IADIS International Journal on Computer Science and Information Systems
Vol. 13, No. 1, pp. 1-13
ISSN: 1646-3692

A MAPREDUCE BASED FRAMEWORK TO PERFORM FULL MODEL SELECTION IN VERY LARGE DATASETS

Angel Díaz Pacheco, Jesús A. Gonzalez-Bernal and Carlos A. Reyes-Garcia
Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE), Computer Science Department. Luis Enrique Erro No.1, Santa María Tonantzintla, Puebla, 72840, Mexico

ABSTRACT

The analysis of large amounts of data has become an important task in science and business that led to the emergence of the Big Data paradigm. This paradigm owes its name to data objects too large to be processed by standard hardware and algorithms. Many data analysis tasks involve the use of machine learning techniques. The goal of predictive models consists on achieving the highest possible accuracy to predict new samples, and for this reason there is high interest in selecting the most suitable algorithm for a specific dataset. Selecting the most suitable algorithm together with feature selection and data preparation techniques integrates the Full Model Selection paradigm and it has been widely studied in datasets of common size, but poorly explored in the Big Data context. As an effort to explore in this direction, this work proposes a framework adjustable to any population based meta-heuristic methods in order to perform model selection under the MapReduce paradigm.

KEYWORDS

Model Selection, MapReduce, Big Data

1. INTRODUCTION

In many fields of human activity there is a growing interest in storing and analyzing information. This information is quickly generated and in large amounts. With the advent of new technologies such as social networks, the quantity and variety of data has grown to unprecedented scales. In 2014, 2.5 quintillions of Bytes of information were daily created (Wu et al., 2014). Big Data became popular due to this phenomenon. A widespread definition of Big Data describes this concept in terms of three characteristics of information in this field: Volume, Velocity and Variety (del Rio et al., 2015). Subsequently other V's have been added: Veracity and Value (Tili & Hamdani, 2014). In this regard, turning the information into a valuable asset is carried out through machine learning techniques and choosing an appropriate learning algorithm is not a trivial task. This process requires finding the combination of learning algorithms together with their hyper-parameters to achieve the lowest misclassification rate in a wide search space (Thornton et al., 2013). In that same spirit, the Full Model Selection (FMS) paradigm proposed by Escalante et al. (2009) considers also the data preparation as the discretization or data normalization and the dimensionality reduction through a feature selection algorithm. FMS has been addressed as an optimization problem varying the search technique employed. As an example, Kaneko & Funatsu (2015) proposed a hybrid method based on grid search and the theoretic hyper-parameter decision technique (ThD) of Cherkassy and Ma for the algorithm SVR (Support Vector Regression). In Bergstra & Bengio (2012) evidence was obtained that random search can get similar or even better models than grid search in a fraction of the computing time. Escalante et al. (2009) tackled and defined the full model selection problem with the use of a particle swarm optimization algorithm (PSO), meanwhile Bansal & Sahoo (2015) proposed the use of the bat algorithm for solving FMS. Chatelain et al. (2010) presented a method for model selection based on a multi-objective genetic algorithm where the evaluation function takes into account sensitivity as well as specificity. Rosales-Perez et al. (2014) also tackled the search phase with a multi-objective genetic algorithm. They take into account the misclassification rate and the complexity of the models computed through the Vapnik-Chervonenkis dimension (VC). In 2015 Rosales-Pérez et al. proposed the inclusion of surrogate functions in order to decrease the use of expensive fitness functions. All of the previous approaches for the model selection problem work with datasets that can be loaded in main memory, but if the amount of data is larger than a conventional personal computer can store, the model exhibiting the lowest error will not be found. This paper has the following organization. In section 2 we present some background on Big Data and MapReduce. Section 3 describes our proposed framework. Section 4 shows the experiments performed to test the validity of our proposal. Finally, section 5 presents conclusions and future work.

2. BIG DATA AND THE MAPREDUCE PROGRAMMING MODEL

MapReduce was introduced by Dean and Ghemawat in 2004 with the goal of enabling the parallelization and distribution of big scale computation required to analyze large datasets. This programming model was designed to work over computing clusters and it works under the master-slave communication model. MapReduce considers the following principles: 1) Low-Cost Unreliable Commodity Hardware. MapReduce is designed to run on clusters of commodity hardware. 2) Extremely Scalable Cluster. Nodes can be taken out of service with almost no impact on the jobs. 3) Fault tolerant. MapReduce replicates data in order to keep processes running in case of failures. In the MapReduce programming model a computing task is specified as a sequence of stages: map, shuffle and reduce that works on a dataset $X = \{x_1, x_2, \ldots, x_n\}$. The map step applies a function μ to each value x_i to produce a finite set of key-value pairs (k, v). To allow for parallel execution, the computation of function $\mu(x_i)$, must depend only on x_i. The shuffle step collects all the key-value pairs produced in the previous map step, and produces a set of lists, $L_k = (k; v_1, v_2, \ldots, v_n)$ where each of such lists consists of all values v_i, such that $k_i = k$ for a key k assigned in the map step. The reduce stage applies a function ρ to each list $L_k = (k; v_1, v_2, \ldots, v_n)$, created during the shuffle step, to produce a set of values $y_1, y_2, \ldots, y_n$. The reduce function ρ is defined to work sequentially on L_k but should be independent of other lists L_k, where $k' \neq k$ (Goodrich et al., 2011). Despite being an extended definition, the 5 V's of Big Data are a little ambiguous and do not provide rules to identify a huge dataset. In this work we propose an alternative definition but related to the model selection problem. For our definition we propose that a huge dataset for the model selection problem must accomplish two rules: 1) The dataset size is big enough that at least one of the considered classification algorithms in their sequential version cannot process it. 2) The dataset size is defined by its file size considering the number of instances (I) and features (F) as long as I » F.

3. A FRAMEWORK FOR FMS UNDER MAPREDUCE

This section describes our proposed framework for FMS. This algorithm was developed under Apache Spark 1.6.0, a framework based on MapReduce. We selected this framework because of its enhanced capacity to deal with iterative algorithms and the possibility to perform data processing in main memory (if memory capacity allows it). FMS was described in the previous sections, but Eq. 1 is given as formal definition. Given a set of learning algorithms A, data preparation techniques P and feature selection algorithms F, the goal of the FMS is to determine the combination of algorithms: $a^*_{wA} \in A$ (a machine learning algorithm with an specific configuration in its hyper-parameter values), $p^* \in P$ and $f^* \in F$ with the lowest misclassification rate. The misclassification rate is estimated over the dataset D and this dataset is splited in two disjoint partitions (D^i_{train} and $D^i_{validation}$ for $i = 1,2, \ldots, k$). The misclassification rate is calculated with the loss function $\frac{1}{k}\sum_{i=1}^{k} \mathcal{L}\,(a_{wA}, p^j, f, D^{(\mathrm{i})}_{\mathrm{train}}, D^{(i)}_{validation})$, training the algorithm a^*_{wA} in the partition $\mathrm{D}^{(\mathrm{i})}_{\mathrm{train}}$, and evaluated in the partition $D^{(i)}_{validation}$. The data partitions are previously transformed by p and f.

$$a^*_{wA}, p^*, f^* \in argmin \frac{1}{k}\sum_{i=1}^{k} \mathcal{L}\left(a_{wA}, p, f, D^{(i)}_{train}, D^{(i)}_{validation}\right) \qquad (1)$$
$$a^{(i)} \in A, p^{(i)} \in P, f^{(i)} \in F, w_A \in W_A$$

3.1 Representation

The solutions encoded in a population based meta-heuristic method needs to be codified in a vector. Depending on the nature of the meta-heuristic method, the codification scheme of such vector varies between real and binary. In this work we choose a real codification scheme because their adaptability of a wide range of meta-heuristic methods. The solution vector $x^i = [x^i_1, x^i_2, \ldots, x^i_{16}]$ is encoded as follows: In position 1 the fitness of the potential models is stored. Position 2 allows to determine which operation will be done first: data-preparation or feature selection. Position 3 indicates if the data-preparation step will be done. Positions 4 to 6 are parameters for the data-preparation step (method identifier, parameter 1 and parameter 2). Position 7 determines if the feature selection step will be done. Positions 8 and 9 are for the feature selection step (Method identifier and number of features to be selected respectively). Positions 10 to 16 are for the machine learning algorithm construction. The range of values that every element in the vector can take is as follows: [0-100]; [0,1], [0,1], [1,30], [1,NF], [1,50], [0,1], [1,5], [1,NF], [1,6], [1,2], [1,4], [1,100], [1,60], [1,400], [-20,20] with NF = Number of Features.

3.2 Model Evaluation

In this work Apache Spark is employed. Spark is an improvement over traditional MapReduce specially regarding to main memory use in order to reduce several reading/writing operations to disc. The cornerstone of Spark is the RDD or Resilient Distributed Dataset which is a collection of partitioned data elements that can be processed in parallel (Guller, 2015). In Algorithm 1 is shown the process to obtain an RDD from a text file. An analysis of the advantages of Spark over traditional MapReduce is out of the scope of this work, but we refer to Zaharia et al. (2016).

Algorithm 1: Get the RDD

```
getRDD(PathDataset,numparts)
 RowRDD = Load(PathDataset,numparts) //Obtains RDD[String]
 RDDcol = RowRDD.map(row->row.split(",")) /* obtains
 RDD[Array[String]] */
 RDDVect = RDDcol.map(row->Vector(row.map(ColInR ->
ColInR.toDouble)))
 // Obtains RDD[Vector[Double]]
Return(RDDVect)
End
```

Algorithm 1. Algorithm that obtain an RDD[Vector[Double]] from plain text

Algorithm 2: Generic search algorithm

```
SearchMethod(TestSet,TrainSet,NIt)
 Population = CreateInitialRandomPopulation()
 fitness = MRfitness(Population,TrainSet,labels)
 /* MRfitness evaluates the performance of the solutions */
 ItCount=0;
 While(GenCount < NIt)
  UpdatedPop = updatesPopulation(Population)
  fitness = MRfitness(Population,TrainSet,labels)
  Population = replacement(Population,UpdatedPop)
  /*Elitistic replacement by the solutions fitness */
  ItCount +=1
 EndWhile
 fModel = buildFM(Population)//Builds final model
 finalFitness = evalFinalModel(fModel,TestSet)
End
```

Algorithm 2. Generic search algorithm for Full Model Selection

The function getRDD is employed to obtain the training and test RDDs that are used as parameters for the search algorithm based on the Spark's RDD for Full Model Selection. In Algorithm 2 a generic search procedure is shown. The models evaluation stage is comprised of the data preparation, feature selection, and training of a classification algorithm. In Algorithm 3 this process is shown.

```
Algorithm 3: Fitness calculation

MRfitness(Population,TrainSet)
 fitness = Array[Double](Population.length)
 For(i = 0; i < Population.length; i++)
  solution = Population(i)
  precedence = solution(2)
  If(precedence == 0)
   RDDPrep = DataPrep(TrainSet,solution)/*Performs data
   Preparation */
   RDDFS = FeatSelection(RDDPrep,solution)/*Performs feature
   Selection */
   fitness(i) = Classification(RDDFS,solution) /*Performs
   classification */
  Else
   RDDFS = FeatSelection(TrainSet,solution)/*Performs
   feature selection */
   RDDPrep = DataPrep(RDDFS,solution)/*Performs data
   Preparation */
   fitness(i) = Classification(RDDPrep,solution)/*Performs
   classification */
  EndIf
 EndFor
Return(fitness)
```

Algorithm 3. Algorithm that obtain the fitness of every solution in the population

The processes of data preparation, feature selection and classification are described in Algorithms 4 to 6. For greater ease of understanding the algorithms are commented (the comments are denoted with the symbols "// or /* */").

```
Algorithm 4: Data preparation

DataPrep(DataSet,solution)
Return(DataSet.map(row->row.toArray.map(col
>Transform(col,solution))))
 /*Transform function is applied to every column of each row in the
RDD*/
End
```

Algorithm 4. Algorithm that performs data-preparation

In our proposal, the mean error over the 2-fold cross validation is used in order to evaluate the performance of every potential model. During the test stage in the development of the framework, different number of folds were evaluated (2,...,10) without significant differences, but adding to the computing time factor, the 2-fold cross validation was the best choice. Under another programming paradigm (other than MapReduce), the construction of a single model with such large amount of data would have been impossible. The algorithms used in this work for data-preparation are: 1) Feature standardization, 2) Normalization, 3) Principal Component analysis, 4) Shift and scale and 5) Discretization. For feature selection, the algorithms are: 1) Joint Mutual Information, 2) Minimum Redundancy Maximum Relevance, 3) Interaction Capping, 4) Conditional Mutual Information Maximization and 5) Informative Fragments. Finally the classification algorithms are: 1) Support Vector Machine (SVM), 2) Logistic Regression (LR), 3) Nave Bayes (NB), 4) Decision Tree (DT), 4) Random Forest (RF) and 5) Gradient-Boosted Trees (GBT).

```
Algorithm 5: Feature Selection

FeatSelection(DataSet,solution)
 numFeat = solution(9)
 rankRDD = DataSet.map(row->RankingCalculation(row))
 /*RankingCalculation function obtains the ranking of the features
*/
 reducedRDD = rankRDD.map(row->getF(row,numFeat)) /*function getF
is applied
 to rankRDD and returns a reduced dataset */
Return(reducedRDD)
End
```

Algorithm 5. Algorithm that performs feature selection

```
Algorithm 6: Classification

Classification(DataSet,solution)
 kFolds=createFold(DataSet,NumFolds=2)/*createFold function
  creates an RDD for k-Fold Cross validation */
  error=kFolds.map{
  case(Training,Validation)/*dataset is separated in
  Training and Validation */
  model = createModel(Training, solution)/*creates a model
  PredictedTargets=Validation.map(Instance->
  model.predict(Instance.features))/*Performs predictions in
  validation set */
  accuracy= getAcc(PredictedTargets,Validation.targets)
  //Obtains the accuracy in each fold
  error = 100-accuracy
 Return(error) }
  meanError=error.sum/error.length
 Return(meanError)
End
```

Algorithm 6. Performs classification in the dataset

4. EXPERIMENTS AND RESULTS

With the purpose to evaluate the performance of the proposed framework, two population based meta-heuristic methods were tested: a Genetic Algorithm (**GA**) and the PSO based FMS algorithm proposed in Escalante et al. (2009) **PSMS**. We experimented with the datasets shown in Table 1 that have been used in several classification research works for Big Data (del Rio et al.,20014; del Rio et al., 2015; Lopez et al., 2014; Lopez et al., 2014a) . We also used two synthetic data sets generated with a tool for synthetic datasets generation in the context of ordinal regression: "Synthetic Datasets Nspheres" provided in Sánchez-Monedero et al. (2013).

Table 1. Datasets used in the experiments

Datasets	Data points	Attributes	Samples by class	Type of variables	File Size
Fars	50164	52	(17,445; 32,719)	Categorical	6.0 MB
Census-income	1999523	40	(187,141; 12,382)	Categorical	18.0 MB
Covtype	495141	54	(283,301; 211,840)	Categorical	61.6 MB
RLCP	5749111	11	(5728197;20915)	Real	261.6 MB
KDD	4856150	41	(972780;3883369)	Categorical	653 MB
Synthetic 1	200000000	3	(100000000;100000000)	Real	5.5 GB
Higgs	11000000	28	(5170877;5829123)	Real	7.5 GB
Synthetic 2	49000002	30	(24500001;24500001)	Real	12.7 GB

The stopping criteria of both algorithms **GA** and **PSMS** were to perform 47 generations/iterations and the swarm/population size was of 30 individuals/particles (1,410 models). As both algorithms are population based we take advantage of that fact and the final model is an ensemble of all the best individuals. This ensemble is performed by a weighted voting scheme. The obtained results (mean error) in the datasets over 44 replications are shown in Table 2.

Table 2. Mean classification error obtained in the test dataset by the algorithms GA and PSMS over 44 replications. The lowest mean errors are in **bold**

Datasets	GA	PSMS
Fars	0.166 ±0.025	**0.162±0.025**
Census-income	5.751±0.154	**5.718±0.397**
Covtype	6.685±0.218	**5.328±0.356**
RLCP	**0.009±0.001**	0.052±0.001
KDD	**0.025±0.007**	0.166±0.148
Synthetic 1	**15.862±0.004**	15.864±0.004
Higgs	29.507±0.143	**28.304±0.064**
Synthetic 2	6.684±0.002	**6.683±0.005**

Table 3. t-statistic obtained from the Student's t-test. The critical values at the 95% confidence level are 2.016 with 43 d.f. Cases that exceed the critical value (absolute value) are considered as a difference that is statistically significant at the fixed level and are in **bold**

Datasets	t-statistic
Fars	0.633
Census-income	0.534
Covtype	**22.486**
RLCP	**-188.965**
KDD	**-6.276**
Synthetic 1	-1.7405
Higgs	**52.156**
Synthetic 2	0.996

Each replication was performed with a particular random sample of the data points with different random samples among replications. For each experiment, the dataset was divided into two disjoint datasets with 60% of the data samples for the training set and 40% for the test set. The 44 replications were performed in order to obtain an statistical power of 90% in a Student's t-test and get evidences of significant differences in the performance of the **GA** and

PSMS. From Table 2, it can be seen that **PSMS** obtains the lowest error in five of the eight evaluated datasets. From Table 3, the t-test shows that exists significant differences in four datasets, favoring to **PSMS** in two of them and favoring to **GA** also in two. The results showed that the performance of both search techniques is similar. In order to contrast the performance of both techniques, two well know classification/model-selection techniques were employed. The Kernel Nearest-Neighbour algorithm (**K-NN**) present in Yu et al., (2002) and a grid search present in the librarys of Apache Spark **AST** for tuning machine learning algorithms (Apache org, 2017). Referring to **AST** we set the number of models to be evaluated to 1,412 (in order to be similar to the number of models evaluated by **PSMS**) using the algorithms mentioned in section 3.2 and for **K-NN** we set k=9,999.

Table 4. Mean classification error obtained in the test dataset by GA, PSMS and the obtained by K-NN (K=9,999) and AST, over 20 replications. The best results are in **bold**

Datasets	GA	PSMS	AST	K-NN
Fars	0.166±0.025	0.159±0.020	**0.157±0.042**	7.586±0.154
Census-income	**5.751±0.154**	5.771±0.444	6.727±1.018	6.408±0.093
Covtype	6.685±0.218	**5.321±0.339**	23.625±0.090	42.791±0.088
RLCP	0.009±0.001	0.052±0.001	**0.001±0.000**	0.500±0.098
KDD	0.025±0.007	0.156±0.134	**0.001±0.000**	19.535±0.261
Synthetic 1	15.862±0.004	**15.862±0.004**	17.011±0.120	50.088±0.028
Higgs	29.507±0.143	**28.299±0.057**	30.955±0.228	46.916±0.408
Synthetic 2	6.684±0.002	**6.681±0.005**	22.152±0.541	50.126±0.115

Once again, in order to obtain an statistical power of 90% in an ANOVA test 20 replications were made. Table 4 shows that **GA** gets the best performance in one of the eight datasets (Census) and **PSMS** obtains the lowest error in four (Covtype,Syntethic 1, Higgs and Syntethic 2) while **AST** in three (Fars, RLCP and KDD).

Table 5. F-statistic obtained from the ANOVA test and q-values from the Tukey HSD test for performing all possible pairwise comparisons among the proposed strategies for the final model construction. The critical values at the 95% confidence level for the ANOVA test are 3.16 (F(2,57)) for all datasets. The critical values at the 95% confidence level for the Tukey HSD test are 3.44 (57 degrees of freedom). Cases that exceed the critical value are considered as a difference that is statistically significant at the fixed level and are marked with an asterisk (*)

Datasets	Anova F	PSMS vs AST	PSMS vs K-NN
Fars	**42436.12***	0.082	**356.76***
Census-income	**11.43***	**6.642***	**4.42***
Covtype	**160999.48***	**391.98***	**802.98***
RLCP	**470.34***	**4.00***	**35.40***
KDD	**87594.48***	**4.10***	**510.56***
Synthetic 1	**1482334.54***	**71.95***	**2143.80***
Higgs	**27501.97***	**43.75***	**306.60***
Synthetic 2	**95223.56***	**216.80***	**608.82***

Table 5 shows that there are significant differences in all datasets respect to **PSMS** and **K-NN**. Regarding to **AST**, there are significant differences in seven of the eight datasets and favoring to **AST** in two (RLCP and KDD). That means that **PSMS** obtained the best performance in six of the eight datasets. Regarding to the **GA** Table 6 shows that there are also significant differences in all datasets compared to **K-NN** and there are significant differences in five datasets favoring to **GA** compared to **AST** (Census, Covtype,Sytethic 1, Higgs and Syntethic 2). In the remaining datasets (Fars, RLCP and KDD) there are not statistical differences in the performance of **AST** and **GA**. The obtained results shows that both search techniques are competitive and in the case of the **GA** even though is not the best in seven of the eight datasets, the statistical tests provide evidence that **GA** has a good performance in all datasets even in those that was not the best because there were no statistically significant differences between the performance of **GA** and the performance of the base line algorithms. The performance of both search algorithms shows that the final user can choose, create or modify the most suitable technique to their necessities and therefore is not limited to one single search option.

Table 6. F-statistic obtained from the ANOVA test and q-values from the Tukey HSD test for performing all possible pairwise comparisons among the proposed strategies for the final model construction. The critical values at the 95% confidence level for the ANOVA test are 3.16 (F(2,57)) for all datasets. The critical values at the 95% confidence level for the Tukey HSD test are 3.44 (57 degrees of freedom). Cases that exceed the critical value are considered as a difference that is statistically significant at the fixed level and are marked with an asterisk (*)

Datasets	Anova F	GA vs AST	GA vs K-NN
Fars	**41758.60***	0.756	**353.56***
Census-income	**11.88***	**6.796***	**4.396***
Covtype	**381028.72***	**578.120***	**1233.700***
RLCP	**509.53***	0.628	**38.779***
KDD	**111462.85***	0.693	**577.920***
Synthetic 1	**1482574.07***	**71.957***	**2144***
Higgs	**23910.89***	**23.168***	**278.660***
Synthetic 2	**95220.24***	**216.77***	**608.800***

5. CONCLUSIONS AND FUTURE WORK

In this work we proposed a framework to deal with the full model selection problem for very large datasets (Big Data). Our framework was implemented under the MapReduce programming paradigm. Experimental results shown that applying a model selection algorithm in order to analyze large and average size datasets is feasible. Our results show a significant predictive power improvement of the employed search algorithms compared with those in the base line (also designed for big data). The main advantages of our work is the adaptability of the framework to different population based meta-heuristic methods and therefore, the model selection can be plausible in datasets of a wide range of sizes and with a wide range of techniques (each one with its own strengths and weakness) according to the preferences of the final user. In future work we will look for the reduction of calls to expensive fitness functions, decrease the variance of the error rate produced by our framework, and find ways to control and penalize the complexity of the models obtained with this framework.

ACKNOWLEDGEMENT

The first author is grateful for the support from CONACyT scholarship no. 428581.

REFERENCES

Apacheorg (2017), `Ml tuning: model selection and hyperparameter tuning', http://spark.apache.org/docs/latest/ml-tuning.html.

Bansal, B., Sahoo, A., Escalante, H. J., Montes, M. & Sucar, L. E. (2015), `o', pp. 1-4.

Bergstra, J. & Bengio, Y. (2012), `Random Search for Hyper-Parameter Optimization', Journal ofMachine Learning Research 13, 281-305.

Chatelain, C., Adam, S., Lecourtier, Y., Heutte, L. & Paquet, T. (2010), 'A multi-model selection framework for unknown and/or evolutive misclassification cost problems', Pattern Recognition 43(3), 815-823. URL: http://dx.doi.org/10.1016/j.patcog.2009.07.006.

Dean, J. & Ghemawat, S. (2008), 'MapReduce', Communications of the ACM 51(1), 107-113.

del Río, S., Lopez, V., Benítez, J. M. & Herrera, F. (2014), 'On the use of mapreduce for imbalanced big data using random forest', Information Sciences 285, 112-137.

del Río, S., López, V., Benítez, J. M. & Herrera, F. (2015), 'A MapReduce Approach to Address Big Data Classification Problems Based on the Fusion of Linguistic Fuzzy Rules', International Journal of Computational Intelligence Systems 8(February), 422-437. URL: http://www.tandfonline.com/doi/abs/10.1080/18756891.2015.1017377.

Escalante, H. J., Montes, M. & Sucar, L. E. (2009), 'Particle swarm model selection', Journal of Machine Learning Research 10(Feb), 405-440.

Eshelman, L. J. (1991), 'The chc adaptive search algorithm: How to have safe search when engaging', Foundations of Genetic Algorithms 1991 (FOGA 1) 1, 265.

Goodrich, M. T., Sitchinava, N. & Zhang, Q. (2011), Sorting, searching, and simulation in the mapreduce framework, in `International Symposium on Algorithms and Computation', Springer, pp. 374-383.

Guller, M. (2015), 'Big data analytics with spark: A practitioners guide to using spark for large scale data analysis. apress', URL: http://www.apress.com/9781484209653.

Guo, X. C., Yang, J. H., Wu, C. G., Wang, C. Y. & Liang, Y. C. (2008), 'A novel LS-SVMs hyper-parameter selection based on particle swarm optimization', Neurocomputing, X. C. Guo, J. H. Yang, C. G. Wu, C. Y. Wang, and Y.C. Liang, A novel LS-SVMs hyper-parameter selection based on particle swarm optimization, Neurocomputing, vol. 71, pp. 32113215, 2008. 71, 3211-3215.

Haupt, R. L. & Haupt, S. E. (2004), Practical genetic algorithms, John Wiley & Sons.

Kaneko, H. & Funatsu, K. (2015), `Fast optimization of hyperparameters for support vector regression models with highly predictive ability', Chemometrics and Intelligent Laboratory Systems 142, 64-69. URL: http://linkinghub.elsevier.com/retrieve/pii/S0169743915000039.

López, V., del Río, S., Benítez, J. M. & Herrera, F. (2014), 'Cost-sensitive linguistic fuzzy rule based classification systems under the MapReduce framework for imbalanced big data', Fuzzy Sets and Systems 258, 5-38. URL: http://dx.doi.org/10.1016/j.fss.2014.01.015.

López, V., Río, S. D., José Manuel Bentez & Herrera, F. (2014), 'On the use of MapReduce to build Linguistic Fuzzy Rule Based Classification Systems for Big Data', IEEE International Conference on Fuzzy Systems (FUZZ-IEEE).

Morales, A. K. & Quezada, C. V. (1998), A universal eclectic genetic algorithm for constrained optimization, in 'Proceedings of the 6th European congress on intelligent techniques and soft computing', Vol. 1, Citeseer, pp. 518-522.

Mukkamala, R. R., Hussain, A. & Vatrapu, R. (2014), Fuzzy-set based sentiment analysis of big social data, in 'Enterprise Distributed Object Computing Conference (EDOC), 2014 IEEE 18th International', IEEE, pp. 71-80.

Parmee, I. C. (2012), Evolutionary and adaptive computing in engineering design, Springer Science & Business Media.

Robertson, S. P., Vatrapu, R. K. & Medina, R. (2010), 'Off the wall political discourse: Facebook use in the 2008 us presidential election', Information Polity 15(1, 2), 11-31.

Roebuck, K. (2011), Information Privacy: High-impact Strategies-What You Need to Know Definitions, Adoptions, Impact, Benefits, Maturity, Vendors, Tebbo.

Rosales-p, A. (2013), 'Surrogate-Assisted Multi-Objective Model Selection for Support Vector Machines'.

Rosales-Perez, A., Gonzalez, J. a., Coello Coello, C. a., Escalante, H. J. & Reyes-Garcia, C. a. (2014), 'Multi-objective model type selection', Neurocomputing 146, 83-94. URL: http://linkinghub.elsevier.com/retrieve/pii/S0925231214008789.

Sakr, S., Liu, A. & Fayoumi, A. G. (2013), 'The family of mapreduce and large-scale data processing systems', ACM Computing Surveys (CSUR) 46(1), 11.

Sánchez-Monedero, J., Gutiérrez, P. A., Pérez-Ortiz, M. & Hervas-Martínez, C. (2013), An n-spheres based synthetic data generator for supervised classification, in 'International Work-Conference on Artificial Neural Networks', Springer, pp. 613-621.

Thornton, C., Hutter, F., Hoos, H. H., Leyton-Brown, K. & Chris Thornton, Frank Hutter, Holger H. Hoos, K. L.-B. (2013), 'Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms', Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining pp. 847-855. URL: http://dl.acm.org/citation.cfm?id=2487629.

Tlili, M. & Hamdani, T. M. (2014), Big data clustering validity, in 'Soft Computing and Pattern Recognition (SoCPaR), 2014 6th International Conference of', IEEE, pp. 348-352.

Wu, X., Zhu, X., Wu, G.-Q. & Ding, W. (2014), 'Data mining with big data', IEEE transactions on knowledge and data engineering 26(1), 97-107.

Yu, K., Ji, L. & Zhang, X. (2002), 'Kernel nearest-neighbor algorithm', Neural Processing Letters 15(2), 147-156.

Zaharia, M., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., Venkataraman, S., Franklin, M. J. et al. (2016), 'Apache spark: a unified engine for big data processing', Communications of the ACM 59(11), 56-65.