DATA PARTITIONING FOR ENSEMBLE MODEL BUILDING

Ates Dagli[1], Niall McCarroll[2] and Dmitry Vasilenko[3]

[1]IBM Big Data Analytics, Chicago, USA
[2]IBM Watson Machine Learning, Hursley, UK
[3]IBM Watson Cloud Platform and Data Science, Chicago, USA

ABSTRACT

In distributed ensemble model-building algorithms, the performance and statistical validity of models are dependent on sizes of the input data partitions as well as the distribution of records among the partitions. Failure to correctly select and pre-process the data often results in the models which are not stable and do not perform well. This article introduces an optimized approach to building the ensemble models for very large data sets in distributed map-reduce environments using Pass-Stream-Merge (PSM) algorithm. To ensure the model correctness the input data is randomly distributed using the facilities built into map-reduce frameworks.

KEYWORDS

Ensemble Models, Pass-Stream-Merge, Big Data, Map-Reduce, Cloud

1. INTRODUCTION

Ensemble models are used to enhance model accuracy (boosting) [1], enhance model stability (bagging) [2], and build models for very large datasets (pass, stream, merge) [3]. In distributed ensemble model- building algorithms, one so-called base model is built from each data partition (split) and evaluated against a sample set aside for this purpose. The best-performing base models are then selected and combined into a model ensemble for purposes of prediction. Both model-building performance and the statistical validity of the models depend on data records being distributed approximately randomly across roughly equal-sized partitions. When implemented in a map-reduce framework, base models are built in mappers [4]. Sizes of data partitions and the distribution of records among them are properties of the input data source. The partition size of the input source is often uneven and rarely of appropriate size for building models. Furthermore, data records are frequently arranged in some systematic order and not randomly ordered. As a result, base models sometimes fail to build or, what is worse, produce incorrect or suboptimal results. The algorithm proposed in this paper eliminates any partition size and ordering variability and as a result improves performance and statistical validity of the generated models.

2. METHODOLOGY

We implement the PSM features *Pass*, *Stream* and *Merge* through ensemble modeling [2], [5], [6], [7], [8], [9]. *Pass* builds models on very large data sets with only one data pass [3]; *Stream* updates the existing model with new cases without the need to store or recall the old training data; *Merge* builds models in a distributed environment and merges the built models into one model. In an ensemble model, the training set will be divided into subsets called blocks, and a model will be built on each block. Because the blocks may be dispatched to different processing nodes in the map reduce environment, models can be built concurrently. As new data blocks arrive, the

DOI: 10.5121/ijccsa.2017.7401

algorithm repeats the procedure. Therefore, it can handle the data stream and perform incremental learning for ensemble modeling [10]. The *Pass* operation includes following steps:

1. Splitting the data into training blocks, a testing set and a holdout set.
2. Building base models on training blocks and a reference model on the testing set. One model is built on the testing set and one on each training block.
3. Evaluating each base model by computing its accuracy based on the testing set and selecting a subset of base models as ensemble elements according to accuracy.
   During the *Stream* step when new cases arrive and the existing ensemble model

Needs to be updated with these cases, the algorithm will:

1. Start a Pass operation to build an ensemble model on the new data, and then
2. Merge the newly created ensemble model and the existing ensemble model.

The *Merge* operation has the following steps:

1. Merging the holdout sets into a single holdout set and, if necessary, reducing the set to a reasonable size.
2. Merging the testing sets into a single testing set and, if necessary, reducing the set to a reasonable size.
3. Building a merged reference model on the merged testing set.
4. Evaluating every base model by computing its accuracy based on the merged testing set and selecting a subset of base models as elements of the merged ensemble model according to accuracy.
5. Evaluating the merged ensemble model and the merged reference model by computing their accuracy based on the merged holdout set.

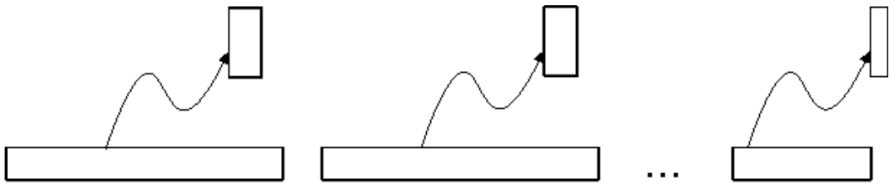

Figure 1. Data blocks and base models

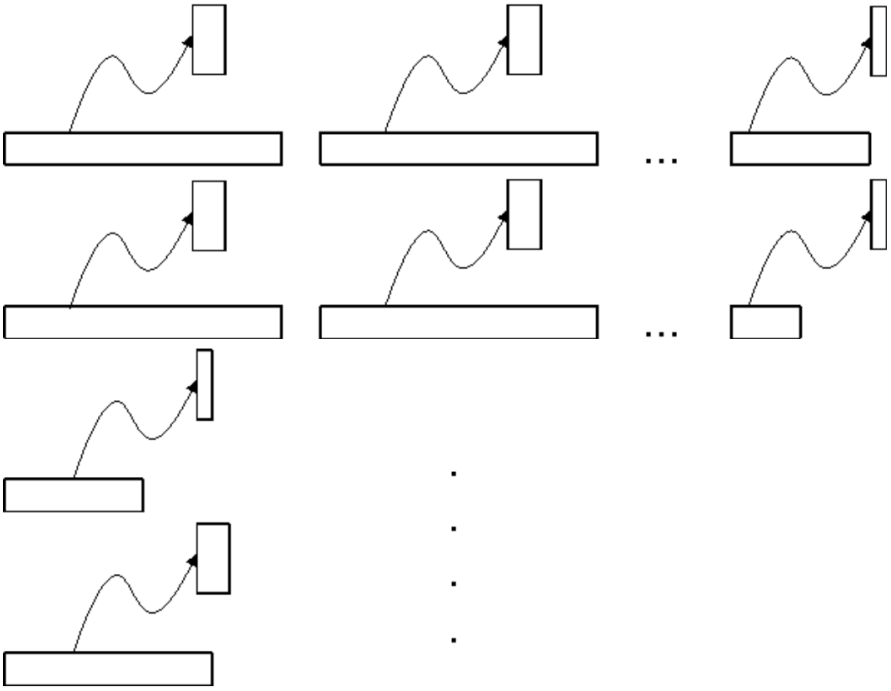

Figure 2. File sizes and base models

It can be shown that in map reduce environments the training block or partition size of the input source is often uneven and rarely of appropriate size for building models. The simplest example of uneven partition sizes is one caused by the fact that the last block of a file in the distributed file system is almost always a different size from those before it.

In the example of Figure 1, the base model built from the last block is built from a smaller number of records. When the dataset is comprised of multiple files as is often the case, the number of small partitions increases. This is illustrated in Figure 2.

Another assumption that is frequently violated is that the input records are randomly distributed among partitions. If the records in the dataset are ordered by the values of the modeling target field, an input field, or a field correlated with them, base models cannot be built or exhibit low predictive accuracy. In the example shown in Figure 3, records are ordered by the binary-valued target. No model can be built from the first or the last partition because there is no variation in the target value. A model can probably be built from the second partition but its quality depends on where the boundary between the two target values lies.

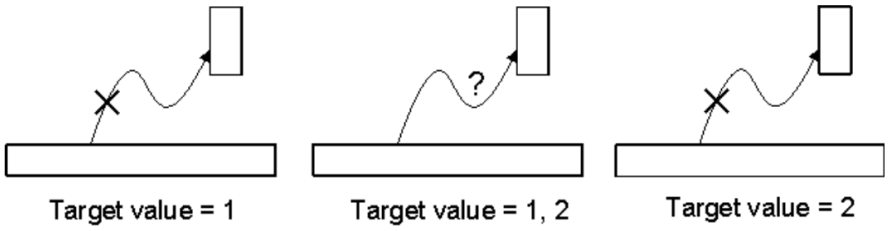

Figure 3. The data sort order and base models

An existing partial solution to the problem is to first run a map-reduce job that shuffles the input records and creates a temporary data set. Shuffling is achieved by creating a random-valued field and sorting the data by that field. The ensemble-building program is subsequently run with the temporary data set as input. The solution is only partial because, while records are now randomly ordered, the partitioning of data is determined by the needs of the random sorting. The resulting partitioning is rarely optimal because the upper limit on the partition size is still dependent on the default block size employed by the map-reduce framework, and smaller, unevenly sized partitions may be created. This approach also requires the duplication of all the input data in a temporary data set.

We propose a method that provides optimally-sized partitions of shuffled, or randomly ordered, records to model-building steps using facilities built into map-reduce frameworks. The model-building step is run in reducers with input partitions whose size is configurable automatically or by the user. The contents of the partitions are randomly assigned. Our approach allows the partition size to be set at runtime. The partition size may be based on statistical heuristic rules, properties of the modeling problem, properties of the computing environment, or any combination these factors. Each partition consists of a set of records selected with equal probability from the input. The advantages of using our method over the known explicit shuffling solution are:

1. Our method guarantees partitions of uniform optimal size. The explicit shuffling solution cannot guarantee a given size or uniform sizes.
2. Map-reduce frameworks have built-in mechanisms for automatically grouping records passed to reducers. Explicit shuffling incurs the additional cost of the creation of a temporary dataset and a sort operation. This description is confined to the portion of ensemble modeling process where base models are built because that is the step our approach improves.

In addition to partitions for building base models, ensemble modeling requires the creation of two small random samples of records called the validation and the holdout samples. The sizes of these samples are preset constants. A so-called reference model is built from the validation sample in order to compare with the ensemble later. The validation sample is also used to rank the predictive performance of the base models in a later step. Also in a later step, the holdout sample is used to compare the predictive performance of the ensemble with that of the reference model. The desired number of models in the final ensemble, E, is determined by the user. It is usually in the range 10-100. The rules we use to determine how to partition the data balance the goal of a desirable partition size with that of building an ensemble of the desired size.

To compute the average adjusted partition size we pick an optimal value for the size of the base model partition, B. B may be based on statistical heuristic rules, properties of the modeling problem, properties of the computing environment, or any combination of these factors.

We also determine a minimum acceptable value for the size of the base model partition, B_{min}, based on the same factors. Given N, the total number of records in the input dataset, the size of the holdout sample H, and the size of the validation sample V, we determine the number of base models, S, as follows:

Algorithm 1 Base Model Number Algorithm

```
Set S = E + min(floor((E+15)/20),5)
Set partitionSize = floor((N-H-V)/S)
if B_min ≤ partitionSize ≤ B then
    accept S
else if partitionSize > B then
    compute S = floor((N-H-V)/B) and accept S
else compute S = floor((N-H-V)/ B_min)
    if S ≥ 3 then
        accept S.
    else there are too few records for this ensemble modeling technique to apply
    end if
end if
```

Given *S*, compute an average adjusted partition size, *B′* as
B′ = (N−H−V)/S.
B′ is usually not a whole number. Sampling probabilities for base partitions, the validation sample, and the holdout sample are *B′/N*, *V/N*, and *H/N*, respectively. Note that
*S *(B′/N) + V/N + H/N = N/N = 1*
so that the sampling probabilities add up to 1.

The map stage consists of randomly assigning each record one of *k+2* keys *1, 2, ... S+2*. Key values *1* and *2* correspond to the holdout and validation samples, respectively. Thus, a given record is assigned key *1* with probability *H/N*, key *2* with probability *V/N*, and keys *3..S+2* each with probability *B′/N*. The resulting value is used as the map-reduce key. The key is also written to mapper output records so that the reducers can distinguish partitions *1* and *2* from the base model partitions. The sizes of the resulting partitions will be approximately *B′*, *H* and *V*.
In the reduce stage, we build models from each partition except the holdout sample.

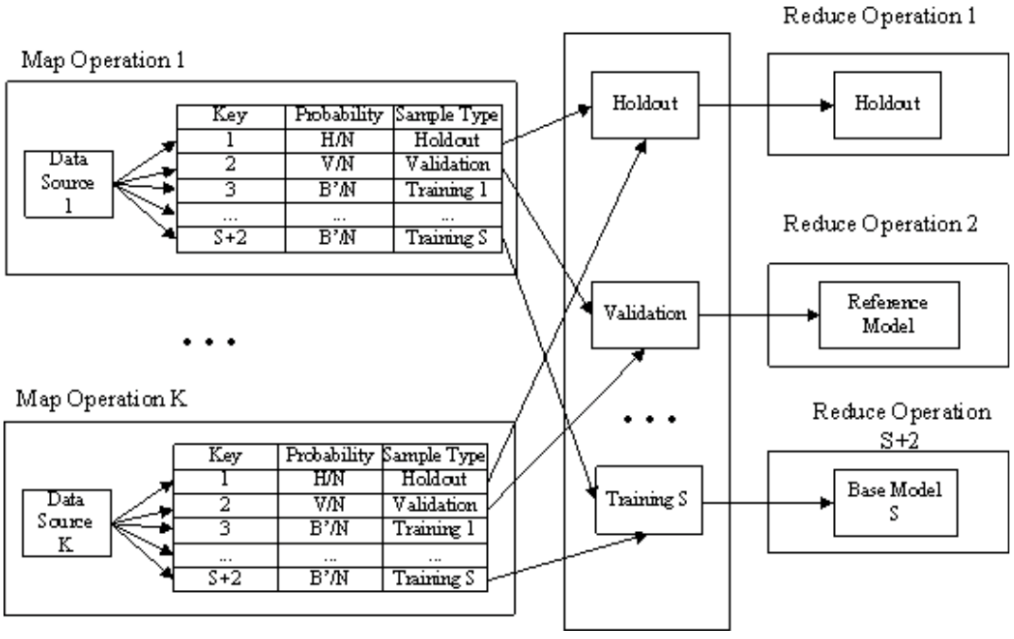

Figure 4. The data flow through mappers to reducers

Figure 4 shows the flow of input data through an arbitrary number K of mappers to the reducers where base and reference models are built. Regardless of the order and grouping of the input data, the expected sizes of the holdout, validation and base model training partitions are as determined above and their contents are randomly assigned.

3. CONCLUSIONS

We have presented an algorithm for creating optimally-sized random partitioning for data-parallel ensemble model building in map-reduce environments. The approach improves performance and statistical validity of the generated ensemble models by introducing random record keys that reflect probabilities for the holdout, validation and training samples. The keys are also written to mapper output records so that the reducers can distinguish holdout and validation partitions from the base model partitions. In the reduce stage, we build models from each partition except the holdout sample.

The future plan is to implement the algorithm in the real cloud setup and test the performance and statistical validity by considering the anticipated workload.

ACKNOWLEDGEMENTS

Authors thank Steven Halter of IBM for invaluable comments during preparation of this paper.

REFERENCES

[1] R. Schapire and Y. Freund. Boosting: Foundations and Algorithms. MIT, 2012.
[2] R. Bordawekar, B. Blainey, C. Apte, and M. McRoberts. A survey of business analytics models and algorithms. IBM Research Report RC25186, IBM, November 2011. W1107- 029.
[3] L. Wilkinson. System and method for computing analytics on structured data. Patent US 7627432, December 2009.
[4] M. I. Danciu, L. Fan, M. McRoberts, J Shyr, D. Spisic, and J. Xu. Generating a predictive model from multiple data sources. Patent US 20120278275 A1, November 2012.
[5] R. Levesque. Programming and Data Management for IBM SPSS Statistics. IBM Corporation, Armonk, New York, 2011.
[6] R. Polikar. Ensemble based systems in decision making. IEEE Circuits and Systems Magazine, 6(3):21–45, 2006.
[7] D. Opitz and R. Maclin. Popular ensemble methods: An empirical study. Journal of Artificial Intelligence Research, 1(11):169–198, 1999.
[8] B. Zenko. Is combining classifiers better than selecting the best one. Machine Learning, 1 (1):255–273, 2004.
[9] Z. Zhihua. Ensemble Methods: Foundations and Algorithms. Chapman and Hall/CRC, 2012.
[10] IBM SPSS Modeler 16 Algorithms Guide. IBM Corporation, Armonk, New York, 2013.

AUTHORS

Ates Dagli is a Principal Software Engineer working on the IBM SPSS Analytic Server line of products. Mr. Dagli graduated from the Columbia University in the City of New York with Master of Philosophy degree in Economics in 1978. Mr. Dagli led design and implementation of the variety of statistical and machine learning algorithms that address the entire analytical process, from planning to data collection to analysis, reporting and deployment.

Niall McCarroll, Ph.D., is an Architect and Senior Software Engineer working on the IBM Watson Machine Learning and predictive analytics tools. In the last few years Niall has been involved in the creation and teaching of a module covering applied predictive analytics in several UK universities.

Dmitry Vasilenko is an Architect and Senior Software Engineer working on the IBM Watson Cloud Platform. He received a M.S. degree in Electrical Engineering from Novosibirsk State Technical University, Russian Federation, in 1986. Before joining IBM SPSS in 1997 Mr. Vasilenko led Computer Aided Design projects in the area of Electrical Engineering at the Institute of Electric Power System and Electric Transmission Networks. During his tenure with IBM Mr. Vasilenko received three technical excellence awards for his work in Business Analytics. He is an author or co-author of 11 technical papers andaUSpatent.