Berkeley Emergent Space Tensegrities

DOCUMENTATION OF PROGRESS

7.23.20

Plan

- Different types of classifiers
- Fully implement leave-one-out cross-validation with probabilities
- Tried Gaussian Naive Bayes classifier. I don't think it is designed to output probabilities other than 0 and 1 so it won't be useful for probability classification

```
[X_train, X_test, y_train, y_test] = train_test_split(data, y, random_state=0)


#Naive Bayes Implementation
nb = GaussianNB().fit(X_train, y_train)
clf_probs1 = nb.predict_proba(X_test)
score = log_loss(y_test, clf_probs1)
nb.fit(X_train, y_train)
clf_probs2 = nb.predict_proba(X_test)

print(score)
print(clf_probs2)
print(nb.score(X_train, y_train))
print(nb.score(X_test, y_test))
print("Predicted labels = " + str(nb.predict(X_test)))
print("Correct labels =   " + str(y_test))
```

```
3.837641821656746
[[1.00000000e+000 8.49963067e-117 0.00000000e+000 0.00000000e+000]
 [0.00000000e+000 5.46124439e-032 1.00000000e+000 0.00000000e+000]
 [0.00000000e+000 1.00000000e+000 0.00000000e+000 0.00000000e+000]
 [0.00000000e+000 2.33171817e-042 1.00000000e+000 0.00000000e+000]
 [0.00000000e+000 3.16657836e-033 1.00000000e+000 0.00000000e+000]
 [0.00000000e+000 1.00000000e+000 0.00000000e+000 0.00000000e+000]
 [2.78105813e-141 1.00000000e+000 4.77461248e-080 0.00000000e+000]
 [5.12392555e-029 1.02346569e-230 0.00000000e+000 1.00000000e+000]
 [0.00000000e+000 1.00000000e+000 0.00000000e+000 0.00000000e+000]
 [3.20850551e-038 2.92838041e-247 0.00000000e+000 1.00000000e+000]
 [2.00761325e-098 1.67276930e-181 0.00000000e+000 1.00000000e+000]
 [0.00000000e+000 4.35690608e-040 1.00000000e+000 0.00000000e+000]
 [0.00000000e+000 3.87055073e-034 1.00000000e+000 0.00000000e+000]
 [0.00000000e+000 1.00000000e+000 0.00000000e+000 0.00000000e+000]
 [0.00000000e+000 1.00000000e+000 0.00000000e+000 0.00000000e+000]
 [0.00000000e+000 1.00000000e+000 6.86953072e-019 0.00000000e+000]
 [1.00000000e+000 2.37886698e-131 0.00000000e+000 0.00000000e+000]
 [0.00000000e+000 3.74956232e-030 1.00000000e+000 0.00000000e+000]]
0.9814814814814815
0.8888888888888888
Predicted labels = [0 2 1 2 2 1 1 3 1 3 3 2 2 1 1 1 0 2]
Correct labels =   [0, 2, 1, 2, 2, 1, 1, 3, 1, 3, 3, 2, 2, 1, 1, 2, 3, 2]
```

- Tried to implement leave-one-out with my current cross-validation function but it seems like arrays with one data point are not compatible so I'm going to try "leave-two-out"

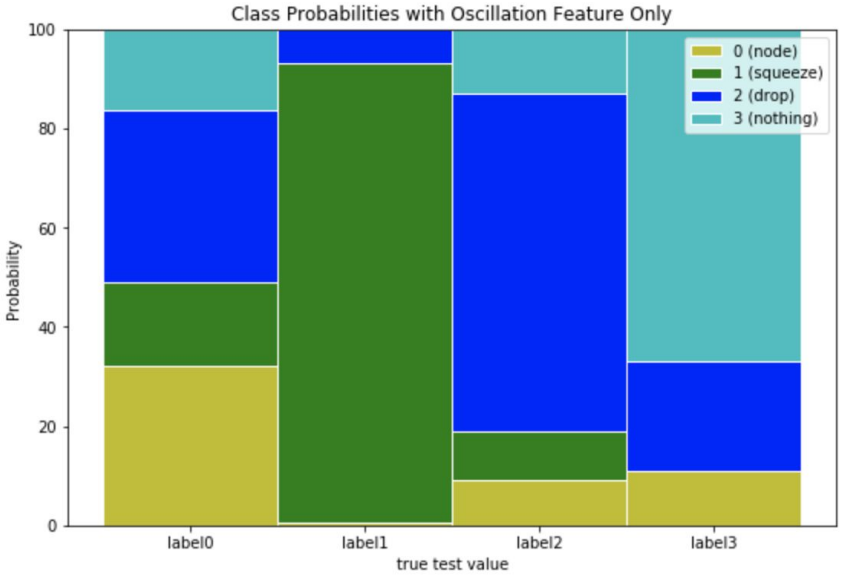
Berkeley Emergent Space Tensegrities

DOCUMENTATION OF PROGRESS

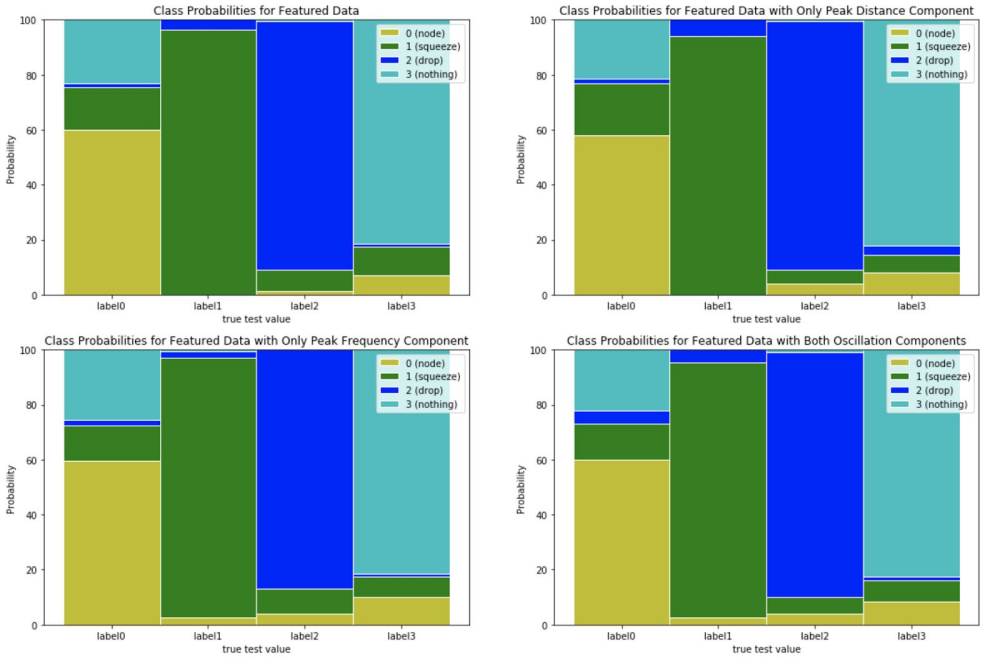
7.21.20

Progress

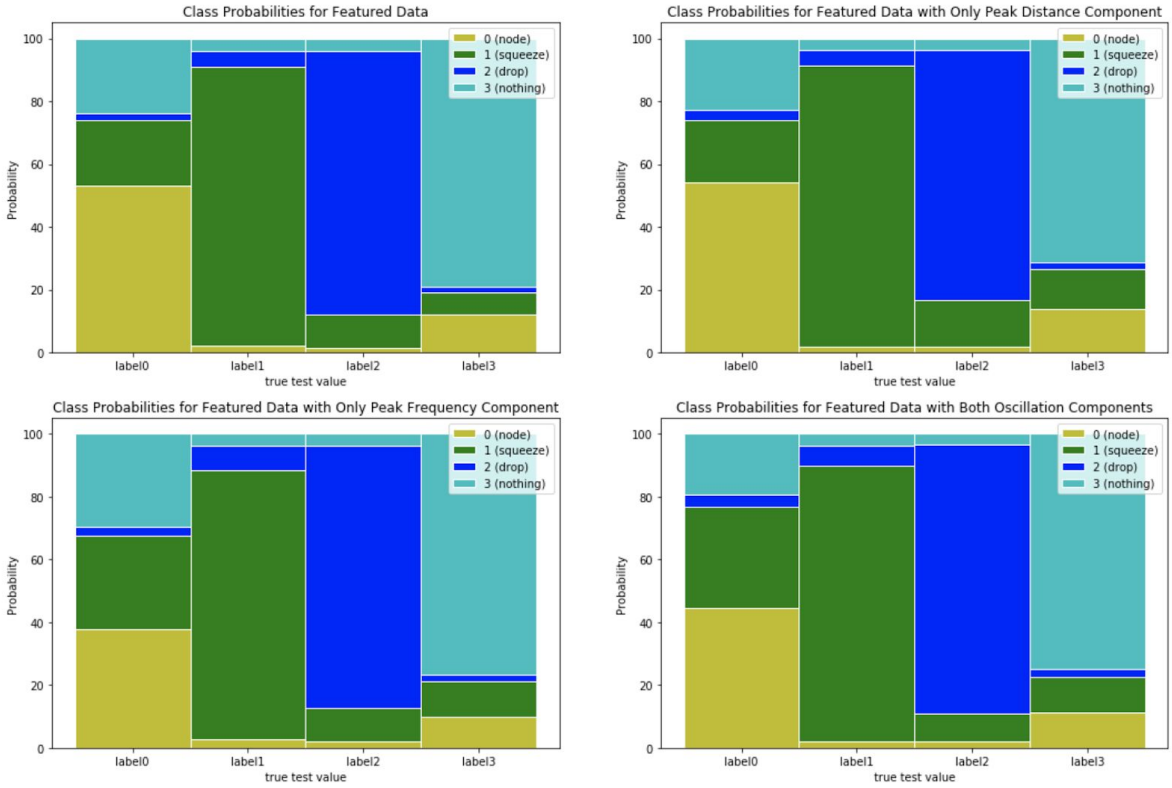
- Ideas for oscillation frequency:
    - Condense data by different amounts (test what factor is most effective)
    - Also find max distance between peaks, which would return the lowest frequency that occurs in the test. Hypothesis is that null tests will have the highest frequency because they're just random noise
    - Take a look at Albert and Akhil's scripts to see what they did, if anything, for oscillation frequency
- Made oscillation frequency feature, added to one_feature function, ran quickly through model and it does sort of well, good with labels 1,2,3 but badly with label0 (thinks nodes are drops which is fair I guess)?
    - This will probably be okay by Andrew though because he said not to focus too much on the node tests since they're not actually a real test
- Will probably perform okay with other features anyway
- Oscillation feature returns two things: the max distance between peaks in the data, as well as the number of local minima and maxima in the particular column



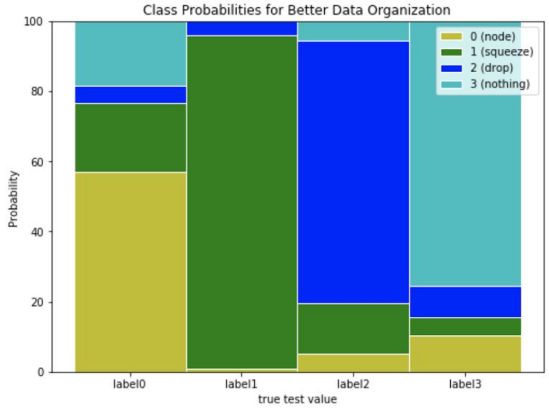
- Also I had the thought that I should maybe try organizing the featured data that I feed into the model in a way that makes more sense
- Honestly I should probably just get rid of node tests altogether... we're going to be getting more data soon and they've just been a pain. Good for practicing machine learning though and thinking about featurization
- Comparison of performance of full feature array with vs. without oscillation frequency feature. With no cross-validation:

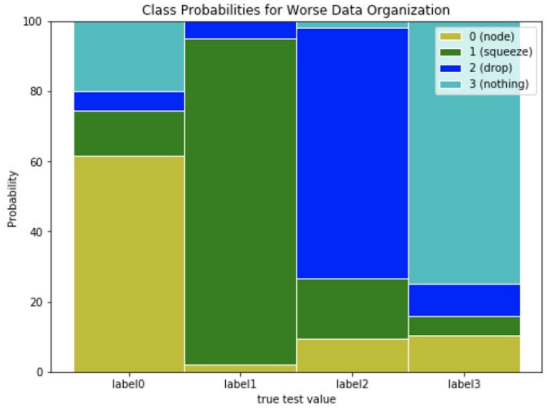


- With cross-validation:



- Helps a little? Without all the other features it was clear that the oscillation feature makes a difference but it's not very discernible with the other features
- Should probably work more on this feature, or take ideas from Akhil and Albert's scripts. For now I'll keep it
- Again, I should maybe try to organize the features better (the two_d function). Each data point has 5-7 features per column, but when I make the whole feature array 2-dimensional each data point with its features becomes a 1-D array, with all of the features lumped out of order
- Changed the two_d function but it didn't really make much of a difference, and the worse one might even be better:





- Again, here's the second two_d function that I wrote just in case:

```
def two_d(data):
    len_data, num_cols, num_points = np.shape(data)
    data_new = np.zeros((len_data, num_cols*num_points))
    for i in range(len_data):
        current = data[i]
        for k in range(num_points):
            for j in range(num_cols):
                data_new[i][num_cols*k+j] = current[j][k]
    return data_new
```

- Wrote a few different jerk functions that contribute nothing so far, so I'm going to work on them more to see if I can actually get them to perform better. Impulse function to come, although I suspect it might have similar results to my sum function, just because it's related to the sum of force

Berkeley Emergent Space Tensegrities

TENSEGRITY HRI MEETING

7.11.20 / 1:00-2:00PM / ZOOM https://berkeley.zoom.us/j/2417093709

My Update

- Wrote summary of data pipeline for my machine learning classifier
- I think it's pretty comprehensive but I still want to work and adapt different things so I think I'll continue to modify it as a I make changes to my script
- Node tests are the ones that are getting the worst results and I started working on a feature that is supposed to help differentiate them but it's not really changing the results yet so I have to modify it
- I did an individual feature comparison between all the ones I have made so far and found a feature that I made earlier literally does nothing so I removed that. It's also interesting to see the different results each feature gives and that node tests tend to do poorly with all of them but the other classes do relatively well
- I also compared a single feature with itself by varying it slightly and got interesting results, but I'm also a little confused because the classifier outputs slightly different probabilities when I run it multiple times so I'm not sure what's going on there but I guess it's good for informal validation
- I also started leave-one-out cross validation and so far it seems like the results are really good (all the labels were correct) but it outputs the label instead of the probabilities so I want to change it so that it outputs probabilities and then it will be really helpful and interesting to see

Meeting Notes

- How to streamline data truncation?
- Prevent having to open payload to test if test was valid → live plotting to make sure data is being read
- Truncate while watching video instead of looking at plots to eliminate bias?

- Will probably not take videos though because it’s too much work
- Use iterator in data to easily see where new test begins (all in same file)
- Treat tunable features as hyperparameters to compare results for different modifications on the classifier
- Run Gaussian Naive Bayes classifier with probabilities to verify Albert’s results
- New data coming in this week hopefully, start truncating hundreds of test to include in train set

Berkeley Emergent Space Tensegrities

DOCUMENTATION OF PROGRESS

7.9.20

Progress

- Summary of data pipeline for my machine learning classifier:
    - Truncate data using Matthew and Andrew's MATLAB script datatrunc.m into 7 second intervals that encompass a test
        - These don't technically have to all be 7 seconds since we are taking the features but it is easier to deal with all the data together when they are of equal length
    - Tests are node tests, squeeze tests, drop tests, and null tests
    - Only take the FSR data (haven't been using the accelerometer data in the classifier but I want to look into that)
    - Feature array:
        - Mean
        - Var
        - Max - min
        - Three quarters
        - Sum over max
    - Condense data by a factor of 10 (10 data points become one) which makes featurization a lot faster
    - Normalize data: data/np.max(data)
    - Featurize data by running it through "features" function
    - Data now has the size (72, 5, 12). Run it through "two_d" function in order to put each of the five features for each of the 12 FSRs in one column, so each of the 72 tests has 60 features. Might be helpful to put all the same kinds of features in order but can deal with that later
    - Classifier:

- Split data into train, validation, and testing sets. Train typically has 32 tests, validation has the next 32, and testing has the last 8. Ideally we will have way more data for all three of these categories
- Use Random Forest Classifier
- No validation: you can just run the classifier on x_train_valid and y_train_valid, which is all 64 of the train and validation set data points
- Will end up with an array of probabilities for each of the 8 test data points. Because there are two of each class, average each of the class probabilities so we end up with 4 arrays. Plot these 4 arrays as a stacked bar plot to visualize the accuracy of the classifier

Berkeley Emergent Space Tensegrities

DOCUMENTATION OF PROGRESS

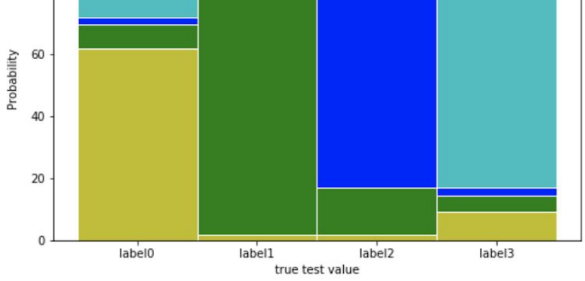
7.6.20

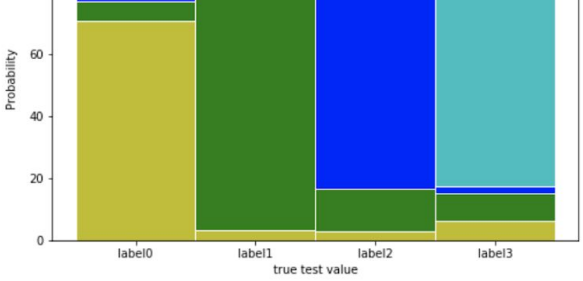
Progress

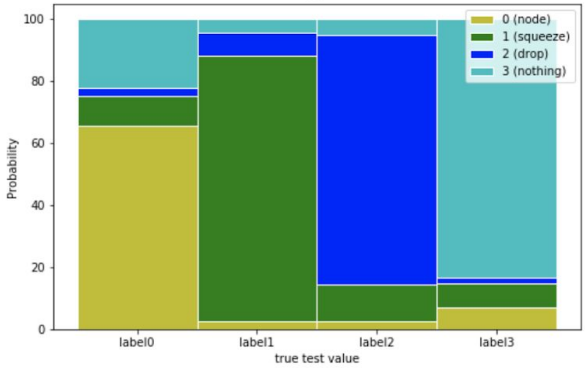
- Plan for today:
    - Feature for node vs. nothing tests
    - Put together summary of data processing pipeline
    - More cross-validation and ROC stuff
        - Different types of cross-validation (look into k-fold and leave-one-out)
        - Individual feature comparison
- The three-quarters feature calculates the maximum reading from each of the 12 FSRs, and returns the percentage of data points that are under 75% of the maximum for each of the 12. It was originally intended to differentiate between drop tests and other tests, given that the majority of the data for a drop test is below 75% of the max. However, I compared the “three quarter” feature for values of 30, 40, 50, 60, 70, 80, 90, and 99% of the maximum and compared the relative probabilities. It looks like 40% yields the highest total accuracy for label0 and about 80% yields highest for label3, so the final feature array should be adapted accordingly (I may potentially have two features with different three-quarter percentages). Note that these probabilities are the ones generated with 2-fold cross-validation, and the max of these features was not taken
- Deleting this from my code to make it more succinct, plus findings weren’t very significant but here’s a screenshot of the block just in case I want to try something similar

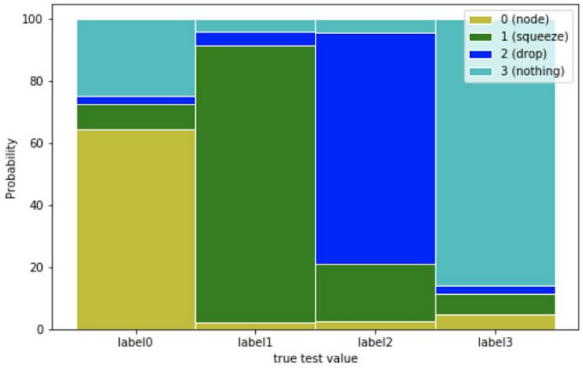
```
# Three-Quarters Comparison —— compare different three-quarter thresholds to determine if some are more helpful

# Condense data to speed up
condensed_data = condense(normalized_data, 10)

# Compare three-quarters values
thirty_data = one_feature(condensed_data, "three quarters", .30)
forty_data = one_feature(condensed_data, "three quarters", .40)
fifty_data = one_feature(condensed_data, "three quarters", .50)
sixty_data = one_feature(condensed_data, "three quarters", .60)
seventy_data = one_feature(condensed_data, "three quarters", .70)
eighty_data = one_feature(condensed_data, "three quarters", .80)
ninety_data = one_feature(condensed_data, "three quarters", .90)
ninety_nine_data = one_feature(condensed_data, "three quarters", .99)
```

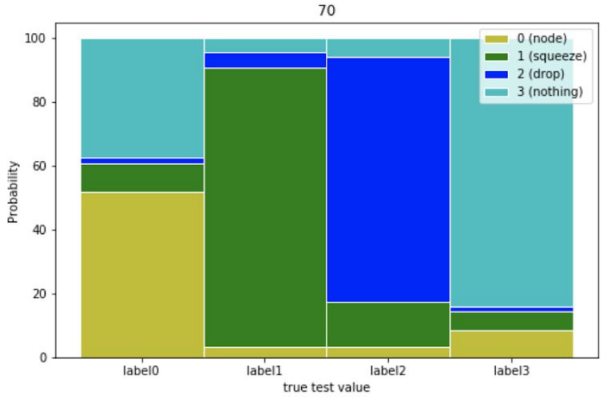
```
# Three-Quarters Comparison

clf_probs1,sig_clf_probs1,_,_,_ = train_xvalidation(thirty_data, 32, 100)
clf_probs2,sig_clf_probs2,_,_,_ = train_xvalidation(forty_data, 32, 100)
clf_probs3,sig_clf_probs3,_,_,_ = train_xvalidation(fifty_data, 32, 100)
clf_probs4,sig_clf_probs4,_,_,_ = train_xvalidation(sixty_data, 32, 100)
clf_probs5,sig_clf_probs5,_,_,_ = train_xvalidation(seventy_data, 32, 100)
clf_probs6,sig_clf_probs6,_,_,_ = train_xvalidation(eighty_data, 32, 100)
clf_probs7,sig_clf_probs7,_,_,_ = train_xvalidation(ninety_data, 32, 100)
clf_probs8,sig_clf_probs8,_,_,_ = train_xvalidation(ninety_nine_data, 32, 100)

fig= plt.figure(figsize=(18,14))
plt.title('Three-Quarters Comparison')
plt.subplot(3,3,1)
generate_plot(sig_clf_probs1, "30")
plt.subplot(3,3,2)
generate_plot(sig_clf_probs2, "40")
plt.subplot(3,3,3)
generate_plot(sig_clf_probs3, "50")
plt.subplot(3,3,4)
generate_plot(sig_clf_probs4, "60")
plt.subplot(3,3,5)
generate_plot(sig_clf_probs5, "70")
plt.subplot(3,3,6)
generate_plot(sig_clf_probs6, "80")
plt.subplot(3,3,7)
generate_plot(sig_clf_probs7, "90")
plt.subplot(3,3,8)
generate_plot(sig_clf_probs8, "99")
```

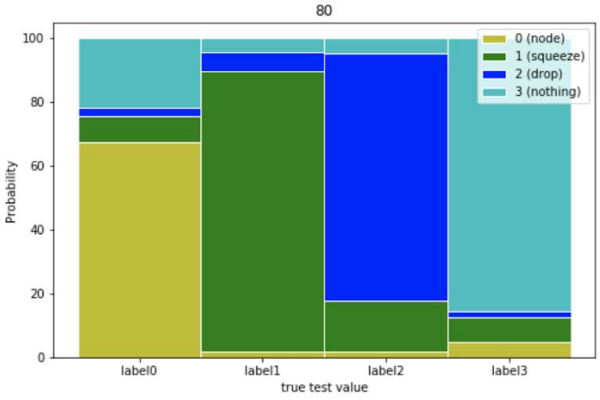

30
0 (node)
1 (squeeze)
2 (drop)
3 (nothing)
Probability
0
20
40
60
80
100
label0
label1
label2
label3
true test value

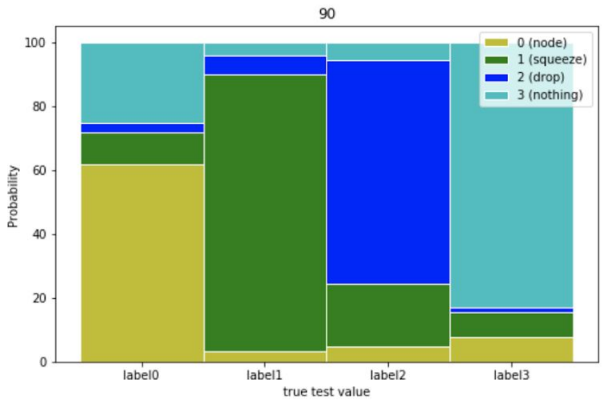

40
0 (node)
1 (squeeze)
2 (drop)
3 (nothing)
Probability
0
20
40
60
80
100
label0
label1
label2
label3
true test value

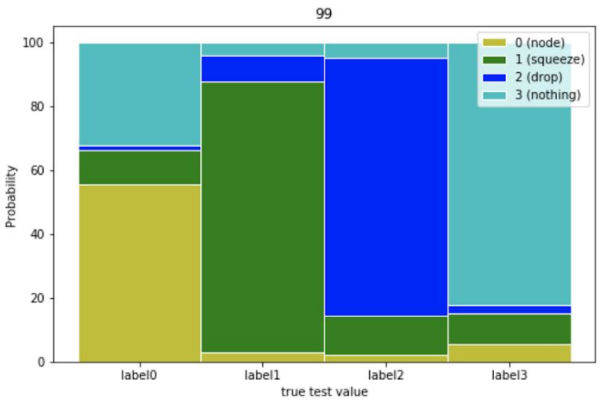

50
0 (node)
1 (squeeze)
2 (drop)
3 (nothing)
Probability
0
20
40
60
80
100
label0
label1
label2
label3
true test value


60
0 (node)
1 (squeeze)
2 (drop)
3 (nothing)
Probability
0
20
40
60
80
100
label0
label1
label2
label3
true test value

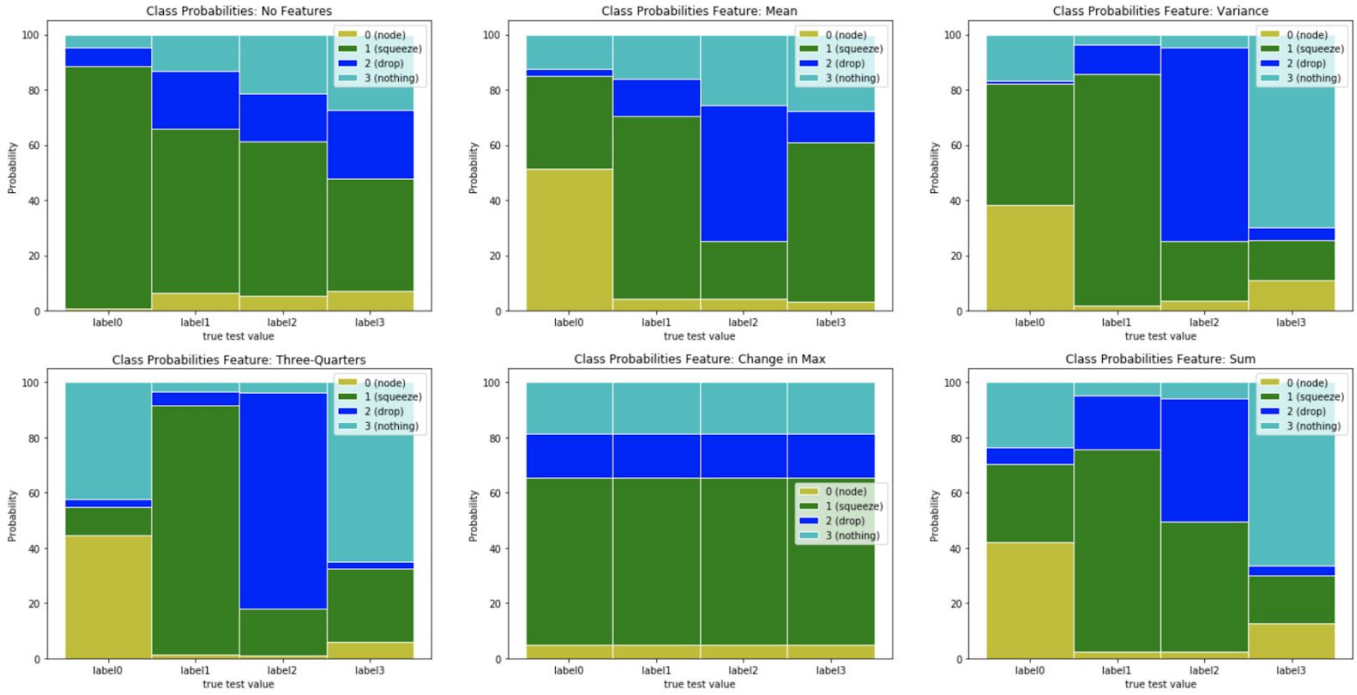

70
0 (node)
1 (squeeze)
2 (drop)
3 (nothing)
Probability
0
20
40
60
80
100
label0
label1
label2
label3
true test value


80
0 (node)
1 (squeeze)
2 (drop)
3 (nothing)
Probability
0
20
40
60
80
100
label0
label1
label2
label3
true test value


90
0 (node)
1 (squeeze)
2 (drop)
3 (nothing)
Probability
0
20
40
60
80
100
label0
label1
label2
label3
true test value

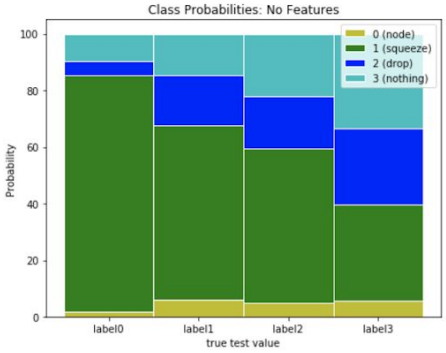

99
0 (node)
1 (squeeze)
2 (drop)
3 (nothing)
Probability
0
20
40
60
80
100
label0
label1
label2
label3
true test value

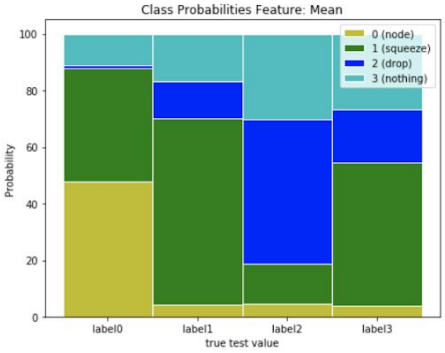
- The change-in-max feature takes the max of each column, takes the data point a certain factor after/before it, and normalizes the result. I intended for it to highlight the features of drop data, because drops should have a larger difference when comparing data points that are close to each other
- From the feature comparison below, it is clear that the change in max feature literally does nothing so I will be removing it from the feature array, but I'll save an image of the code here just in case I think of a way to improve it

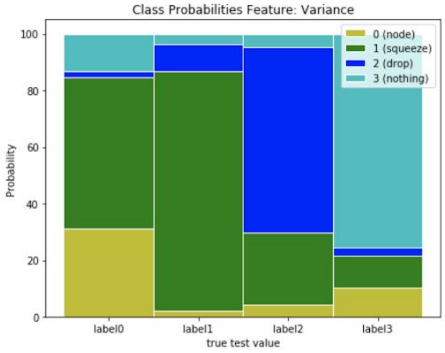


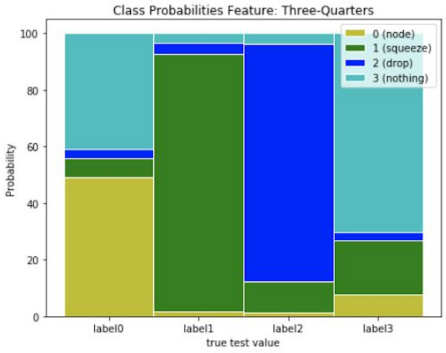

```
# This feature takes the max of each column, takes the data point [change_factor] ms after that, and then normalizes
# the difference

def change_in_max(column, change_factor):
    max_index = np.argmax(column)
    minus_change_index = max_index - change_factor
    if minus_change_index > 0:
        minus_change_value = column[minus_change_index]
    else:
        minus_change_index = max_index + change_factor
        minus_change_value = column[minus_change_index]
    diff = np.max(column) - minus_change_value
    normalized = diff/np.std(column)
    return normalized
```

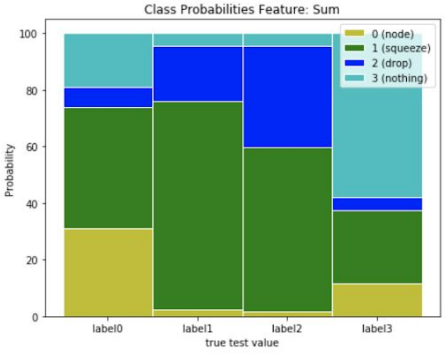
- New feature comparison plot without change in max:



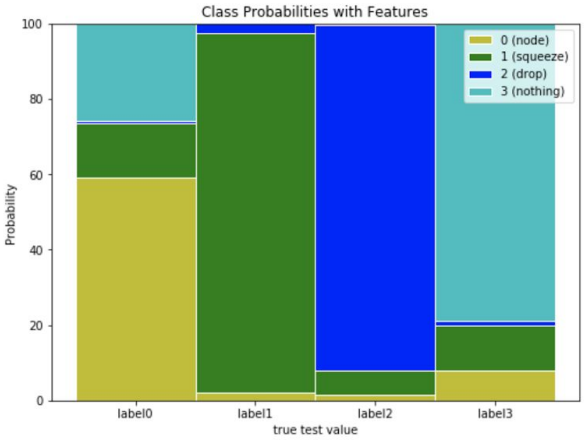


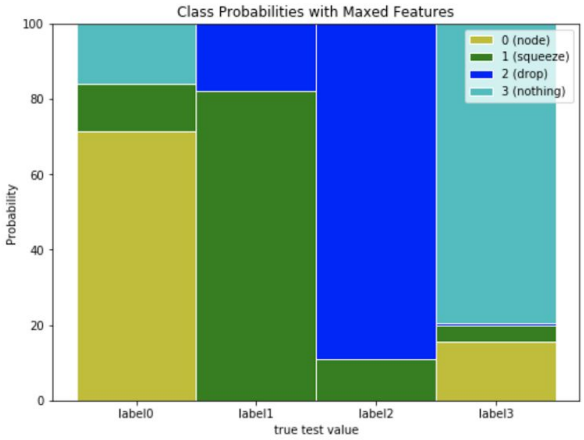




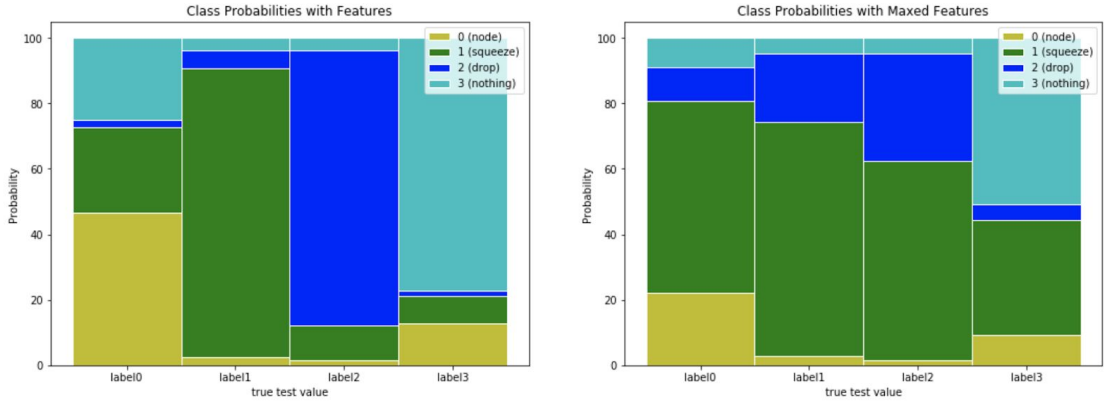


- A lot of the features do really well but some may cancel others out for certain categories and it's clear we need a good one for node tests
- I tried taking the max of each feature instead of taking all the features and ran the basic features vs. no features as well as the cross validation stuff again and it's definitely worse so let's not do that. This is with no cross-validation:





- With cross-validation:



- So going to take out max feature. Here's a screenshot of the block for reference

```
# Takes the max of all features from the 16 datapoints. Output is 72 sets of 1 by 6 data, where each of the 6 points
# is the max of each feature from the set.

def everything_max(array):
    len_dataset, num_features, num_datapoints = np.shape(array)
    maxed_data = np.zeros((len_dataset, 1, num_features))
    for i in range(len_dataset):
        datapoint = array[i]
        D = np.zeros((1, num_features))
        for j in range(num_features):
            D[0,j] = max(datapoint[:,j])
        maxed_data[i] = D
    return maxed_data
```

- Started on leave-one-out cross validation. Seems that it got all the tests right, so I want to make a similar visual representation of the probabilities of the tests, assuming that I can run the classifier for the probabilities instead of just the result label. It's likely that each individual result will have much better results than my 2-fold cross validation because the classifier is being trained on all but one data point and tested on the last

```
# Leave one out cross validation

for train_index, test_index in loo.split(X):
    #print("TRAIN:", train_index, "TEST:", test_index)
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]
    print("Test: {}".format(y[test_index]), "results: {}".format(y_test), end= "        ")
```

```
Test: [0] results: [0]        Test: [0] results: [0]        Test: [0] results: [0]        Test: [3] results: [3]
Test: [1] results: [1]        Test: [1] results: [1]        Test: [1] results: [1]        Test: [1] results: [1]
Test: [1] results: [1]        Test: [1] results: [1]        Test: [3] results: [3]        Test: [0] results: [0]
Test: [1] results: [1]        Test: [1] results: [1]        Test: [1] results: [1]        Test: [1] results: [1]
Test: [1] results: [1]        Test: [1] results: [1]        Test: [2] results: [2]        Test: [2] results: [2]
Test: [2] results: [2]        Test: [2] results: [2]        Test: [2] results: [2]        Test: [2] results: [2]
Test: [2] results: [2]        Test: [2] results: [2]        Test: [0] results: [0]        Test: [2] results: [2]
Test: [2] results: [2]        Test: [2] results: [2]        Test: [2] results: [2]        Test: [2] results: [2]
Test: [2] results: [2]        Test: [1] results: [1]        Test: [1] results: [1]        Test: [1] results: [1]
Test: [1] results: [1]        Test: [0] results: [0]        Test: [3] results: [3]        Test: [1] results: [1]
Test: [1] results: [1]        Test: [1] results: [1]        Test: [1] results: [1]        Test: [1] results: [1]
Test: [1] results: [1]        Test: [1] results: [1]        Test: [1] results: [1]        Test: [1] results: [1]
Test: [1] results: [1]        Test: [1] results: [1]        Test: [1] results: [1]        Test: [1] results: [1]
Test: [1] results: [1]        Test: [1] results: [1]        Test: [1] results: [1]        Test: [2] results: [2]
Test: [3] results: [3]        Test: [3] results: [3]        Test: [3] results: [3]        Test: [3] results: [3]
Test: [3] results: [3]        Test: [2] results: [2]        Test: [2] results: [2]        Test: [2] results: [2]
Test: [0] results: [0]        Test: [0] results: [0]        Test: [1] results: [1]        Test: [1] results: [1]
Test: [2] results: [2]        Test: [2] results: [2]        Test: [3] results: [3]        Test: [3] results: [3]
```

Berkeley Emergent Space Tensegrities

TENSEGRITY HRI MEETING

6.26.20 / 1:00-2:00PM / ZOOM https://berkeley.zoom.us/j/2417093709

My Update

- Been working a lot on cross-validation stuff and a little on ROC stuff
- I was able to use ROC on my classifier and got a curve that looked promising, but other than that I haven't spent a ton of time on ROC. I'm planning on trying to compare ROC with binary classification vs. multi-class and also using ROC to evaluate different types of classifiers
- I started working a lot on using probability calibration with the data so this would output an array of probabilities that a certain label is each class, and I'm excited about this because I think it's already helped a lot with thinking about how to improve our feature engineering because e.g. with features it is very confident about squeeze tests and drop tests but less confident about node and nothing tests, i.e. classifier is about 25% sure that the node tests are a nothing test with the current feature engineering
- This would indicate that I should make a feature that better differentiates a node test from a nothing test
- I also am standardizing the training vs. testing by testing only the last 8 data points, which have 2 tests in each category and then averaging the probabilities for each test
- I also compared the classifier results with no cross-validation vs. the probabilities when training the model on the first 32 data points, validating it on the next 32, and then testing it on the last 8 which is pretty much 2-fold validation. I got really interesting and promising results -- the classifier is discernibly less sure in labeling each test which makes sense because I'm training it on half the original amount of data and then averaging the results of this but the probabilities between this and no validation are similar enough that we know the original probabilities/success of the classifier aren't a fluke
- I also made visuals for all these comparisons so I can share my screen or they're also available on github where I uploaded my research diary

- Really excited! I haven't done too much in comparing cross-validation methods yet so I want to do more with that
- Also want to compare different types of classifiers using ROC and cross-validation, and I still need to do more feature engineering to further differentiate node and nothing tests

Meeting Notes

- Hardware works fine, can resume collecting raw data
- IRB approval -- can do human experiments but don't know what to do about COVID
- Goal for next meeting: have experimental setup
    - Refined data pipeline -- write-up on what kind of data, what classifier, what features, what cross-validation, nominal results, and pretty much anything else you can think of
    - Better data truncation? Embedding label into data

Berkeley Emergent Space Tensegrities

DOCUMENTATION OF PROGRESS

6.22.20

Progress

- Plan for today:
    - k-fold or leave-one-out cross-validation
    - ROC curves for 1 category vs. other categories for all 4
    - Use different classification methods and compare ROC curves among those (using AUC if possible)
    - Come up with new features
    - Become familiar with different types of classifiers
    - TODAY:
        - Develop new features based on probabilities
        - Vary feature sets and look at cross validation
    - All the scikit learn links I have open right now (might be repeated from below):
        - ROC:
            1. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html
            2. https://scikit-learn.org/0.15/auto_examples/plot_roc.html
            3. https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html
            4. ROC/AUC: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html
        - Probability for 3-class classification: https://scikit-learn.org/stable/auto_examples/calibration/pl

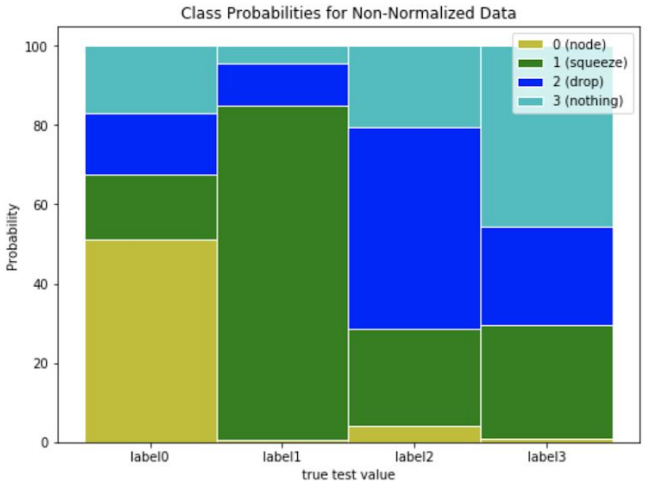
ot_calibration_multiclass.html#sphx-glr-auto-examples-calibration-plot-calibration-multiclass-py

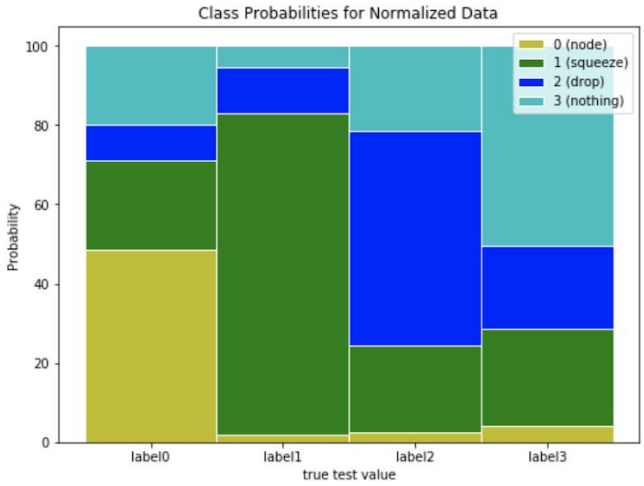
        - ROC with cross validation: https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc_crossval.html
        - General intro to scikit learn: https://scikit-learn.org/stable/tutorial/basic/tutorial.html
        - Classification probability example: https://scikit-learn.org/stable/auto_examples/classification/plot_classification_probability.html
        - K-fold cross validation: https://scikit-learn.org/0.16/modules/generated/sklearn.cross_validation.KFold.html
        - Nested vs. non-nested cross validation: https://scikit-learn.org/stable/auto_examples/model_selection/plot_nested_cross_validation_iris.html#sphx-glr-auto-examples-model-selection-plot-nested-cross-validation-iris-py
        - Visualizing cross validation: https://scikit-learn.org/stable/auto_examples/model_selection/plot_cv_indices.html#sphx-glr-auto-examples-model-selection-plot-cv-indices-py
        - Not really sure:
            1. https://scikit-learn.org/stable/auto_examples/cluster/plot_feature_agglomeration_vs_univariate_selection.html#sphx-glr-auto-examples-cluster-plot-feature-agglomeration-vs-univariate-selection-py
            2. https://scikit-learn.org/stable/auto_examples/ensemble/plot_gradient_boosting_oob.html#sphx-glr-auto-examples-ensemble-plot-gradient-boosting-oob-py
        - Log-loss wikipedia: http://wiki.fast.ai/index.php/Log_Loss
    - Guidelines for Phase 2 research recovery: https://docs.google.com/document/d/1ONuIP5YYeGnLKb5jsvoo0-4xlA_OMDeWWMLiysyUhrI/edit#heading=h.wpe0h7pavdni

Progress

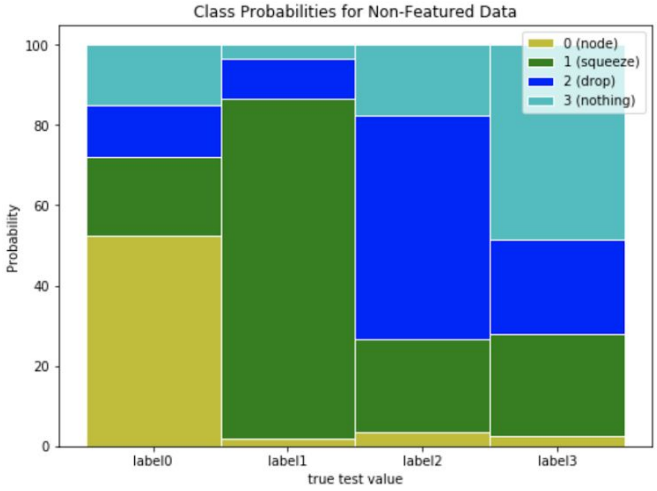
- Figured out a visual for comparing probabilities -- stacked bar plots of probabilities the classifier determined for each type of label

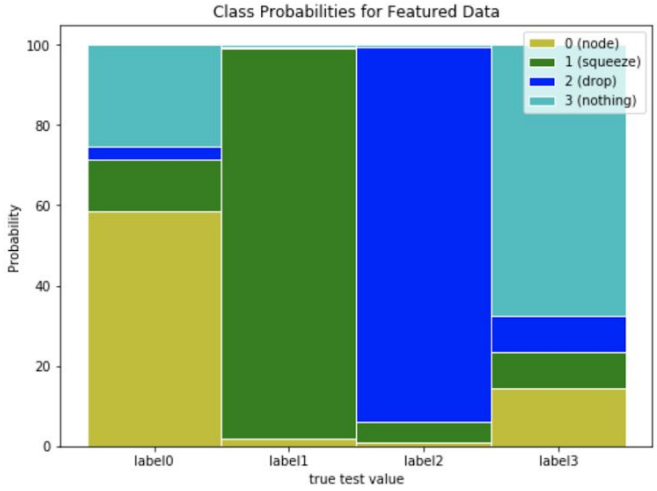
- Normalizing doesn't really make much of a difference (these and probably the rest will be with 64 train data points and 8 test, 2 of each class):



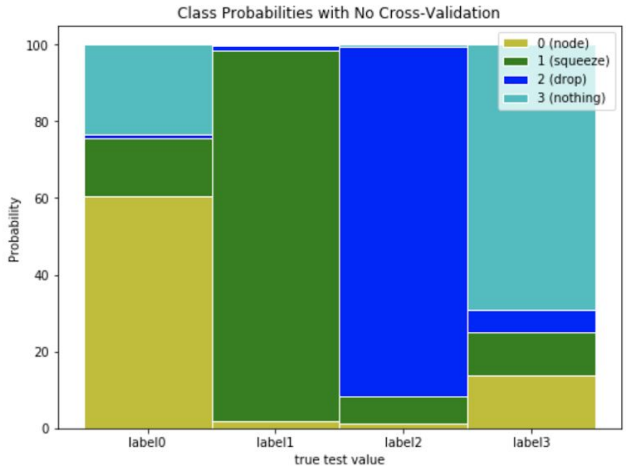


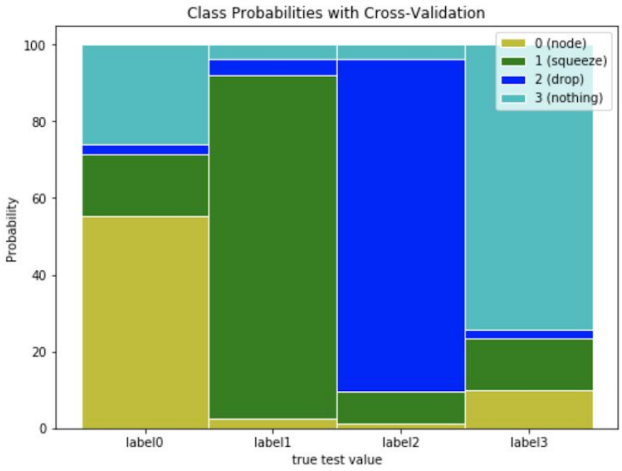
- Non-feature engineered data vs. data with all the features I've come up with so far





- Cross-validation: compares data that was trained on the first 64 data points and evaluated on the last 8 vs. data that was trained on the first 32 data points, validated on the next 32, and then trained on the last 8





- Next:
    - Compare different features
    - Compare max vs. not
    - Come up with new features and compare
    - Compare number of estimators (trees)
    - Compare different numbers of training vs. testing data?
    - Compare types of classifiers
    - Compare max features vs. not max features
    - Cross-validation: compare probabilities for different numbers of k for k-folds
    - Based on probability plots, most urgently need features that differentiate node tests from nothing tests

Berkeley Emergent Space Tensegrities

DOCUMENTATION OF PROGRESS

6.16.20

Progress

- Re-truncated the raw data, adding a bunch of node and nothing tests because I only had about 8 and 10 I think, respectively
- Ran it through the probability classifier, and it didn't go so well. Gonna have to work on that a little bit more before I present anything on it
- Make visualization of probabilities using cross-validation vs. not
- Article: "Cross-Validation in Machine Learning"
    - https://towardsdatascience.com/cross-validation-in-machine-learning-72924a69872f
    - "You need some sort of assurance that your model has got most of the patterns from the data correct, and it's not picking up too much on noise, or in other words it's low on bias and variance."
    - Holdout method
        - Remove part of training data and get predictions from model using the other part it trained on
        - Still suffers from issues of high variance (not sure I understand how)
    - K-fold cross validation
        - Data is divided into k subsets, and the holdout method is repeated k times, so one of the subsets is used as the test set and the other k-1 are used as training
        - Error estimation averaged over all k trials
    - Stratified K-fold cross validation
        - Each fold contains about the same percentage of samples of each target class as the complete set (if there is an imbalance in the categories)

    - Leave-P-Out cross validation
        - Leaves p data points out of training data. If n data points in original sample, then n-p samples are used to train the model and p are used as validation. Repeated for all combinations in which original sample can be separated
- Cross validation video
    - https://www.youtube.com/watch?v=fSytzGwwBVw
        - Side note: understand different machine learning methods
            1. Logistic regression
            2. Support vector machines
            3. K-nearest neighbors
            4. Random forest
            5. etc.
    - “Cross validation allows us to compare different machine learning methods and get a sense of how well they will work in practice.”
    - Need to estimate the parameters for the machine learning methods (training the algorithm)
    - Evaluate how well the machine learning methods work (how well the model categorizes new data)
    - Compare methods by seeing how well each one categorized the test data
    - Cross validation uses all the data for testing (in separate chunks) one at a time and summarizes the results at the end
    - e.g. start by using the first 75% to train and last 25% to test, keeps track of how well the test did
    - Then it may use first 50% and last 25% for training, and second to last 25% for testing
    - Eventually, every block of data is used for testing and compare methods
        - This would be four-fold cross validation, but the number of blocks is arbitrary

    - If you call each individual sample a block, this is “leave one out cross validation”
    - Could use cross validation to find the best value for tuning parameters
- But first, confusion matrix video
    - https://www.youtube.com/watch?v=Kdsp6soqA7o
    - To summarize how each method performed on the testing data, we can use a confusion matrix
    - The rows on the confusion matrix correspond to what the machine learning algorithm predicted, and the columns correspond to the known label
    - Top left corner is true positives, and bottom right is true negatives
    - Bottom left is false negatives, and top right are false positives
    - Compare confusion matrices of different algorithms
    - If 3 categories, then the confusion matrix has 3 rows and 3 column (size is determined by number of things we want to predict). The diagonal shows how many samples were correctly classified
- Sensitivity and Specificity video
    - https://www.youtube.com/watch?v=vP06aMoz4v8
    - How to calculate sensitivity and specificity
    - Sensitivity tells us what percentage of patients with heart disease were correctly identified
        - Sensitivity = true positives/(true positives + false negatives)
    - Specificity tells us what percentage of patients without heart disease were correctly identified
        - Specificity = true negatives/(true negatives + false positives)
    - How to calculate for a confusion matrix with 3 rows and 3 columns: have to calculate different sensitivities and specificities for each category

        - Sensitivity = (number that were predicted correctly in the category)/(number that were predicted correctly in the category + number that were predicted incorrectly in the category)
        - Specificity = (number that were predicted correctly outside the category)/(number that were predicted correctly outside the category + number that were predicted out of the category incorrectly as the category)
    - We can use sensitivity and specificity to help us decide which machine learning method is best for our data
    - If correctly identifying positives is the most important, we should choose a method with higher sensitivity
- ROC and AUC explained video! By same guy
    - https://www.youtube.com/watch?v=4jRBRDbJemM
    - When doing logistic regression, the y-axis is fitted to the probability that the data is in one of two categories
    - Set threshold to classify
    - Lower threshold increases false positives, reduce false negatives, and reduce number of true negatives
    - In order to test the different thresholds, we can use receiver operator characteristic (ROC) graphs to summarize the information
    - Y-axis: true positive rate (sensitivity)
    - X-axis: false positive rate (1 - specificity = false positives/(false positives + true negatives)
    - Use one threshold → plot point on ROC curve
    - Points above the diagonal indicate that the proportion of correctly classified samples (true positives) is greater than the proportion of samples incorrectly classified (false positives)
    - The ROC graph summarizes all of the confusion matrices that each threshold produced
        - Is there a way to see these thresholds in scikit learn?
        - Depending on how many false positives you are willing to accept, different points can be optimal thresholds

    - The area under the curve (AUC) makes it easy to compare one ROC curve to another
    - Can also use precision instead of false positives
        - Precision = true positives/(true positives + false positives) = proportion of positive results that were correctly classified

Plan

- Try to implement k-fold cross validation or another type. Understand metrics and try to come up with a visual for how well the model is doing
    - https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html
    - https://scikit-learn.org/stable/modules/cross_validation.html
    - https://scikit-learn.org/0.16/modules/generated/sklearn.cross_validation.KFold.html
- Calculate ROC curves for 1 category vs. other categories for all 4
- Use different classification methods and compare ROC curves among those (using AUC if possible)
- Compare ROC with multiple categories vs. 1 vs. rest

My Update/Email to Andrew

- Was able to use the ROC method on my classifier by slightly modifying the format of the data. Used same data processing techniques and features from my model to condense the data and get it into the right format
- Got a curve that looked promising based on what I had previously learned but I realized I wasn't sure exactly what I was looking at, so I watched a bunch of clarifying videos and now I understand better
- I found on scikit learn how to do ROC with both binary classification and multi-class, so I'm going to implement both those methods and compare
- I also tried using a probability calibration method where instead of the classifier outputting a guess for the label it outputs the probabilities of the sample being each class

- Learned that if the classifier is trained on all the data points designated for training then it is overly confident in its predictions which leads to a large log-loss, and this led me to investigate various cross-validation methods. Worked on this before the ROC stuff, and I think using both cross-validation and ROC could be really useful to develop and use in the paper to show that our model work and isn't just lucky
- It was also really interesting to look at the probabilities using basic cross-validation vs. not, i.e. model is less sure of correct classification when using cross-validation but often still correct (highest probability for correct label)
- Also the non-validated classifier is really sure about the node tests and squeeze tests (60-90% sure) but isn't as sure about drop tests (30-40%) and thinks nothing data is node tests. I think these probabilities are really useful for evaluating my existing features and for thinking about how to develop new/better ones.
- I also re-truncated some of the raw data and added it to my set because I had relatively few node tests and nothing data
- Going to try to get concrete results on the ROC method and cross-validation stuff to see which classifier is best and which cross-validation method is best
- Planning on working on k-fold and leave-one-out cross-validation methods, and hopefully come up with some sort of visual for how well these are working. I also want to familiarize myself with different types of classifiers in order to compare results, as well as compare multi-class ROC vs. binary classification
- I'm going to spend a little more time trying to develop new features based on the class probabilities, but otherwise mostly work on the validation stuff

Berkeley Emergent Space Tensegrities

DOCUMENTATION OF PROGRESS

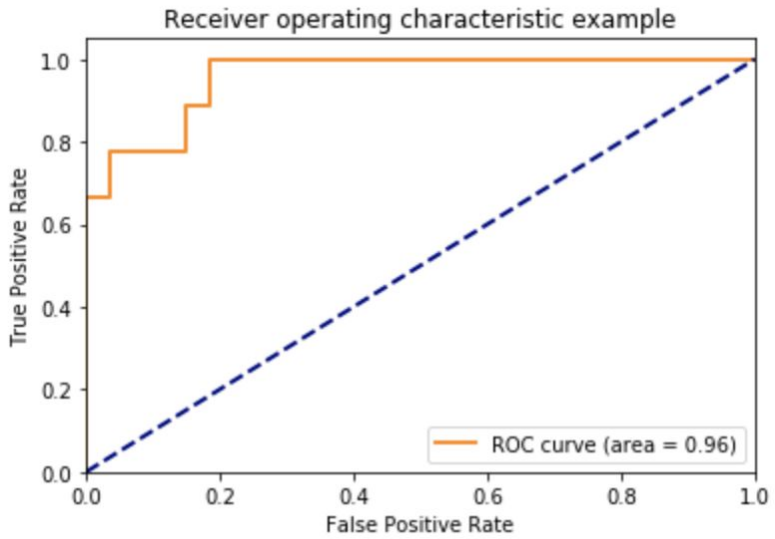
6.10.20

Progress

- Trying to convert my current ML script into one that is formatted for ROC packages
- Taking data → normalize → “featurify” → take the max of each feature → make 2D array of featured data (ROC uses 2D data whereas I originally had it in 3D, but easy to convert to 2D)
- Following along with the sci kit learn example script: https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html

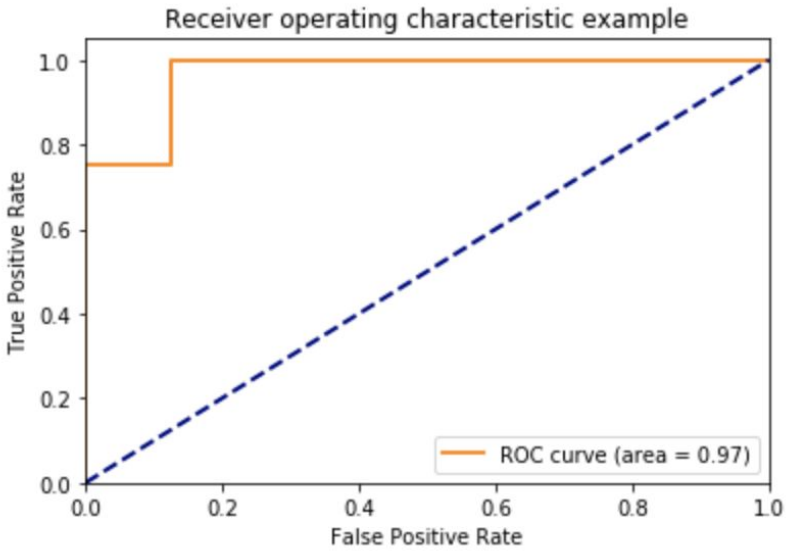
In [94]:
```
# shuffle and split training and test sets
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=.5, random_state=0)
```

In [95]:
```
# Learn to predict one class against the other
classifier = OneVsRestClassifier(svm.SVC(kernel='linear', probability=True,
                                         random_state=random_state))
y_score = classifier.fit(x_train, y_train).decision_function(x_test)
```

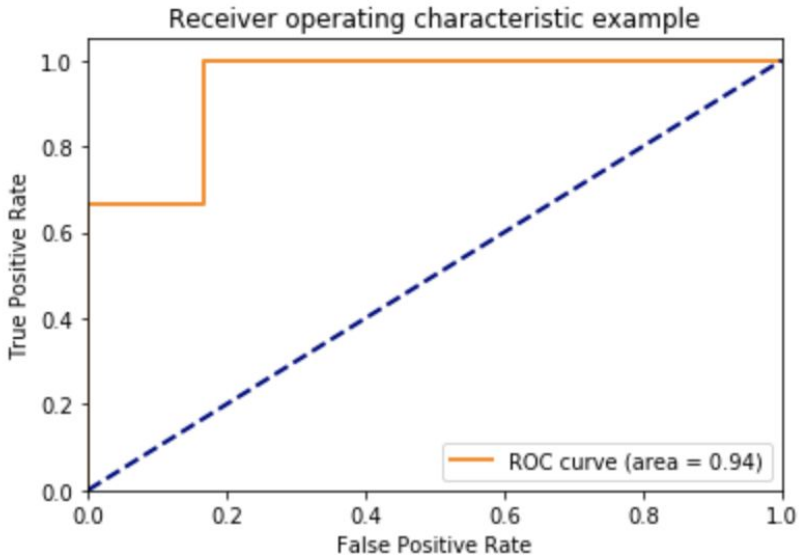


- Got good results, but not super sure what they mean honestly
    - Have Andrew explain more in detail?

- For 12 testing points (instead of half and half above):



- For a testing sample of 9:



- How does it know which are true positive and which are false positive? Does it just check the labels against the results of the model?
- Where can we see the model results? I don't understand the value of y_score
- Will probably continue on to try using probability models because those seem like they might be more intuitive to me

- Just kidding! I do understand (sort of) the value of y_score. This is for a sample size of 5 tests, and y_score is a measure of the probability of each label for each test

```
In [142]: y_score

Out[142]: array([[  1.39051113,  16.61217209, -22.44062681,  -7.19542744],
                 [-36.89831605,  13.32412107,  -0.62279804, -20.68490746],
                 [ -2.16851379,  19.38150573, -13.61203736, -14.72544425],
                 [-29.34769553,   8.55962063,  -1.18369171, -22.06859798],
                 [-27.2132914 ,   2.40230902,   6.32252924, -27.48814128]])
```

- Still next, going to look into scripts that return probabilities, not y_score because I'm not really sure what the exact values mean
    - I thought it was values closest to 0 but that doesn't make sense for some of them??
- Basic probability script, following along with https://scikit-learn.org/stable/auto_examples/calibration/plot_calibration_multiclass.html#sphx-glr-auto-examples-calibration-plot-calibration-multiclass-py

```
In [140]: # Train uncalibrated random forest classifier on whole train and validation
          # data and evaluate on test data
          clf = RandomForestClassifier(n_estimators=25)
          clf.fit(x_train_valid, y_train_valid)
          clf_probs = clf.predict_proba(x_test)
          score = log_loss(y_test, clf_probs)

In [141]: # Train random forest classifier, calibrate on validation data and evaluate
          # on test data
          clf = RandomForestClassifier(n_estimators=25)
          clf.fit(x_train, y_train)
          clf_probs = clf.predict_proba(x_test)
          sig_clf = CalibratedClassifierCV(clf, method="sigmoid", cv="prefit")
          sig_clf.fit(x_valid, y_valid)
          sig_clf_probs = sig_clf.predict_proba(x_test)
          sig_score = log_loss(y_test, sig_clf_probs)

In [142]: clf_probs

Out[142]: array([[0.4 , 0.  , 0.  , 0.6 ],
                 [0.52, 0.  , 0.  , 0.48],
                 [0.76, 0.  , 0.  , 0.24],
                 [0.44, 0.  , 0.  , 0.56],
                 [0.88, 0.  , 0.  , 0.12],
                 [0.  , 0.28, 0.72, 0.  ],
                 [0.  , 0.28, 0.72, 0.  ],
                 [0.  , 0.24, 0.76, 0.  ],
                 [0.92, 0.  , 0.08, 0.  ],
                 [0.76, 0.04, 0.  , 0.2 ],
                 [0.  , 0.48, 0.52, 0.  ],
                 [0.4 , 0.24, 0.36, 0.  ],
                 [0.  , 0.32, 0.68, 0.  ],
                 [0.  , 0.24, 0.76, 0.  ],
                 [0.88, 0.  , 0.  , 0.12],
                 [0.68, 0.  , 0.  , 0.32]])

In [134]: y_test

Out[134]: [3, 3, 3, 3, 3, 2, 2, 2, 0, 0, 1, 1, 2, 2, 3, 3]
```

- clf_probs returns the probability of the test being classified as each label
- Super cool!! If you compare the labels with the highest probability, some are actually are right! e.g. test 9 is 92% sure that it is a node test, which is correct!
- Can use these probabilities, looking at the plot of the raw data, to try to figure out why the classifier isn't working for some of the tests
    - e.g. really things most of the "nothing" tests are node tests -- should add more nothing data or figure out feature that would differentiate the node test from the nothing data, because the nothing data definitely sometimes has some random bumps that would look like a node test
- Figure out my own way of visually understanding the probabilities, or make my own metric for determining how accurate the classifier actually is
- Make improvements -- figure out how this stuff really works
- Maybe make an initial plot of how sure the classifier is about each label, on average for positive tests

My Update (Tensegrity HRI Meeting 6.12.20 cancelled)

- Was able to use the receiver operating characteristic method on my machine learning model by slightly modifying the format of the data, but I used all the same data processing techniques and features from my existing model which proved to be fairly successful
- The curve I got for ROC looked promising but I'm not entirely sure what's going on
- I get that the top left corner is the "ideal point" because it's a false positive rate of 0 and a true positive rate of 1 but I don't know if I understand where the curve comes from
- Was able to binarize the output, so each of the 4 labels turned into an array of three 0's and one 1, depending on what class if was
- Made an ROC curve using binary prediction (which I also don't entirely understand)
- Not done trying to understand ROC, but I also tried using a probability calibration method where instead of the classifier outputting a guess for the label it outputs the probabilities of the sample being each class

- Learned that if the classifier is trained on all the data points designated for training then it is overly confident in its predictions which leads to a large log-loss
- So if you train the classifier on less data and use the “sigmoid” method on the remaining training data points to calibrate the classifier, this reduces the confidence of the predictions which results in a lower log-loss
- Increasing the number of base estimators also decreases log-loss
- The probabilities are really interesting to look at, e.g. the non-validated classifier is really sure about node tests and squeeze tests (60-90% sure) but isn’t sure about drop tests (30-40%) and thinks nothing data is node tests
- I want to think about these probabilities more and try to come up with more features that would differentiate between these tests based on these observations

Plan for this week

- Look at CS189!! Will definitely clarify a lot of stuff
    - http://people.eecs.berkeley.edu/~jrs/189/?fbclid=IwAR2BH8K-mMfbEbBSepFumKQCgqV2y5B_b1L5mHlyBYXYx_RiDCpgaYZWJXE
- Understand ROC more
- Refresh on log-loss and sigmoid curves (3Blue1Brown intro to ML series)
    - https://www.youtube.com/watch?v=aircAruvnKk&t=3s
- Brainstorm more features based on probabilities
- Come up with my own metric on how to understand these probabilities?
- Email Andrew progress

Berkeley Emergent Space Tensegrities Laboratory

TENSEGRITY HRI MEETING

6.5.20 / 1:00-2:00PM / ZOOM https://berkeley.zoom.us/j/2417093709

My Update

- Brainstorm of ways to conduct experiments while being careful about COVID-19:
    - 3 people:
        - We are already capable of remaining 6 ft apart while conducting the tests since one person is operating the camera, one is manipulating the robot, and one is collecting data on the computer
        - Everyone must sanitize hands and be clean when they come into the lab, remain at their respective stations (robot, camera, computer)
        - If the wires get disconnected from the board or the robot needs to be reset, the person operating the robot must fix/reset it
        - Everyone wears masks
        - If someone's help is really needed at a different station, maintain distance of 6 ft apart and use sanitary wipe to clean
        - Or everyone washes hands, puts on gloves, and avoids touching their face the whole time in lab. Then we wouldn't need sanitary wipes as often
    - 2 people:
        - Same as above, but one person manipulates the robot and writes on white board, and other person operates both camera and computer
- Read article comparing ROC (receiver operating curves) and precision recall curves https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/

- Learned that ROC curves should be used when there are roughly equal numbers of observations for each class -- would be OK to use this when we have a larger set of data in which we have purposefully run a set number of tests (although not sure exactly what they mean by “roughly equal”
- Precision-recall curves should be used when there is a moderate to large class imbalance
- ROC curves “present an optimistic picture of the model on datasets with a class imbalance” and “might be deceptive and lead to incorrect interpretation of the model skill”
- PR plots can provide an accurate prediction of future classification performance because they evaluate fraction of true positives among positive predictions and doesn’t account for true negatives
- Either way, these curves are dealing with probabilities and thresholds to determine how sure the model is about making a classification or if it’s not sure, so we could start by converting our model to one that predicts the likelihood of a certain class and this would be more reliable data to publish than just the rate the model is correct

Plan for this week

- Going to start trying to implement ROC methods between two classes of data and see if I can get okay results for the area under the curve metric
  - Sci kit learn links:
    - https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html
    - https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html
    - https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc_crossval.html
    - https://scikit-learn.org/0.15/auto_examples/plot_roc.html
- Also going to start looking into models that return probabilities of something being in a certain class instead of the model just labelling it as the most likely one
- General sci kit learn links:

    - https://scikit-learn.org/stable/
    - https://scikit-learn.org/stable/getting_started.html
    - https://scikit-learn.org/stable/auto_examples/classification/plot_digits_classification.html#sphx-glr-auto-examples-classification-plot-digits-classification-py
- Other machine learning-related stuff
    - https://machinelearningmastery.com/how-to-develop-and-evaluate-naive-classifier-strategies-using-probability/
    - https://machinelearningmastery.com/what-is-bayesian-optimization/
- Don't forget to keep working on human factors training:
    - https://cphs.berkeley.edu/quickguideCITItraining.pdf

Meeting Notes

- One label vs. rest of labels, not one vs. one for all 24 (?) combinations
- More trees improve classifier performance

Berkeley Emergent Space Tensegrities

TENSEGRITY HRI MEETING

5.29.20 / 1:00-2:00PM / ZOOM https://berkeley.zoom.us/j/2417093709

Meeting Notes

- How to move forward with experimenting
    - Work remotely, use first prototype to get more data because we really need more data to write the paper
- Think about ways to conduct experiments effectively given the circumstances
- Look through Andrew's critiques
- Understanding how we're analyzing the results of our classification model
- So far we're just evaluating the ability of model to output correct label
- Can determine probability of model to label correctly
- Another metric: area under the receiving operating curve -- compares true positive rate vs. false positive rate. Curve based on number of samples (this is for binary classification)
- Packages in sci kit learn for AUC. see what the results are for our current model
- Biasing: the way we are training the model is such that it is predisposed to look for certain scenarios rather than looking or anything
- Look at different model types on sci kit learn
- In general learn sci kit learn
- Conditional probabilities to inform our understanding of how our model will perform based on different conditions
- CS 189 website and Data 100 textbook

Berkeley Emergent Space Tensegrities

PAPER SECTION DRAFT

5.9.20

V. EXPERIMENTS

Our research goal is to show that a tensegrity robotic system can be used to robustly differentiate between various interactions it can experience with humans. In order to prove that the classification model described above may be used reliably to distinguish between these interactions, it must be verified on an example dataset that describes a variety and multitude of human interaction data. We thus designed our experiments with our tensegrity robotic system around a defined set of human interactions to use with the system, modeled after the types of interactions we would expect to see in applications of this research. Our goal is to assemble a large set of data from these sample interactions to show that the classification modeling can be used to distinguish between them.

As described in detail in previous sections, we assembled a 6-bar tensegrity (compliant) robot prototype equipped with an array of 12 force sensing resistors placed on the ends of the compression rods, in addition to 3 accelerometers located in the central payload of the system. We hypothesized that distinct interactions would yield distinct datasets of force sensor and accelerometer data, and thus be distinguishable by a classification model. The tensegrity robot was tested on the following human interactions:

- Drop
  - A researcher holds the robot from a consistent height and releases it
  - Data of the release and impact, until the tensegrity comes back to rest, is recorded
- Squeeze
  - The robot is placed on a table. A researcher uses a large piece of material (e.g. plexiglass) to place on the top of the robot and compress the tensegrity structure three times
  - Data of the three successive squeezes is recorded together

Qualitatively, it is expected that the data between these two types of interactions, when plotted, will be visibly very distinct. In particular, when dropped, the tensegrity experiences very high forces within a very short interval due to its impact with the floor, as well as erratic vibrations post-impact. On the other hand, the tensegrity will experience lower, smoother forces over longer periods of time when the entire structure is intentionally compressed by a human. Based on these expectations, we expect to be able to both visually distinguish between the interactions as well as apply our classification model to distinguish between them based on distinct patterns in the data.

To make the experimentation robust, these interactions were tested from a variety of starting orientations. To expand upon the description of the tensegrity robotic system given in Section III.C Force-Sensor Array, we will describe the specific locations of the numbered force sensors for clarity in the description of the experimental setup. The tensegrity has 12 distinct nodes, each of which houses a single force sensing resistor, and each node and respective sensor is labeled with an integer from 1 to 12. We expect, depending on the orientation of the system

when the interaction takes place, and despite the fact that the forces propagate across the system to all compression elements, tension elements, and nodes, that different sensors will see different forces. For example, when squeezed, we expect that the force sensors in the nodes that are in contact with the table and the plexiglass will return higher force readings than the nodes on the sides.

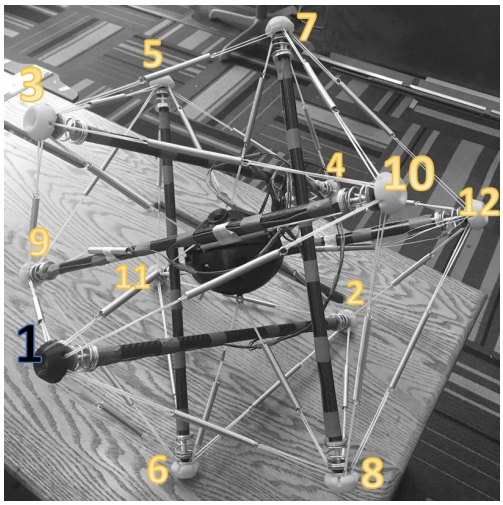


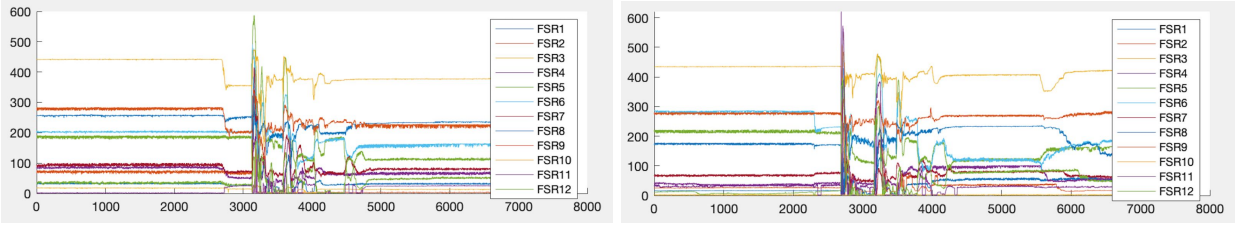
Fig. [] 6-bar tensegrity with labeled nodes

Hence, a dataset from e.g. two different drops, each with a different starting orientation, may look distinct. Therefore from a data analysis standpoint, in order to ensure that our classification model is as robust as possible, it is necessary to perform experiments with many various starting orientations.

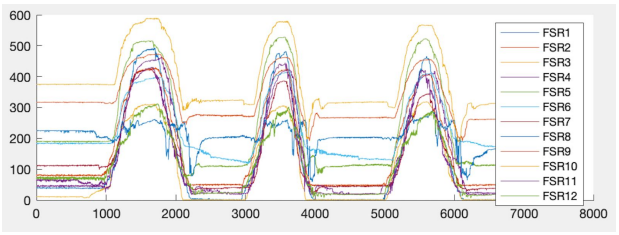
** Describe experimental setup in more detail, including using wireless connection to microcontroller inside payload to collect FSR data **

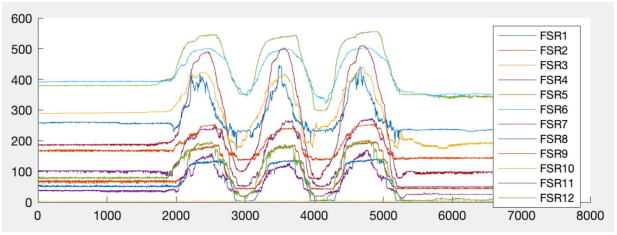
We performed drop tests from a variety of orientations, classifying each test as either “closed-face,” “open-face,” or “double rod,” depending on the node or string orientation we expected the tensegrity to land on. The squeeze tests were each classified as either “closed-face,” “open-face,” “double rod,” or “single rod.” Examples of two drop tests and two squeeze tests performed from different orientations are shown below:

Drop:



Squeeze:



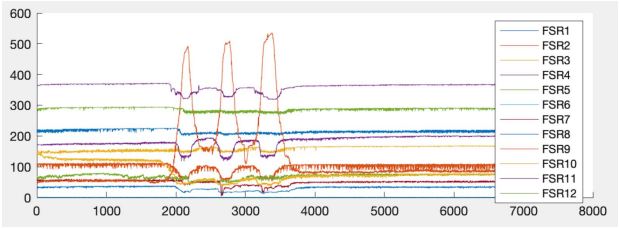


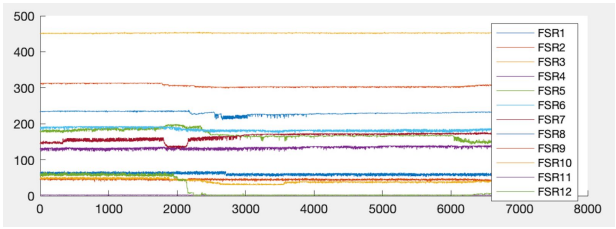
Accelerometer data is plotted separately and not shown here. Visually, the force plots of the drop and squeeze tests appear very distinct from one another. We are also able to infer that the two drop tests and two squeeze test were executed from different orientations based upon which FSR has the highest force readings. Note also that each FSR also has its own degree of noise, which may affect the accuracy of our classification model.

A classification model ideally has thousands of data points with which to train. Due to limited time we were only able to collect a few dozen squeeze and drop tests. Because we aim to prove that our classification model works as a proof of concept, and because our plans for the future of this research include addition more interaction classes to our data collection (including but not limited to roll tests and throw tests), we decided to include two more classes of data we passively collected to train our proof-of-concept classification model on:

- ➢ Node tests
  - A researcher squeezes a single node three times
  - Data was originally collected while doing this simply in order to make sure the wireless connection to the robot was working properly
- ➢ Null tests
  - Data is collected when no interaction is happening

Examples of a node test (left) and a null test (right) are shown below:





Discussions on the design of our classification model and results of running the model on our dataset follow in the next two sections.

VI. CLASSIFICATION MODEL

Designing and implementing a working classification model on our dataset completes our proof-of-concept that physical interaction can serve as a reliable platform for human-robot interaction. In proper human-robot interaction, the ideal tensegrity robotic system responds in certain ways according to specific manipulations of the compliant system that the human performs intentionally. Therefore the robotic system is required to properly classify the interaction based on live force sensor data in order to respond accordingly. We have not yet developed the technology in order to implement real-time response, but our preliminary classification model and results will show that it is possible to leverage physical interaction in human-robot interaction.

We began the design of our classification model based on our experimental data and the classes of interaction we specifically chose to analyze. The purpose of this model is simple: to properly classify each test data point as a certain type of test. A more specific model might do this as well as identify the starting orientation of the robotic system based on its highest force readings. For now, we will proceed with just distinguishing between categories of tests.

. . .

- Comments from Andrew in original google doc 5.12.20
    - https://docs.google.com/document/d/1XcnhHoGMeiBvQVPjQDSxanErEMPMxFOP8WKKO7FQLyg/edit

Berkeley Emergent Space Tensegrities

TENSEGRITY HRI MEETING

5.6.20 / 1:00-2:00PM / ZOOM https://berkeley.zoom.us/j/2417093709

Meeting Notes

- Discussion on datasets
- Normalize STD = 1
- Tuning sensitivity -- making sure random noise isn't classified as a test
- Want it to be as insensitive to outliers as possible
- Different size interval tests
- Discrete constant time windows over range of datatypes
- Featurization works well
- Use slopes of accelerometer data
- PCA: principle component analysis -- modes of a signal are its features
- Maximum jerk, total impulse
- Try minimizing features
- Numerical approach to see what features are the best
- Try slow, real-time classification with moving window?
- Spend more time on different classifiers, but can probably press forward with the best one -- random forest
- Random forest -- averages subset of decision trees for robustness
- Message: given FSR data and human manipulation of robot, can infer useful information for HRI
- How will we analyze the results of the classifier?
    - Confusion matrices -- true positive, false positive rates
    - Way to quantify ability of binary classifier to robustly classify

    - Find out a way to do this with more tests
- Data 100 textbook
- Decide what matrices to use to determine accuracy of classifiers
- What quantitative results from classifiers to use with results

Berkeley Emergent Space Tensegrities

TENSEGRITY HRI MEETING

5.1.20 / 1:00-2:00PM / ZOOM https://berkeley.zoom.us/j/2417093709

My Update

- Found a mistake that was lowering my model's accuracy (kept in the time data)
- Played around a lot with different features and tried doing various models with all data vs. condensed data, featured data vs. no features, with accelerometer data vs. without accelerometer data -- it's all on my Jupyter notebook which is available on my github branch

In [5]:
```
# This feature returns the percent of data points that are below 75% of the max

def three_quarters(column,percent):
    max = np.max(column)
    three_fourths = percent*max
    e = []
    j=0
    for i in range(len(column)):
        if column[i] <= .75*np.max(column):
            j+=1
    return j/len(column)
```

In [6]:
```
# This feature takes the max of each column, takes the data point [change_factor] ms after that, a
nd then normalizes
# the difference

def change_in_max(column, change_factor):
    max_index = np.argmax(column)
    minus_change_index = max_index - change_factor
    if minus_change_index > 0:
        minus_change_value = column[minus_change_index]
    else:
        minus_change_index = max_index + change_factor
        minus_change_value = column[minus_change_index]
    diff = np.max(column) - minus_change_value
    normalized = diff/np.std(column)
    return normalized
```

In [7]:
```
# This function takes all 72 data points of size (7001,16) and puts them into a feature-engineered
array.
# Desired shape: 72 data points of size (6,16) --> (72,6,16)

def features(array, three_quarters_percent, change_in_max_change_factor):
    len_dataset, _, number_of_columns = np.shape(array)
    number_of_features = 6
    condensed_data = np.zeros((len_dataset,number_of_features,number_of_columns))
    for j in range(72):
        data_point = array[j]
        number_of_columns = len(data_point[0,:])
        D = np.zeros((number_of_features,number_of_columns),dtype=np.float64)
        for i in range(number_of_columns):
            column = data_point[:,i]
            D[0,i] = np.mean(column)
            D[1,i] = np.var(column)
            D[2,i] = np.max(column)-np.min(column)
            D[3,i] = three_quarters(column,three_quarters_percent)
            D[4,i] = change_in_max(column,change_in_max_change_factor)
            D[5,i] = np.sum(column)/np.max(column) #or np.std(column). MAX WORKS THE BEST! went fr
om 80% or so to 100%
        condensed_data[j]=D
        if np.mod(j,6) == 0:
            print(j, end=" ")
    return condensed_data
```

In [8]:
```
# Condense data into averages over a specified number of data points

def condense(array, condense_factor):
    len_dataset, _, number_of_columns = np.shape(array)
    length = len(array[0])
    condensed_length = int((length/condense_factor))
    condensed_data = np.zeros((len_dataset,condensed_length,number_of_columns))
    for j in range(len_dataset):
        data_point = array[j]
        for k in range(number_of_columns):
            column = data_point[:,k]
            for i in range(condensed_length):
                condensed_data[j][i,k] = np.mean(column[i*condense_factor:(i*condense_factor+conde
nse_factor)])
    return condensed_data
```

In [9]:
```
# Takes the max of all features from the 16 datapoints. Output is 72 sets of 1 by 6 data, where ea
ch of the 6 points
# is the max of each feature from the set.

def everything_max(array):
    len_dataset, num_features, num_datapoints = np.shape(array)
    maxed_data = np.zeros((len_dataset, 1, num_features))
    for i in range(len_dataset):
        datapoint = array[i]
        D = np.zeros((1, num_features))
        for j in range(num_features):
            D[0,j] = max(datapoint[:,j])
        maxed_data[i] = D
    return maxed_data
```

In [10]:
```
# Make model

def make_model(dataset,num_train):
    nsamples, nx, ny = np.shape(dataset)
    reformated_data = np.reshape(dataset,(nsamples,nx*ny))

    train = reformated_data[0:num_train]
    test = reformated_data[num_train:nsamples]
    train_labels = labels[0:num_train]
    test_labels = labels[num_train:nsamples]

    # Return percent of correct labels
    def percent_correct(test_dataset, test_labels):
        total = 0
        a = model.predict(test_dataset)
        for i in range(len(a)):
            if a[i] == test_labels[i]:
                total += 1
        correct = total/len(a)*100
        return correct

    model = GaussianNB().fit(train, train_labels)
    print("Predicted labels = " + str(model.predict(test)))
    print("Correct labels =   " + str(np.asarray(test_labels)))
    return percent_correct(test, test_labels)
```

- I got 100! By playing around with one of my features -- I think it was Yash's idea to take the sum of all data, but I was testing out different ways to normalize (max was the best)
    - np.sum(column)/np.std(column)
    - np.sum(column)/np.max(column)
- Trying different numbers of training vs. testing data points, different features, using maximization or not, using normalization or not:

```
In [12]: # Normalize and feature-ify the data
         normalized_data = data/np.max(data)
         normalized_featured_data = features(normalized_data,.75,50)
         np.shape(normalized_featured_data)

         # Make non-normalized, featured dataset (dataset size (72,6,15))
         featured_data = features(data,.75,50)

         # Make dataset that only has max of each feature
         maxed_data = everything_max(featured_data)

         0 6 12 18 24 30 36 42 48 54 60 66 0 6 12 18 24 30 36 42 48 54 60 66

In [13]: # Condense and featurify the normalized data
         condensed_data = condense(normalized_data,10)
         condensed_data = features(condensed_data,.75,5)

         0 6 12 18 24 30 36 42 48 54 60 66

In [19]: np.shape(maxed_data)

Out[19]: (72, 1, 6)

In [40]: # Train model on normalized and feature-ified data
         make_model(normalized_featured_data,60)

         Predicted labels = [3 2 2 2 0 0 1 1 2 2 0 3]
         Correct labels =   [3 2 2 2 0 0 1 1 2 2 3 3]

Out[40]: 91.66666666666666

In [41]: # Train model on non-normalized, featured data (dataset size (72,6,15))
         make_model(featured_data,60)

         Predicted labels = [3 2 2 2 1 0 1 1 2 2 1 3]
         Correct labels =   [3 2 2 2 0 0 1 1 2 2 3 3]

Out[41]: 83.33333333333334

In [43]: # Train model on max-feature-engineered data
         make_model(maxed_data,60)

         Predicted labels = [0 2 2 2 0 0 1 1 2 2 1 3]
         Correct labels =   [3 2 2 2 0 0 1 1 2 2 3 3]
```

Meeting Notes

- Work on with paper: force sensor array, my work relevant to classification modeling, have a good understanding of experiments
- Andrew writes most, we contribute some and comment
- Novel HRI approach

- Importance: 1st attempt of a platform for real mobile robotic platform. In future, propose that robotics will be largely used by humans. How leverage contact to make interactions better. Robot that is easy to manipulate, leverages contact to improve communication. Here are some preliminary results on how to do that
    - Intellectual merit
- Pose classification modeling on context of force sensing, argue why it works (not practically), thinking about how we will do classification modeling
- Details: what classification model, what featurization
- How design to best control for sources of error -- how error can make our classification models more robust?
- Analyzing results of how well classification model performs. Exclusivity or reliability, not just how right it is with given data
- Tensegrity system can use force sensing data to distinguish between interactions which has implications for the future of human-robot interaction. Info from sensors is useful, at the very least for proof of concept/demonstration of classification abilities from data
- If we had more reactions (everything you could think of) → much better argument

Plan for next week

- Pick one specific area of the paper we want to take a stab at and try to flesh out bullet points more. Have done by Wednesday
- How best to express results in terms of classifier
- Comb through results -- figure out how to express results based on different metrics for interpreting results
- Meeting for Tuesday -- consolidate and merge results that everyone has put together

Berkeley Emergent Space Tensegrities

TENSEGRITY HRI MEETING

4.24.20 / 1:00-2:00PM / ZOOM https://berkeley.zoom.us/j/2417093709

Meeting Notes

- Paper: everything up to and including data processing (not 2nd iteration prototype -- maybe in future work). 1st paper ever published on tensegrity HRI
- Physical HRI with mobile robots, compliant robots
- Compliant mobile robotic system capable of doing physical HRI
- Akhil's software update: created function that creates dataset of different sizes, splits all data up into chunks of the same size
- Discussions of how to best combine approaches -- results for paper
- Ensemble learning method

Plan for next week

- Moving forward, can put max or min of each feature into smaller dataset, try normalizing data against max of everything
- Paper contribution -- experimental design and results
- Idea of what we'll be working on for paper

Berkeley Emergent Space Tensegrities

TENSEGRITY HRI MEETING

4.17.20 / 1:00-2:00PM / ZOOM https://berkeley.zoom.us/j/2417093709

My Update

- Put a bunch of features into my script including some basic ones like mean, max, and variance and a couple other ones that I thought of, been interesting to to run the script taking out certain features and seeing which ones affect the results
- My results have improved slightly since last time but what's interesting is that my script refuses to guess that anything is a drop which is weird because I was hoping that my features would target the differences between drops and the other tests, so maybe I should send Andrew what I have and he can give me feedback
- Also tried compressing my data into averages over ten or so data points but thus far it hasn't made a difference but I'll keep working on it

Plan for next week

- Total force -- sum every data point, maybe normalize with max or standard deviation
- Max change in data across a few data points
- Take max of each feature → only ID dataset for training

Berkeley Emergent Space Tensegrities

TENSEGRITY HRI MEETING

4.10.20 / 1:00-2:00PM / ZOOM https://berkeley.zoom.us/j/2417093709

My Update

- Decided to start over my ML process because the one I started wasn't working
- Spent some time re-truncating and labelling the data using Akhil's algorithm, captured entire tests in a consistent time interval of 7 seconds so instead of having snippets of data I was wanting to see how well it can differentiate entire tests, because I think Akhil and Albert are only doing short time intervals
- Setting up the correct format of data that I wanted took a while so I didn't get around to feature engineering, but I have a script using sci kit learn that functions, it isn't very good it's about 50% accurate but it's using only the raw data so all things considered with 0 feature engineering it's 25% more accurate than random guessing (have 4 categories of tests)
- This is a learning process for me so I'm glad that I finally have a model that actually gives results even though they're not great so far

Plan for next week

- Have ideas for feature engineering that I don't think have been done yet so I'm going to work on that next week and I'm still interested in trying to use Tensorflow which is more neural networks
- By next week my goal is to have a script that incorporates full feature engineering on the entire test
- Once I have a viable script with my 7 second tests I'm going to try to split up the data into much smaller time intervals

Meeting Notes

- Bayesian map between the features -- correlations between features → if certain features have the same information
- Data compression -- averaging microseconds over 10 data points or something

- Another idea: figure out max of all data (all force sensors), figure out how many force sensors get above 75% of that value
- PCA - helpful tool
- TSNE - dimensional reduction tool
- SVM
- random forest
- cross-validation
- boot-strapping

Berkeley Emergent Space Tensegrities

TENSEGRITY HRI MEETING

4.3.20 / 1:00-2:00PM / ZOOM https://berkeley.zoom.us/j/2417093709

My Update

- Learned Anaconda, Jupyter, went over some python basics, a bunch of numpy tutorials
- Brainstormed features, thinking about trying a conventional script that analyzes features like max force, oscillation frequency, maybe variance and stuff like that and have about 60 lines of code for that
- Also had the idea to do image processing on the .png files themselves because if we should be able to tell that a drop test is different from a squeeze test then a computer should be able to also
    - Andrew: don't do that, literally just use the raw data :(
- Going really slowly because each .png has 2.9 million pixels that each have 3 RGB values (I could also do black and white but that's still a lot)
- I'm interesting in trying to follow this through but if it seems like it's not worth it then I'll try something else
- I'm using Tensorflow so far which seems ok but I'm also probably going to try to learn sci kit learn

Plan for next week

- Same thing, just progress on stuff again
- Different data processing, featurization, model development directions

Berkeley Emergent Space Tensegrities

DOCUMENTATION OF PROGRESS

4.1.20

Progress

- So much coding! Mostly data organization and feature engineering for two different ML scripts
- Data processing is taking a while

Next

- Things from previous days, finish putting data into appropriate format for ML model, hopefully complete ML models

Berkeley Emergent Space Tensegrities

DOCUMENTATION OF PROGRESS

3.29.20

Progress

- A lot of Tensorflow Udacity course, learning the framework for writing a machine learning script
- Idea for our ML: image processing with the image of our data?
- Did some of the human factors training

Next

- Same as from 3.28

Berkeley Emergent Space Tensegrities

DOCUMENTATION OF PROGRESS

3.28.20

Progress

- Watched Python Anaconda tutorial, went over how to use Anaconda, Jupyter notebooks, python basics, a little bit of data visualization
    - Jupyter Notebook Tutorial: https://www.youtube.com/watch?v=fiQTb7-rCPo
- Watched Python for Data Analysis -- Python Pandas Tutorial
    - https://www.youtube.com/watch?v=B42n3Pc-N2A
    - Learned basic commands in pandas for data manipulation
    - Learned how to plot .csv files in python

Next

- Saved videos (more Edureka python basics stuff)
    - Full course for Python beginners: https://www.youtube.com/watch?v=rfscVS0vtbw
    - Python OOP tutorial: https://www.youtube.com/watch?v=ZDa-Z5JzLYM
    - Intro to Python: https://www.youtube.com/watch?v=uYjRzbP5aZs
    - Python full course (12 hours): https://www.youtube.com/watch?v=WGJJIrtnfpk
    - Python numpy tutorial: https://www.youtube.com/watch?v=8JfDAm9y_7s
    - Another python numpy tutorial: https://www.youtube.com/watch?v=GB9ByFAIAH4
- Convert intro to ML files to Jupyter, delete Colab notebooks
- Continue Tensorflow course
- Start looking at sci kit learn

- Feature engineering tutorials -- could this include basic stuff like mean, max, min, data of each force sensor
- Data 100 textbook
- Features: max, mean, oscillation frequency, number of occurrences of higher data points, e.g. like take ¾ max, count what percentage of data is greater than that, maybe number of data points, etc. (more!!) variance/standard deviation

Berkeley Emergent Space Tensegrities

TENSEGRITY HRI MEETING

3.27.20 / 1:00-2:00PM / ZOOM https://berkeley.zoom.us/j/2417093709

My Update

- Interested in continuing to work on node CADs and sending them to whoever has 3D printing capabilities
- Also want to start thinking about redesigning the payload
- Learn Jupyter notebooks
- Data processing -- sci kit learn, pandas
- paper!

Meeting Notes

- Resources from Andrew to expand skill set in data processing
- IEEE conference paper. Which conference?
- Data science resources:
    - Andrew's document for data science and machine learning resources: https://docs.google.com/document/d/19A4sOuFYcOr0Egi6gq1y-QX2AnPVL2OgtmjeMiP9Bjo/edit?ts=5e7e5a3e
    - https://www.datacamp.com/
    - Data 100 textbook: https://www.textbook.ds100.org/intro
- Anyone can work on data processing side of project, try out different models and evaluate accuracy
- Learn python, pandas (python package to manipulate tables easily)
- Python script on github
- Data 100 textbook
- Sci kit learn
- Logistic regression -- probabilities from 0 to 1

- Get scripts Andrew gives us and try to understand

Plan for this week

- Spend time on task we're prioritizing most -- report on progress. e.g. train first model, make first pipeline (?)
- Explore resources given

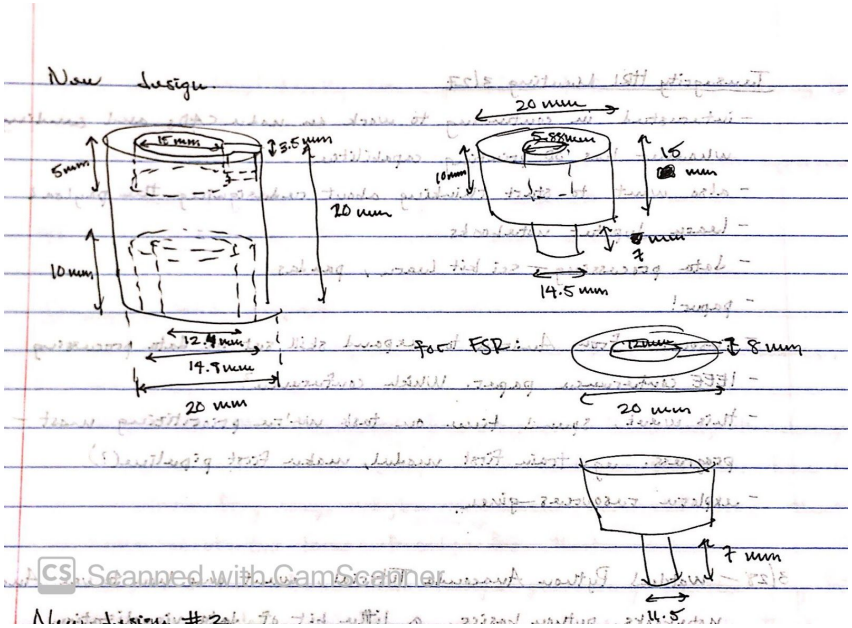
Berkeley Emergent Space Tensegrities

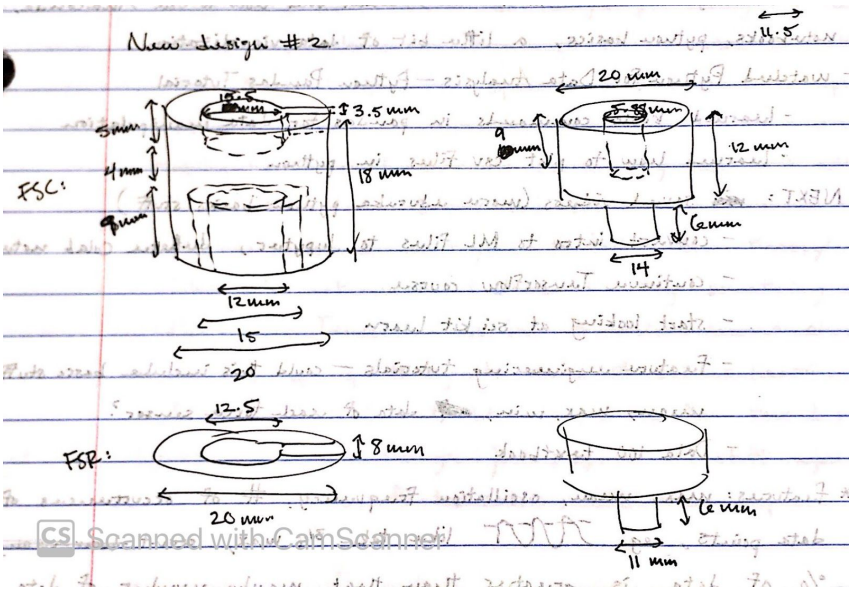
NODE DESIGN SKETCHES & CADS

3.27.20

Designs

- File path for STL files: /Users/salatiemann/Documents/UC Berkeley/Year 3 Sem 2 (Spring 2020)/node redesign2





Berkeley Emergent Space Tensegrities

TENSEGRITY HRI MEETING

3.13.20 / 1:00-2:00PM / 230 Hesse

Meeting Notes (Transcript from Bi-Weekly Tensegrity HRI Meeting Notes)

- Andrew's deliverables:
    - Final sensor selection (it seems like FSR is the way to go)
    - Testing of FSR circuit (look into FSR integration doc that Andrew sent)
        - https://www.sparkfun.com/datasheets/Sensors/Pressure/fsrguide.pdf
    - Testing of several (2 or 3) different hardware configurations for sensor integration

Berkeley Emergent Space Tensegrities

DESIGN DISCUSSION

3.9.20 / 1:00-2:00PM / 230 Hesse

Meeting Notes (Transcript from Bi-Weekly Tensegrity HRI Meeting Notes)

- Pick sensor and node design we want and actually build a whole robot
- Friday -- update Andrew about what we found/how we want to proceed
- Payload redesign? Get CAD designs from original
    - Bare minimum for testing

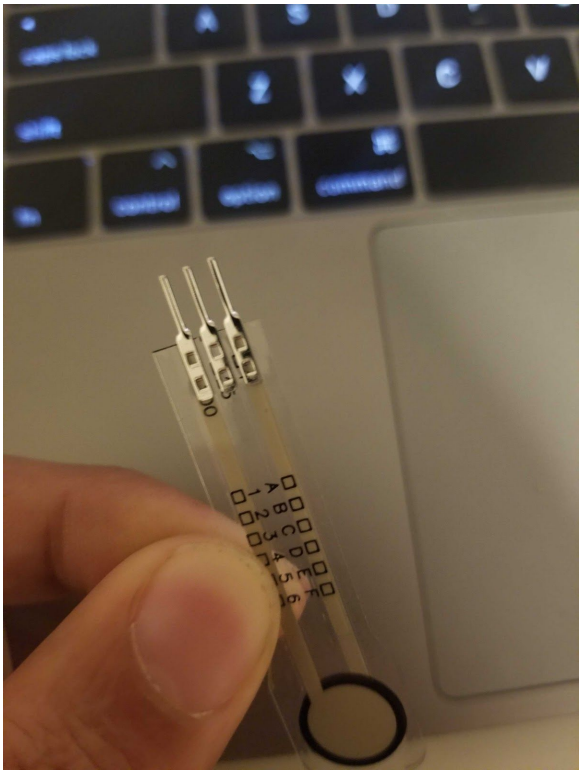
Berkeley Emergent Space Tensegrities

TESTING MEETING

3.8.20 / 1:00-2:00PM / 230 Hesse

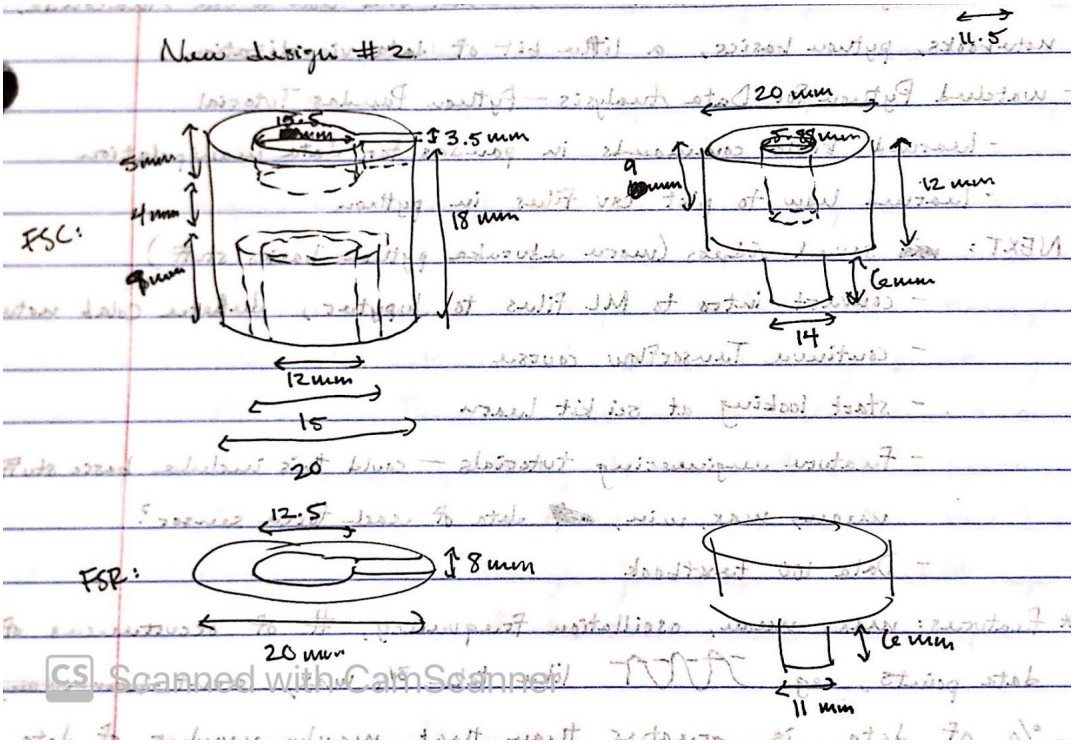
Meeting Notes (Transcript from Bi-Weekly Tensegrity HRI Meeting Notes)

- Force sensing resistor testing meeting
- Tried testing the FSR with the op amp circuit but it didn't work as we didn't have the right op amp
- Got values from the FSR which made sense using a voltage divider
- Need to verify the op amp circuit and calibrate it ourselves



Node Redesign Mockup

New design #2



Berkeley Emergent Space Tensegrities

TESTING MEETING

3.4.20 / 1:00-2:00PM / 230 Hesse

Meeting Notes (Transcript from Bi-Weekly Tensegrity HRI Meeting Notes)

- Capacitive force testing meeting
- Need plot for 0-100 pounds from SingleTact document to fit uncalibrated data
- When placing weights, first kilo had a larger effect on the readings than subsequent kilos
- Prove relationship exists if we can get this 0-100 lb plot
- Issues that we didn't foresee: sensor neck is too short before it needs to connect to the board so we would need to either extend the ribbon cable or make our own boards
- Is there enough space to fit the electronics from all 12 sensors in the payload?
- Plan for Friday at 1: run validation tests on FSRs because they should be delivered soon
- From this point, we think the FSRs will ultimately be simpler to implement

Berkeley Emergent Space Tensegrities

DESIGN MEETING

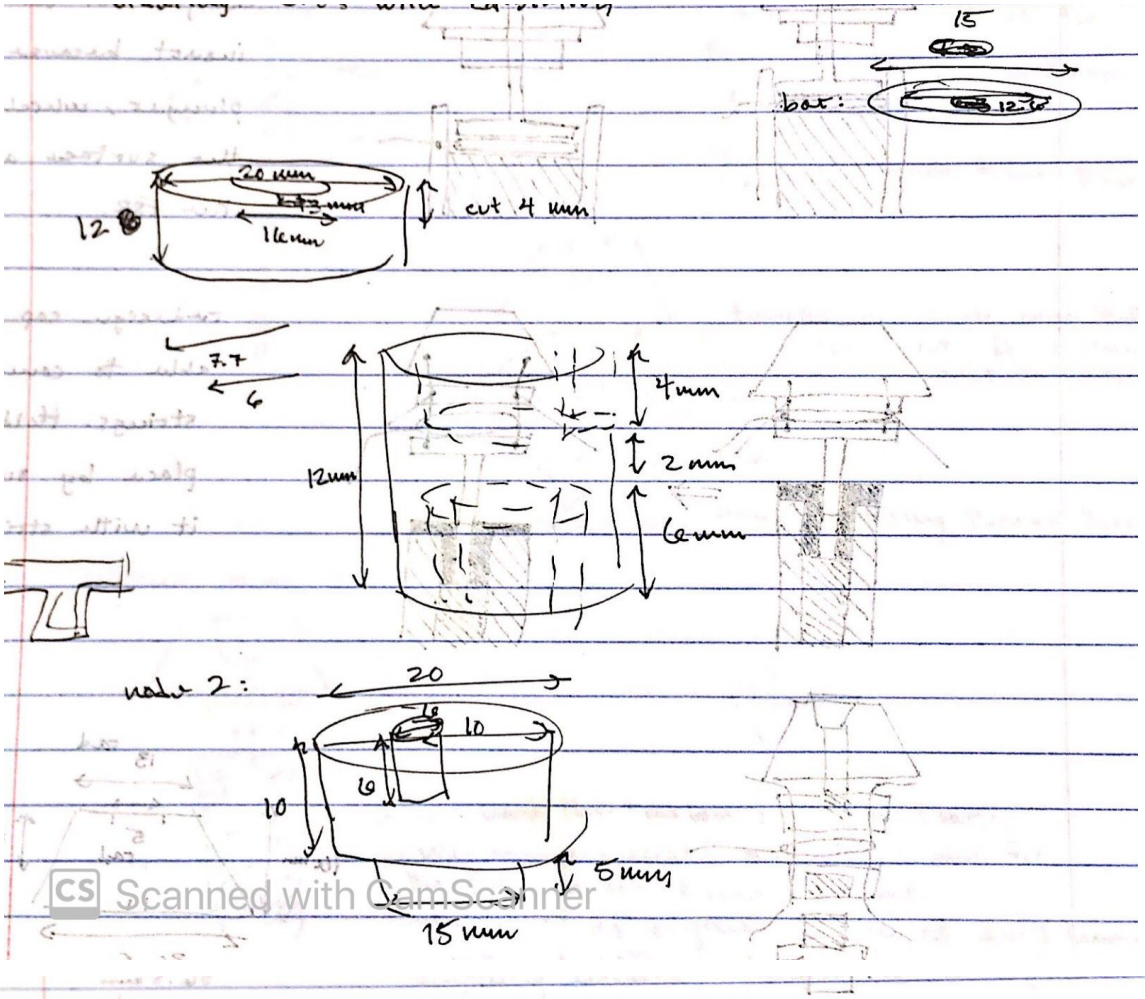
2.20.20 / 1:00-2:00PM / 230 Hesse

Meeting Notes (Transcript from Bi-Weekly Tensegrity HRI Meeting Notes)

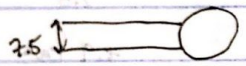
- Discussed calibrated capacitive sensor vs. non calibrated
- We want to buy one of each to see if we can calibrate the non calibrated one ourselves
    - Ordering CFSs with calibration
- Tasks:
    - test/interface with calibrated capacitive force sensor: Matthew/Yash
    - write/test out code to calibrate non calibrated capacitive force sensor: Matthew/Yash
    - write/test code to calibrate new FSR: Matthew/Yash
    - Some CAD designs
        - CAD with common parts: Akhil
        - “Implementation 1.5”: hybrid washer sandwich and plunger: Albert
        - Screw in end cap going into plunger/washer sandwich: Sala
        - Varun’s boltless design with washer sandwich: Varun
        - Anything else we think of
    - 3D print CAD designs for next Friday

Node Redesign Mockup

- Filepath for CAD files: /Users/salatiemann/Documents/UC Berkeley/Year 3 Sem 2 (Spring 2020)/node redesign



NEW FSR



If want 3 mm hole, dimension hole to 3 mm & tap by hand.

FSC

(make 3.5 or more).

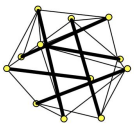
hole for screw: 5.85 - 5.88 mm diameter

Berkeley Emergent Space Tensegrities

FSR ARRAY DESIGN REVIEW MEETING

2.14.20 / 1:00-2:00PM / 230 Hesse

Presentation



Tensegrity Prototype V2 Force Sensors & Implementation

Matthew Kanter
Sala Tiemann
Akhil Padmanabha
Albert Lee
Varun Save

February <3 14th, 2020

Berkeley
UNIVERSITY OF CALIFORNIA

Presentation Outline

- **Ruled Out Sensors**
  Can't be Implemented
- **Recommended Sensors**
  Can be Implemented
- **Implementation Methodology 1**
  Washer Sandwich
- **Implementation Methodology 2**
  Bar and Plunger
- **Implementation Methodology 3**
  Boltless

Ruled Out

- **Load Cells**
  Too Costly (~$500/sensor)
  Too Heavy (~0.2 lbs/sensor)
  Large Accompanying Electronics
- **Optical Sensors**
  Typically used for Robotic Manipulation Tasks
  Very Costly (Upwards of $3K)
  Large Accompanying Electronics

Images Retrieved From:
https://www.futek.com/store/Load%20Cells/LTH/LTH300
https://www.transducertechniques.com/tha-load-cell.aspx
https://trends.directindustry.com/project-15720.html
http://www.quadratec-ltd.co.uk/optoforce-force-sensing-systems.asp

Capacitive Force Sensors

Pros
- less error than current FSR's
- higher force range (no springs needed)

Cons
- more complicated electronics needed
- $70 for calibrated sensor with electronics

Specifications
- Force Range: 0 to 450 N
- Size: 15 mm diameter
- Repeatability Error: <1%

Link: https://www.singletact.com/micro-force-sensor/standard-sensors/15mm-450newton/

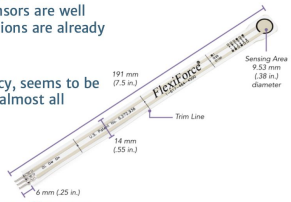
Better FSRs

Pros:

- Small size allows for easy implementation
- Current model uses FSRs, so these sensors are well understood and possible implementations are already in place

Cons:

- In terms of sensing power and accuracy, seems to be worse than capacitive force sensor in almost all regards

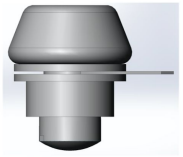


Specs for FlexiForce A201 Sensor:

- **Force Range**: 0-111N
- **Repeatability**: <±2.5% error
- **Dimensions**: 14 mm width, 9.53 mm diameter, trimmable length (maximum length: 191 mm)
- **Cost**: $73.30 for 4 pack, 131.80 for 8 pack

Sensor Implementation Methodology 1 : Washer Sandwich

- **Advantages**
  - Eliminate need for force divider
  - FSR can be larger than diameter of bar (no size constraint)
  - Hold FSR in place with surrounding strings
  - Can sense all forces, i.e. those from node compression and spring tension
  - No movement, no friction
- **Disadvantages**
  - Completely new design

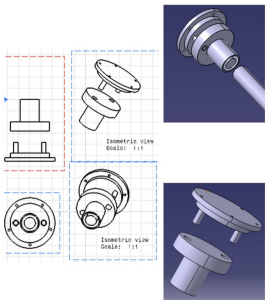


Sensor Implementation Methodology 2 : Bar & Plunger

- **Advantages**
  - Eliminate need for force divider
  - Plunger covers top surface area of FSR
  - No major reconfigurations from original design – only redesign screw element
  - No movement, no friction
- **Disadvantages**
  - Need to fit FSR inside bar (size constraint)
  - Modify bar to allow FSR wires through

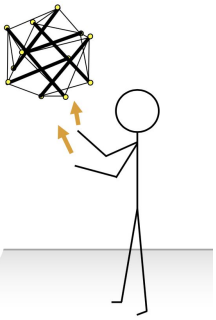
Implementation Methodology 3 : Boltless

- **Advantages**
  - Eliminates need for a force divider
  - Size of force sensor not constraint by rod diameter
  - No Bolt needed
  - No movement, no friction
- **Disadvantages**
  - Fundamental design changes required



Conclusion

- **Sensor**
  - Buy samples of the better FSRs and the Capacitive Force Sensors
  - We recommend the Capacitive Force Sensors for 2nd iteration prototype
- **Implementation**
  - No recommendation yet
  - Depends on the sensor



Berkeley Emergent Space Tensegrities

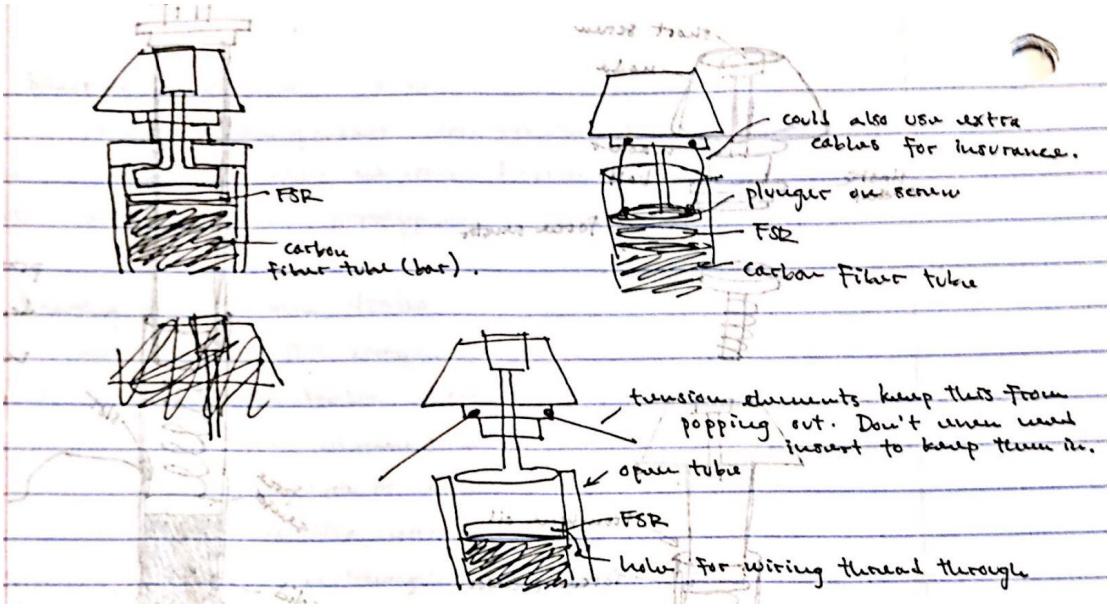
DESIGN MEETING

2.7.20 / 1:00-2:00PM / 230 Hesse

Meeting Notes (Transcript from Bi-Weekly Tensegrity HRI Meeting Notes)

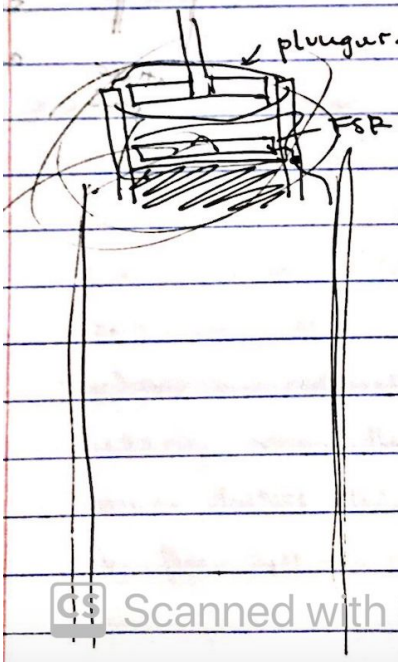
- Load cells are too expensive
- Akhil's force sensor comparison
- Redesign ideas for implementation:
    - Redesign end cap out of ninjaflex
    - Reimagine plunger system
    - Bigger washer "sandwich" by threading rope through both washers
    - Redesign end cap to house sensor and run ropes through end cap to ensure we still maintain rope tension
- Slides for Friday presentation: bad ideas, FSRs, capacitive force sensors, implementation ideas -- conclusion on best idea

Node Redesign Mockup

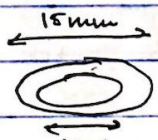


original node design:

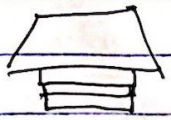
if we have this carbon fiber tube (bar), we can eliminate the insert and fit the force sensor inside the bar.
Same logic as original design ⇒ don't need anything holding plunger in except tension elements.

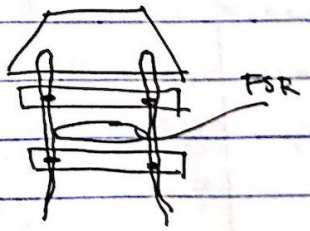


Only downside: might have more friction.

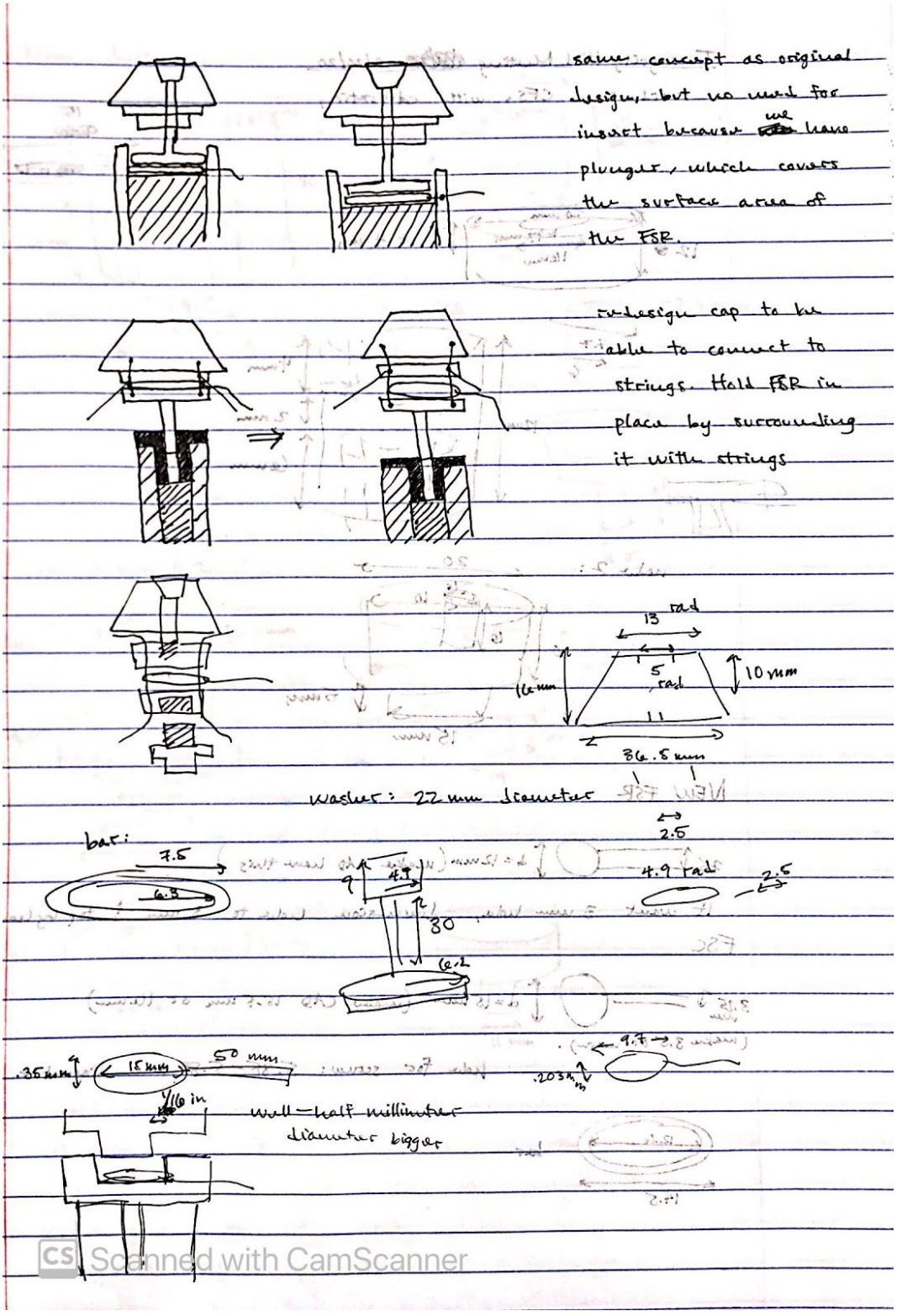


ideas: reprint end cap to eliminate need for screw.



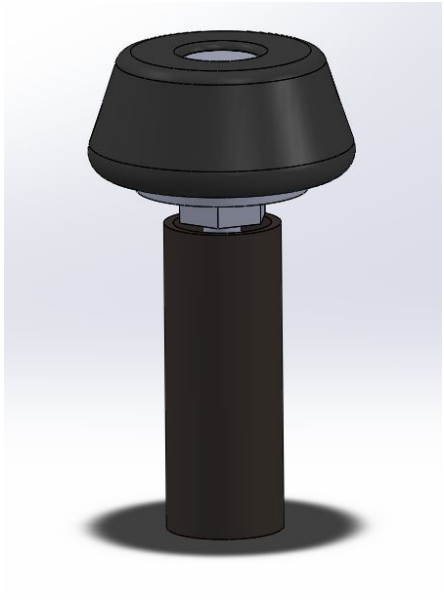


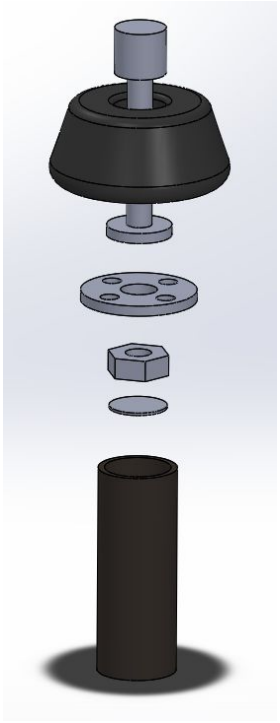
(exploded)

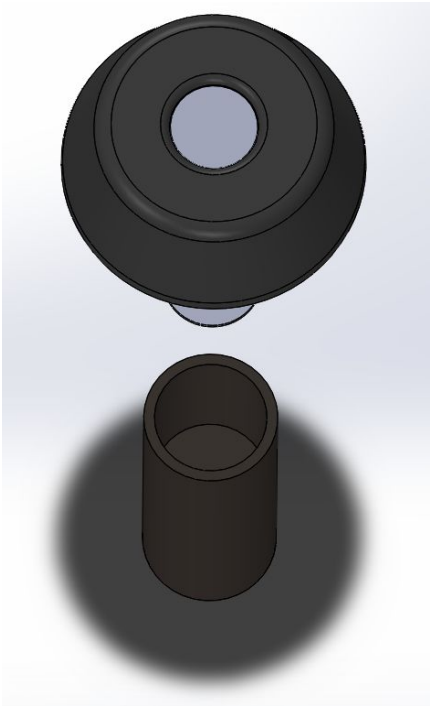


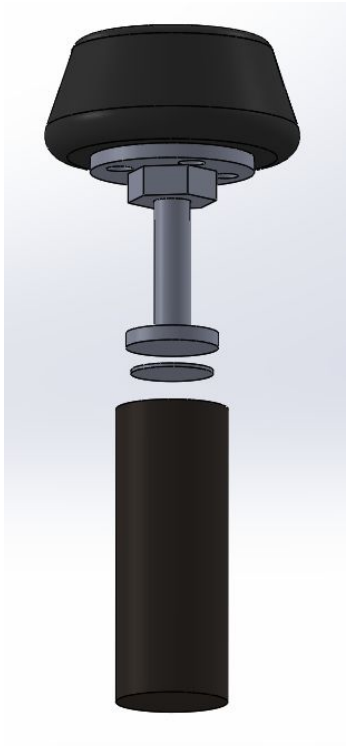
Node Redesign CADs

- Filepath for CAD files: /Users/salatiemann/Documents/UC Berkeley/Year 3 Sem 2 (Spring 2020)/end cap design









Berkeley Emergent Space Tensegrities

DESIGN MEETING

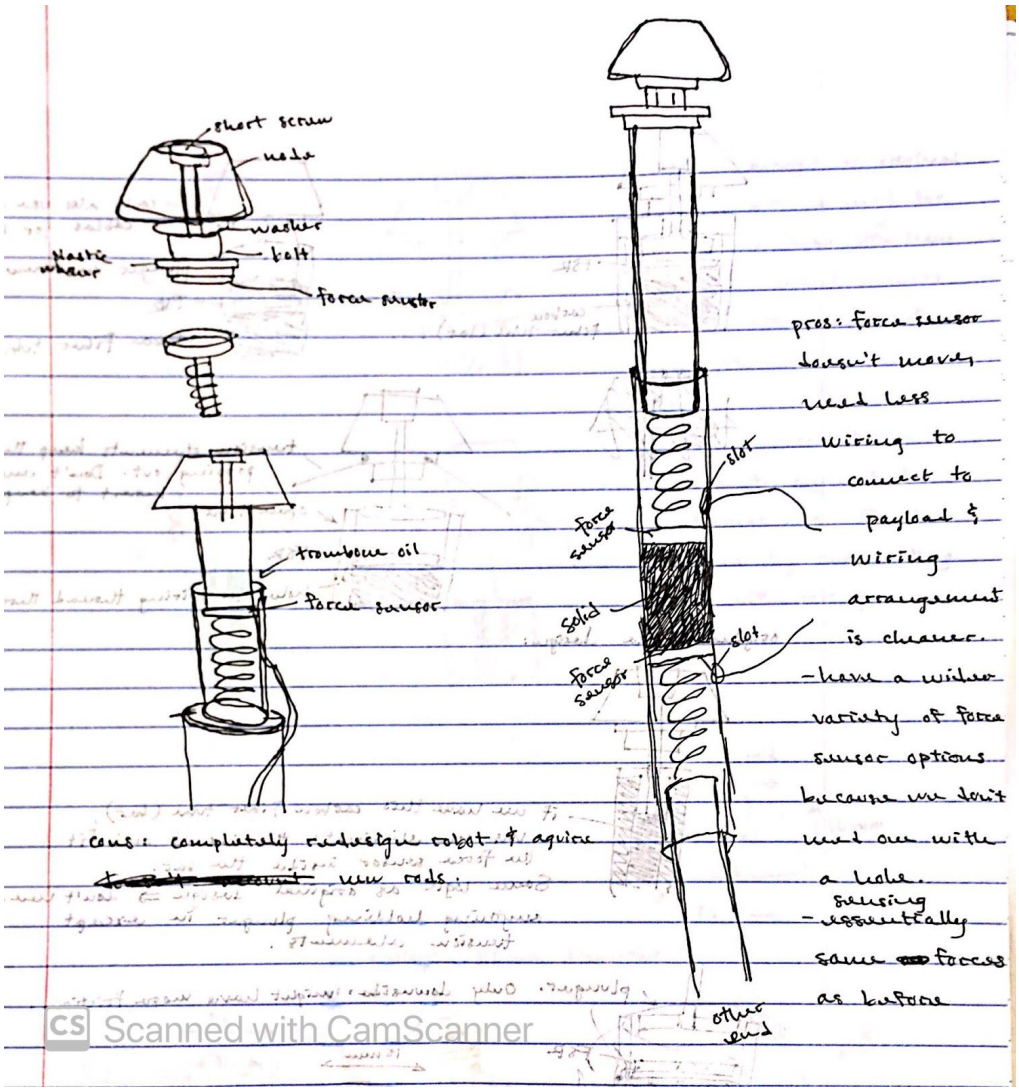
2.3.20 / 1:00-2:00PM / 230 Hesse

Meeting Notes (Transcript from Bi-Weekly Tensegrity HRI Meeting Notes)

- OptoForce sensors won't work because of large accompanying electronics
- Looking into better FSRs and capacitive force sensors
- Capacitive give absolute values but require small packaging
- Concrete ideas:
    - Capacitive force sensing
        - https://www.robotshop.com/en/capacitive-force-sensor-15-mm-45-n-10-lbs.html
    - Better FSRs (Albert)
        - https://www.tekscan.com/store/category/force-sensors-flexiforce
    - Load cells (Matt and Varun)
        - https://www.transducertechniques.com/tha-load-cell.aspx
        - https://www.flintec.com/media/downloads/mba-tw-miniature-datasheet-en.pdf
        - https://www.futek.com/store/Load%20Cells/LTH/LTH300
    - Current FSR
        - https://www.digikey.com/product-detail/en/interlink-electronics/34-00065/1027-1034-ND/7672219
    - Pressure-sensitive conductive sheet
        - https://www.adafruit.com/product/1361?gclid=Cj0KCQiAvc_xBRCYARIsAC5QT9mqIlQL9OnIdTFyc0Hf-zoJ6DOgCwAobY2htpIvyZw_SFjBmSRPpJYaAsVCEALw_wcB
    - Combinations of sensors, e.g. FSRs + vibration sensors
    - Akhil's force sensor comparison

- https://docs.google.com/spreadsheets/d/17eA4m2vv65IkrAYLVaAEp8R34-JoTdPCml7lj6MaIwQ/edit#gid=0

Node Redesign Mockup



Berkeley Emergent Space Tensegrities

TENSEGRITY HRI MEETING

1.28.20 / 12:00-1:00PM / 230 Hesse

Meeting Notes

- Deliverable for ideas for improvement on FSR array
- Andrew will be here Friday afternoons on biweekly basis
- 5 solutions each

Berkeley Emergent Space Tensegrities

TENSEGRITY HRI MEETING

1.23.20 / 1:00-2:00PM / 230 Hesse

Meeting Notes

- Next iteration FSR array:
    - Issues with friction (screws)
    - Make node elements more compact (less interference)
    - Critical analysis of different types of force sensing
        - Weight, size restrictions, budget
    - Options for switching/or why we should/not change it up
- Ideas for moving forward/changing force sensor array
- Ideas for how to use gyroscope data in conjunction with FSR data to get information that we couldn't get from gyroscopes alone
- Can classify which nodes it falls on with FSR data

Berkeley Emergent Space Tensegrities

WEEKLY UPDATE

1.13.20

My Update

- Have been looking at data on repository and using Matthew's truncating script to isolate the action events
- Didn't see anything uploaded on the drive, so started from the beginning and worked through all the data from the 10/18 and 10/21 folders
    - Drop test data seems to have only one action event per file but squeeze data has 12 -- was that on purpose?
    - Truncated data once to include all action events and then individually separated each squeeze set of data
- Uploaded the .csv files of the truncated data and the corresponding FSR plot
- Had the idea to work on not only classifying the type of interaction with the tensegrity but also the primary nodes involved in the interaction, given that the data given by each node is very distinct depending on the orientation
- Was able to connect to the internet using an ethernet connection on my Linux partition
- Followed setup instructions on the Spherical Tensegrity readme, but someone should probably check my work (not sure if I was successful in setting up Teensyduino)

Berkeley Emergent Space Tensegrities

TENSEGRITY HRI MEETING

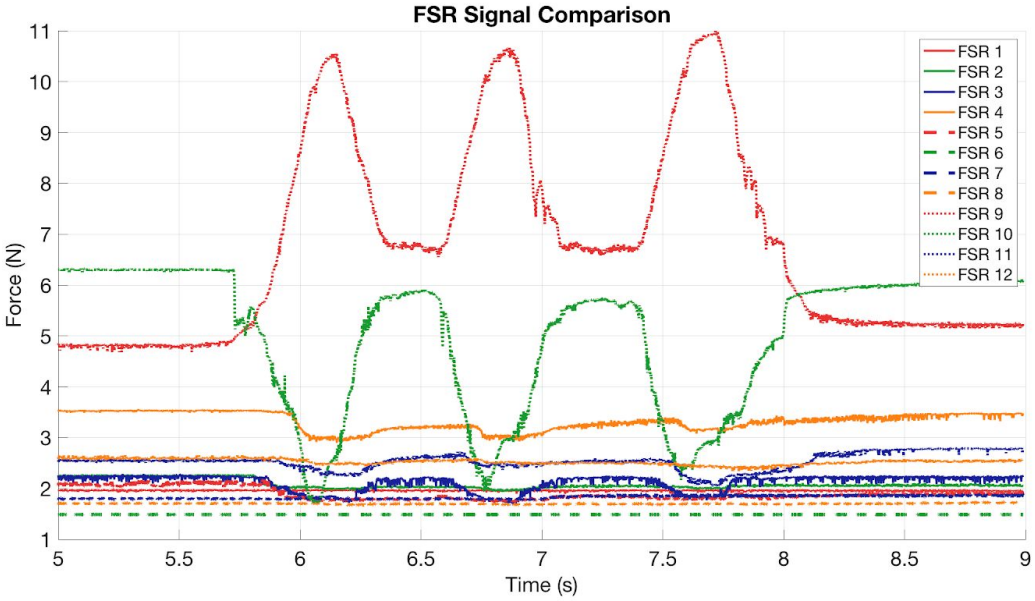
1.7.20

Meeting Notes

- pandas python package (data processing)
- Conference paper submission March 1st
- 2nd iteration prototype

Current Work

- MATLAB script to trim datasets
    - Filepath: /Users/salatiemann/Documents/UC Berkeley/Year 3 Sem 2 \(Spring 2020)/BEST/fsr_comparison_visual_with_csvexport.m
    - 1 second of data before and after the action event begins and ends
    - Adjust name of file, adjust range, export data to .csv



Berkeley Emergent Space Tensegrities

WEEKLY UPDATE

1.3.20

My Update

- Looked at Spherical Tensegrity repo, but unable to set it up because Linux partition is having trouble with WiFi again
    - Hope that it’s just because my home WiFi is flaky
- Have been looking at Linux tutorials to help familiarize myself
- Reading up more on machine learning looking at scikit learn tutorials
    - https://scikit-learn.org/stable/tutorial/index.html
    - Working less in data analysis because it seems like Akhil and Matthew are amending the data so far, but will get started with that soon!

Berkeley Emergent Space Tensegrities

WEEKLY UPDATE

12.24.19

My Update

- Started looking at Python sci kit learn documentation to get acquainted with machine learning
    - https://scikit-learn.org/stable/
- Watched a 3Blue1Brown series on neural networks
    - https://www.youtube.com/watch?v=aircAruvnKk&t=3s
- Planning on getting more familiar with Linux now that it's all set up and looking at the data on the repository

Andrew's Assignments

- Look at setup instructions for the Spherical Tensegrity repo -- want to finish getting set up with software stack

Berkeley Emergent Space Tensegrities

TENSEGRITY HRI MEETING

12.5.19 / 3:00-4:00PM / 230 Hesse

Meeting Notes

- Get familiar with Linux, make sure everything is stable
- Basic commands
- Expect email by Monday every week regarding progress
- Operating exclusively in Jupyter notebooks
- Ubuntu 18.04
- Anaconda to install Python 2.7

Berkeley Emergent Space Tensegrities

TENSEGRITY HRI MEETING

11.18.19 / 3:00-4:00PM / 230 Hesse

Meeting Notes

- Lots of data from experiments! Large datasets
- Want to distinguish between interactions on trained data
- Next semester: heavily involved on current system and make 2nd prototype
- Idea! With data, can create simulation of what robot looked like during interaction
- Machine learning with Python/Jupyter -- sci py
- Come up with solutions to problems that are currently coming up with current prototype -- wires popping out
- Serious design project to iterate on prototype
- Idea that I had before -- push node 1 3 times to interact
- 6 axis IMU gyroscope -- translational and angular acceleration
- Noise combing
- Model icosahedron in MATLAB -- Simulink??
- learning/having Linux? By end of Thanksgiving break?
    - Version, packages
- Python for data processing. Sci-kit-learn (Python ML packages)
    - “Super easy it’s like... make model... dot... do”
- Train model on 200 demonstrations
- Debugging code with Arduino C++
- Bring Arduino home!
- Take data from force sensors and gyroscope, simulate entire motion of robot → real time

Berkeley Emergent Space Tensegrities

ANDREW'S QUALIFYING EXAM PRACTICE

11.6.19 / 4:00-5:00PM / 230 Hesse

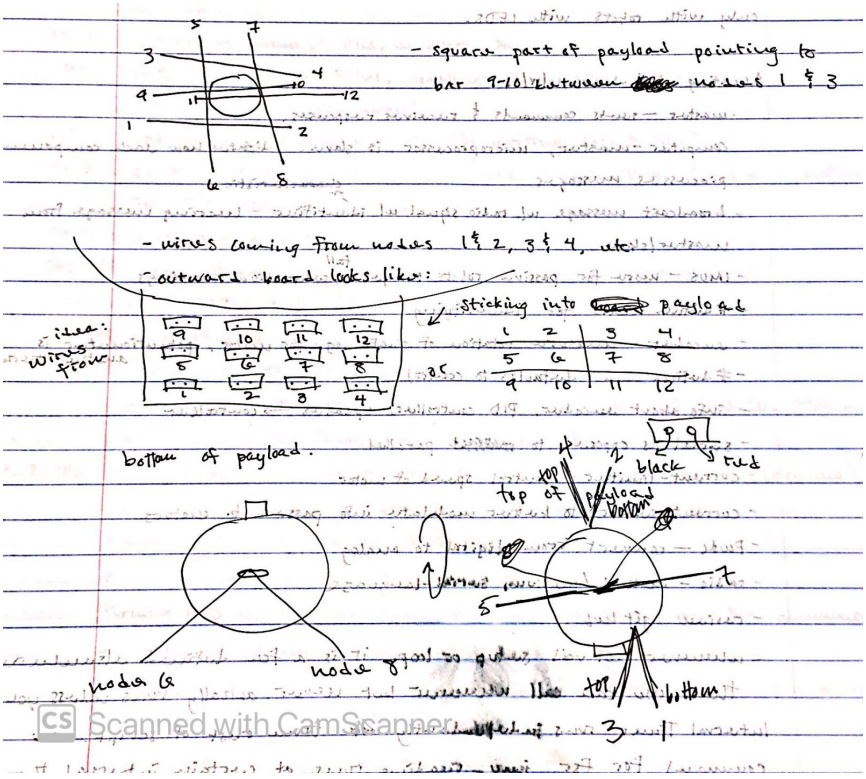
Presentation Notes

- Challenges: promoting context and modeling soft robotics
- Compliant -- deformability, as opposed to soft components (soft robots)
- Tensegrity as a platform for HRI -- compliant, configurable, robust, mobile. Don't yet have good HRI
- Goal 1: design force-sensing tensegrity platform. Reliably detect variety of physical interactions
- Goal 2: interaction model in which we can distinguish among different interactions (types of physical interactions) using a force-sensing tensegrity
- Two parallel spring elements to extend range of force sensors
- Use demonstrations and classify them -- train classification model
- Use raw FSR readings to find tangible features
- Drop tests very distinguishable from squeeze tests, in terms of max force, oscillation frequency, and total impulse
- Gather large dataset spanning a variety of interactions for training
- Task-oriented HRI experiments with uninformed users to assess intuitiveness
- How much help do the robots offer in a disaster scenario, based on efficiency analysis?

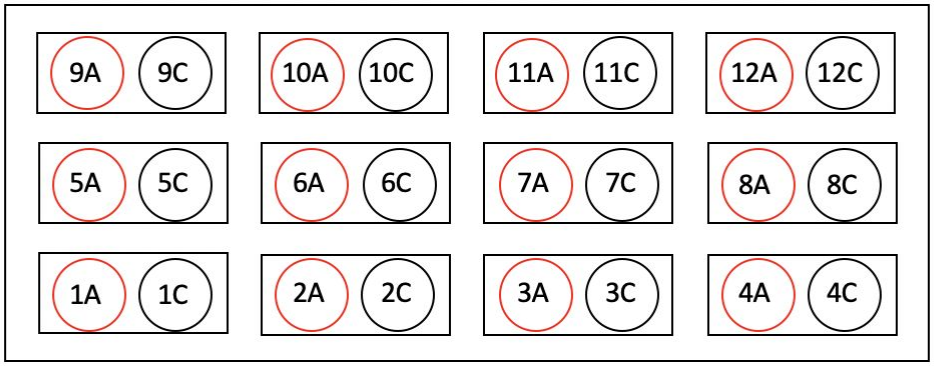
Berkeley Emergent Space Tensegrities

DEVELOPING CONVENTION FOR ROBOT

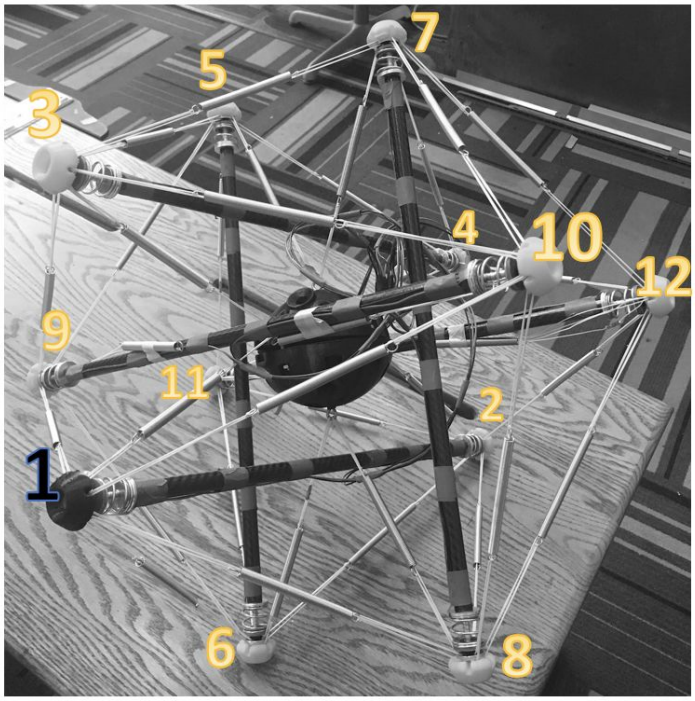
10.28.19



Labeling denotes anode and cathode coming from the indicated node. E.g. 1A indicates the anode (red wire) from node 1



Node numbering convention:

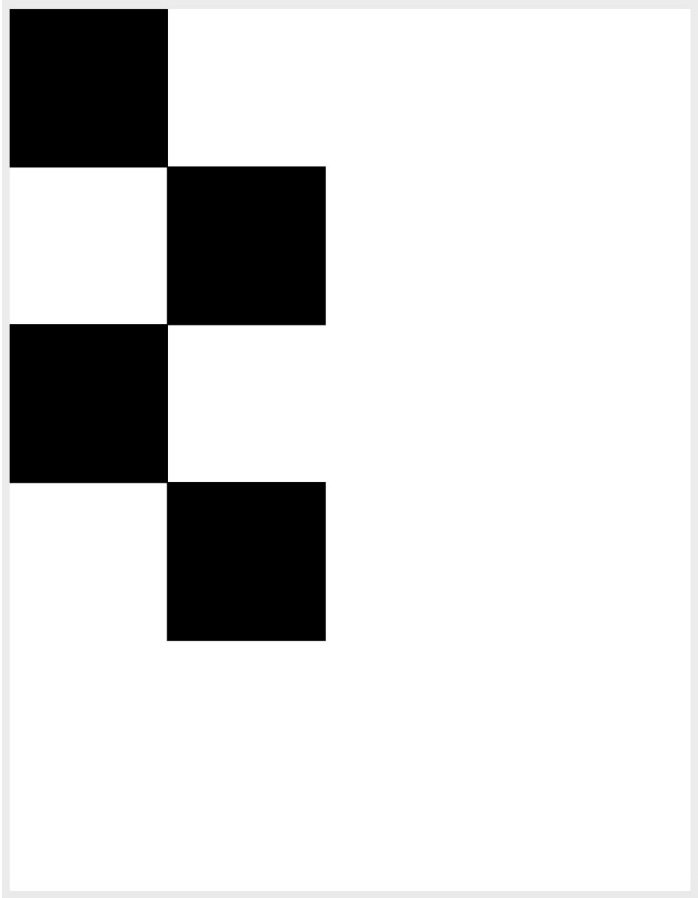


Berkeley Emergent Space Tensegrities

EXPERIMENTAL

10.26.19 / 3:00-5:00PM / 230 Hesse

Scale for Videos:



Test Node Configuration Document:

- Closed-face
    1. 2-6-11
    2. 2-8-12
    3. 1-8-10
    4. 3-7-10
- Open-face
    5. 2-6-8
    6. 10-8-12
    7. 1-3-10
    8. 3-5-7
- Rod
    9. 2-4
    10. 6-8
    11. 10-12
- Squeeze
    12. 2-6-11
    13. 2-6-8
    14. 6-8
    15. 6

Berkeley Emergent Space Tensegrities

ONE-ON-ONE WITH ANDREW

10.14.19 / 2:00-3:00PM / 230 Hesse

Meeting Notes

- Master -- sends commands and receives responses
- Computer - master, microprocessor is slave. Dictates how each component processes messages
- Broadcast message with radio signal with identifier (characteristic) -- hearing message from master/slave
- IMUs -- more for passive robots to get fall acceleration readings
- #define macro for identifying
- Encoder -- measure rotation of shaft, e.g. for motor, potentiometer is another term
- # define __ defaults to control
- Info about encoder, PID controller, updates → controller
- Serial as opposed to parallel
- Current-limiter -- control speed of motor
- Current sensors to better modulate info passed to motors
- PWM -- convert from digital to analog
- Radio -- method/medium, serial-language
- Review github
- Whenever not void setup or loop, it is a function defined elsewhere that you can call whenever but never actually runs unless you call it
- Interval timer runs independently from rest of script, command for fsr, imu_reading runs at certain interval throughout running code

Berkeley Emergent Space Tensegrities

NOTES ON ARDUINO CODE

10.3.19

Notes

- Filepath for folder: /Users/salatiemann/Documents/UC Berkeley/Year 3 Sem 1 (Fall 2019)/BEST/FSR_Board
- CONSTANTS_H:
    - #ifndef checks whether the given token has been #defined earlier in the file or in an included file; if not, it includes the code between it and the closing #else or the #endif
    - Defines CONSTANTS_H as constants for Robot to use, default/initial values
    - Line 11 Are “slaves” other controllers that link to master controllers? Why are there so many slaves? Do they each do different specific things for the robot?
    - # define __ __ is search and replace
    - int __ __ is a variable in which the sketch can modify that variable under desired circumstances
    - const int is read-only → can’t be changed in script
    - PID -- proportional-integral-derivative controller. Closed loop mechanism employing feedback widely used in industrial control systems and other applications
    - 17 What is IMU_QUAT? Why microseconds? ENCODER?
    - 27-29 feedback constants?
    - 24 What is LED for?
- Board.h
    - 4 important constants/definitions from “Constants.h” file
    - PCB -- printed circuit board
    - Sometimes with #define, have identifier, replacement. Sometimes only have identifier, sometimes have both. What is the difference,

and how come you use #define without a replacement? Also e.g. wrote #define ROBOT FSR_BOARD before #define FSR_BOARD, 6. How does that work? What kind of object is the replacement?

- 14 is this calling the whole FSR_BOARD.ino file? i.e. controller scheme
- 16-23 different robot options. Types of robots. Difference between TT5 and TT4 motors?
- 29 if robot type = TT4, import stuff from that file, define input/output pins for specific board
- 33 what do encoders do?
- 51-54 initializing arrays for PID control
    - Does “do use” imply that PID control is being defined in one of the imported files but it isn’t necessarily called until you do this?
- 56-58 initializing motor position limiting commands
- 62 DRV8833 variable defined in #include file, updating array of motor data
- 63 updating array of encoder data. What is this?
- 65-69 “&” is address, acts as pointer
    - This is the feedback. 65-68 updating controller
- 71 what does this line do (serial)
- If TT5 Robot, include these existing files
- 81-84 LEDs for TT5 robot *** is “LED” defined elsewhere? Why need to define again?
- 88 what is PHS? i.e. MOTOR1_RHS
- 86-104 Motor encoders -- what do they do? What do LEDs blink for?
    - Defining which pins correspond to which motors
- 109-112 initializing PID control
- 114 what is PWM?
    - Pulse width modulation -- use electricity to control analog devices via digital signal

    - 116-118 why do we need motor limits on this robot, as opposed to the other one? What physically does this do?
    - 121-128 feedback control
    - 131-136 updating controller with limiting values
    - 137-143 from <ACS712.h> file, write data from sensors
    - 147-151 or if robot using is TT5 with 6 motors, include these packages
    - 154-157 also uses LED array, defines which pins
    - IMU -- inertial measurement unit. Motion sensor basically. Reports a body's specific force, angular rate, sometimes orientation
    - 159-188 defining the pin locations of encoder, PWM, current motors for each motor (?)
    - 193-196 initializing arrays for PIDs, encoders, PWMs
    - 204 stuff from imported files, updating values written to PIDs
    - 207-213 not sure what REVERSE means?
    - 216-223 updating current limiters
    - 224-231 writing data to file?
    - 236-243 or if robot is the TT5 with 12 motors, include those files and define the LED array which is empty
    - 246-257 locations of motors
    - 265-278 array of motor information. Is this data or writing to motors?
    - 282 or if ROBOT is the FSR board, initialize LED array
    - 289-300 a lot of pins
    - 306-309 Wait... is this the data collection?
        - So the robots with motors are the mobile robots, correct? And the FSR one is new code for the new force sensor robot
- FSR_BOARD

    - Arduino code -- writing to board (.h file [C++] vs .ino file [Arduino]) including built-in packages, Encoder, other scripts and packages
    - #defines ROBOT_TYPE and FSR_BOARD from other files
    - No idea what Adafruit does
    - Turning LED on, not sure what serial commands do
        - used for communication between the Arduino board and a computer or other devices
        - serial.printIn establishes connection -- prints confirmation that things are working
    - No idea what Baud Rate/Xbee Channel are
    - Error messages if connection to IMU (BN0055) detected
    - Switching on and off LEDs but not sure why
    - Reading messages from the master board
    - Replying to Echo? Encoder reading?
        - If slave board receives new rates from master, sets update
    - Broadcast from master?
    - Repeats every millisecond
    - Parallel loop -- a second void? Sends readings from IMU sensors
    - Another parallel loop for fsr sensors. Sensors on both sides of each of 6 pins, and the readings are collected with simple analogRead(FSR1_PIN) command. Sends readings from all 12 sensors
- led.ino
    - Include Board.h not sure what LED script tells LEDs to do but has to do only with robots with LEDs

Berkeley Emergent Space Tensegrities

ONE-ON-ONE WITH ANDREW

10.2.19 / 2:00-3:00PM / 230 Hesse

Meeting Notes

- Longer rod is cathode
- Form for Machine Shop and Hesse access
- Look at Arduino code on current board, practice parsing code
- Sets up board and calibrates, be able to make adjustments in code and run own experiments

Berkeley Emergent Space Tensegrities

ARDUINO INFORMATION SEARCH

9.26.19

Notes

- Microcontroller platform program to output certain things when you input certain things
- Connect sensors, code inside, trigger events. Connect to LEDs, motor, etc.
- Open source -- anyone can manufacture or modify Arduino hardware
- Arduino widely used for prototyping -- eventually design device with components they need for specific device
- BOARD is: write code in IDE and upload onto board, where main power for system goes into, where sensors/modules/shields connect to
- Shields connect directly to arduino board to add functionality
- Autonomous things -- sensors in locations farther away from arduino board get data from board → modules/sensors using wires
- Proximity sensor for autonomous vehicles
- Modules -- put arduino board in 1 location, control motors or servos from different location: connect motors/servos to motor/servo module, put arduino board to module. Modules add functionality, use sensors (can connect to many devices)
- Breadboard allows you to create projects without soldering
- Servomotor -- can tell specific position to go to
- TUTORIALS:
    - EEEnthusiast
    - Eli the Computer Guy -- Arduino Introduction
    - Robert Paz: Introduction to Embedded Systems (EE260)
    - Jeremy Blum: Tutorial 01 for Arduino
    - Programming Electronics: Tutorial 01 Hardware Overview

Berkeley Emergent Space Tensegrities Laboratory

ONE-ON-ONE WITH ANDREW

9.24.19 / 1:00-2:00PM / 230 Hesse

Meeting Notes

- Get familiar with Arduinos
- Weekly meeting Wed @ 2PM
- Submit keycard access to 6th floor Etcheverry
- squishy/BEST meeting next week
- Machine shop training! And also experience machining
- Teensy 3.2

Berkeley Emergent Space Tensegrities Laboratory

ONE-ON-ONE WITH ALAN

9.10.19 / 3:00-4:00PM / 230 Hesse

Questions

- What does the team look like? Is it a lot of grads and undergrads working together or are you the main supervisor on the project?
- What are you specifically working on/most interested in?
- Do you interact/work directly with Dr. Agogino a lot? Do you have group meetings on a regular basis?
- If I am working on a specific aspect of the project such as manufacturing/controls, will I have the opportunity to learn about other aspects such as the software you are designing for the robots?
- You mentioned you have projects in mind that could fit with my skill set. What are those?
- What are long-term applications of these projects? What is the specific long-term application of the project/s I would be working on?
- What does mentorship look like in this lab?
- What is the balance between individual and collaborative work?

Tasks

- Read the article on using Squishy robots to explore Saturn's moon Titan
    - Easier land on/explore moon, deploy scientific equipment
- Deep learning of tensegrity locomotion