Incremental Learning for Large Scale Classification Systems

Athanasios Alexopoulos, Andreas Kanavos, Konstantinos Giotopoulos, Alaa Mohasseb, Mohamed Bader-El-Den, Athanasios Tsakalidis

Incremental Learning for Large Scale Classification Systems

Athanasios Alexopoulos[1], Andreas Kanavos[1,2], Konstantinos Giotopoulos[2], Alaa Mohasseb[3], Mohamed Bader-El-Den[3] and Athanasios Tsakalidis[1]

1. Computer Engineering and Informatics Department
University of Patras, Patras, Greece
{atalex, kanavos, kgiotop, tsak}@ceid.upatras.gr
2. Technological Educational Institute of Western Greece, Greece
3. School of Computing, University of Portsmouth, Portsmouth, UK
{alaa.mohasseb, mohamed.bader}@port.ac.uk

Abstract. One of the main characteristics of our time is the growth of the data volumes. We collect data literally from everywhere; smart phones, smart devices, social media and the health care system, which defines a small portion of the sources of the big data. The big data growth poses two main difficulties, storing and processing them. For the former, there are certain new technologies that enable us to store large amounts of data in a fast and reliable way. For the latter, new application frameworks have been developed. In this paper, we perform classification analysis using Apache Spark in one real dataset. The classification algorithms that we have used are multiclass, and we are going to examine the effect of the dataset size and input features on the classification results.

Keywords: Apache Spark, Apache MLlib, Big Data, Classification, Computing Performance, DataFrame, Spark SQL

1 Introduction

Nowadays, one of the most frequently used term is Big Data, which is utilized so as to define the increase of the data volumes, as modern datasets are ever expanding both in size and complexity. But with the rapid growth of the stored data, difficulties have appeared [12]. The first obvious problem deals with where to store these huge amounts of data, and then how to process them and retrieve the information they hide. Researchers soon found out that even storing data can easily outperform the capabilities of a single computer, no matter how sophisticated that could be. So they decided not to opt for the popular scale up solution but came up with the idea of scaling out. Instead of continuously updating a single computer, spreading the data or the computation to a cluster of computers interconnected via network, would prove handy.

In order to solve the data storage problem, the idea of NoSQL, which stands for Not Only SQL, arose. Those database systems were originally supported and used by major Internet companies (Google, Amazon, Facebook), since they had

difficulties in dealing with huge amounts of data with the conventional RDBMS solutions [13]. Somewhat similar, researchers cope the need of processing those data, with the use of cluster programming models [1]. In order to unify, and thus simplify, the mode to process big data, Apache Spark [20] project came up. This provides an efficient and scalable analysis of big data accordingly and can support a wide range of workloads as well.

Now that we have discussed how to store and process with big data, one query remains; what the origin of such data is. There is not a single answer to this question since we can collect data from everywhere and have ample forms. Ranging from a post or a tweet in social media, the graph that shows relations between persons who use the specific aspect of the media or share a trend, reviews of products, data collected from smart devices to even more sophisticated data like health records of patients.

In this paper we aim to perform classification analysis in a real dataset, using Apache Spark's MLlib, a machine learning library optimized for distributed systems. We investigate several classification models (Decision Trees, Random Forest, Logistic Regression) and the way they evolve along the size of the dataset.

The remainder of the paper is structured as follows: Section 2 presents the related work and cloud computing methodologies, while Section 3 presents the machine learning (e.g. classification) algorithms used in our proposed system. Section 4 presents the steps of training as well as the classification type, e.g. ternary. Moreover, Section 5 presents the evaluation experiments conducted and the results gathered. Ultimately, Section 6 presents conclusions and draws directions for future work.

2 Related Work

Data mining is an idea that came into existence during the 1990's and its main advantage is that it is suitable for discovering hidden patterns and other useful information from a known dataset. It provides the researcher with the tools to discover correlations that exist and where not obvious at first, and thus makes the task of Decision Making easier [7]. As Fayyad et al stated [4], the knowledge discovery has a wide range of application areas, with marketing, finance (especially investment) and fraud detection defining a portion of them. The process of knowledge discovery is structured in several stages, the first of which is data selection. The preprocessing of data is bound to follow and the transformation of it into the appropriate format with the final stage of Data Mining, in which a suitable algorithm (or technique in general) is applied in order to extract the hidden information, in a form appropriate for future use [18].

There are several tools that encapsulate those principles, with Weka [6] being one of the most widely used and accepted in business and academia. Although its popularity, it has the limitation of the capabilities of a single computer, in terms of memory and computational power. Using the notion of distributed

[1] `https://en.wikipedia.org/wiki/NoSQL`

computation, researchers came up with a distributed framework or Weka, the Distributed Weka Spark [10], which is implemented on top of Apache Spark.

In addition, frameworks like Hadoop, Apache Spark, Apache Storm as well as distributed data storages like HDFS and HBase are increasingly becoming popular, as they are engineered in a way that makes the process of very large amounts of data almost effortless. Such systems are gaining much attention and consecutively libraries (like Apache Spark's MLlib), which make the use of Machine Learning techniques possible in the cloud, are introduced [5].

One of the first widely used framework that supports data processing in a distributed manner across clusters of computers is Apache Hadoop [16], which makes use of the MapReduce paradigm. There were several efforts to use Hadoop for data mining problems, in [19] researchers implemented KD-tree algorithm on Hadoop, in [21] they developed a fast parallel k-means clustering algorithm based on MapReduce whereas in [9] the researchers developed Pegasus, a big graph mining tool built on MapReduce.

In social media analysis, a cloud-based architecture was proposed in [1] where authors aim at creating a sentiment analysis tool for Twitter data based on Apache Spark cloud framework. There, tweets are classified using supervised learning techniques and the classification algorithms are used for ternary classification. In addition, in [17], a survey of machine learning algorithms implemented on Apache Spark over DHT-structures (Distributed Hash Table) is presented; authors experimented across a POS dataset that had been stored in Cassandra with some of the most influential algorithms that have been widely used in the machine learning community. Finally, a novel distributed framework implemented in Hadoop as well as in Spark for exploiting the hashtags and emoticons inside a tweet is introduced in [8]. Moreover, Bloom filters are also utilized in order to increase the performance of the proposed algorithm.

2.1 Distributed Computing

In this section we will briefly discuss the tools we used to perform the analysis. As above mentioned, the classification analysis will be performed with the use of Apache Spark. For clarity reasons though, a brief mentioning of Spark's predecessor, Hadoop MapReduce, is vital and of essential importance.

2.2 MapReduce Model

MapReduce [3] is a programming technique used in distributed systems. Its main advantage being that it is easily scalable over a number of computing nodes. It consists of two main procedures, namely Map() and Reduce(), which is performed after the Map() job. A MapReduce program is executed in three stages the map, shuffle and reduce. In the map stage, the input data is split into a number of chunks and each chunk is sent to a mapper to execute the Map() algorithm. Its result is a set of $< key, value >$ pairs which, during the shuffle stage are grouped by the key, and each key is fed into the corresponding reducer in which they execute the Reduce() function.

2.2.1 Apache Spark Apache Spark[2] [20] was founded in 2009 at the University of California, Berkley. Although it shares the same principles as Hadoop, its philosophy differs. It uses the abstraction of "Resilient Distributed Dataset" (RDD's), which represent a fault-tolerant correlation of elements, distributed across many compute nodes that can be manipulated in parallel. Using them, a wide range of tasks, including SQL, streaming, machine learning and graph processing, in a unified manner, can be captured. Its main advantage over MapReduce paradigm is that we don't have to flush the intermediate data to the disk, just to read them at the reduce stage, since it can perform iterative computations in memory, which can have a positive impact on the performance [15].

2.2.2 MLlib Spark ships with MLlib[3], a distributed machine learning library [12]. It consists of implementations of common machine learning algorithms, that can be used with scalable environments, for several types of analysis including classification, regression, clustering and collaborative filtering. MLlib also includes Java, Scala and Python APIs.

3 Classification Models

In this section we will discuss the classification methods that we utilized. We examined multiclass classification, while the focus of the research concerned the way that the size of the dataset affected the computation time needed to perform each as well as the way the metrics evolved. As a quick reminder, classification is an instance of supervised learning, and its purpose is, based on previous knowledge (set of observations and the output class of each observation) to identify in which output class a new observation belongs.

3.1 Decision Trees

Decision Trees is a classification algorithm, which can be represented in a tree form [11]. The nodes of the tree represent the feature of the dataset, and depending on each feature's value we navigate through the tree structure until we reach a leaf, which represents the output classes. The root of the tree is the feature that best divides the training data, by minimizing a loss function. The procedure in following is recursively executed to each partition of the data to form the subtrees, until the training data is divided into subsets of the same class.

3.2 Random Forest

Random Forest is a generalization of a Decision Tree classifier [2]. It consists of a set of Decision Tree classifiers, so in order to classify for a new input we

[2] http://spark.apache.org/
[3] http://spark.apache.org/mllib/

insert it in each tree of the forest, ending in the "vote" of an output class. The class that will receive the most votes, is the output class the Random Forest will return. To construct each tree, we sample with replacement a number of cases from the original data, equal to the number of trees our forest has, and then we use a subset of the features of that specific selection in order for the size level of tree to be increased.

3.3 Logistic Regression

Logistic Regression is a regression model that can be used when the dependent value (output class) is categorical. It uses a logistic function to express the relationship between the dependent value and the independent (features of each class). Apache Spark supports both binomial and multinomial logistic regression, in which case, assuming we have K output classes, one class is chosen as pivot, $K-1$ models are created and the final result is the class with the largest probability among the $K-1$ models.

3.4 One-Vs-Rest

The One-Vs-Rest (or One-Vs-All) classifier, uses a base classifier, that can efficiently perform binary classification and trains a single classifier per output class (considering the portion of the data that belong to that class as positive and the rest as negative). In order to determine the label of the output class, it uses the classifier with the highest confidence score.

3.5 Multilayer Perceptron

A Multilayer Perceptron is an artificial neuron network model used to map a set of input data (input classes) to a set of output classes. It consists of layers of nodes fully connected, with a certain weight, to the ones of the next level. Using back-propagation, a technique that changes the connection weights of each node to the ones of the next layer in order to minimize the output error, we can train the Multilayer Perceptron for a given dataset.

4 Implementation

Our approach follows the proposal of [4], as presented in Section 2. Firstly, we need to discuss the framework on which the computation took place.

The overall architecture of the proposed system is depicted in Figure 1 taking into account the corresponding modules of our approach. Initially, a preprocessing step, as shown in the following subsection, is utilized and afterwards the classifiers for prediction scope are employed.

The creators of Apache Spark have also founded Databricks that supplies researchers with a web based platform in which they can store and analyse their data with Spark. It offers researchers a mini cluster with 6 Gb of RAM for their

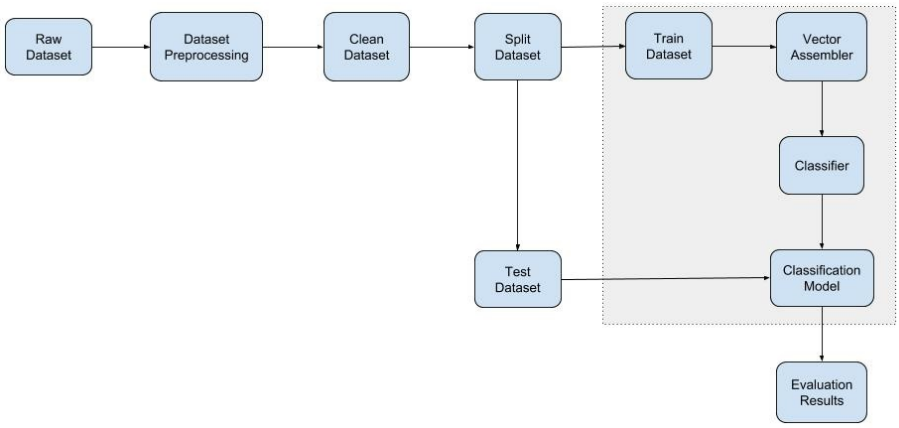

Fig. 1. Proposed System Architecture

analysis and also cloud storage. As programming language, Python (PySpark) was chosen.

The realistic dataset is the PAMAP dataset [14], also provided by UCL [4]. It contains 2.872.533 rows of data from 8 subjects and 12 different activities. The devices that were used to obtain those measurements were inertial measurement units in the ankle, chest and hand of the subject as well as a heart rate monitor. Since the dataset is realistic, some of the data were missing. There were two main reasons for that, data dropping due to using wireless sensors and problematic hardware setup. Also, there was another issue with the equipment used; the sampling frequency of the heart rate monitor was lower than the inertial measurement units. All those missing values were indicated as NaN.

Our first attempt to clean up the data was by withdrawing all the rows that contained at least one NaN value. As a result we had to withdraw 2.610.265 rows of data. To overcome this obstacle, we replaced all the intermediate NaN values of the heart rate monitor with the last known value. After that replacement, the dataset contained only 13.141 rows indicated as NaN, which we ignored in the final analysis.

In order to perform the analyses, we split the original data into segments of 5.000, 10.000, 25.000, 50.000, 75.000, 100.000, 200.000 rows. But, in order for the results to be comparable in each case, we had to keep the ratio of the number of instances that belong to a specific class to the total number of instances, stable. So, we calculated the fraction of each class in the original dataset, and at the examined cases we assured that each class had the same fraction of observations.

[4] `http://archive.ics.uci.edu/ml/datasets/PAMAP2+Physical+Activity+Monitoring`

4.1 Analysis Cases

For each segment of both datasets we performed a series of classification analyses.

- Decision Trees, optimized regarding the depth of the tree, with values ranging between 2 and 30.
- Random Forest, optimized regarding the number of trees in the forest, with values ranging between 1 and 60, with a step of two and maximum tree height equal to ten.
- Logistic Regression, multinomial.
- Multilayer Perceptron with two, three and four levels of hidden layers. For the case of two levels, the numbers of nodes were 15-14, 20-15, 25-20 respectively, whereas for case of three levels the numbers of nodes were 25-20-15, 35-25-20, 40-30-20 and for the case of four levels 30-25-20-15, 35-30-25-20 and 40-35-30-25. Moreover, the number of epochs was set equal to 1000.

5 Evaluation

The results of our work are presented in the following Tables 1 to 5. Recall, Precision and F-Measure are used as the evaluation metrics of the different algorithms. Also, the time that was needed to train each classifier is presented, as well as information about the best settings.

In PAMAP Dataset, a series of classification analyses are conducted. In Table 1, the results of the Decision Tree classifier are presented. As the dataset increased in size, we kept getting higher scores, as expected. Worth noting is the fact that in this case, the height of the tree had the highest possible value in all cases, but its complexity increased since nodes kept increasing in number. Also, the computational time needed, as expected, kept rising with the size of the dataset, but not in a linear manner; when the size of the dataset increased by $40 times$ (from 5.000 rows to 200.000), the time needed nearly doubled.

Table 1. Decision Tree

Dataset Size	Precision	Recall	F-Measure	Time	Depth	Number of Nodes
5.000	0.862	0.859	0.860	0:15:25	17	731
10.000	0.923	0.923	0.923	0:17:21	16	1007
25.000	0.954	0.954	0.954	0:21:09	18	1625
50.000	0.966	0.966	0.966	0:24:44	25	2441
75.000	0.968	0.968	0.968	0:39:55	27	3363
100.000	0.981	0.981	0.981	0:41:43	27	3139
200.000	0.989	0.989	0.989	0:57:26	30	4083

Next, the Random Forest classifier (Table 2) is discussed. Again as the dataset size grew bigger, so did the metrics' scores. The only exception is the case of

the 10.000 rows in which we had the best observed scores, but this might be caused by the split of the dataset. Overall, this classifier produced lower metrics compared with the Decision Tree classifier; the best F-Measure score is 76.7% while Decision Tree's was 90.0%. Furthermore, the training time of this classifier is overall higher than the Decision Tree one. When the size of the dataset increased 40 times, it needed 4 times more for the classifier to be trained.

Table 2. Random Forest

Dataset Size	Precision	Recall	F-Measure	Time	Number of Trees	Number of Nodes
5.000	0.907	0.907	0.907	0:29:27	60	28908
10.000	0.927	0.925	0.926	0:36:01	50	28522
25.000	0.945	0.945	0.945	0:49:12	44	31618
50.000	0.953	0.953	0.953	1:04:11	42	35836
75.000	0.950	0.949	0.949	2:01:05	44	38600
100.000	0.958	0.958	0.958	2:28:11	48	45134
200.000	0.951	0.950	0.950	3:42:04	56	56274

The next classifier is the Multinomial Logistic Regression (Table 3), where there was no parameter optimization performed. With few exceptions, we can argue that as the dataset size increases, so do the metrics, although some fluctuation exists. The metrics' score is higher than the Random Forest classifier, but did not perform as well as Decision Trees. As for the scaling of time regarding the size of the dataset, for 20 times bigger dataset, from 10.000 to 200.000 rows, the classifier's training time increased by 7.5 times.

Table 3. Logistic Regression

Dataset Size	Precision	Recall	F-Measure	Time
5.000	0.858	0.861	0.860	0:00:26
10.000	0.861	0.862	0.861	0:00:26
25.000	0.814	0.818	0.816	0:00:32
50.000	0.803	0.807	0.804	0:00:42
75.000	0.802	0.806	0.803	0:01:00
100.000	0.825	0.828	0.826	0:01:14
200.000	0.800	0.804	0.802	0:01:56

The last classifier examined is One-Vs-All (Table 4). As the base classifier, we have chosen Logistic Regression. Its performance was rather poor, both in time needed to train and in the metrics' score. Examining again F-Measure, it achieved 79.5% while the Logistic Regression achieved 80.8%. Considering time scaling, with regards to the dataset size, for 20 times bigger dataset, the classifier's training time has been increased by 3.23 times, and in every case the training was 7 to 11 times higher.

Table 4. One-vs-All

Dataset Size	Precision	Recall	F-Measure	Time
5.000	0.755	0.755	0.749	0:02:03
10.000	0.789	0.791	0.789	0:02:15
25.000	0.788	0.792	0.789	0:02:29
50.000	0.781	0.784	0.781	0:02:56
75.000	0.770	0.773	0.771	0:04:56
100.000	0.796	0.798	0.796	0:07:17
200.000	0.774	0.778	0.775	0:11:16

Table 5. Multilayer Perceptron

Dataset Size	Precision	Recall	F-Measure	Time	Best Layer
5.000	0.639	0.646	0.641	0:09:20	[31, 35, 25, 20, 12]
10.000	0.691	0.708	0.697	0:14:27	[31, 35, 25, 20, 12]
25.000	0.720	0.719	0.719	0:32:22	[31, 40, 30, 20, 12]
50.000	0.707	0.732	0.721	1:01:55	[31, 40, 30, 20, 12]
75.000	0.724	0.738	0.727	1:30:39	[31, 40, 30, 20, 12]
100.000	0.755	0.776	0.759	1:55:59	[31, 35, 25, 20, 12]
200.000	0.775	0.778	0.775	3:59:41	[31, 35, 25, 20, 12]

Overall, it can be argued that the best choice in PAMAP dataset is the Decision Tree classifier, since it performed better in the metric scores from all other classifiers and its training time scaled well regarding the dataset size.

6 Conclusions and Future Work

In our work we delved into the performance of various classification algorithms in a distributed environment. Each classification method had strengths and limitations, depending on the dataset. The Decision Tree classifier outperformed all the other classification methods, which was a pleasant surprise.

As for future work, more datasets with the same classification methods to establish a better understanding, will be checked. Of course, the cluster on which we run the classification methods is another key aspect of the cloud computing in general, so checking the same or similar algorithms and dataset on different clusters may prove fundamental and reveal well-hidden secrets.

References

1. Baltas, A., Kanavos, A., Tsakalidis, A.: An apache spark implementation for sentiment analysis on twitter data. In: International Workshop on Algorithmic Aspects of Cloud Computing (ALGOCLOUD). pp. 15–25 (2016)
2. Breiman, L.: Random forests. Machine Learning 45(1), 5–32 (2001)
3. Dean, J., Ghemawat, S.: Mapreduce: simplified data processing on large clusters. Commun. ACM 51(1), 107–113 (2008)

4. Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P.: From data mining to knowledge discovery in databases. AI Magazine 17(3), 37–54 (1996)
5. Fovet, B., Gasque, M., Horel, F., Hourquebie, H., Lahjouji, A., Seddiki, A.: Machine learning and data mining with apache spark
6. Hall, M.A., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA data mining software: an update. SIGKDD Explorations 11(1), 10–18 (2009)
7. Hand, D.J., Mannila, H., Smyth, P.: Principles of Data Mining. MIT Press (2001)
8. Kanavos, A., Nodarakis, N., Sioutas, S., Tsakalidis, A., Tsolis, D., Tzimas, G.: Large scale implementations for twitter sentiment classification. Algorithms 10(1), 33 (2017)
9. Kang, U., Faloutsos, C.: Big graph mining: algorithms and discoveries. SIGKDD Explorations 14(2), 29–36 (2012)
10. Koliopoulos, A., Yiapanis, P., Tekiner, F., Nenadic, G., Keane, J.A.: A parallel distributed weka framework for big data mining using spark. In: IEEE International Congress on Big Data. pp. 9–16 (2015)
11. Kotsiantis, S.B.: Supervised machine learning: A review of classification techniques. Informatica (Slovenia) 31(3), 249–268 (2007)
12. Meng, X., Bradley, J.K., Yavuz, B., Sparks, E.R., Venkataraman, S., Liu, D., Freeman, J., Tsai, D.B., Amde, M., Owen, S., Xin, D., Xin, R., Franklin, M.J., Zadeh, R., Zaharia, M., Talwalkar, A.: Mllib: Machine learning in apache spark. The Journal of Machine Learning Research 17(1), 1235–1241 (2016)
13. Moniruzzaman, A.B.M., Hossain, S.A.: Nosql database: New era of databases for big data analytics - classification, characteristics and comparison. International Journal of Database Theory and Application 6(4), 1–14 (2013)
14. Reiss, A., Stricker, D.: Introducing a new benchmarked dataset for activity monitoring. In: International Symposium on Wearable Computers (ISWC). pp. 108–109 (2012)
15. Shi, J., Qiu, Y., Minhas, U.F., Jiao, L., Wang, C., Reinwald, B., Özcan, F.: Clash of the titans: Mapreduce vs. spark for large scale data analytics. PVLDB 8(13), 2110–2121 (2015)
16. Shvachko, K., Kuang, H., Radia, S., Chansler, R.: The hadoop distributed file system. In: IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST). pp. 1–10 (2010)
17. Sioutas, S., Mylonas, P., Panaretos, A., Gerolymatos, P., Vogiatzis, D., Karavaras, E., Spitieris, T., Kanavos, A.: Survey of machine learning algorithms on spark over dht-based structures. In: International Workshop on Algorithmic Aspects of Cloud Computing (ALGOCLOUD). pp. 146–156 (2016)
18. Witten, I.H., Eibe, F., Hall, M.A., Pal, C.J.: Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann (2016)
19. Yang, L., Shi, Z.: An efficient data mining framework on hadoop using java persistence API. In: 10th IEEE International Conference on Computer and Information Technology (CIT). pp. 203–209 (2010)
20. Zaharia, M., Xin, R.S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., Venkataraman, S., Franklin, M.J., Ghodsi, A., Gonzalez, J., Shenker, S., Stoica, I.: Apache spark: a unified engine for big data processing. Commun. ACM 59(11), 56–65 (2016)
21. Zhao, W., Ma, H., He, Q.: Parallel K-means clustering based on mapreduce. In: First International Conference on Cloud Computing (CloudCom). pp. 674–679 (2009)