Analysis of GMM Training Times Using Spark

Dane Jordan
Master of Science Data Science
University of Washington
Seattle, WA
danejordan@live.com

***Abstract*—Apache Spark is a cluster framework designed to handle big data. It provides high-level machine learning tools via its Machine Learning Library (MLlib). When data is used to train a machine learning model, the process can be optimized in different ways to reduce cost and/or decrease the time required to train the model.**

In this paper, we test and analyze different hypotheses regarding the time to train a Gaussian Mixture Model (GMM) using Apache Spark. Parameters that we experiment with include: the number of partitions a file is split into, the size of the data, the number of instances and instance types, and the number of components used to construct the GMM.

I. Introduction

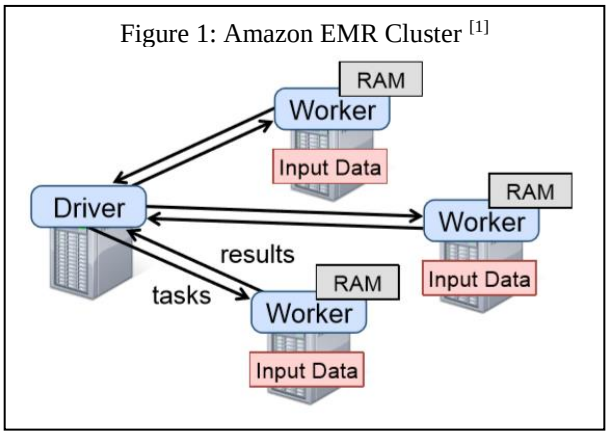
Spark is an open-source distributed system used for processing big data. Based on Hadoop MapReduce, it has been optimized using Directed Acyclic Graphs (DAGs) and actively stores data in memory. Integrated into Spark is a practical Machine Learning Library (MLlib) that makes analyzing data simple and scalable. Included in the library is a clustering algorithm called the Gaussian Mixture Model (GMM).

The GMM is a probabilistic model used to represent normally distributed subpopulations within an overall population. It does not inherently know which subpopulation a data point belongs to and the model assigns the points to various subpopulations. Like K-Means, it is an unsupervised learning model.

The time required to train a GMM with data can vary greatly depending on a variety of factors. With a basic understanding of Spark, we pose research questions investigating how particular aspects of the GMM including instance type, data size, parallelization via partitions, affect the amount of time required to train the model to the data.

An interesting result is the relationship between the instance type used, the number of instances, and the number of partitions the data is divided into—a measure of parallelization. While increasing the number of partitions, decreases the training time, there appears to be a relationship between the number of partitions and the number of instances that is optimal. This too is dependent on the type of instance utilized.

Figure 1: Amazon EMR Cluster [1]



II. Evaluated Systems

A. Overall Architecture

Amazon Elastic MapReduce (EMR) is a system that allows users to work in various distributed frameworks with large amounts of data. It utilizes Amazon Simple Storage Service (S3) and Amazon Elastic Cloud Compute (EC2) instances, allowing for a framework that can dynamically scale as shown in Figure 1. The EC2 instances that will be used in this analysis are the m3.2xlarge and the m2.xlarge.

Apache Spark is one of the frameworks available on Amazon EMR. It is an open-source, distributed processing system commonly used for big data workloads.[4] It also supports applications written in Python that integrate with MLlib via a package in Python 2 called pyspark.

The MLlib library utilizes both a DataFrame-based API, and a Resilient Distributed Dataset (RDD) API. The DataFrame API is the primary API and while the RDD API continues to work, it is deprecated and will no longer receive new features. For the purposes of this analysis the DataFrame API will be used as it is the primary API and DataFrames process significantly faster than RDDs.

B. Highlighted Features

The focus of the analysis is on what factors contribute to a machine learning model training time in Spark, and how can these training times efficiently be reduced. Utilizing two different instance types on Amazon Web Services (AWS), and a different number of instances, can we reduce the time required to train a GMM to data? Another feature that will be used when conducting the analysis is the number of partitions the data is broken into. We will investigate how increasing

or decreasing the number of partitions the data is split into effects the GMM training times.

III. Workload / Evaluation Method

Testing the model training times requires a significant amount of time. Not including the 40-45 minutes required to start an EMR cluster, the 10-20 minutes required to resize the cluster, the time to subset the data, and the time to repartition the data, the total time spent waiting for the GMM to be trained was approximately 16 hours, 50 minutes, and 59 seconds (this is the total time after fitting the model 252 times to get enough data). With unlimited resources and time, many more experiments could be performed, however, the time required to train the model took up most of the time for this analysis.

For more information on commands and code, please see the appendix.

Using Python 2 and the pyspark library, we read the data from S3 into memory as a DataFrame. To prepare the data, it needs to be in a format such that the DataFrame contains an ID column and a features column, where the features are an array of multiple features. A function from pyspark called VectorAssembler() is used to transform the data.

Prior to beginning any analysis, we begin with creating the following research questions, and studies that would be performed to potentially answer them.

Unless explicitly noted, the parameters not mentioned in the research question are assumed to be uniform across each experiment (i.e. when testing the number of partitions, the number of instances, instance type, components used to train, and data size remain the same while the number of partitions changes).

1. **Does decreasing the data size decrease training time?**
    - **a.** 75% of data
    - **b.** 50% of data
    - **c.** 25% of data
    - **d.** Compare a, b, and c with RQ1.b
2. **Does decreasing the number of components used to train the GMM decrease training time?**
    - **a.** 5 components
    - **b.** 7 components
3. **Does increasing the number of instances decrease training time?**
    - **a.** 4 instances
    - **b.** Compare a with RQ1.b
4. **Does increasing the number of partitions decrease training time?**
    - **a.** 1 partition
    - **b.** 2 partitions
5. **Does changing the instance type of the slave instances, not the master, to a smaller instance type on AWS, increase training time?**
    - **a.** 2 instances (m3.2xlarge)
    - **b.** 2 instances (m2.xlarge)
6. **With a smaller instance type, does increasing the number of instances decrease training time?**
    - **a.** 4 instances (m2.xlarge)
    - **b.** Compare a with RQ5.b

After defining the research questions, using python 2, the small astronomy dataset was divided into smaller datasets equal to 75% (subset2), 50% (subset3), and 25% (subset4) of the original data (subset1). It was split randomly, but in such a way that a random seed was used for reproducibility. A loop is written in python 2 to change the dataset size, the number of partitions (powers of 2), and to obtain three training times for the same parameters.

Originally, the large astronomy dataset was read into a DataFrame, however, after attempting to train the model for nearly two hours we determined that more experiments with shorter training times would be more beneficial for this analysis.

IV. Results

The results are saved to a comma-separated value (csv) file where they are read back in after terminating the EMR cluster. The average of the three training times is used for the analysis. First, we will look at the results pertaining to the research questions, then we will take a more wholistic view of the data and visualize the analysis.

RQ1				
Parameters held constant	**Components (Fit # of Clusters): 7** **Instance Type: m3.2xlarge** **# of Instances: 2** **# of Partitions: 2**			
Parameter Changed	**Dataset**			
	100%	75%	50%	25%
Train Time (s)	299	230	157	79

Yes, decreasing the data size decreases training time. This result is intuitive and expected. Less data to process means less processing time when training a model. Refer to Figures 2 and 3.

Figure 2: GMM Training Times (heatmap)

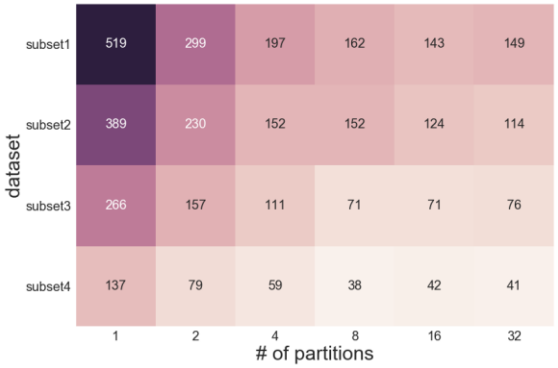
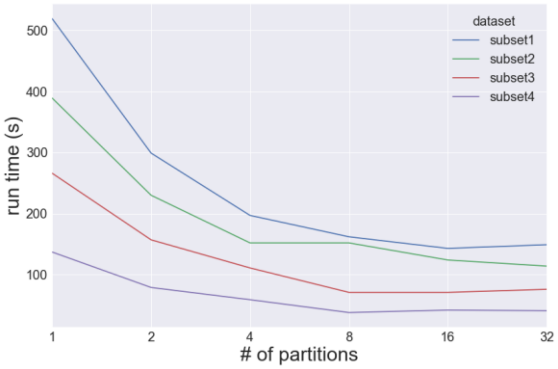


Figure 3: GMM Training Times (line graph)

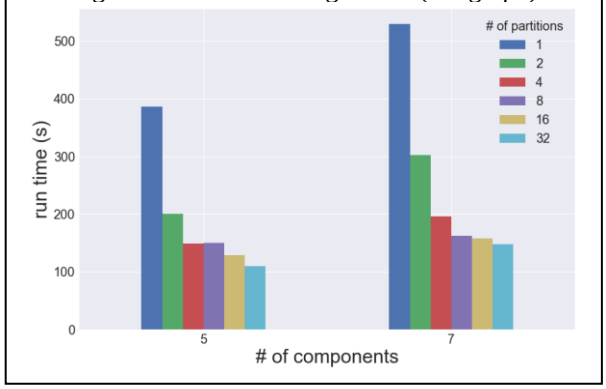


RQ2		
Parameters held constant	**Dataset: 100%** **Instance Type: m3.2xlarge** **# of Instances: 1** **# of Partitions: 2**	
Parameter Changed	**Components (Fit # of Clusters)**	
	5	7
Train Time (s)	201	302

Yes, decreasing the number of components used to train the GMM decreases training time. As the GMM is a clustering model, reducing the number of clusters being identified reduces the number of calculations required to train the model. If we would like to see the extent to which the number of components effects model training times, we would need to run more experiments. Refer to Figure 4.

RQ3		
Parameters held constant	**Components (Fit # of Clusters): 7** **Dataset: 100%** **Instance Type: m3.2xlarge** **# of Partitions: 2**	
Parameter Changed	**# of Instances**	
	2	4
Train Time (s)	299	300

Figure 4: GMM Training Times (bar graph)



No, increasing the number of instances does not decrease training time. Initially, our intuition would suggest that increasing the number of instances should decrease the training time, however, the number of partitions also plays a role when determining the time to train the model. We see in Figure 5, that even when increasing the number of partitions, the training times remain the same. In Figure 5, the number of components is fixed at 7, the 100% data set is used, and the instance type is m3.2xlarge. We believe this has to do with the instance type that is being used and the results from research questions 5 and 6 confirm these assumptions.

We will now proceed and investigate further how partitions and instance types effect the training time of the GMM.

RQ4		
Parameters held constant	**Components (Fit # of Clusters): 7** **Dataset: 100%** **Instance Type: m3.2xlarge** **# of Instances: 2**	
Parameter Changed	**# of Partitions**	
	1	2
Train Time (s)	519	299

Yes, increasing the number of partitions decreases training time. Increasing the number of partitions increases the parallelizability of the data with relation to the number of virtual machines on the Spark cluster. This means more tasks can be performed simultaneously decreasing the training time of the model. Refer to Figures 2 and 3.

RQ5		
Parameters held constant	**Components (Fit # of Clusters): 7** **Dataset: 100%** **# of Instances: 2** **# of Partitions: 2**	
Parameter Changed	**Instance Type**	
	m2.xlarge	m3.2xlarge
Train Time (s)	452	143

Figure 5: GMM Training Times (line graph)

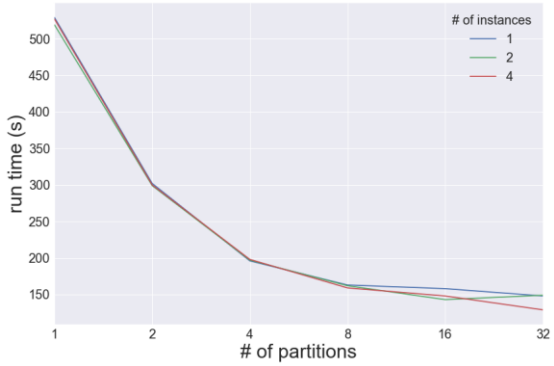


Figure 6: GMM Training Times (bar graph)

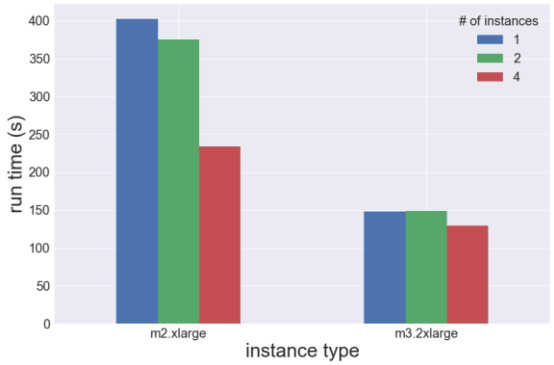


Figure 7: GMM Training Times (heatmap)

Yes, changing the instance type of the slave instances to a smaller instance type increases training time. This is intuitive and expected, but necessary to confirm before proceeding with the final research question. Using an instance with less virtual machines and computing power, trains a model in more time.

RQ6		
Parameters held constant	**Components (Fit # of Clusters): 7** **Dataset: 100%** **Instance Type: m2.xlarge** **# of Partitions: 16**	
Parameter Changed	**# of Instances**	
	2	4
Train Time (s)	452	277

Yes, with a smaller instance type, increasing the number of instances decreases training time. We see that the number of instances does affect the training time when working with instances with less virtual machines and computing power in Figure 6. Figure 7 shows how the number of instances does decrease training time depending on the number of partitions when working with the smaller instance type, m2.xlarge. While this relationship is clear, it is not clear precisely what the relationship between all three variables is. Figure 7 uses the m2.xlarge instance type, 100% of the data, and 7 components. We do see that increasing from 4 to 8 instances results in a slight decrease to training times. A potential explanation is that the overhead cost of distributing the partitions prior to processing outweighs the parallelization.

For the most part, the results are indicative of what we would expect when adjusting the size of the data, the number of partitions, and the number of components. Research question #4 raises some questions about why the training time does not change when more instances are used to train the model, and research questions 5 and 6 provide a possible explanation, but further experiments are required to fully understand the optimization between the number of instances, number of partitions, and instance types.

V. Conclusion

Spark is a powerful distributed processing cluster framework that is highly scalable and relatively simple to use. The size of the data and the number of partitions which it is split into directly affects the model training time; smaller data trains faster than larger data. We see from the analysis using different numbers of partitions that eventually the overhead required to parallelize the data across many virtual machines would most likely reach a point where it is detrimental to the training time and no longer provides any added benefit. Although this was not tested, we can assume that the size of the data training times, continues its trend at any scale Spark is capable of handling.

Although not thoroughly analyzed, we see that the number of components selected when training a model directly impacts the training time. For further analysis it would be interesting to see the trend regarding these training times: Is it a linear relationship? Asymptotic? Parabolic?

The most motivating result for further analysis and experimentation is the relationship between instances and number of partitions (including the type of instance that is being used). It appears that there is a relationship, unfortunately resources did not permit the depth of study required to accurately determine what this relationship might be. Having read some background material and blogs online, it appears that the optimal number of partitions is anywhere from 2-4 times the number of virtual machines on an instance or set of instances. It also looks like the overhead required to parallelize the data is a limiting factor when comparing identical numbers of partitions across differing numbers of instances. The relationship may be more noticeable with a larger dataset that would benefit from more parallelization, or an even smaller instance type such that the number of cores is only 1.

There are many factors that contribute to the training times of models using Spark, and they may vary

depending on the data being used. It is possible to reduce costs while increasing performance of a model's training time by adjusting the instance type, number of instances, and number of partitions. When properly tuned, these training times can be reduced significantly due to the parallelization and cached memory framework of Spark.

REFERENCES

[1] M. Zaharia et al. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In NSDI, 2012.

[2] MLlib: Machine Learning in Apache Spark Xiangrui Meng, Joseph Bradley, Burak Yavuz, Evan Sparks, Shivaram Venkataraman, Davies Liu, Jeremy Freeman, DB Tsai, Manish Amde, Sean Owen, Doris Xin, Reynold Xin, Michael Franklin, Reza Zadeh, Matei Zaharia, Ameet Talwalkar Journal of Machine Learning Research, 17 (34), Apr. 2016.

[3] Spark SQL: Relational Data Processing in Spark Michael Armbrust, Reynold Xin, Yin Huai, Davies Liu, Joseph K. Bradley, Xiangrui Meng, Tomer Kaftan, Michael Franklin, Ali Ghodsi, Matei Zaharia. ACM SIGMOD Conference 2015, May. 2015.

[4] Amazon EMR. (n.d.). Retrieved November 28, 2017, from https://aws.amazon.com/emr

[5] Apache Spark. (n.d.). Retrieved November 28, 2017, from http://docs.aws.amazon.com/emr/latest/ReleaseGuide/emr-spark.html

[6] Machine Learning Library (MLlib) Guide. (n.d.). Retrieved November 28, 2017, from https://spark.apache.org/docs/latest/ml-guide.html

[7] ML Pipelines. (n.d.). Retrieved November 28, 2017, from https://spark.apache.org/docs/latest/ml-pipeline.html#dataframe

[8] Gaussian Mixture Model (GMM). (n.d.). Retrieved November 28, 2017, from https://spark.apache.org/docs/latest/ml-clustering.html#gaussian-mixture-model-gmm