Data Clustering

using MapReduce: A Look at Various Clustering Algorithms Implemented with MapReduce Paradigm

Data Clustering has always been one of the most sought after problems to solve in Machine Learning due to its high applicability in industrial settings and the active community of researchers and developers applying clustering techniques in various fields, such as Recommendation Systems, Image Processing, Gene sequence analysis, Text analytics, etc., and in competitions such as Netflix Prize and Kaggle.

In this article I will describe three clustering algorithms – K-Means, Canopy clustering and MinHash clustering implemented by both sequential approach and MapReduce approach.

MapReduce paradigm has recently gained popularity due to the ability to solve many of the problems associated with rising data volume by modeling algorithms using MapReduce paradigm. These algorithms essentially work with data in a partitioned (into shards) form and have to consider the distributed nature of partitioned data and model the algorithm accordingly. Also, implementations such as Apache Hadoop, Phoenix and others have paved way for algorithm writers to model their approach in MapReduce fashion without worrying about the underlying architecture of the system. One such field of algorithms is the field of Machine Learning, which has worked tremendously towards implementing algorithms in MapReduce paradigm which work on a MapReduce system. Open source implementations such as Apache Mahout (*http://mahout.apache.org*) is being developed by team of active contributors and used in various organizations (*https://cwiki.apache.org/confluence/display/MAHOUT/Powered+By+Mahout*). This article introduces MapReduce in a nutshell, but assumes familiarity of programming paradigms.

MapReduce 101

Google introduced and patented MapReduce – A programming paradigm of processing data with partitioning, and aggregation of intermediate results. An overloaded term in Google's context – its also a software framework to support distributed computing on large data sets on clusters of computers, implemented in C++ to run Jobs written with MapReduce paradigm. The name "MapReduce" and the inspiration came from map and reduce functions in functional programming world.

`map function`

In functional programming, `map()` function applies a function on input data. In MapReduce paradigm, `map()` function applies the function on an individual chunks (shard) of partitioned data. This partitioning is done by the underlying file system (Google file system, HDFS). The size of the partition is a tunable parameter.

Let us take an example of a function, f, which converts input rectangle into a sphere. Now the `map()` function in the MapReduce paradigm get the following parameters – map(f, input).

f – A processing function. In out case, its

input – Input is split into many shards, based on the input split size (Figure 1).

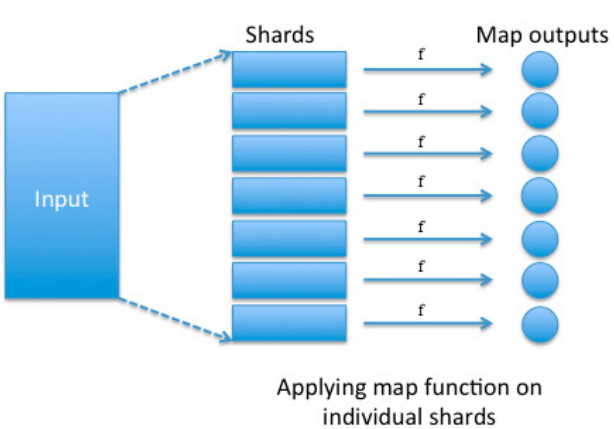

Figure 1. *Map(f, input) applied to input*

The programming paradigm for Hadoop MapReduce requires map function to take inputs as key-value pairs in the form of records. The `map()` takes in input key and input values to produce intermediate value lists of the processed keys.

```
map (input_key, input_value) -> (output_key,
 intermediate_value_list)
```

`map()` functions run in parallel, creating different intermediate values from different input data sets.

`reduce function`

After the map phase is over, all the intermediate values for a given output key are combined together into a list. `reduce()` combines those intermediate values into one or more final values for the same output key. `reduce()` functions also run in parallel, each working on a different output key. It is important to note that reduce method cannot start until and unless the map phase is completely finished (Figure 2).

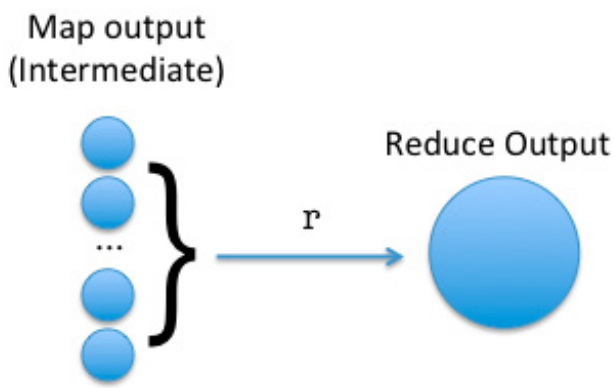

Figure 2. *Reduce(r, intermediate_value_list) applied to intermediate_value_list to get the final output of the mapreduce phase*

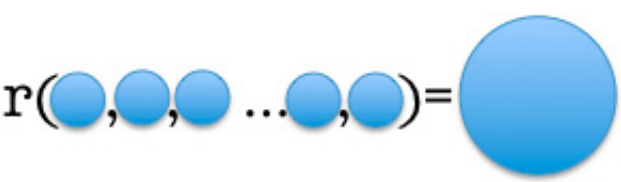

Figure 3. *Reducer function r()*

As seen above, the reducer function takes a list of values associated with a key as an input, and performs processing over the list. In the above representation, we have only one reducer task, but it depends on the configuration settings of the system. In our example, the reducer function `r()` takes the intermediate map output and produces the final output. For our example, reducer function `r()` looks like: Figure 3.

Machine Learning 101

Machine Learning is a branch of Artificial Intelligence which deals with building of systems that 'learn' from data without the need of explicit programming for all the use-cases. Its a study of ways using which a computing system can 'learn' – understand the input, make predictions based on the input, and learn from the feedback. The applications of machine learning have been seen in many varied fields, e.g. e-mail analytics, Genome project, image processing, supply chain optimization, and many more. These applications are also motivated by the simplicity of some of the algorithms and their efficiency, which is useful in real-world applications. Its differentiates itself from the field of Data Mining by focussing only on the known properties learned from the data. Data Mining techniques may inherently use machine learning techniques to mining for the unknown properties.

Machine Learning algorithm can be classified into the following subcategories (source: Wikipedia; see Figure 4).

Supervised learning – Generates a model which takes the input and produces the desired output. It gets trained on a training set to understand the input and later works on the incoming data to provide the output.

Unsupervised learning – It models the input into a set of representations produced after the algorithm. It does not follow the train-test model as used in supervised learning methods, but works on finding a representation of the input data. Clustering is one of the prominent approaches in this space.

Semi-supervised learning – Combines supervised and unsupervised learning approaches to generate ap-

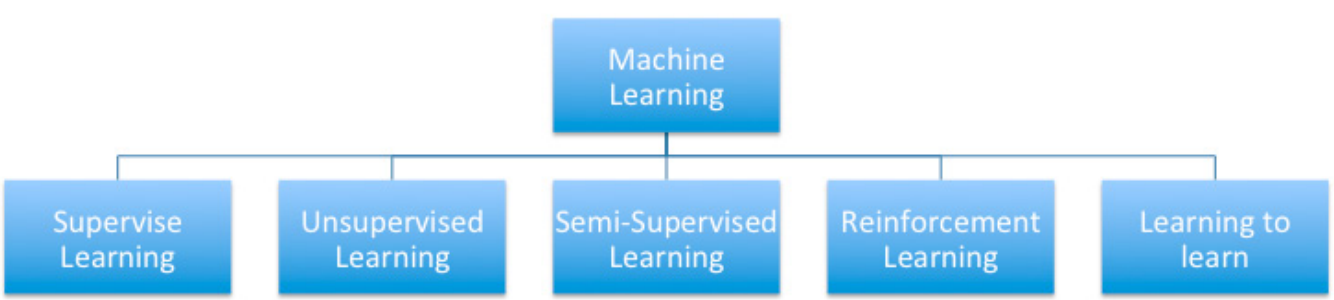

Figure 4. *Machine Learning Taxonomy (Source: Wikipedia)*

proximate classifiers. Predicting outputs from fixed set of inputs (train) and applying the learnings to improve the test set of input.

Reinforcement learning – Learns from the feedback to the output. It improves as the interactions with the system increase. One of the most important approaches is to use the Artificial Neural Networks, which feedback the response, with each interaction with the system.

Learning to learn – It uses experience as a part of learning. The experience factor comes while learning a problem together with other similar problems simultaneously. It allows to share the experience from various problems and build a better model, applied to the all the problems used for learning simultaneously.

Example of Supervised Learning

Google Prediction API (*https://developers.google.com/prediction/*) takes a training set as an input (Listing 1).

After getting trained, it predicts the language of the input by making a model based on the training input.

Unsupervised Learning

As stated above, unsupervised learning refers to the problem of finding hidden structure or representation in unlabeled data. Simply stated – finding a way to store similar data points into a data structure (e.g. cluster them together). Since the examples given to the learner are unlabeled, there is no error or reward signal to evaluate a potential solution. This distinguishes unsupervised learning from supervised learning and reinforcement learning.

Clustering

Cluster analysis refers to the process of grouping similar objects without labels. Similarity is directed by the business needs or by the requirements of the clustering algorithm. Its eventual goal is to group the similar objects into same cluster and distinct objects into distinct groups. The distinction is directed similarity measure. It is mainly used in explorative mining, and in statistical analysis techniques such as pattern recognition, information retrieval, etc.

Purposes of using data clustering [1]:

- Underlying Structure: to gain insight into data, generate hypotheses, detect anomalies, and identify salient features.
- Natural Classification: to identify the degree of similarity among forms or organisms (phylogenetic relationship).
- Compression: as a method for organizing the data and summarizing it through cluster prototypes.

We'll look into three ways to cluster data points: K-Means, Canopy Clustering, and MinHash Clustering, and look into the approaches to implement these algorithms: Sequential and MapReduce paradigm. These three algorithms have proved to be useful in various applications such as recommendation engines [2], image processing, and the use as a pre-clustering algorithm [3].

K-Means Clustering

K-Means clustering is a method of cluster analysis which aims to form k groups from the n data points taken as an input. This partitioning happens due to the data point associating itself with the nearest mean.

Classical Approach

The main steps of k-means algorithm are as follows:

Listing 1. *Input file to the model with two fields – <Language, Sample Phrase in Language>*
(Source: Excerpts from file available at https://developers.google.com/prediction/docs/language_id.txt)

```
"French",  "Cette menace acheva d'intimider l'hôte. Apres le roi et M. le cardinal, M. de Tréville était l'homme
                    dont le nom peut-etre était le plus souvent répété par les militaires et meme par les
                    bourgeois. Il y avait bien le pere Joseph, c'est vrai; mais son nom a lui n'était jamais
                    prononcé que tout bas, tant était grande la terreur qu'inspirait l'Éminence grise, comme on
                    appelait le familier du cardinal."
"Spanish", "Soneto"
"French","«Voyons, l'hôte, dit-il, est-ce que vous ne me débarrasserez pas de ce frénétique? En conscience,
                    je ne puis le tuer, et cependant, ajouta-t-il avec une expression froidement menaçante,
                    cependant il me gene. Ou est-il?"
"English","This was not an encouraging opening for a conversation. Alice replied, rather shyly, 'I--I hardly
                    know, sir, just at present--at least I know who I WAS when I got up this morning, but I
                    think I must have been changed several times since then.'"
"English","And she went on planning to herself how she would manage it. 'They must go by the carrier,' she
                    thought; 'and how funny it'll seem, sending presents to one's own feet! And how odd the
                    directions will look!"
```

- Randomly select k data points to represent the seed centroids.
- Repeat steps 3 and 4 until cluster membership stabilizes – either number of iterations specified by the user, or the dimensions of centroid does not change.
- Generate a new partition by assigning each data point to its closest cluster center – assignment happens based on the closest mean.
- Compute new cluster centers – calculating new centroids using the mean formulae for multidimensional data-points (Figure 5).

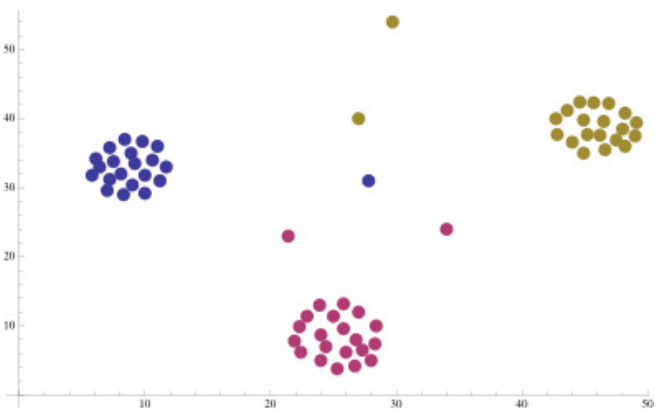

Figure 5. *K-Means representation applied on a dataset of 65 data points (n=65) and number of clusters to be 3 (k=3). (Source of the dataset – [4], and representation plot produced by Wolfram Mathematica)*

Theoretically, Let X = {xi}, i = 1,...,n be the set of n dimensional points to be clustered into a set of K clusters, C ={ck, k=1,…,K}. K-means algorithm finds a partition such that the sum of squared error (distances) between the empirical mean of a cluster and the points in the cluster is minimized. Let μk be the mean of cluster ck. The sum of the squared error between μk and the points in cluster ck is defined as:

$$J(c_k) = \sum_{x_i \in c_k} \| x_i - \mu_k \|^2 .$$

The goal of K-Means is to minimize the sum of the squared error over all K clusters,

$$J(C) = \sum_{k=1}^{K} \sum_{x_i \in c_k} \| x_i - \mu_k \|^2 .$$

As we can observe, we would require to get all the data-points in memory to start their comparison with the centroids of the cluster. This is plausible only in conditions where the data sets are not very large (very large depends on the size of your RAM, processing power of your CPU and the time within which the answer is expected back). This approach does not scale well for two reasons – first, the complexity is pretty high – k * n * O (distance metric) * num (iterations), and second, we need a solution which can scale with very large datasets (Data Files in GBs, TBs, ...). MapReduce paves

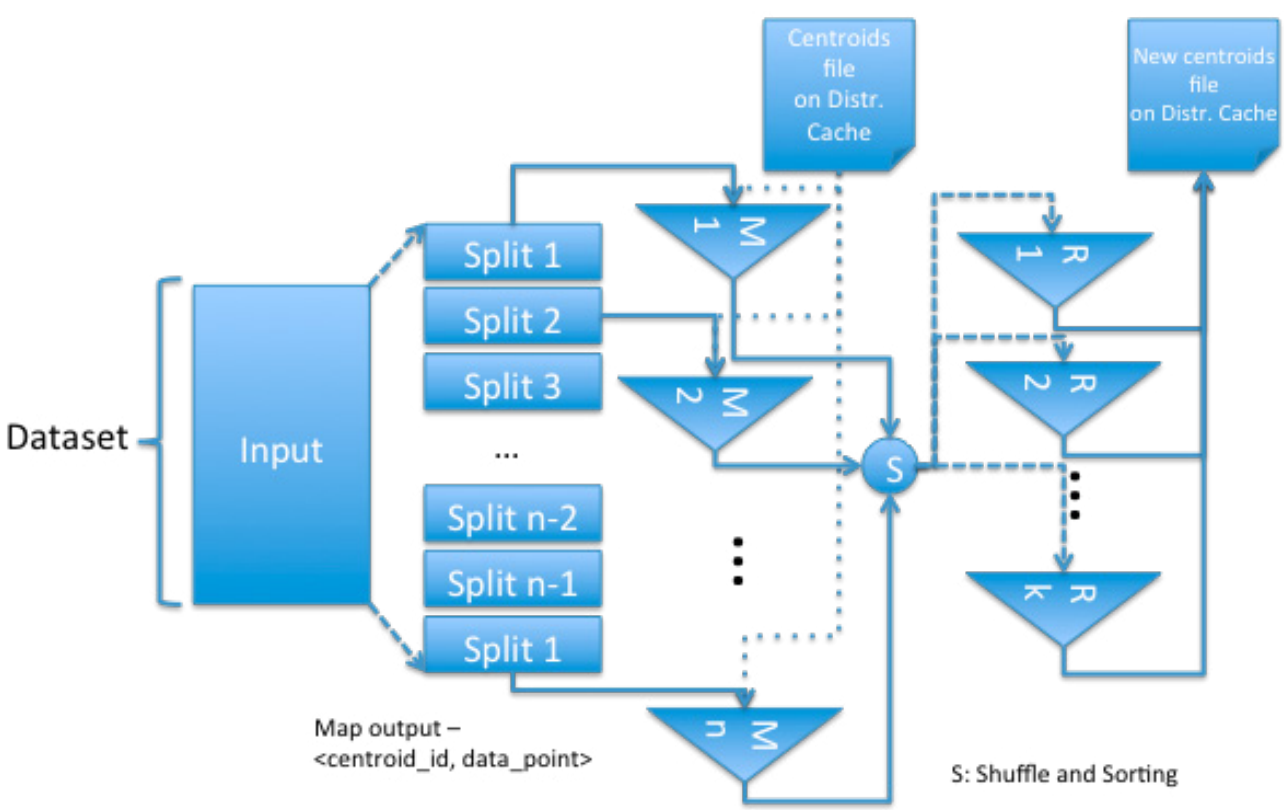

Figure 6. *Single pass of K-Means on MapReduce*

way for making such a solution feasible with distributing into tasks to work on smaller chunks of data, and making use of a multi-node setup (distributed system). There can be multiple approaches to do the sequential approach better, or many improved ways to implement MapReduce version of k-means.

Distance Metrics

The k-means is typically used with the Euclidean metric for computing the distance between points and cluster centers. Other distance metrics, with which k-means is used are Mahalanobis distance metric, Manhattan distance, Inner Product Space, Itakura–Saito distance, family of Bregman distances, or any metric you define over the space.

MapReduce Approach

MapReduce works on keys and values, and is based on data partitioning. Thus, the assumption of having all data points in memory fails in this paradigm. We have to design the algorithm in such a manner that the task can be parallelized and doesn't depend on other splits for any computation (Figure 6).

The Mappers do the distance computation and spill out a key-value pair – <centroid_id, datapoint>. This steps finds the associativity of a data point with the cluster.

The Reducers work with specific cluster_id and a list of the data points associated with it. A reducer computes new means and writes to the new centroid file.

Now, based on the user's choice, algorithm termination method works – specific number of iterations, or comparison with centroid in the previous iteration.

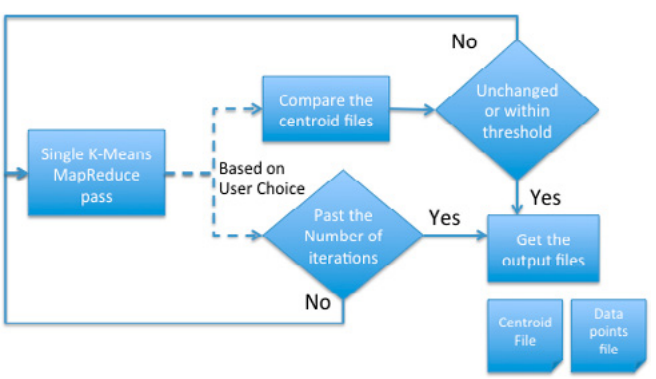

Figure 7. *K-Means Algorithm. Algorithm termination method is user-driven*

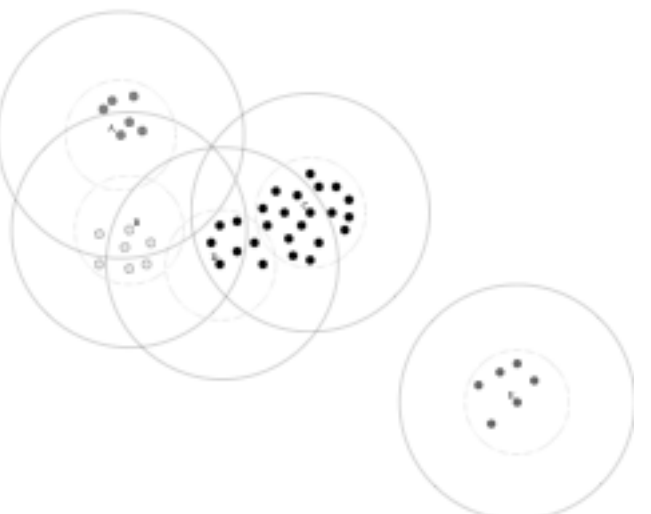

Figure 9. *An example of four data clusters with the canopies superimposed. Point A was selected at random and forms a canopy consisting of all points within the outer (solid) threshold. (Source: [3])*

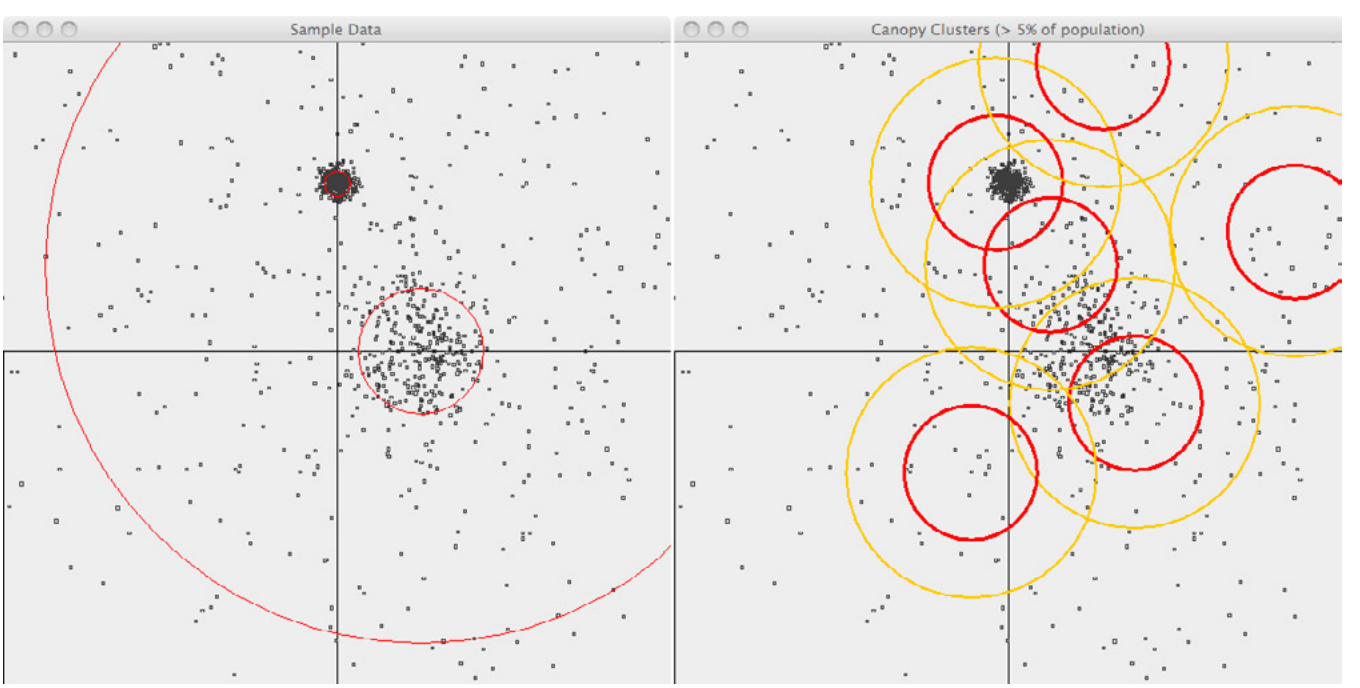

Figure 8. *(a) 1100 samples generated (b) Resulting Canopies superimposed upon the sample data. Two circles, with radius T1 and T2. (Source: https://cwiki.apache.org/MAHOUT/canopy-clustering.html)*

MapReduce based K-Means implementation is also implemented part of Mahout [7][9] (Figure 7).

Canopy Clustering

Canopy clustering is an unsupervised clustering algorithm, primarily applied as a pre-clustering algorithm – its output is given as input to classical clustering algorithms such as k-means. Pre-clustering with canopy clustering helps in speeding up the clustering of actual clustering algorithm, which is very useful for very large datasets.

Cheaply partitioning the data into overlapping subsets (called "canopies") Perform more expensive clustering, but only within these canopies (Figure 8 and Figure 9).

Algorithm

The main steps of partitioning are as follows:

Let us assume that we have a list of data points named X.

1. You decide two thresholds values – T1 and T2 where T1 > T2.
2. Randomly pick 1 data point, which would represent the canopy centroid, from X. Let it be A.
3. Calculate distances, d, for all the other points with point A (see 2). If d < T1, add the point in the canopy. If d < T2, remove the point from X.
4. Repeat steps 2 and 3 for all the data points, until X is not empty.

Mahout [11] uses a three step process for finding the canopy centroids [10].

- The first step involves using the above method aver a subset of the input data to find canopy centers.
- The second step involves using MapReduce to normalize its centroids and get a final list of canopy clusters. In the mapper, each point which is found to be within an existing canopy will be added to an internal list of Canopies. The mapper updates all of its Canopies and produces new canopy centroids, which are output, using a constant key ("centroid") to a single reducer <“centroid”, vector>. The reducer receives all of the initial centroids and again applies the canopy measure and thresholds to produce a final set of canopy centroids which is output (i.e. clustering the cluster centroids).
- The third step involves actual clustering, similar to single pass K-Means, producing the canopies.

MinHash Clustering

It is a probabilistic clustering method which comes from the family of Locality Sensitive Hashing (LSH) algorithms. It is generally used in a setting where clustering needs to be done based on the range of the dimensions of the data points (especially only a single dimension, such as in [2], where the clustering is done only on the basis of the news article seen by the user). The probability of a pair of data points assigned to the same cluster is proportional to the overlap between the set of items that these users have voted for (clicked-on). So if the overlap of the set of the items clicked is high, the probability of the two data-points ending up in the same cluster is high (‘high’ is domain specific).

Example of MinHash Clustering

Let us assume hat we have some e-commerce website with a catalog of products.

- Get a random permutation of the catalog. We’ll refer it as R (Figure 10).
- The random permutation generated is then used to find out the clusters of users. We try to find clusters the users belong to.
  - Checking the first product from the R clicked by the user gives you the cluster number. The cluster ID is the product ID itself.
  - As we do this computation, we try to find the packets of users getting associated with the same product number.
  - After we get this number (the Hash), Add this information to the User Record. It can act as a separate field, and used by later processes. (Assuming the use of NoSQL database in this case)
  - The Clusters can be merged after or before this step.
    - To do a merged hash sets clustering, we need to make sure that the number of prod-

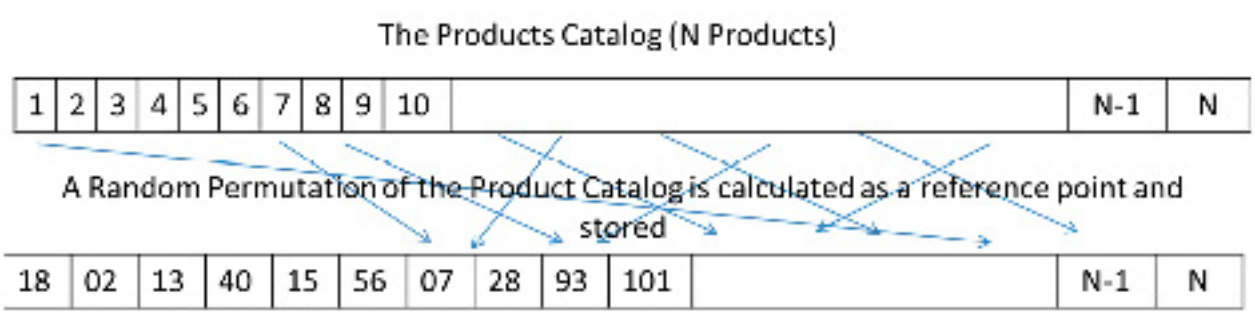

Figure 10. *Random Permutation of Catalog*

Figure 11. *The mappers share a common file (on distributed cache) containing the permutation of the catalog*

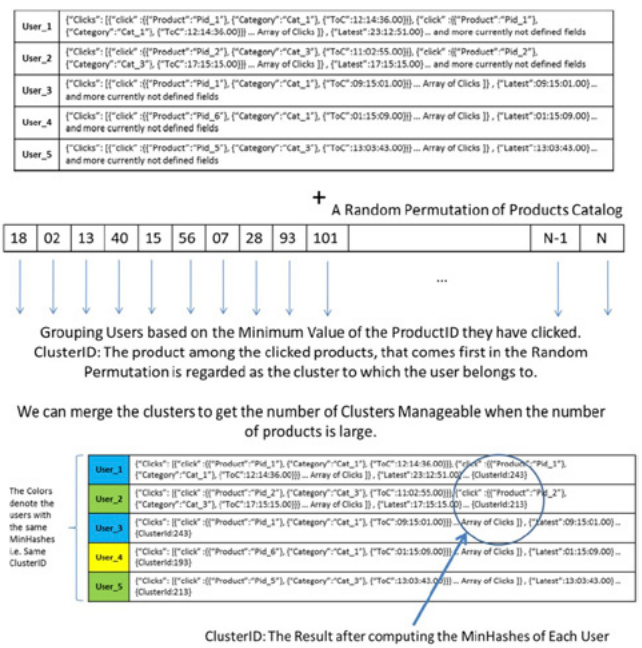

Figure 12. *MinHash Clustering. The example assumes a datastorage in a NoSQL database, such as MongoDB, in which ClusterID is appended as a column entry to the user*

References

[1] Jain, A. K. (2010) "Data clustering: 50 years beyond K-means", Pattern Recognition Letters 31, pp. 651–666.

[2] Das, A. S.; Datar, M.; Garg, A.; and Rajaram, S. (2007) "Google news personalization: scalable online collaborative filtering", WWW '07, pp. 271-280.

[3] McCallum, A.; Nigam, K.; and Ungar L.H. (2000) "Efficient Clustering of High Dimensional Data Sets with Application to Reference Matching", SIGKDD '00, pp. 169-178.

[4] Lingras, P. and West, C. (2004) "Interval Set Clustering of Web Users with Rough K-means", Journal of Intelligent Information System, Vol. 23, No. 1, pp. 5-16.

[5] Lingras, P. and West, C. (2004) "Interval Set Clustering of Web Users with Rough K-means", Journal of Intelligent Information System, Vol. 23, No. 1, pp. 5-16.

[6] Jain, A. K.; and Dubes, R. C. (1988) "Algorithms for Clustering Data", Prentice Hall.

[7] Introducing Apache Mahout – IBM developerWorks – *http://www.ibm.com/developerworks/java/library/j-mahout*

[8] Chu, C. T.; Kim, Sang K.; Lin, Y. A.; Yu, Y.; Bradski, G. R.; Ng, A. Y.; Olukotun, K. (2006) "Map-Reduce for Machine Learning on Multicore", In NIPS, pp. 281-288 available at http://www.cs.stanford.edu/people/ang/papers/nips06-mapreducemulticore.pdf

[9] Apache Mahout – K-Means Clustering – *http://cwiki.apache.org/MAHOUT/k-means-clustering.html*

[10] Apache Mahout – Canopy Clustering – *https://cwiki.apache.org/confluence/display/MAHOUT/Canopy+Clustering*

[11] Apache Mahout – *http://mahout.apache.org*

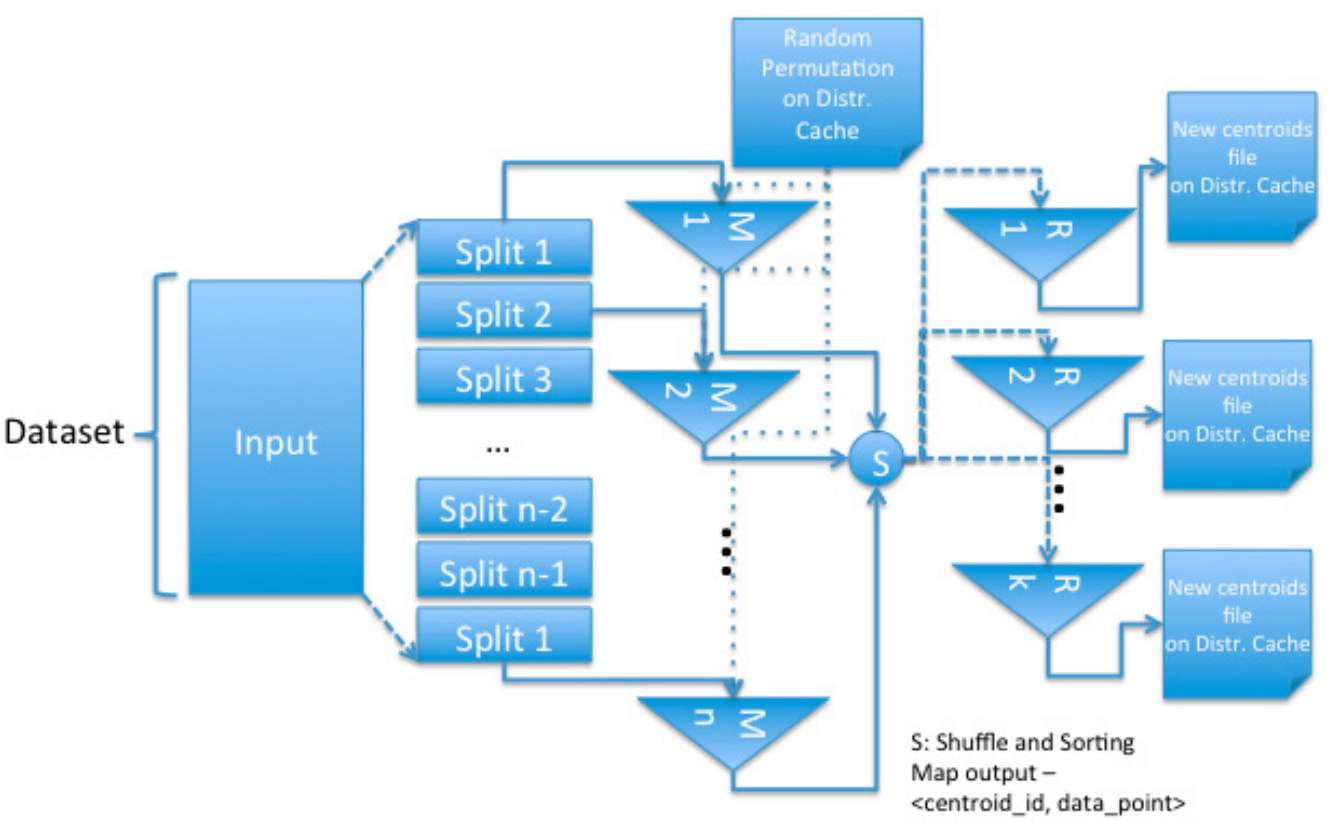

Figure 13. *MinHash Clustering MapReduce workflow. The reference range for finding cluster is shared across mapper using distributed cache*

ucts are known beforehand and the clusters formed are not too big which reduce the accuracy of the clusters.

- After: Merging here mean to put the records into a separate collection (table) based on the merged values.

Before: We can group the random permutations into 'q' groups early in the first step itself and then hash them based on these groups.

MapReduce Approach

As displayed in the example, for ease of understanding the algorithm, we are taking datapoints to be a click logger entry for a user. This would contain each click entry for products made by the user. The representation in Figure 14 describes the process of clustering the users based on clicks.

In the map phase, mappers would take the data points as an input and produce a 2-tuple entry contain cluster id (the minimum entry in the click list, from the product catalog, R). The mappers share a common file (on distributed cache) containing the permutation of the catalog (Figure 11).

Reduce phase would not perform any processing and just list of the users associated with the cluster_id (Figure 12 and Figure 13).

Conclusion

Clustering Algorithm are time-tested way to finding patterns in data, and with rising volume of data, its evident that these algorithms must scale. Industry and Academia is already doing a lot to scale clustering algorithms, by scaling it both vertically and horizontally. MapReduce helps in managing data partitions and parallel processing over these data shards. In this article, we saw only a handful algorithms from the vast number of machine learning techniques that can be molded into MapReduce[8][9].

VARAD MERU

Varad Meru, Software Development Engineer at Orzota, Inc.
I completed my Bachelors in Computer Science from Shivaji University, Kolhapur in 2011. After graduating, I joined Persistent Systems as a Software Engineer and worked on Recommendations Engines, Search Engines, E-mail Analytics, etc. At Orzota, I work Algorithms, Distributed Computing and Data Science. At Orzota, we are trying to Make 'Big Data' platform easy for consumption. I am active with HUGs, gave talks on Big Data and related technologies at various places and like mentoring next generation of professionals in the domain on Data Science.
LinkedIn: in.linkedin.com/in/vmeru/
Company Website: www.orzota.com