Group Goals

Two Phase Approach

Phase I

- Create a stable release of random lasso in python. (3-4 people) ~Phase 2
- Create a robust testing framework and generate simulated data. (1-2 people)
- Unsupervised learning (~ people)

Phase II

- Get results and analyze to present. (Everyone) ← Good enough to finish here...

Phase III

- Test novel permutations/flavors of random lasso. (3 people)
- Test random lasso on varying permutations of simulated data and on real data. Analyze results and determine use-cases. (1 person)
- Test stable release against existing literature and state-of-the-art methods of regression. (1 person)

Some Benefits of this Approach

- Most of the team will be writing the random lasso algorithm during Phase 1, this means it will be easy for most of the team to manipulate this algorithm during Phase 2.

- No one is waiting on the progress or results from anouther person. The entire Random Lasso function is created early-on.

Implementing Random Lasso in Python

Goal I

- Ridge Regression
- Lasso
- Elastic Net
- Adaptive Lasso (hard)
- Pro-to-Package: Hyperparameter estimation.

Implementing Random Lasso in Python

Goal II

- Sudo-normalize features.
- Sample random features, then run regression on just random features.
- Concat the coefficients from the random sampling. Return a sudo-average of coefficients for all features.
- Repeat the above, but do a weighted sampling of random features. Weights being the coefficients.

Part 1 & Part 2

- Recall that random lasso has two nearly identical parts.
- The upcoming sudo-code of random lasso only describes part 1, since part 2 is mostly redundant.
- Part 2 uses the the coefficients from part 1 to perform a weighted random sampling.

Sudo-code for Random Lasso

```
let's say x is samples=100 x features=1000
let's say y is #samples
y = sudo-normalize(y)
x = sudo-normalize(x)
matrix_coef = matrix(rows = #boostraps, cols = #features)
for (ii =1...#bootstraps) {
    sample_index = random_sample(1:#features, #samples)
    sample_coef = empty_vector(0 or NA, #features)
    sample_coef[sample_index] = lasso(x[:, sample_index], y)
    matrix_coef[ii, :] = sample_coef
}
final_coef = meanOfColumns(matrix_coef) 1 x 1000
```

RandomLasso(x, y, ...)

RandomLasso(x, y, alpha = c(0.5, 1), cores = 16, verbose = FALSE, nfold = 10)

- x Matrix of independent data.
- y Matrix or vector of dependent data.
- bootstraps Number of times features are randomly sampled.
- alpha Regression method, e.g. ridge=0, elastic=0.5, lasso=1.
- lambda_1se Largest value of lambda such that error is within 1 standard error of the minimum.
- box_width Number of features sampled when randomly sampling.
- box_width will be equal to the number of samples.
- nfold Number of folds tested to find the optimal hyperparameter.
- cores Number of cores used when running in parallel.
- verbose Supresses all printing and time estimation.
- verbose_output Returns additional information.

Alpha

- Can be any value from 0 to 1. This the norm jargon.
    - ridge → 0
    - 0.25
    - elastic-net → 0.5
    - lasso → 1

Concept | Stable Release

- Stable Release: Our best preforming stable random lasso.
- When a better flavor of random lasso is discovered, it will replace the current stable release. Everyone will use that new stable release here-after for any testing.

Simulated Data Generation

- Created tools that generating semi-randomized simulated data.
- During Phase 1, create robust testing framework and library for peers to use during Phase 2.

Parameters:

1. # of Features
2. # of Samples
3. # of Important Features

Real Data Generation

- One or two real data sets.

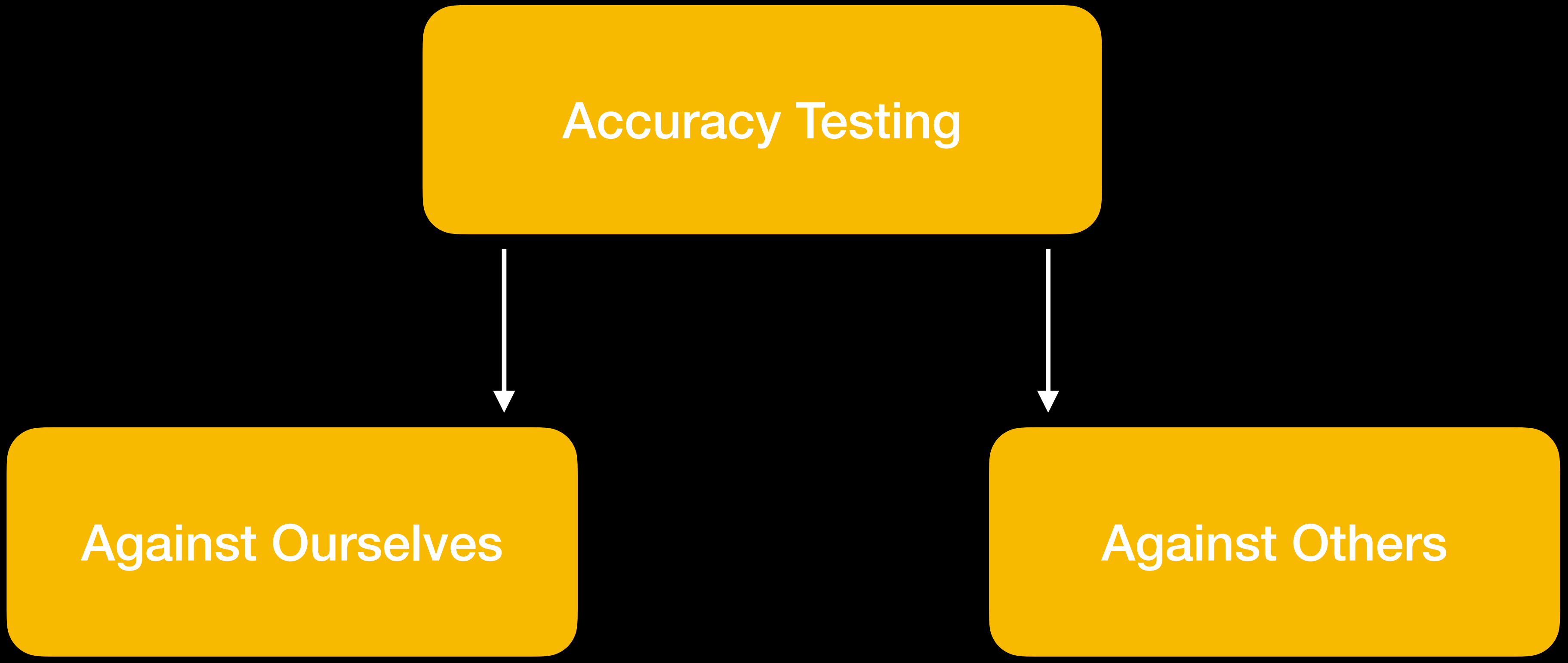
- One dataset should be a dataset that researcher commonly use to show the efficacy of regression algorithms.

- Another could be a newer dataset that solves a real world. This serves as an example for researchers.

    - e.g. bioinformatic dataset

Accuracy Testing

- Main: RMSE, F1
- Other Candidates: R2, other truth table methods, ...
- Simulated Data (~90%) & Real Data (~10%)



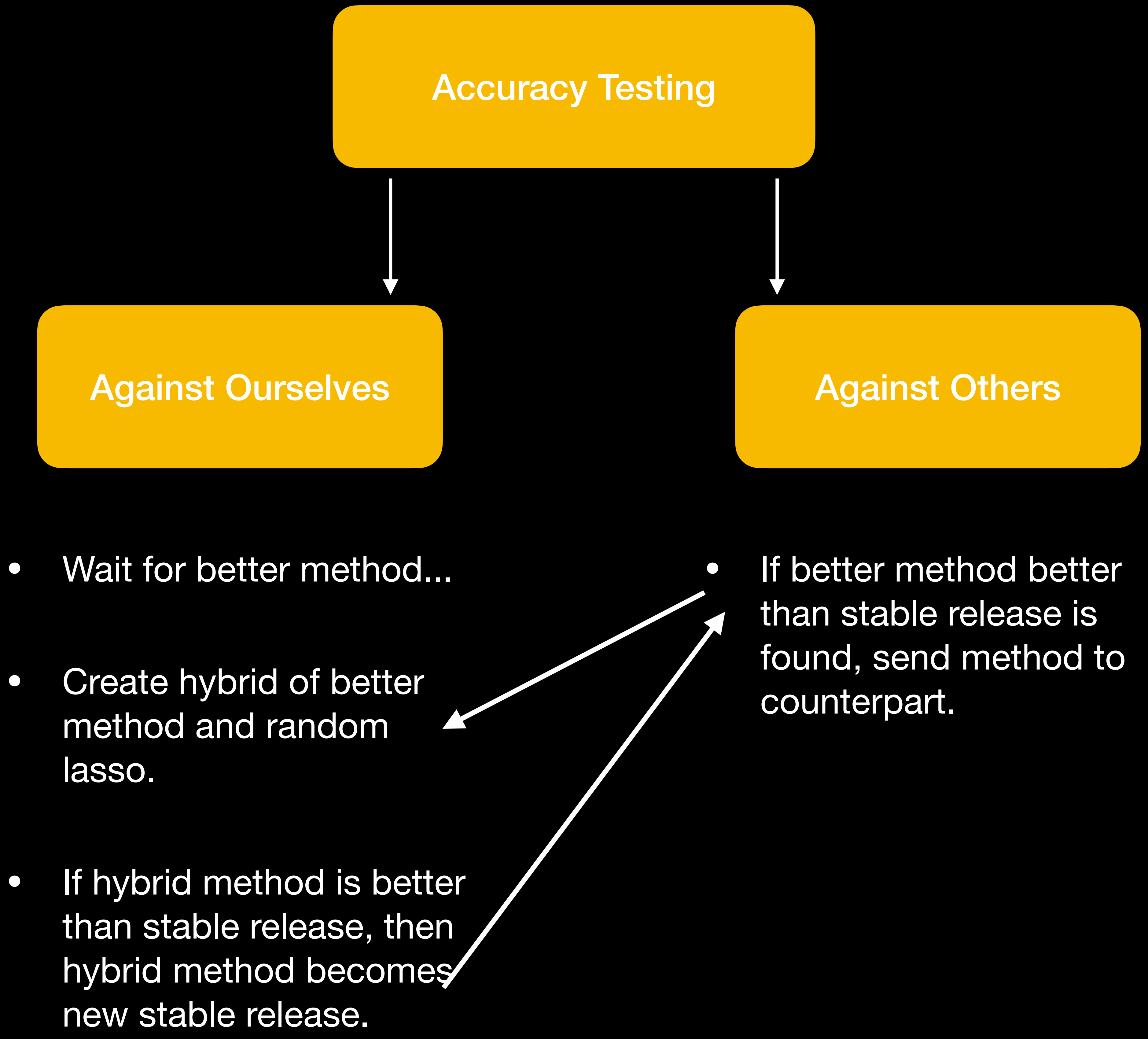
Accuracy Testing

Against Ourselves

Against Others

- Test accuracy of new flavors of Random Lasso
- R & D

- Test accuracy of stable release against other state-of-the-art methods.
- Literature Review



Accuracy Testing

Against Ourselves

Against Others

- Wait for better method...
- Create hybrid of better method and random lasso.
- If hybrid method is better than stable release, then hybrid method becomes new stable release.

- If better method better than stable release is found, send method to counterpart.

Against Ourselves

Test novel permutations/flavors of random lasso.

- Altering regression methods.
- Altering number of parts.
- Altering number of bootstraps.
- ~~Altering cutoff value.~~
- Altering ratio of features randomly sampled and samples.

- Logistic Regression
- Altering Normalization
- Altering number of folds for running hyper-parameters.
- Finding a global hyper-parameters.

What's Next

- Project Proposal:
  -

Due: June 15th
Ask me questions/call via WhatsApp. Matthew will send group these slides.
Ask questions on Piazza.
Pre-due: June ~12th Video: June ~12th (Discuss draft as group)
We need to propose a feasible amount of work. What we propose is set-in-stone.

Project Proposal (10%)

A project proposal should be just one page pdf (less than 500 words single spaced)

A project proposal should include:

- Introduction/Background
- Methods
- Potential results
- Discussion
- At least three references (preferably peer reviewed)
  → RandomLasso paper, Hi-Lasso paper, ...

A checkpoint to make sure you are working on a proper machine learning related project.

Your group needs to submit a presentation of your proposal. Please provide us a public link which includes a 3 minutes recorded video.

Great Discussion!!!