**Master.rmd**

Large dataset BigData.

Different experiments [i\_exp] to perform on BigData to test the CBDA-SL and Knockoff algorithms.

Each experiment is uniquely identified by a set of input specifications that are read from an argument file.

Input Specifications = [**M , missing values, min\_FSR, max\_FSR, min\_CSR, max\_CSR**], where:

**M**: number of the BigData subsets on which perform Knockoff Filter and CBDA-SL feature mining

**Missing values**: % of missing values to introduce in BigData (used 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\_CSR**: Lower bound for the % of cases/rows sampling

**max\_CSR:** Upper bound for the % of cases/rows sampling

Sampling ranges for cases(**CSR -** Cases Sampling Range)and features(**FSR –** Feature Sampling Range) are then defines as follow:

**FSR** = [**min\_FSR, max\_FSR**]

**CSR =** [**min\_CSR, max\_CSR**]

Before each experiment is performed, the necessary data wrangling (e.g., cleaning, harmonization, aggregation and more) is performed on BigData to generate 2 sets of data:

1. Data to set aside for validation [Xpred,Ypred] and never used for learning/training. We hold off 20% of the subjects for validation and confusion matrix construction. This fraction to hold off can be set as an input as well.
2. Data to be sampled for learning/training by the CBDA-SL and Knockoff algorithms [Xtemp,Ytemp]. We fix the seed so the sets [Xpred,Ypred] and [Xtemp,Ytemp] are the same across the M subsamples.

Here we refer to X-datasets as datasets of features, while Y-datasets are datasets of outcomes (binomial/multinomial). If BigData is within the order of GBs, alternative strategies can be implemented to avoid loading large datasets before launching the CBDA-SL and Knockoff algorithms. For example, we could generate all the M data subsets offline and save them in separate RData workspaces, and then load the appropriate ones for each of the M job.

Also, the set of input specifications should be carefully designed so that, given certain constraints of the algorithms used, all the conditions are fulfilled (e.g., for the knockoff filter 🡪 # subjects > # of features, see below).

For each j=1:M, we load the BigData and generate a Xtempj , Xpredj and a Ytempj by sampling Xtemp, Xpred and Ytemp, respectively. 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 then used to generate the matrices Xtempj , Xpredj and a Ytempj to be passed to the Knockoff Filter and CBDA-SL algorithms. We can enforce balanced Ypredj , Xpredj , Xtempj and Ytempj (i.e., equal number of cases for each outcome category). The package SMOTE in R automatically balanced the datasets if the discrepancy is greater than a threshold (e.g., 50%).

Xtempj and Xpredj are then imputed and normalized. Imputation and normalization can also be done on the BigData, 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. For imputation we use the function missForest, with max # of iterations ~5-10 for convergence (see the description of the different algorithms for details).

**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]

{

Knockoff.filter (j)(Xj,Yj,…)

SuperLearner (j)( Xj,Yj,…)

Save results in RData workspaces […….light\_j.RData]

}

}

Knockoff.filter(j)

{

1. The knockoff filter is a procedure for controlling the false discovery rate (FDR) when performing variable selection. The version implemented here has the constraint that n>p (# of subjects > # of features). Thus, particular attention is needed in building the set of input specification.
2. Returns features selected after passing matrix Xj and outcome 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 and 2 metrics to later rank them: Mean Square Error [MSE\_j] and Accuracy [Accuracy\_j] comparing to Ypred

}

Save results in RData workspaces […….light\_j.RData]

{

The workspace j generated after performing the operations above is too big to be saved as is. Thus, the SLj object (~50-100Mb) is deleted, and the workspace j saved for post-optimization purposes.

}

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 and metrics**

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

{

for (j=1:M)

{

Load …..Light\_j.RData

}

1. The consolidated array of the metrics MSE and Accuracy are generated, sorted and the top 20 jobs [j] are selected. The selection of the top jobs is a parameter in the code and can be expanded or trimmed.
2. To avoid stack overflow errors, several named objects are deleted (i.e., j=1,2,…, M 🡪 MSE\_j and Accuracy\_j are deleted
3. The consolidated Rdata workspace is saved (~5-10 Mb).
4. The single j Rdata workspaces are then deleted, resulting in 1 complete Rdata workspace with all the results (out ot 9000).

}

**2nd Step – Results display (Rmd file)**

A R Markdown file generates results for single or multiple experiments pertaining the analysis of a specific BigData. The distribution of the features (kj) suggested by the best metric measures (e.g., lowest MSE, highest Accuracy) is plotted in a density histogram for the KO, MSE and Accuracy, to possibly display spikes/signals 🡪 Feature Mining

Tables are also generated with information on frequency of occurrences and densities of each of the features in the top 20 jobs.

The top features selected are also displayed by label.

**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 CBDA-SuperLearner predictions.

* 1. Calculate Mean Square Error between data to be predicted and the CBDA-SL predictions – MSE\_j
  2. Calculate the Accuracy Error between data to be predicted and the CBDA-SL predictions – Accuracy\_j (we use the confusion matrix tool).
  3. Select the TOP MSEs and Accuracy (TOP=~10-50)
  4. Extract the features kj used for each of the TOP MSEs and Accuracies and build an histogram
  5. Merge all the 9000 features selected by each knockoff.filter(j) and build an histogram
  6. 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)