Full Model Selection in Huge Datasets and for Proxy Models Construction

Angel Díaz-Pacheco(✉) and Carlos Alberto Reyes-García

Computer Science Department, Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE), Luis Enrique Erro No.1, Santa María Tonantzintla, 72840 San Andrés Cholula, Puebla, Mexico
diazpacheco@inaoep.mx

Abstract. Full Model Selection is a technique for improving the accuracy of machine learning algorithms through the search of the most adequate combination on each dataset of feature selection, data preparation, a machine learning algorithm and its hyper-parameters tuning. With the increasingly larger quantities of information generated in the world, the emergence of the paradigm known as Big Data has made possible the analysis of gigantic datasets in order to obtain useful information for science and business. Though Full Model Selection is a powerful tool, it has been poorly explored in the Big Data context, due to the vast search space and the elevated number of fitness evaluations of candidate models. In order to overcome this obstacle, we propose the use of proxy models in order to reduce the number of expensive fitness functions evaluations and also the use of the Full Model Selection paradigm in the construction of such proxy models.

Keywords: Big Data · Model Selection · Machine learning

1 Introduction

Data can be considered as an expenditure in storage or a valuable asset, this valuation relies on the analysis made to such information. One of the main tendencies to analyze data is the use of machine learning techniques though, to choose an adequate learning algorithm to a specific dataset is not a trivial task. This process requires to find the combination of algorithms together with their hyper-parameters to achieve the lowest misclassification rate in a wide search space [17]. Other factors that have a major impact in the generalization capacities of a classification algorithm are: feature selection and data-preparation. These factors, in combination with the selection of a classification algorithm integrates the Full Model Selection paradigm (FMS). Under this paradigm, every factor combination represents a set of data transformations and performing the learning process over the training set [6]. Although this paradigm is useful, has

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

been poorly explored in the Big Data context due the huge search space and the time every transformation and learning process takes in a dataset of this context. FMS has been addressed as an optimization problem varying the search technique employed. As an example, in [11] 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) was proposed. In [12] a genetic algorithm was employed for the hyper-parameter tuning of the SVM algorithm. In [6] they tackled and defined the full model selection problem with the use of a particle swarm optimization algorithm (PSO), in [9] a PSO algorithm was also used but just for hyper-parameter optimization for the ls-SVM algorithm meanwhile in [2] was proposed the use of the bat algorithm for solving FMS. Similar problems where the computing time is prohibitive, have been approached in the literature by means of proxy models also known as surrogate functions. A proxy model is a computationally inexpensive alternative to a full numerical simulation and can be defined as mathematically, statistically, or data-driven model defined function that replicates the simulation model output for selected input parameters [1]. Some of the main approaches analyzed can be grouped as follows: **(a) The surrogate function is based on a single regression algorithm to predict the fitness in the objective or objectives to optimize**. In [5] a neural network based on fuzzy granules was employed as a proxy model. In [15] a multi-objective genetic algorithm assisted by surrogate functions was proposed for model selection. A neural network with the parameters selected by hand was employed in order to predict each objective. Every generation, the surrogate predicts the performance of the individuals and the promissory ones were evaluated by the complete fitness function and the neural networks were re-trained with the new samples. **(b) Model selection is employed in order to increase the quality of the surrogate functions**. An island model genetic algorithm is employed in [4] for model selection and hyper-parameter optimization. The GA is combined with the Expected Improvement algorithm (EI) for the selection of the interest data points that can improve the performance of the surrogate model. In [14] the model selection is performed among several algorithms with their hyper-parameters previously configured by hand. From all the analyzed works we can see two fundamental aspects for the development of this work. The first one is that, the full model selection problem could be benefited by the use of surrogate models in order to reduce the high number of fitness evaluations during the search step. The second one is that, proxy models can be used as a way to reduce the fitness evaluations and also guide the search process as a compass, therefore, a wise decision is to built the best compass possible and a way to do that is through the FMS paradigm. This work has the following organization. In Sect. 2 we present some background on Big Data and MapReduce. Section 3 describes our proposed algorithm. Section 4 shows the experiments performed to test the validity of our proposal. Finally, Sect. 5 presents the conclusions.

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 analyse the large datasets. This programming model was designed to work over computing clusters and it works under the master-slave communication model. In MapReduce a computing task is specified as a sequence of stages: map, shuffle and reduce that works on a dataset $X = \{x_1, x_2, ..., x_n\}$. 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, $L_k = (k; v_1, v_2, ..., v_n)$ 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 $L_k = (k; v_1, v_2, ..., v_n)$, created during the shuffle step, to produce a set of values $y_1, y_2, ..., y_n$. The reduce function ρ is defined to work sequentially on L_k but should be independent of other lists L_k, where $k^{'} \neq k$ [7]. A widespread definition of Big Data describes this concept in terms of three characteristics of information in this field: Volume, Velocity and Variety [18] referring to the huge quantity, the high speed of generation and the different formats of the information. This definition does not provide rules to identify a dataset that belongs to Big Data and the justification of using a dataset in the literature only considers its size. The size of a dataset is relative to resources available to manage it and, in a country like Mexico (where this research was done), the availability of specialized hardware is an important limitation. Taking these factors into account, we propose an alternative definition of Big Data relative to the FMS problem. We propose that a huge dataset for the model selection problem must to 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 their file size considering the number of instances (I) and features (F) as long as $\text{I} \gg \text{F}$.

3 PSMS for FMS and Proxy Model Construction

One of the most popular and successful algorithms to perform the FMS analysis is the PSMS algorithm proposed by [6]. This algorithm is based on the PSO algorithm which is a population-based search inspired by the behavior of biological communities that exhibit both individual and social behavior [6]. PSMS is faster and easy to implement because relies in just one operator unlike the evolutionary algorithms. Due to the aforementioned reasons the PSMS algorithm was chosen and adapted to MapReduce in order to deal with datasets of any size.

3.1 Codification and Functioning

The solutions encoded in PSMS needs to be codified in a vector called particle. Each particle $x_i^t = [x_{i,1}^t, x_{i,2}^t, ..., x_{i,16}^t,]$ 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. At each time t, each particle, i, has a position in the search space. A set of particles $S = \{x_1^t, x_2^t, ..., x_m^t\}$ is called a swarm. Every particle has a related velocity value that is used to explore the search space and the velocity of such particle at time t is as follows $V_i^t = [v_{i,1}^t, v_{i,2}^t, ..., v_{i,16}^t]$ where $v_{i,k}^t$ is the velocity for dimension k of the particle i at time t. The search trajectories are adjusted employing the following equations:

$$v_{i,j}^{t+1} = W \times v_{i,j}^t + c1 \times r1 \times (p_{i,j} - x_{i,j}^t) + c2 \times r2 \times (p_{g,j} - x_{i,j}^t) \tag{1}$$

$$x_{i,j}^{t+1} = x_{i,j}^t + v_{i,j}^{t+1} \tag{2}$$

from the previous equations $p_{i,j}$ is the value in dimension j of the best solution found so far, also called personal best. $p_{g,j}$ is the value in dimension j of the best particle found so far in the swarm. Regarding to $c1, c2 \in \mathbb{R}$, are constants weighting the influence of local and global best solutions, and $r1, r2 \sim U[0, 1]$ are values that introduce randomness into the search process. The inertia weight W controls the impact of the past velocity of a particle over the current one, influencing the local and global exploration. As in the original paper, the inertia weight is adaptive and specified by the triplet $W = (w_{start}, w_f, w_{end})$; where w_{start} and w_{end} are the initial values of W, w_f indicates the fraction of iterations in which W is decreased. W is decreased by $W = W - w_{dec}$ from the first iteration where $W = W_{start}$ to the last iteration where $W = w_{end}$ and $w_{dec} = \frac{w_{start} - w_{end}}{Number\_of\_iterations}$ [6].

3.2 Models Evaluation

This version of PSMS was developed under Apache Spark 1.6.0, that is based on MapReduce. Apache Spark was selected 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). An analysis of the advantages of Spark over traditional MapReduce is out of the scope of this work, but we refer to [19]. The cornerstone of Apache Spark is the RDD or Resilent Distributed Dataset which is a collection of partitioned data elements that can be processed in parallel [8]. As described above, the models evaluation stage is comprised of data preparation, feature selection, and training of a classification algorithm, in the following algorithms this process is described.

Algorithm 1. Data preparation

```
1: procedure DataPrep(DataSet,particle)
2:     Return(DataSet.map(row → row.toArray.map(col → Transform(col,particle))))
3:   ▷ The Transform function is applied to every column of each row in the RDD according to the
       parameters encoded in the particle
4: end procedure
```

Algorithm 2. Feature Selection

```
1: procedure FeatSelection(DataSet,particle)
2:     numFeat = particle(9)
3:     rankRDD = DataSet.map(row → RankingCalculation(row))
4:   ▷ The RankingCalculation function obtains the ranking of the features of the dataset
5:     reducedRDD = rankRDD.map(row → getF(row,numFeat))
6:   ▷ The function getF is applied to every row in rankRDD and returns a reduced dataset
7:     Return(reducedRDD)
8: end procedure
```

Algorithm 3. Classification

```
1: procedure Classification(DataSet,particle)
2:     NumFolds = 2
3:     kFolds = createFold(DataSet,NumFolds)
4:   ▷ The createFold function creates an RDD for k-Fold Cross validation
5:     error=kFolds.map {
6:       case(Training,Validation)
7:   ▷ The dataset is separated in Training and Validation partitions
8:       model = createModel(Training, particle)
9:   ▷ The createModel function create a model using the parameters codified in the particle
10:      PredictedTargets = Validation.map(Instance → model.predict(Instance.features))
11:  ▷ Performs the predictions in the validation set
12:      accuracy= getAcc(PredictedTargets,Validation.targets)
13:  ▷ Obtains the accuracy in each fold
14:      error = 100-accuracy
15:      Return(error)
16:    }
17:    meanError=error.sum/error.length
18:    Return(meanError)
19: end procedure
```

In the proposed version of PSMS, 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 algorithm, 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. As to choose a single final model is not a trivial decision, another major change in our PSMS version was that the final model is a weighted ensemble of the best models found during the search process.

3.3 Proxy Model Construction Through the FMS Paradigm

As mentioned earlier, the search space of the FMS problem is huge even if restrictions are imposed to hyper-parameter values. The bio-inspired search methods have proved a high capacity to deal with this kind of problems and, particularly, PSMS has shown that is capable to solve the FMS problem. It is not easey to asses the complexity of the FMS problem, which varies from linear to quadratic or higher orders. The fitness function in a normal execution of PSMS should be evaluated $\rho = m \times (I + 1)$ times, where m is the swarm size and I the number of iterations. Supposing that the complexity of a model λ is bounded by λ_0, the complexity of PSMS will be bounded by $\rho \times \lambda_0$. The computing time of FMS is related to the dimensionality and size of the dataset, and when high volume datasets are analyzed, MapReduce allows to divide the load among MapReduce nodes. Although the complexity is the same $(\lambda_0 \times \rho)$, the computing time is reduced to manageable levels. An excellent alternative to reduce the computing time, is through the use of proxy models in order to reduce the value of ρ and to guide the search in a similar fashion of a compass. An effective way to built the best compass possible is through PSMS, to automate the selection of the best combination of regression algorithm, feature selection and data preparation of the meta-dataset. In this work, the meta-dataset, was built employing the particles evaluated by the PSMS algorithm for FMS in huge datasets. All the evaluated particles, are vectors that describe combinations of factors that represent a full model along with the particle performance, therefore, the meta-dataset constitutes a regression problem in order to predict the performance of new not analyzed particles and just to evaluate those that are promising. For the PSMS algorithm assisted by proxy models constructed with the FMS paradigm (onwards FMSProxy-PSMS), we used the regression and data-preparation algorithms available in the machine learning tool WEKA [10]. As feature selection algorithm the Principal Components Analysis (PCA) was employed because the available algorithms for FS need discrete classes in order to perform an analysis and of course the problem at hand is a regression problem with a continuous target. Finally the process of the FMSProxy-PSMS algorithm can be described as follows: (1) Evaluate all the particles in the first "N" iterations of the algorithm with the costly fitness function, (2) use this particles to build a first proxy model that will be used during the next "N" iterations, (3) evaluate the promising particles with the costly fitness function and discard the rest, (4) after complete "N" iterations build a new proxy model, (5) iterate until the termination criteria is met. All the particles evaluated with the costly fitness function are stored in a separated file used as meta-dataset.

4 Experiments and Results

With the purpose to evaluate the proposed algorithm performance, we experimented with the datasets shown in Table 1. The datasets "Synthetic 1" and "Synthetic 2" were created using the tool for synthetic datasets generation in the context of ordinal regression: "Synthetic Datasets Nspheres" provided in [16]. Despite of have been developed for ordinal regression, the tool can be properly adjusted for traditional binary or multi-class problems and provides the mechanism to control the overlaps and classes balance. Another major feature of the aforementioned datasets is its intrinsic dimension. The intrinsic dimension (ID) is the minimum number of parameters needed to represent the data without information loss [13]. The id of the employed datasets was estimated with the "Minimum neighbor distance estimator" (MNDE) [13] and the "Dimensionality from angle and norm concentration" (DANCO) estimator [3]. The importance of the estimation of the "id" of each dataset is to ensure that each dataset represent a different computational problem and, therefore, that proposed algorithm have the capability to deal with a wide range of problems and in the context of this work also with datasets of different sizes. In the Table 2 the calculated intrinsic dimension using the aforementioned estimators is shown.

Table 1. Datasets used in the experiments.

Datasets	Data points	Attributes	Samples by class	Type of variables	File size
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
Epsilon	500, 000	2000	(249778;250222)	Real	15.6 GB

Table 2. Intrinsic dimension of the datasets.

Datasets	MNDE	DANCO
RLCP	2	2
KDD	1	1
Synthetic 1	3	3
Higgs	12	15
Synthetic 2	22	28
Epsilon	160	78

The performance of the proposed approach FMSProxy-PSMS was contrasted against the complete search (PSMS) and a surrogate-assisted version of PSMS based on a Multi-layer perceptron (onwards MLP-PSMS) as in [15] whose hyper-parameters were determined with a grid search at the beginning of the process and during the rest of the search was just re-trained with the new samples of the meta-dataset. The termination criteria of PSMS was to complete 50 iterations, this in conjunction with a swarm size of 30 particles means that the search explored 1500 possible models before to build the final model. As mentioned earlier the proxy model approach can be though as a reduction of the expensive fitness functions or as a compass that can guide the search and, therefore, the best model must be find earlier in the search. With this in mind, as a termination criteria of the surrogate-assisted searches, the evaluation of a certain number of models was employed. In this case, the termination criteria was set to complete 500 evaluations with the expensive fitness function with no differences in the increment of this value (500, 1000, 1500 evaluations). In order to obtain an statistical power of 90% in an ANOVA test, 20 replications for every dataset were performed. Each replication was performed with a particular random sample of the data points with different random samples among replications. The dataset was divided in two disjoint datasets with 60% of the data samples for the training set and 40% for the test set. In Table 3, the mean error of the contrasted methods is shown.

Table 3. Mean classification error obtained in the test dataset by FMSProxy-PSMS, MLP-PSMS and PSMS, over 20 replications. The best results are in **bold**

Dataset	FMSProxy-PSMS	MLP-PSMS	PSMS
RLCP	0.059 ± 0.123	0.426 ± 1.593	**0.052 ± 0.001**
KDD	**0.079 ± 0.003**	2.556 ± 7.638	0.156 ± 0.134
Synthetic 1	15.865 ± 0.005	15.863 ± 0.004	**15.862 ± 0.004**
Higgs	30.193 ± 0.685	29.491 ± 2.220	**28.299 ± 0.057**
Synthetic 2	6.682 ± 0.003	6.682 ± 0.003	**6.681 ± 0.005**
Epsilon	**28.816 ± 2.762**	32.223 ± 5.397	54.008 ± 0.926

From previous table, the best method was PSMS, obtaining the lowest errors in the datasets RLCP, Synthetic 1, Higgs and Synthetic 2. The second best in the comparative was FMSProxy-PSMS in the datasets KDD and Epsilon. As was mentioned before, the complete search PSMS, explores 1500 models before to find the best model, and the surrogate-based methods only explores 500 models. In order to understand the search process of the methods under evaluation, Fig. 1 is provided.

The previous figure shows the approximate number of evaluation models in the "X" axis and the estimated error of the best particle found in the "Y" axis in Higgs dataset. It can be seen that PSMS got the lowest error with 39.2%, the second best was FMSProxy-PSMS with 42% and finally MLP-PSMS with 42.2%. Regarding to the number of evaluated models, MLP-PSMS was the fastest with 68 models evaluated until build the final model, FMSProxy-PSMS with 120 models and finally PSMS with 1220 evaluated models. Though, PSMS was the best in the comparative with a mean error of 28.299%, the surrogate-based searches were pretty near with 29.491% for MLP-PSMS and 30.193% for FMSProxy-PSMS with an smaller amount of models evaluations. Regarding to computing time factor, in Fig. 2, average execution times are shown as a bar chart in each dataset. The experiments were performed in a workstation of 12 threads with a Intel(R) Xeon(R) CPU E5-2695 at 2.40 GHz and 30 GB in RAM.

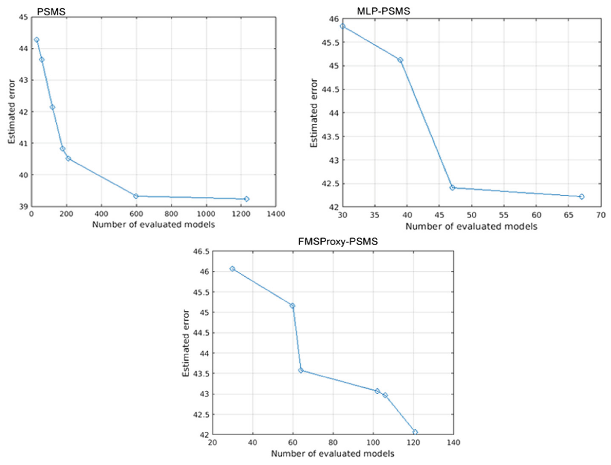

Fig. 1. Evolution of the search of the proposed methods in dataset Higgs. In X axis are the number of evaluated models (as a way to point out the progress of the search) and in Y axis the estimated error.

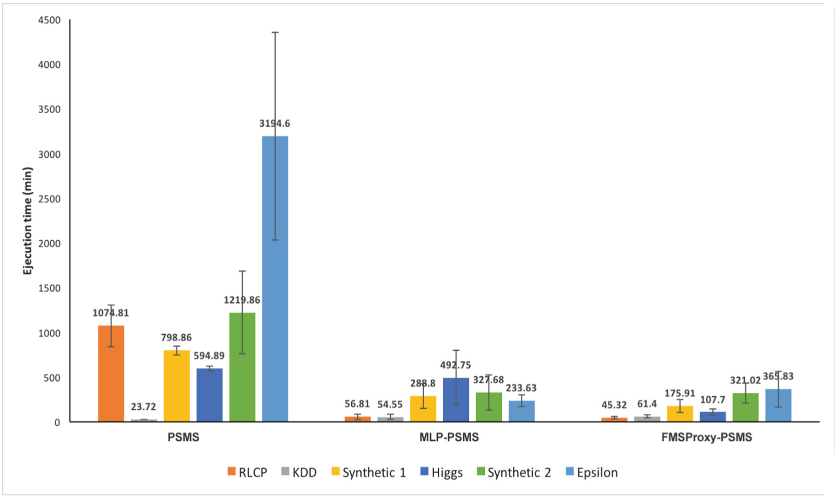

Fig. 2. Bar chart with the average execution time (in minutes) performed in 20 replications in each dataset. Each color represent a different dataset and bars are grouped by algorithm. The standard deviation is depicted as a solid black line on the bars. (Color figure online)

On Fig. 2 it can be appreciated the average execution times of contrasted algorithms with the aforementioned hardware configuration. With no surprise, highest execution times are for PSMS (the complete search), while surrogate-assisted approaches are considerably faster and specially for Epsilon dataset. Due its random nature (the initial swarm is randomly initialized), PSMS has also the largest standard deviation, but, taking into account that all the surrogated versions of PSMS have a random initial swarm too, we can conclude that a proxy model is capable to guide the search in an effective way. Concerning to standard deviation, a simpler visual analysis shows that FMSProxy-PSMS has a smaller standard deviation that traditional approach (MLP-PSMS). Finally, although computing times of FMSProxy-PSMS are no the smallest in all datasets, results in Table 3 shows that this approach is almost as good as the complete search but with moderate to low execution times. In order to find significant differences in the performance of the evaluated methods, an ANOVA test was performed. In the following table the results of the test are shown (Table 4).

Table 4. 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 (*)

Dataset	ANOVA F	FMSProxy-PSMS vs MLP-PSMS	FMSProxy-PSMS vs PSMS
RLCP	1.076	1.7785	0.035
KDD	2.039	2.511	0.079
Synthetic 1	1.629	2.070	2.328
Higgs	**10.189***	2.338	**6.313**
Synthetic 2	0.486	0.346	1.344
Epsilon	**298.013***	**4.303**	**31.819**

The statistic tests shows that there are almost no differences in the performance of the surrogate-based algorithms except in the Epsilon dataset where the lowest error was obtained by the FMSProxy-PSMS algorithm. Regarding to PSMS it can bee seen that exist significant differences in the Higgs dataset and the Epsilon dataset, with PSMS winning in the Higgs dataset and FMSProxy-PSMS winning in the Epsilon dataset. From Table 2 it can be seen that from the perspective of the intrinsic dimension analysis, the Epsilon dataset is the hardest one in the comparative with an ID of 160/78 against the one of Higgs with 12/15. Considering the previous analysis, we can see that the FMSProxy-PSMS algorithm is almost as good as PSMS in a wide range of dataset sizes and with different intrinsic dimensions. The exploration performed by the FMSProxy-PSMS algorithm is more conservative than the one performed by MLP-PSMS and therefore, the guide provided by the proxy models created under the FMS paradigm promote a better exploration of the search space that leads to better final models than using a single regression algorithm as proxy model. The mean error obtained by FMSProxy-PSMS in the dataset Epsilon shows its capacity to perform a better exploration than the complete search of PSMS with just a third part of the explored models (500 against 1500). Though the creation of a new proxy model based on the FMS paradigm every so often could be think as a major drawback of our approach, the time employed in the process is nothing in comparison to the transformation and training on a big dataset. The time dedicated to the training and transformation of the dataset of just one model commonly take several hours even under MapReduce and considering that a good search explores several potential models, the time of the entire process takes a lot more. A final consideration of our proposed approach shows that although the traditional approach of proxy model makes a quicker search, the FMS approach reduces the time employed and also perform a better exploration.

5 Conclusions

The full model selection paradigm provides a way to improve the predictive accuracy of learning algorithms and to determine the best one for a determined dataset, which is of big help because there is no a thumb rule to chose an algorithm for a dataset. This paradigm is poorly explored in the context of huge datasets due to the higher computing times intrinsically related to the size of the dataset. As a way to perform a reduction of the time employed in the search, the use of proxy models in conjunction with the FMS paradigm was proposed in this work. The use of the FMS paradigm in the construction of proxy models showed an improvement over traditional proxy models based on a single regression algorithm, performing a better exploration of the search space and reducing the number of the explored models to a third part in comparison to the complete search. Though the construction of FMS-based proxy models every so often could be think as a major drawback of our approach, it is not hard to see that the time employed is insignificant in comparison to the time employed in the training in a big dataset of several non promising models that will not be used in the final model construction. Our approach provides a tool comparable to a compass to provide a better guide of the search and reducing the time of the process.

References

1. Alenezi, F., Mohaghegh, S.: A data-driven smart proxy model for a comprehensive reservoir simulation. In: Saudi International Conference on Information Technology (Big Data Analysis) (KACSTIT), pp. 1–6. IEEE (2016)
2. Bansal, B., Sahoo, A.: Full model selection using bat algorithm. In: 2015 International Conference on Cognitive Computing and Information Processing (CCIP), pp. 1–4. IEEE (2015)
3. Ceruti, C., Bassis, S., Rozza, A., Lombardi, G., Casiraghi, E., Campadelli, P.: DANCo: dimensionality from angle and norm concentration. arXiv preprint arXiv:1206.3881 (2012)
4. Couckuyt, I., De Turck, F., Dhaene, T., Gorissen, D.: Automatic surrogate modeltype selection during the optimization of expensive black-box problems. In: Proceedings of the 2011 Winter Simulation Conference (WSC), pp. 4269–4279. IEEE (2011)

5. Cruz-Vega, I., García, C.A.R., Gil, P.G., Cortés, J.M.R., Magdaleno, J.d.J.R.: Genetic algorithms based on a granular surrogate model and fuzzy aptitude functions. In: 2016 IEEE Congress on Evolutionary Computation (CEC), pp. 2122–2128. IEEE (2016)
6. Escalante, H.J., Montes, M., Sucar, L.E.: Particle swarm model selection. J. Mach. Learn. Res. **10**(Feb), 405–440 (2009)
7. Goodrich, M.T., Sitchinava, N., Zhang, Q.: Sorting, searching, and simulation in the mapreduce framework. In: Asano, T., Nakano, S., Okamoto, Y., Watanabe, O. (eds.) ISAAC 2011. LNCS, vol. 7074, pp. 374–383. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25591-5_39
8. Guller, M.: Big Data Analytics with Spark: A Practitioner's Guide to Using Spark for Large Scale Data Analysis. Apress, New York City (2015). https://www.apress.com/9781484209653
9. Guo, X., Yang, J., Wu, C., Wang, C., Liang, Y.: A novel LS-SVMs hyper-parameter selection based on particle swarm optimization. Neurocomputing **71**(16), 3211–3215 (2008)
10. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA data mining software: an update. SIGKDD Explor. **11**(1), 10–18 (2009)
11. Kaneko, H., Funatsu, K.: Fast optimization of hyperparameters for support vector regression models with highly predictive ability. Chemom. Intell. Lab. Syst. **142**, 64–69 (2015). https://doi.org/10.1016/j.chemolab.2015.01.001, http://linkinghub.elsevier.com/retrieve/pii/S0169743915000039
12. Lessmann, S., Stahlbock, R., Crone, S.F.: Genetic algorithms for support vector machine model selection. In: 2006 International Joint Conference on Neural Networks. IJCNN 2006, pp. 3063–3069. IEEE (2006)
13. Lombardi, G., Rozza, A., Ceruti, C., Casiraghi, E., Campadelli, P.: Minimum neighbor distance estimators of intrinsic dimension. In: Gunopulos, D., Hofmann, T., Malerba, D., Vazirgiannis, M. (eds.) ECML PKDD 2011. LNCS (LNAI), vol. 6912, pp. 374–389. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23783-6_24
14. Pilat, M., Neruda, R.: Meta-learning and model selection in multi-objective evolutionary algorithms. In: 2012 11th International Conference on Machine Learning and Applications (ICMLA), vol. 1, pp. 433–438. IEEE (2012)
15. Rosales-Pérez, A., Gonzalez, J.A., Coello Coello, C.A., Escalante, H.J., Reyes-Garcia, C.A.: Surrogate-assisted multi-objective model selection for support vector machines. J. Neurocomputing **150**, 163–172 (2015)
16. Sánchez-Monedero, J., Gutiérrez, P.A., Pérez-Ortiz, M., Hervás-Martínez, C.: An n-spheres based synthetic data generator for supervised classification. In: Rojas, I., Joya, G., Gabestany, J. (eds.) IWANN 2013. LNCS, vol. 7902, pp. 613–621. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38679-4_62
17. Thornton, C., Hutter, F., Hoos, H.H., Leyton-Brown, K.: Auto-WEKA: combinedselection and hyperparameter optimization of classification algorithms. In: Proceedings of the 19th ACM SIGKDD International Conference on Knowledgediscovery and Data Mining, pp. 847–855. ACM (2013)
18. Tlili, M., Hamdani, T.M.: Big data clustering validity. In: 2014 6th International Conference of Soft Computing and Pattern Recognition (SoCPaR), pp. 348–352. IEEE (2014)
19. Zaharia, M., et al.: Apache spark: a unified engine for big data processing. Commun. ACM **59**(11), 56–65 (2016)