Big Data Analytics Suite

Purpose of Benchmark

The big data analytics suite includes unsupervised and supervised machine learning algorithms: K-Means clustering, Principle Component Analysis (PCA), and Support Vector Machine (SVM). The suite is based on R.

Characteristics of Benchmark

K-Means: The benchmark is defined by computing the observation labels (class assignments) and centroids for k=2, 3, and 4. The data should consist of rows sampled from one of 3 random normal distributions: one with mean 0, one with mean 2, and one with mean 10. Each should have variance 1. The rows should be drawn at random from these distributions. To reduce the variance in steps needed for convergence, the maximum number of iteration has been set to 2. Our implementation of the benchmark can be found at “benchmarks/r/kmeans.r”.

PCA: The benchmark is defined as computing the first and last of the "standard deviations" from a PCA on a distributed matrix by way of taking the square roots of the eigenvalues of the covariance matrix. The "first and last" requirement is to avoid approximate methods. Using the covariance matrix is mathematically equivalent to computing the SVD of the mean-centered input matrix, although it is computationally easier. The data should be random normal. Our implementation of the benchmark can be found at “benchmarks/r/princomp.r”.

SVM: The benchmark consists of a linear 2-class SVM fit using 500 iterations, calculating the feature weights. The data should consist of an intercept term together with rows sampled from one of 2 random normal distributions: one with mean 0, and one with mean 2; each should have variance 1. The rows should be drawn at random from these distributions. The SVM is not expected to converge given the number of features and iterations; this allows for easier comparisons of benchmarks as data sizes/layouts change. Our implementation of the benchmark can be found at “benchmarks/r/svm.r”.

Mechanics of Building Benchmark

1. Install R (see instruction https://cran.r-project.org/doc/manuals/r-patched/R-admin.html)
2. Install rlecuyer, pbdMPI, and kazaam (under “benchmarks/source/r”) in following order:
   ```
   R CMD INSTALL rlecuyer_0.3-4.tar.gz
   R CMD INSTALL pbdMPI_0.3-3.tar.gz
   R CMD INSTALL kazaam_0.2-0.tar.gz
   ```

Mechanics of Running Benchmark

Multi-node benchmarks (K-Means, PCA, SVM) accepts a number of “local” rows and “global” columns. The total number of rows grows proportionally to the number of MPI ranks. Thus, each benchmark measures scaling in the weak sense.

Run the R code via:

```
mpirun -np num_ranks Rscript princomp.r num_local_rows num_global_cols
```

For example, to run the PCA code with 16 ranks, 250 total columns, and a total of 16,000 rows (local number of rows 1000), you would run:

```
mpirun -np 16 Rscript princomp.r 1000 250
```

CORAL class problem:

1. Ensemble of individual jobs of input data size at least 1024GB (feature columns is fixed at 250) for K-Means, PCA, and SVM to fill up the entire system.
2. Figure of Merit (FOM) is defined as total-input-data-size (in TB) / average-runtime (in seconds). Note, the benchmark outputs min, mean, and max time of MPI ranks, and the max time is taken as the runtime of an individual job.
3. The overall improvement in FOM S_{bdas} for big data analytic suite is the geometric mean of improvements in FOM for K-Means, PCA and SVM, i.e.

$$S_{bdas} = \left(\prod_{i=1}^{3} S_i\right)^{\frac{1}{3}}$$

   where S_i = *projected FOM$_i$ / baseline FOM$_i$*, and i runs from K-Means, PCA, to SVM.
4. Expected overall improvement in FOM: 50 e.g. solve 10x larger problem 5x faster.

Verification of Results

We use random data in the benchmarks to keep the benchmarks as amenable to every hardware solution possible. We strongly believe this approach is to the advantage of every vendor. However, this makes the benchmark runs difficult to verify. So, we have included small verification scripts to be run in companion with the benchmarks themselves.

Each validation script should be run on two nodes and use the same (specified) kernel as its benchmark counterpart. Each will use the famous iris dataset of R. A. Fisher (included). The rows of the dataset have been randomly shuffled and the species variable has been coded to 1=setosa, 2=versicolor, and 3=virginica. Any other settings we leave to the vendor. Performance measurements are not desired, only the validation.

Each of the validation scripts is completely self-contained. You can run any of them via:

```
mpirun -np 2 Rscript $BENCHMARK.r
```

K-Means: Using k=3 centroids (the true value), and 100 starts using seeds 1 to 100, the labels for each observation should be computed. These will be compared against the true values (from the 'species' label of the dataset) using [rand measure] (https://en.wikipedia.org/wiki/Rand_index). Take the largest among these values. This should be greater than 75% to be considered successful. Our implementation of the validation script can be found at "validation/r/kmeans.r".

PCA: This validation script shows that the PCA benchmark is working correctly by testing the SVD kernel. The validation consists of reading the iris dataset, removing the species column, factoring the resulting matrix, and then multiplying the factored matrices (from SVD) back together. The mean absolute error (average of the difference in absolute value) of the two matrices should be computed. The test passes if this value is less than the square root of machine epsilon for each type (as specified by IEEE 754). Our implementation of the validation script can be found at “validation/r/svd.r”.

SVM: The data consists of an intercept term together with the iris dataset without the species column. The response should be taken to be 1 for setosa (coding 1 in the species variable) and -1 otherwise. Use a maximum of 500 iterations to fit an SVM on the data and response. Report the accuracy (number correctly predicted). This should be greater than 80%. Our implementation of the validation script can be found at “validation/r/svm.r”.

Figure of Merit on Titan

Baseline FOM on Titan: Size of individual job = 128 nodes, Size of individual input = 1024 GB, Size of ensemble = 30

Benchmarks	K-Means	PCA	SVM
Projected Baseline FOM (TB/s)	1.8	6.0	0.24

For example, the calculation of the projected baseline FOM for PCA: FOM = 146*1.024 (TB) /24.8 (s) = 6.03 TB/s, where 146 is the total number of 128-node jobs that Titan can accommodate, and 24.8 is the average “max” time.