The impact of data selection strategies on distributed model performance

Miguel Guimarães[1 [0000−0003−0573−9122]], Filipe Oliveira[1 [0000−0002−2424−5024]], Davide Carneiro[1 [0000−0002−6650−0388]] and Paulo Novais[2 [0000−0002−3549−0754]]

[1] CIICESI, Escola Superior de Tecnologia e Gestão, Instituto Politécnico do Porto
mam@estg.ipp.pt, 8170164@estg.ipp.pt, dcarneiro@estg.ipp.pt
[2] ALGORITMI Research Centre/LASI, University of Minho, Braga, Portugal
pjon@di.uminho.pt

Abstract. Distributed Machine Learning, in which data and learning tasks are scattered across a cluster of computers, is one of the answers of the field to the challenges posed by Big Data. Still, in an era in which data abounds, decisions must still be made regarding which specific data to use on the training of the model, either because the amount of available data is simply too large, or because the training time or complexity of the model must be kept low. Typical approaches include, for example, selection based on data freshness. However, old data are not necessarily outdated and might still contain relevant patterns. Likewise, relying only on recent data may significantly decrease data diversity and representativity, and decrease model quality. The goal of this paper is to compare different heuristics for selecting data in a distributed Machine Learning scenario. Specifically, we ascertain whether selecting data based on their characteristics (meta-features), and optimizing for maximum diversity, improves model quality while, eventually, allowing to reduce model complexity. This will allow to develop more informed data selection strategies in distributed settings, in which the criteria are not only the location of the data or the state of each node in the cluster, but also include intrinsic and relevant characteristics of the data.

Keywords: Machine Learning · Distributed Computing · Data Selection.

1 Introduction

New approaches to Machine Learning (ML), such as Distributed ML [9] or Federated ML [5], emerged in the last years as a response to the challenges of learning from Big Data [11].

The appearance of Distributed ML followed a previous trend, in which storage was moved from large monolithic data architectures to distributed mesh ones [6]. Evidently, once data were distributed, it would not be feasible or possible to move all the data to a single location in order to run learning tasks. Hence, learning tasks became parallel and distributed, starting to include coordination and/or message passing mechanisms such as ZooKeeper [3] or ZeroMQ [4].

Even so, decisions on which data to use when training a ML model must often still be made, be it because the data are simply too much, or because there are restrictions on training time or model complexity.

In the past we have proposed approaches to deal with this issue [7], namely by selecting the data (and running the distributed learning tasks) in the nodes in which the data are located, and according to the state of each node. Given that modern distributed file systems have some form of replication, this can be explored to select from the best candidate nodes according to their state.

Other typical approaches include selecting data based on their freshness. That is, more recent data will be more likely to be selected. While this is, at first sight, intuitive, it might not always hold the best results in terms of model quality. This happens because old data might describe patterns that are still relevant and useful, as happens in scenarios of streaming data with recurring concept drift (i.e. seasonality) [10], and that would otherwise be lost.

The main goal of this paper is to analyze the influence that different data selection strategies have on the quality and complexity of ML models. Specifically, we aim to ascertain whether selecting data based on a diversity criteria allows to obtain better models, or smaller but equivalent models (in terms of predictive performance) than when the full set of data are used. This will allow to devise new strategies of data selection, based not only on data locality, data availability and cluster state, but that also take into account intrinsic characteristics of the data, in an attempt to select data that are as rich and diverse as possible. This will contribute to the design of overall more efficient ML pipelines.

2 Distributed Machine Learning

The experiments described in this paper, in which a large number of ML Ensembles were trained, were carried out in the CEDEs platform. CEDEs - Continuously Evolving Distributed Ensembles - is a Distributed Learning platform developed in the context of a funded research project.

This section outlines the major modules comprising the architecture of CEDEs. The Storage Layer serves as the central element, implemented on top of an HDFS cluster. In this file system, large datasets are broken down into fixed-size blocks, usually of 128MB. Various data sources, including files in different formats (e.g., CSV, Parquet, Avro) and databases, can be employed.

Thus, the user of CEDEs initiates a ML project by uploading a data source into the HDFS, which is subsequently split, distributed, and replicated throughout the cluster. Once this process is concluded, the user can create an actual ML project by choosing the desired dataset, and supplying the relevant information such as project name, description and owners, or base model(s) hyperparameters. After configuring the project, the user can begin training a new Ensemble. This task is controlled by the Coordination module, which interacts with the Optimization and Metadata modules.

Training an ensemble occurs in a distributed manner and benefits from two aspects: the replication of blocks within files, enabling them to be accessed con-

currently on multiple nodes, and the principle of data locality. To train an ensemble, one base model is trained for each block of a file (excluding replicas). These base models will collectively form the Ensemble. Consequently, it is essential to determine the optimal node for training each base model, based on criteria such as the state of the node where the block is available within the cluster.

Specifically, the Coordination module uses the following inputs:

- Block locations are used to know where in the cluster each individual block exists. Generally, each block is simultaneously available in at least 3 nodes;
- Task cost prediction, in terms of time and computational resources (e.g. RAM). This is given by a previously developed meta-learning module [2];
- The state of nodes, in terms of the makespan of their tasks and a prediction of the necessary computational resources;

The goal of this paper, as elaborated further below, is to determine whether the characteristics of the data blocks can also be used as a relevant optimization criteria when selecting the blocks for training a new Ensemble.

Once the Coordination module defines the allocation of tasks, the process of training the base models initiates. In order to coordinate the worker nodes, a communication protocol was implemented in ZeroMQ: a brokerless asynchronous messaging library that allows the implementation of multiple socket communication patterns, useful for implementing distributed systems.

During the distributed training process, worker nodes regularly report their progress to the coordinator, allowing client applications to access this information in real time. When a worker node finishes training a base model, it notifies the Coordinator, which then moves on to the next task in its queue, if it exists. Additionally, the base model is serialized and stored in the HDFS, and automatically replicated and distributed across the cluster, based on the replication and balancing factors defined in the configuration. This allows to serve the same base model on multiple nodes shortly after the end of its training, significantly increasing model availability and the robustness of the system.

This architecture is illustrated in Figure 1, in which the following terminology is used: $F_iB_jR_k$ represents a replica k of block j of file i. The names of base models follow the same logic, but start with the letter 'M'. The arrow between a specific block replica and a model means that the base model was created based on that specific block, on that node. Replicas of that base model may however exist in other nodes, created through replication.

When a client requests predictions from an ensemble, a similar process to that of training base models takes place. The Coordination module first queries the Optimization module for information about which base models will be used in the ensemble. Indeed, not all base models may be included, depending on the specific project configuration. If the number of intended base models is smaller than the number of blocks in the dataset, the Optimization module selects the best group of base models based on a specific heuristic, that may include models' performance metrics and the state of the nodes where they are located. The main goal of this work is to check whether including a criteria of data diversity improves the quality and decreases the necessity for data, of the Ensemble.

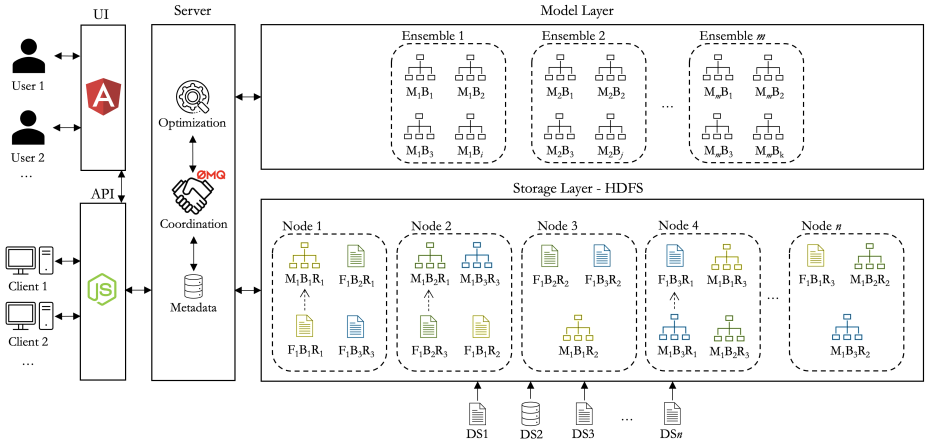

Fig. 1. Depiction of the CEDEs architecture.

When the Ensemble is defined, the Coordinator requests all the corresponding worker nodes for predictions for a specific dataset. Once all predictions are received, they are combined to calculate the final predictions by the ensemble. For regression problems, the mean is used, while for classification problems, the mode is used. Ensembles are thus abstract and dynamic constructs, as depicted in Figure 1 in the Model Layer, built out of a selection of available base models for a given ML problem.

The architecture includes two more modules: the API module, and the User Interface module. The API module provides endpoints for interacting with the system's services, which in turn communicate with the Coordination module, when applicable, via ZeroMQ messages. External client applications can use this API. Finally, the User Interface module is intended for human users and communicates with the API module.

3 Methodology

This work analyzes how different data selection strategies impact the performance of a distributed Machine Learning model, implemented as an Ensemble, with the goal of determining the strategy that produces the best model with the least amount of data and base models.

To this end, 60 different Machine Learning experiments were run. Each experiment, described below, refers to a publicly available dataset (from those described in Table 1), one block size (i.e. 4MB, 8MB and 16MB) and one of five different heuristics for buillding the Ensemble. The datasets are all traditional tabular datasets, with the exception of the "covid infections" dataset, which is a time-series one.

The interest in experimenting with different block sizes lies in its potential influence on the quality of the base models: in principle, the smaller the block,

the lower and less reliable the quality of the base models (due to significant variations between blocks and the data not being representative of the whole problem). Thus, in each scenario, CEDEs was restarted and configured with different block sizes to implement the experiments described.

Each dataset was first split into training (75%) and testing (25%) sets. The training data was uploaded into CEDEs, as described below, while the test data was held-out to assess the generalization ability of the Ensembles created.

Table 1. Characterization of the datasets used.

Dataset	Rows	Columns	Size
covid infections	4623663	19	585 MB
solar production	2782080	15	399 MB
temperature	2434608	20	351 MB
car prices	1957675	1108	4160 MB

The methodology followed in each experiment, which is also depicted in Figure 2, was as follows. CEDEs was configured with one of the tested block sizes (i.e. 4MB, 8MB and 16MB) and one of the datasets was uploaded into CEDEs. The dataset was then automatically split into blocks, distributed and replicated. For some datasets, some data preparation tasks were executed (e.g. cleaning, feature engineering, encoding), which were implemented in Apache Spark. However, given that the goal is not at this stage to find the best model possible but only to compare different approaches, no special effort was put on a further improvement of the dataset.

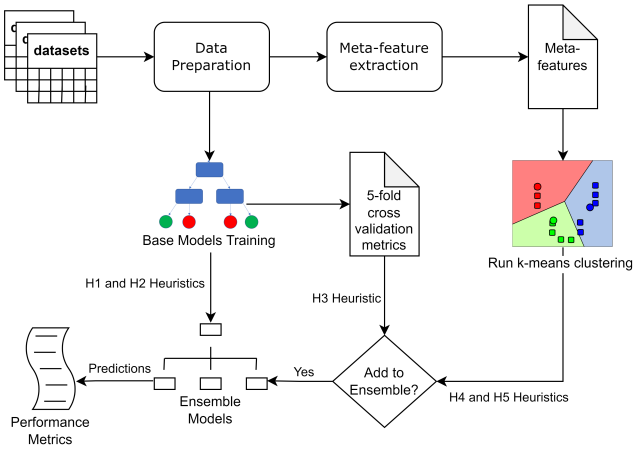

Fig. 2. Methodology followed in each of the experiments.

Once the dataset was pre-processed, the meta-features were extracted for each block. This was achieved using the pymfe - Python Meta-Feature Extractor library[8]. Meta-features are features that encode characteristics of an underlying dataset and may represent many different dimensions such as data distribution measures, data quality measures, measures of variability, other statistical measures, etc.

Once meta-features were extracted for each block, a base model was also trained for each block and, as usually happens in CEDEs, their performance metrics (e.g. RMSE, MAE, MSE, Logloss) recorded in a database.

Next, a k-means algorithm was run on the meta-features dataset. The goal was to find clusters of blocks that share similar characteristics (meta-features), thus allowing to select data (and consequently blocks) also based on their similarity. K-means was run with $k = 5$, thus it always finds 5 groups of datasets, based on their meta-features.

Then, for each dataset and block size, 5 different Ensembles were trained, with the following heuristics being used for base-model selection:

- H_1 - The Ensemble is created following a Bagging approach, with all the base models being considered. However, each base model has a weight that is inversely proportional to its quality, given by the RMSE calculated through 5-fold cross-validation.
- H_2 - The Ensemble is created following a traditional Bagging approach, with all the base models being used, and each having equal weight.
- H_3 - The Ensemble is created by selecting only the top 50% base models, and with all selected models having the same weight. Models are ranked by their RMSE, calculated through 5-fold cross-validation.
- H_4 - The Ensemble is created by picking one random model from each of the 5 clusters. The goal of this heuristic is to maximize data diversity, by picking models trained on blocks with different characteristics.
- H_5 - The Ensemble is created by picking 5 random models from the cluster that has the highest density, i.e. the datasets have more in common or share similar traits, according to the meta-features.

Although CEDEs allows to build heterogeneous Ensembles, for the work described in this paper, all the Ensembles trained are Random Forests, i.e., collections of Decision Trees. For the sake of fairness, all Decision Trees were trained with the same configuration: max_depth=25, min_samples_split=2, max_leaf_nodes=infinite and ccp_alpha=0.5

Each Ensemble was then evaluated both through 5-fold cross validation and on the test data that was held-out. The results of this evaluation and of comparing the different heuristics are analyzed in Section 4.

4 Results

This section describes the results obtained while comparing the 5 previously mentioned heuristics. The experiments were run in a CEDEs instance, that was

configured with 3 different block sizes. The goal is to analyze the impact of the data selection heuristic on the quality of the ensemble, and whether block size affects it.

Specifically, we want to evaluate the proposed heuristic (H_4), and determine whether it produces simpler but equally good ensembles, independently of the number of blocks of the original dataset. This selection is based on the diversity of the data, through the extraction of meta-features [1]. Specifically, meta-features were considered block-wise. As previously mentioned, in this heuristic, data blocks are selected based on their belonging to a different cluster. This means that under H_4, ensembles always have 5 models. The same is true for H_5. However, in this case, the blocks are all obtained from the same cluster.

The analysis carried out was extensive, taking into consideration several factors including block size (x3), dataset (x4) and heuristics (x5). Moreover, we looked at several indicators including the number of base models whose training was spared using H_4 and H_5, the amount of CPU time saved, and model performance metrics (e.g. RMSE). For this reason, this section presents only a summary of the results.

Table 2 presents some results with the 'solar production' dataset. Here we simply compare our proposed approach H_4 with a traditional bagging one H_2. The proposed approach only outperforms the traditional one in one scenario (8MB block size), in which the ensemble build with 5 diverse data blocks has an RMSE that is 0.58% lower. In the other two scenarios (4MB and 16MB), the proposed approach results in Ensembles that are slightly worse in terms of RMSE (0.11% and 0.3% worse, respectively).

While this lower performance is quite marginal, the ensembles build clearly outperform the traditional ones in terms of the number of models trained or the time saved. In this case, the number of models trained is between ≈80% and ≈95% smaller, depending on the block size, which also results in a reduction of ≈80% to ≈95% of the ensemble training time.

Table 2. Summary of the results for the 'solar production' dataset with ensembles trained with 4, 8 and 16 MB block size. **M** - the number of models whose training was avoided (%); **T** - CPU time saved (%); **R** - difference in RMSE (%).

Block Size	Heuristic	M	T	R
4 MB	$\boldsymbol{H_2}$ vs H_4	-94.79%	-94.81%	+0.11%
8 MB	H_2 vs $\boldsymbol{H_4}$	-89.56%	-89.53%	**-0.58%**
16 MB	$\boldsymbol{H_2}$ vs H_4	-79.17%	79.38%	+0.3%

Tables 3 and 4 provide the remaining insights in terms of the analysis carried out. Table 3 contains an overall analysis, concerning all the experiments carried out. In what concerns heuristics H_1 to H_3, 618 base models were trained in total, for each one. This includes the 3 different block sizes and the 4 different datasets.

The only difference is that the Ensembles built with H_3 only use half the base models: the half with the best RMSE. However, these base models need to be trained to calculate their RMSE, the number is the same.

On the other hand, H_4 and H_5 only need the training of 5 base models per Ensemble, for the 12 experiments made (4 datasets * 3 block sizes), which results in 60 base models trained in total. This means that these heuristics require 558 fewer models each, in comparison with the other 3. Following the same rationale, these heuristics avoided approximately 3790 seconds of CPU time in the training of models, and approximately 2409 GB of used RAM (measured at half-second interval). In this table, the columns TR and RR represent the reduction in terms of CPU time and RAM, when compared to traditional approaches.

Table 3. Compiled summary of the total results containing the four datasets with models trained in 4, 8 and 16 MB block size. **TM** - total number of models trained; **TT** - total training time (seconds); **TR** - total time reduction (seconds); **UR** - used RAM (GB); **RR** - RAM reduction (GB).

Heuristic	**TM**	**TT**	**TR**	**UR**	**RR**
H_1	618	4561.99	-	2680.82	-
H_2	618	4561.99	-	2680.82	-
H_3	618	4561.99	-	2680.82	-
H_4	60	771.24	3790.74	272	2409
H_5	60	765.37	3796.62	268	2413

Table 4, on the other hand, shows the average values for each experiment performed, grouped by heuristic. For each experiment the results were ranked, and analyzed relatively. For instance, for each experiment the best heuristic in terms of RMSE was identified. The remaining were ranked by decreasing RMSE, and their RMSE was analyzed relatively to the lowest one. Thus, column AR depicts how much worse the RMSE of a given heuristic was, in percentage. The results were then averaged.

The table shows that the proposed heuristic H_4 is the one that has the lowest value in this indicator, which means that it was most frequently the best approach in terms of predictor performance.

On average, the proposed approach manages to obtain the best performance in terms of RMSE, and saving training resources, both in terms of CPU time and of memory used, saving on average approximately 85% of models per ensemble: corresponding to $\approx$5 minutes of training time avoided and $\approx$201 GB spared RAM usage. This was the difference of training an Ensemble in only one minute (heuristics H_4 and H_5) instead of 6 minutes (heuristics H_1 to H_3). However, this difference might be much bigger in real scenarios with the typical 128MB block size and with even bigger datasets.

Thus, heuristic H_4 results in ensembles that train quicker and achieve better performance most of the times, although not always. Nonetheless, even when

Table 4. Compiled summary of the mean results considering the 4 datasets with models trained with 4, 8 and 16 MB block sizes. **AM** - average number of avoided models (%); **AR** - average difference in RMSE compared with the best ensemble (%); **TT** - average training time (s); **TR** - average time reduction (s); **UR** - used RAM (GB); **RR** - average RAM reduction (GB).

Heuristic	AM	AR	TT	AT	UR	RR
H_1	0%	-6.480%	380.17	-	223.40	-
H_2	0%	-6.680%	380.17	-	223.40	-
H_3	0%	-3.310%	380.17	-	223.40	-
$\boldsymbol{H_4}$	85.16%	**-2.710%**	64.27	315.90	22.64	200.76
H_5	85.16%	-3.280%	63.78	316.39	22.31	201.09

the performance is not the best, it is close to it. In opposition, the amount of necessary resources and the complexity of the resulting ensemble is far smaller, which is a significant advantage, especially in Big Data and streaming scenarios.

The proposed heuristic for data selection is thus one that might sometimes result in slightly worse model performance, but one that significantly reduces training time and computational resources.

5 Conclusions and Future Work

As is widely accepted, data and their quality are a key determinant of model quality. While in some cases all the available data is used, in others a subset of the data must be selected. This happens when the available data is too much, or when there are strict requirements concerning model training time or model complexity. In such cases, the strategies used for selecting data must be carefully considered, to maximize the quality of the model.

In this paper we analyze 5 different heuristics for selecting data. Two of them are novel approaches, and are based on meta-features, that is, characteristics of the data. Our goal was to ascertain whether selecting a smaller number of subsets of data but while maximizing diversity, would result in better ensembles. The results show not only this is generally true for the cases tested, but also that the resulting Ensembles are much smaller. Moreover, we avoid the training of a significant amount of models, which results in significant savings in terms of computational resources and time.

There are, however, limitations to this work. One of the major ones is that each experiment was only run once. Given that a methodology based on a holdout dataset was used, the split of data and the results are dependent on luck. In future work we will run the same experiments with cross-validation, to have more reliable results (cross-validation was used but only at the level of the base models). Moreover, in order to reduce variability in the results, a single algorithm was used (Decision Trees). In future work we will repeat the experiment with other algorithms, in far more extensive experiments. Also, the number of clusters (5) was arbitrarily selected and is, most likely, not the best one for each problem.

In the future, a prior heuristic for selecting the optimum value of k (number of clusters) will be implemented.

Finally, the findings of this work will be included in the optimization module of CEDEs, so that it can also take into account the characteristics of the data when deciding which blocks to use.

6 Acknowledgments

This work has been supported by national funds through FCT – Fundação para a Ciência e Tecnologia through projects UIDB/04728/2020, EXPL/CCI-COM/0706/2021 and CPCA-IAC/AV/475278/2022.

References

1. Alcobaça, E., Siqueira, F., Rivolli, A., Garcia, L.P.F., Oliva, J.T., de Carvalho, A.C.P.L.F.: Mfe: Towards reproducible meta-feature extraction. Journal of Machine Learning Research **21**(111), 1–5 (2020), http://jmlr.org/papers/v21/19-348.html
2. Guimarães, M., Carneiro, D., Palumbo, G., Oliveira, F., Oliveira, Ó., Alves, V., Novais, P.: Predicting model training time to optimize distributed machine learning applications. Electronics **12**(4), 871 (2023)
3. Hunt, P., Konar, M., Junqueira, F.P., Reed, B.: Zookeeper: wait-free coordination for internet-scale systems. In: USENIX annual technical conference. vol. 8 (2010)
4. Lauener, J., Sliwinski, W., CERN, G.: How to design & implement a modern communication middleware based on zeromq. In: Proc of ICALEPCS. vol. 17, pp. 45–51 (2017)
5. Liu, J., Huang, J., Zhou, Y., Li, X., Ji, S., Xiong, H., Dou, D.: From distributed machine learning to federated learning: A survey. Knowledge and Information Systems **64**(4), 885–917 (2022)
6. Machado, I.A., Costa, C., Santos, M.Y.: Data mesh: concepts and principles of a paradigm shift in data architectures. Procedia Computer Science **196**, 263–271 (2022)
7. Monteiro, J., Oliveira, O., Carneiro, D.: Task scheduling with makespan minimization for distributed machine learning ensembles. In: 2022 IEEE 4th Eurasia Conference on IOT, Communication and Engineering (ECICE). pp. 435–438 (2022). https://doi.org/10.1109/ECICE55674.2022.10042894
8. Rivolli, A., Garcia, L.P., Soares, C., Vanschoren, J., de Carvalho, A.C.: Characterizing classification datasets: a study of meta-features for meta-learning. arXiv preprint arXiv:1808.10406 (2018)
9. Verbraeken, J., Wolting, M., Katzy, J., Kloppenburg, J., Verbelen, T., Rellermeyer, J.S.: A survey on distributed machine learning. Acm computing surveys (csur) **53**(2), 1–33 (2020)
10. Wang, H., Abraham, Z.: Concept drift detection for streaming data. In: 2015 international joint conference on neural networks (IJCNN). pp. 1–9. IEEE (2015)
11. Zhou, L., Pan, S., Wang, J., Vasilakos, A.V.: Machine learning on big data: Opportunities and challenges. Neurocomputing **237**, 350–361 (2017)