Feedback - Study and Improvement of the Random Forest of the Mahout library in the context of marketing data of Orange

C. Thao[*,**], N. Voisine[*], V. Lemaire[*], R. Trinquart[*]

[*] Orange Labs, 2 avenue Pierre Marzin, 22300 Lannion, France
[**]Predicsis, 5 rue de Broglie, 22300 Lannion, France

Abstract. In the realm of Big Data systems, Hadoop has emerged as one of the most popular systems and a very diverse ecosystem has grown around it, meeting all kinds of functional and technical needs. One niche that should have been a place of choice in this ecosystem is data analytics: first because getting value out of large datasets requires efficient Machine Learning (ML) algorithms, second because large clusters with abundant CPUs resources seem like appropriate playfields for ML algorithms which are often very resource-intensive computing tasks. Unfortunately among the myriad of open source projects, there are very few data analytics tools that have been ported to the Hadoop framework. Apache Mahout stands out among those rare initiatives: this project is mainly known for its recommendation application, but it also offers a warehouse of ML algorithms, advertised to run on Map/Reduce. We did investigate the twenty algorithms proposed within Mahout and in this report we focus on the most promising one: the Random Forest implementation. Relying on extensive tests, including specific marketing data from Orange, we provide an in-depth feedback on the use of this tool, both from a practical and theoretical perspective, and we suggest several improvements.

1 Introduction

The decreasing cost of data storage has led to the accumulation of large and complex datasets, which are widely seen as new opportunities for business. Orange - a multinational telecommunications corporation - has to analyze the data of its network to improve profitability and create new services. To give a sense of scale, in order to increase customer satisfaction on services, Orange has to analyze Quality of Services (QoS) and Quality of Experience (QoE) indicators for its 150 million of mobile customers. Those QoS and QoE indicators result from the combination of different data sources (network probe, SI). The main purpose consists in real time detection or prediction of QoS or QoE. This would allow Orange either to improve quality of network or to provide new services based on QoE. Therefore applying data mining techniques to these vast amounts of data is crucial. This raises numerous issues such as scalability of data mining algorithms, automation of the data mining process and control of over-fitting.

The scalability issue is usually the first one that people have in mind with big data. The availability of efficient computing environments such as Hadoop (Had) clusters with the map-reduce framework is often considered as the solution to the scalability issue. One open source project, named Mahout (Mah), actually claims to provide implementation of several learning algorithm that not only get data out of Big Data repository but actually run on a Hadoop cluster, thus taking advantage of the parallelization. This project has gained increasing attention after some companies reported using it with successful results. To be more precise, Mahout gathers libraries for both supervised and unsupervised learning. The success stories about Mahout all relate to unsupervised learning libraries. On the contrary, there is very little said about the supervised libraries.

Still, these computing environments have been originally designed for the search engine tasks, based on indexing billions of documents; they are efficient for some families of tasks but cannot be considered as universal models for parallel computing. Among the data mining tasks, the deployment phase is data intensive and is likely to fit well into the map-reduce framework. On the contrary, the modeling phase is CPU intensive, and exploiting efficiently the resources of a Hadoop cluster is an open problem.

In a previous study (Dream, 2013) we have observed that Random Forrest (RF) is (one of) the best method of Mahout, both in terms of quality of the models and in terms of scalability since this algorithm is natively a parallel process. The aim of our work is to study and improve if necessary the RF algorithm of Mahout for use on Hadoop cluster. In the first part of this report we show a preliminary study of the Random Forest of the Mahout library.

In the second part we suggest several improvements of the initial library. In a third part we propose a new decision tree algorithm to improve performance and reduce over-fitting. We provide results on both academic datasets and Orange data before coming to a conclusion.

2 Preliminary study

In the first part of our study we investigated the behavior of the Random Forests (RF) of the Mahout library as it is packaged in its latest available version (0.9). This section begins with a reminder of the RF algorithm. Then the second subsection provides the experimental conditions we used for all the experiments of these report. In the third subsection, we present the obtained results, which leads to a discussion about potential issues and solutions.

2.1 Random Forest

The "Random Forests" is a supervised classifier introduced by Leo Breiman (Breiman, 2001). It is related to the decision tree approach but the predictive model no longer consists in a single tree: instead it gathers a multitude of trees. Random forests are a combination of tree predictors such that each tree depends on the values of a random vector set sampled independently and with the same distribution for all trees in the forest. The generalization error for forests converges to a limit as the number of trees in the forest becomes large. The generalization error of a forest of tree classifiers depends on the strength of the individual trees in the forest and the correlation between them. Using a random selection of features to split each node yields error rates that compare favorably to Adaboost (Freund and Schapire, 1996), but are more robust with respect to noise. Internal estimates monitor error, strength, and correlation and these are used to show the response to increasing the number of features used in the splitting. Internal estimates are also used to measure variable importance. In the Random Forests, it is not necessary to use a set of validation data. Indeed, during the bagging, many instances are not used for the construction of a tree. Each classifier learns only a portion of the data. Unused data are called Out-Of-Bag. They provide a good way to estimate the generalization performance of the classifier.

2.2 Experimental conditions

2.2.1 Industrial Context

In this report we are focusing on the behavior of Mahout when dealing with data which have the characteristics of “Orange Data", *i. e.*:

— Data constraints: (i) Heterogeneous, (ii) Missing values, (iii) Two or Multiple classes, (iv) Heavily unbalanced distributions; with
— Many scales: (i) Tens to millions of instances, (ii) Tens to tens of thousands of variables;
— Many types of data: (i) Numerical, (ii) Categorical, (iii) Text, (iv) Image.

We restrict the study to the task of supervised learning, *id est* classification problem such as churn detection, appetency prediction ... And among potential datasets for evaluation, we have a strong interest in those closer to marketing problems.

2.2.2 Parameters that influence the behavior of Mahout

When installing the Mahout library on a Hadoop platform there are several parameters that will influence the performances obtained using the Mahout’s RF: the number of machines (nodes) in the cluster and their characteristics, the configuration of the distributed filesystem HDFS, especially regarding the way blocks are created (size and replication factor), the split criterion used in the RF algorithm, the number of trees in the Forests, the way to combine the trees. The following paragraphs describe each of these points. We also indicate at the end of this section the baseline we used to check the validity of the results obtained with the RF.

Cluster: All tests are performed on a small exploration Hadoop[1] cluster (Had) of Orange Labs which has the following properties: (i) 6 nodes with: 2 Intel(R) Xeon(R) CPU E5-2407 0 @ 2.20GHz, total cpu cores per node: 8, 32 Go of RAM, Hadoop version: Cloudera CDH4 (Hadoop 0.20), Mahout 0.7 (as packaged within Cloudera CDH4.2).

Data Nodes & Block Size: The primary components of a Hadoop platform are HDFS and MapReduce. HDFS is a distributed system designed to store files of very large volumes of data on a large number of machines, whereas MapReduce is a framework for distributing processing over large files. When those two components are combined on the same set of machines (cluster nodes), the resulting platform acts as a single system, providing high availability, load

1. Hadoop (High-availability distributed object-oriented platform) is a distributed system that addresses the issues of “big data”. Hadoop uses a distributed storage system, which is called HDFS (Hadoop Distributed File System) and incorporates analysis systems like MapReduce, Mahout or Spark. Hadoop allows splitting data and the execution of analysis on parallel processing.

balancing, and parallel processing. With HDFS, any large data file is decomposed into blocks and those are distributed across the nodes of the cluster. Based on HDFS blocks, the Map and Reduce functions can be distributed across the cluster and perform on subsets of large data sets, allowing better scalability. Unlike a conventional storage system, where blocks are a matter of kilo-octets, the block size is set by default to 64 MB. This default value can be adjusted by the cluster administrator and even changed on the fly by a user for any specific fly and/or processing. Typical block sizes are 128MB, 256 MB, 512 MB or 1 GB. Our cluster was configured with a default size of 128MB.

Note: The behavior of Random Forests in Mahout is somewhat different from the original version of Breiman and therefore his theory. First, the bootstrap samples are not done on the entire data set. Because the data are divided into the data nodes (and thus the mappers), each mapper realizes its own bootstrap. The mappers therefore realize a training of several trees in the Forest. The algorithm of the Random Forests of Breiman is therefore partly “fulfilled". Each mapper performs a part of the forest but on data which are not a “bag” in the sense of a bagging. Then the forest of each mapper are combined to realize the “global" forest.

Split criterion: The RF in Mahout uses by default non-binary trees. The information gain (Quinlan, 1986) is the default split criterion used. When a variable is chosen to perform the split: (i) for numeric attributes two leaves are elaborated after a node, (ii) for categorical variables Q leaves are elaborated after a node where Q is the number of modalities of the chosen variable.

Note: It is important to note that Mahout does not handle missing values. So in a first part of our experiment we replaced all missing values by -9999. For a categorical variable, substituting -9999 to missing values entails that we consider the missing value as information. For a numeric variable, the aim is to put a lower value than any other value.

Number of trees: The number of trees influences the AUC value: increasing the number of trees increases the chances of having discriminating trees. This improvement is especially observable in the beginning of a curve when plotting the AUC versus the number of trees in the Forest. For this database, KDD small upselling, the asymptotic good value is close to 4000 but the gain in performances after 1000 is low. The number of trees has also a big influence on the time which will be necessary to deploy the model. In the experiments presented below in this paper we set the number of trees to 1000 (therefore to $1000/M$ for each mapper where M is the number of mappers).

Combining the trees: After a large number of trees is generated, they vote for the most popular class.

Our baseline results Orange has developed a powerful software named Khiops `www.khiops.com`) which is able to works on very large dataset (Guyon et al., 2010) or on multi table dataset (Boullé, 2014)). We used this software as a baseline in its “standard version": applied on a single database (not on a distributed database) to evaluate the results obtained with the RF from Mahout. The classifier is an Averaging of Selective Naive Bayes described in (Boullé, 2007)

2.3 Datasets

We used 3 types of data (8 datasets) which are:

— Adult: data belonging to the UCI Machine Learning Repository (Bache and Lichman, 2013);

— OCR of the Pascal Large Scale Learning Challenge (Sonnenburg et al., 2008);

— KDD (Large and Small) marketing data provided by Orange for the challenge KDD 2009 (Orange, 2009; Guyon et al., 2010)

Their characteristics are given in Table 1. The criteria used to evaluate the obtained results is the Area Under the ROC curve (AUC (Fawcett, 2006)). As the name suggests, there is randomness in the algorithm of the Random Forests. We have therefore performed ten training. In the remaining of this paper the Train AUC and Test AUC presented in Tables will be the average AUC over the ten experiments.

Dataset	Ni	Nn	Nc	C	P	PCTrain	PCTest	#Mappers
Adult	48 842	6	8	2	76.17%	70%	30%	8
OCR	3 500 000	1156	-	2	50%	60%	40%	71
KDD Large Upselling	50 000	14740	260	2	7.4%	50%	50%	27
KDD Large Churn	50 000	14740	260	2	7.3%	50%	50%	27
KDD Large Appetency	50 000	14740	260	2	1.8%	50%	50%	27
KDD Small Upselling	50 000	190	40	2	7.4%	50%	50%	20
KDD Small Churn	50 000	190	40	2	7.3%	50%	50%	20
KDD Small Appetency	50 000	190	40	2	1.8%	50%	50%	20

TAB. 1 – *Dataset used: Ni=Number of Instances, Nn=Number of numerical variables , Nc=Number of Categorical variabkes, C=Number of Classes, P:Percentage of the target class, PCTrain: Percentage used for training, PCTest: Percentage used for the test.*

2.4 Results with the “standard” library

We present the results that we obtained with the Random Forest of Mahout when using them with the installation by default of the library: (i) the split criterion is the information gain, (ii) the missing value are dealt with according to the description given in the previous section, (iii) the number of trees is 1000, (iv) the number of mappers is indicated in the last column of the Table 1, (v) the predicted class is the most popular predicted class in the forest.

The table 2 shows (columns 1 and 2) the performances obtained on the Test AUC and the ratio Train AUC / Test AUC (column 3) to give an element of robustness (a value above 1 indicates an overfitting).The key points of these results are (i) RF exhibit poor results on the small dataset but have good performance on the largest dataset OCR (the result seems to be known by the machine learning community (see (Debreuve) slide 17), (ii) the results are not bad but lower than results of the literature on these datasets, (iii) the results are lower (except for OCR) than those obtained by Khiops, (iv) a “large” overfitting exists on many of the dataset.

We analyze in deep these results to understand where come from the problems. One of them comes from the split criterion - the information gain - and another one is the way to deal with the missing values. For the first one the information gain is biased when there is categorical variables which have a lot of modalities (Harris, 2002). In this case the individual trees are less deep and wider. For the second one the initial way to replace the missing value is not appropriate and has to be changed to obtained better results. However the Hadoop framework allows decreasing a lot the training time as showed in table 3

Dataset	Khiops	Mahout	Robustness Mahout
Adult	0.925	0.910	1.02
KDD Small Upselling	0.872	0.648	1.45
KDD Small Churn	0.731	0.612	1.49
KDD Small Appetency	0.814	0.682	1.44
OCR	0.830	0.953	1.00
KDD Large Upselling	0.897	0.877	1.07
KDD Large Churn	0.743	0.681	1.26
KDD Large Appetency	0.852	0.735	1.30

TAB. 2 – *First result with the "standard RF" in Mahout*

	OCR	KDDSmall Upselling	KDDLarge Upselling
Khiops	4h55m	28s	45min
Mahout	3m44s	39s	1m9s

TAB. 3 – *Training time for Khiops and Mahout*

2.5 First Ideas to circumvent the observed problems

Faced with the results obtained in the previous section we try to apply several patches to Mahout's RF in order to improve the performance of the library:
— Mahout "2": where we change the Information Gain to the Gain Ratio;
— Mahout "3": where the missing values of numerical variables are replaced by the corresponding median value (Breiman).

Dataset	Khiops	Mahout Default	Mahout 2	Mahout 3
Adult	0.925	0.910	0.919	0.917
KDD Small Upselling	0.872	0.648	0.690	0.778
KDD Small Churn	0.731	0.612	0.596	0.621
KDD Small Appetency	0.814	0.682	0.667	0.694
OCR	0.830	0.953	0.946	0.953
KDD Large Upselling	0.897	0.877	0.705	0.873
KDD Large Churn	0.743	0.681	0.503	0.681
KDD Large Appetency	0.852	0.735	0.597	0.763

TAB. 4 – *Result with the "modified RF" in Mahout. Mahout 2: Mahout with Gain Ratio, Mahout 3: Mahout with information Gain and missing values replaced by the median value.*

The results obtained for the Test AUC with these two modified version are presented in Table 4. Only the Mahout 3 version has better results than the standard version. But this improvement could be costly since the median value of all the numerical variables is required and because the median value has to be computed on the complete dataset (which is split in different nodes). We also test others modification (not reported here) for the unbalanced database as the use of square root in the computation of the information gain ((Flach, 2012), Section "Sensitivity to skewed class distributions" page 143)) ... but without improvement of the results. Faced with these results and still wishing for more robust performances, we turn to the MODL approach which is known to be robust (Boullé, 2006, 2005).

3 Adding regularization in Random Forest

In this section we show two modifications of decision tree used by Mahout Random Forest to increase classification performance and reduce over-fitting. The first one relies on the MODL approach developed for decision tree successfully (Voisine et al., 2010). It consists in changing the splitting method for numerical variable and introduces a grouping method for categorical variables. The second improvement consists in changing the voting method of Mahout RF. By default Mahout uses the majority voting method to estimate class probabilities. We propose to estimate class probabilities by averaging probability estimates in each leaf of all decision trees.

3.1 Modifications

Adding regularization: We describe new splitting and grouping methods for two classes learning classification task. The splitting method used MODL discretization criteria based on MDL approaches. The MODL discretization method for supervised classification provides the most probable discretization given the data. Extensive comparative experiments report high performance (Boullé, 2006). The case of value grouping of categorical variables is treated in (Boullé, 2005) using a similar approach.

A Bayesian approach is applied to select the best discretization model, which is found by maximizing the probability $p(Model|Data)$ of the model given the data. Using the Bayes rule and since the probability $p(Data)$ is constant under varying the model, this is equivalent to maximizing $p(Model)p(Data|Model)$.

We decided in this study to elaborate RF with binary decision tree for classification problem of two classes which are the more present in the Orange problem. Therefore in each node the split criterion will be dedicated to find the best split in two intervals for numerical variables or the best grouping in two groups for categorical variables. This setting allows having fast algorithm (not detailed in this report due to place consideration).

Let N be the number of instances of a leaf. N_i denotes the number of instances in the interval i and N_{ij} the number of instances of output value j in the interval i. In our context, the number of instances N and the number of classes J are supposed to be known. The number of interval is fixed by two.

For numerical variables (Boullé, 2006) the evaluation criterion when the number of intervals and the number of classes are both equal to 2 and the number of classes is: $\log 2 + \log(N+1) + \log(N_1+1) + \log(N_2+1) + \sum_{i=1}^{2} \log \frac{N_i!}{N_{i1}!N_{i2}!}$. Once the evaluation criterion is established, the problem is to design a search algorithm in order to find a discretization model that minimizes the criterion. In Boullé (2006), a standard greedy bottom-up heuristic is used to find a good discretization.

For categorical variables Boullé (2005) the evaluation criterion when the numberof groups is equal to two groups is: $V \log 2 + \log(1 - \frac{1}{2^{V-1}}) + \log(N+1) + \log(N_1+1) + \log(N_2+1) + \sum_{i=1}^{2} \log \frac{N_i!}{N_{i1}!N_{i2}!}$, where V is the number value of a catagorical variable.

Voting: Usually the predicted class is based on the votes of the trees for the most popular class. We change this rule to perform averaging probability estimates in each leaf of all decision trees, such as: $P(C_J|X) = \text{argmax}_J \frac{1}{T}\sum_{t=1}^{T} P(C_J^t)$, where $P(C_J^t)$ is the terminal leave of the tree t. In this case the predicted class will be the one with highest probability and the confidence probability will the corresponding averaging probability. The estimated confidence will allow a better estimation for the computation of the AUC.

3.2 Results

Dataset	Khiops	Mahout Default	Mahout 2: Gain Ratio	Mahout 3: IG + Media	Mahout MODL	Mahout MODL + new voting
Adult	0.925	0.910	0.919	0.917	0.911	0.916
KDD Small Upselling	0.872	0.648	0.690	0.778	0.800	0.820
KDD Small Churn	0.731	0.612	0.596	0.621	0.619	0.675
KDD Small Appetency	0.814	0.682	0.667	0.694	0.555	0.754
OCR	0.830	0.953	0.946	0.953	0.947	0.948
KDD Large Upselling	0.897	0.877	0.705	0.873	0.866	0.875
KDD Large Churn	0.743	0.681	0.503	0.681	0.628	0.667
KDD Large Appetency	0.852	0.735	0.597	0.763	0.686	0.725

TAB. 5 – *Result of the test AUC with the "modified RF" in Mahout. Mahout 2: Information Gain replaced with the Gain Ratio, Mahout 3: Mahout with information Gain and missing values replaced by the median value, Mahout MODL with prediction based on the most popular class and Mahout MODL with prediction based on averaging probability estimates.*

Dataset	Number of Nodes		Robustness	
	Mahout Default	Mahout MODL	Mahout MODL	Mahout MODL
Adult	1763	60	1.02	1.00
KDD Small Upselling	9895	5	1.45	1.06
KDD Small Churn	9954	3	1.49	1.12
KDD Small Appetency	4013	2	1.44	1.15
OCR	3396	969	1.00	1.00
KDD Large Upselling	1658	13	1.07	1.02
KDD Large Churn	1864	8	1.26	1.10
KDD Large Appetency	688	5	1.30	0.94

TAB. 6 – *Robutness and Model size of the different RF (* mean number per tree)*

We compare the new RF which used the MODL approach as split criterion to the results obtained in the previous section. The results of the "Mahout MODL Random Forest" are interesting (i) the results are better, see Table 5 (ii) the Robustness is better , see Table 6 and the models are "smaller" see Table 6. This results are interesting even if they are worse than the ones obtained by Khiops.

4 Future works: a closer look at bootstrap sampling and data distribution in HDFS

In our study, we have focused so far on understanding how the Mahout's RF setting did impact the tree building process and more especially we did focus on the splitting process. One thing that we did mention but did not dig into is the way the datasets are spread across nodes and used as a basis for bootstrap sampling. This is what is under scrutiny in this section.

4.1 Block size and data shuffle

When a dataset file is dropped into HDFS, it is divided into blocks of fixed size and those blocks are spread across the nodes of the cluster. On the Hadoop cluster we have used for experiments, the default block size is 128MB and we did not change this.

The learning process for Mahout RF consists in a Map-Reduce job: mappers build trees which are collected by a single Reducer. Each mapper proceeds only one HDFS block: so the trees built by a single mapper are all based on a sub-sampling limited by the block the mapper is working on. Hence from the start, the principle of bootstrap sampling is different from the original version of Breiman: this is no random sampling with replacement over the whole dataset.

This difference has two major consequences:

— First the initial distribution of the target in the dataset file may have a deep impact on the trees generated by different mappers. In the worst possible setting, if the rows in the dataset are sorted by target value, one mapper might proceeds rows that are all associated with the same target's modality.

— Second, depending on the number of columns, a block may contain relatively few rows and trees may be built upon a small sample. This really depends on the block size that was effective when the dataset was put into HDFS. It may be an issue for those really large marketing datamarts, which easily encompass over 5000 columns.

The problem of block size might be tackled by studying the impact of block size over prediction performance for different datasets. Hopefully we may find where to place the cursor so that we may configure our HDFS system properly for a given datamart. But then this is not a very agile approach in an environment where we have to consider different datamarts and where it is easier to add variables to a datamart. And this would not solve the issue of target distribution.

A more robust approach to simultaneously tackle the two issues mentioned above would be to design a job that takes care of splitting the initial dataset while taking care of the following constraints:

— the job should produce N subsets, where N is the degree of parallelism that we wish for,

— the job should take care of storing each output subset with a specific block size so that each subset get fully processed by exactly one mapper,

— the job should populate each subset through a process of random sampling with replacement over the whole initial dataset

— the job should offer an option to control the way the target's modalities are distributed in the subsets (stratified sampling)

Algorithm 1 Prepare Dataset Algorithm

```
Input:
    Sample learning Z stored in HDFS
    N: degree of parallelism
    $colTarget$: index of the target column in the dataset
Output: Sample learning prepared for Mahout RF stored in HDFS
This is a sequence of two Map-reduce jobs: $TargetDistribution \to DatasetSplit$
The job $TargetDistribution$ takes for input the dataset $Z$ and outputs distribution of the target modalities.
This first job has only one reducer.
The job $DatsetSplit$ takes for input the dataset $Z$ and outputs the splitted sample learning.
The mappers of $DatasetSplit$ also access the distribution of the target modalities.
This second job has $N$ reducers.

Mapper $TargetDistribution$
    1: $targetModality \leftarrow$ extract value at column $colTarget$ from current Row
    2: write to reducer $< targetModality, 1 >$

Reducer $TargetDistribution$
    1: input is of the form $< targetModality, List\ of\ counters >$
    2: write $< targetModality, sum\ of\ counters >$

Mapper $datasetSplit$
    1: Before iterating over the rows of the input block, load the target distribution
    2: For row from the input block
    3:   For i=1 to N
    4:     $mod\_target \leftarrow$ extract target's modality from current row
    5:     Draw if you keep the row in this split based on $mod\_target$ distribution
    6:         if so then write $< i$,filterRows($currentrow >$ to reducer.

Reducer $DatasetSplit$
    1: input is of the form $< reducerindex, List\ of\ rows >$
    2: For $aRow \in rows$
    3:   write $aRow$ to output Split file.
```

Such a job could easily be designed as a sequence of two map-reduce jobs. The first one would estimate the target modalities distribution and the second one would actually perform the stratified random sampling.

Note that the main subtlety in such an approach resides in correctly handling the size of the splits with respect to the average size of a row and the number of individuals in the original datasets. If the rows are really large, it will impact the number of rows we can fit in a single split, and consequently the size of the sample for building a tree. So large rows call for larger block size. Another strategy might be to decide that not all columns should be carried into every split of the datasets: the first step of sampling consists in picking a sub-family of all columns for any final split of the dataset. This would allow for smaller rows and thus larger number of rows in each split. The sketch for the map and reduce functions of such a job is presented in Algorithm 2.

4.2 Shifting trees across nodes

The approach that was sketched in the previous subsection revolved around one main idea: prepare the data in a more appropriate way so that we could take the best out of Mahout RF without actually changing the library. But it has a cost in terms of use: one has to duplicate the data from the inital dataset to the prepared splits. This process might take extra-time and cost extra storage space.

Another approach could be used to the same issue: making sure a tree is built from the broadest sample of the initial dataset. Our idea is that each mapper is bound to work on only one HDFS block. So instead of having each mapper responsible for growing complete trees, we propose to transform the Mahout RF process into an iteration of map-reduce jobs, with trees only growing by a fraction at each run. Consider you want to build a forest of a T trees (say a thousand) and the dataset has been split into B blocks (say 10). The process would be initiated by generating B files, each of these containing T/B (that's a hundred here) empty trees. Then we would randomly affect each of these files to each mapper and run one step of the learning process. This step would grow the trees by two depth degree and store the updated trees into B files again. Next step is to randomly associate the updated files to mappers again and iterate until all trees are fully developed.

Such an approach would require substantial coding over the existing Mahout library, especially to store and parse set of incomplete trees. Plus there are certainly a few parameters to tune (how much should we develop trees at each step ?). But the advantage is that it is then possible to build trees from a wider sample set without moving the dataset itself.

5 Conclusion

This study exhibits the behavior of the Random Forest implemented in the Mahout framework on several datasets of interest for Orange. The first statement is the observation, for the standard version, of performances far below the one that we can observe in the state of the art. The practical implementation in the framework of Hadoop and Mahout does not respect totally the theoretical framework. This translates into performances below expectation. We propose a few evolutions, the main one being the introduction of explicit regularization during the training of the trees. Our tests on various datasets demonstrate an improvement over the standard library.

There is nevertheless an important work to realize, according to us, so that random forest implemented within the framework of Mahout reaches the performances of a batch algorithm. There is a known compromise between precision and speed / volumetry but the prize to be paid seems rather important for the moment.

References

The Apach^{TM} Hadoop® project develops open-source software for reliable, scalable, distributed computing, `http://hadoop.apache.org/`.

The Apach^{TM} Mahout® project's goal is to build a scalable machine learning library, `http://mahout.apache.org/`.

Bache, K. and M. Lichman (2013). UCI machine learning repository `http://archive.ics.uci.edu/ml`.

Boullé, M. (2005). A Bayes optimal approach for partitioning the values of categorical attributes. *Journal of Machine Learning Research 6*, 1431–1452.

Boullé, M. (2006). MODL: a Bayes optimal discretization method for continuous attributes. *Machine Learning 65*(1), 131–165.

Boullé, M. (2014). Towards automatic feature construction for supervised classication. In *ECML/PKDD 2014*. accepted for publication.

Boullé, M. (2007). Compression-based averaging of selective naive Bayes classifiers. *Journal of Machine Learning Research 8*, 1659–1685.

Breiman, L. Missing value replacement for the training set `https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm#missing1`.

Breiman, L. (2001). Random forests. *Mach. Learn. 45*(1), 5–32.

Debreuve, E. An introduction to random forests `http://perso.math.univ-toulouse.fr/motimo/files/2013/07/random-forest.pdf`.

Dream, A. (2013). Evaluer l'apport des plateformes hadoop pour la classification. Master's thesis, Polytech Lille.

Fawcett, T. (2006). An introduction to roc analysis. *Pattern Recogn. Lett. 27*(8), 861–874.

Flach, P. (2012). *Machine Learning: The Art and Science of Algorithms That Make Sense of Data*. Cambridge University Press.

Freund, Y. and R. E. Schapire (1996). Experiments with a new boosting algorithm. In *International Conference on Mchine Learning (ICML)*, pp. 148–156. Morgan Kaufmann.

Guyon, I., V. Lemaire, M. Boullé, G. Dror, and D. Vogel (2010). Design and analysis of the kdd cup 2009: Fast scoring on a large orange customer database. *SIGKDD Explor. Newsl. 11*(2), 68–76.

Harris, E. (2002). Information gain versus gain ratio: A study of split method biases. In *AMAI*.

Orange (2009). Kdd cup 2009: Customer relationship prediction – data kddlarge et kddsmall `http://www.sigkdd.org/kdd-cup-2009-customer-relationship-prediction`.

Quinlan, J. R. (1986). Induction of decision trees. *Mach. Learn. 1*(1), 81–106.

Sonnenburg, S., V. Franc, E. Yom-Tov, and M. Sebag (2008). Pascal large scale learning challenge – ocr `http://largescale.ml.tu-berlin.de/about/"`.

Voisine, N., M. Boullé, and C. Hue (2010). A bayes evaluation criterion for decision trees. *Advances in Knowledge Discovery and Management (AKDM-1) 292*, 21–38.

Résumé

L'apprentissage automatique a fait son apparition dans l'écosystème Hadoop créant, de par la puissance promise, une opportunité sans précédent pour ce domaine. Dans cet écosystème, Apache Mahout est une réponse à la question du temps de calcul et/ou de la volumétrie: il consiste en un entrepôt d'algorithmes d'apprentissage automatique, tous portés afin de s'exécuter sur Map/Reduce. Ce rapport se concentre sur le portage et l'utilisation de l'algorithme des Random Forest dans Mahout. Il montre à travers notre retour d'expérience les difficultés qui peuvent être rencontrées tant pratiques que théoriques et suggère une piste d'amélioration.