**Master.rmd**

Large dataset XLarge.

Read data / Input Specs

Input Specifications = [**M , missing values, min\_FSR, max\_FSR, min\_SSR, max\_SSR**]

Where:

**M**: number of the CBDA-SL iteration to be performed on XLarge

**Missing values**: % of missing values to introduce in the dataset (just for testing, to mimic real cases)

**min\_FSR** : Lower bound for the % of features/columns sampling

**max\_FSR**: Upper bound for the % of features/columns sampling

**min\_SSR**: Lower bound for the % of subjects/rows sampling

**max\_SSR:** Upper bound for the % of subjects/rows sampling

For each j=1:M, the Big Data Matrix XLarge is sampled to generate a Xj and a Yj by sampling a value in the ranges [**min\_FSR, max\_FSR**] and [**min\_SSR, max\_SSR**], respectively.

Xj is then imputed and normalized. Imputation and normalization can be done on the XLarge, before sampling is performed. A trade-off in computation time can be investigated; however it should be significant only when the original dataset is very large and the FSR/SSR are not small. If FSR/SSR are very small (~5%), an inline/real-time imputation might be more efficient (we use missForest, with max # of iterations ~5-10 for convergence).

FSR and SSR specifications are used to generate kj and nj, which represent the subsets of features/columns and subjects/rows, respectively. kj and nj are used to generate the matrices Xj and Yj, to be passed to the CBDA-SL algorithm.

**Pseudocode (sort of) of the CBDA-SL algorithm as implemented in the LONI pipeline workflow**

For (row i=1: i\_exp) [# experiments 5-50]

{

for (j=1:M, M~5K-50K) [M is j\_global in the workflow]

{

SuperLearner (j)(Xj,Yj,…)

}

}

SuperLearner (j)

{

1. Combines many learners/optimization algorithms [10-30 different algorithms]
2. Performs Cross Validation for each one (~ 10 CV folds)
3. Generate predictions
4. Save results in RData workspaces […….light\_j.RData]

}

Knockoff.filter(j)

{

Returns features selected after passing matrix Xj and outcome Yj

}

The workflow is now parallelized; each j is an independent job. Multiple experiments can be combined in a single pipeline workflow. However we are constrained by the total number of jobs that can be submitted in a single workflow on Cranium (3K).

The latest version of the pipeline workflow combines 3 modules of 3K jobs each for each experiment, for a total of 9K jobs for each experiment.

**Post Optimization process**

**1st Step – Workspaces consolidation**

For (i=1: i\_exp) [# experiments 5-50]

{

for (j=1:M)

{

Load …..Light\_j.RData

}

Save the combine workspace as Exp\_i.RData

}

**2nd Step – MSE Ranking and Feature Mining**

Feature Mining can be performed in 3 ways:

1. By single experiment
2. Combining all the experiments
3. Combining subsets of experiments

The following points are the same across 1, 2 and 3 above for the SuperLearner predictions.

* 1. Calculate Mean Square Error between data to be predicted and the CBDA-SL predictions – MSE\_j
  2. Select the TOP MSEs (TOP=~10-50)
  3. Extract the features kj used for each of the TOP MSEs and build an histogram
  4. Merge all the 9000 features selected by each knockoff.filter(j) and build an histogram
  5. Spikes in the histograms will indicate the features that are most common among the best predictions 🡪 Feature Mining

We are testing the CBDA-Sl and the Knockoff filter on ad hoc datasets to determine their FDRs and LOD (level of detection) of TP (true positive) features. We are still working on determining the rate of convergence of the algorithms under different specs (primarily the # of jobs needed). Preliminary analyses of the results suggest 2-3K as an optimal size for M; however that can be dataset-dependent.

Each job is ~2-10 minutes. If we use the CV.SuperLearner function ~20-100 minutes (still within the limits of XSEDE-OSG – Condor pool optimal scenario, which is 1-2 hrs per job)