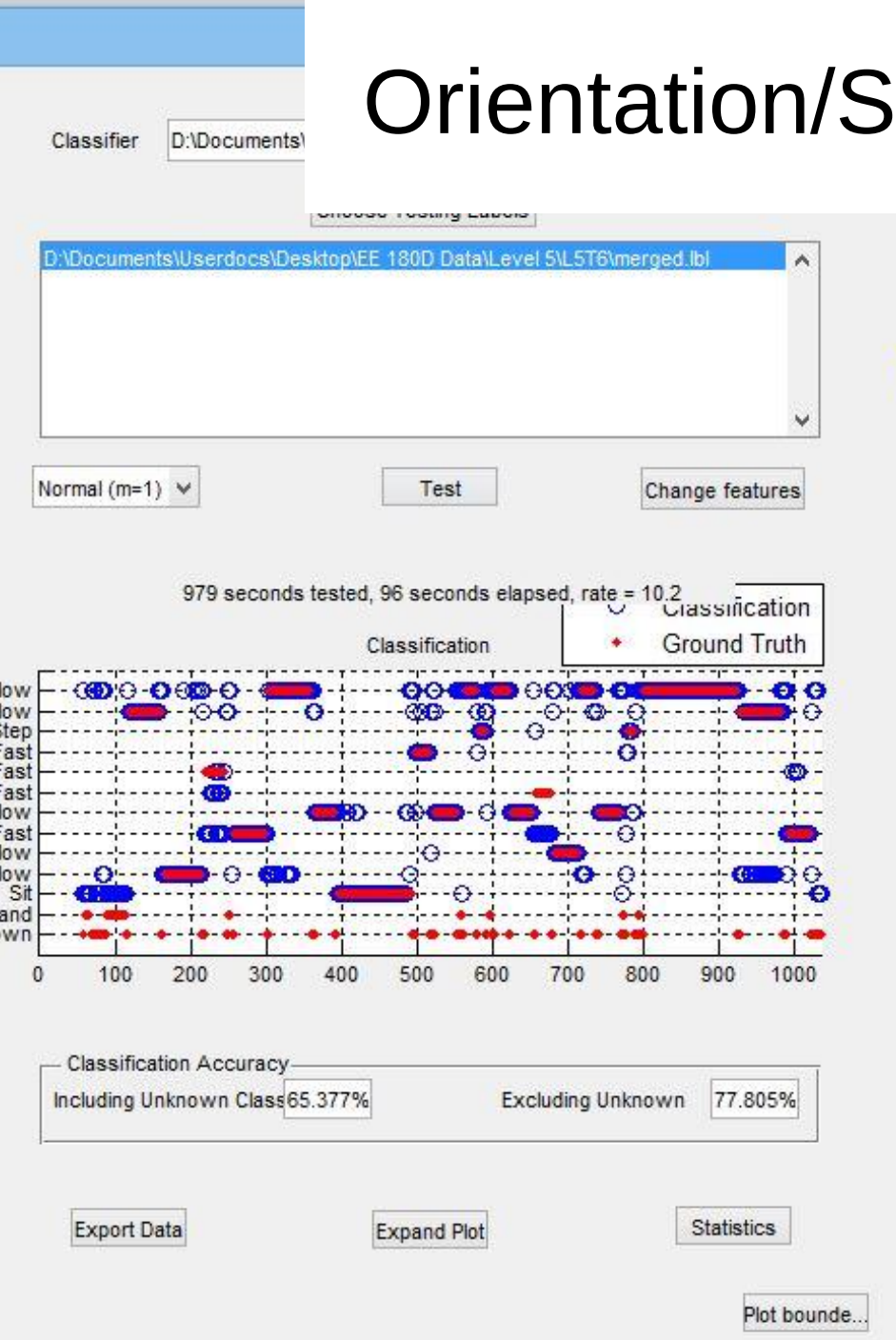
Motion/Speed/Orientation Tree

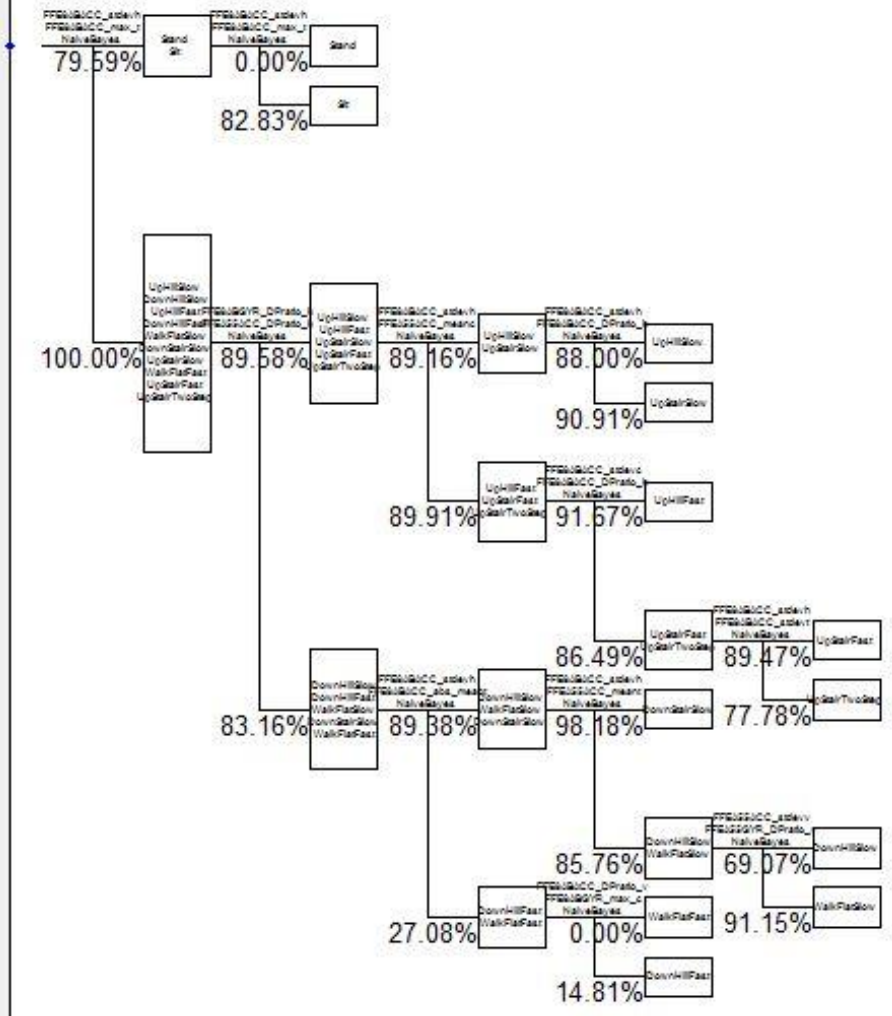
- Main Branch: Motion(in feature finding this branch gives ~80% accuracy)
  - Hill, Walk Flat, vs Stair
- Sub-Branch: Speed
  - Fast vs Slow
- Sub-Branch: Orientation
  - Up and Down
- Accuracy: 78.267%

Orientation/Speed/Motion Tree

- Main Branch: Orientation(in feature finding this branch gives ~90% accuracy)
  - Up vs Down
- Sub-Branch : Speed
  - Fast vs Slow
- Sub-Branch: Motion
  - Hill, Walk Flat vs Stair
- Accuracy: 77.805%

Orientation/Speed/Motion Result

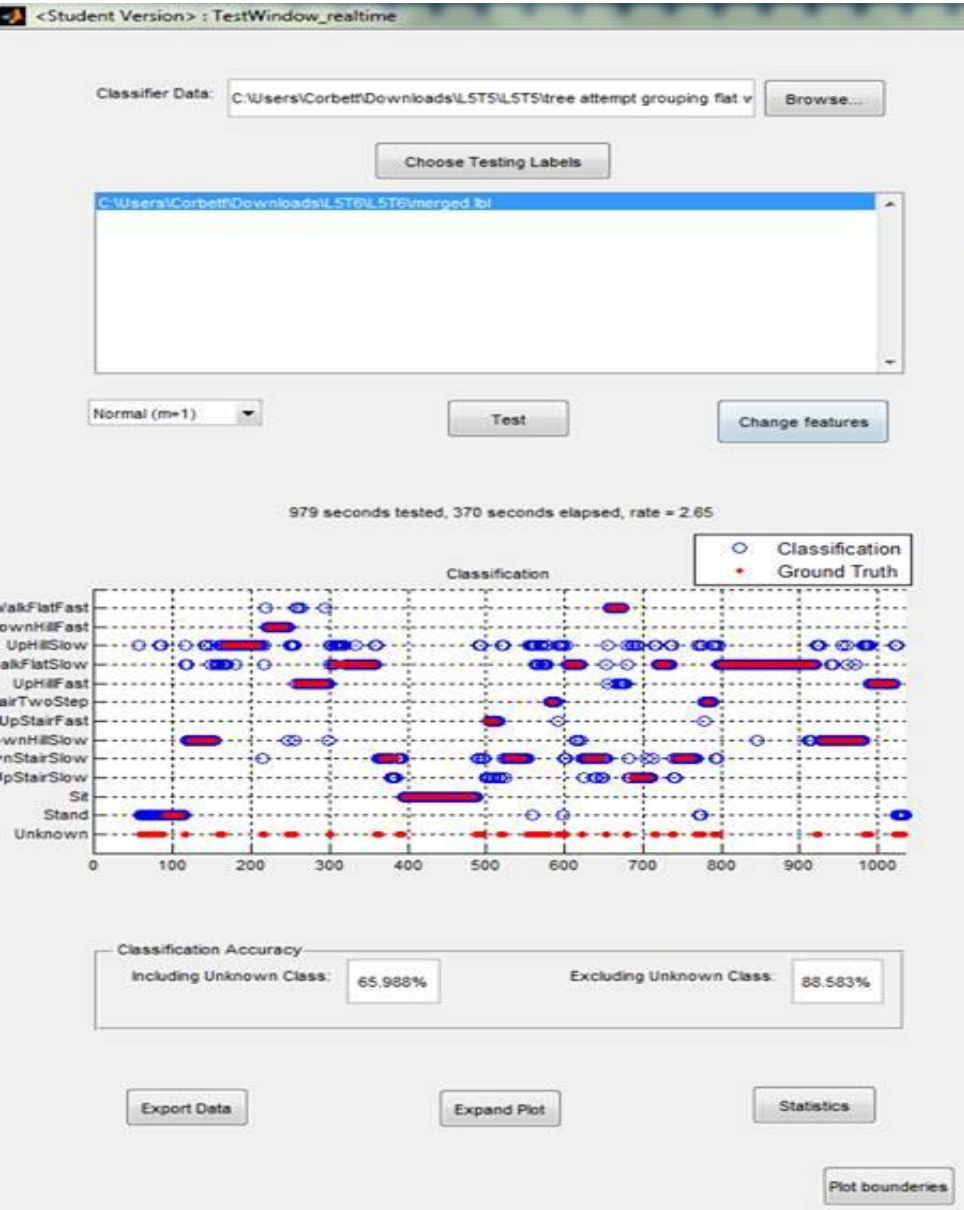


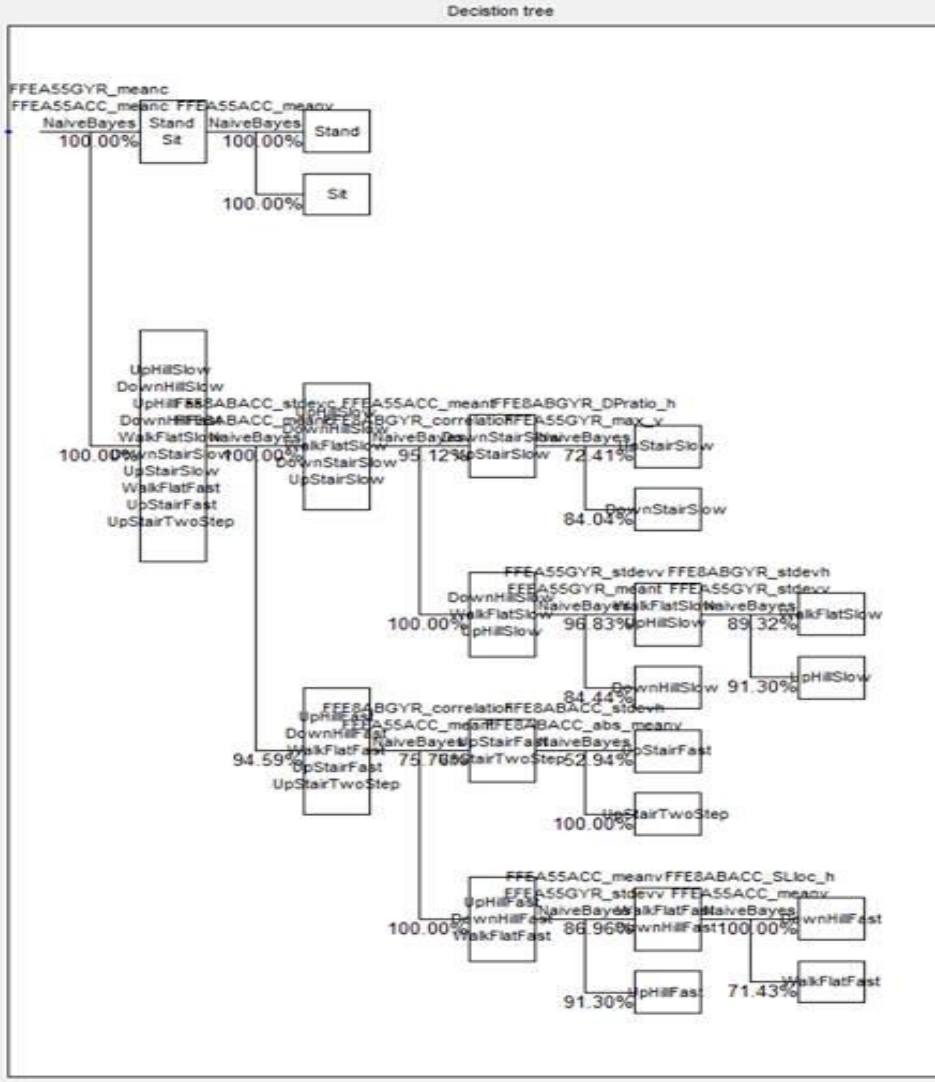


Speed/Orientation/Motion Tree

- Main Branch: Speed(in feature finding this branch gives ~100% accuracy)
  - Slow vs Fast
  - Up Stair Two-Step (Fast)
- Sub-Branch: Orientation
  - Up vs Down
- Sub-Branch: Motion
  - Hill, Walk Flat, and Stair
- Accuracy: 88.563%

Speed-Orientation-Motion Result





Testing and Verification

- It takes long time the analyze/test on one whole long complicated data set at a time
- For future development, we hope we can have a feature that can test one branch at a time instead of testing the whole tree and the branches that are irrelevant to the branch that has the feature change

Results to Date

- Level 1: 100%
- Level 2: 97.059%
- Level 3: 89.216%
- Level 4: 100%
- Level 5 (with transition): 85.000%
- Level 5 (without transition): 88.583%
- Level 6: 99.492%

Variable Meanings Level 5

- Branch 2: sit vs. stand
  - EA55ACC_meanv (thigh average acceleration for x axis)
- Branch 3: fast actions vs. slow actions
  - E8ABACC_meanc (foot average accelerometer energy)
  - E8ABACC_stdevc (foot standard deviation of accelerometer energy)
- Branch 4: fast hill & flat walking vs. stairs
  - EA55ACC_meant (thigh average acceleration for z axis)
  - E8ABGYR_correlation (foot correlation between pitch and roll variables)
- Branch 5: slow hill & flat walking vs. stairs
  - Same variables as branch 4, but potentially with different thresholds
- Other branches could be distinguished using standard deviation / mean of various axes, or variables we already established as effective in previous steps

Variable Meanings Level 6

- Branch 1: normal gait vs. affected gait
  - 627FACC_stdevc (right foot standard deviation of energy)
  - E915ACC_stdevc (left foot standard deviation of energy)
- Branch 2: cane vs. no cane
  - E8CBACC_stdevt (right ankle standard deviation of z axis)
  - EA15GYR_meanc (left ankle average energy)
- Branch 3: severe vs. mild
  - E8CBACC_SLloc_h (right ankle highest frequency magnitude)
  - EA15GYR_DPratio_v(degree of spread of frequencies for x axis)

Team and Responsibilities

- Data Collection
  - Anthony Nguyen and Corbett Cappon as primary test subjects
- Feature Finding and Tree Building
  - Zhe Wan, Jincheng Zhou and Corbett Cappon
- Features and Reality
  - Corbett Cappon

Conclusion & Anticipation

- Difficulty comes from large tree with many motions and also the irregularity of the motion.
- In practice, training data can be related to the cloud
- In stead of categorizing the existing motions in training, the system might discover “symptoms” with big data
- Multi-dimension features for classification
- Real-time testing

UCLA

Thank You!

Questions?

Reference

- Professor
- TA