SIGNAL PROCESSING & ML IN UBICOMP (CONT)

CSE 590 Ubiquitous Computing | Lecture 4 | April 19

Jon Froehlich • Liang He (TA)

DESIGN USE BUILD

MAKEABILITY LAB

PAUL G. ALLEN SCHOOL
OF COMPUTER SCIENCE & ENGINEERING

UNIVERSITY of WASHINGTON

SCHEDULE TODAY: 6:30-9:20

06:35-06:55: Daisy Isibor’s discussion of the Saponas et al., CHI’08 reading

06:55-07:05: Apurv Suman’s discussion of the Harrison and Hudson Scratch Input paper

07:05-08:00: Signal processing & ML in ubicomp systems cont (w/ Jupyter Notebook exercises)

07:55-08:05: Break

08:05-08:30: Continue lecture and exercises

08:30-09:20: Work on Part 2 of A2: Gesture Recognizer. Specifically, implementing an SVM.

BACKGROUND

MACHINE LEARNING

> An approach to achieve artificial intelligence through systems that can learn from experience to find patterns in a set of data.

Jason Mayes
Senior Creative Engineering, Google

Source: Mayes, Machine Learning 101, (link)

BACKGROUND

MACHINE LEARNING

> The goal of machine learning is to build computer systems that can adapt and learn from their experience.

Tom Dietterich
Professor, University of Oregon

BACKGROUND

MACHINE LEARNING

ML involves **teaching a computer to recognize patterns** by example rather than programming a system to recognize such patterns with specific rules.

In other words, **ML is about creating algorithms that learn complex models/functions from data** and then make predictions on similar data.

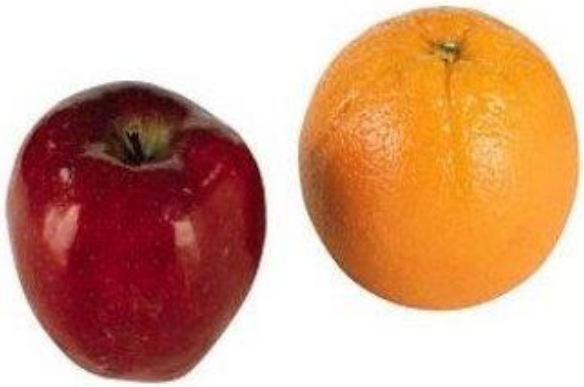
Source: Mayes, Machine Learning 101, (link)

SUPERVISED LEARNING

1. Input representative data to train ML system
2. ML system learns patterns from data
3. Then feed in data ML system has never seen before and it returns its best guess for what that data is

Source: Mayes, Machine Learning 101, (link)

SUPERVISED LEARNING



Example: let's build an ML model for recognizing apples and oranges….

SUPERVISED LEARNING

Input Features

Class Label
(aka Ground Truth Label)

Color	Weight	Label
Red	200g	Apple
Orange	300g	Orange
Red	150g	Apple
...		

Training Data

ML System

ML OVERVIEW

SUPERVISED LEARNING

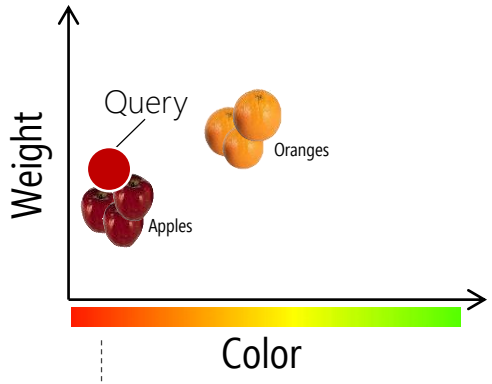
Input Features

Class Label
(aka Ground Truth Label)

Color	Weight	Label
Red	200g	Apple
Orange	300g	Orange
Red	150g	Apple
...		

Training Data

ML Model



Color	Weight	Label
Red	170g	Unknown

Query Data

ML OVERVIEW

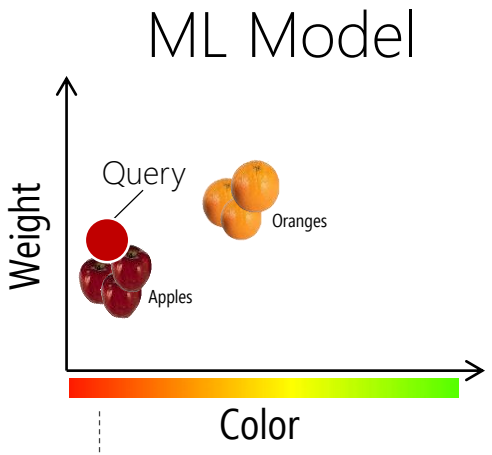
SUPERVISED LEARNING

Input Features

Class Label
(aka Ground Truth Label)

Color	Weight	Label
Red	200g	Apple
Orange	300g	Orange
Red	150g	Apple
...		

Training Data



ML System

Color	Weight	Label
Red	170g	Unknown

Query Data

Color	Weight	Label
Red	170g	Apple

Prediction

SUPERVISED LEARNING PROBLEMS AND APPROACHES

CLASSIFICATION PROBLEMS

When you need to classify some input vector into a category (*e.g.,* Apple or Orange). Output is some class label *C*.

Popular approaches:

Support Vector Machines (SVMs)

Random forests

Neural Networks

Deep Learning

REGRESSION PROBLEMS

When you want to predict some numeric value (*e.g.,* house prices next year in Seattle). Output has infinitely many values (a number).

Popular approaches:

Linear regression

Random forests

Source: Brownlee, Supervised and Unsupervised Machine Learning Algorithms, 2016 (link)

UNSUPERVISED LEARNING

Input Features

No Class Label!
(aka No Ground Truth Labels)

Color	Weight	Label
Red	200g	?
Orange	300g	?
Red	150g	?
...		

Training Data

ML System

UNSUPERVISED LEARNING

Input Features

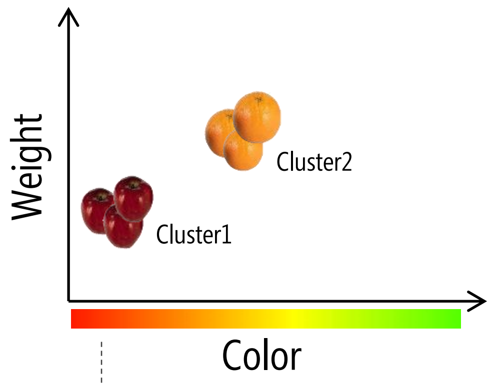
No Class Label!
(aka No Ground Truth Labels)

Color	Weight	Label
Red	200g	?
Orange	300g	?
Red	150g	?
...		

Training Data

ML System

ML Model



ML OVERVIEW

UNSUPERVISED LEARNING

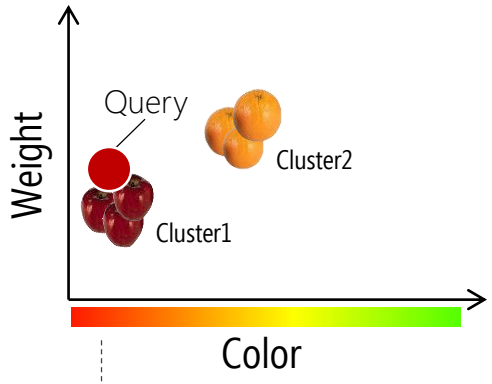
Input Features

No Class Label!
(aka No Ground Truth Labels)

Color	Weight	Label
Red	200g	?
Orange	300g	?
Red	150g	?
...		

Training Data

ML Model



ML System

Color	Weight	Label
Red	170g	?

Query Data

Color	Weight	Label
Red	170g	Cluster1

Prediction

UNSUPERVISED LEARNING PROBLEMS AND APPROACHES

CLUSTERING PROBLEMS

When you want to discover inherent groupings in your data (*e.g.,* grouping customers by purchasing behavior)

Popular approaches:

K-means clustering

ASSOCIATION PROBLEMS

When you want to discover associative rules between large portions of data (*e.g.,* people that buy X also tend to buy Y)

Popular approaches:

Apriori algorithm

Collaborative filtering

Source: Brownlee, Supervised and Unsupervised Machine Learning Algorithms, 2016 (link)

SEMISUPERVISED LEARNING

Input Features

Some Class Labels
(aka Ground Truth Labels)

Color	Weight	Label
Red	200g	Apple
Orange	300g	Orange
Red	150g	Apple
Green	150g	?
...		

Training Data

ML System

ML OVERVIEW

SEMISUPERVISED LEARNING

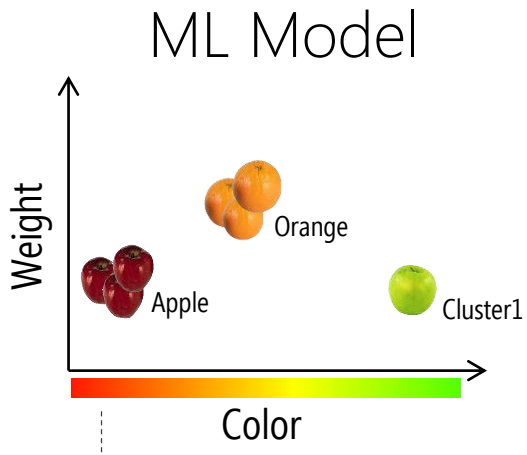
Input Features

Some Class Labels
(aka Ground Truth Labels)

Color	Weight	Label
Red	200g	Apple
Orange	300g	Orange
Red	150g	Apple
Green	150g	?
...		

Training Data

ML Model



ML OVERVIEW

SEMISUPERVISED LEARNING

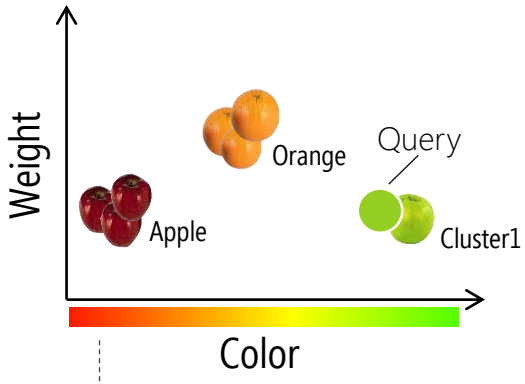
Input Features

Some Class Labels
(aka Ground Truth Labels)

Color	Weight	Label
Red	200g	Apple
Orange	300g	Orange
Red	150g	Apple
Green	150g	?
...		

Training Data

ML Model



ML System

Color	Weight	Label
Green	170g	?

Query Data

Color	Weight	Label
Red	170g	Cluster1

Prediction

ML OVERVIEW

SEMISUPERVISED LEARNING

Input Features

Some Class Labels
(aka Ground Truth Labels)

Color	Weight	Label
Red	200g	Apple
Orange	300g	Orange
Red	150g	Apple
Green	150g	?
...		

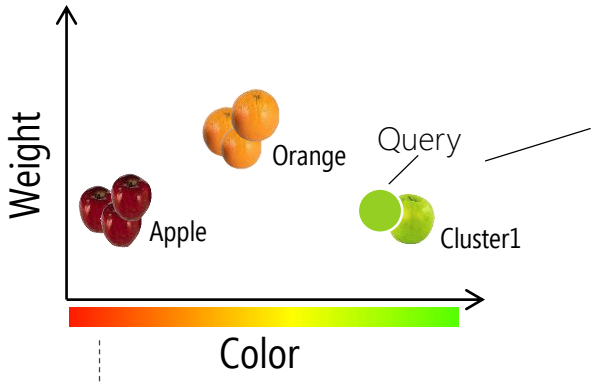
Training Data

ML System

Color	Weight	Label
Green	170g	?

Query Data

ML Model



UI may prompt user for label. "It looks like you just did something I didn't recognize, mind telling me what it was?"

Color	Weight	Label
Red	170g	Cluster1

Prediction

HOW TO CHOOSE A MODEL?

Your decision should be based on the nature of the dataset (*e.g.*, images, videos, text, continuous time-series)

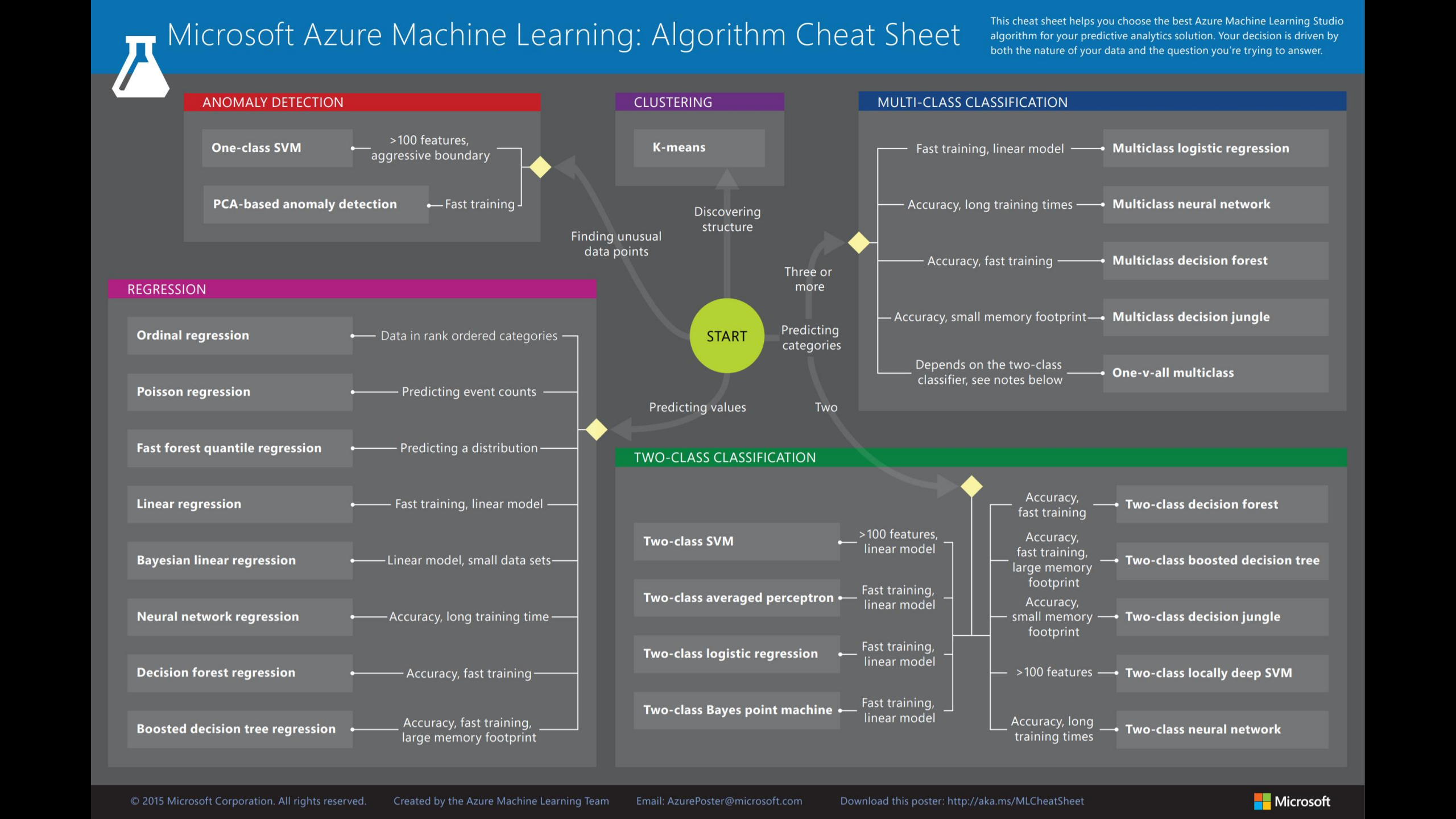
And the question you're trying to answer (*e.g.*, What is likely to happen next? What are the anomalies in this dataset? What objects are in this image? What gesture is the user performing?)

Often, to answer these questions, we explore the literature to see how others have solved similar problems and adapt…

For A2, we strongly recommend an SVM

Microsoft Azure Machine Learning: Algorithm Cheat Sheet

This cheat sheet helps you choose the best Azure Machine Learning Studio algorithm for your predictive analytics solution. Your decision is driven by both the nature of your data and the question you're trying to answer.



Created by the Azure Machine Learning Team

Email: AzurePoster@microsoft.com

Download this poster: http://aka.ms/MLCheatSheet

Microsoft

MACHINE LEARNING APPROACHES: ORGANIZED BY TYPE

Source: Brownlee, A Tour of Machine Learning Algorithms, 2013 (link)

How to select our **input features**?

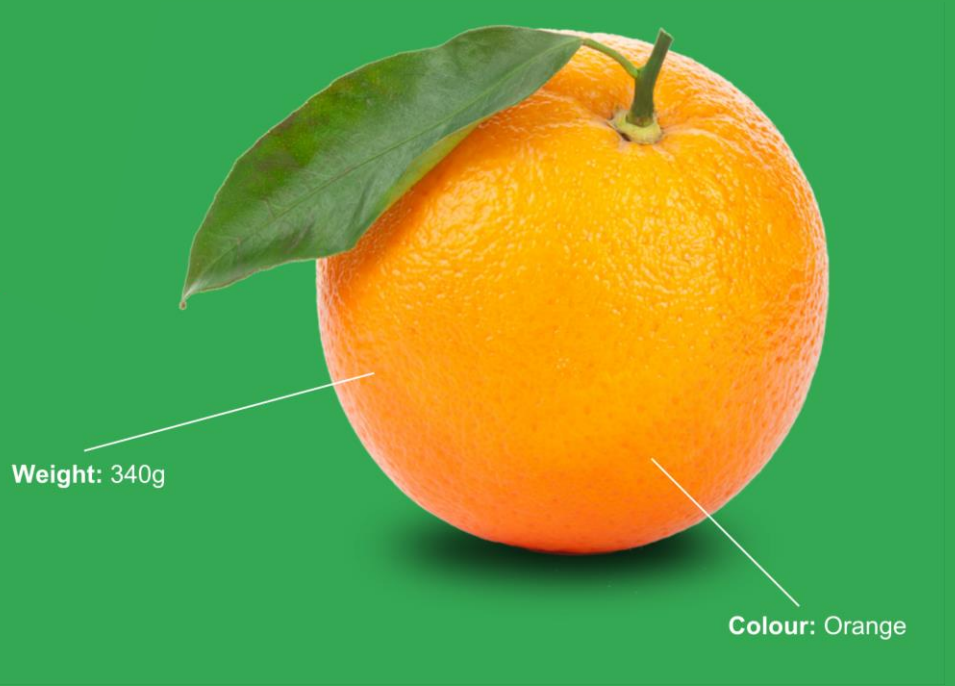
ML TRAINING

FEATURES

Features are used to **train** the ML system.

You want to **pick features that you think help discriminate** your dataset.

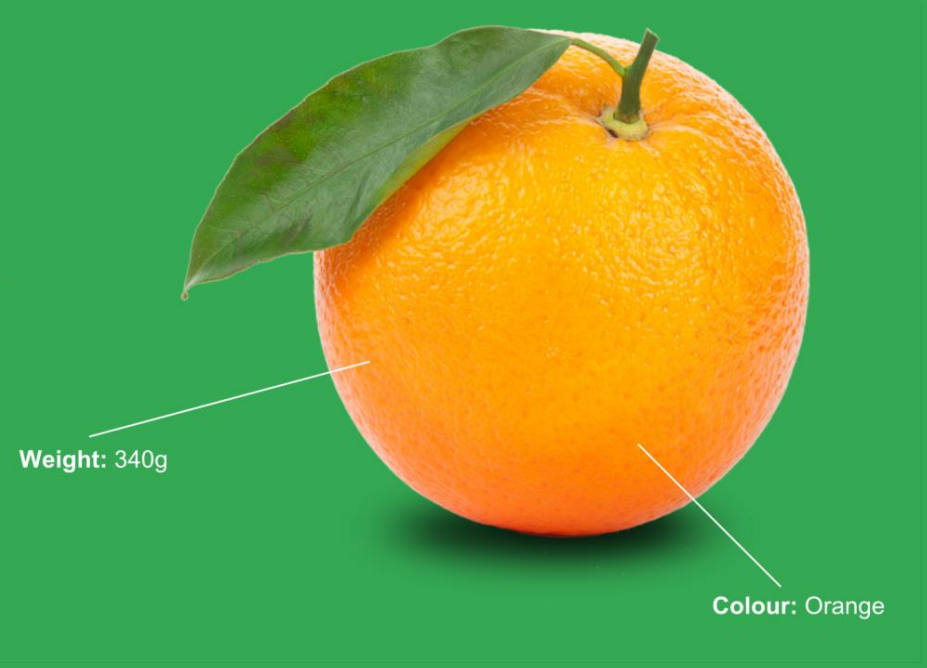
With fruit, for example, you might try **weight** and **color**. **Two features**, which means your featureset is **two-dimensional**



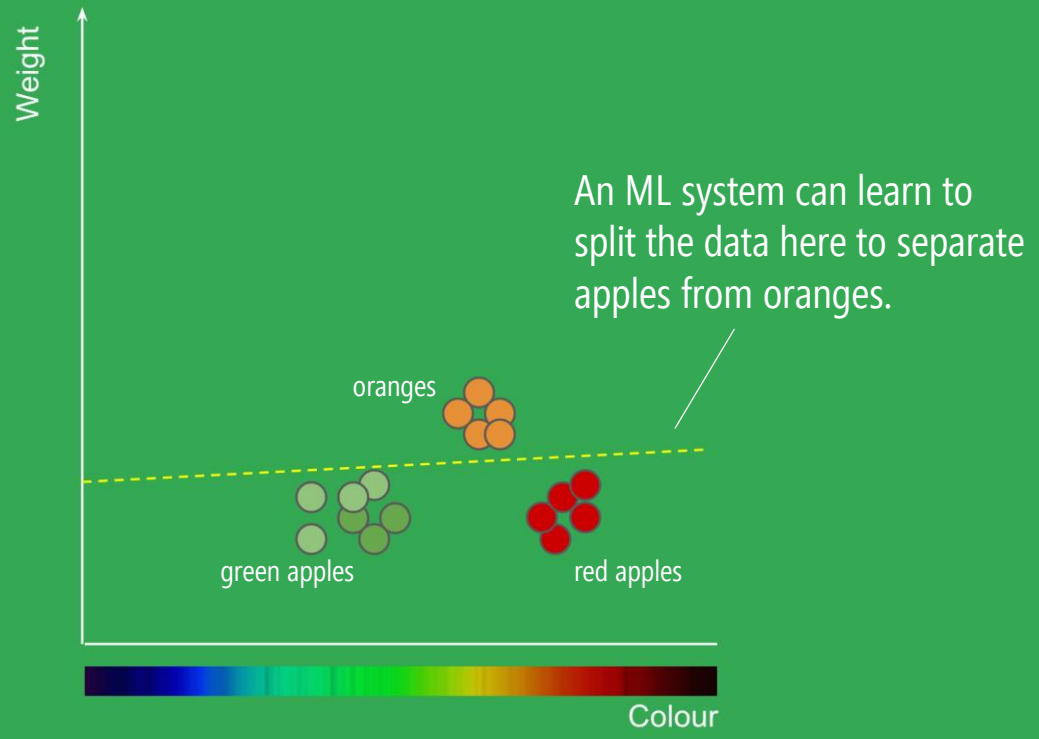
Source: Mayes, Machine Learning 101, (link)

ML TRAINING

SELECTING FEATURES



Picking features. If we were building a classifier to automatically distinguish fruits, we could setup a 2D feature vector: weight and color

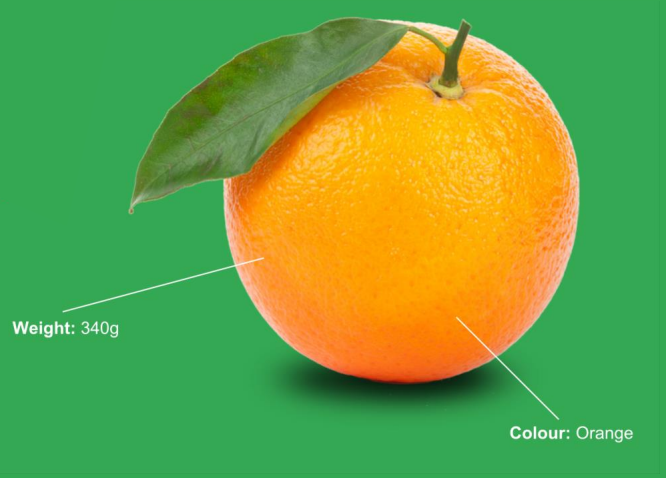


Brainstorm, visualize, investigate, & iterate your feature space. When exploring features, it's a good idea to plot them on a graph (when possible) to develop intuition & assess discriminability.

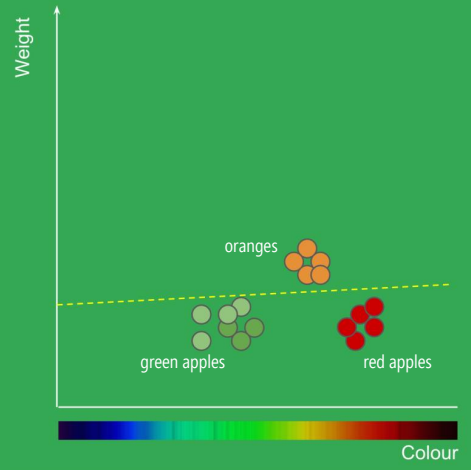
Source: Mayes, Machine Learning 101, (link)

ML TRAINING

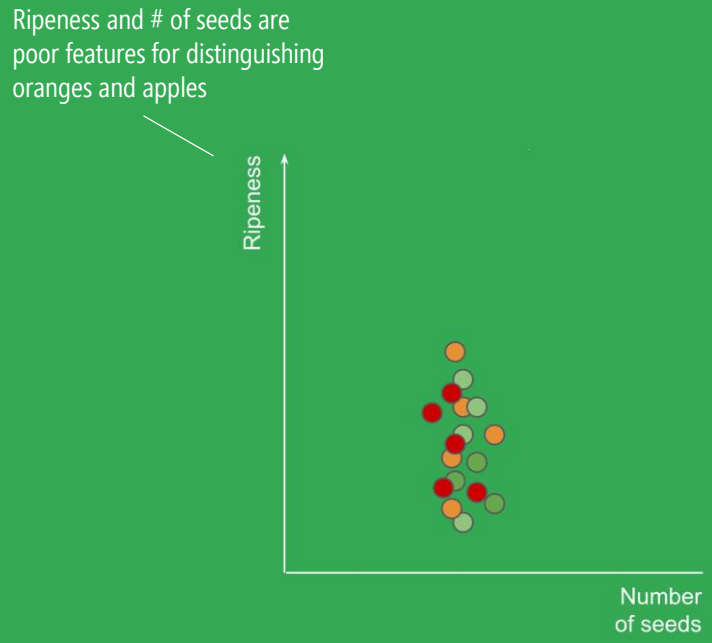
SELECTING FEATURES



Picking features. if we were building a classifier to automatically distinguish fruits, we could setup a 2D feature vector: weight and color



Brainstorm, visualize, investigate, & iterate your feature space. When exploring features, it's a good idea to plot them on a graph (when possible) to develop intuition and understand their discriminability.



Choose your features carefully. Some features may not help you distinguish data. For example, ripeness and number of seeds are bad features.

Source: Mayes, Machine Learning 101, (link)

SELECTING FEATURES: HIGHER DIMENSIONALITY

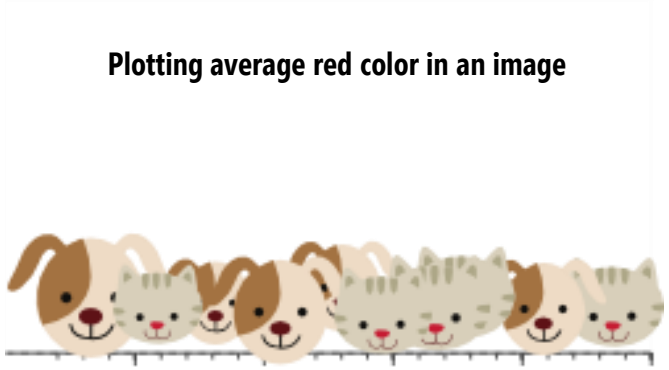
Generally, ML systems are trained with (super!) high dimensional feature vectors that are hard (impossible) to plot and develop an intuition for.

We can plot maybe ~4 dimensions at a time (x, y, z-axis + some visual attribute of the marker).

ML TRAINING

SELECTING FEATURES: HIGHER DIMENSIONALITY

Let's look at another example: this time trying to separate cats and dogs using average pixel color in image.



1D Feature Vector

Plotting average red color in image. Notice how a single feature does not result in a perfect separation of our training data.

Source: http://www.visiondummy.com/2014/04/curse-dimensionality-affect-classification/

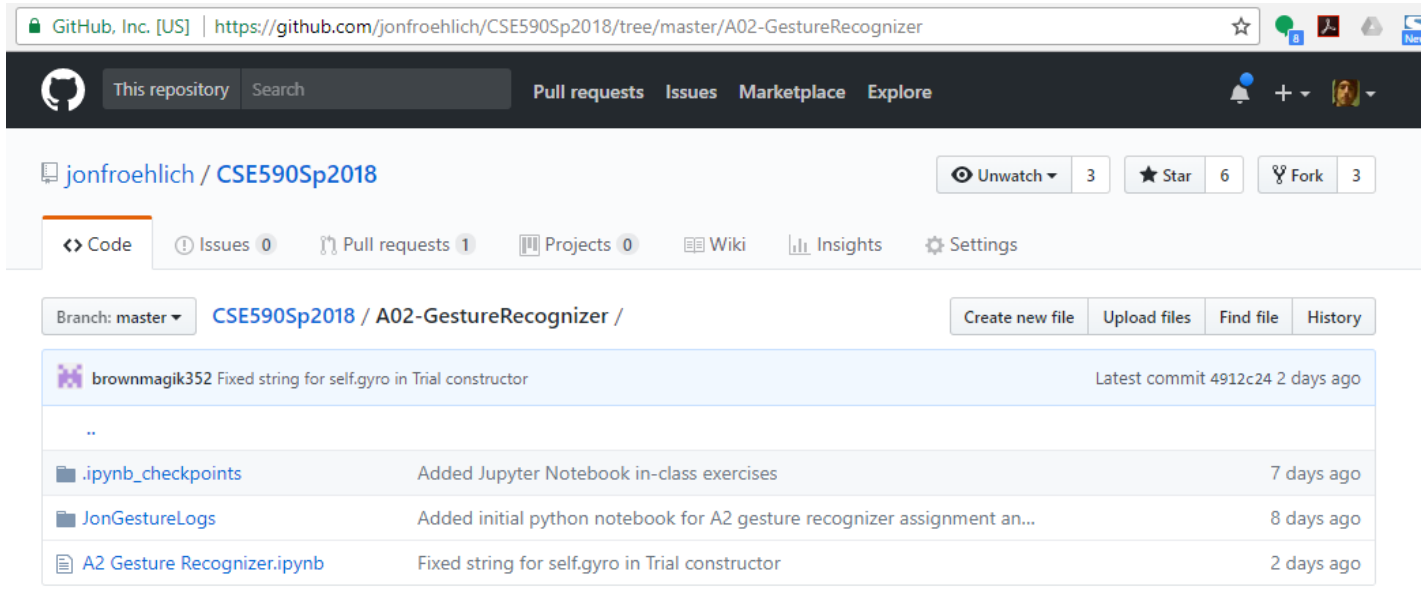
ML TRAINING

EXERCISE: BRAINSTORM & PLOT FEATURES

We are going to be doing exercises in Jupyter Notebook again.

Please clone/pull a fresh copy of the Android Gesture Recognizer.ipynb

Scroll to "**Model-Based ML Exercises**"



Feel free to work with a partner or alone!

EXERCISE: BRAINSTORM & PLOT FEATURES

What's the best 1D feature vector you can think of for recognizing gestures?

Length of signal

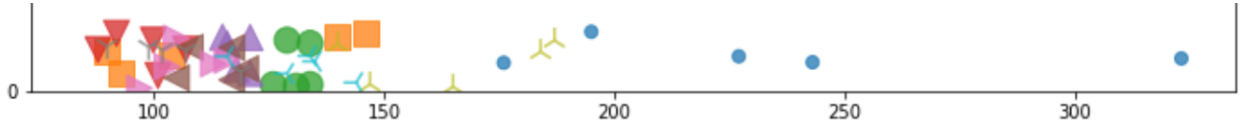
Max accel magnitude

Fundamental frequency of FFT

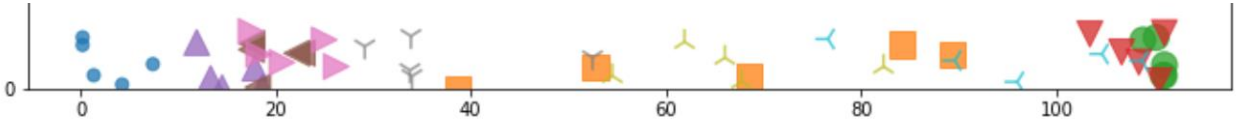
...

Plot your ideas on individual scatterplot graphs; color the gesture classes uniquely (you'll make a 2D scatter graph where the y-values are just small random perturbations to view your data better). We want to see **classes grouped together** and a **clear separation between them**.

1D Plot of Signal Length



1D Plot of Max Accel Mag

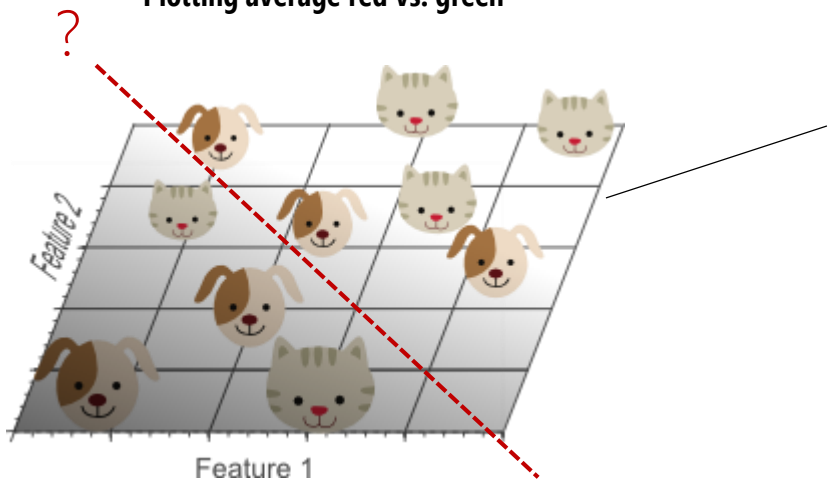


SELECTING FEATURES: HIGHER DIMENSIONALITY

Let's look at another example: this time trying to separate cats and dogs using average pixel color in image.

Plotting average red color in an image

Plotting average red vs. green



Is there any straight line that we can draw through our data that separates cats from dogs?

1D Feature Vector

Plotting average red color in image. Notice how a single feature does not result in a perfect separation of our training data.

2D Feature Vector

Plotting average red and green colors in image. Now we begin to see some more separation but not perfect.

Source: http://www.visiondummy.com/2014/04/curse-dimensionality-affect-classification/

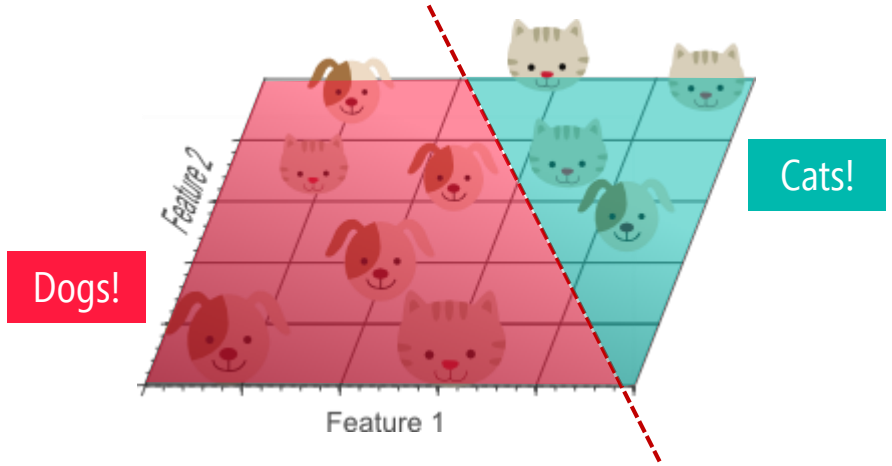
ML TRAINING

SELECTING FEATURES: HIGHER DIMENSIONALITY

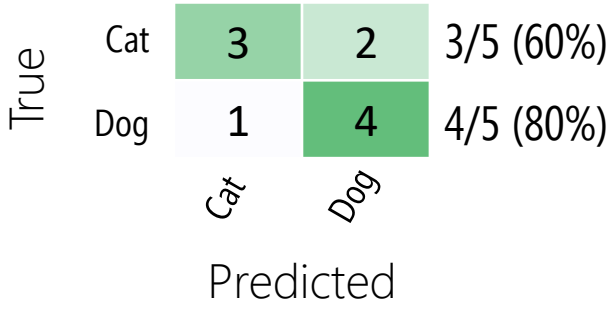
Let's look at another example: this time trying to separate cats and dogs using average pixel color in image.

Plotting average red color in an image

Plotting average red vs. green



Overall Accuracy: 7/10 (70%)



1D Feature Vector

Plotting average red color in image. Notice how a single feature does not result in a perfect separation of our training data.

2D Feature Vector

Plotting average red and green colors in image. Now we begin to see some more separation but not perfect.

Results

Overall accuracy is 70% with more cats being misclassified as dogs.

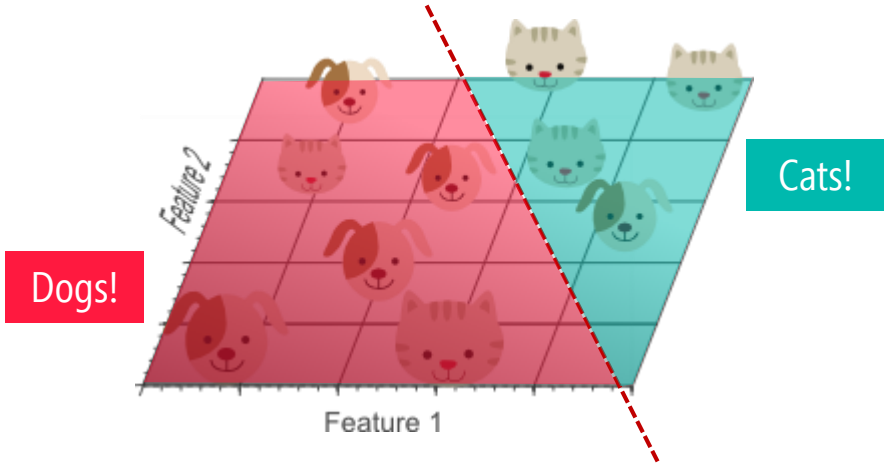
Source: http://www.visiondummy.com/2014/04/curse-dimensionality-affect-classification/

SELECTING FEATURES: HIGHER DIMENSIONALITY

Let's look at another example: this time trying to separate cats and dogs using average pixel color in image.

Plotting average red color in an image

Plotting average red vs. green



1D Feature Vector

Plotting average red color in image. Notice how a single feature does not result in a perfect separation of our training data.

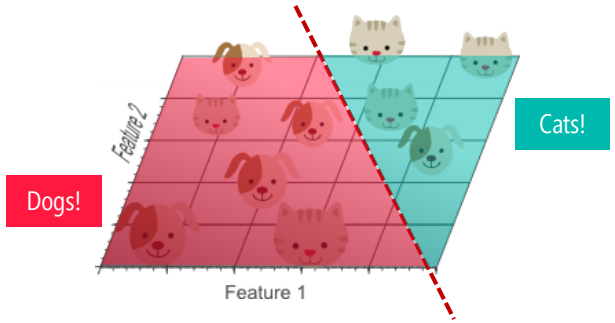
2D Feature Vector

Plotting average red and green colors in image. Now we begin to see some more separation but not perfect.

Source: http://www.visiondummy.com/2014/04/curse-dimensionality-affect-classification/

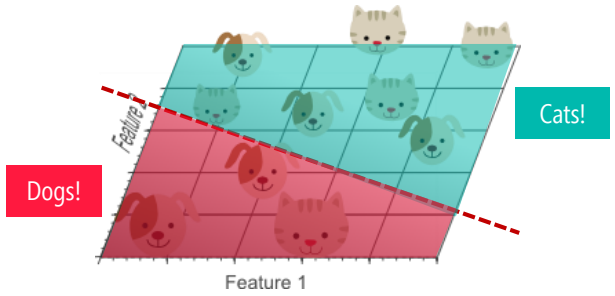
SELECTING FEATURES: HIGHER DIMENSIONALITY

And, of course, you can find different splits of the data. But we want to minimize error—an optimization problem—to find best split.



Overall Accuracy: 7/10 (70%)

True \ Predicted	Cat	Dog	
Cat	3	2	3/5 (60%)
Dog	1	4	4/5 (80%)
	Cat	Dog	



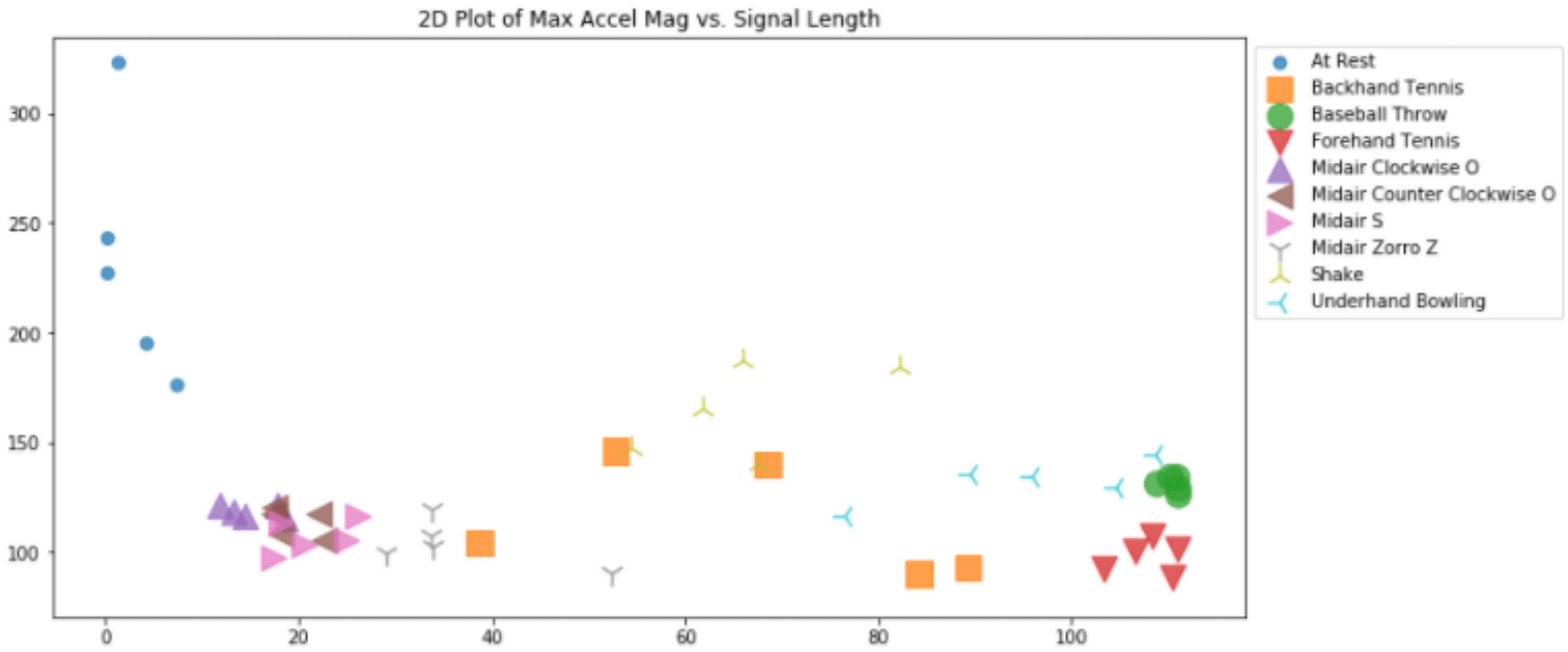
Overall Accuracy: 6/10 (60%)

True \ Predicted	Cat	Dog	
Cat	4	1	4/5 (80%)
Dog	3	2	2/5 (40%)
	Cat	Dog	

Source: http://www.visiondummy.com/2014/04/curse-dimensionality-affect-classification/

EXERCISE: PLOT FEATURES IN 2D SPACE

Now in a new cell, start creating 2D scatter plots by combining your 1D features (assign one feature to x, another to y and graph!).



SELECTING FEATURES: HIGHER DIMENSIONALITY

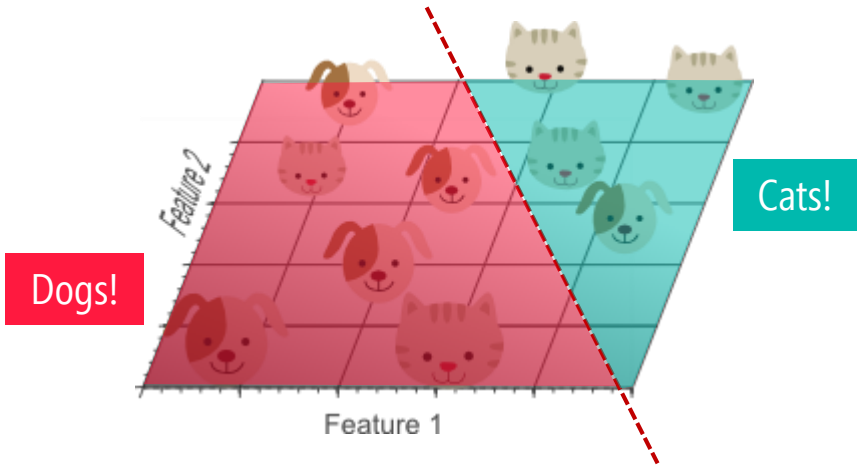
Let's look at another example: this time trying to separate cats and dogs using average pixel color in image.

Plotting average red color in an image

1D Feature Vector

Plotting average red color in image. Notice how a single feature does not result in a perfect separation of our training data.

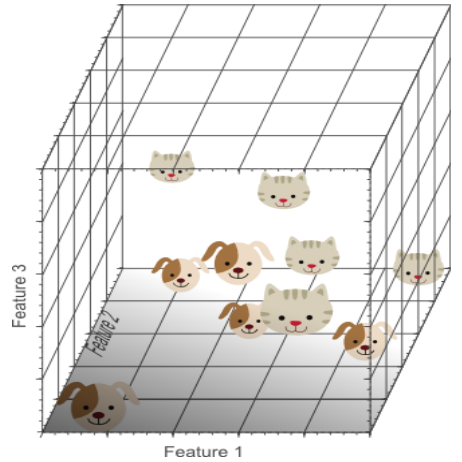
Plotting average red vs. green in image



2D Feature Vector

Plotting average red and green colors in image. Now we begin to see some more separation but not perfect.

Plotting average red, green, blue in an image



3D Feature Vector

Using a 3D-input feature vector (R,G,B), we begin to see more separation. However, this is harder to see in the graph.

Source: http://www.visiondummy.com/2014/04/curse-dimensionality-affect-classification/

ML TRAINING

SELECTING FEATURES: HIGHER DIMENSIONALITY

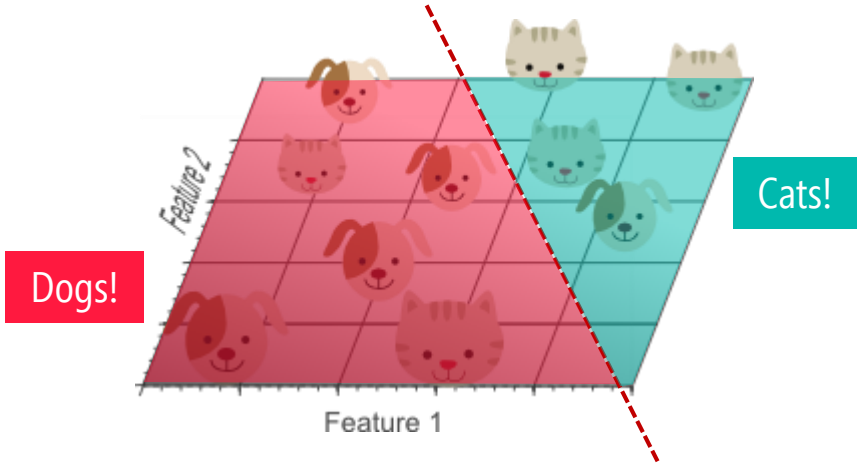
Let's look at another example: this time trying to separate cats and dogs using average pixel color in image.

Plotting average red color in an image

1D Feature Vector

Plotting average red color in image. Notice how a single feature does not result in a perfect separation of our training data.

Plotting average red vs. green in image

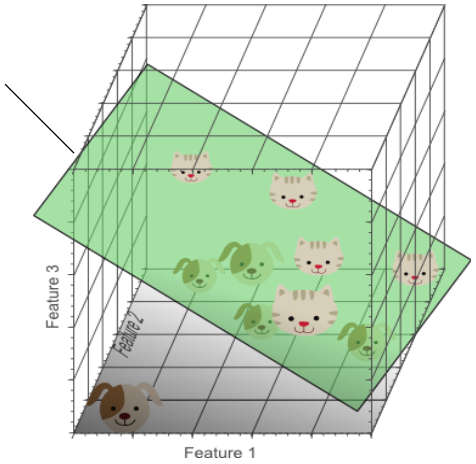


2D Feature Vector

Plotting average red and green colors in image. Now we begin to see some more separation but not perfect.

Plotting average red, green, blue in an image

Now, in a three-dimensional feature space, we can find a plane that perfectly splits dogs from cats.



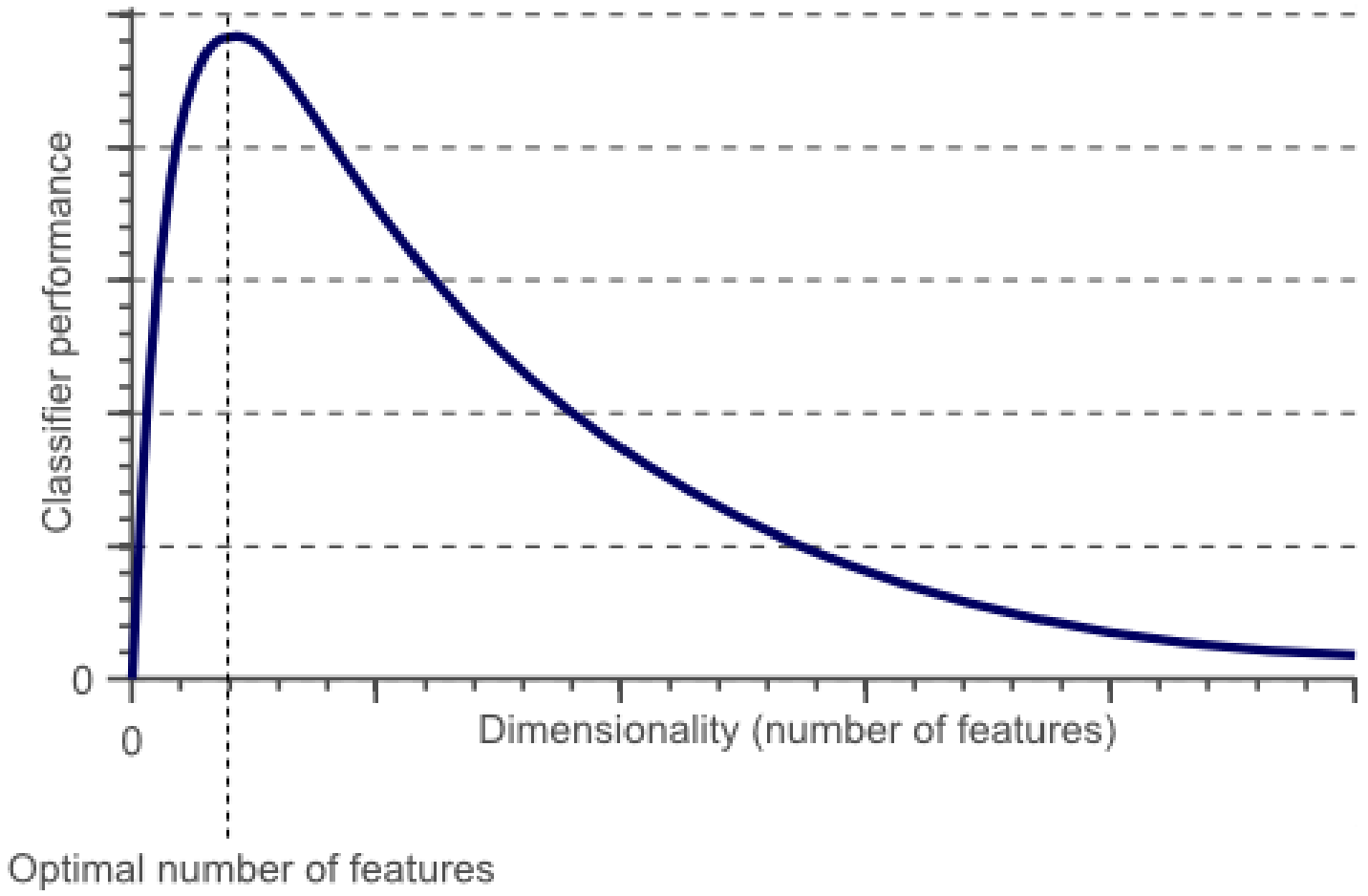
3D Feature Vector

Using a 3D-input feature vector (R,G,B), we begin to see more separation. (Of course, plotting in 3D often requires interactivity to tilt, rotate, explore data)

Source: http://www.visiondummy.com/2014/04/curse-dimensionality-affect-classification/

THE CURSE OF DIMENSIONALITY: OVERFITTING

You might think that higher dimensionality always results in a better classifier...



Source: http://www.visiondummy.com/2014/04/curse-dimensionality-affect-classification/

ML TRAINING

DIMENSIONALITY REDUCTION

Dimensionality reduction techniques transform feature vectors from high-dimensional space to fewer dimensions.

The most common technique is Principal Component Analysis, which performs a linear mapping to a lower-dimensional space such that the variance in the low-dimensional representation is maximized

e.g., You might find that those using HMMs in gesture recognition systems reduce high dimensional data using knn clustering or PCA

Source: https://en.wikipedia.org/wiki/Dimensionality_reduction

HOW DO WE SELECT FEATURES?

Based on **domain expertise** (*e.g.*, a medical doctor helping select features for automatic melanoma recognition in images)

Based on **intuitions, experimentations,** and **plotting the data** (*e.g.*, like we did in the in-class exercises)

Feature selection algorithms!

Source: http://www.visiondummy.com/2014/04/curse-dimensionality-affect-classification/

FEATURE SELECTION ALGORITHMS

Given a giant list of input features, a feature selection algorithm can be seen as a search technique for finding the best combination of feature subsets

Simplest approach: test each possible subset of features finding the one that minimizes the error rate. This is computationally intractable.

Sci-kit learn and other toolkits have support for this. Note: we are not expecting you to do this for A2

Source: https://en.wikipedia.org/wiki/Feature_selection

FEATURE SELECTION EXAMPLE

Classification Features for Touchscreen Typing

The above subsections defined three dimensions of adaptation for personalized touchscreen keyboards. Here we focus on *Dimension 1*, key-press classification models, and consider what features may be most useful for classification (Figure 4). Ultimately, the set of possible features will be determined by the input hardware used (*e.g.*, a vision system may provide more features than a capacitive touchscreen). While our focus is on ten-finger typing, many features listed here are also applicable to smaller devices. Our goal is to highlight the breadth of features that exist for personalizing keyboards as well as to introduce the particular features that we later test.

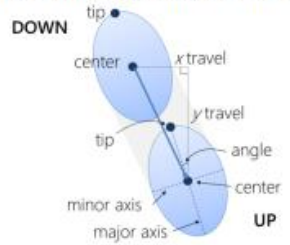


Figure 4. A subset of classification features based on a single key press event. This subset was tested in our simulations.

Touch location features are based on finger-down and finger-up locations. These features can include: (1) *finger center* (*x*,*y*) and (2) *fingertip* (*x*,*y*) locations. On large touchscreens, the keyboard can move, so these coordinates must be relative to some point(s) of reference. For adaptation to the left and right hands, *two* relative center locations can be calculated, *e.g.*, one as an offset from the F key and one as an offset from the J key (the *home* keys). Absolute values can be used for smaller screens.

Touch area features describe the size and angle of the finger on the screen; these features greatly depend on the hardware device. We used a Microsoft Surface, which provides ellipses fitted to the touch area and allows for (3) *major axis*, (4) *minor axis* and (5) *orientation*.

Movement features describe the movement of the finger during the key-press action. Different keys may result in different movement signatures based on the finger used and how the user reaches for that key. Specific features include (6) *travel* and (7) *elapsed time* between finger down and up. Travel can be calculated as the distance in *x* and *y* directions for either the fingertip or the finger center.

Features relative to previous key presses incorporate knowledge about previous key presses. These include (8) *inter-key-press timing*, or the elapsed time between key presses, and (9) (*x*,*y*) *distance offset from previous key*. The latter may be useful if, for example, the current key-press location is affected by whether the last key was to its left or right. A simulation study by Rashid and Smith [28] showed low accuracies for a touch model based only on this feature.

Hand or arm features are based on the position of the hand and/or arm. Again, these features are highly dependent on the underlying touchscreen system in that only certain systems can provide this information.

To select which subset of classification features to include in the J48 classification model, we used a wrapper-based feature selection technique [18]. That is, we incrementally added features to the model until no improvement in performance occurred. This process resulted in all features in Table 1 marked with a '*'. Additionally, we included the finger center *y* relative to the F and J keys for continuity, since the finger center *x* was already in the feature set; this inclusion did not reduce classification accuracy. With the

Feature Set	Classification Accuracy
Tip *x* relative to F and J keys*	53.3% (*SD* = 3.3)
Center *x* relative to F and J keys*	43.4% (*SD* = 6.7)
Tip *y* relative to F and J keys*	35.4% (*SD* = 2.0)
Center *y* relative to F and J keys*	29.7% (*SD* = 1.5)
Angle	28.0% (*SD* = 3.9)
Tip travel in *x* and *y* directions*	27.4% (*SD* = 1.9)
Center travel in *x* and *y* directions	24.8% (*SD* = 1.4)
Major axis	22.4% (*SD* = 1.5)
Minor axis	19.0% (*SD* = 1.9)
Elapsed time	14.2% (*SD* = 1.0)
Inter-key-press time	11.7% (*SD* = 2.3)
Majority classifier (right space key)	11.8%

* Used in feature set for final implementation.

Table 1. Explanatory power of classification features based on the pilot study data: mean 10-fold cross-validation results from a J48 classifier trained on individual features or pairs of related features. The majority classifier is the accuracy rate if we always classify as the most frequent key (right space key).

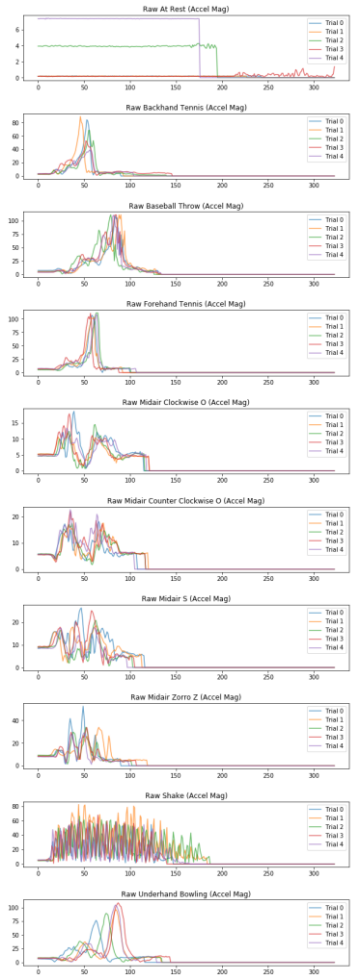
Where do we get our labeled training data?

ML TRAINING

THINKING ABOUT LABELING GESTURES IN A2

I labeled pre-segmented streams. What about a more "realistic" approach where you label continuous streams (*i.e.*, every datapoint has a label)

Name	Date modified
At Rest_Accelerometer_1523388944961.csv	4/10/2018 12:37 PM
At Rest_Accelerometer_1523388953684.csv	4/10/2018 12:37 PM
At Rest_Accelerometer_1523388964725.csv	4/10/2018 12:37 PM
At Rest_Accelerometer_1523388984213.csv	4/10/2018 12:37 PM
At Rest_Accelerometer_1523389002850.csv	4/10/2018 12:37 PM
At Rest_Gyroscope_1523388944961.csv	4/10/2018 12:37 PM
At Rest_Gyroscope_1523388953684.csv	4/10/2018 12:37 PM
At Rest_Gyroscope_1523388964725.csv	4/10/2018 12:37 PM
At Rest_Gyroscope_1523388984213.csv	4/10/2018 12:37 PM
At Rest_Gyroscope_1523389002850.csv	4/10/2018 12:37 PM
Backhand Tennis_Accelerometer_1523343191085.csv	4/9/2018 11:54 PM
Backhand Tennis_Accelerometer_1523343197400.csv	4/9/2018 11:54 PM
Backhand Tennis_Accelerometer_1523343205168.csv	4/9/2018 11:54 PM
Backhand Tennis_Accelerometer_1523343214494.csv	4/9/2018 11:54 PM
Backhand Tennis_Accelerometer_1523343221881.csv	4/9/2018 11:54 PM
Backhand Tennis_Gyroscope_1523343191085.csv	4/9/2018 11:54 PM
Backhand Tennis_Gyroscope_1523343197400.csv	4/9/2018 11:54 PM
Backhand Tennis_Gyroscope_1523343205168.csv	4/9/2018 11:54 PM
Backhand Tennis_Gyroscope_1523343214494.csv	4/9/2018 11:54 PM
Backhand Tennis_Gyroscope_1523343221881.csv	4/9/2018 11:54 PM
Baseball Throw_Accelerometer_1523388888168.csv	4/10/2018 12:37 PM
Baseball Throw_Accelerometer_1523388895884.csv	4/10/2018 12:37 PM
Baseball Throw_Accelerometer_1523388904126.csv	4/10/2018 12:37 PM
Baseball Throw_Accelerometer_1523388911971.csv	4/10/2018 12:37 PM
Baseball Throw_Accelerometer_1523388920465.csv	4/10/2018 12:37 PM
Baseball Throw_Gyroscope_1523388888168.csv	4/10/2018 12:37 PM
Baseball Throw_Gyroscope_1523388895884.csv	4/10/2018 12:37 PM
Baseball Throw_Gyroscope_1523388904126.csv	4/10/2018 12:37 PM
Baseball Throw_Gyroscope_1523388911971.csv	4/10/2018 12:37 PM
Baseball Throw_Gyroscope_1523388920465.csv	4/10/2018 12:37 PM

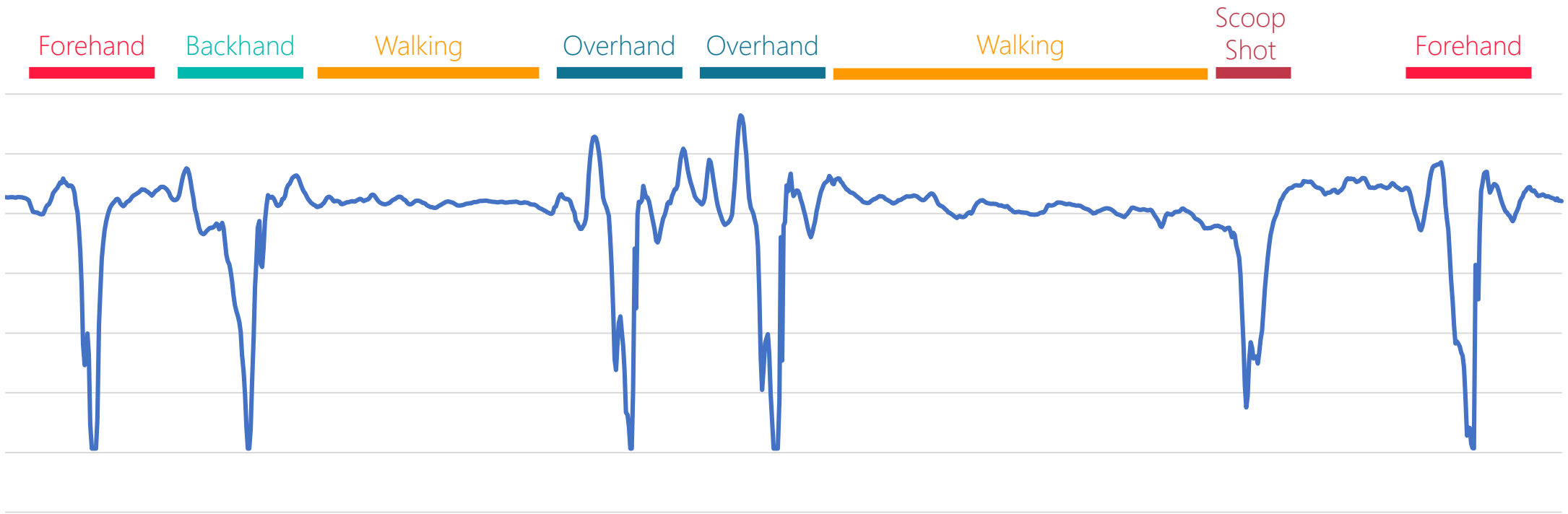


Gestures are pre-segmented

Labels are at gesture level
(rather than data point level)

GETTING TRAINING DATA

HOW SHOULD WE COLLECT & LABEL CONTINUOUS SENSOR STREAM DATA?

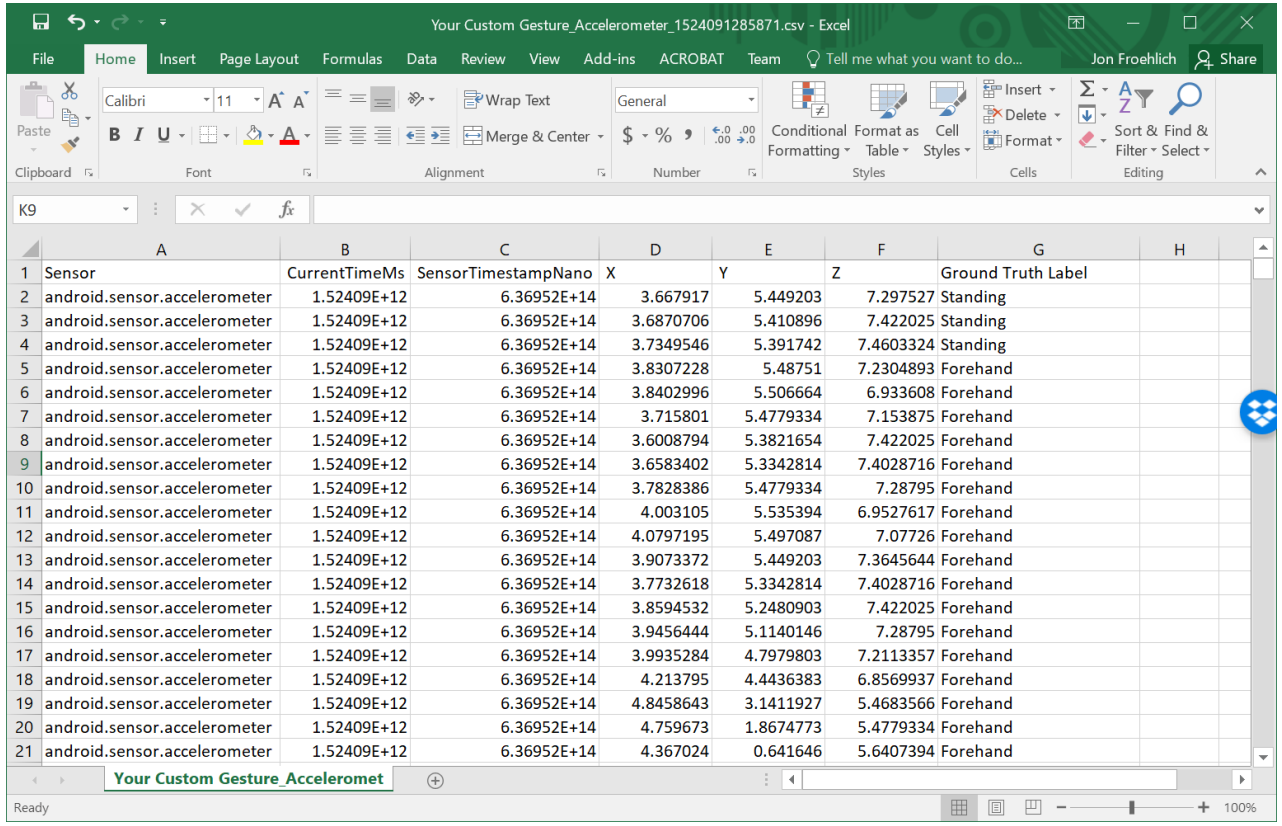


Accelerometer sensor stream when multiple gestures performed over time

Source: http://www.visiondummy.com/2014/04/curse-dimensionality-affect-classification/

HOW SHOULD WE COLLECT & LABEL CONTINUOUS SENSOR STREAM DATA?

Every timestamped record (*i.e.*, row) in gesture log is labeled with a ground truth activity label



	A	B	C	D	E	F	G	H
1	Sensor	CurrentTimeMs	SensorTimestampNano	X	Y	Z	Ground Truth Label	
2	android.sensor.accelerometer	1.52409E+12	6.36952E+14	3.667917	5.449203	7.297527	Standing	
3	android.sensor.accelerometer	1.52409E+12	6.36952E+14	3.6870706	5.410896	7.422025	Standing	
4	android.sensor.accelerometer	1.52409E+12	6.36952E+14	3.7349546	5.391742	7.4603324	Standing	
5	android.sensor.accelerometer	1.52409E+12	6.36952E+14	3.8307228	5.48751	7.2304893	Forehand	
6	android.sensor.accelerometer	1.52409E+12	6.36952E+14	3.8402996	5.506664	6.933608	Forehand	
7	android.sensor.accelerometer	1.52409E+12	6.36952E+14	3.715801	5.4779334	7.153875	Forehand	
8	android.sensor.accelerometer	1.52409E+12	6.36952E+14	3.6008794	5.3821654	7.422025	Forehand	
9	android.sensor.accelerometer	1.52409E+12	6.36952E+14	3.6583402	5.3342814	7.4028716	Forehand	
10	android.sensor.accelerometer	1.52409E+12	6.36952E+14	3.7828386	5.4779334	7.28795	Forehand	
11	android.sensor.accelerometer	1.52409E+12	6.36952E+14	4.003105	5.535394	6.9527617	Forehand	
12	android.sensor.accelerometer	1.52409E+12	6.36952E+14	4.0797195	5.497087	7.07726	Forehand	
13	android.sensor.accelerometer	1.52409E+12	6.36952E+14	3.9073372	5.449203	7.3645644	Forehand	
14	android.sensor.accelerometer	1.52409E+12	6.36952E+14	3.7732618	5.3342814	7.4028716	Forehand	
15	android.sensor.accelerometer	1.52409E+12	6.36952E+14	3.8594532	5.2480903	7.422025	Forehand	
16	android.sensor.accelerometer	1.52409E+12	6.36952E+14	3.9456444	5.1140146	7.28795	Forehand	
17	android.sensor.accelerometer	1.52409E+12	6.36952E+14	3.9935284	4.7979803	7.2113357	Forehand	
18	android.sensor.accelerometer	1.52409E+12	6.36952E+14	4.213795	4.4436383	6.8569937	Forehand	
19	android.sensor.accelerometer	1.52409E+12	6.36952E+14	4.8458643	3.1411927	5.4683566	Forehand	
20	android.sensor.accelerometer	1.52409E+12	6.36952E+14	4.759673	1.8674773	5.4779334	Forehand	
21	android.sensor.accelerometer	1.52409E+12	6.36952E+14	4.367024	0.641646	5.6407394	Forehand	

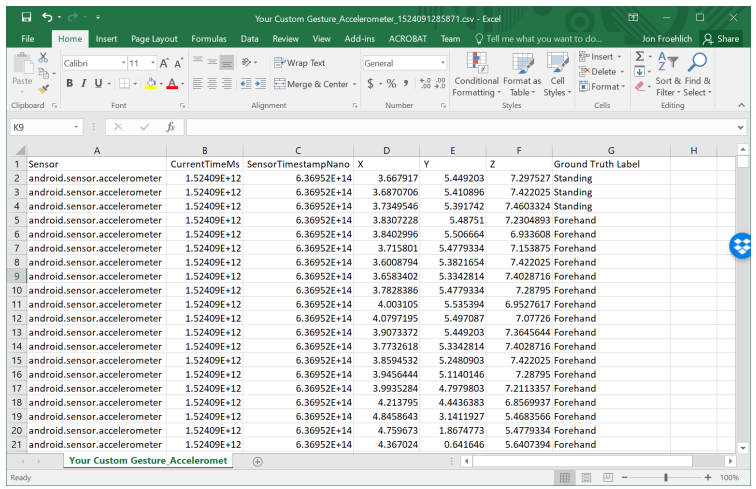
Think/Pair/Share

How would you design a study and analysis approach to get this granular ground truth data?

Source: http://www.visiondummy.com/2014/04/curse-dimensionality-affect-classification/

HOW SHOULD WE COLLECT & LABEL CONTINUOUS SENSOR STREAM DATA?

Every timestamped record (*i.e.*, row) in gesture log is labeled with a ground truth activity label



	Sensor	CurrentTimeMs	SensorTimestampNano	X	Y	Z	Ground Truth Label
2	android.sensor.accelerometer	1.52409E+12	6.36952E+14	3.667917	5.449203	7.297527	Standing
3	android.sensor.accelerometer	1.52409E+12	6.36952E+14	3.6870706	5.410896	7.422025	Standing
4	android.sensor.accelerometer	1.52409E+12	6.36952E+14	3.7349546	5.391742	7.4603324	Standing
5	android.sensor.accelerometer	1.52409E+12	6.36952E+14	3.8307228	5.48751	7.2304893	Forehand
6	android.sensor.accelerometer	1.52409E+12	6.36952E+14	3.8402996	5.506664	6.933608	Forehand
7	android.sensor.accelerometer	1.52409E+12	6.36952E+14	3.715801	5.4779334	7.153875	Forehand
8	android.sensor.accelerometer	1.52409E+12	6.36952E+14	3.6008794	5.3821654	7.422025	Forehand
9	android.sensor.accelerometer	1.52409E+12	6.36952E+14	3.6583402	5.3342814	7.4028716	Forehand
10	android.sensor.accelerometer	1.52409E+12	6.36952E+14	3.7828386	5.4779334	7.28795	Forehand
11	android.sensor.accelerometer	1.52409E+12	6.36952E+14	4.003105	5.535394	6.9527617	Forehand
12	android.sensor.accelerometer	1.52409E+12	6.36952E+14	4.0797195	5.497087	7.07726	Forehand
13	android.sensor.accelerometer	1.52409E+12	6.36952E+14	3.9073372	5.449203	7.3645644	Forehand
14	android.sensor.accelerometer	1.52409E+12	6.36952E+14	3.7732618	5.3342814	7.4028716	Forehand
15	android.sensor.accelerometer	1.52409E+12	6.36952E+14	3.8594532	5.2480903	7.422025	Forehand
16	android.sensor.accelerometer	1.52409E+12	6.36952E+14	3.9456444	5.1140146	7.28795	Forehand
17	android.sensor.accelerometer	1.52409E+12	6.36952E+14	3.9935284	4.7979803	7.2113357	Forehand
18	android.sensor.accelerometer	1.52409E+12	6.36952E+14	4.213795	4.4436383	6.8569937	Forehand
19	android.sensor.accelerometer	1.52409E+12	6.36952E+14	4.8458643	3.1411927	5.4683566	Forehand
20	android.sensor.accelerometer	1.52409E+12	6.36952E+14	4.759673	1.8674773	5.4779334	Forehand
21	android.sensor.accelerometer	1.52409E+12	6.36952E+14	4.367024	0.641646	5.6407394	Forehand

Some Potential Approaches

Have users perform a set of gestures in a prescribed order (follow a script)

Build an interactive system (like a game) that has user perform gestures in reaction to prompts

Record video of gesture study session and build a tool that aligns video with sensor streams then manually label data in tool

Have users wear motion capture suits and use Vicon motion tracking to automatically measure movement and label data

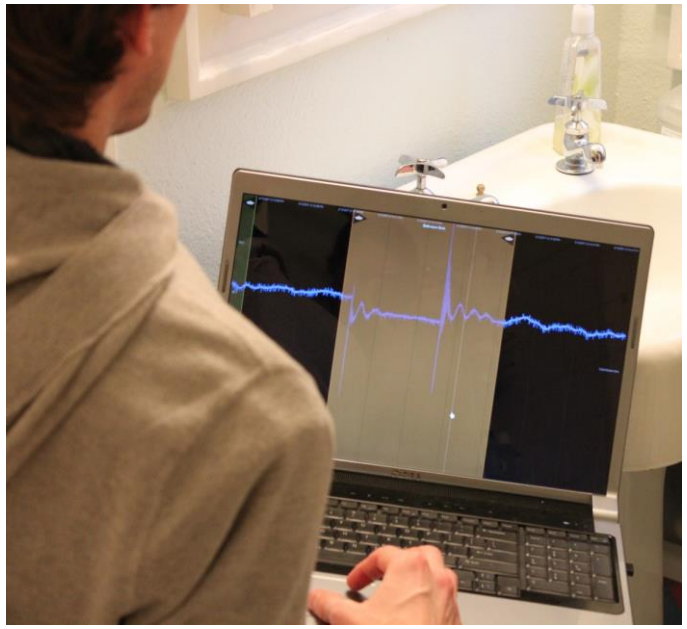
… other ideas?

Source: http://www.visiondummy.com/2014/04/curse-dimensionality-affect-classification/

GETTING TRAINING DATA

EXAMPLE FROM UBICOMP: HYDROSENSE

Evaluating HydroSense
Recall that HydroSense is a single-sensor based approach for automatically identifying water usage activities using pressure signatures.



Initial Study (UbiComp'09)
Proof-of-concept study used controlled water use trials that were manually labeled *in situ* using a custom tool.

Deployment Study (Pervasive'11)
Five-week real-world deployment study that analyzed real water usage data. Ground truth data was collected using custom embedded hardware kit.

Source: Froehlich *et al.*, UbiComp 2009; Froehlich *et al.*, Pervasive 2011

FROEHLICH ET AL., UBICOMP'09

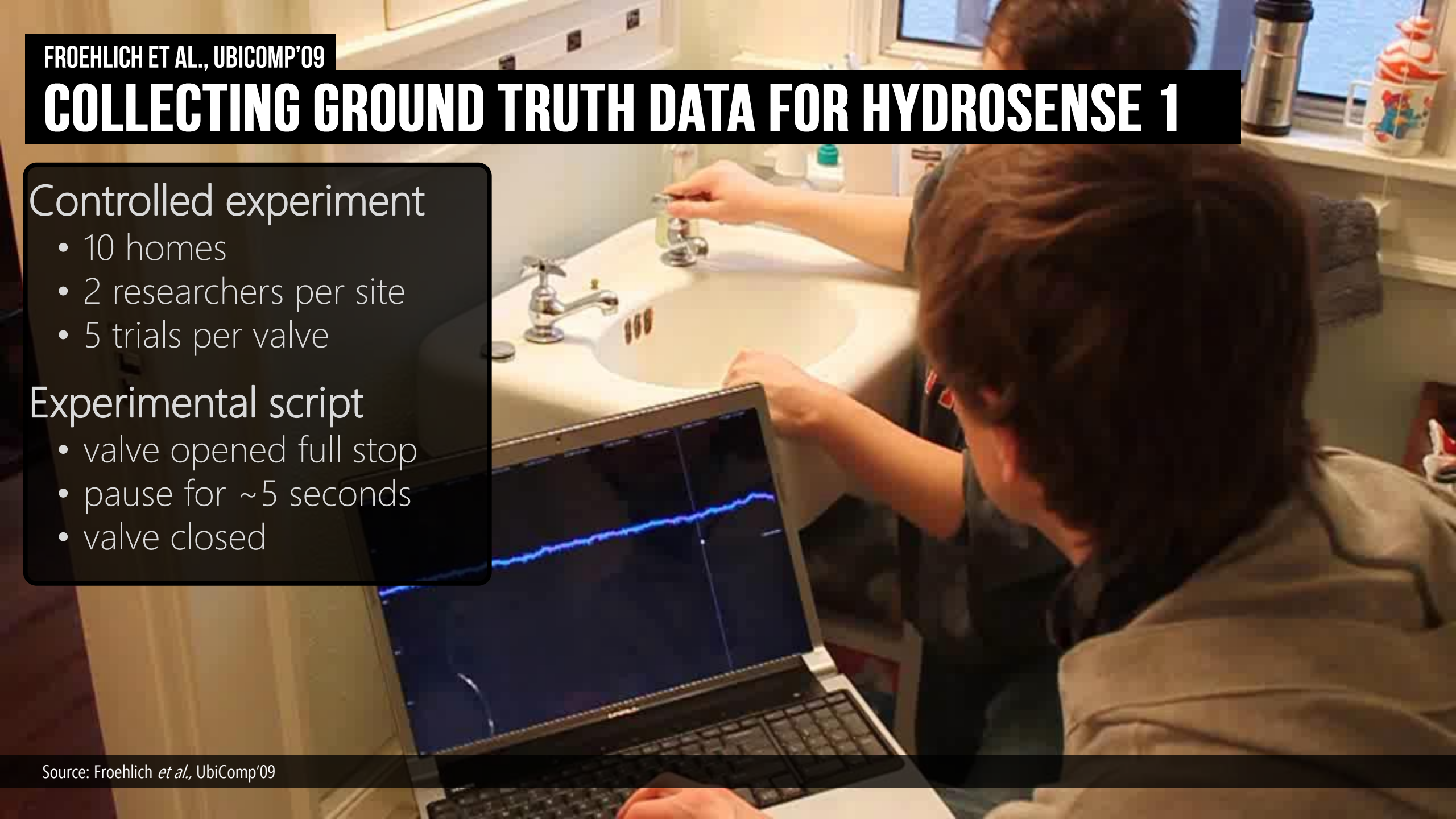
COLLECTING GROUND TRUTH DATA FOR HYDROSENSE 1

Controlled experiment

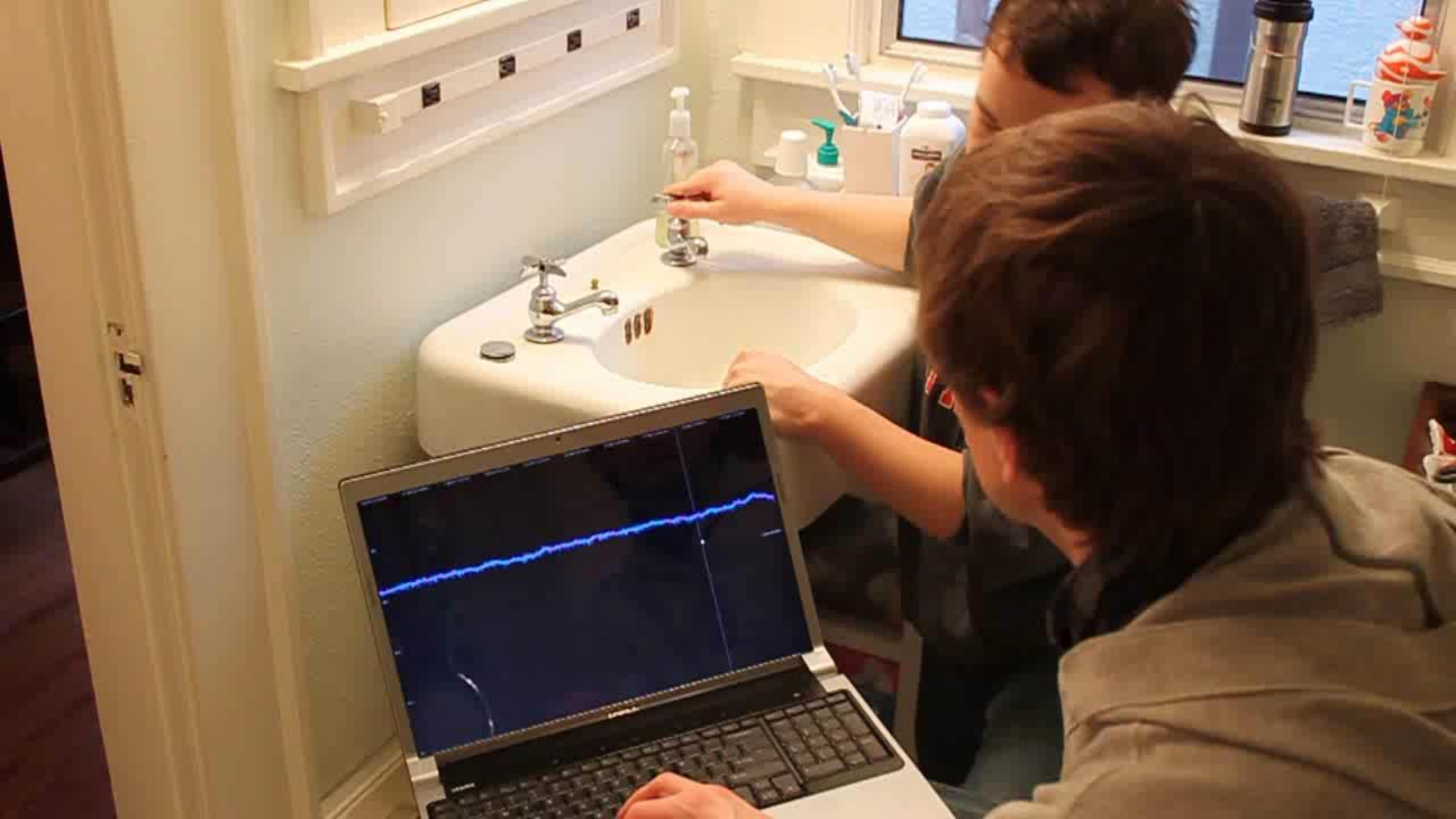
- 10 homes
- 2 researchers per site
- 5 trials per valve

Experimental script

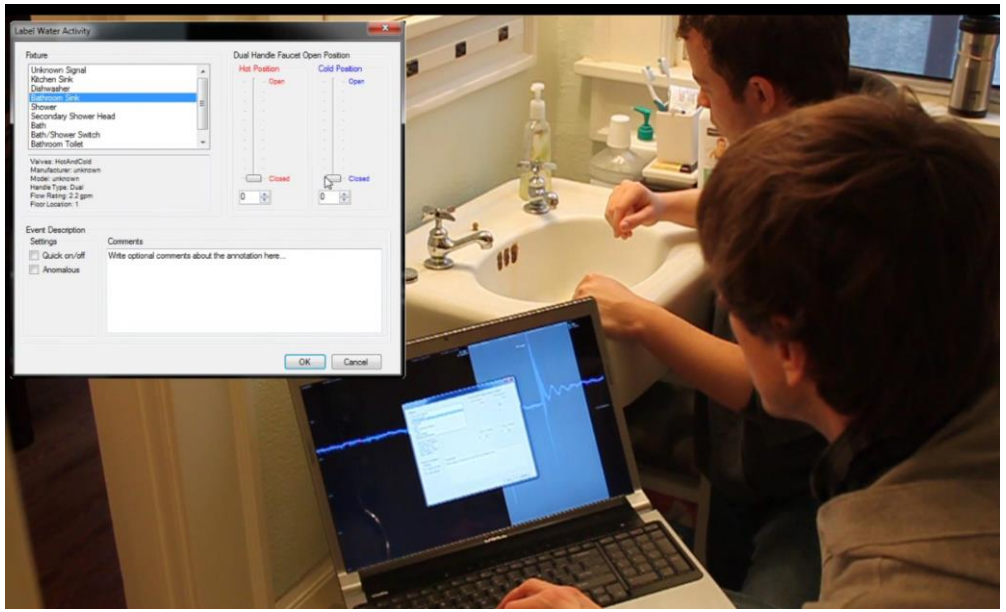
- valve opened full stop
- pause for ~5 seconds
- valve closed



Source: Froehlich *et al.*, UbiComp'09



HOW CAN WE COLLECT GROUND TRUTH LABELS DURING DEPLOYMENT?



How can we automate this?

How can we automate ground truth data collection so that we can get ground truth labels during our field deployment of HydroSense?

Towards automatic ground truth data collection

Hardware Capabilities

1. Wireless communication
2. Low-power
3. Water resistant

Sensing Capabilities

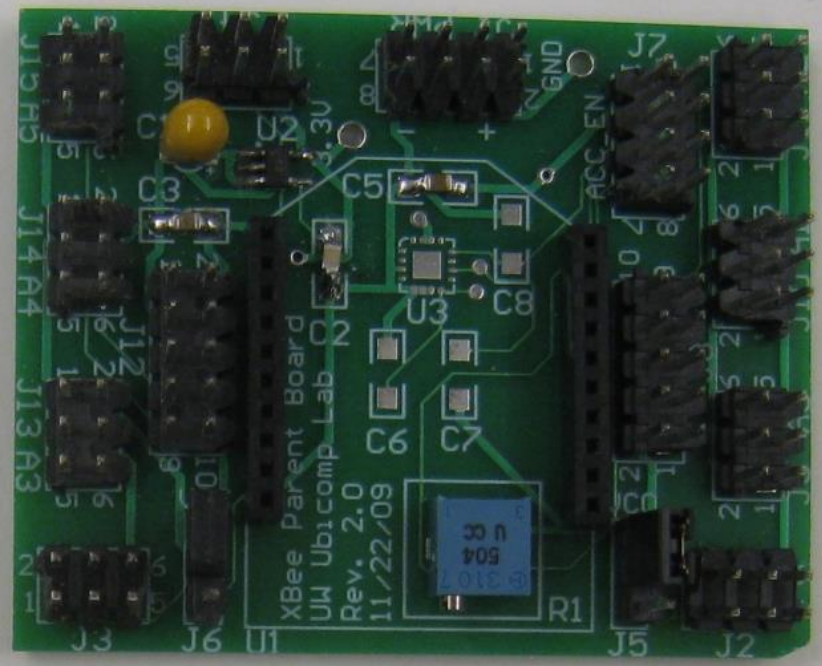
1. Work across fixtures/appliances
2. Detect opens/closes
3. Discriminate hot/cold/mixed

Source: Froehlich *et al.*, UbiComp 2009; Froehlich *et al.*, Pervasive 2011

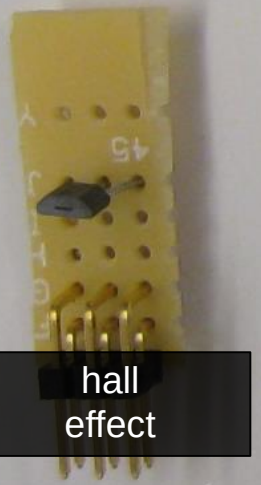
GETTING TRAINING DATA

CUSTOM GROUND TRUTH DATA COLLECTION KIT HYDROSENSE 2.0

xbee wireless modem



fixture usage sensor board



hall effect

reed switch

3-axis accelerometer

unidirectional ball switch

omnidirectional ball switch


accelerometer
Accelerometer
& Ball Switch
Taped on

GETTING TRAINING DATA

HYDROSENSE 2.0: 5-WEEK FIELD DEPLOYMENT

						Total
residents	2	2	4	2	2	12
size	3000 sqft	750 sqft	1200 sqft	700 sqft	750 sqft	N/A
floors	3	2	2	3rd flr	6th flr	N/A
fixtures	17	8	13	8	8	54
valves	28	13	21	13	13	88

GETTING TRAINING DATA

HYDROSENSE 2.0: INSTALLING GROUND TRUTH KIT

Sinks

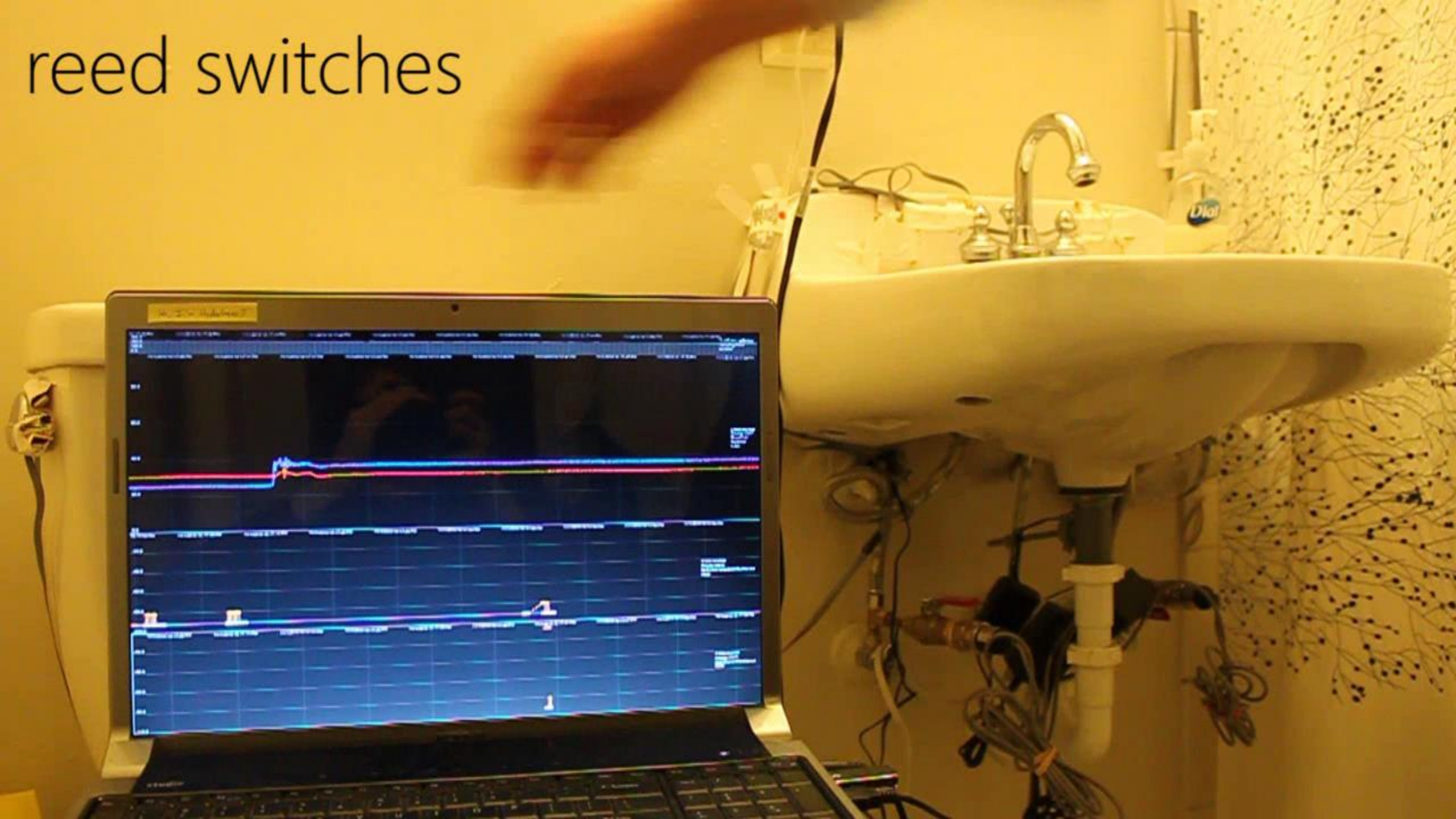
Showers/Baths

Toilets


PB32
XBee
KILL A WATT™
Thermistor to measure laundry cycle (*e.g.*, warm/cold)
TEMPERATURE
EXTRA RINSE
PULL KNOB TO START-PUSH TO STOP
WARM COLD
HOT COLD
COLD COLD
WARM WARM
OFF
SET
COTTONS REGULAR
PERMANENT PRESS KNITS
PERMANENT PRESS
DELICATES
LIGHT SOIL
MEDIUM
HEAVY
POWER WASH
30 MIN AUTO SOAK
= RINSE
= EXTRA RINSE
= SPIN
9 Cycle 3 Speed Combinations
Heavy Duty Super Capacity

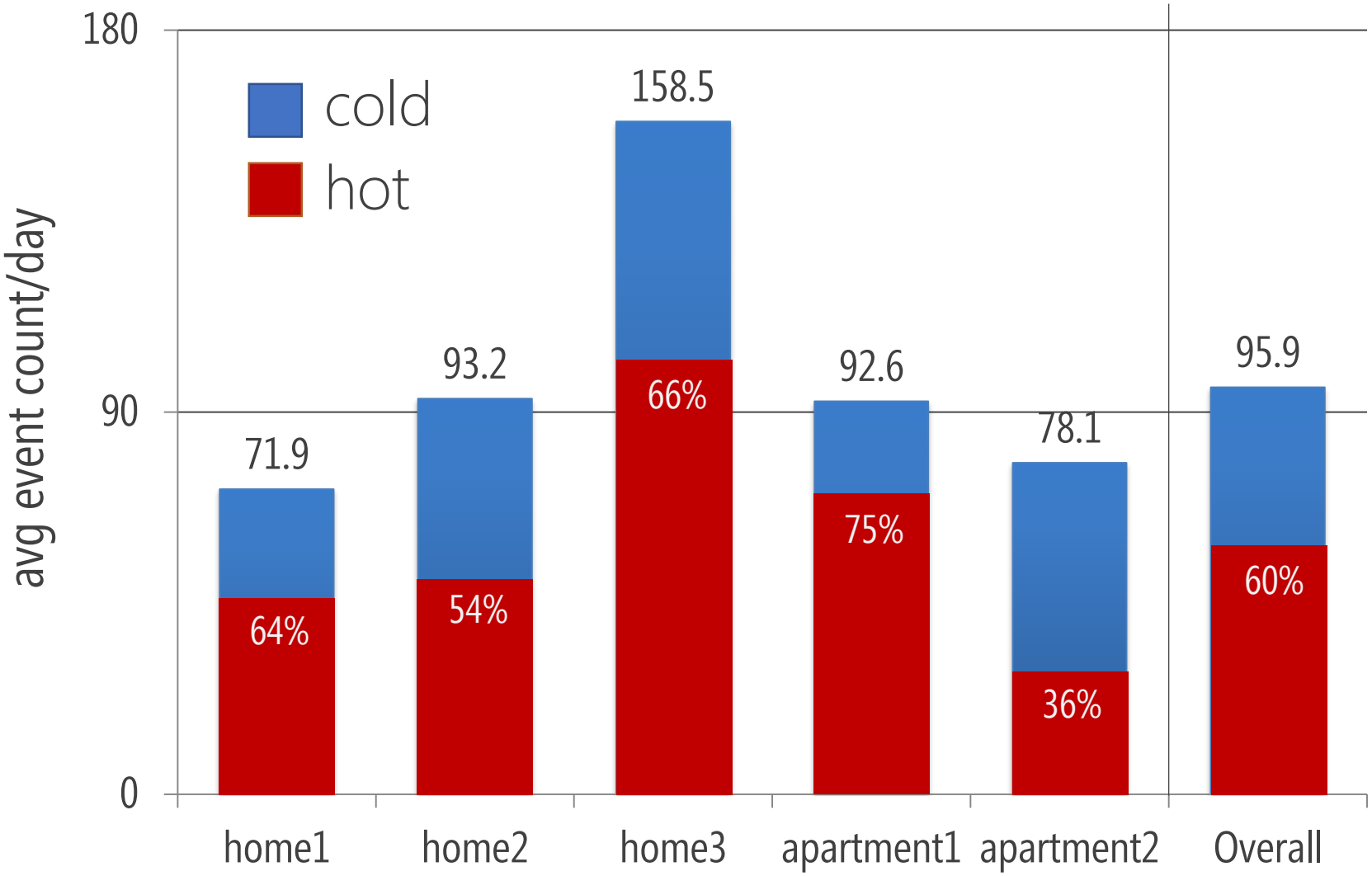

KitchenAid
CHILLED WATER

reed switches



HYDROSENSE 2.0 FIELD DEPLOYMENT RESULTS

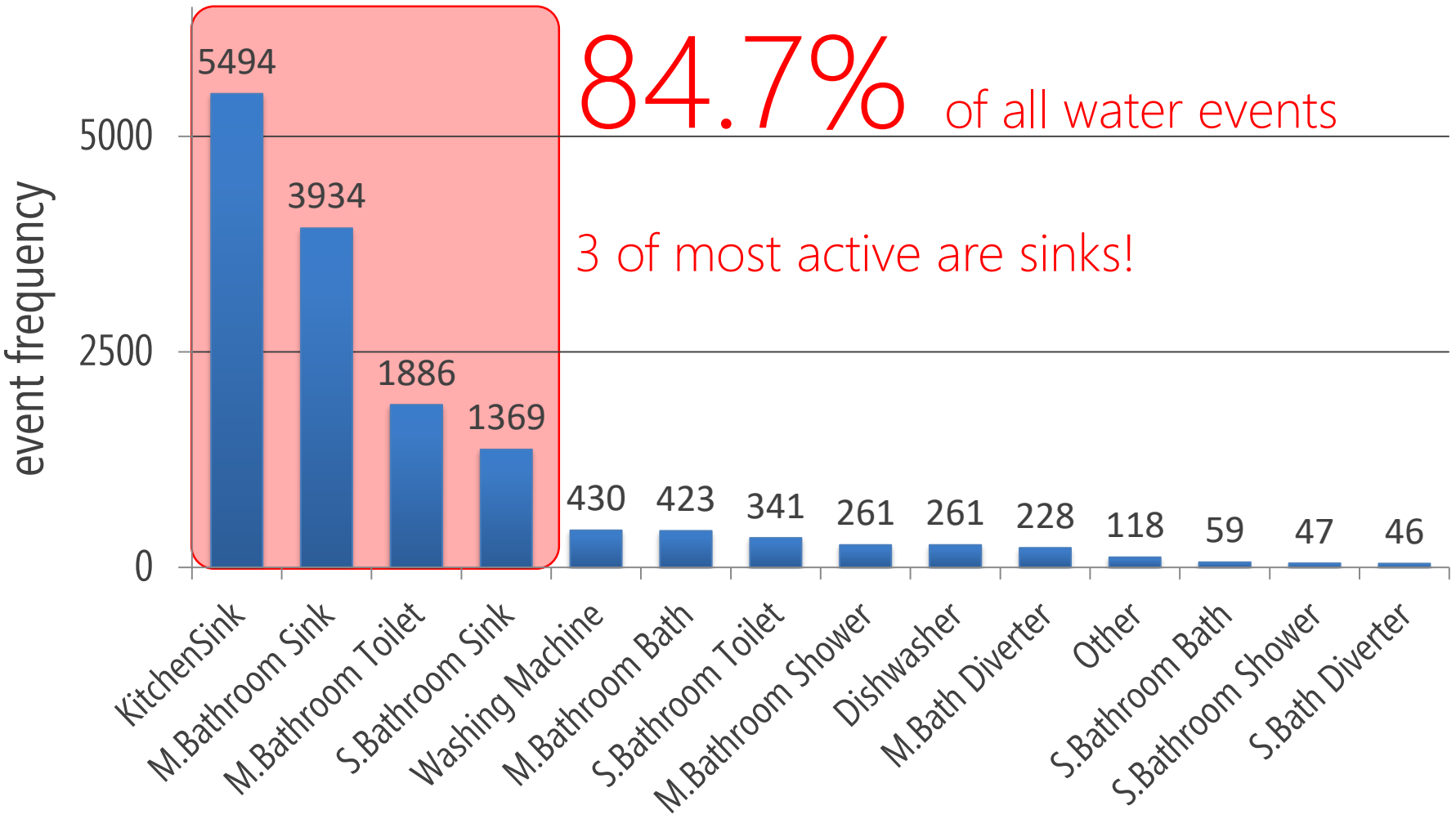
AVG WATER EVENTS/DAY



Source: Froehlich *et al.*, UbiComp 2009; Froehlich *et al.*, Pervasive 2011

HYDROSENSE 2.0 FIELD DEPLOYMENT RESULTS

FIXTURE ACTIVITY FREQUENCY



Source: Froehlich *et al.*, UbiComp 2009; Froehlich *et al.*, Pervasive 2011

HYDROSENSE 2.0 FIELD DEPLOYMENT RESULTS

HOW MANY EVENTS WERE COMPOUND?

How many events happened simultaneously with another water usage event?

22%

of all water events were compound

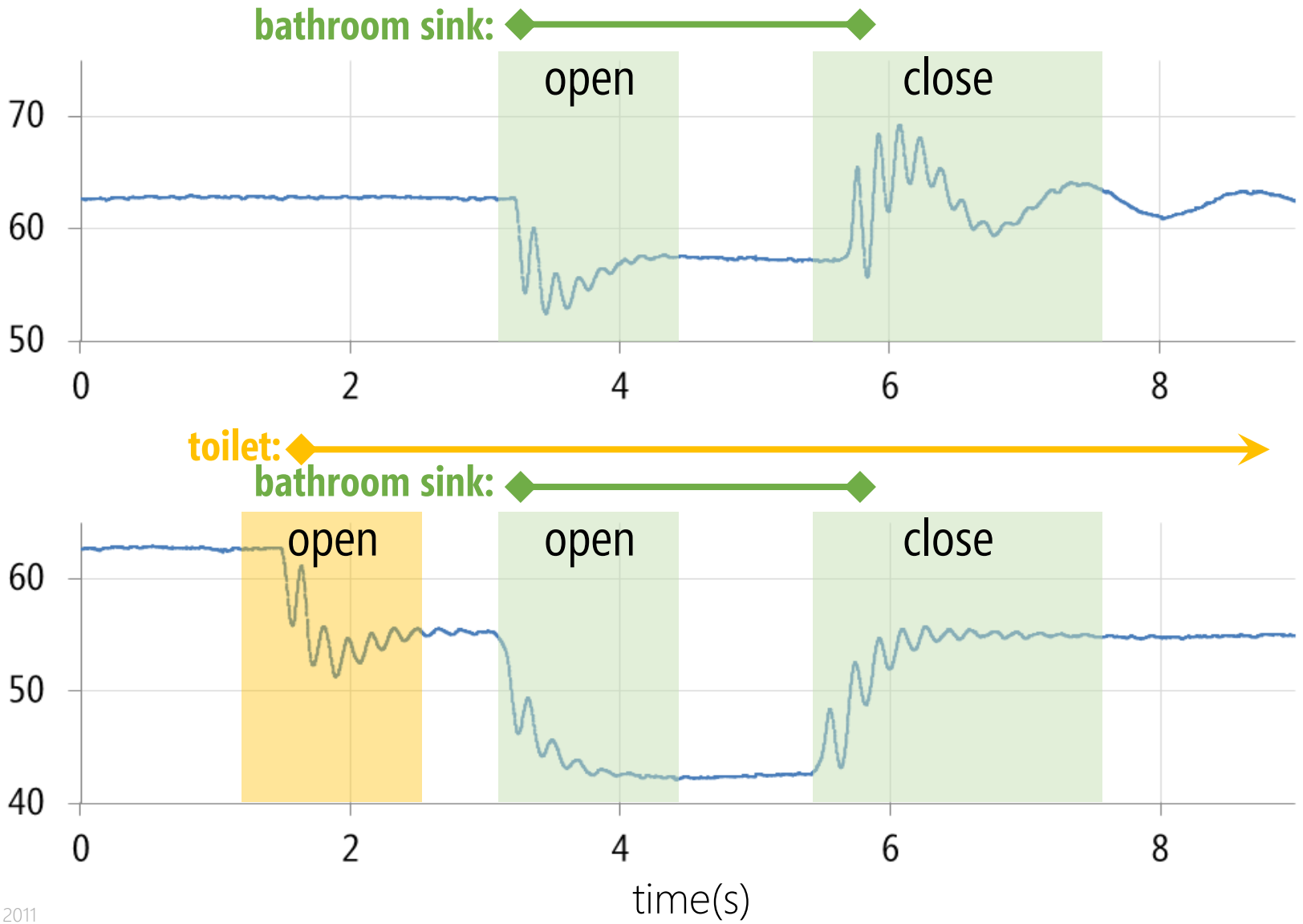
41.8%

of all bathroom sink events were compound

Source: Froehlich *et al.*, Pervasive 2011

HYDROSENSE 2.0 FIELD DEPLOYMENT RESULTS

COMPOUND EVENT EXAMPLE

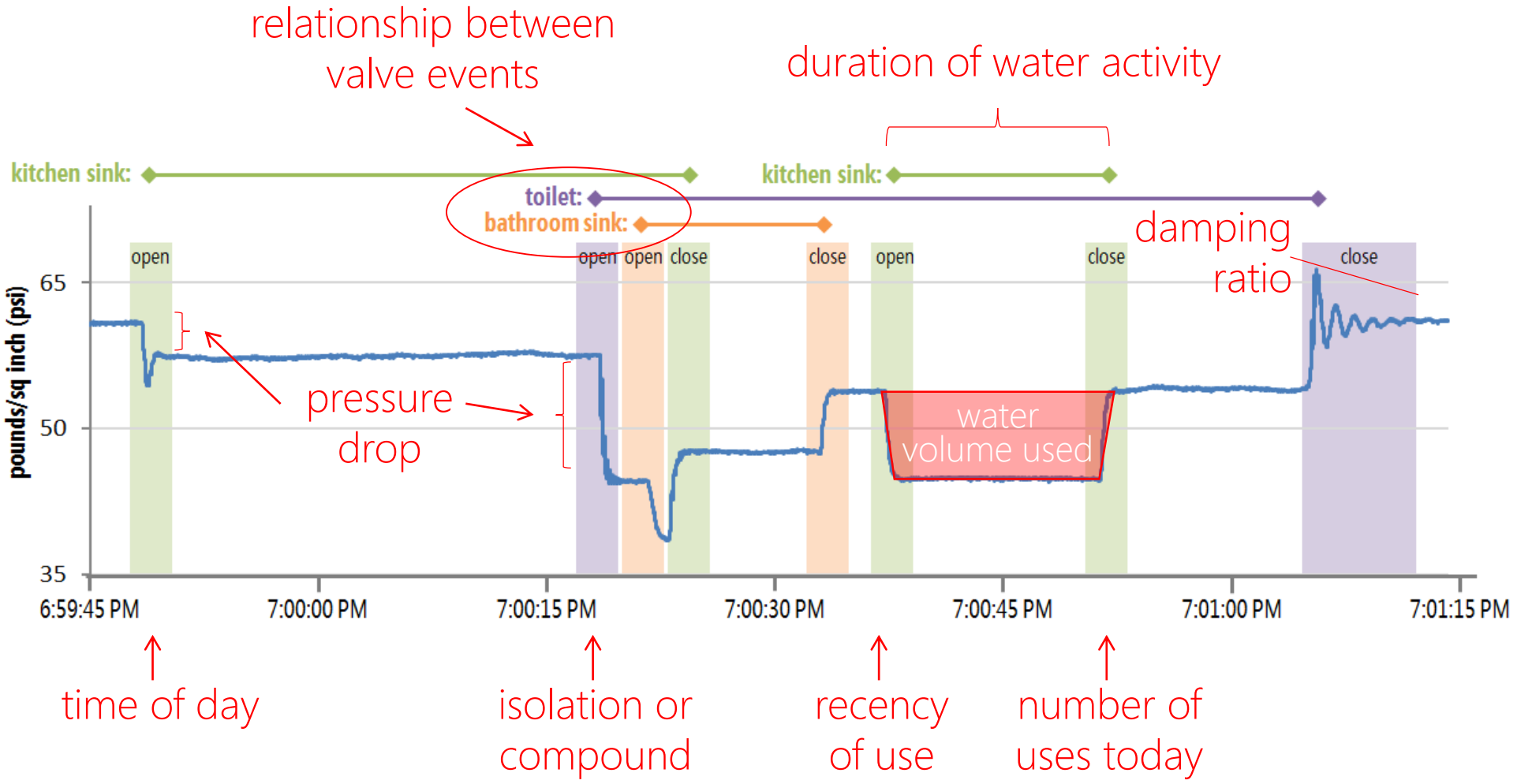


Source: Froehlich *et al.*, Pervasive 2011

HYDROSENSE 2.0 FIELD DEPLOYMENT RESULTS

MOVING BEYOND TEMPLATE MATCHING

There's lots of information beyond just the pressure signal that we weren't taking advantage of...



Source: Froehlich *et al.*, Pervasive 2011

bayesian approach

New algorithm borrows from Bayesian inference in speech recognition—a probabilistic approach (assumes each input feature contributes independent information)

signal

behavior

$$P(\mathbf{V}) = \underbrace{\prod_{r=0}^{R-1} f_r(\hat{\mathbf{S}}_r \mid \hat{\mathbf{V}}_r)}_{P(\mathbf{S}|\mathbf{V})} \underbrace{\prod_{n=0}^{N-1} P(v_n \mid v_{n-1}) \prod_{i \notin \beta} f_p(v_i) \prod_{k=0}^{K-1} \prod_{\langle a,b \rangle \in \beta} f_k(\langle v_a, v_b \rangle)}_{P(\mathbf{V})}$$

(i) templates and signal features

(ii) bigram language model

(iii) grammar

(iv) paired valve priors

bayesian approach

New algorithm borrows from Bayesian inference in speech recognition

signal

behavior

$$P(\mathbf{S}|\mathbf{V})\,P(\mathbf{V}) = \underbrace{\prod_{r=0}^{R-1} f_r(\hat{\mathbf{S}}_r \mid \hat{\mathbf{V}}_r)}_{\text{(i) templates and signal features}} \underbrace{\prod_{n=0}^{N-1} P(v_n \mid v_{n-1})}_{\text{(ii) bigram language model}} \underbrace{\prod_{i \notin \beta} f_p(v_i)}_{\text{(iii) grammar}} \underbrace{\prod_{k=0}^{K-1} \prod_{\langle a,b \rangle \in \beta} f_k(\langle v_a, v_b \rangle)}_{\text{(iv) paired valve priors}}$$

e.g., matched filtering and stabilized pressure drop

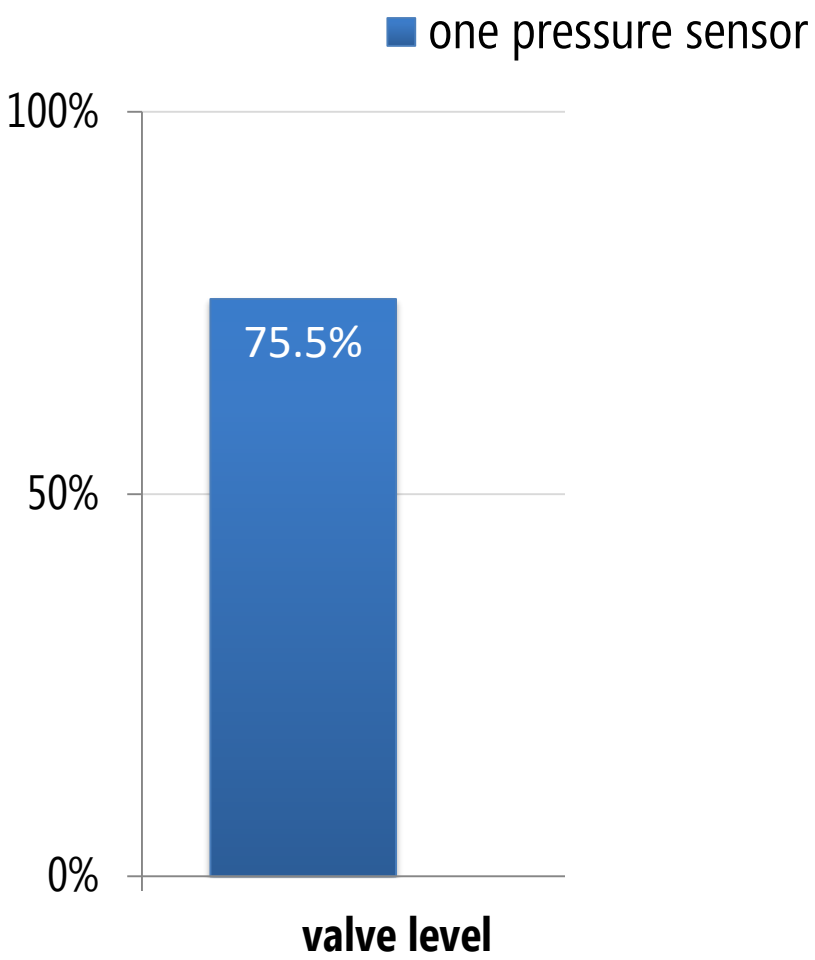
e.g., transition probability for toilet open->bathroom sink open

e.g., opening of valve v_x must be followed by closing of v_x

e.g., water usage duration

HYDROSENSE 2.0 FIELD DEPLOYMENT RESULTS

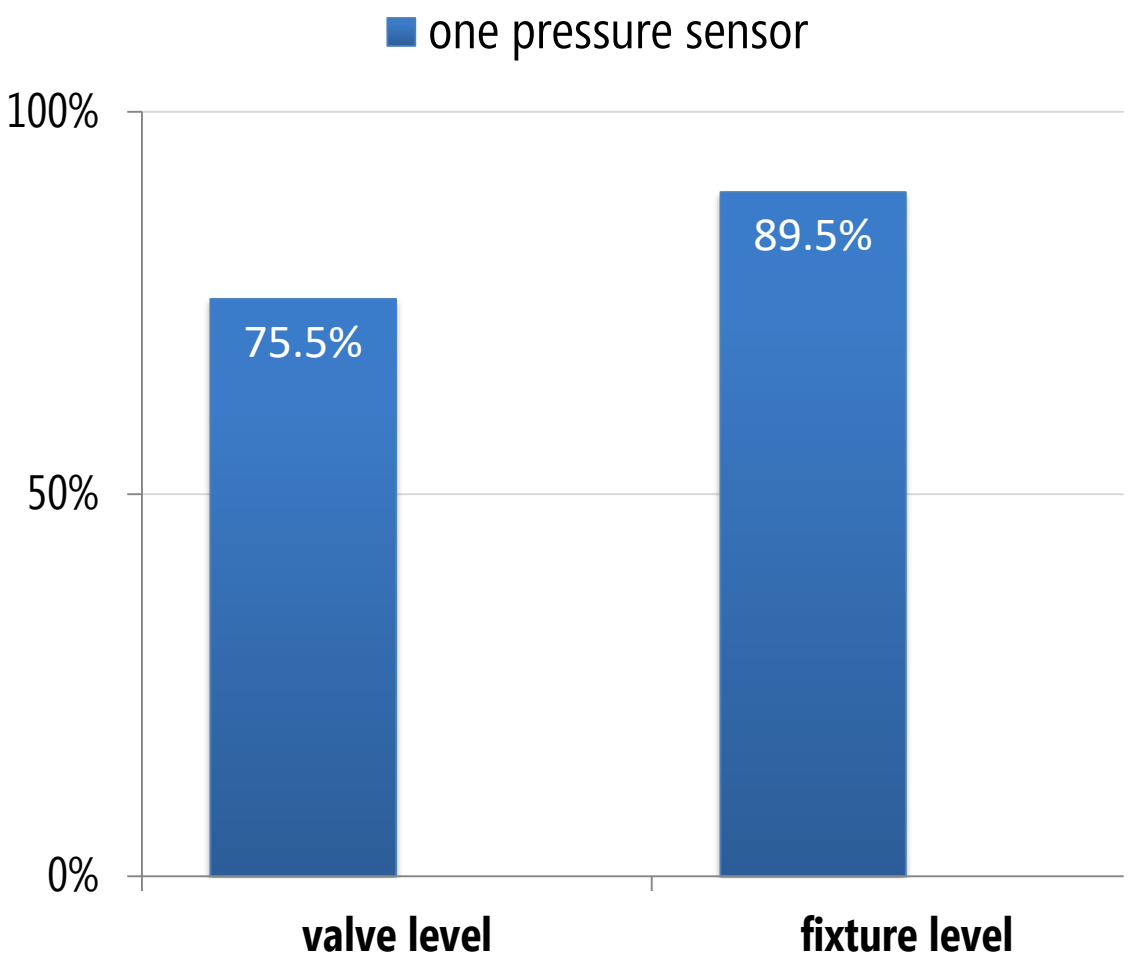
BAYESIAN CLASSIFIER RESULTS



Source: Froehlich *et al.*, Pervasive 2011

HYDROSENSE 2.0 FIELD DEPLOYMENT RESULTS

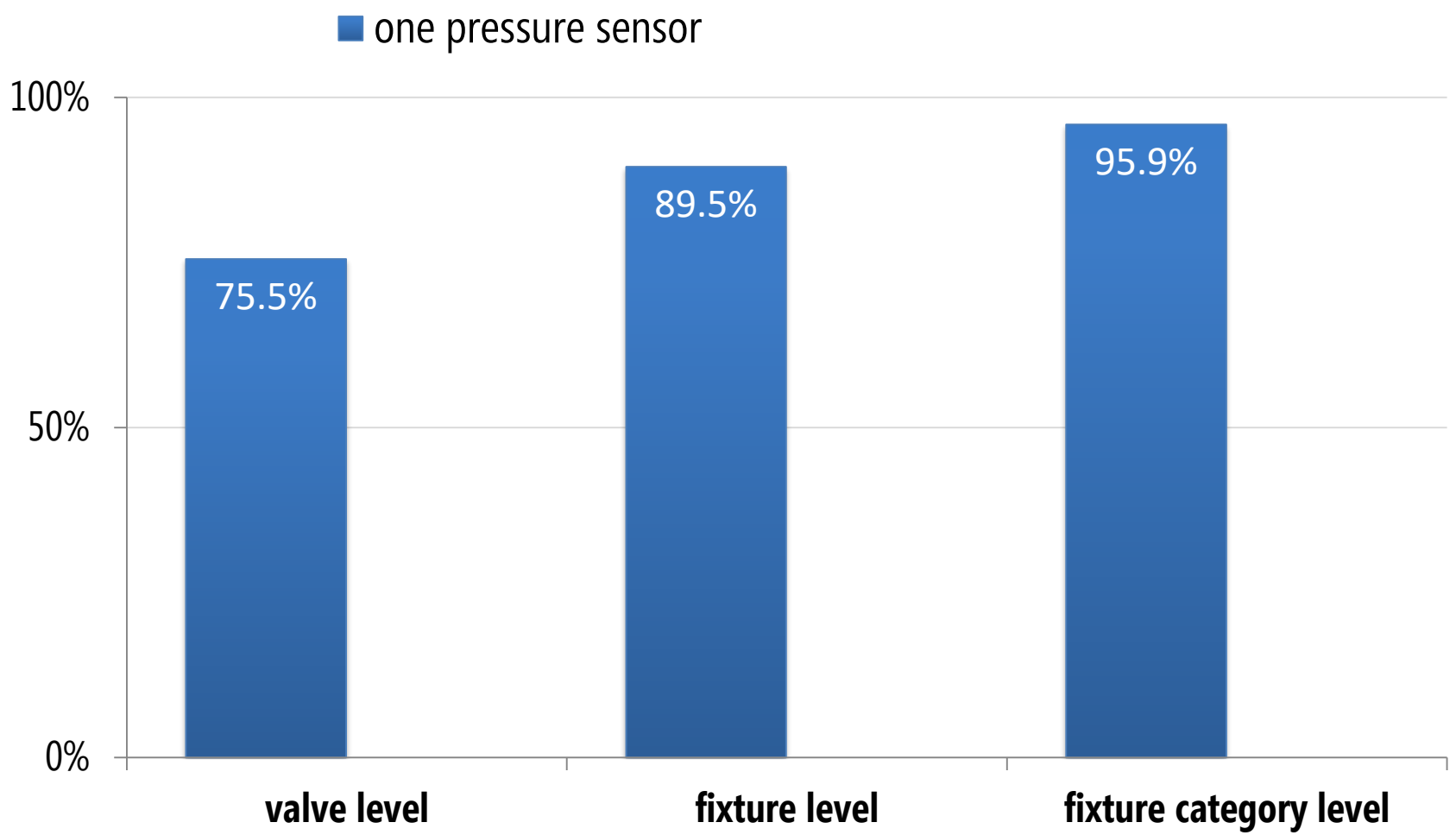
BAYESIAN CLASSIFIER RESULTS



Source: Froehlich *et al.*, Pervasive 2011

HYDROSENSE 2.0 FIELD DEPLOYMENT RESULTS

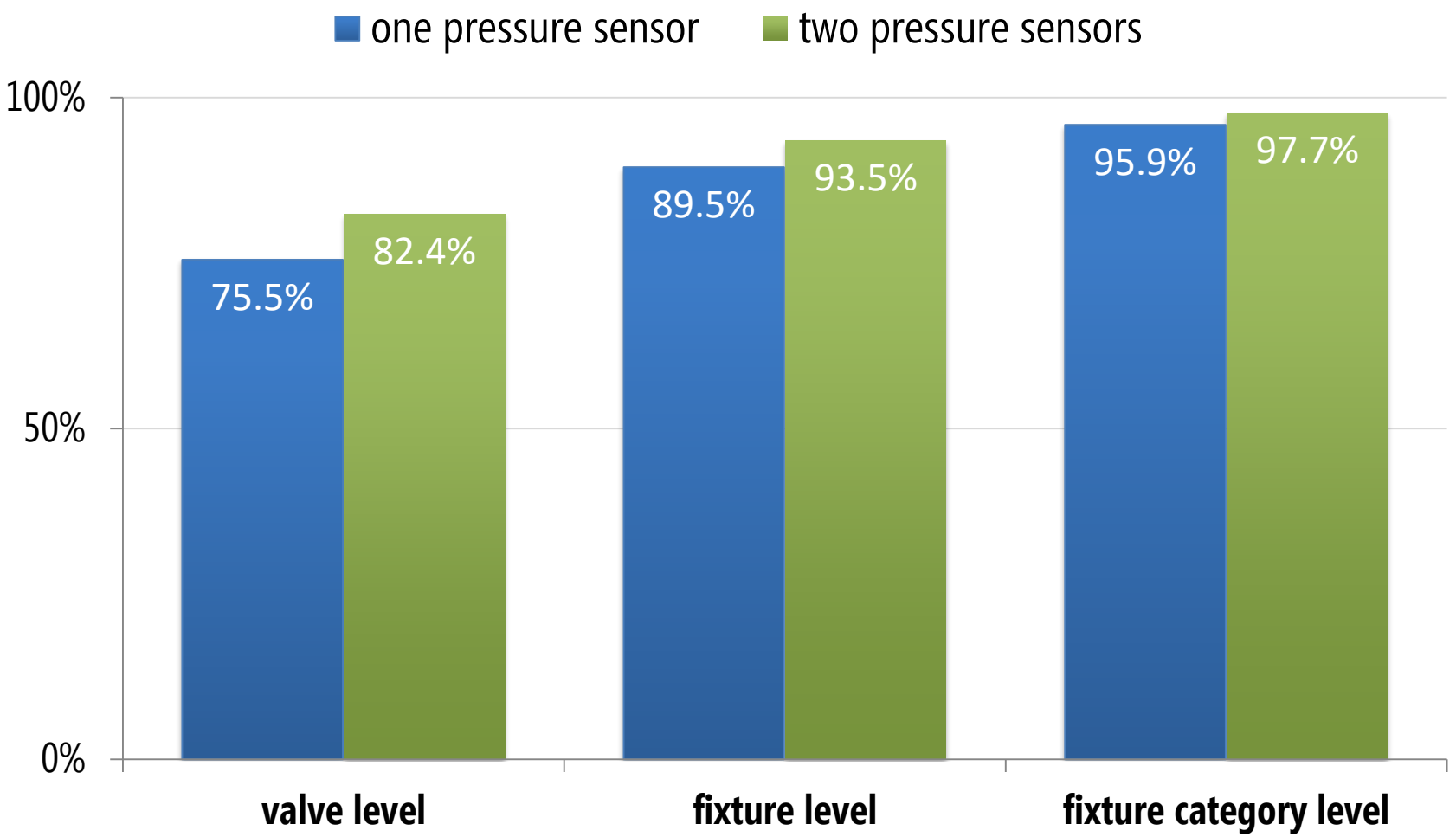
BAYESIAN CLASSIFIER RESULTS



Source: Froehlich *et al.*, Pervasive 2011

HYDROSENSE 2.0 FIELD DEPLOYMENT RESULTS

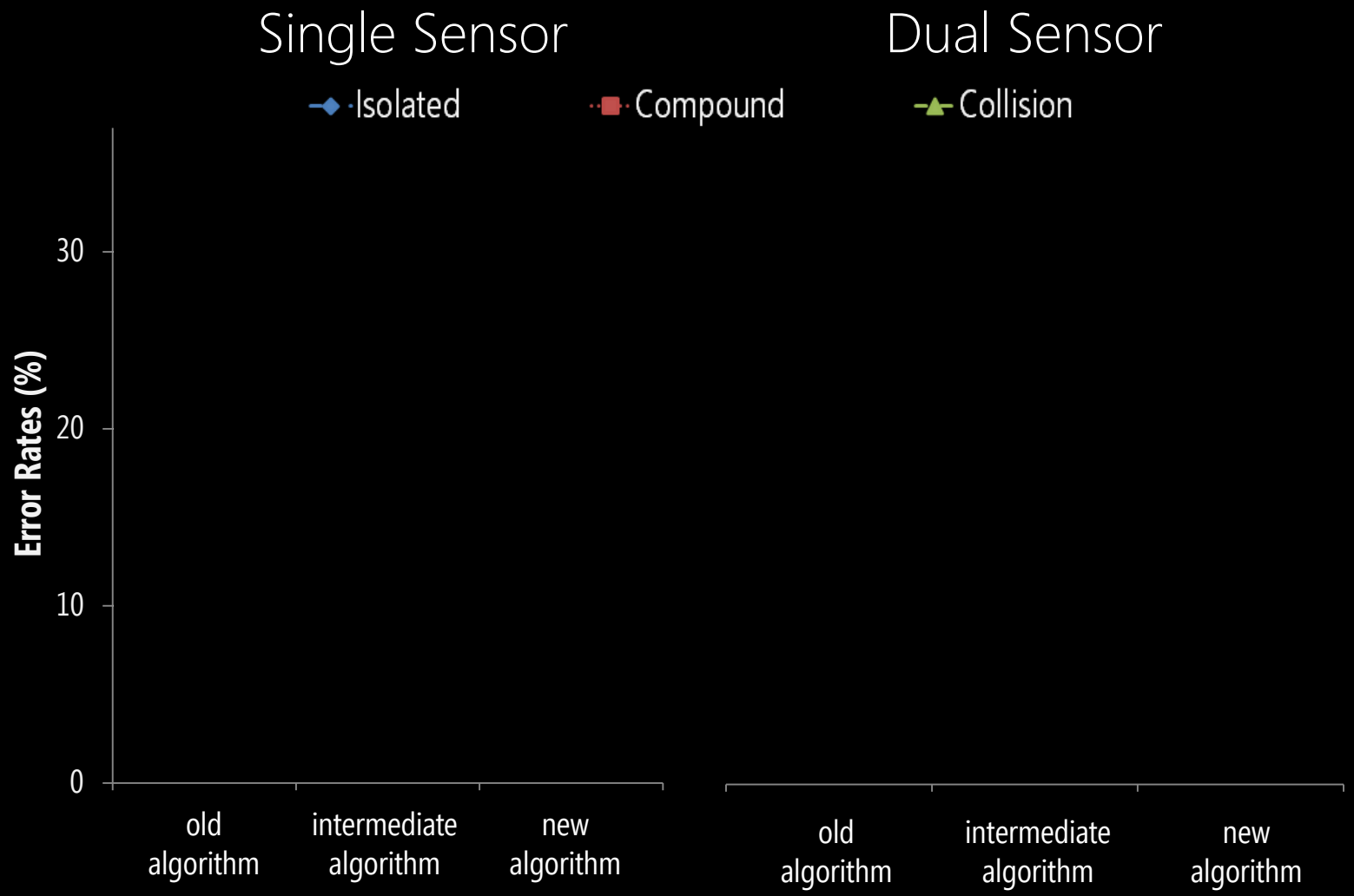
BAYESIAN CLASSIFIER RESULTS



Source: Froehlich *et al.*, Pervasive 2011

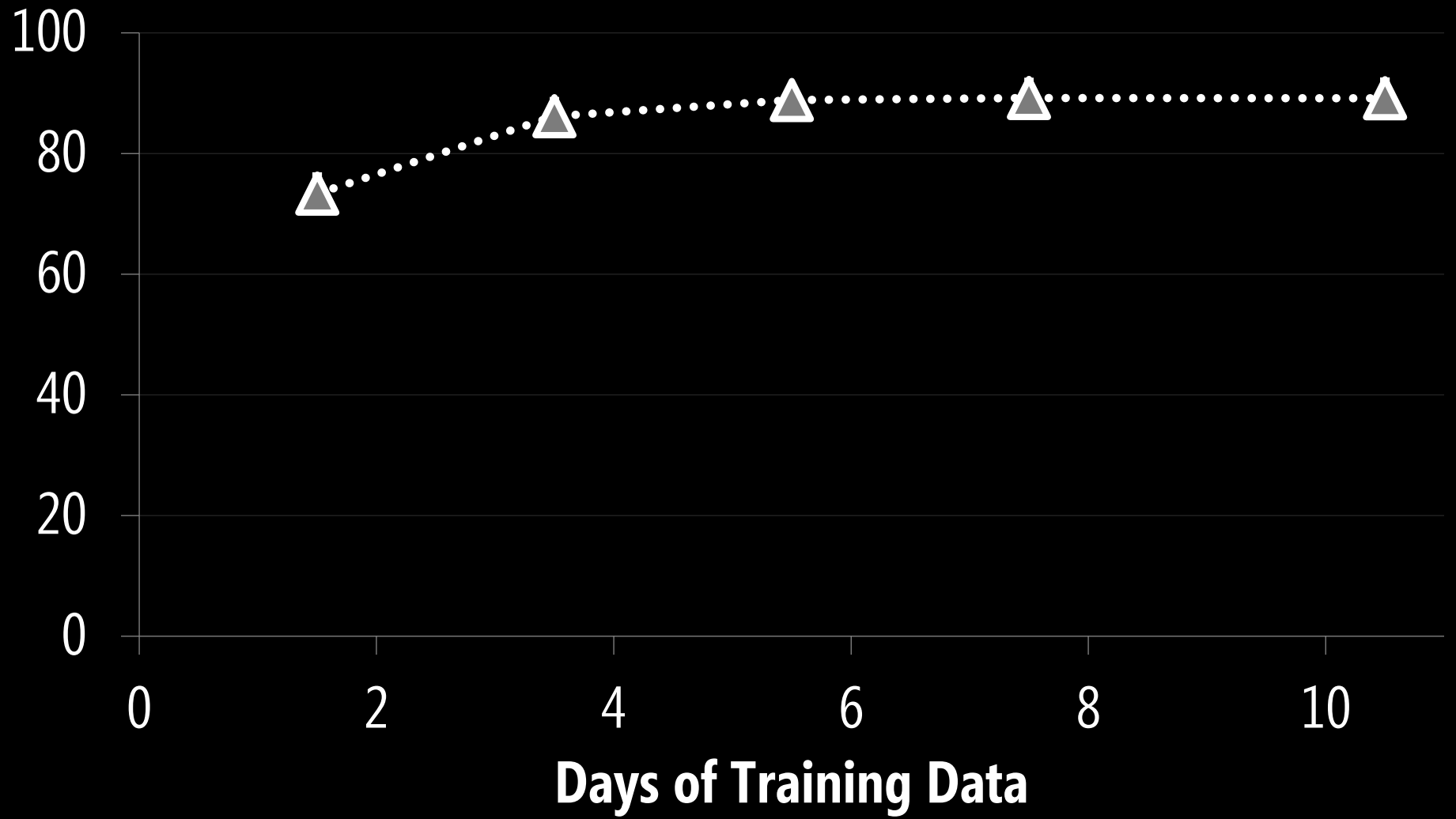
compound events results

real-world water usage data



hydrosense training results

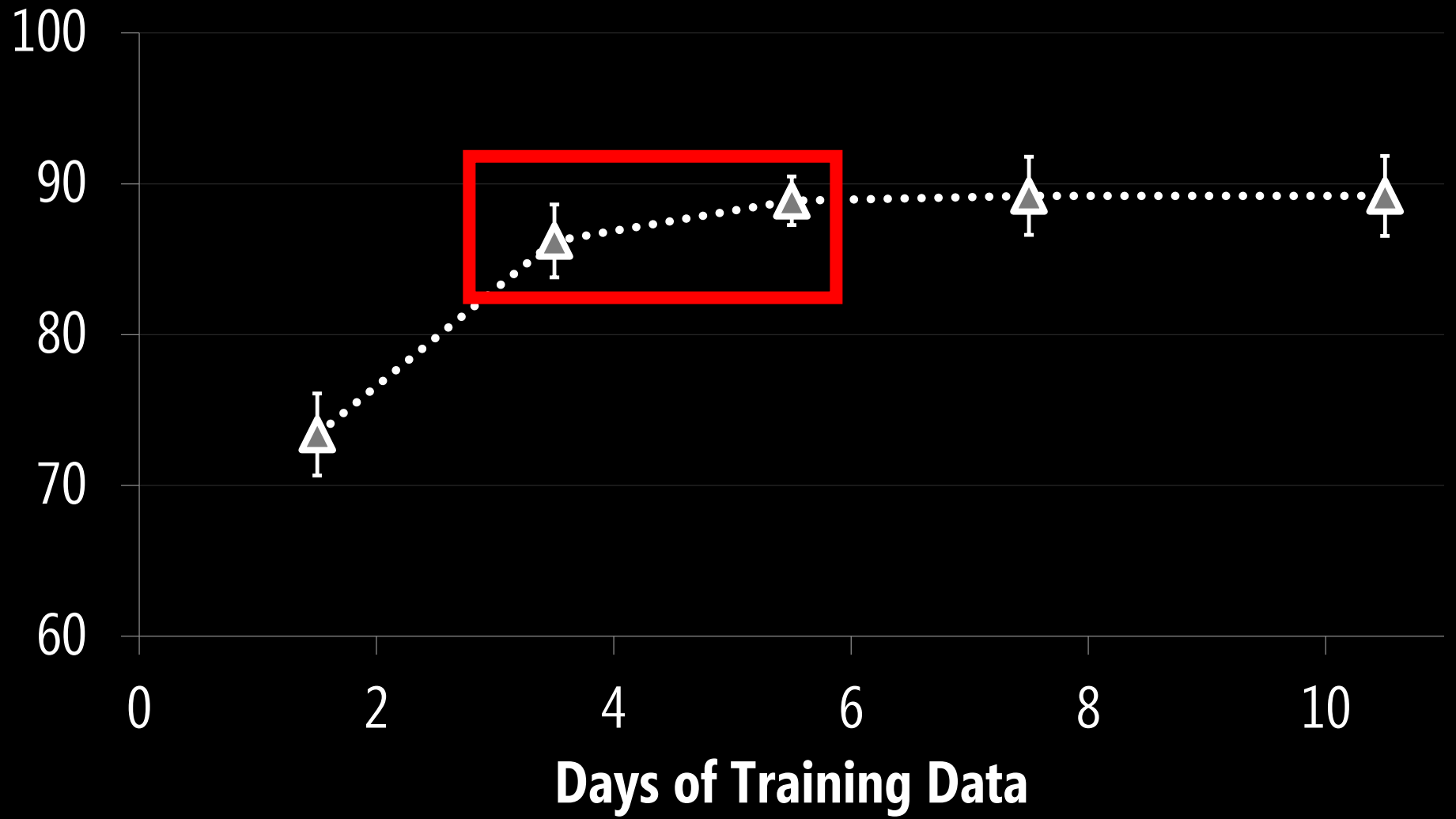
real-world water usage data



*error bars = std error

hydrosense training results

real-world water usage data



*error bars = std error

GETTING HIGH-QUALITY TRAINING DATA CAN BE A SIGNIFICANT BARRIER TO ML IN ACADEMIA AND INDUSTRY

TRANSPORTATION

HOW TESLA AND WAYMO ARE TACKLING A MAJOR PROBLEM FOR SELF-DRIVING CARS: DATA

Autonomous cars won't happen without tons of data, but Tesla and Waymo have a big head start

By Sean O'Kane | @sokane1 | Apr 19, 2018, 8:00am EDT

Illustration by William Joel

SHARE

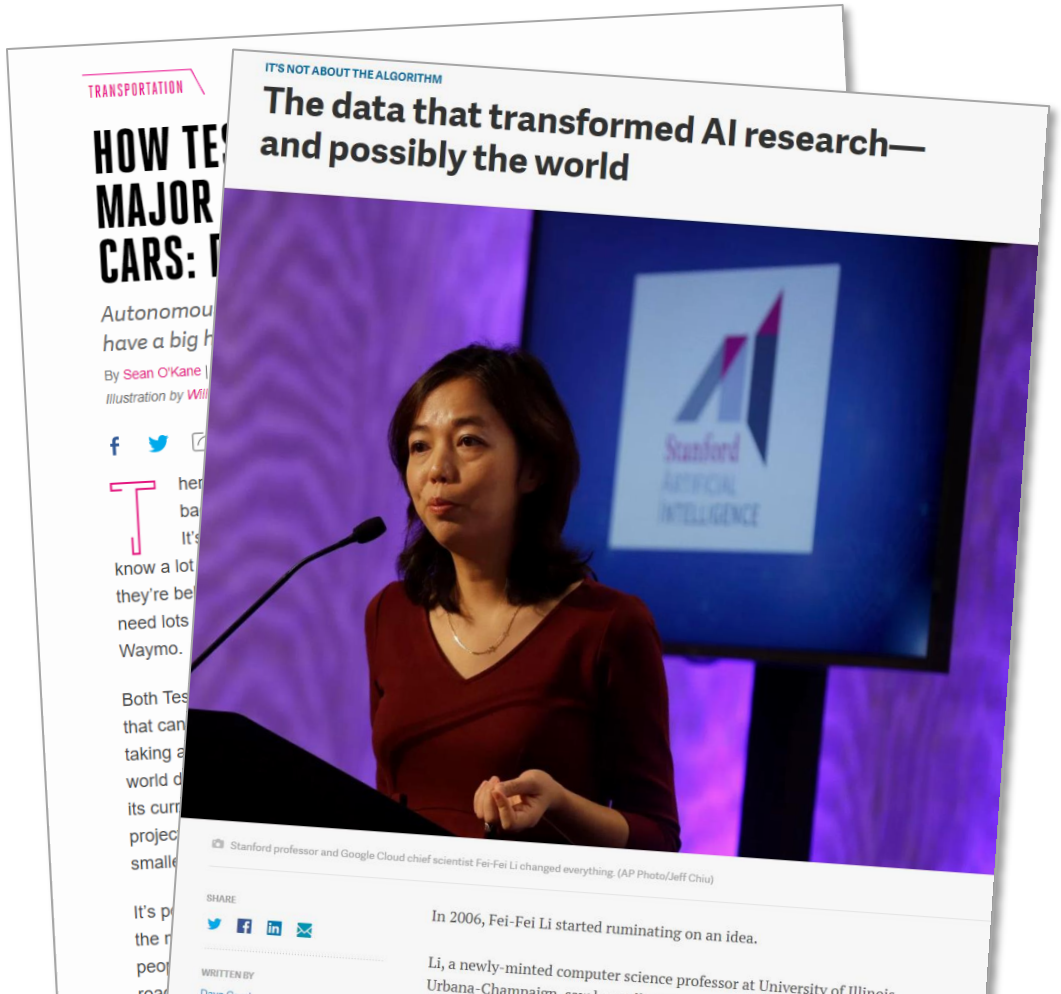
There's a race happening right now that stretches from Silicon Valley to Detroit and back: who can make a self-driving car that behaves better than a human driver? It's a far harder task than it sounded even a few years ago because human drivers know a lot — not just about their cars but about *how people behave* on the road when they're behind the wheel. To reach that same kind of understanding, computerized cars need lots of data. And the two companies with the most data right now are Tesla and Waymo.

Both Tesla and Waymo are attempting to collect and process enough data to create a car that can drive itself. And they're approaching those problems in very different ways. Tesla is taking advantage of the hundreds of thousands of cars it has on the road by collecting real-world data about how those vehicles perform (and how they *might* perform) with Autopilot, its current semi-autonomous system. Waymo, which started as Google's self-driving car project, uses powerful computer simulations and feeds what it learns from those into a smaller real-world fleet.

IT'S NOT ABOUT THE ALGORITHM

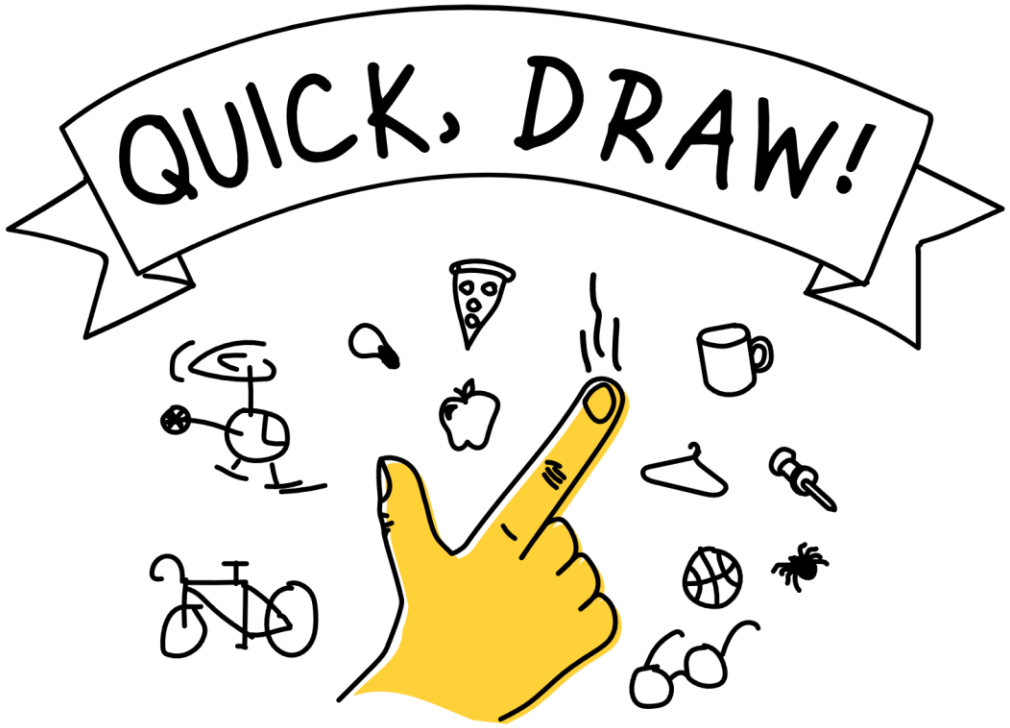
The data that transformed AI research—and possibly the world

GETTING HIGH-QUALITY TRAINING DATA CAN BE A SIGNIFICANT BARRIER TO ML IN ACADEMIA AND INDUSTRY



> In 2006, Fei-Fei Li started ruminating on an idea.
>
> Li, a newly-minted computer science professor at University of Illinois Urbana-Champaign, saw her colleagues across academia and the AI industry hammering away at the same concept: a better algorithm would make better decisions, regardless of the data.
>
> But she realized a limitation to this approach—the best algorithm wouldn't work well if the data it learned from didn't reflect the real world.

Fei-Fei Li is now a Stanford professor and Chief Scientist at Google Cloud



Can a neural network learn to recognize doodling?

Google for: Google Quick Draw

Let's Draw!

GETTING TRAINING DATA

GAMES WITH A PURPOSE

Labeling Images with a Computer Game

Luis von Ahn and Laura Dabbish
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA, USA
{biglou,dabbish}@cs.cmu.edu

Abstract

We introduce a new interactive system: a game that is fun and can be used to create valuable output. When people play the game they help determine the contents of images by providing meaningful labels for them. If the game is played as much as popular online games, we estimate that most images on the Web can be labeled in a few months. Having proper labels associated with each image on the Web would allow for more accurate image search, improve the accessibility of sites (by providing descriptions of images to visually impaired individuals), and help users block inappropriate images. Our system makes a significant contribution because of its valuable output and because of the way it addresses the image-labeling problem. Rather than using computer vision techniques, which don't work well enough, we encourage people to do the work by taking advantage of their desire to be entertained.

Categories & Subject Descriptors: I.2.6 [**Learning**]: Knowledge acquisition. H.3.m [**Information Retrieval**]: miscellaneous. H.5.3 [**HCI**]: Web-based interaction.

General Terms: Design, Human Factors, Languages

Keywords: Distributed knowledge acquisition, image labeling, online games, World Wide Web.

INTRODUCTION

Images on the Web present a major technological challenge. There are millions of them, there are no guidelines about providing appropriate textual descriptions for them, and computer vision hasn't yet produced a program that can determine their contents in a widely useful way. However, accurate descriptions of images are required by several applications like image search engines and accessibility programs for the visually impaired. Current techniques to categorize images for these applications are insufficient in many ways, mostly because they assume that the contents of images on the Web are related to the text appearing in the page. This is insufficient because the text adjacent to the images is often scarce, and can be misleading or hard to process [4].

CHI 2004, *April 24–29, 2004, Vienna, Austria.*

The only method currently available for obtaining precise image descriptions is manual labeling, which is tedious and thus extremely costly. But, what if people labeled images without realizing they were doing so? What if the experience was enjoyable? In this paper we introduce a new interactive system in the form of a game with a unique property: the people who play the game label images for us.

The labels generated by our game can be useful for a variety of applications. For accessibility purposes, visually impaired individuals surfing the Web need textual descriptions of images to be read aloud. For computer vision research, large databases of labeled images are needed as training sets for machine learning algorithms. For image search over the Web and inappropriate (e.g., pornographic) content filtering, proper labels could dramatically increase the accuracy of current systems.

We believe our system makes a significant contribution, not only because of its valuable output, but also because of the way it addresses the image-labeling problem. Rather than making use of computer vision techniques, we take advantage of people's existing perceptual abilities and desire to be entertained.

Our goal is ambitious: to label the majority of images on the World Wide Web. If our game is deployed at a popular gaming site like Yahoo! Games and if people play it as much as other online games, we estimate that most images on the Web can be properly labeled in a matter of weeks. As we show below, 5,000 people continuously playing the game could assign a label to all images indexed by Google [10] in 31 days.

We stress that our method is not necessarily meant to compete against the other techniques available for handling images on the Web. The labels produced using our game can be combined with these techniques to provide a powerful solution.

The Open Mind Initiative

Our work is similar in spirit to the Open Mind Initiative (e.g., [18,19]), a worldwide effort to develop "intelligent" software. Open Mind collects information from regular Internet users (referred to as "netizens") and feeds it to machine learning algorithms. Volunteers participate by answering questions and teaching concepts to computer programs. Our method is similar to Open Mind in that we plan to use regular people on the Internet to label images for us. However, we put greater emphasis on our method being fun because of the scale of the problem that we want

Labeling images with a computer game
Von Ahn & Dabbish, CHI'04
2,325 citations

Figure 2. The ESP Game. Players try to "agree" on as many images as they can in 2.5 minutes. The thermometer at the bottom measures how many images partners have agreed on.

Dog
Leash
German
Shepard
Standing
Canine

Figure 4. An image with all its labels.

GETTING TRAINING DATA

RECAPTCHAS

reCAPTCHA: Human-Based Character Recognition via Web Security Measures

Luis von Ahn,* Benjamin Maurer, Colin McMillen, David Abraham, Manuel Blum

CAPTCHAs (Completely Automated Public Turing test to tell Computers and Humans Apart) are widespread security measures on the World Wide Web that prevent automated programs from abusing online services. They do so by asking humans to perform a task that computers cannot yet perform, such as deciphering distorted characters. Our research explored whether such human effort can be channeled into a useful purpose: helping to digitize old printed material by asking users to decipher scanned words from books that computerized optical character recognition failed to recognize. We showed that this method can transcribe text with a word accuracy exceeding 99%, matching the guarantee of professional human transcribers. Our apparatus is deployed in more than 40,000 Web sites and has transcribed over 440 million words.

A CAPTCHA (*1*, *2*) is a challenge-response test used on the World Wide Web to determine whether a user is a human or a computer. The acronym stands for Completely Automated Public Turing test to tell Computers and Humans Apart. A typical CAPTCHA is an image containing several distorted characters that appears at the bottom of Web registration forms. Users are asked to type the wavy characters to "prove" they are human. Current computer programs cannot read distorted text as well as humans can (*3*), so CAPTCHAs act as sentries against automated programs that attempt to abuse online services. Owing to their effectiveness as a security measure, CAPTCHAs are used to protect many types of Web sites, including free-e-mail providers, ticket sellers, social networks, wikis, and blogs. For example, CAPTCHAs prevent ticket scalpers from using computer programs to buy large numbers of concert tickets, only to resell them at an inflated price. Sites such as Gmail and Yahoo Mail use CAPTCHAs to stop spammers from obtaining millions of free e-mail accounts, which they would use to send spam e-mail.

According to our estimates, humans around the world type more than 100 million CAPTCHAs every day (see supporting online text), in each case spending a few seconds typing the distorted characters. In aggregate, this amounts to hundreds of thousands of human hours per day. We report on an experiment that attempts to make positive use of the time spent by humans solving CAPTCHAs. Although CAPTCHAs are effective at preventing large-scale abuse of online services, the mental effort each person spends solving them is otherwise wasted. This mental effort is invaluable, because deciphering CAPTCHAs requires people to perform a task that computers cannot.

We show how it is possible to use CAPTCHAs to help digitize typeset texts in nondigital form by enlisting humans to decipher the words that computers cannot recognize. Physical books and other texts written before the computer age are currently being digitized en masse (e.g., by the Google Books Project and the nonprofit Internet

Computer Science Department, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA.
*To whom correspondence should be addressed. E-mail: biglou@cs.cmu.edu

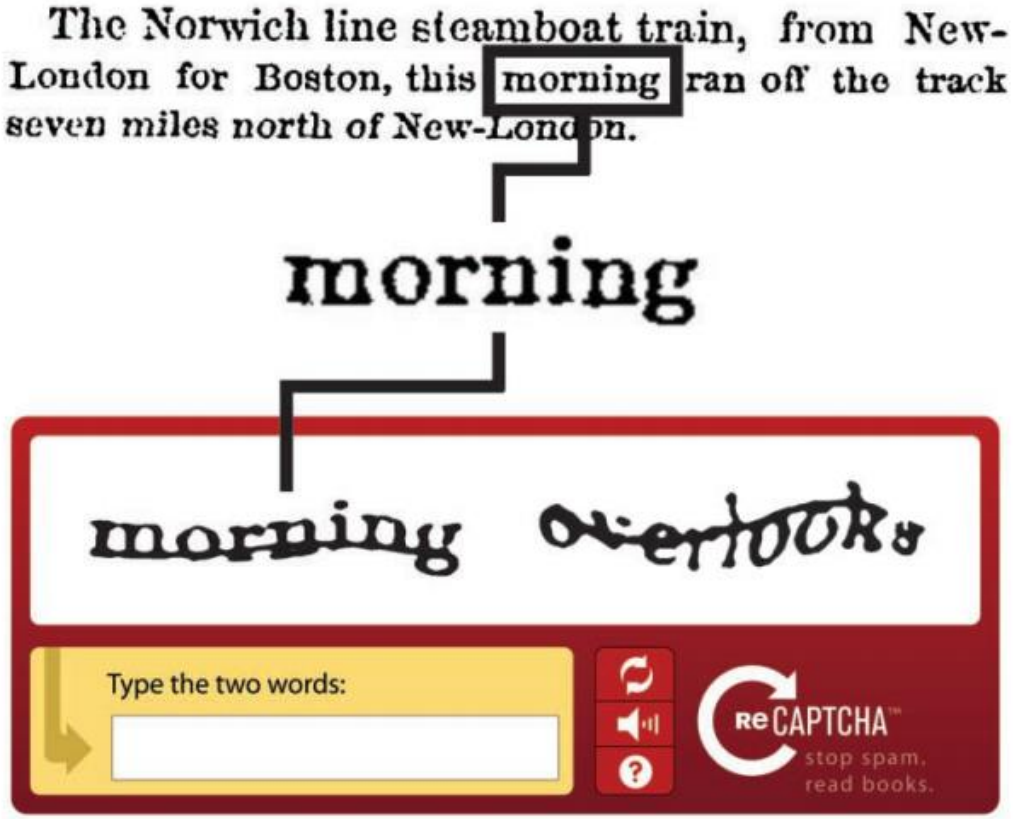
of random characters rendered by a computer, reCAPTCHA displays words taken from scanned texts. The solutions entered by humans are used to improve the digitization process. To increase efficiency and security, only the words that automated OCR programs cannot recognize are sent to humans. However, to meet the goal of a CAPTCHA (differentiating between humans and computers), the system needs to be able to verify the user's answer. To do this, reCAPTCHA gives the user two words, the one for which the answer is not known and a second "control" word for which the answer is known. If users correctly type the control word, the system assumes they are human and gains confidence that they also typed the other word correctly (Fig. 1). We describe the exact process below.

We start with an image of a scanned page. Two different OCR programs analyze the image; their respective outputs are then aligned with each other by standard string matching algorithms (*7*) and compared to each other and to an English dictionary. Any word that is deciphered differently by both OCR programs or that is not in the English dictionary is marked as "suspicious." These are typically the words that the OCR programs failed to decipher correctly. According to our analysis, about 96% of these suspicious words are recognized incorrectly by at least one of the OCR programs; conversely, 99.74% of the words not marked as suspicious are deciphered correctly by both programs. Each suspicious word is then placed in an image along with another word for which the answer is already known, the two words are distorted further to ensure that automated programs cannot decipher them, and the resulting image is used as a CAPTCHA. Users are asked to

match the highest-quality guarantee given by dedicated human transcription services. A random sample of 50 scanned articles from five different years (1860, 1865, 1908, 1935, and 1970) of the *New York Times* archive (http://nytimes.com) was chosen and manually transcribed by two professionals to estimate the per-word accuracy of reCAPTCHA, including the postprocessing corrections mentioned above. The total number of words was 24,080. Each word counted as a "hit" if the algorithm deciphered the entire word correctly or a "miss" if any of the letters were wrong. The error rate was defined as the number of misses divided by the

Fig. 1. The reCAPTCHA system displays words from scanned texts to humans on the World Wide Web. In this example, the word "morning" was unrecognizable by OCR. reCAPTCHA isolated the word, distorted it using random transformations including adding a line through it, and then presented it as a challenge to a user. Because the original word ("morning") was not recognized by OCR, another word for which the answer was known ("overlooks") was also presented to determine if the user entered the correct answer.

Fig. 1. The reCAPTCHA system displays words from scanned texts to humans on the World Wide Web. In this example, the word "morning" was unrecognizable by OCR. reCAPTCHA isolated the word, distorted it using random transformations including adding a line through it, and then presented it as a challenge to a user. Because the original word ("morning") was not recognized by OCR, another word for which the answer was known ("overlooks") was also presented to determine if the user entered the correct answer.



reCAPTCHA: Human-Based Character Recognition via Web Security Measures

Von Ahn *et al.*, CHI'08

1,043 citations

GETTING TRAINING DATA

RECAPTCHAS

reCAPTCHA: Human-Based Character Recognition via Web Security Measures

Luis von Ahn,* Benjamin Maurer, Colin McMillen, David Abraham, Manuel Blum

CAPTCHAs (Completely Automated Public Turing test to tell Computers and Humans Apart) are widespread security measures on the World Wide Web that prevent automated programs from abusing online services. They do so by asking humans to perform a task that computers cannot yet perform, such as deciphering distorted characters. Our research explored whether such human effort can be channeled into a useful purpose: helping to digitize old printed material by asking users to decipher scanned words from books that computerized optical character recognition failed to recognize. We showed that this method can transcribe text with a word accuracy exceeding 99%, matching the guarantee of professional human transcribers. Our apparatus is deployed in more than 40,000 Web sites and has transcribed over 440 million words.

CAPTCHA (*1*, *2*) is a challenge-response test used on the World Wide Web to determine whether a user is a human or a computer. The acronym stands for Completely Automated Public Turing test to tell Computers and Humans Apart. A typical CAPTCHA is an image containing several distorted characters that appears at the bottom of Web registration forms. Users are asked to type the wavy characters to "prove" they are human. Current computer programs cannot read distorted text as well as humans can (*3*), so CAPTCHAs act as sentries against automated programs that attempt to abuse online services. Owing to their effectiveness as a security measure, CAPTCHAs are used to protect many types of Web sites, including free-e-mail providers, ticket sellers, social networks, wikis, and blogs. For example, CAPTCHAs prevent ticket scalpers from using computer programs to buy large numbers of concert tickets, only to resell them at an inflated price. Sites such as Gmail and Yahoo Mail use CAPTCHAs to stop spammers from obtaining millions of free e-mail accounts, which they would use to send spam e-mail.

According to our estimates, humans around the world type more than 100 million CAPTCHAs every day (see supporting online text), in each case spending a few seconds typing the distorted characters. In aggregate, this amounts to hundreds of thousands of human hours per day. We report on an experiment that attempts to make positive use of the time spent by humans solving CAPTCHAs. Although CAPTCHAs are effective at preventing large-scale abuse of online services, the mental effort each person spends solving them is otherwise wasted. This mental effort is invaluable, because deciphering CAPTCHAs requires people to perform a task that computers cannot.

We show how it is possible to use CAPTCHAs to help digitize typeset texts in nondigital form by enlisting humans to decipher the words that computers cannot recognize. Physical books and other texts written before the computer age are currently being digitized en masse (e.g., by the Google Books Project and the nonprofit Internet Computer Science Department, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA.

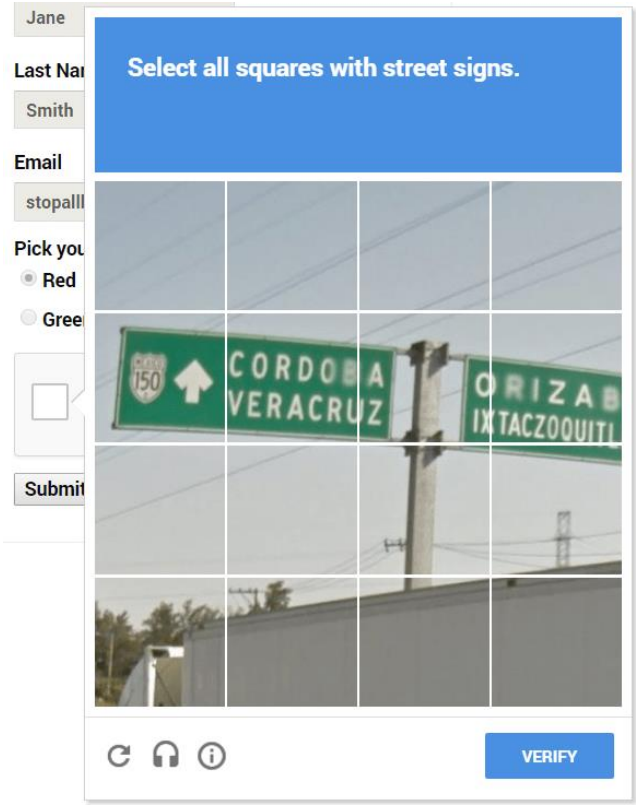
*To whom correspondence should be addressed. E-mail: biglou@cs.cmu.edu

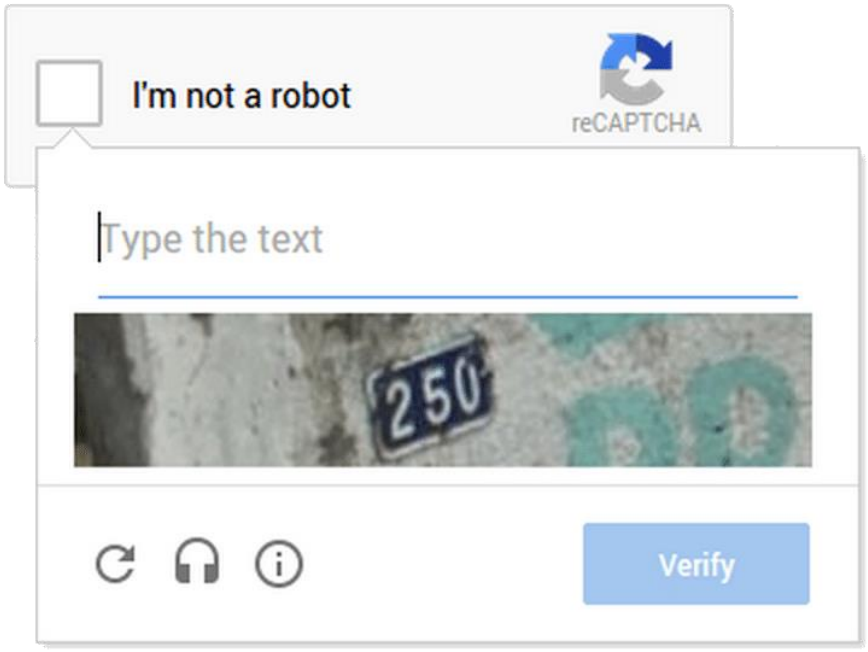
of random characters rendered by a computer, reCAPTCHA displays words taken from scanned texts. The solutions entered by humans are used to improve the digitization process. To increase efficiency and security, only the words that automated OCR programs cannot recognize are sent to humans. However, to meet the goal of a CAPTCHA (differentiating between humans and computers), the system needs to be able to verify the user's answer. To do this, reCAPTCHA gives the user two words, the one for which the answer is not known and a second "control" word for which the answer is known. If users correctly type the control word, the system assumes they are human and gains confidence that they also typed the other word correctly (Fig. 1). We describe the exact process below.

We start with an image of a scanned page. Two different OCR programs analyze the image; their respective outputs are then aligned with each other by standard string matching algorithms (*7*) and compared to each other and to an English dictionary. Any word that is deciphered differently by both OCR programs or that is not in the English dictionary is marked as "suspicious." These are typically the words that the OCR programs failed to decipher correctly. According to our analysis, about 96% of these suspicious words are recognized incorrectly by at least one of the OCR programs; conversely, 99.74% of the words not marked as suspicious are deciphered correctly by both programs. Each suspicious word is then placed in an image along with another word for which the answer is already known, the two words are distorted further to ensure that automated programs cannot decipher them, and the resulting image is used as a CAPTCHA. Users are asked to guesses match each other, but differ from both of the OCRs' guesses, then (and only then) the word becomes a control word in other challenges. In case of discrepancies among human answers, reCAPTCHA sends the word to more humans as an "unknown word" and picks the answer with the highest number of "votes," where each human answer counts as one vote and each OCR guess counts as one half of a vote (recall that these words all have been previously processed by OCR). In practice, these weights seem to yield the best results, though our accuracy is not very sensitive to them (as long as more weight is given to human guesses match the highest-quality guarantee given by dedicated human transcription services. A random sample of 50 scanned articles from five different years (1860, 1865, 1908, 1935, and 1970) of the *New York Times* archive (http://nytimes.com) was chosen and manually transcribed by two professionals to estimate the per-word accuracy of reCAPTCHA, including the postprocessing corrections mentioned above. The total number of words was 24,080. Each word counted as a "hit" if the algorithm deciphered the entire word correctly or a "miss" if any of the letters were wrong. The error rate was defined as the number of misses divided by the

Fig. 1. The reCAPTCHA system displays words from scanned texts to humans on the World Wide Web. In this example, the word "morning" was unrecognizable by OCR. reCAPTCHA isolated the word, distorted it using random transformations including adding a line through it, and then presented it as a challenge to a user. Because the original word ("morning") was not recognized by OCR, another word for which the answer was known ("overlooks") was also presented to determine if the user entered the correct answer.

reCAPTCHA: Human-Based Character Recognition via Web Security Measures
Von Ahn *et al.*, CHI'08
1,043 citations





Google purchased von Ahn's CAPTCHA company and has integrated it into its products

PARAMETER TUNING

Most ML algorithms have parameters to set

When you train a model, you must provide values for those parameters and the efficacy of the model depends on those parameters

The optimal set of parameters is known as *model selection*

(Yes, this is similar to feature selection and the approaches are generally similar too!)

Source: https://docs.microsoft.com/en-us/azure/machine-learning/studio/algorithm-parameters-optimize

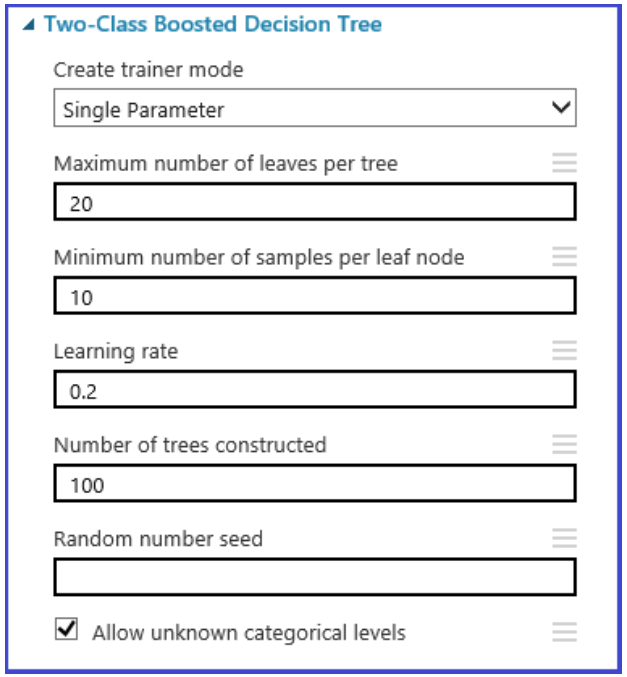
PARAMETER TUNING PROCESS

1. Define the parameter space. For a given ML model, decide what range of parameter values you want to consider.
2. Define the evaluation method (*e.g.*, decide how to choose cross-validation folds for the dataset)
3. Define your key metrics (*e.g.*, accuracy, precision, recall, f-measure, or root-mean squared error)
4. Train and evaluate. For each unique combination of the parameter values, cross-validation performed and your key metrics produced.
5. Compare output and choose the best-performing parameters for your model.

Based on: https://docs.microsoft.com/en-us/azure/machine-learning/studio/algorithm-parameters-optimize

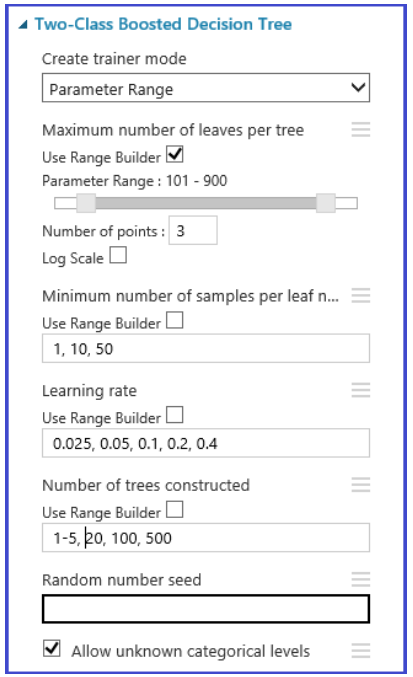
PARAMETER TUNING: DECISION TREE EXAMPLE ON AZURE

Most ML toolkits offer parameter tuning functionality. This example is from Microsoft Azure's ML toolkit.



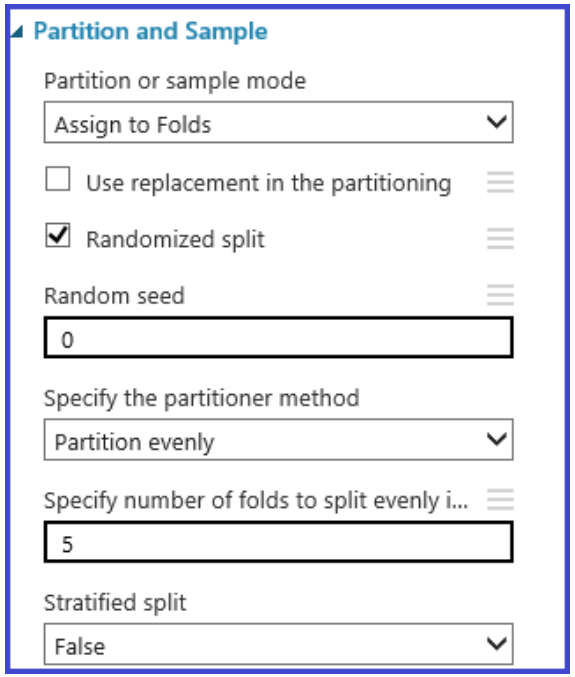
1. Choose and Setup Model

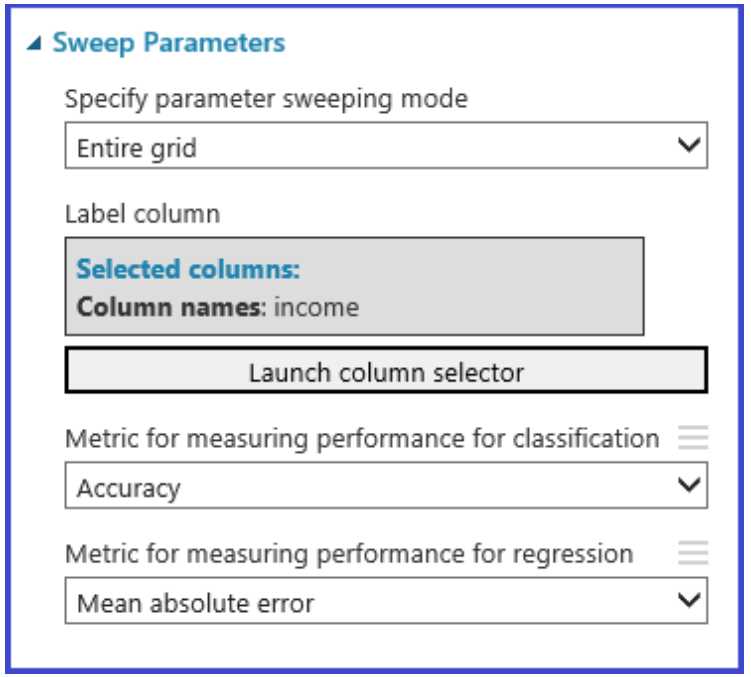
Select an ML model to tune



2. Parameter Ranges

Decide what range of values to input and compare





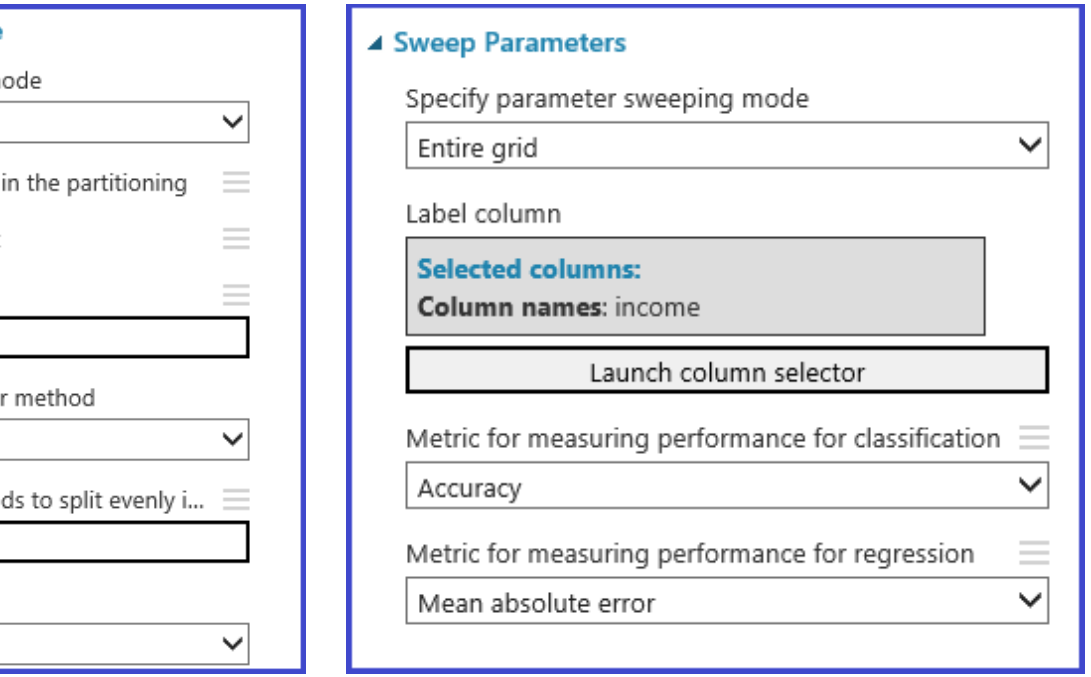
3. Setup Evaluation

Decide how many folds in cross-validation and key error metrics.

Based on: https://docs.microsoft.com/en-us/azure/machine-learning/studio/algorithm-parameters-optimize

PARAMETER TUNING: DECISION TREE EXAMPLE ON AZURE

Most ML toolkits offer parameter tuning functionality. This example is from Microsoft Azure's ML toolkit.



ation

y folds in cross-validation and key error metrics.

rows 180 columns 11

Number of leaves	Minimum leaf instances	Learning rate	Number of trees	Accuracy	Precision	Recall	F-Score	AUC	Average Log Loss	Training Log Loss
8	10	0.1	500	0.866128	0.75785	0.652595	0.701295	0.922837	0.290561	47.363286
8	1	0.05	500	0.866097	0.762796	0.644306	0.698562	0.922672	0.289389	47.575474
8	1	0.2	100	0.865944	0.763813	0.641755	0.697484	0.921469	0.291591	47.176649
32	1	0.05	100	0.865913	0.759367	0.648769	0.699725	0.921503	0.29273	46.970281
8	10	0.2	100	0.865882	0.763821	0.641372	0.697262	0.922052	0.290024	47.46048
32	1	0.025	500	0.865668	0.749604	0.663946	0.70418	0.923187	0.292484	47.014813
8	10	0.05	500	0.865514	0.761322	0.643158	0.697269	0.922803	0.288929	47.658898
32	1	0.1	100	0.865453	0.751234	0.659737	0.702519	0.922703	0.292779	46.961499
8	1	0.025	500	0.865391	0.764778	0.636909	0.695011	0.921386	0.29142	47.207519
8	50	0.05	500	0.865115	0.760932	0.641372	0.696055	0.92261	0.288991	47.647701
32	10	0.025	500	0.865115	0.748308	0.662798	0.702962	0.923435	0.29145	47.202173
8	50	0.025	500	0.865084	0.764336	0.635761	0.694145	0.921226	0.291554	47.183315
32	10	0.05	100	0.865084	0.756395	0.648642	0.698387	0.92155	0.29239	47.031913
8	10	0.025	500	0.864992	0.764066	0.635633	0.693957	0.921674	0.290844	47.312033
32	10	0.2	20	0.864961	0.757939	0.645326	0.697114	0.920488	0.294732	46.607674
8	1	0.1	500	0.864808	0.752756	0.653105	0.699399	0.922545	0.291764	47.145278
32	10	0.1	100	0.864777	0.748626	0.66012	0.701593	0.92295	0.291274	47.234005
8	1	0.1	100	0.864715	0.763255	0.635251	0.693395	0.920514	0.293092	46.904731
8	10	0.1	100	0.864715	0.763416	0.634996	0.693309	0.920639	0.292815	46.954929
8	10	0.4	100	0.864623	0.754863	0.648387	0.697585	0.921302	0.292915	46.936722
32	1	0.2	20	0.864439	0.757089	0.64354	0.695712	0.920124	0.295428	46.481506

4. Train, Evaluate, and compare

You can see the parameters in left-hand columns and the output produced.

Based on: https://docs.microsoft.com/en-us/azure/machine-learning/studio/algorithm-parameters-optimize

EVALUATING ML ALGORITHMS

Splitting up training and testing data (most common approach: k-fold cross validation)

Evaluating performance as a function of training data (*e.g.*, train ML model with minimum dataset, evaluate, and repeat with incrementally more data)

Evaluating performance as a function of training data type (*e.g.*, articulation speed of gesture)

Evaluating performance with N-Best lists

Evaluating performance with confusion matrices

KEY ML EVALUATION METRICS

The simplest evaluation metric is **accuracy**

$$Accuracy = \frac{Number\ of\ correct\ classifications}{Total\ number\ of\ classifications}$$

KEY ML EVALUATION METRICS

We can reframe many classification problems as retrieval problems. That is, rather than saying "classify gesture X," we say, "find all instances of gesture X in the dataset." For example, let's imagine we are searching for the "Forehand Tennis" gesture. (Note: for this to work, we would have to have either a learned or manually set "correct match" score threshold)

$$True\ positive = \#\ of\ correctly\ returned\ objects$$

e.g., # of returned "Forehand Tennis" gestures

$$False\ positive = \#\ of\ incorrectly\ returned\ objects$$

e.g., # of returned objects that were not "Forehand Tennis" gestures—they could be "At Rest" or "Backhand Tennis" gestures, etc.

$$True\ negative = \#\ of\ objects\ that\ were\ correctly\ not\ returned$$

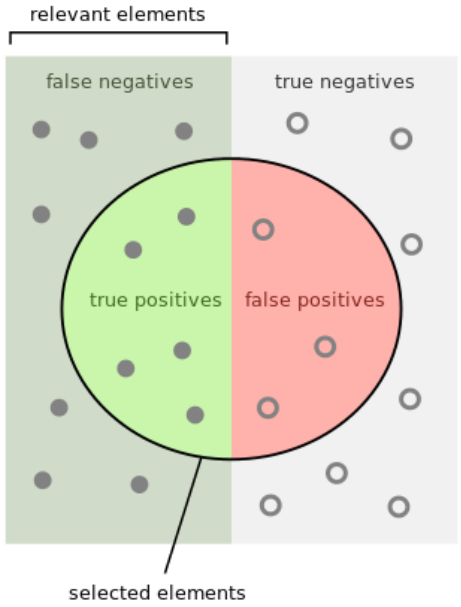
e.g., # of objects that were not "Forehand Tennis" gestures in our dataset and were correctly not returned

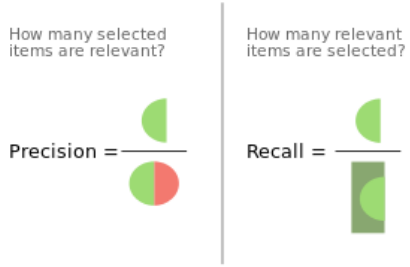
$$False\ negative = \#\ of\ objects\ that\ should\ have\ been\ found\ but\ were\ not$$

e.g., # of objects that were "Forehand Tennis" gestures in our dataset but were not actually found and returned.

KEY ML EVALUATION: PRECISION AND RECALL

Recall is the fraction of relevant instances that have been retrieved over the total amount of relevant instances in our dataset and Precision is the fraction of relevant instances among the retrieved instances.





Recall

How much of the correct things did we find?

$$Recall = \frac{\#\ of\ True\ Positives}{\#\ of\ True\ Positives + \#\ of\ False\ Negatives}$$

Precision

Of those things found, how many were correct?

$$Precision = \frac{\#\ of\ True\ Positives}{\#\ of\ True\ Positives + \#\ of\ False\ Positives}$$

Source: https://en.wikipedia.org/wiki/Precision_and_recall

HOW TO EVALUATE ML ALGORITHMS

Input Features

Class Label
(aka Ground Truth Label)

Index	Color	Weight	Label
1	Red	200g	Apple
2	Orange	300g	Orange
3	Red	150g	Apple
N	...		

Total Labeled Dataset

How much of our dataset should we reserve for training and how much for testing?

ML EVALUATION

K-FOLD CROSS VALIDATION

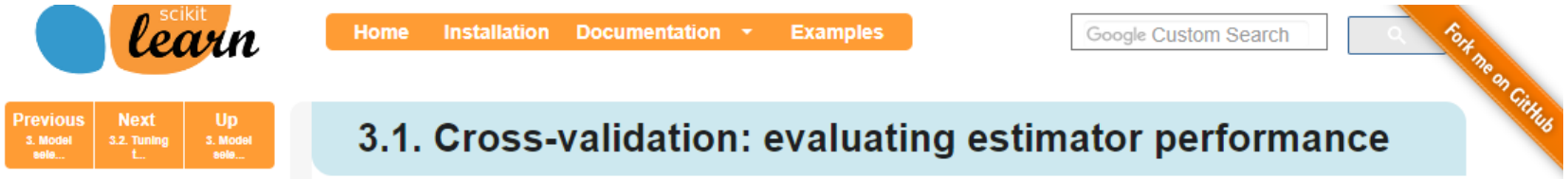
Source: Udacity, K-fold cross validation, https://youtu.be/TIgfjmp-4BA

C

Source: http://scikit-learn.org/stable/modules/cross_validation.html

ML EVALUATION

SCI-KIT SUPPORT FOR K-FOLD



Previous (3. Model sele...) | Next (3.2. Tuning t...) | Up (3. Model sele...)

scikit-learn v0.19.1
Other versions

Please **cite us** if you use the software.

3.1. Cross-validation: evaluating estimator performance

Learning the parameters of a prediction function and testing it on the same data is a methodological mistake: a model that would just repeat the labels of the samples that it has just seen would have a perfect score but would fail to predict anything useful on yet-unseen data. This situation is called **overfitting**. To avoid it, it is common practice when performing a (supervised) machine learning experiment to hold out part of the available data as a **test set** `X_test, y_test`. Note that the word "experiment" is not intended to denote academic use only, because even in commercial settings machine learning usually starts out experimentally.

In scikit-learn a random split into training and test sets can be quickly computed with the `train_test_split` helper function. Let's load the iris data set to fit a linear support vector machine on it:

```
>>> import numpy as np
>>> from sklearn.model_selection import train_test_split
>>> from sklearn import datasets
>>> from sklearn import svm

>>> iris = datasets.load_iris()
>>> iris.data.shape, iris.target.shape
((150, 4), (150,))
```

We can now quickly sample a training set while holding out 40% of the data for testing (evaluating) our classifier:

```
>>> X_train, X_test, y_train, y_test = train_test_split(
...     iris.data, iris.target, test_size=0.4, random_state=0)

>>> X_train.shape, y_train.shape
((90, 4), (90,))
>>> X_test.shape, y_test.shape
((60, 4), (60,))

>>> clf = svm.SVC(kernel='linear', C=1).fit(X_train, y_train)
>>> clf.score(X_test, y_test)
0.96...
```

When evaluating different settings ("hyperparameters") for estimators, such as the `C` setting that must be manually set for an SVM, there is still a risk of overfitting *on the test set* because the parameters can be tweaked until the estimator performs optimally. This way, knowledge about the test set can "leak" into the model and evaluation metrics no longer report

ML EVALUATION

SCI-KIT SUPPORT FOR K-FOLD

3.1.2.1.3. Leave One Out (LOO)

`LeaveOneOut` (or LOO) is a simple cross-validation. Each learning set is created by taking all the samples except one, the test set being the sample left out. Thus, for n samples, we have n different training sets and n different tests set. This cross-validation procedure does not waste much data as only one sample is removed from the training set:

```
>>> from sklearn.model_selection import LeaveOneOut

>>> X = [1, 2, 3, 4]
>>> loo = LeaveOneOut()
>>> for train, test in loo.split(X):
...     print("%s %s" % (train, test))
[1 2 3] [0]
[0 2 3] [1]
[0 1 3] [2]
[0 1 2] [3]
```

Potential users of LOO for model selection should weigh a few known caveats. When compared with k-fold cross validation, one builds n models from n samples instead of k models, where $n > k$. Moreover, each is trained on $n - 1$ samples rather than $(k - 1)n/k$. In both ways, assuming k is not too large and $k < n$, LOO is more computationally expensive than k-fold cross validation.

In terms of accuracy, LOO often results in high variance as an estimator for the test error. Intuitively, since $n - 1$ of the n samples are used to build each model, models constructed from folds are virtually identical to each other and to the model built from the entire training set.

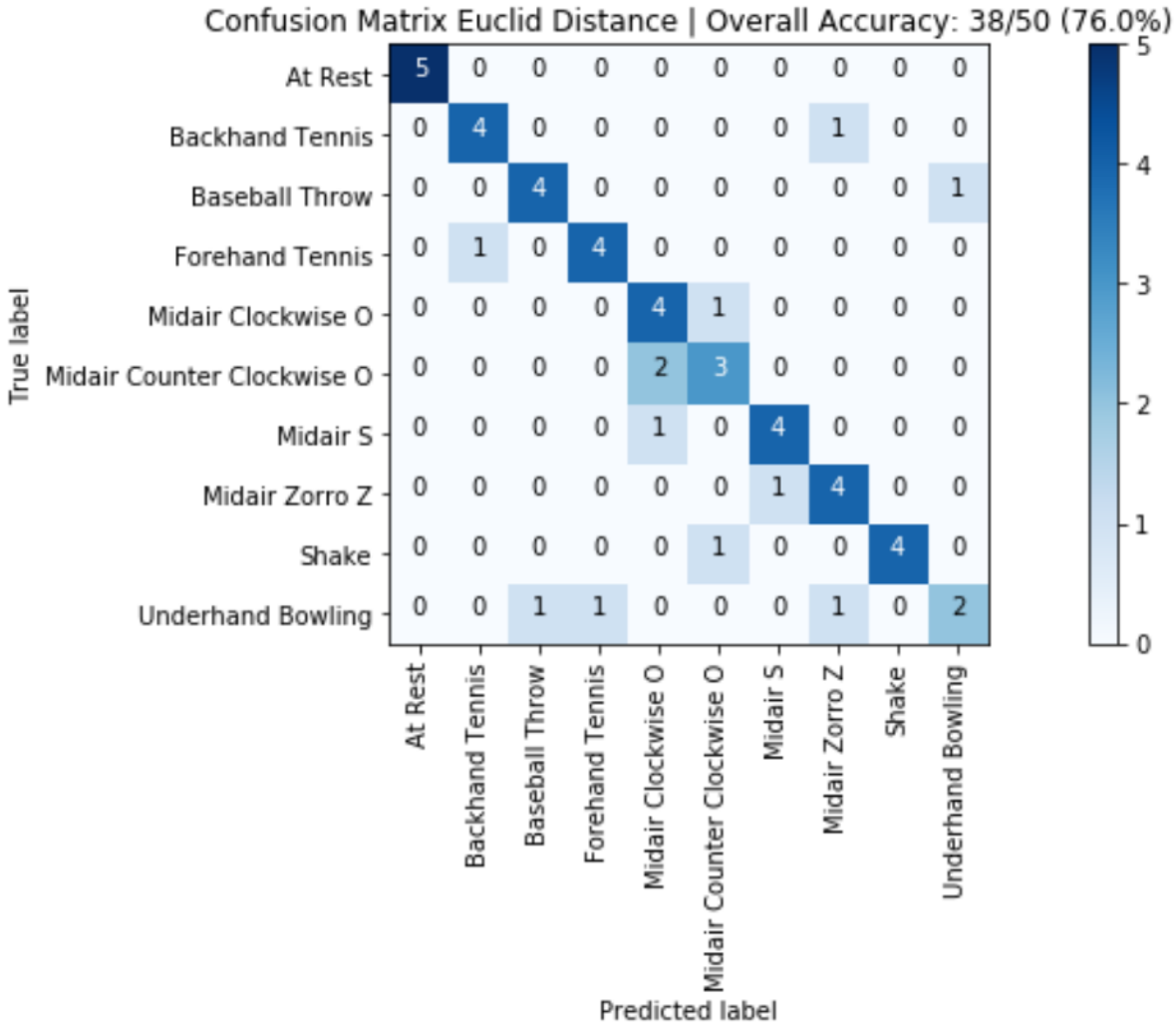
However, if the learning curve is steep for the training size in question, then 5- or 10- fold cross validation can overestimate the generalization error.

As a general rule, most authors, and empirical evidence, suggest that 5- or 10- fold cross validation should be preferred to LOO.

Don't do this!

ML EVALUATION

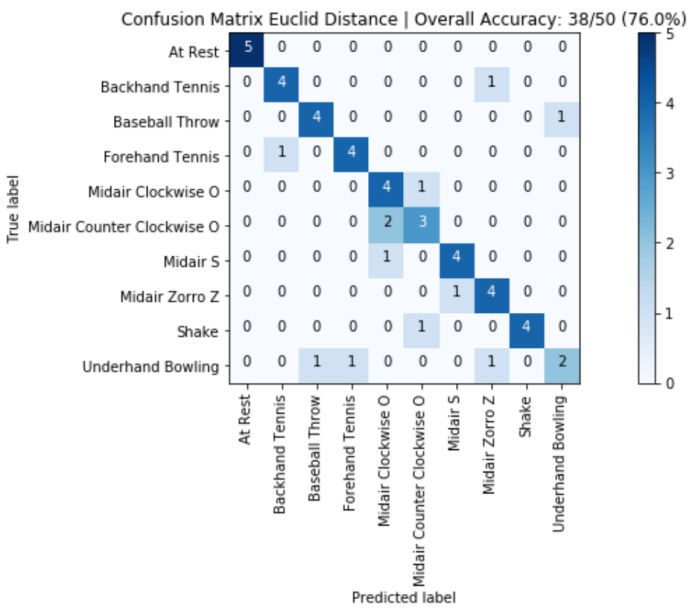
CONFUSION MATRICES

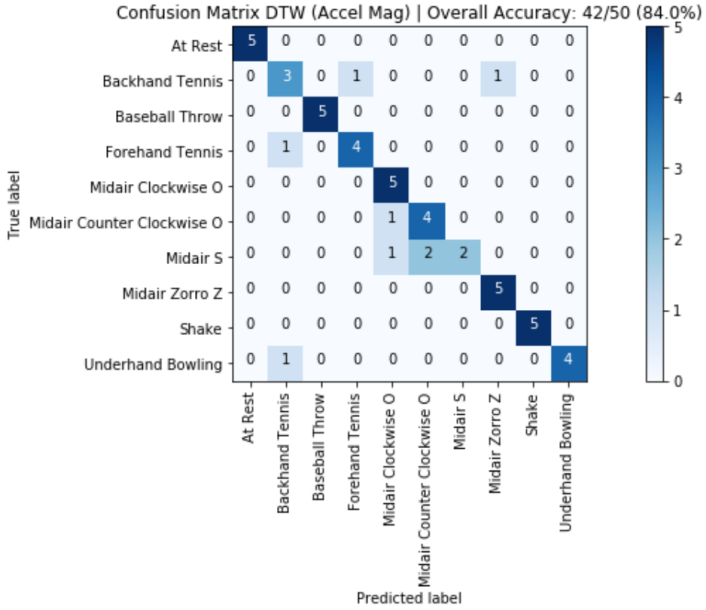


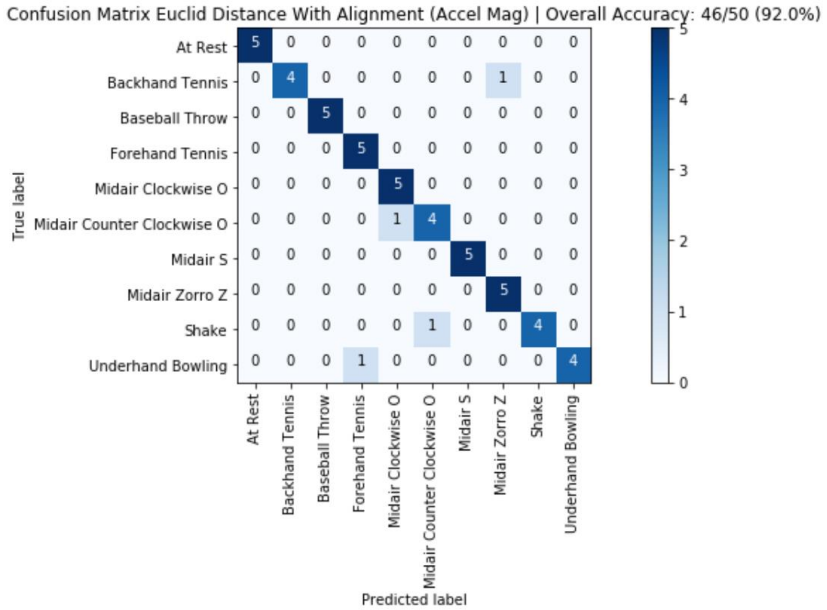
ML EVALUATION

CONFUSION MATRICES

Just Using Accel Magnitude



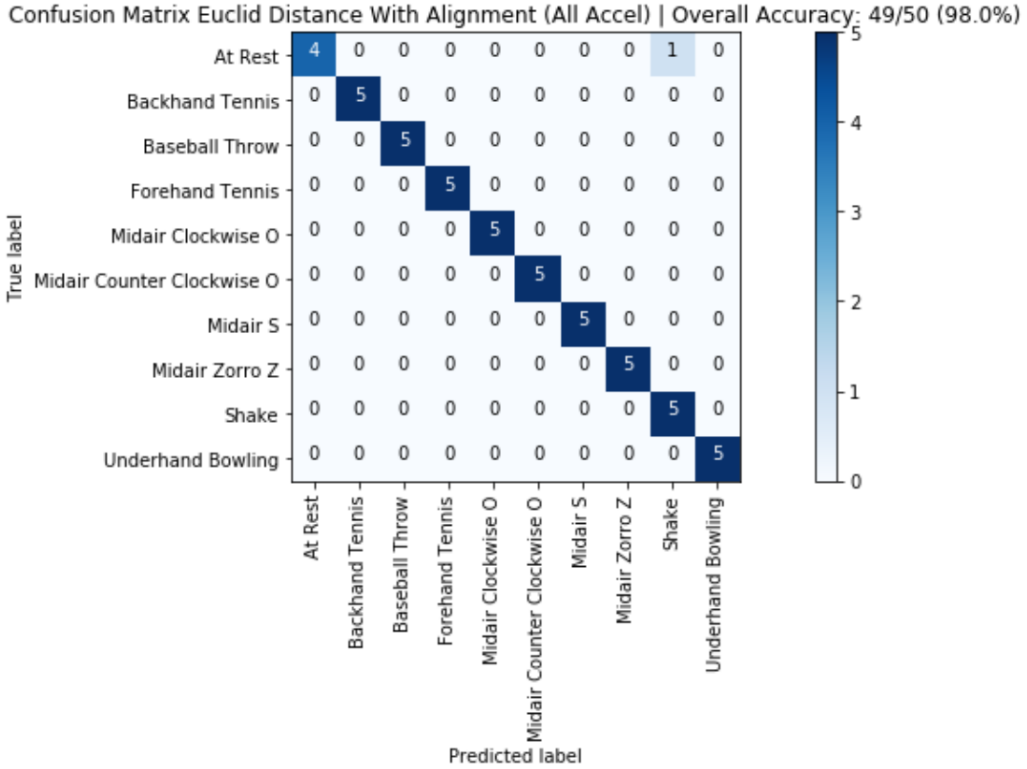


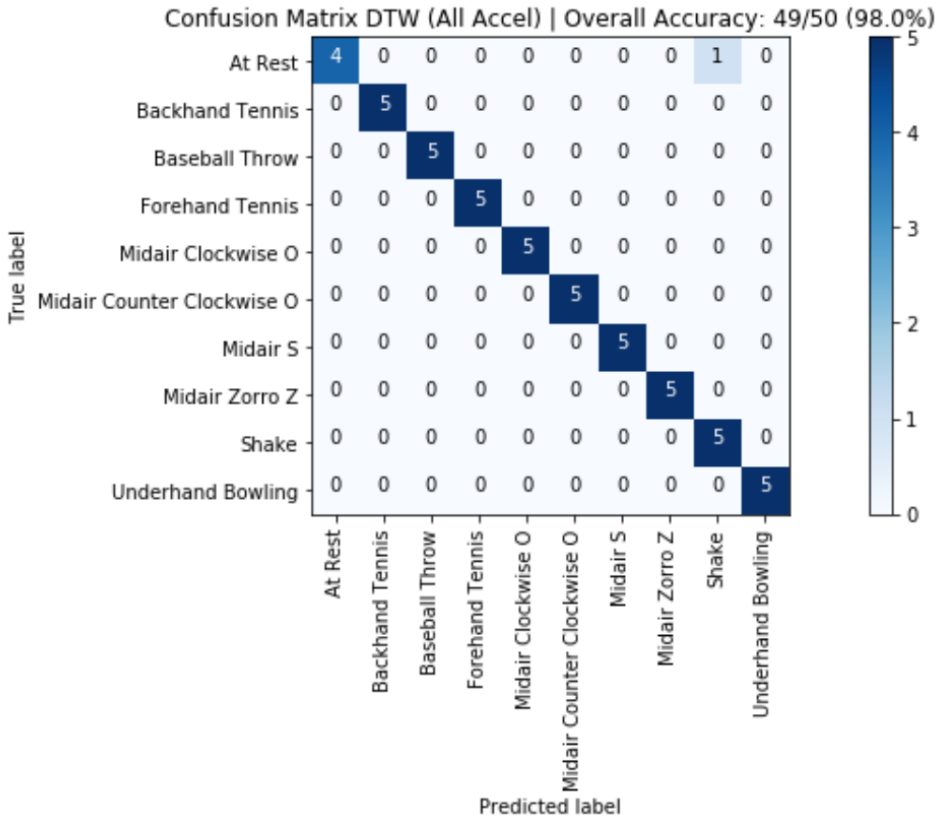


ML EVALUATION

CONFUSION MATRICES

Using Combination of Signals





ML EVALUATION

N-BEST LISTS

For algorithms that provide per item classification scores, we can compute an N-best list

N-best lists depict where the “right” answer is in an ordered list of matching scores between items

Change your matching functions to return a sorted list of (matches, scores) rather than just the (best match trial, score)

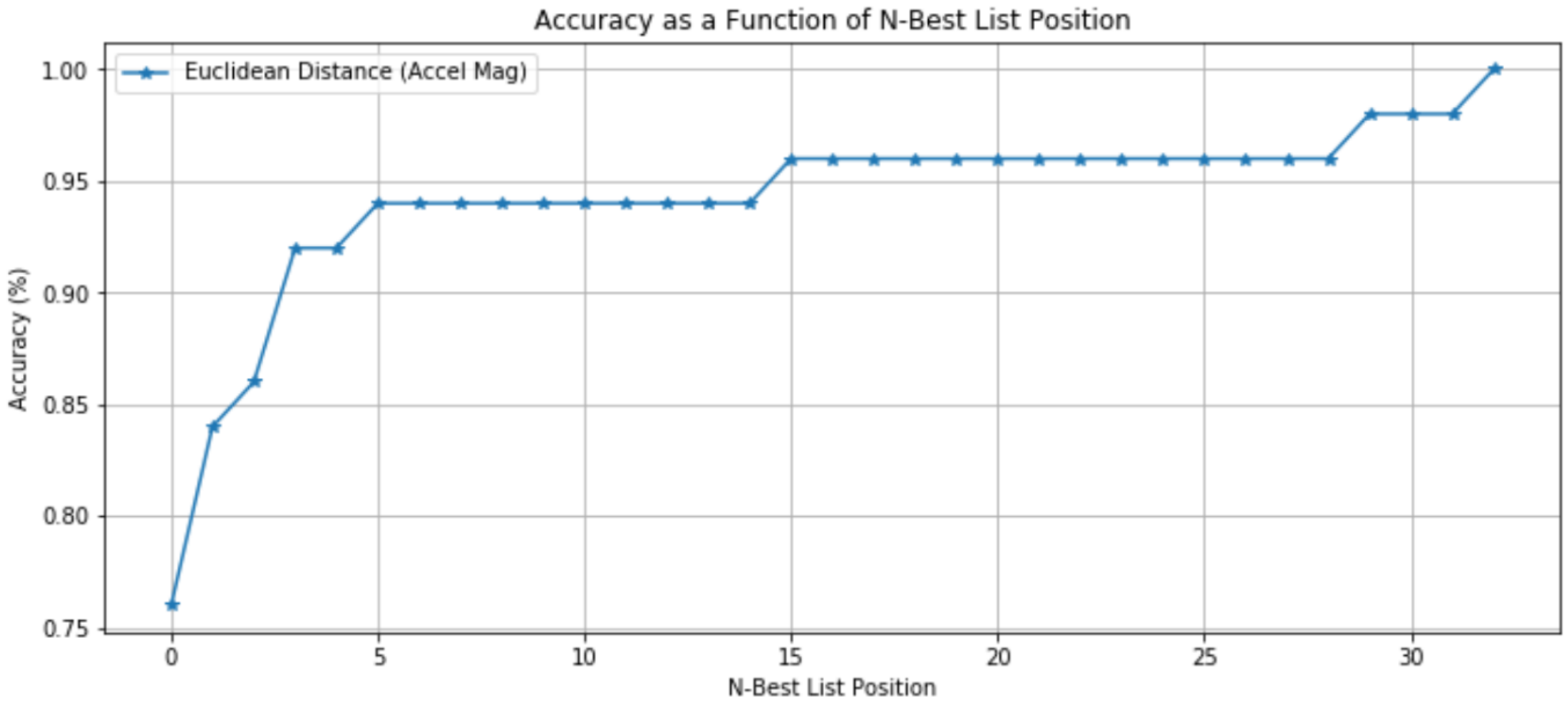
N-BEST LIST EXAMPLE CODE

```
# Just returns the best match based on euclidean distance
def find_closest_match_euclid_distance(testTrial, templateFolds, **kwargs):
    minDistance = float('inf')
    bestMatchTrial = None
    for fold in templateFolds:
        for foldGestureName, foldTrial in fold.items():
            # Calculate Euclidean distance
            euclidDistance = distance.euclidean(testTrial.signal, foldTrial.signal)
            if euclidDistance < minDistance:
                minDistance = euclidDistance
                bestMatchTrial = foldTrial

    # Return the best match
    return (bestMatchTrial, minDistance)


# Returns an nBestList tuple of (foldTrial, score) based on euclidean distance
def find_closest_match_euclid_distance(testTrial, templateFolds, **kwargs):
    # Tuple list of trial to score
    nBestList = list()
    for fold in templateFolds:
        for foldGestureName, foldTrial in fold.items():
            # Euclidean distance
            euclidDistance = distance.euclidean(testTrial.signal, foldTrial.signal)
            nBestList.append((foldTrial, euclidDistance))

    # sort the nBestList by score
    nBestList.sort(key=lambda x: x[1])
    return nBestList
```

ML EVALUATION

N-BEST LIST: SHAPE-MATCHING GESTURE REC

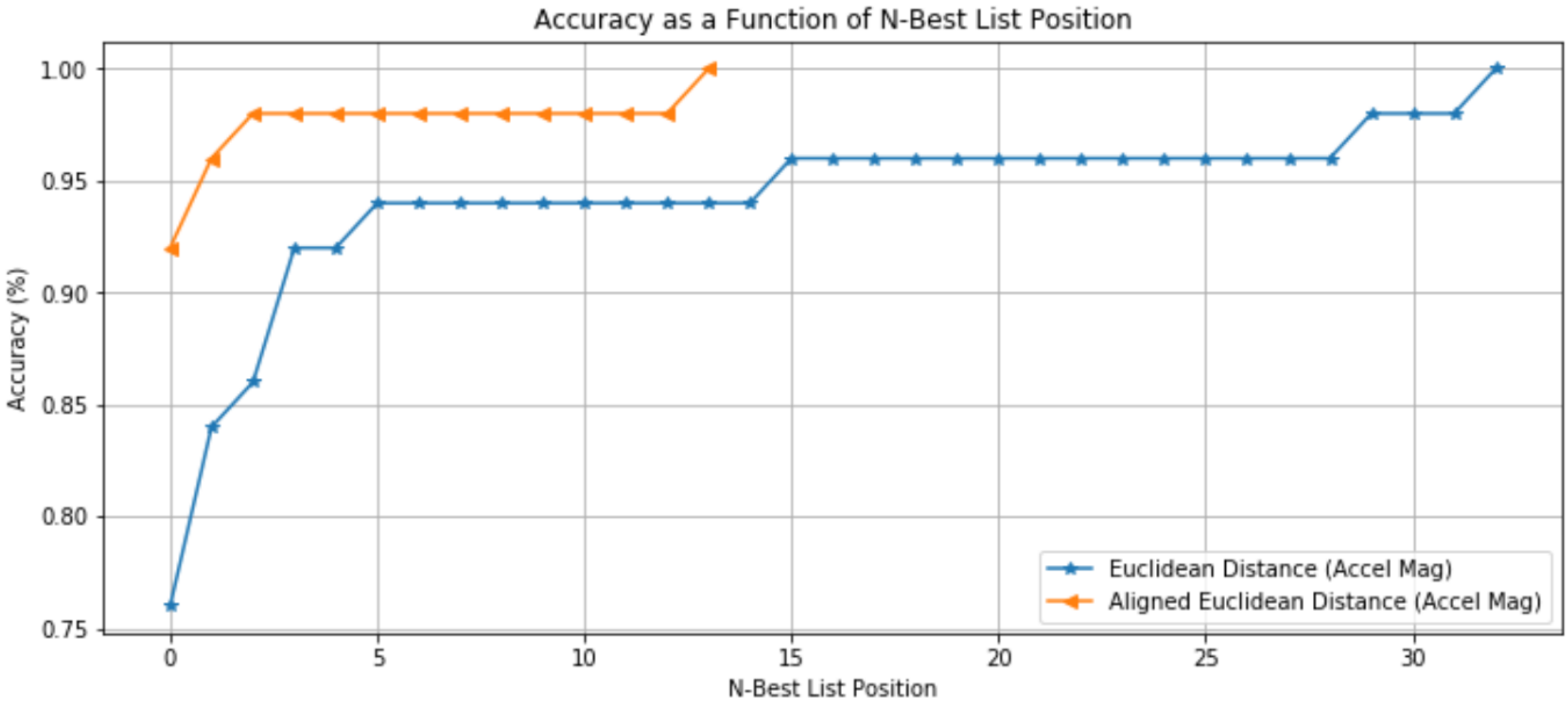
This is on the JonGestureLogs from https://github.com/jonfroehlich/CSE590Sp2018/tree/master/A02-GestureRecognizer



ML EVALUATION

N-BEST LIST: SHAPE-MATCHING GESTURE REC

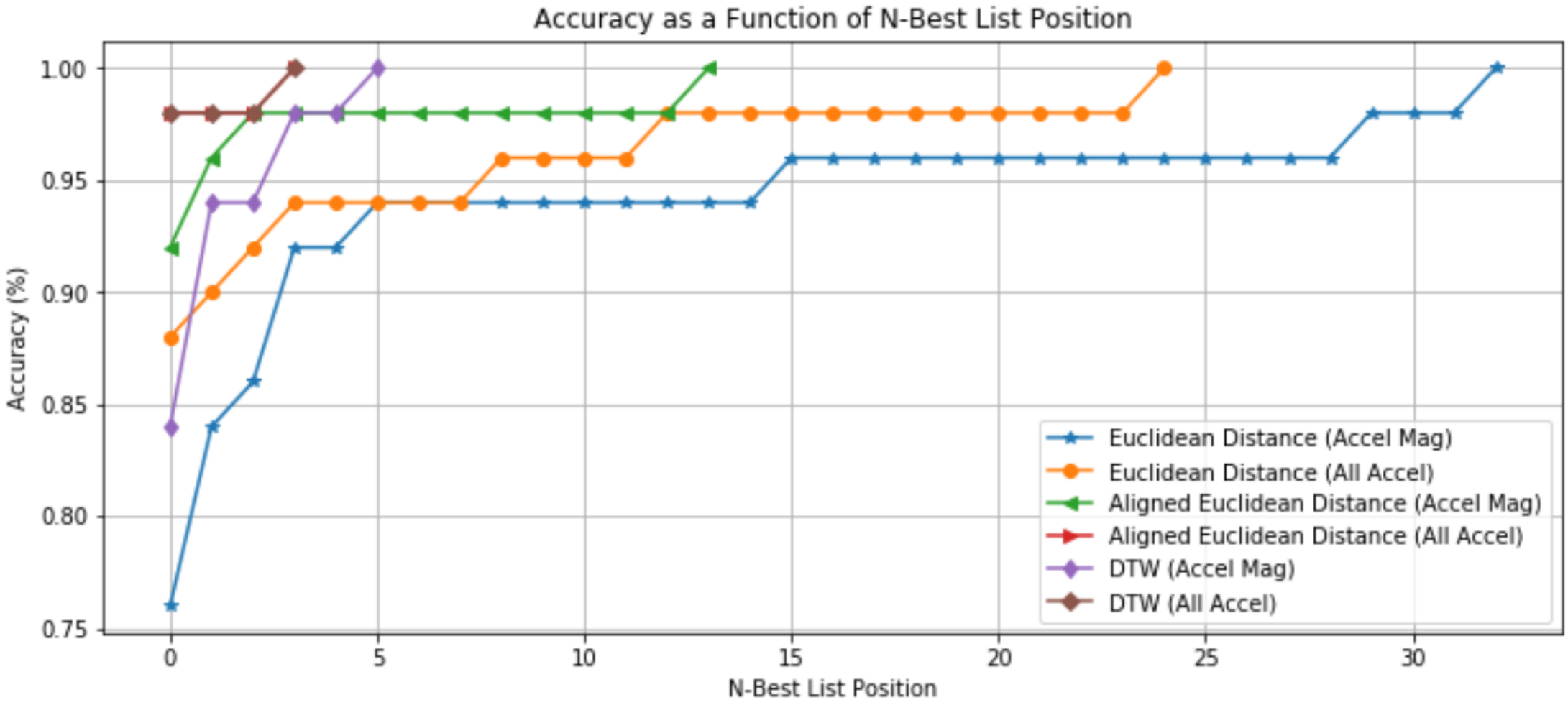
This is on the JonGestureLogs from https://github.com/jonfroehlich/CSE590Sp2018/tree/master/A02-GestureRecognizer



ML EVALUATION

N-BEST LIST: SHAPE-MATCHING GESTURE REC

This is on the JonGestureLogs from https://github.com/jonfroehlich/CSE590Sp2018/tree/master/A02-GestureRecognizer



ML EVALUATION

N-BEST LIST EXAMPLE 2

Graphing average recognition scores as a function of N-best list index. Prefer a dramatic dropoff between 1st and 2nd position, indicating a good separation of scores.

Labeling Images with a Computer Game

Luis von Ahn and Laura Dabbish
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA, USA
{biglou,dabbish}@cs.cmu.edu

Abstract

We introduce a new interactive system: a game that is fun and can be used to create valuable output. When people play the game they help determine the contents of images by providing meaningful labels for them. If the game is played as much as popular online games, we estimate that most images on the Web can be labeled in a few months. Having proper labels associated with each image on the Web would allow for more accurate image search, improve the accessibility of sites (by providing descriptions of images to visually impaired individuals), and help users block inappropriate images. Our system makes a significant contribution because of its valuable output and because of the way it addresses the image-labeling problem. Rather than using computer vision techniques, which don't work well enough, we encourage people to do the work by taking advantage of their desire to be entertained.

Categories & Subject Descriptors: I.2.6 [**Learning**]: Knowledge acquisition. H.3.m [**Information Retrieval**]: miscellaneous. H.5.3 [**HCI**]: Web-based interaction.

General Terms: Design, Human Factors, Languages

Keywords: Distributed knowledge acquisition, image labeling, online games, World Wide Web.

INTRODUCTION

Images on the Web present a major technological challenge. There are millions of them, there are no guidelines about providing appropriate textual descriptions for them, and computer vision hasn't yet produced a program that can determine their contents in a widely useful way. However, accurate descriptions of images are required by several applications like image search engines and accessibility programs for the visually impaired. Current techniques to categorize images for these applications are insufficient in many ways, mostly because they assume that the contents of images on the Web are related to the text appearing in the page. This is insufficient because the text adjacent to the images is often scarce, and can be misleading or hard to process [4].

CHI 2004, *April 24–29, 2004, Vienna, Austria.*

The only method currently available for obtaining precise image descriptions is manual labeling, which is tedious and thus extremely costly. But, what if people labeled images without realizing they were doing so? What if the experience was enjoyable? In this paper we introduce a new interactive system in the form of a game with a unique property: the people who play the game label images for us.

The labels generated by our game can be useful for a variety of applications. For accessibility purposes, visually impaired individuals surfing the Web need textual descriptions of images to be read aloud. For computer vision research, large databases of labeled images are needed as training sets for machine learning algorithms. For image search over the Web and inappropriate (e.g., pornographic) content filtering, proper labels could dramatically increase the accuracy of current systems.

We believe our system makes a significant contribution, not only because of its valuable output, but also because of the way it addresses the image-labeling problem. Rather than making use of computer vision techniques, we take advantage of people's existing perceptual abilities and desire to be entertained.

Our goal is ambitious: to label the majority of images on the World Wide Web. If our game is deployed at a popular gaming site like Yahoo! Games and if people play it as much as other online games, we estimate that most images on the Web can be properly labeled in a matter of weeks. As we show below, 5,000 people continuously playing the game could assign a label to all images indexed by Google [10] in 31 days.

We stress that our method is not necessarily meant to compete against the other techniques available for handling images on the Web. The labels produced using our game can be combined with these techniques to provide a powerful solution.

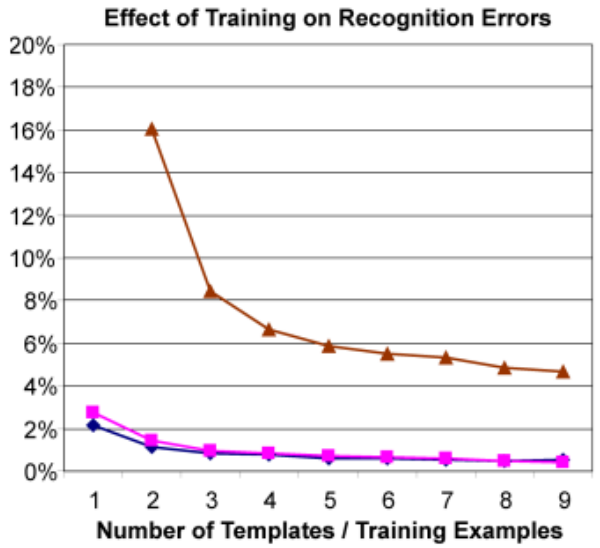
The Open Mind Initiative

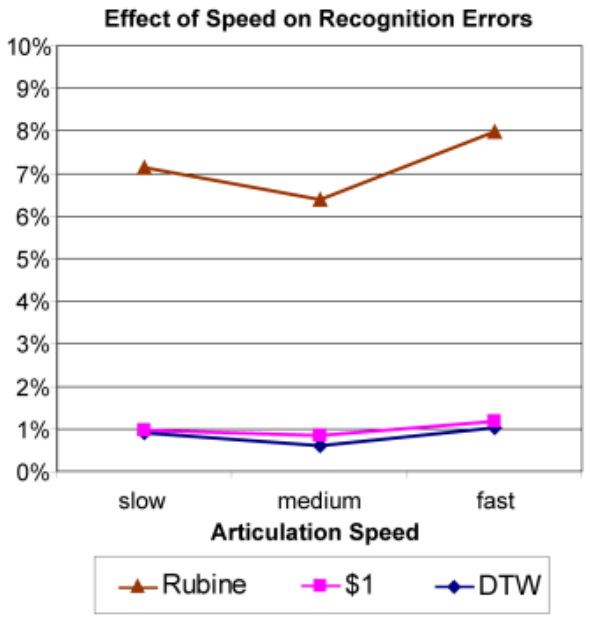
Our work is similar in spirit to the Open Mind Initiative (e.g., [18,19]), a worldwide effort to develop "intelligent" software. Open Mind collects information from regular Internet users (referred to as "netizens") and feeds it to machine learning algorithms. Volunteers participate by answering questions and teaching concepts to computer programs. Our method is similar to Open Mind in that we plan to use regular people on the Internet to label images for us. However, we put greater emphasis on our method being fun because of the scale of the problem that we want

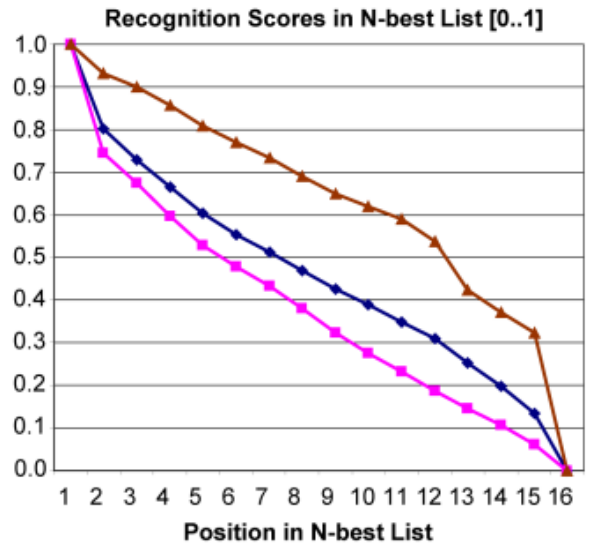
Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes

Wobbrock *et al.*, UIST'07

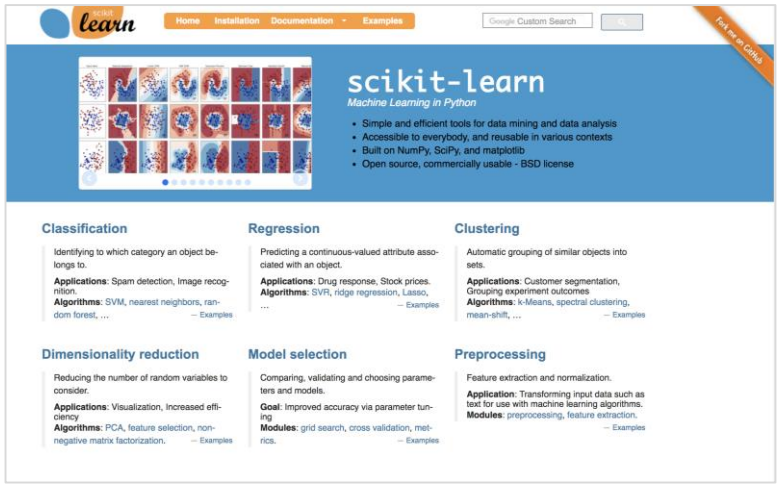
744 citations







ML TOOLKITS: LANGUAGES AND LIBRARIES

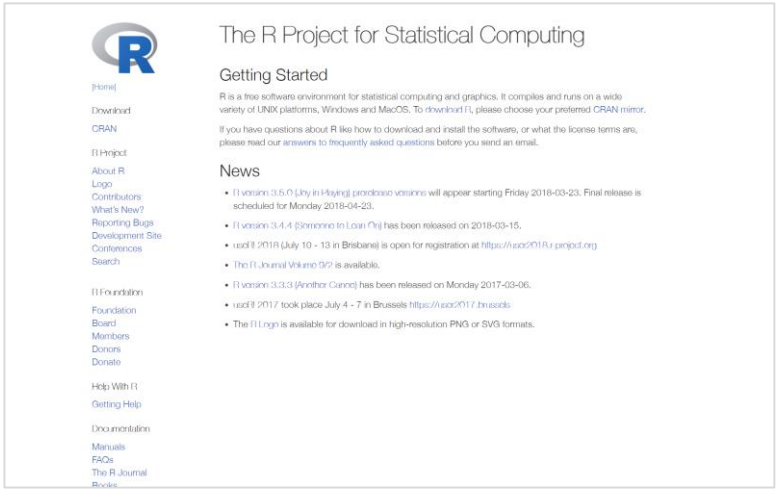


Sci-Kit Learn

Python-based library for classification, regression, clustering, dimensionality reduction, model selection, preprocessing data, etc.

Weka

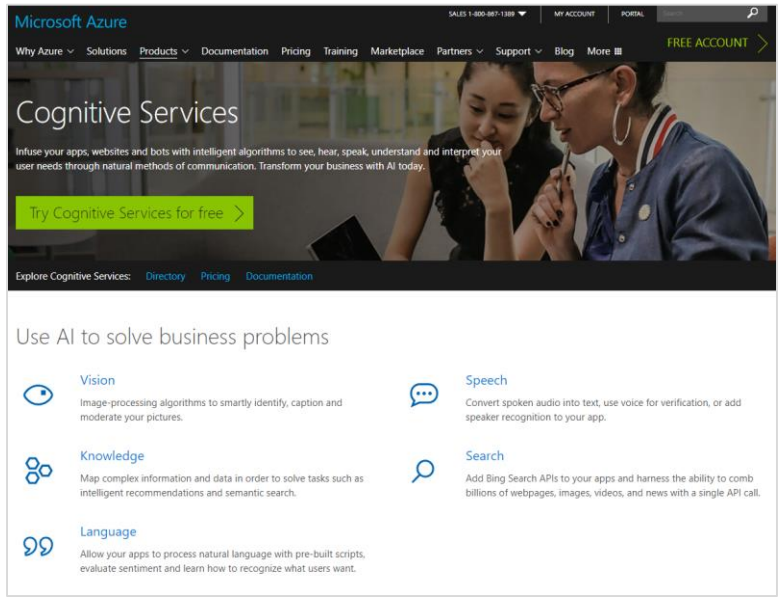
Java-based library for pre-processing, classification, regression, clustering, and visualizing data. Used to be the standard. People now shifting to sci-kit learn, R, and cloud-based APIs.



R

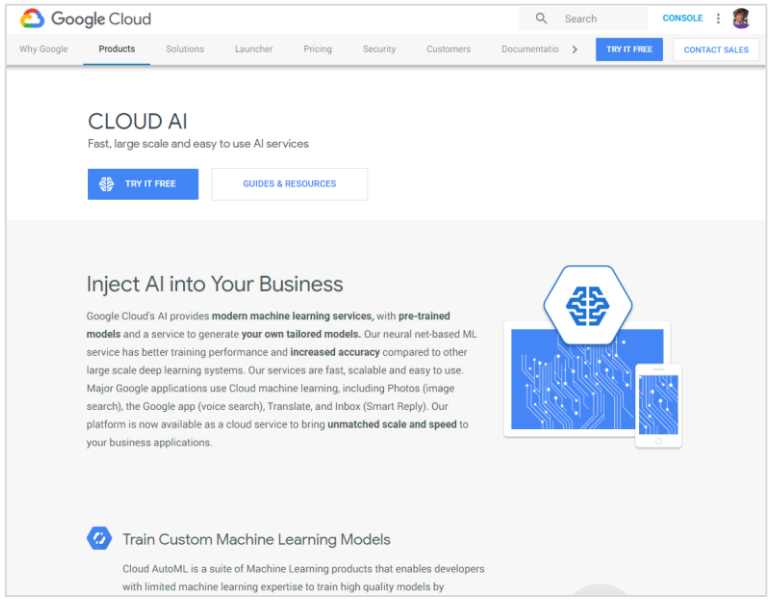
R arose as a statistical computing language but is now widely used in ML and has a number of well-respected ML packages.

ML TOOLKITS: CLOUD APIS



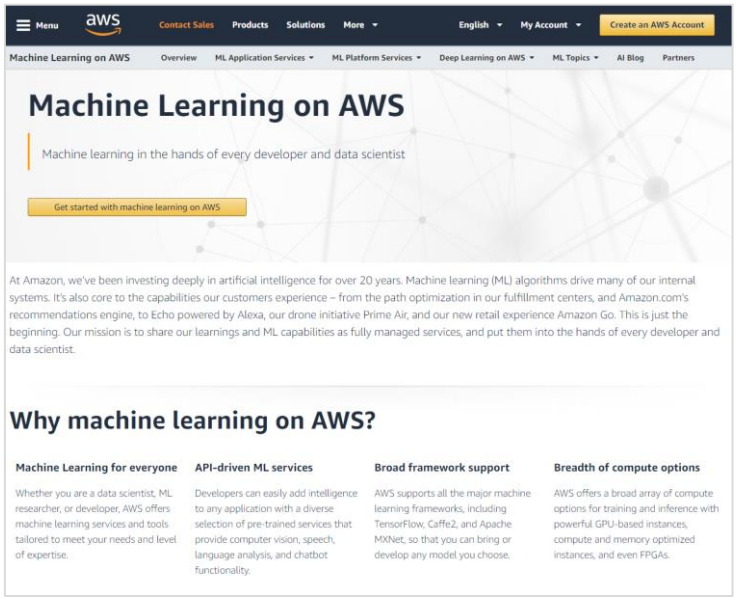
Microsoft Cognitive Services

Cloud API for computer vision, NLP, speech recognition, knowledge representation, chatbots, and search.



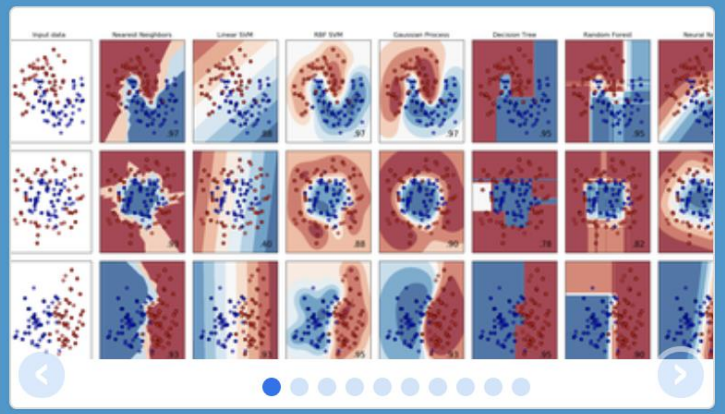
Google Cloud AI

Cloud API for large-scale machine learning, including video and image analysis, speech recognition, speech synthesis, NLP, and language translation



Amazon AWS ML Platform

Cloud API for general ML with image and video analysis, chatbots, and language services such as NLP, translation, transcription, and speech synthesis.



scikit-learn

Machine Learning in Python

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying to which category an object belongs to.

Applications: Spam detection, Image recognition.
Algorithms: SVM, nearest neighbors, random forest, …

— Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.
Algorithms: SVR, ridge regression, Lasso, …

— Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, spectral clustering, mean-shift, …

— Examples

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency
Algorithms: PCA, feature selection, non-negative matrix factorization.

— Examples

Model selection

Comparing, validating and choosing parameters and models.

Goal: Improved accuracy via parameter tuning
Modules: grid search, cross validation, metrics.

— Examples

Preprocessing

Feature extraction and normalization.

Application: Transforming input data such as text for use with machine learning algorithms.
Modules: preprocessing, feature extraction.

— Examples

ML TOOLKITS

SCIKIT-LEARN FEATURES INCLUDE

FEATURE EXTRACTION, SELECTION, & REDUCTION

Feature Extraction: helper methods for extracting features from image and text data.

Feature Selection: for identifying meaningful attributes from which to create supervised models.

Dimensionality Reduction: for reducing the number of features (*e.g.,* Principal component analysis)

ML MODELS

Clustering Algorithms: for grouping unlabeled data such as KMeans.

Supervised Models: a vast array of supervised models, including naive bayes, neural networks, SVMs and decision trees.

Ensemble Methods: for combining the predictions of multiple supervised models.

EVALUATION

Cross Validation: for estimating the performance of supervised models on unseen data.

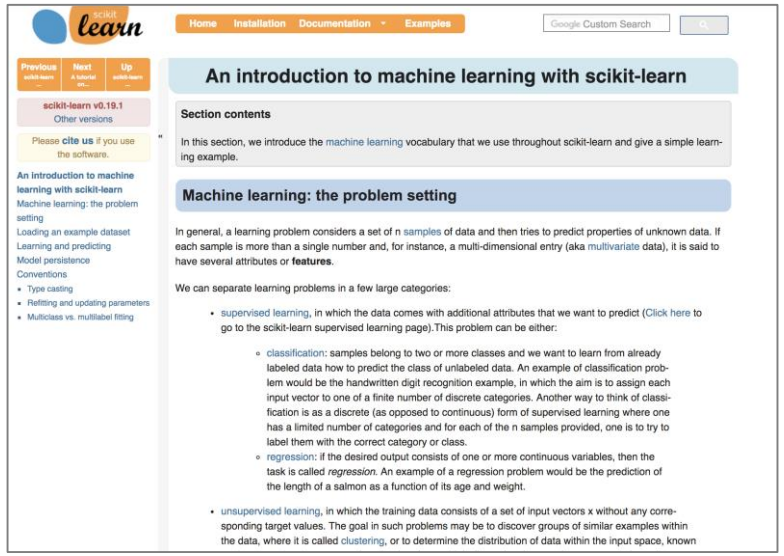
Parameter Tuning: for getting the most out of supervised models.

Datasets: for test datasets and for generating datasets with specific properties for investigating model behavior.

Source: http://scikit-learn.org/; https://machinelearningmastery.com/a-gentle-introduction-to-scikit-learn-a-python-machine-learning-library/

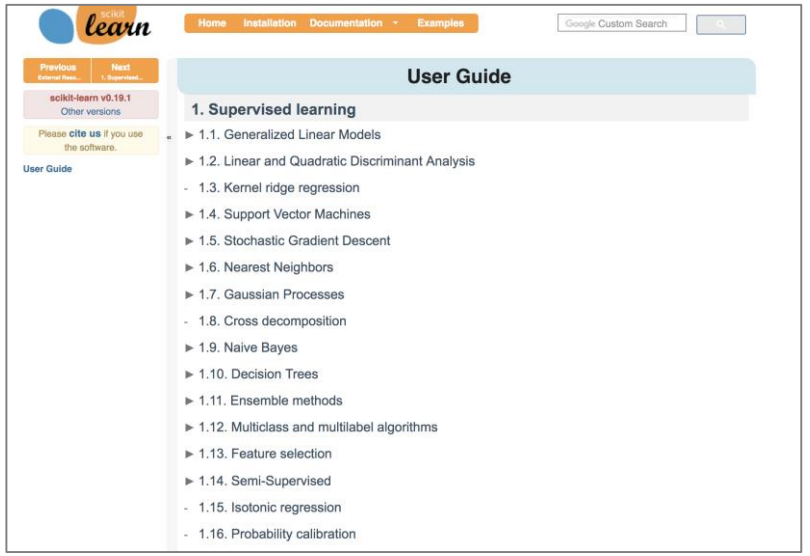
ML TOOLKITS

GETTING STARTED



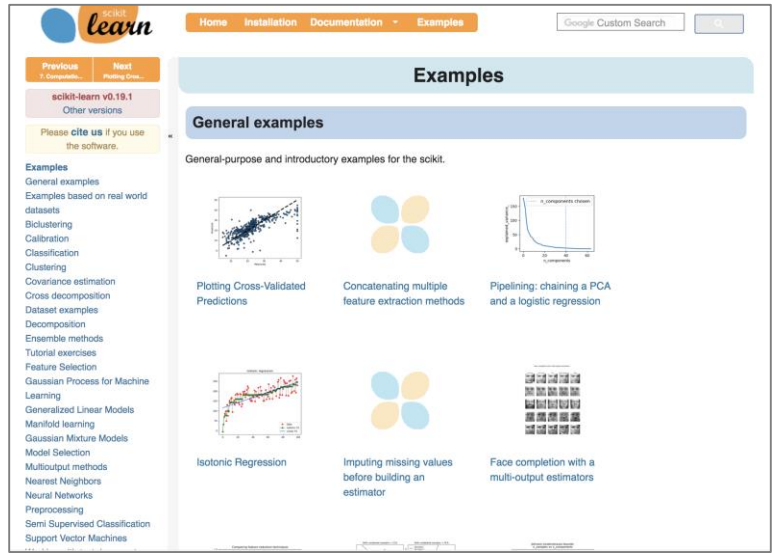
Quick Start Tutorial

Introduces some basic vocabulary and walks through a simple example



User Guide

Goes through each major component of the sci-kit library from supervised and unsupervised models to model selection and evaluation.



Examples

Lots of different examples of using, visualizing, and evaluating models

ML MODEL

SVM

SVM is very similar to the Apple / Oranges and Dog / Cat example

Linear classifier

Can be slow to train but fast to use

We'll watch a short video

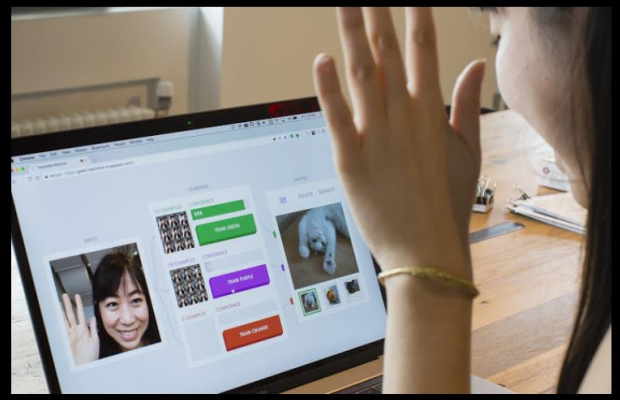
Support Vector Machines

A Visual Explanation + Sample Python Code

https://github.com/adashofdata

Explore machine learning in simple, hands-on ways.
Read more

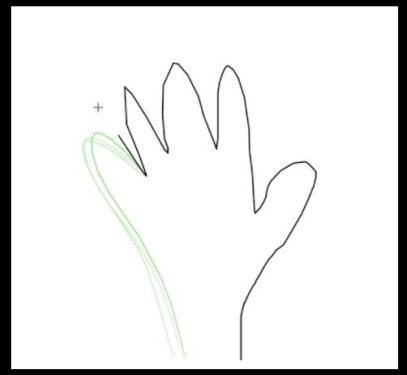
FEATURED EXPERIMENTS



TEACHABLE MACHINE
by Google Creative Lab

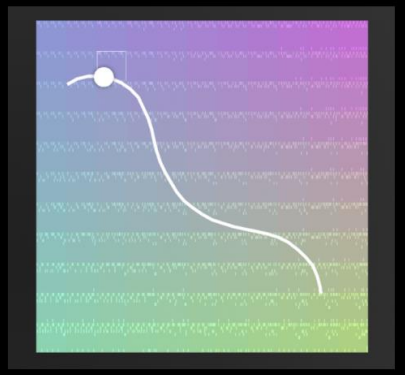
NSYNTH SUPER
by Magenta / Google Creative Lab

BROWSE TAGS ▾ SORT BY: NEWEST ▾



SCRYING PEN
by Andy Matuschak

EMOJI SCAVENGER HUNT
by Google Brand Studio



BEAT BLENDER
by Creative Lab + Magenta

RESOURCES

FIND OUT MORE: ML RESOURCES

BASIC OVERVIEWS

Machine Learning 101
Jason Mayes, Senior Creative Engineer Google
A intro slidedeck that overviews ML approaches and methods

The 7 Steps of Machine Learning
Yufeng Guo, Google Cloud
A short article and video about the ML pipeline

COURSES / IN-DEPTH GUIDES

Machine Learning Mastery
Jason Brownlee, PhD
A free step-by-step guide on many aspects of ML.

Supervised Learning with Sci-kit Learn
Datacamp.com
A paid course (1st chapter free) on using sci-kit learn

Coursera's Machine Learning Course
Andrew Ng, former Stanford Prof & Google Brain head
The most popular MOOC on ML of all-time.

MATHEMATICS OF ML

Machine Learning: A Probabilistic Perspective
Kevin Murphy, former UBC Prof, now at Google
An in-depth introduction to probabilistic and inference models

Elements of Statistical Learning
Hastie, Tibshirani, & Friedman
Recommended by Machine Learning Mastery