Progress Report

Pranav & Aaryan

11/02/2021

Summary of Ideas Discussed

Changing architecture - not much difference

Individual results are not accurate, these would help reduce the bounds of CFF distribution → More confident in our averages if they are more localized

- Ant Hill Optimization
- Averaging Weights
- Parallel Training

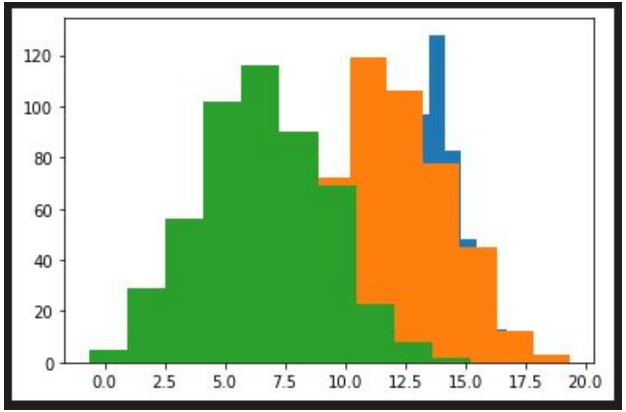
XGBoost

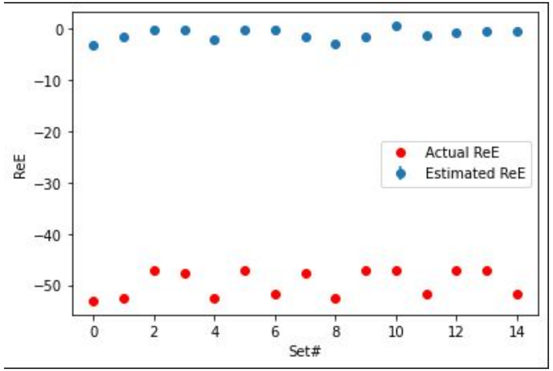
Observations on Architecture

- Modifications Attempted
  - Dropouts (should reduce overfitting)
  - Additional dense/other layers
  - Changed node amounts in each layer
  - Activation functions
- After utilizing combinations of these modifications, no significant differences in bounds or accuracy of CFF distributions were observed
- Possible limiting factor: training time; more combinations could be tried in more time, as it takes ~2 hours to run the local fits

Overall Observations

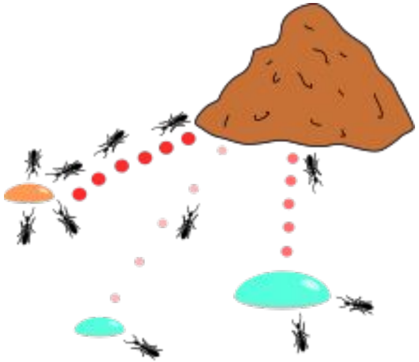
- Each individual replica does not create an accurate ‘guess’ of the CFFs but overall they can give a much more accurate representation of CFFs (Regression towards the mean)
- What we are thinking:
  - How can we have networks communicate with each other so they go towards a successful path?
  - How can we increase speed taken for regression towards the mean?
  - Can we improve the results of the individual replicas?
  - Can throwing out “bad” intermediate networks reduce training time?





Possible Ideas

- Intermediate weight averaging/combination
    - after a certain n training epochs or some frequency, average weights of a certain number of subsets of all the models and begin training again with a reduced number of models
- After half of the replicas are done we find the mean that they are tending towards → encourage new networks to follow path towards the means
    - Possible Methods: Pruning out networks not going towards that path, utilizing stigmergic optimization, track initial starting weights of ‘successful’ networks and use them to create new networks



XGBoost

- Another commonly used machine learning algorithm in industry
- Implements “Gradient Boosting Tree” algorithm
- Combines many “weak learners,” or small decision trees
  - Similar to random forests; difference is that combination of weak learners is done differently
- Utilizes gradient descent to minimize loss while combining learners
- Extremely fast training and easier architecture optimization compared to neural networks
- Can operate on large datasets efficiently
- Could potentially be used in place of a neural network in the first and/or second stage of the model

Implementations to Try

- Intermediate weight averaging
- Parallel training/intermediate mean evaluation
- Stigmergic optimization
    - Stimergic Reinforcement Learning: Attractor Selection, ‘Pheromone Placement’, Rewards, Action Priority, Defining Loss for Evaluation and Behavior Modules
- XGBoost as part of model architecture