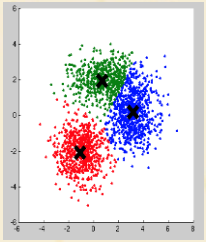
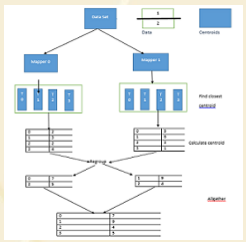
MiniBatch K-Means Implementation using Harp on Hadoop

Vikrant Kaushal, Shalabh@IU

ABSTRACT

Mini batch K-means algorithm. Its main idea is to use small random batches that are subset of data. Each iteration sample a new random batch and updates the clusters. Same is repeated until convergence or fixed maximum iterations. Convergence can be detected when there is no changes in the clusters occur in several consecutive iterations.





MATERIALS AND METHODS

We used Euclidean distance to find distance between two data-points for assigning centroids. We implemented the mini-batch k-means to cluster documents in the given data set. We have implemented traditional k-means that has helped us to grasp idea behind the algorithm and also helped us understand why we need mini-batch k-means. Time-complexity of Mini Batch k-means is many folds less than traditional k-means. Formulas used:

Euclidean Distance: $\sqrt{\sum_{i=1}^{n_2}(q_i - p_i)^2}$.

Precision: $\frac{True\ Positive}{True\ Positive + False\ Positive}$

DataSet

We have used Newsgroup dataset. The 20 newsgroups dataset comprises around 18000 newsgroups posts on 20 topics. It contains more than 96k features in a inverted index. Our dataset is helping us to parse and work on inverted index efficiently. We wrote a combiner also as required by harp for our class. Here is the structure of our class:

Example data:

Class	Feature:Value	
1	10:3000	20:25.

Implementation

Here are steps we followed for implementation:

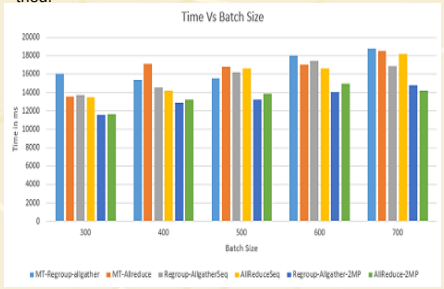
1: Split file into number of mappers & move files to HDFS.
2: Pick initial centroids randomly.
3: Master process will read centroid file and broadcast.
4: We created a new data structure to hold our data and thus wrote combiner for same.Here is our Class:

```
public class MiniBatch extends Writable {
  private int classLabel;
  private int allocatedCentroid;
  private Int2DoubleOpenHashMap featureValueMap;
}
```

5. Each Iterations does following:
   a. Randomly picks data points equal to size batch.
   b. Each mapper spawns maximum number of thread. Data is divided into size/noOfThreads for every thread.
   c. Closest centroid is found using Euclidean distance.
   d. We are working with Regroup-Allgather and AllReduce.
   e. Centroids are updated using assigned data points.
6: Evaluation: We are using purity for evaluation. We find majority class for every clusters and thus calculate Precision.
   To evaluate better we also recorded timings of Same Algorithms sequentially.

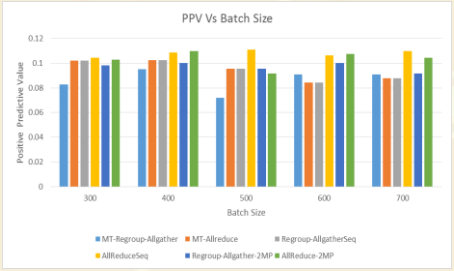
Time v/s Batch Size

We have compared performance of Regroup-Allgather, Allreduce with and without Multi-Threading and can see that as the batch size is increasing AllReduce without multi threading is performing best among all methods tried.



Precision

As per our experiments we have found that Accuracy is better If we run Harp with Multi-Threading.



High CPU Usage

While Using multi-threading during the project CPU usage exceeded more than 100% and thus it leads to low performance. Below is the snap from Juliet for same.

```
USER      PR  NI  VIRT  RES  SHR S %CPU %MEM
sshalabh  20   0 91.9g 3.1g  19m S 3978.7  2.4
sshalabh  20   0 3072m 718m  19m S  4.3  0.6
```

CONCLUSIONS

Harp framework aims to provide fast machine learning solutions for Big Data applications.
We can see the that the quality of cluster(positive predictive value) is much better for the all-reduce without multi-threading paradigm as compared to others when the batch size is being altered. Also the clustering process is surprisingly fastest for the all reduce without multi threading.
So we can also conclude that on overdoing multi threading much of the computer resources gets used in distributing and collecting data from threads thus leads to less performance.

REFERENCE

•[1] L. Bottou and Y. Bengio. Convergence properties of the k means algorithm. In Advances in Neural Information Processing Systems. 1995.

•[2] Paul S. Bradley, Usama M. Fayyad, and Cory Reina. Scaling clustering algorithms to large

•databases. In Rakesh Agrawal, Paul E. Stolorz, and Gregory Piatetsky-Shapiro, editors, KDD,

•pages 9–15. AAAI Press, 1998.

•[3] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the l1-ball for learning in high dimensions. In ICML '08: Proceedings of the 25th international conference on Machine learning, 2008.