A3 REFLECTION + INTRO TO MODEL-BASED

CSE 599 Prototyping Interactive Systems | Lecture 14 | May 1

Jon Froehlich • Jasper O'Leary (TA)

DESIGN USE BUILD

MAKEABILITY LAB

PAUL G. ALLEN SCHOOL
OF COMPUTER SCIENCE & ENGINEERING

UNIVERSITY of WASHINGTON

A3: Signal Processing + Machine Learning 1: Offline Gesture Recognizer

Published | Edit

Related Items
SpeedGrader™

Overview

Imagine working for a new hardware startup designing new input controllers. You've been asked to prototype a new, custom input device with gesture recognition (using an accelerometer)--for example, to use the input controller as a paddle in tennis, as a ball in bowling or to recognize the "overhand throwing motion" in baseball. In this assignment you will build your own 3D gesture recognizer to automatically recognize these gestures.

A3 SHARE OUTS!

While in the "real world" you would ultimately need to create a real-time gesture recognizer, for this assignment you will make an *offline* version in Jupyter Notebook (we strongly recommend the Anaconda Distribution). Specifically, you will make two recognizers:

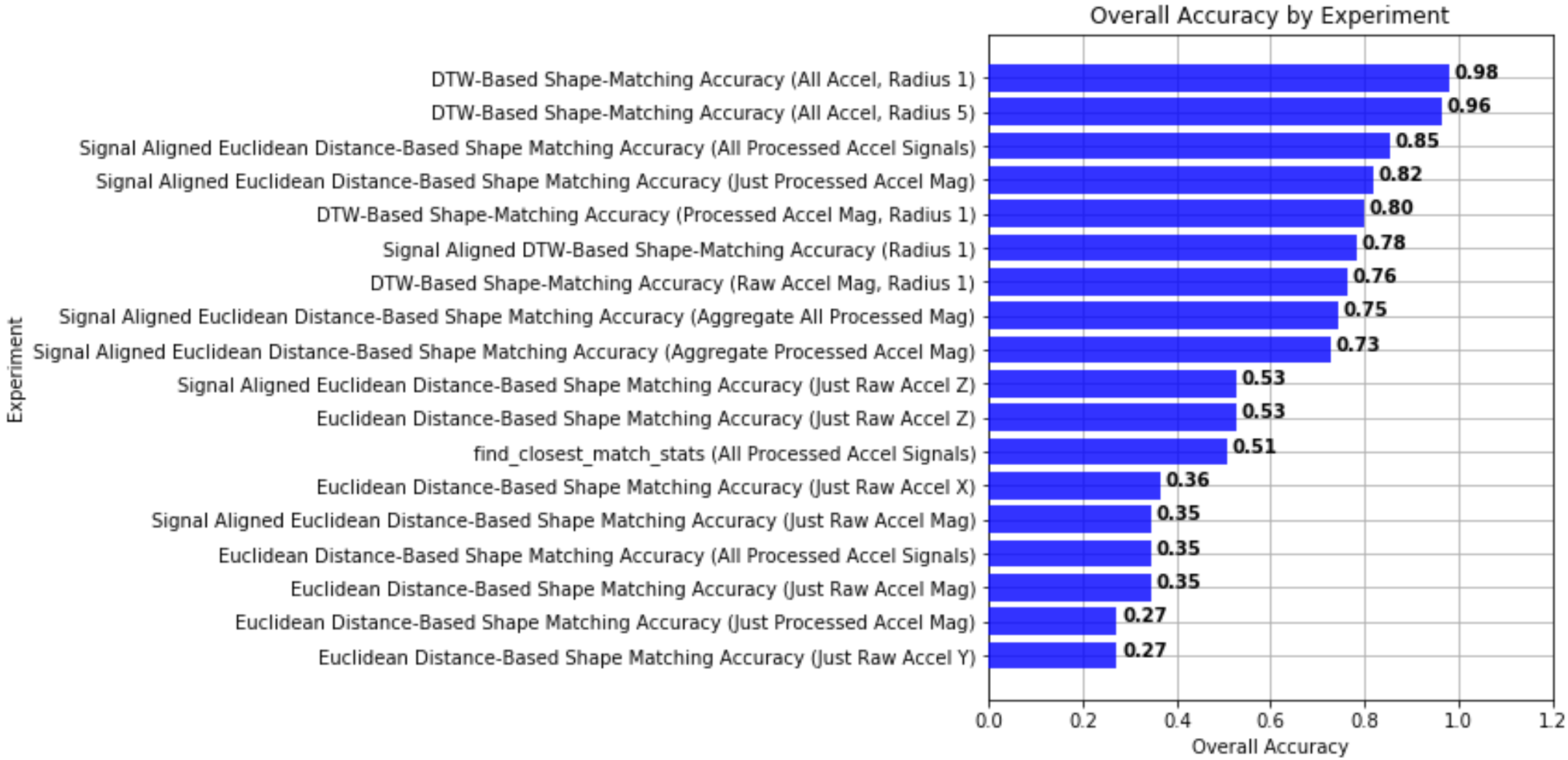
- (i) a *shape-matching* recognizer such as via a Euclidean distance metric or Dynamic Time Warping and
- (ii) a *feature-based* (or *model-based*) recognizer using a support-vector machine (SVM) (recommended) or an alternative supervised learning approach of your choosing (*e.g.*, an HMM).

Within Jupyter Notebook, we will use Python 3 and these amazing libraries numpy, scipy, matplotlib, and scikit-learn. Numpy and scipy provide numeric array handling and signal processing, matplotlib provides visualization, and scikit-learn is the de facto machine learning library in Python. You are welcome to use other libraries as well (*e.g.*, this DTW library).

For your deliverables, you will turn in your Jupyter Notebook, your recorded gestures, and a slide deck

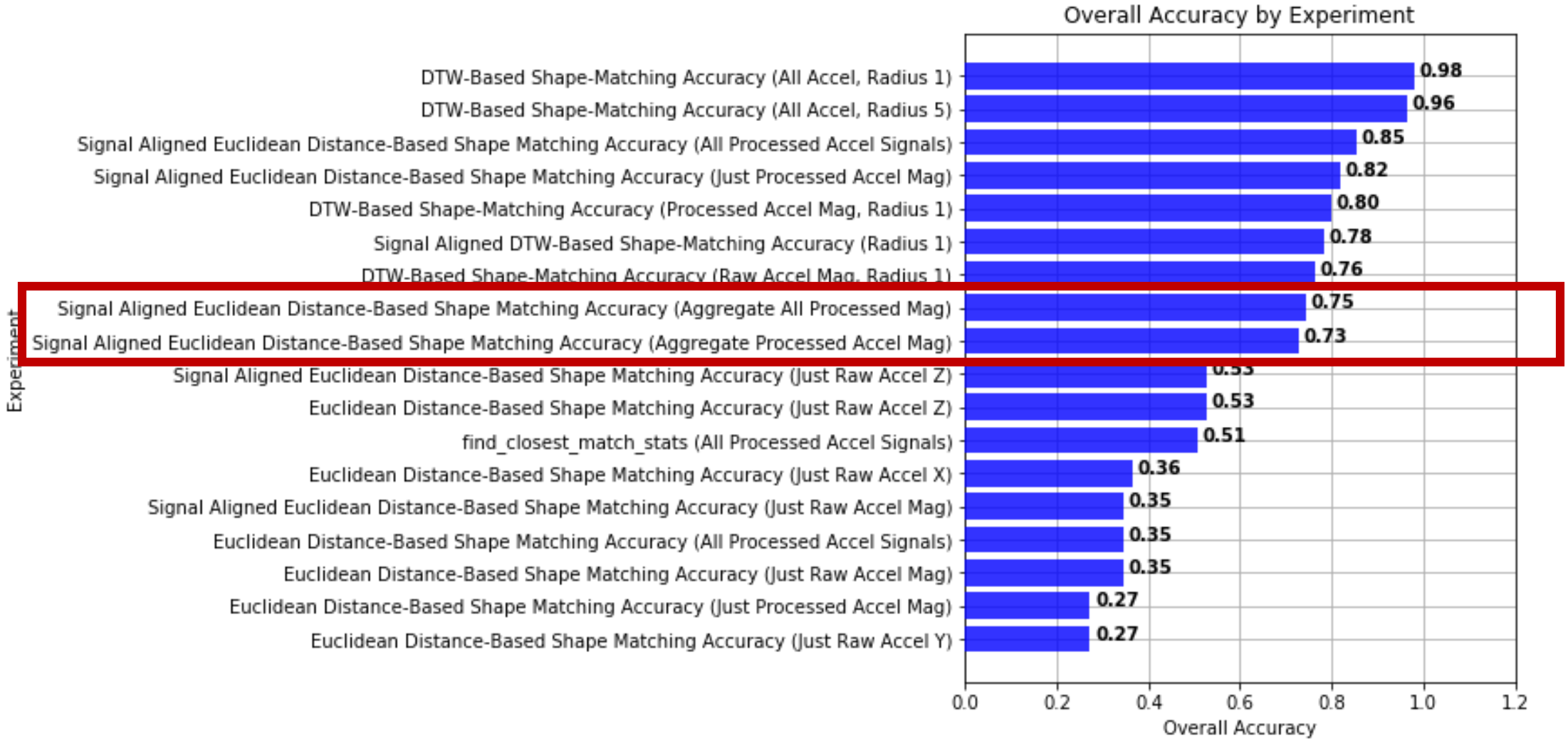
A3: REFLECTION

BEST PERFORMING ON MY “HARD” GESTURE SET



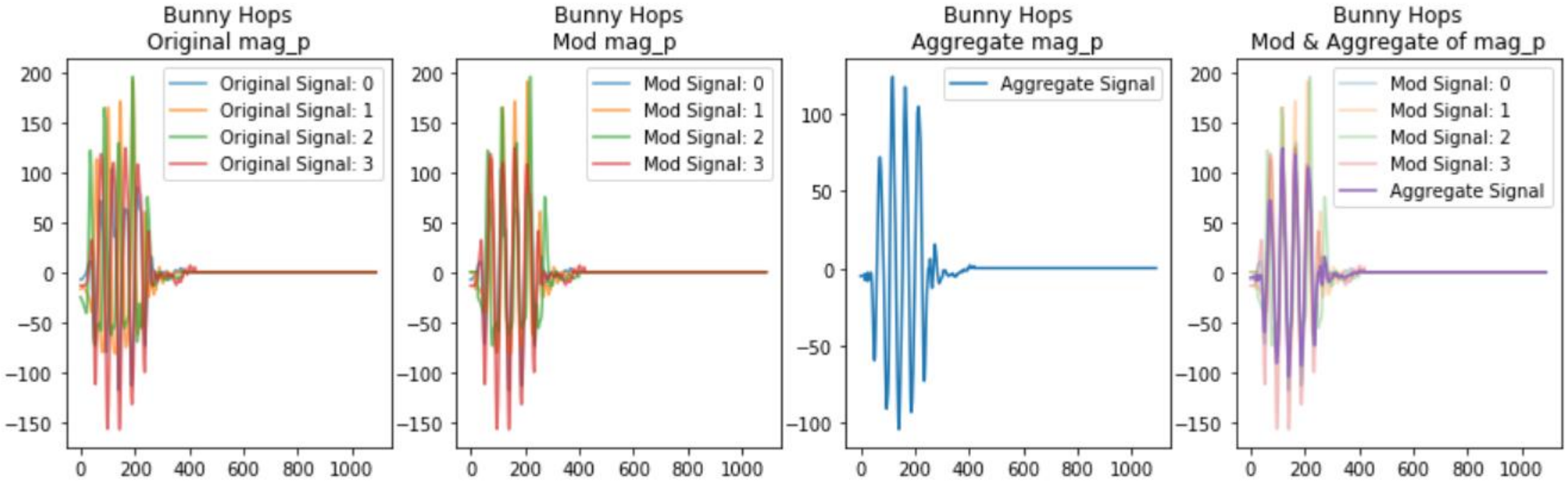
A3: REFLECTION

BEST PERFORMING ON MY “HARD” GESTURE SET



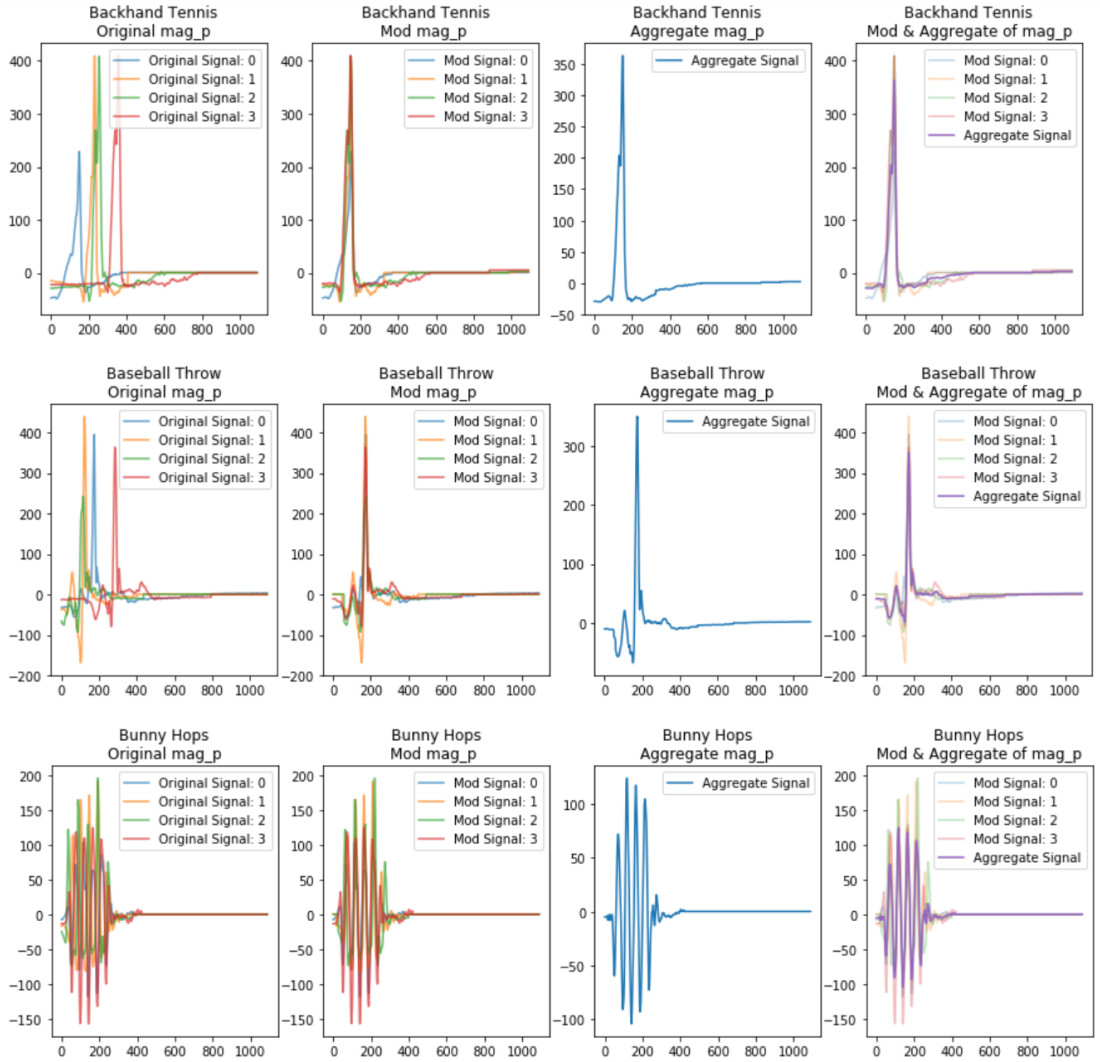
A3: REFLECTION

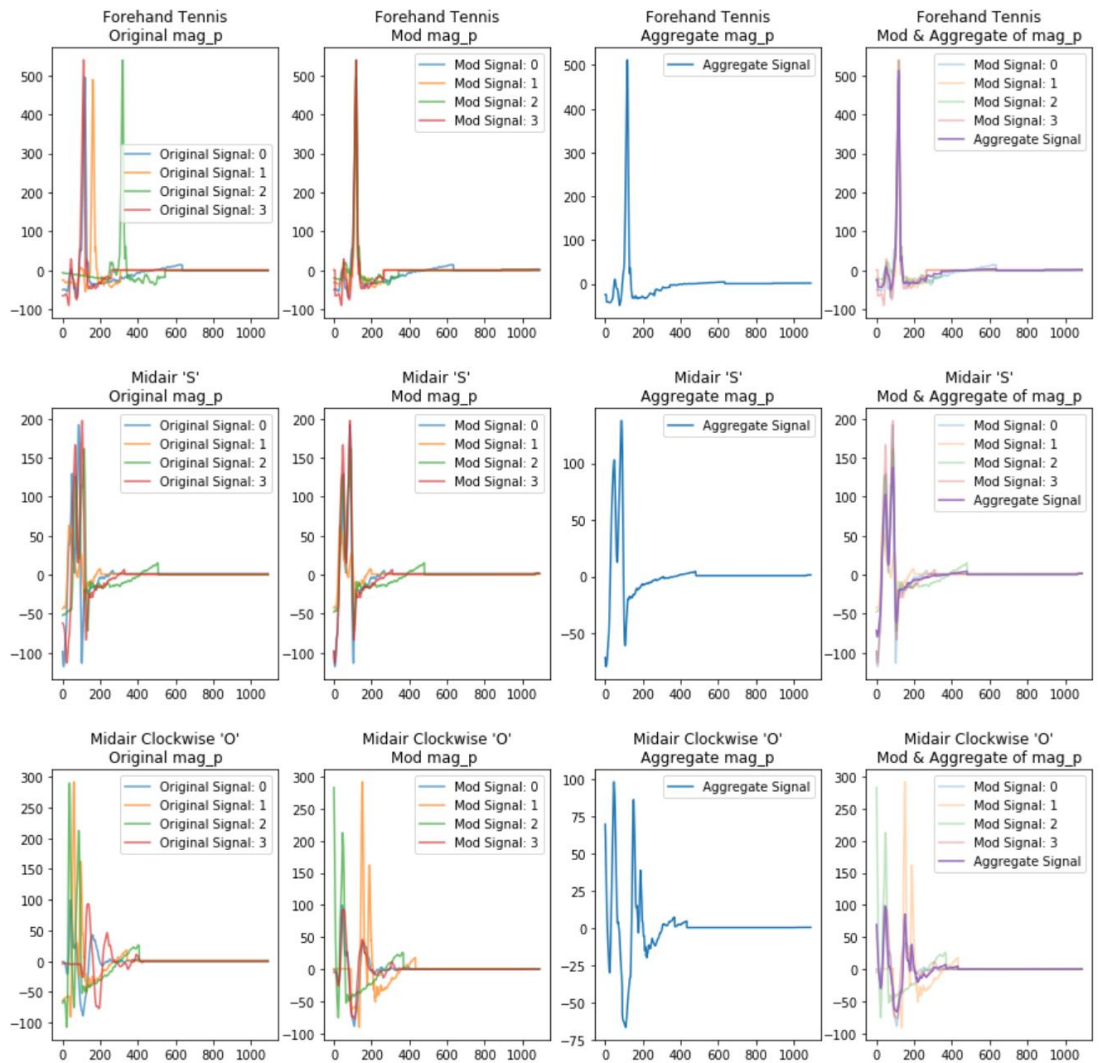
CREATING AGGREGATE MODEL FROM TRAINING FOLDS



A3: REFLECTION

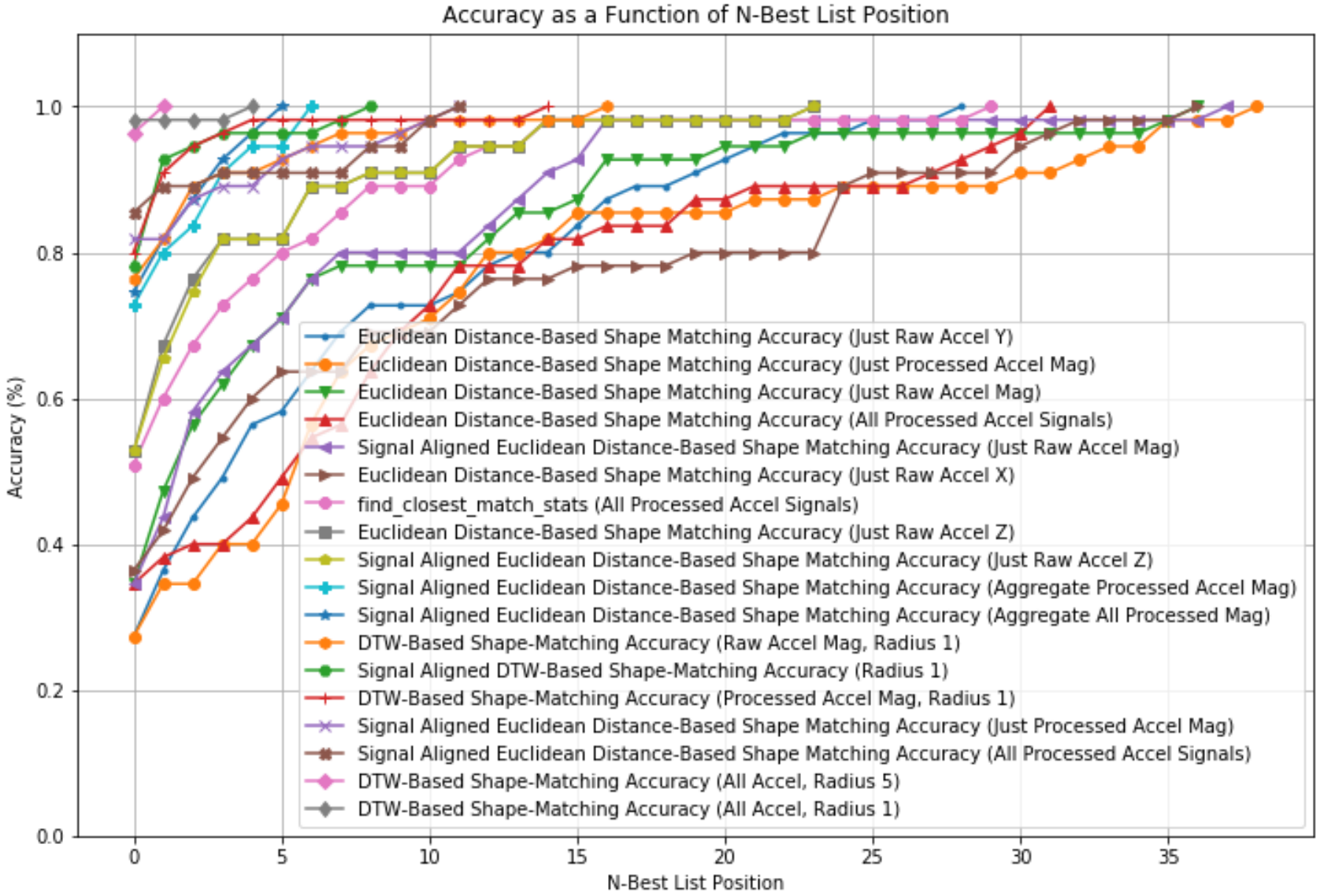
CREATING AGGREGATE MODEL FROM TRAINING FOLDS



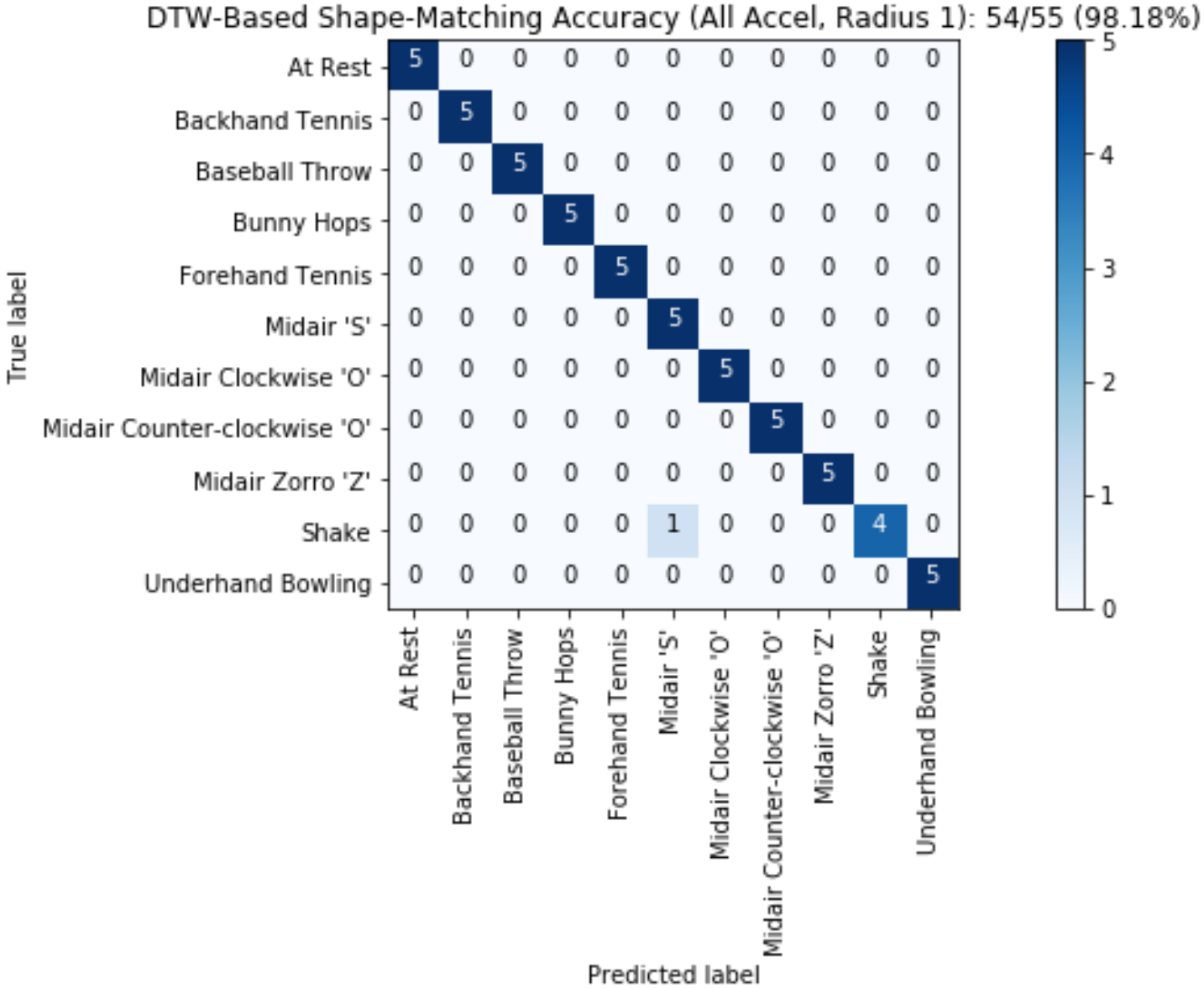


A3: REFLECTION

BEST PERFORMING ON MY "HARD" GESTURE SET

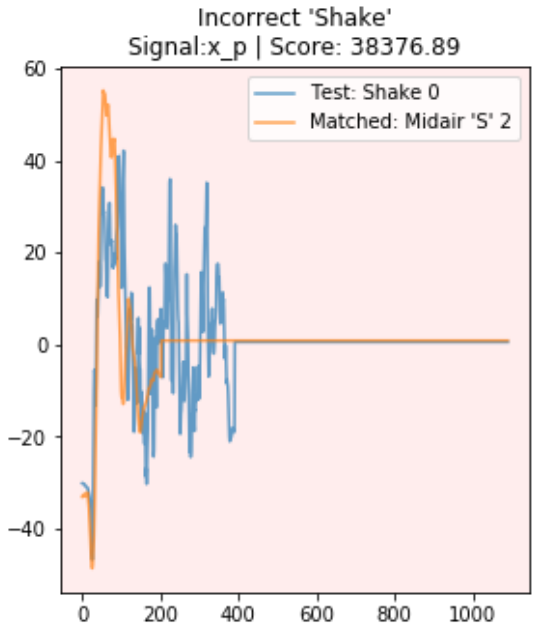


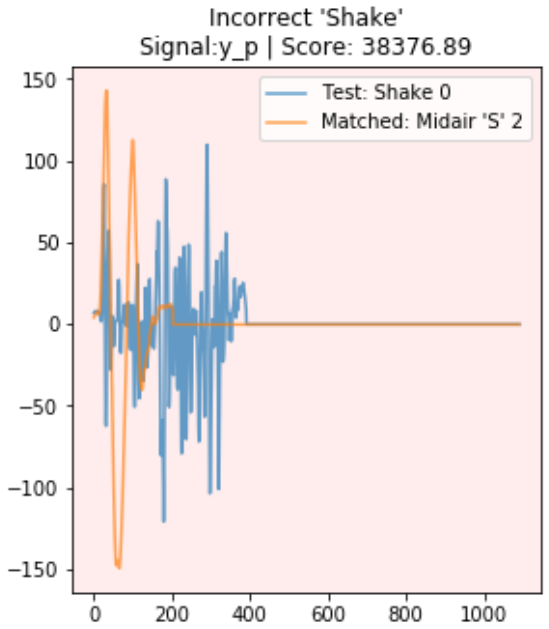
CONFUSION MATRIX FOR CLASSIFYING ‘HARD’ GESTURE SET

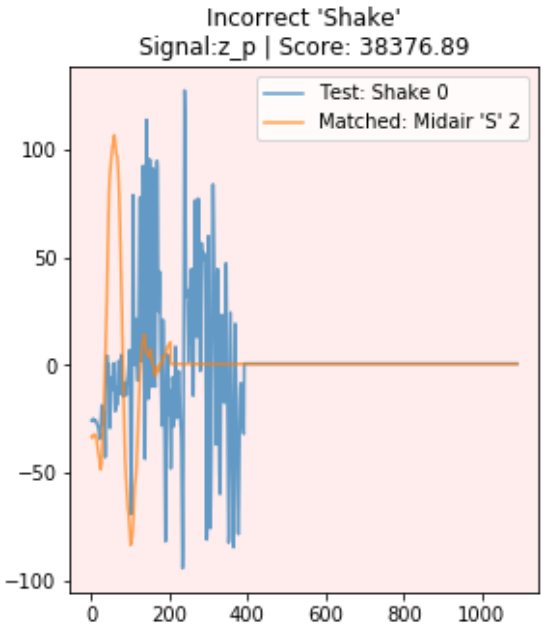


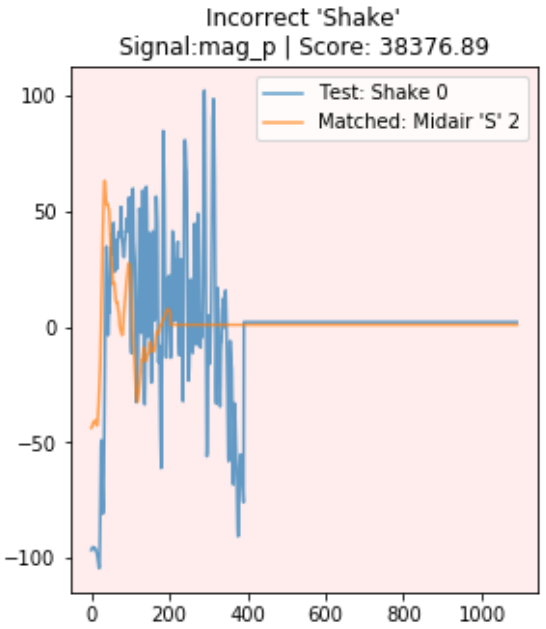
A3: REFLECTION

VISUALIZING THE MISTAKE... CORRECTABLE?



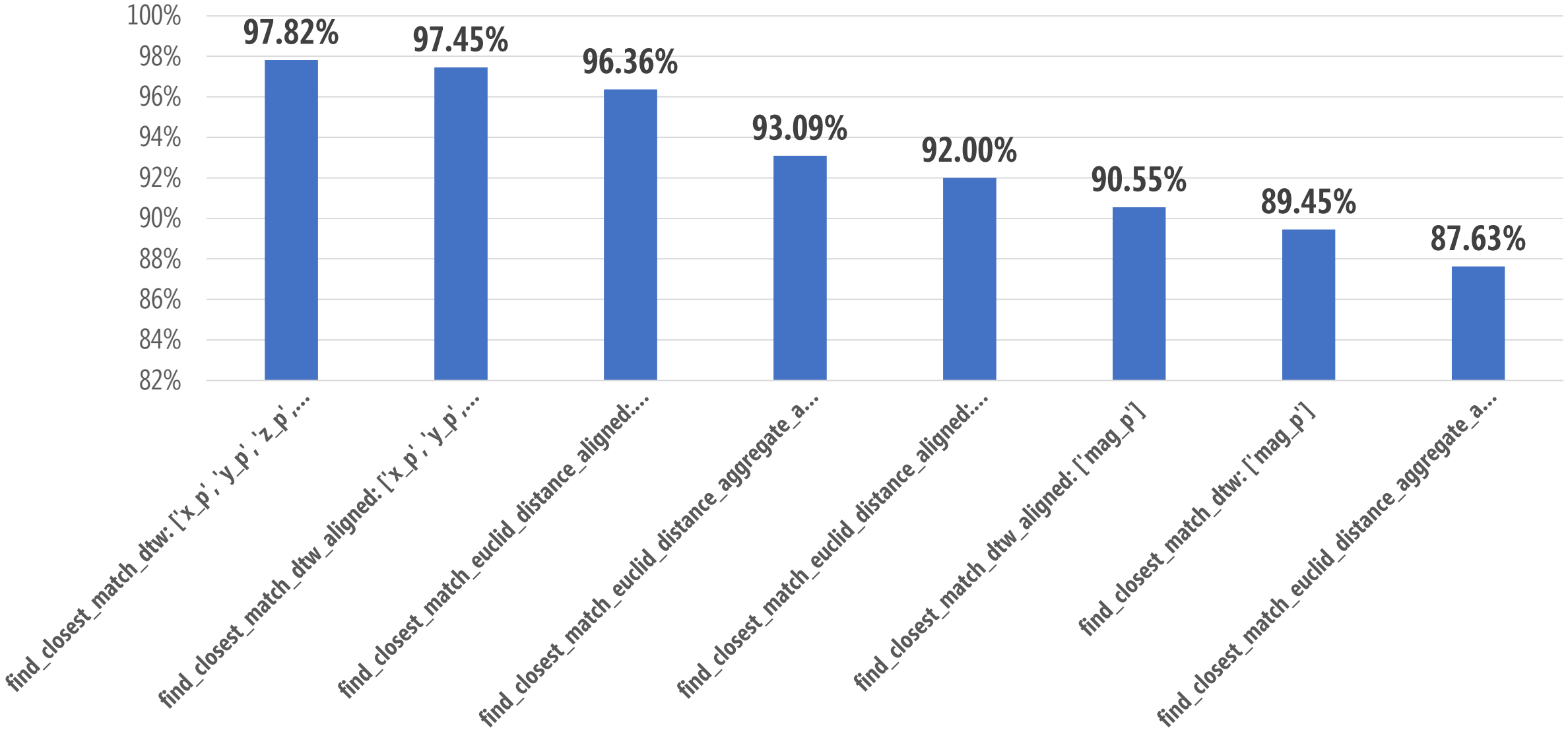






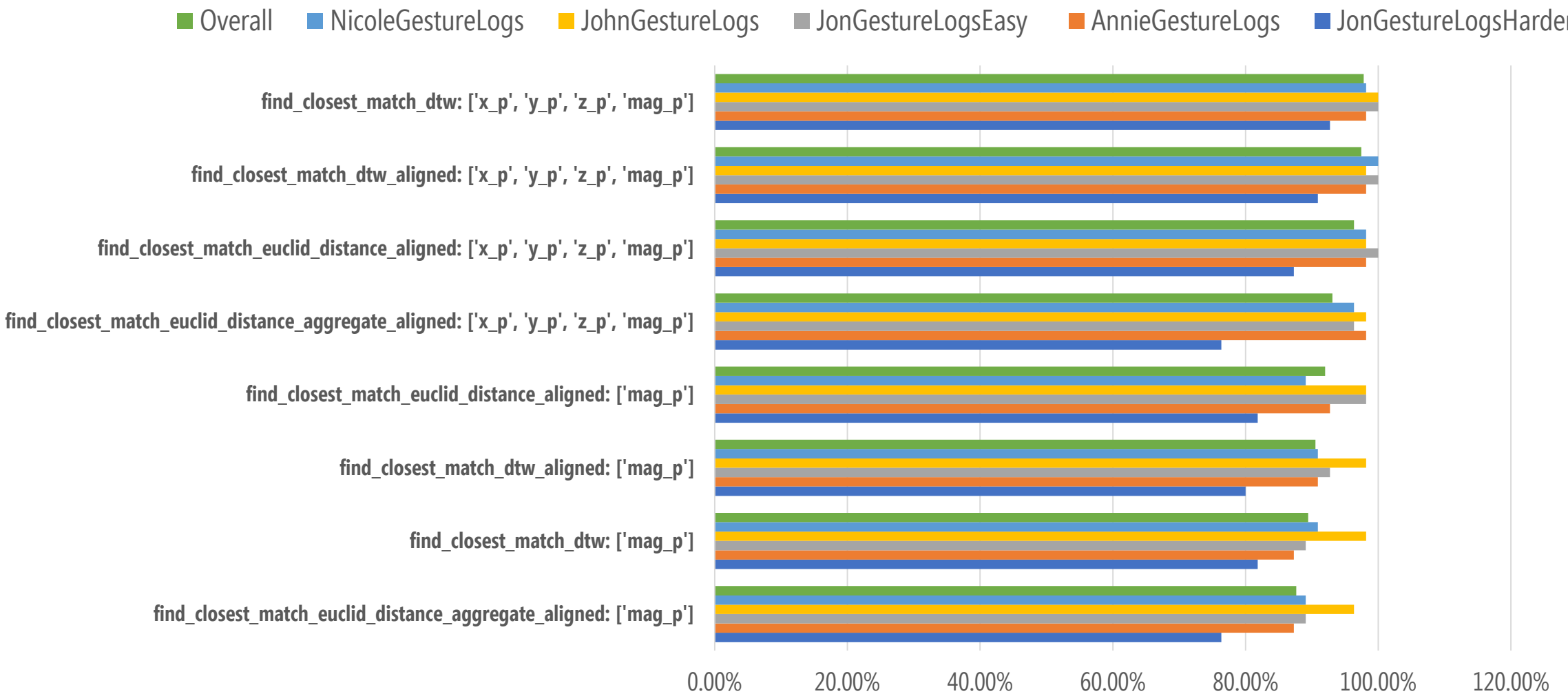
A3: REFLECTION

OVERALL BEST ACROSS 3 STUDENTS + MY 2 GESTURE SETS



A3: REFLECTION

OVERALL BEST ACROSS 3 STUDENTS + MY 2 GESTURE SETS



TODAY

A3 REFLECTION + INTRO TO MODEL-BASED

A3 reflection. What worked? What didn't? Challenges? Solutions?

Quick intro to **model-based approaches**

Jupyter Notebook **exercises** with SVMGestureRecognizer.ipynb

A4: Signal Processing + Machine Learning 2: Model-Based Gesture Recognizer

Published | Edit

Overview

This assignment directly builds on A3: you will build a *feature-based* (or *model-based*) recognizer using a support-vector machine (SVM) (recommended) or an alternative supervised learning approach of your choosing (*e.g.*, an HMM). You will also compare your results to the shape-matching algorithm from A3. At a minimum, you must test on my gesture set (11 gestures) and your gesture set (11 gestures). You need to upload your gesture set to this Google Drive folder as part of the assignment.

To get started, download and open this bas[illegible] ve provided you with some parsing code, data structures, and so[illegible] ve also built the full training+test pipeline for training and testing the SVM using k-fold cross validation. Note: for the file handling code to work, you must have a folder called 'GestureLogs' in the same dir as this Notebook. Within 'GestureLogs', place folders of your gesture sets (at the very least, mine and yours but you could also download others from the Google Drive)

You need to decide how to incorporate your A3 shape-matching approach into this notebook (*e.g.*, you could make a copy of your A3 assignment notebook and then copy over relevant SVM code or you could copy over your A3 code into this notebook).

Learning Goals

- Reinforce signal processing skills from A2 and complement them frequency signal analysis
- Introduce and learn feature extraction and feature classification concepts
- Introduce and learn notions of training + test and reinforce use of k-fold cross validation

Parts

INTRO TO A4

A4: MODEL-BASED CLASSIFICATION

MODEL-BASED RECOGNIZER

Drastically reduced scope of A4 to help you focus on your projects. No longer need to do event segmentation. I also added a full end-to-end pipeline to train and test your SVM.

Build a model-based (aka feature-based) recognizer using a Support Vector Machine

Compare your shape-matching results to this new approach

Upload your gesture set to the Google Drive

The SVM

In [9]:

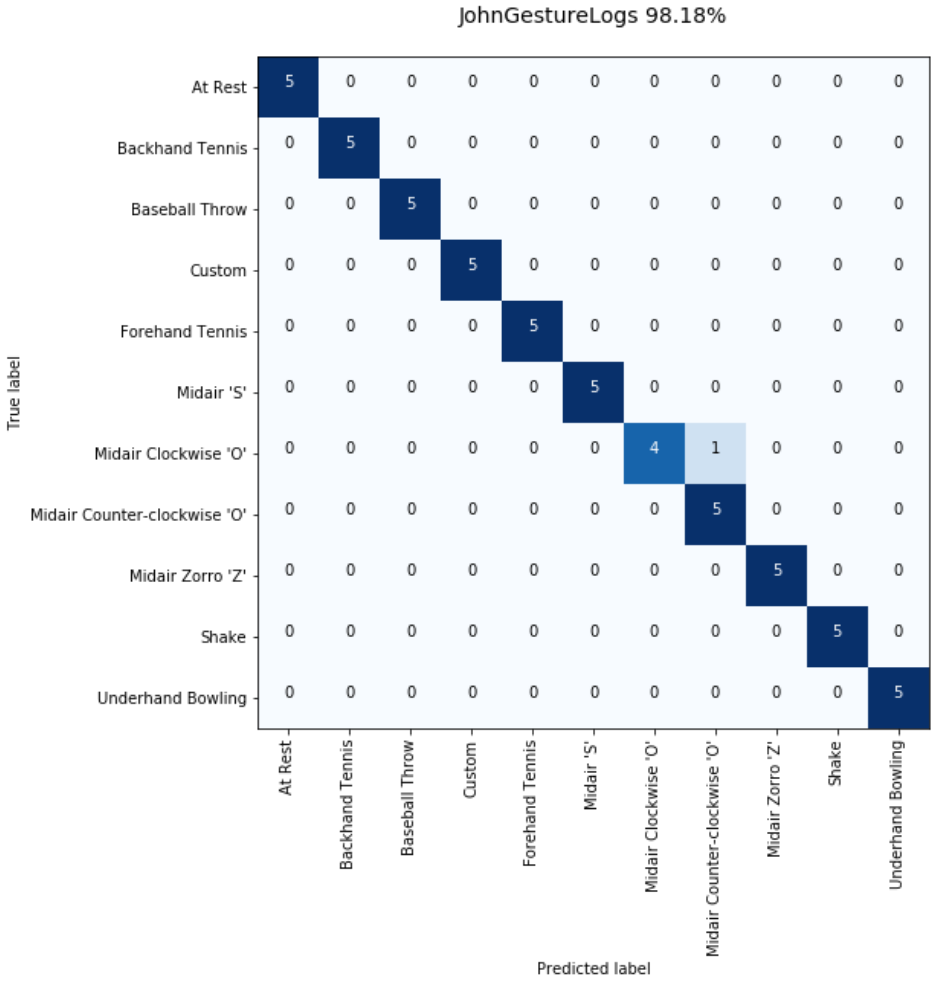
```
# This is the simplest possible SVM using only a few features but gives you a sense of the overall approach
# Some nice resources:
#  - A very simple classification example using scikit:
#     https://dbaumgartel.wordpress.com/2014/03/10/a-scikit-learn-example-in-10-lines/
#  - A nice video overview of SVM: https://youtu.be/N1vOgolbjSc
#  - Official sci-kit Learn: http://scikit-Learn.org/stable/modules/svm.html

from sklearn import svm
import itertools

# Returns a feature vectof for the given trial
def extract_features_example(trial):

    # Play around with features to extract and use in your model
    # Brainstorm features, visualize ideas, try them, and iterate
    features = []
    features.append(trial.accel.mag.max())
    features.append(np.std(trial.accel.mag))
    return features

numFolds = 5
selected_gesture_set = get_gesture_set_with_str("Hard")
numGestures = selected_gesture_set.get_num_gestures()
numTrialsTotal = selected_gesture_set.get_total_num_of_trials()

# Setting a seed here keeps producing the same folds set each time. Take that out
# if you want to randomly produce a fold set on every execution
foldToMapGestureToTrial = generate_kfolds(numFolds, selected_gesture_set.map_gestures_to_trials, seed=5)
mapGestureToCorrectMatches = dict()

y_true = []
y_pred = []

gestureNamesSorted = selected_gesture_set.get_gesture_names_sorted()
for gestureName in gestureNamesSorted:
    mapGestureToCorrectMatches[gestureName] = 0

for i in range(0, len(foldToMapGestureToTrial)):
    trainingFolds = foldToMapGestureToTrial.copy()
    testFold = trainingFolds.pop(i)

    trainingData = []
    classLabels = np.array([])

    # build training data for this set of folds
    for trainingFold in trainingFolds:
        for trainingGestureName, trainingTrial in trainingFold.items():
            features = extract_features_example(trainingTrial)
            trainingData.append(features)
            classLabels = np.append(classLabels, trainingGestureName)

    # Here, we train SVM, the 'rbf' kernal is default
    # if you use rbf, need to set gamma and C parameters
    # play around with different kernels, read about them, and try them. What happens?
    # see:
    # - https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html#sphx-glr-auto-examples-svm-plot-rbf-parameter
    # - https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
    clf = svm.SVC(kernel='linear') # kernel='rbf'
    clf.fit(np.array(trainingData), classLabels)

    # make predictions for this test set
    for testGestureName, testTrial in testFold.items():
        features = extract_features_example(testTrial)

        svmPrediction = clf.predict([features])
        y_true.append(testGestureName)
        y_pred.append(svmPrediction)

        if testGestureName == svmPrediction[0]:
            mapGestureToCorrectMatches[testGestureName] += 1

totalCorrectMatches = 0
print("SVM Results:\n")
for gesture in mapGestureToCorrectMatches:
    c = mapGestureToCorrectMatches[gesture]
    print("{}: {}/{} ({}%)".format(gesture, c, numFolds, c / numFolds * 100))
    totalCorrectMatches += mapGestureToCorrectMatches[gesture]

print("\nTotal SVM classifier accuracy {:0.2f}%\n".format(totalCorrectMatches / numTrialsTotal * 100))

cm = confusion_matrix(y_true, y_pred, gestureNamesSorted)
plt.figure(figsize=(10,10))
plot_confusion_matrix(cm, classes=gestureNamesSorted, title='Confusion Matrix')
plt.show()
```

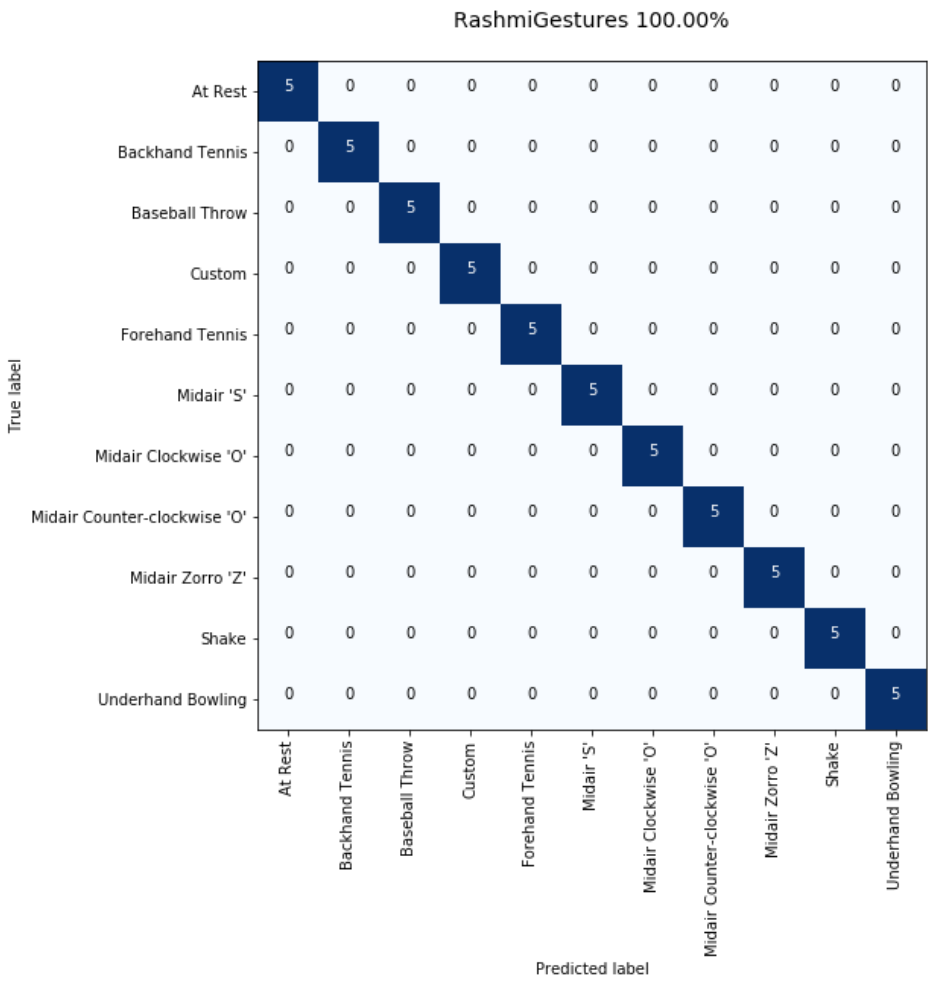
My Drive > CSE599Sp2019-A3GestureSets

Name	Owner	Last modified	File size
AnnieGestureLogs	Anne Ross	May 15, 2019 Anne Ross	–
JohnGestureLogs	John Akers	May 15, 2019 John Akers	–
JonGestureLogs	me	May 15, 2019 me	–
JonGestureLogsHarder	me	May 15, 2019 me	–
NicoleGestures	Nicole Riley	May 15, 2019 Nicole Riley	–
RashmiGestures	Rashmi Mudduluru	May 15, 2019 Rashmi Muddulu...	–

A4: MODEL-BASED GESTURE RECOGNIZER

EXAMPLE SVM RESULTS ON GESTURE LOGS





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.

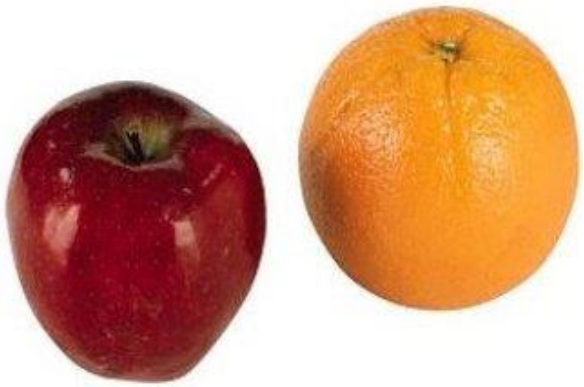
Source: Mayes, Machine Learning 101, (link)

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

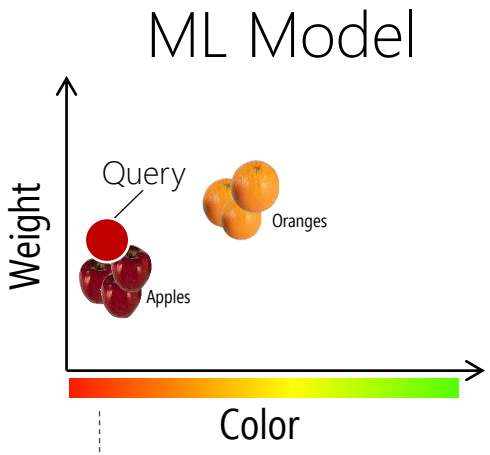
SUPERVISED LEARNING

Input Features

Class Label
(aka Ground Truth Label)

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

Training Data



ML System

Color	Weight	Label
Red	170g	Unknown

Query Data

ML OVERVIEW

SUPERVISED LEARNING

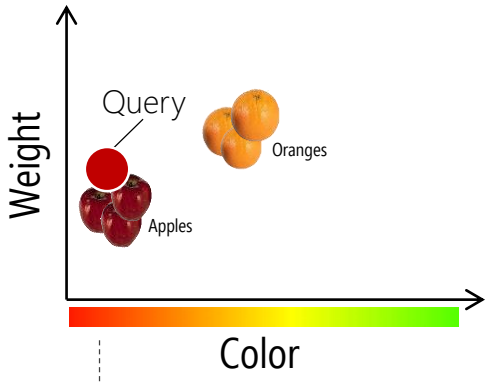
Input Features

Class Label
(aka Ground Truth Label)

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

Training Data

ML Model



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

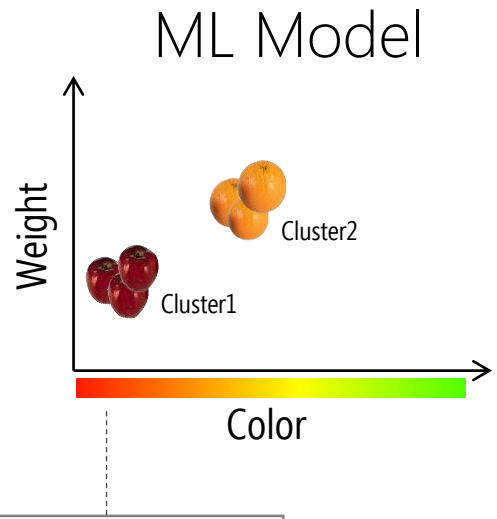
UNSUPERVISED LEARNING

Input Features

No Class Label!
(aka No Ground Truth Labels)

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

Training Data



ML OVERVIEW

UNSUPERVISED LEARNING

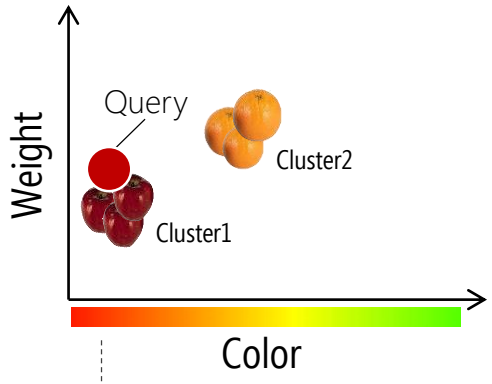
Input Features

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

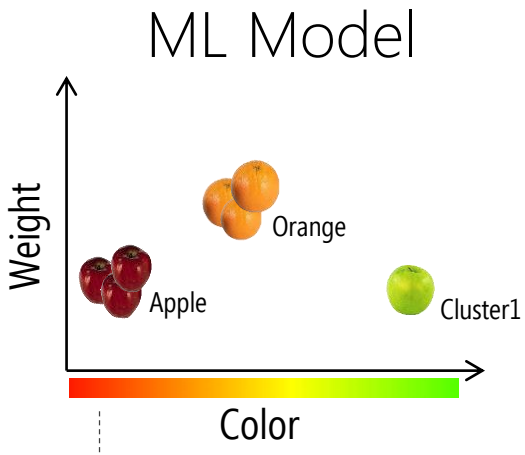
Input Features

Some Class Labels
(aka Ground Truth Labels)

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

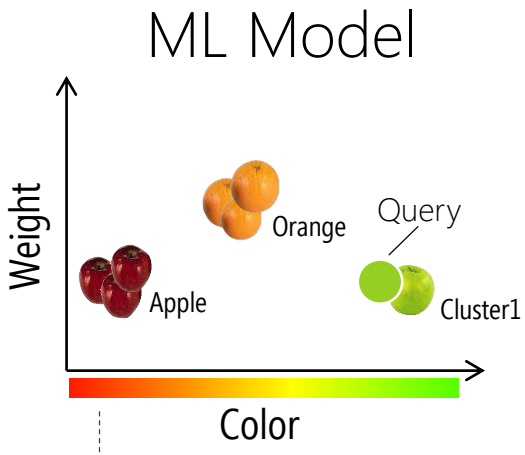
Training Data

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 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	?
...		

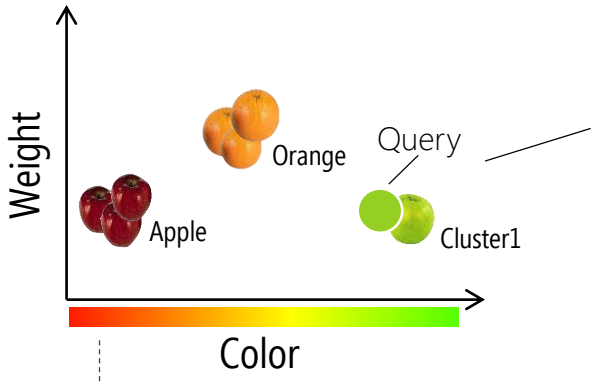
Training Data

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

How to select our **input features**?

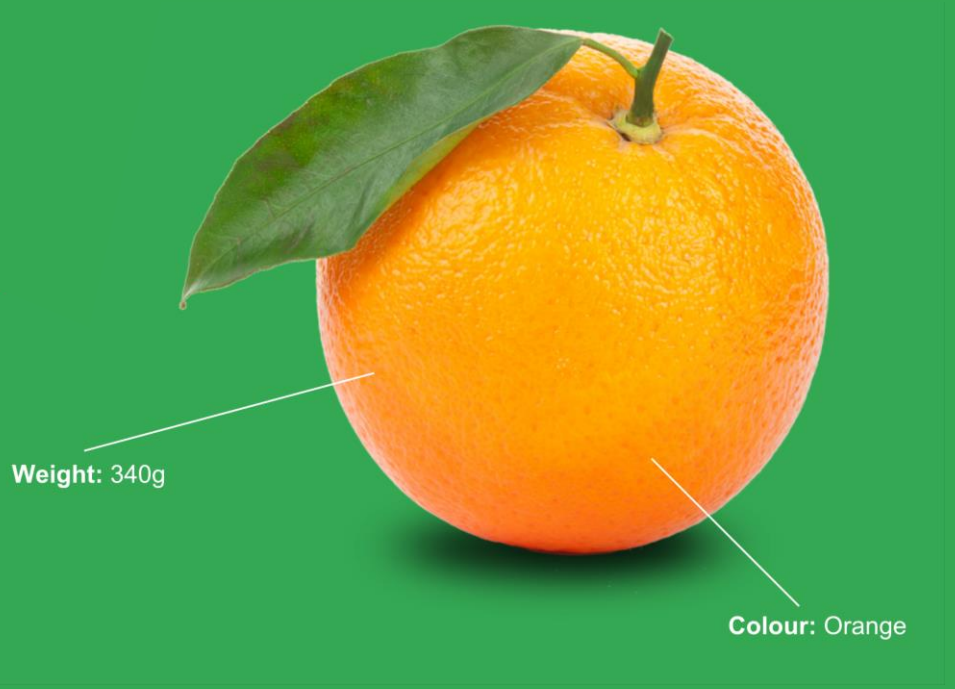
ML TRAINING

FEATURES

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

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

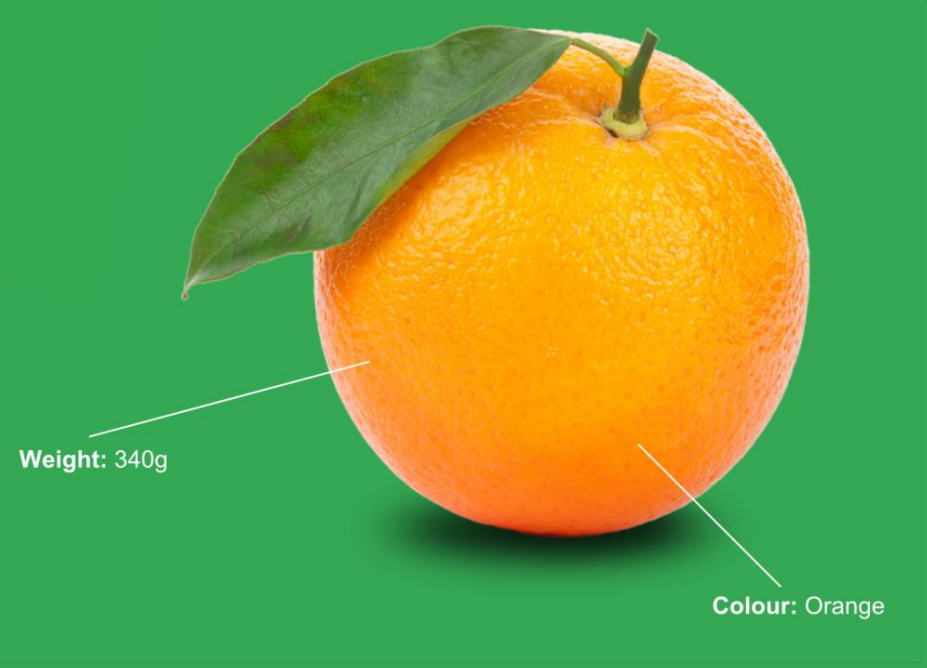
With the fruit example, we tried just **weight** and **color**. **Two features**, which means your featureset is **two-dimensional**



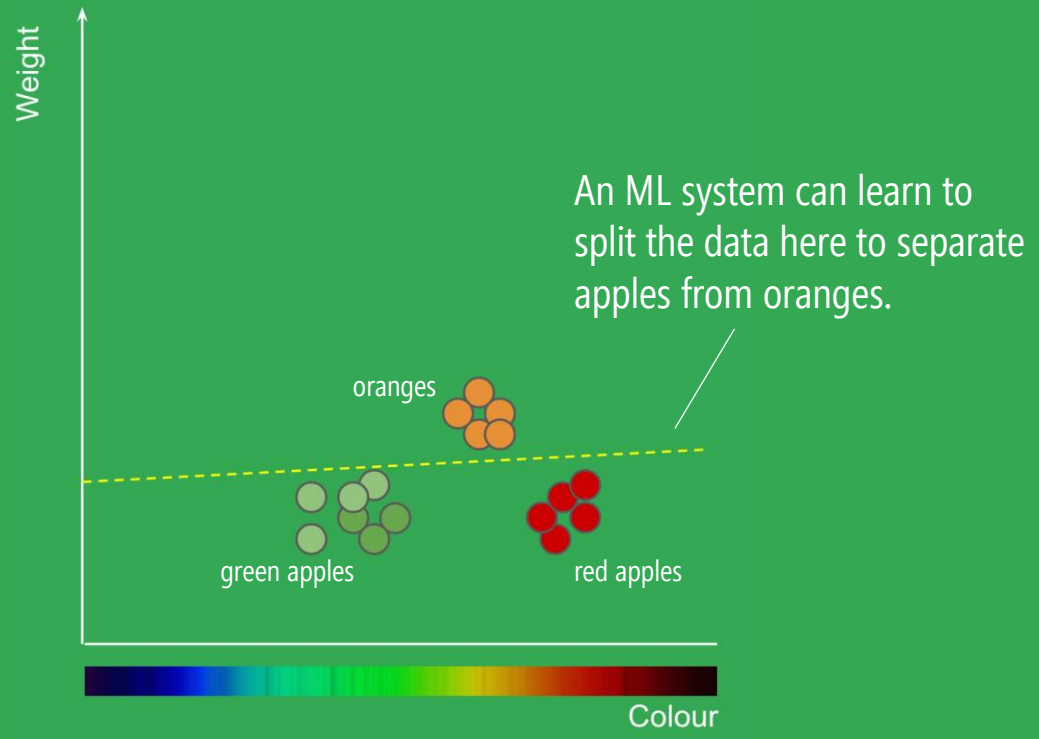
Source: Mayes, Machine Learning 101, (link)

ML TRAINING

SELECTING FEATURES



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

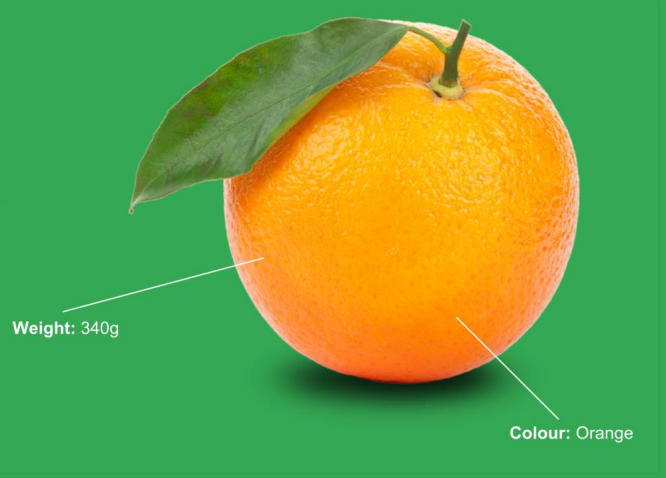


Brainstorm, visualize, investigate, & iterate your feature space. When exploring features, it's a good idea to plot them on a graph (when possible) to develop intuition & assess discriminability.

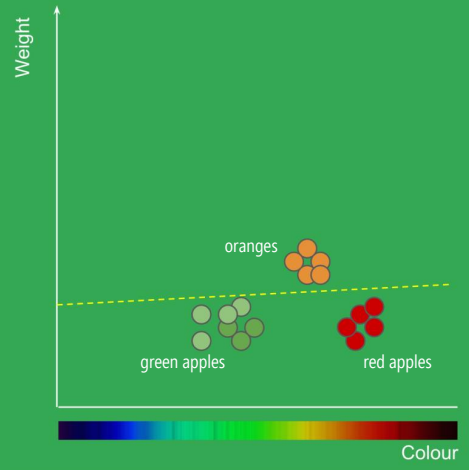
Source: Mayes, Machine Learning 101, (link)

ML TRAINING

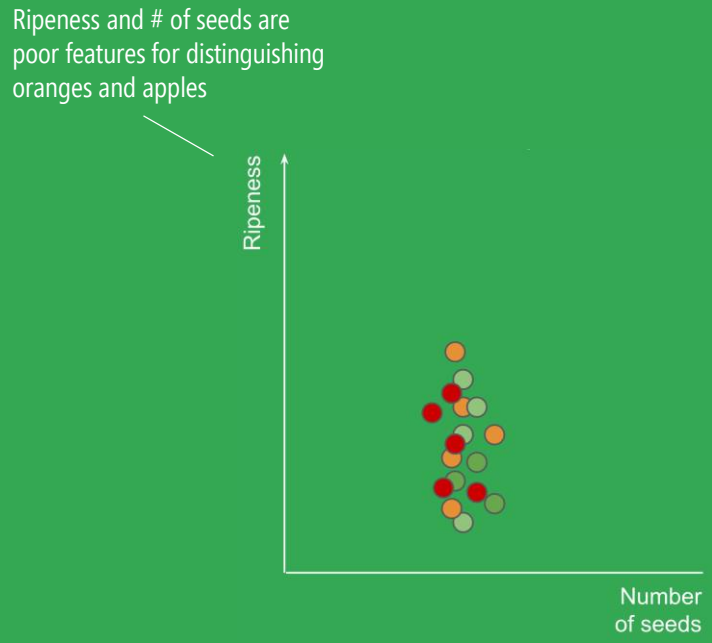
SELECTING FEATURES



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

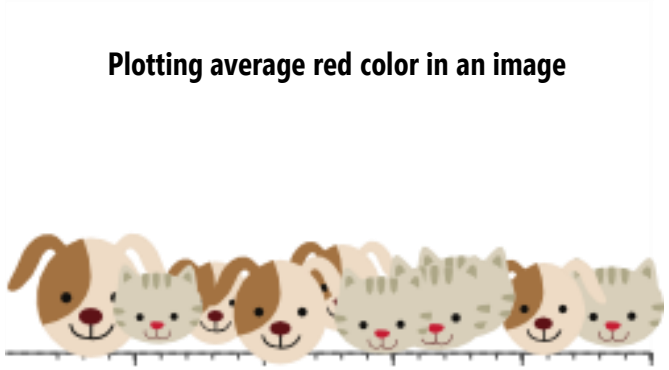
Generally, ML systems are trained with (super!) high dimensional feature vectors that are hard (impossible) to plot and develop an intuition for.

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

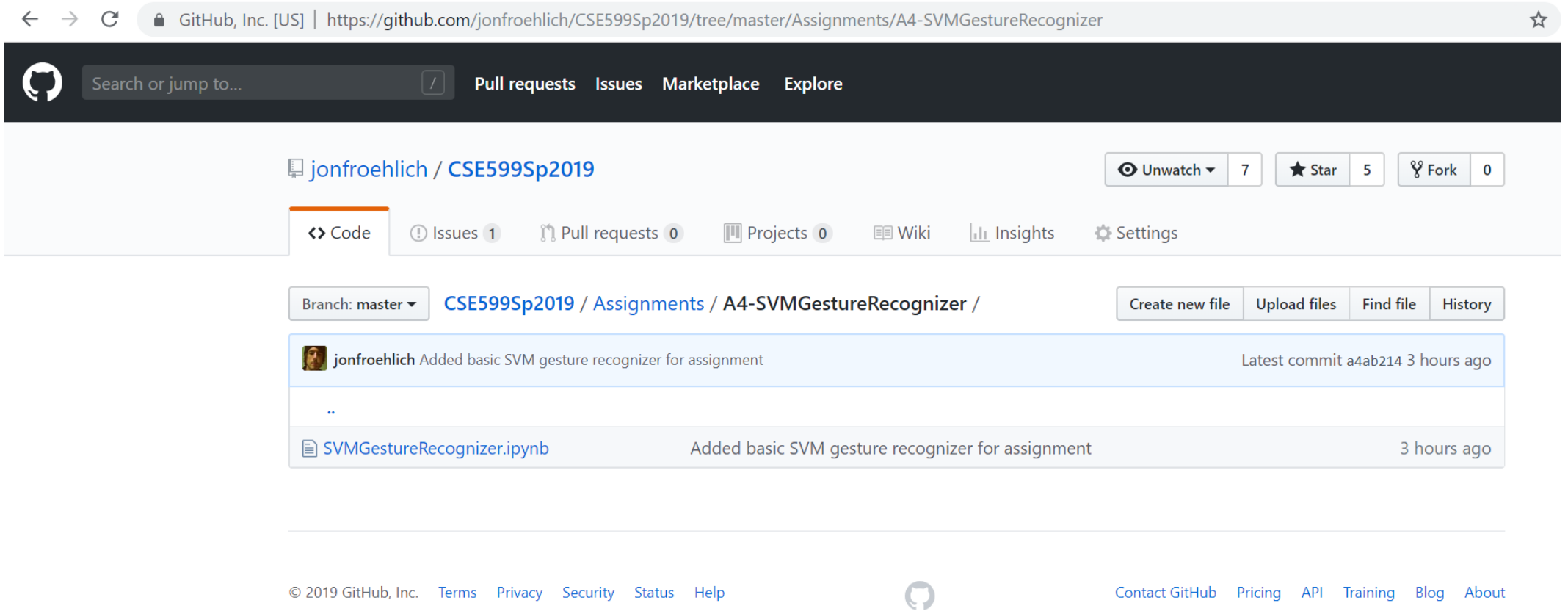
Plotting average red color in image. Notice how a single feature does not result in a perfect separation of our training data.

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.



SVMGestureRecognizer

localhost:8889/notebooks/SVMGestureRecognizer.ipynb

jupyter SVMGestureRecognizer (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3

Run | Markdown

Table of Contents

In [1]:

```
# This cell includes the major classes used in our classification analyses
import matplotlib.pyplot as plt # needed for plotting
import numpy as np # numpy is primary library for numeric array (and matrix) handling
import scipy as sp
from scipy import stats, signal
import random
from sklearn import svm # needed for svm
from sklearn.metrics import confusion_matrix
import itertools

# Each accelerometer log file gets parsed and made into a SensorData object
class SensorData:

    # Constructors in Python look like this (strangely enough)
    # All arguments are numpy arrays except sensorType, which is a str
    def __init__(self, sensorType, currentTimeMs, sensorTimestampMs, x, y, z):
        self.sensorType = sensorType

        # On my mac, I could cast as straight-up int but on Windows, this failed
        # This is because on Windows, a Long is 32 bit but on Unix, a Long is 64bit
        # So, forcing to int64 to be safe. See: https://stackoverflow.com/q/38314118
        self.currentTimeMs = currentTimeMs.astype(np.int64)

        # sensorTimestampMs comes from the Arduino function
        # https://www.arduino.cc/reference/en/language/functions/time/millis/
        # which returns the number of milliseconds passed since the Arduino board began running the current program.
        self.sensorTimestampMs = sensorTimestampMs.astype(np.int64)

        self.x = x.astype(float)
        self.y = y.astype(float)
        self.z = z.astype(float)

        # Calculate the magnitude of the signal
        self.mag = np.sqrt(self.x**2 + self.y**2 + self.z**2)

        self.sampleLengthInSecs = (self.currentTimeMs[-1] - self.currentTimeMs[0]) / 1000.0
        self.samplesPerSecond = len(self.currentTimeMs) / self.sampleLengthInSecs

    # Returns a dict of numpy arrays for each axis of the accel + magnitude
    def get_data(self):
        return {"x":self.x, "y":self.y, "z":self.z, "mag":self.mag}

    # Returns a dict of numpy arrays for each axis of the accel + magnitude
    def get_processed_data(self):
        return {"x_p":self.x_p, "y_p":self.y_p, "z_p":self.z_p, "mag_p":self.mag_p}

# A trial is one gesture recording and includes an accel SensorData object
# In the future, this could be expanded to include other recorded sensors (e.g., a gyro)
# that may be recorded simultaneously
class Trial:

    # We actually parse the sensor log files in the constructor--this is probably bad practice
    # But offers a relatively clean solution
    def __init__(self, gestureName, endTimeMs, trialNum, accelLogFilenameWithPath):
        self.gestureName = gestureName
```

EXERCISE: BRAINSTORM & PLOT FEATURES

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

Length of signal

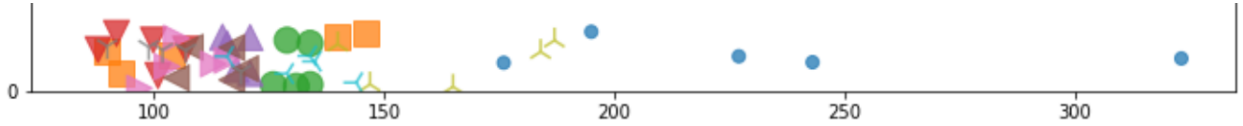
Max accel magnitude

Fundamental frequency of FFT

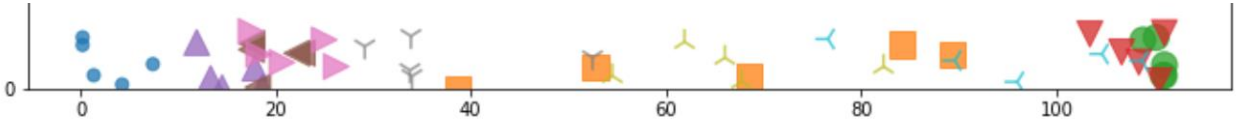
…

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

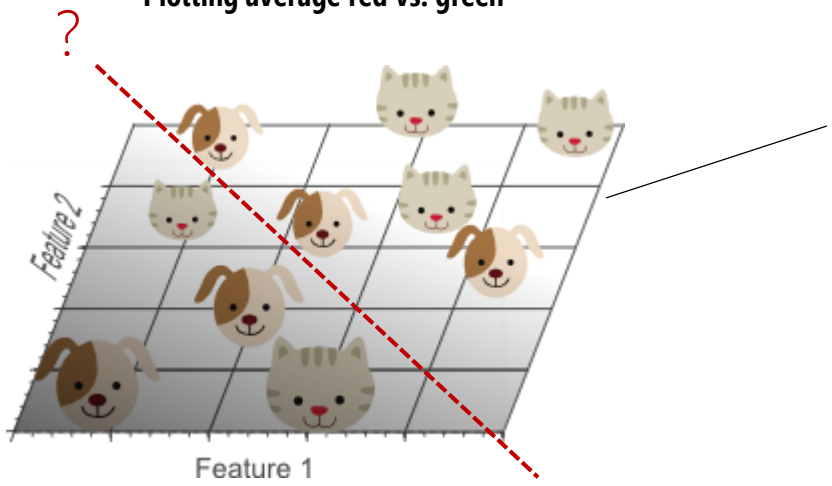


SELECTING FEATURES: HIGHER DIMENSIONALITY

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.

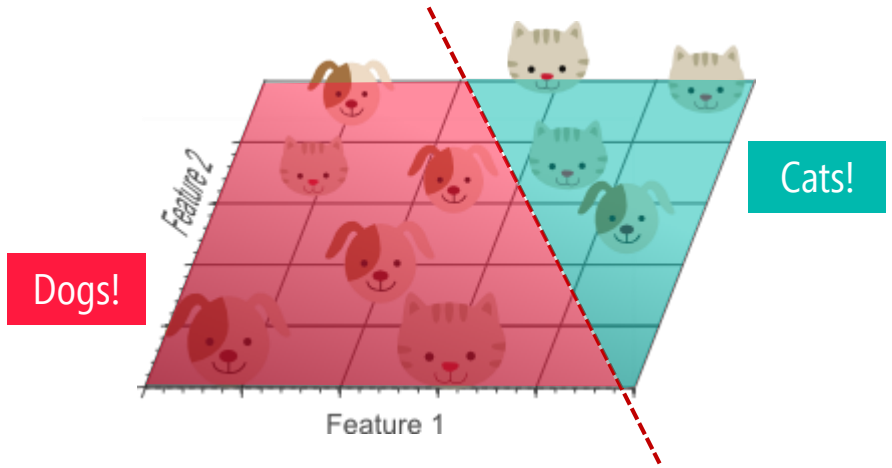
Source: http://www.visiondummy.com/2014/04/curse-dimensionality-affect-classification/

SELECTING FEATURES: HIGHER DIMENSIONALITY

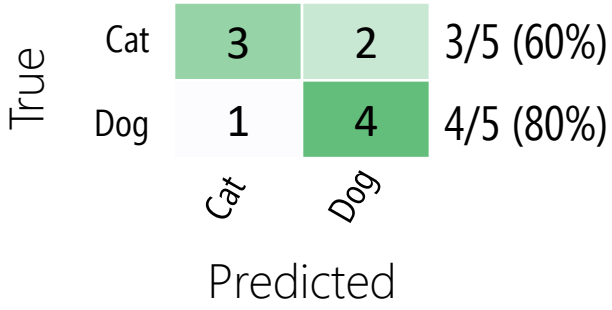
Let's look at another example: this time trying to separate cats and dogs using average pixel color in image.

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.

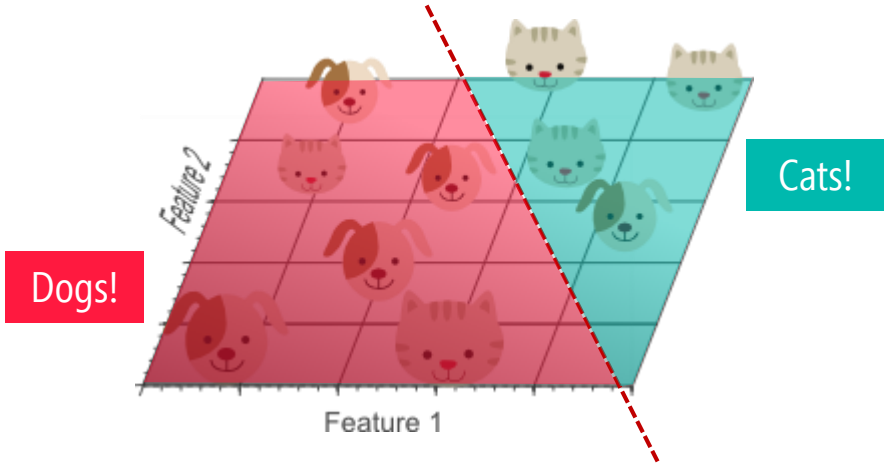
Source: http://www.visiondummy.com/2014/04/curse-dimensionality-affect-classification/

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.

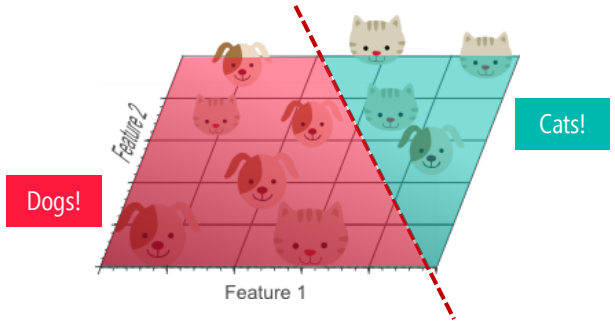
2D Feature Vector

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/

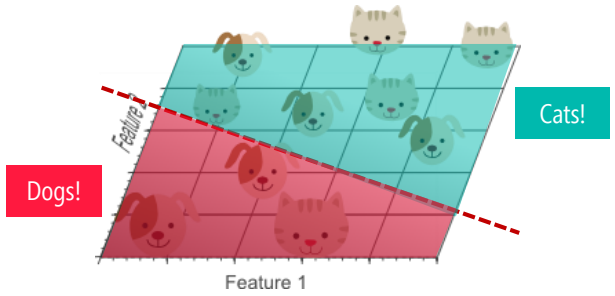
SELECTING FEATURES: HIGHER DIMENSIONALITY

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	



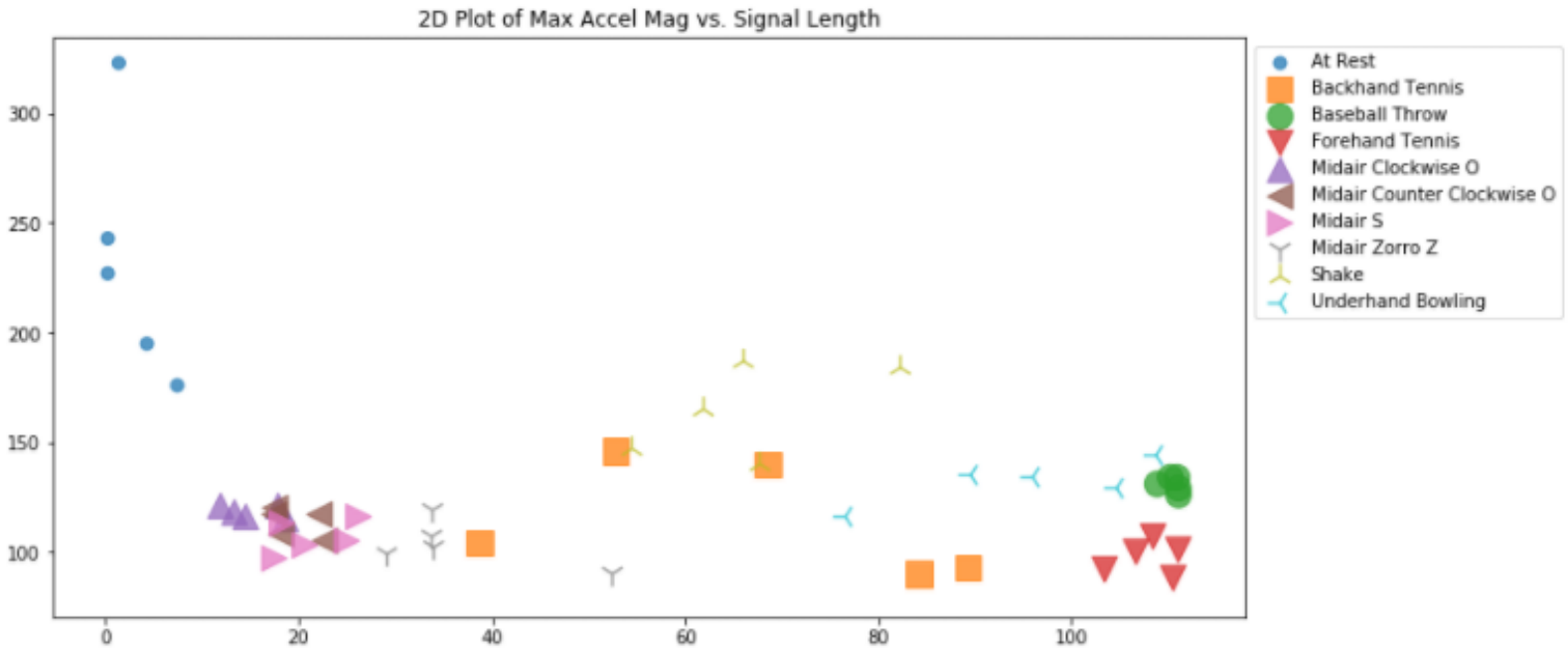
Overall Accuracy: 6/10 (60%)

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

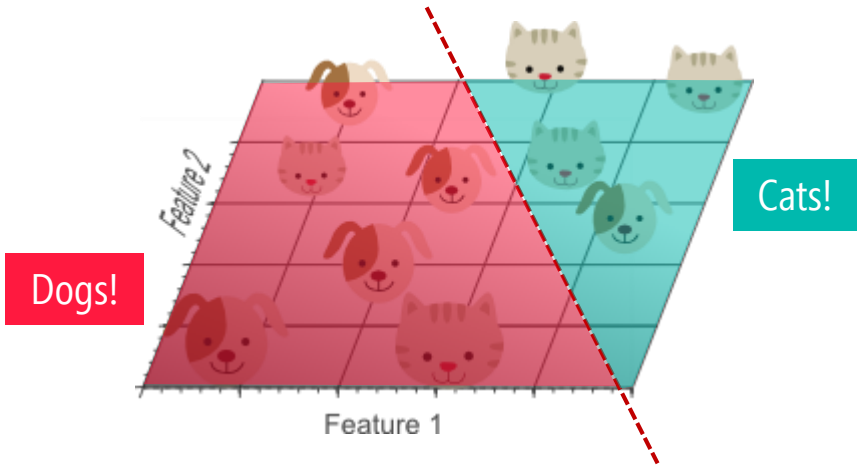
Let's look at another example: this time trying to separate cats and dogs using average pixel color in image.

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.

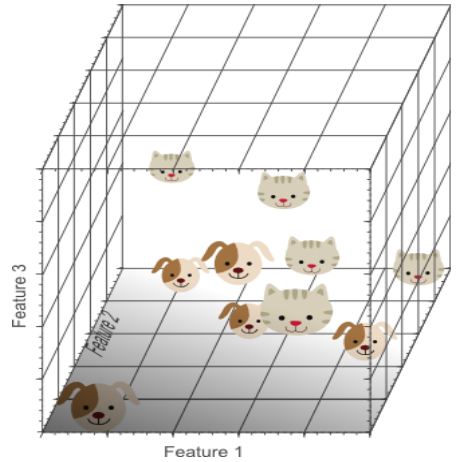
Plotting average red vs. green in image



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/

SELECTING FEATURES: HIGHER DIMENSIONALITY

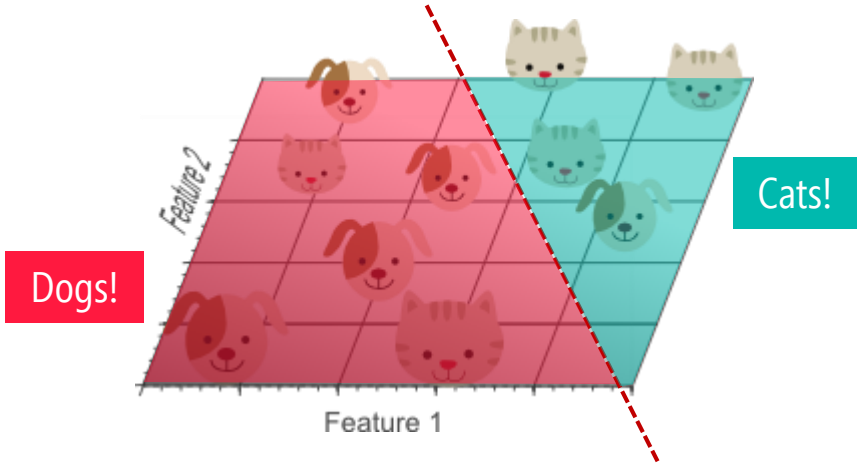
Let's look at another example: this time trying to separate cats and dogs using average pixel color in image.

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

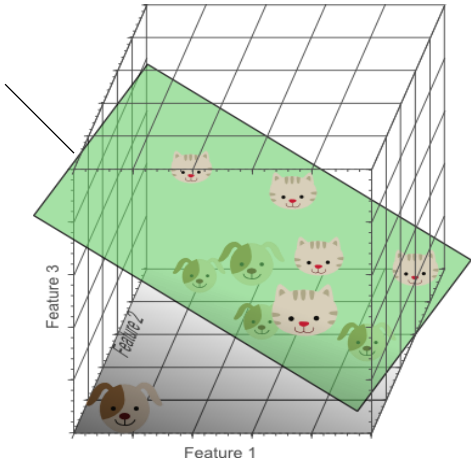


2D Feature Vector

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

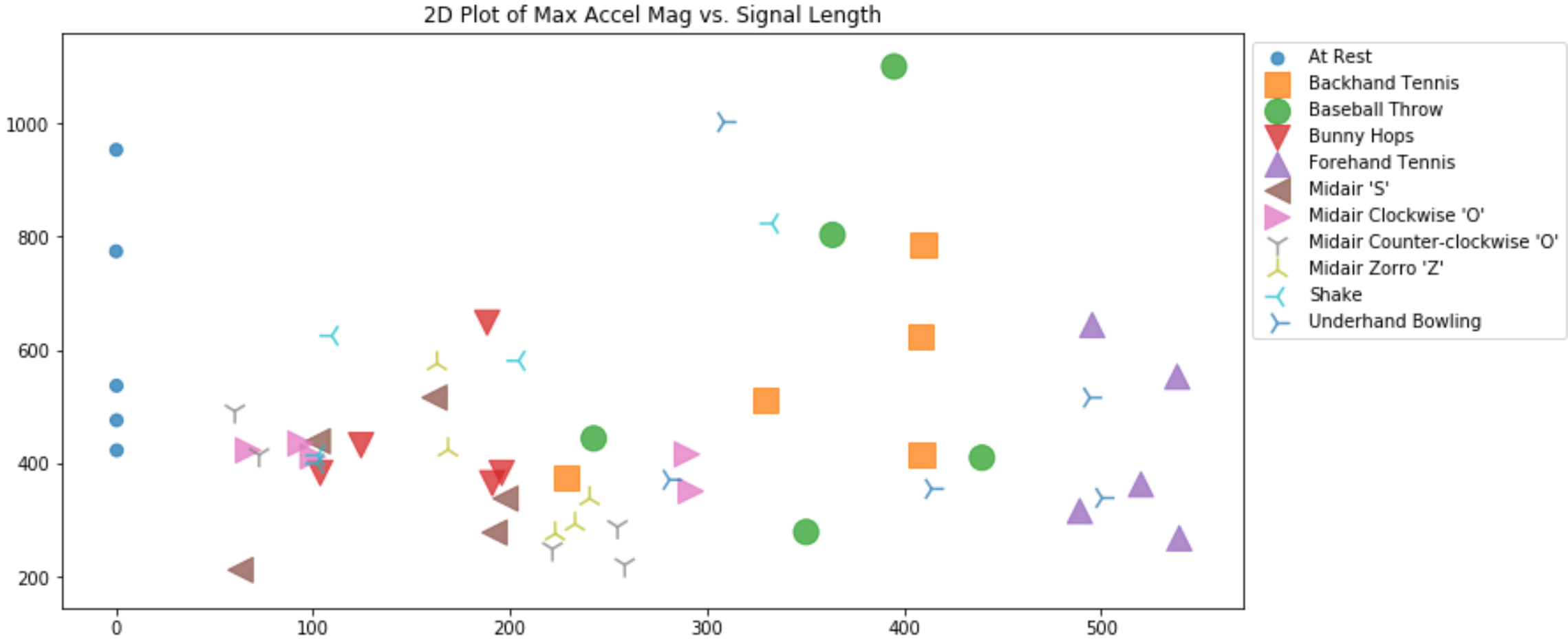
Now, in a three-dimensional feature space, we can find a plane that perfectly splits dogs from cats.

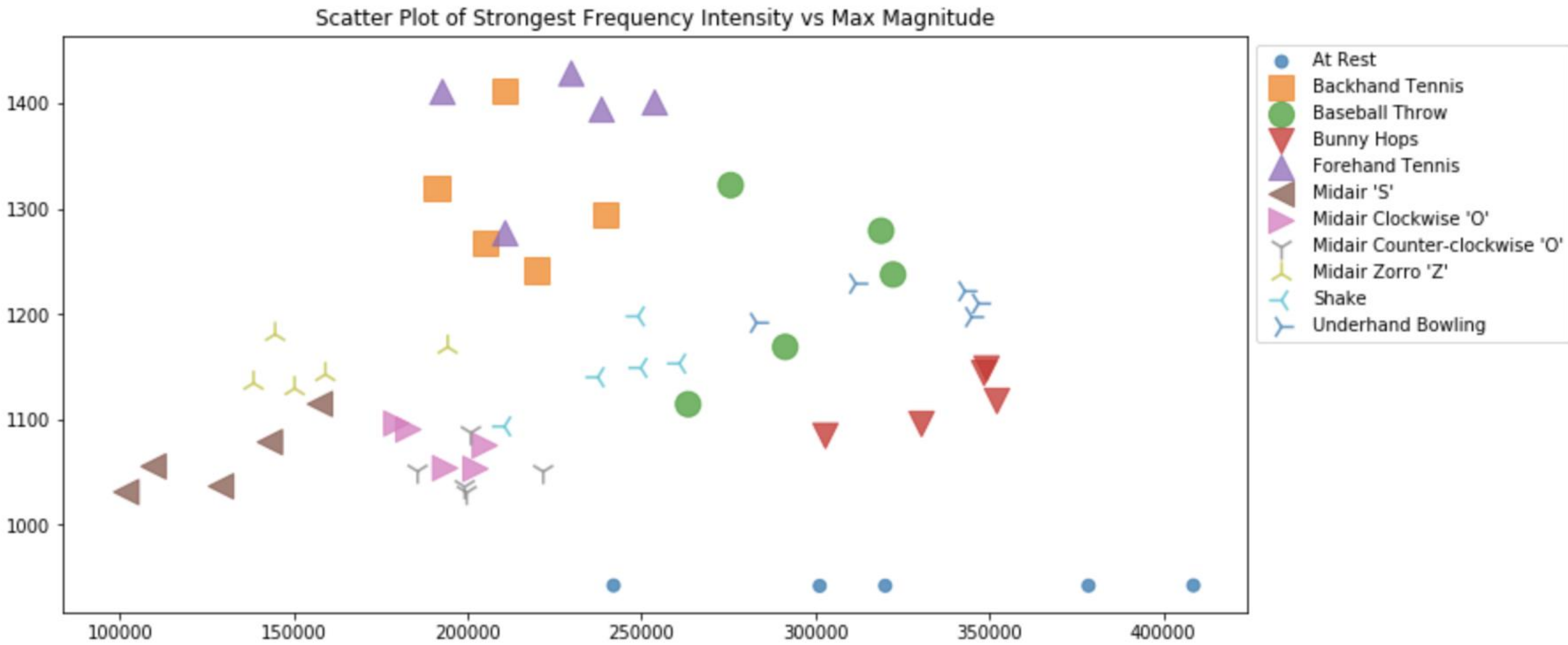


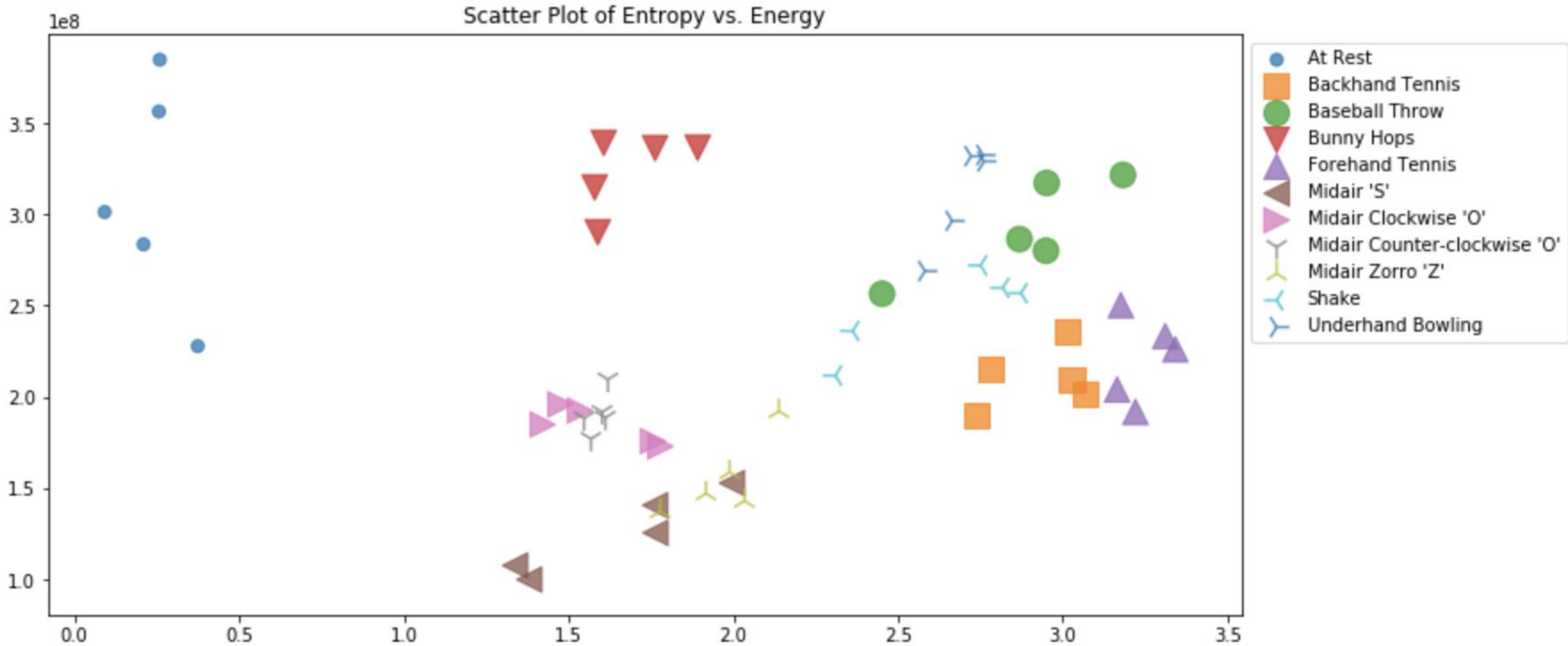
3D Feature Vector

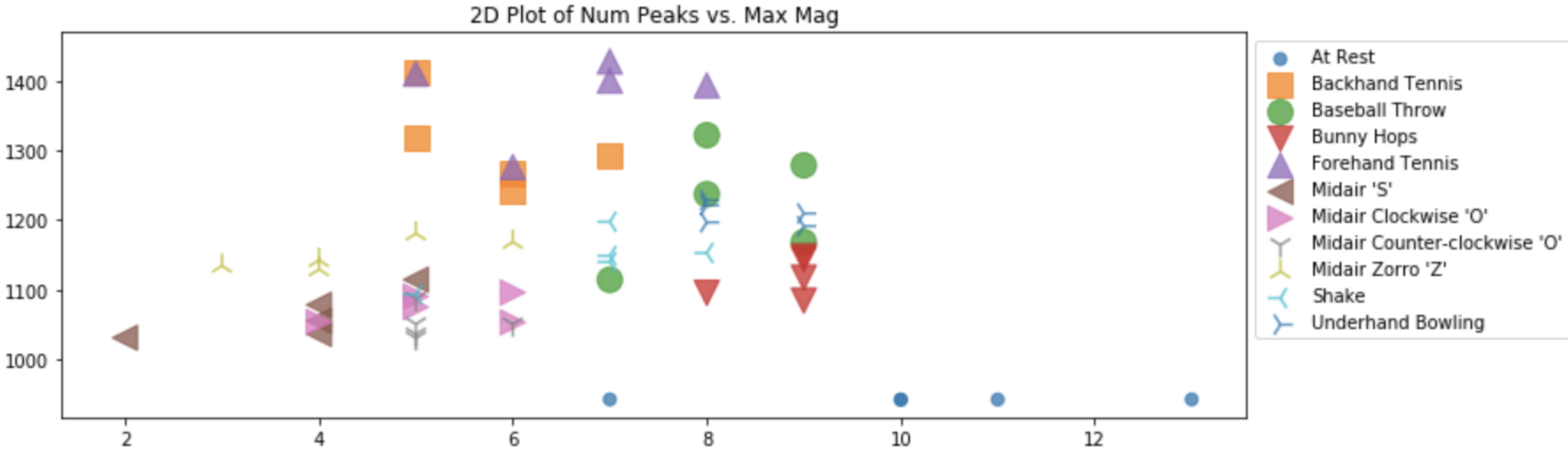
Using a 3D-input feature vector (R,G,B), we begin to see more separation. (Of course, plotting in 3D often requires interactivity to tilt, rotate, explore data)

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


2D Plot of Max Accel Mag vs. Signal Length
1000
800
600
400
200
0
100
200
300
400
500
At Rest
Backhand Tennis
Baseball Throw
Bunny Hops
Forehand Tennis
Midair 'S'
Midair Clockwise 'O'
Midair Counter-clockwise 'O'
Midair Zorro 'Z'
Shake
Underhand Bowling


Scatter Plot of Strongest Frequency Intensity vs Max Magnitude
1400
1300
1200
1100
1000
100000
150000
200000
250000
300000
350000
400000
At Rest
Backhand Tennis
Baseball Throw
Bunny Hops
Forehand Tennis
Midair 'S'
Midair Clockwise 'O'
Midair Counter-clockwise 'O'
Midair Zorro 'Z'
Shake
Underhand Bowling


Scatter Plot of Entropy vs. Energy
1e8
3.5
3.0
2.5
2.0
1.5
1.0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
At Rest
Backhand Tennis
Baseball Throw
Bunny Hops
Forehand Tennis
Midair 'S'
Midair Clockwise 'O'
Midair Counter-clockwise 'O'
Midair Zorro 'Z'
Shake
Underhand Bowling


2D Plot of Num Peaks vs. Max Mag
1400
1300
1200
1100
1000
2
4
6
8
10
12
At Rest
Backhand Tennis
Baseball Throw
Bunny Hops
Forehand Tennis
Midair 'S'
Midair Clockwise 'O'
Midair Counter-clockwise 'O'
Midair Zorro 'Z'
Shake
Underhand Bowling

A3 REFLECTION + INTRO TO MODEL-BASED

CSE 599 Prototyping Interactive Systems | Lecture 14 | May 1

Jon Froehlich • Jasper O'Leary (TA)

DESIGN USE BUILD

MAKEABILITY LAB

PAUL G. ALLEN SCHOOL
OF COMPUTER SCIENCE & ENGINEERING

UNIVERSITY of WASHINGTON