Conceptualization of an Autonomic Machine Learning Platform for Non-Expert Developers

KEON MYUNG LEE[1], JAESOO YOO[2], JIMAN HONG[3]
[1]Dept of Computer Science,
Chungbuk National University,
Cheongju, KOREA
kmlee@cbnu.ac.kr
[2]School of Information and Communication Engineering,
Chungbuk National University,
Cheongju, KOREA
yjs@cbnu.ac.kr
[3]School of Computer Science and Engineering,
Soongsil University,
Seoul, KOREA
jiman@ssu.ac.kr

Abstract: - Machine learning is an approach to develop some algorithm for problem solving from data of the problem domain without coding programs. Although there are various machine learning tools with which machine learning applications can be developed relatively easily, non-experts have yet difficulties in developing machine learning applications. To be a successful developer, it is required to understand machine learning algorithms and to make right design choices. This paper addresses the decision choices to be made and which tasks need to be automated by the platform for non-expert developers to get an effective and efficient machine learning application. It presents the autonomicity levels which specify the level of automation in machine learning application development. It describes the requirements of an autonomic machine learning platform which helps non-expert developers build a machine learning application. It also introduces an architecture of an autonomic machine learning platform.

Key-Words: - machine learning, distributed computing, autonomic computing, machine learning platform

1 Introduction

Machine learning techniques have been recently made great progress with the invention of new excellent machine learning algorithms like deep learning and the availability of massive data and computing resources. ML algorithms are very effective in extracting knowledge in the form of patterns or models from a collection of data. Various machine learning algorithms have been developed and in use for various tasks[1-16]. The adoption of machine learning could increase productivity, quality, or profit by saving much cost and labour in various domains.

Excellent machine learning tools have been developed and available in public even in open software or open source software.[1-4] Machine learning takes experts to be successfully applied to a specific problem due to the following reasons: First, appropriate machine learning task should be identified to the problem. The tasks include classification, regression, clustering, recommendation, density estimation, dimensionality reduction, feature representation, and so on. Second, it is of paramount importance to use proper features by selecting or extracting features from data sets with various attributes. Third, there are many candidate machine learning algorithms for a specific task, each of which has some advantages or disadvantages over others. Fourth, most machine learning algorithms have some hyperparameters which are a kind of parameters that influence on the algorithm's behaviour. The existence of hyperparameters requires the engineering work for selecting their proper values. With some insight of experts and mainly generate-and-test manner, many hyperparameter combinations should be examined by executing the employed machine learning algorithms with the hyperparameter values.

Recent progresses in machine learning, especially in deep learning and probabilistic graphical models, take considerable amount of computations due to their model complexity, i.e., the number of parameters in a model. Machine learning algorithms with hyperparameters take considerable computing resources for training and inference.[6-11] External computing resources such as cloud services or distributed computing systems allow us to harness the computing power for machine learning service systems.[12,13,14] As mentioned above, however, the well-trained human experts on machine learning is a rare resource to take. Even though there are many open machine learning tools are available, most sectors in industry, business, and government have difficulty in adopting machine learning techniques to their work because they do not have such machine learning experts.

There has been some work to automate machine learning tasks with less human involvement. [7,8,9,10] This paper addresses a new platform for machine learning which automates machine learning for a given data set and uses the external computing resources. The platform works in an autonomic manner of which architecture is organized so that some components take care of machine learning planning, some components manage executing of specific machine learning task, some components evaluate the learned models, and others take care of distributed computing resource management.

The remainder of the paper is organized as follows: Section 2 presents some related work to machine learning automation. Section 3 presents the design choices made by the machine learning application developers which make non-expert successfully develop machine learning application, the requirements of an autonomic machine learning platform. It also presents an architecture of an autonomic machine learning platform. It also describes the functionalities of component modules in the platform. Section 4 presents the implement issues and the current progress of the platform development. Finally, Section 5 draws the conclusions.

2 Related Work

There have been some efforts to automatically determine hyperparameters in machine algorithm algorithms. Grid method[12] generates candidate hyperparameter combinations each of which hyperparameters is equally spaced in its domain, and chooses a combination which gives the best performance by executing the corresponding ML algorithm for each candidate. Random method[9] generates candidate hyperparameter combinations randomly which gives empirically better chances to find better hyperparameters within the given computation budget. Sequential model-based algorithm configuration method [7] selects a hyperparameter combination based on a model rather than uniformly at random. Tree-structured Parzen estimator method [10] sequentially constructs models to approximate the performance of hyperparameters based on historical measurements, and then subsequently chooses new hyperparameters to test based on this model. Gaussian process-based method [6] is also used, which uses a Gaussian process as a surrogate for hyperparameter distributions and updates the Gaussian process by combining sampling results and prior distribution.

Due to considerable computation demands, there have been various works on distributed processing and parallel processing in machine learning. OptiML, GraphLab, SystemML, SimSQL, and MLBase are such machine learning platform that provide programming and runtime support.[4] OptiML is developed to allow the machine learning practitioner to write code in a MATLAB-like declarative manner of which code runs on various hardware platforms such as a multi-core CPU, a GPU, a clusters of computing nodes, and other specialized accelerators. GraphLab is a graph-based, distributed computation framework which uses the graph-parallel abstraction for sparse iterative graph algorithms and works in pull-based and asynchronous manner. SystemML is an ML framework in which a declarative ML language is used to describe ML algorithms and the codes expressed in the ML language are compiled and optimized into hybrid runtime plans of multi-threaded, in-memory operations in a single node or distributed map-reduce or Spark operations on a cluster of nodes. SimSQL is an SQL-based platform which runs on top of Hadoop, and is designed for supporting scalable Bayesian machine learning. MLBase is an ML framework which is based on Spark for lower-level data processing and works on Hadoop platform.

Giraph, Spark, and DryadLinq are some of frameworks with which ML tasks are programmed and executed even though they have been not designed only for ML tasks.[17] Giraph is a graph-based, distributing computing framework, in which models are push-based and synchronous. Spark is a cluster computing framework for large scale data analytics, which utilizes Resilient Distributed

Datasets (RDDs) that allows in-memory computation and provides fault-tolerance by managing data lineage. DryadLinq is a distributed computing framework which uses the distributed execution engine Dryad and the .Net language integrated query LINQ, and allows to easily develop ML algorithms as well as other distributed computing applications.

3 Autonomic Machine Learning Platform

3.1 Design Choices in Machine Learning Application Development

Despite many available ML tools, it takes an expert to effectively use them in a real problem because there are various factors to be selected in the applications of ML algorithms. The followings are some typical design choices made by the developers in ML application developments:

Acquisition of training data and partition : ML models are developed from training data. Hence the acquisition of proper training data is of paramount importance which collects a collection of data that reflect the problem domain in a cost-effective manner. The collected data are used for training, validation, and testing. The data collection needs to be partitioned into smaller collections according to the employed ML algorithm and the application context such as the number of available data, the test and validation methods, and the budget of computing resources.

Preprocessing of data : Sometimes, data need to be preprocessed because they are usually not clean and well curated. Data transformation and normalization are typical tasks conducted in this stage. Most ML algorithms have their own format for the input data. Raw data usually do not follow the format. The data transformation hence should be performed to convert data into the amenable format, if the data format is not comparable. Data normalization is one of fundamental processing in data preparation. The domains of attributes in data are different each other. When distance or similarity is computed, the differences in range of data attributes may distort the metrics. The typical normalization is to standardize the data.

Feature extraction : Most ML algorithms are strong enough to extract features while learning some model for the designated task. The quality of features has the strong influence on the performance of the developed ML systems. The choice of quality features requires expertise and sometimes is a time-consuming task. The feature extraction techniques include both the feature selection methods and the feature generation methods. Feature selection methods choose some features among available ones. There are some measures such as χ^2 method, cross-entropy, for evaluating the relevance of attributes to the task of interest. Feature generation methods produces new features by combining the available attributes. Recent development of deep learning techniques allow the developers not to pay much attention to feature extraction.

Task decision : There are various kinds of ML tasks such as classification, regression, recommendation, clustering, policy construction, and so on. Classification is to map data into one of pre-specified categories. Regression is to map input into output numerical domain. Recommendation is to suggest related items based on the previous history of activities. Clustering is to group similar ones into subgroups. Policy construction is to select actions proper to the situations.

ML algorithm selection : ML approaches are broadly categorized into supervised learning, unsupervised learning, and reinforcement learning. In the ML approaches, there are many choices of ML algorithms each of which has its own strength and weakness. It takes an expert to choose the proper ML algorithm for the given problem domain.

ML algorithm configuration : Most algorithms require some configuration such as model configuration and hyperparameter setting. Some ML algorithms specify the general strategy of problem solving, and thus the developers need to determine the details of the algorithms. In genetic algorithm, chromosome coding of candidate solutions and genetic operators are the main components which developers come up with for problem handling. In deep learning and neural networks, the network configuration is critical to the performance of the ML applications. Hyperparameters are the parameters for the developer to decide based on their own experience and insights.

Performance measure selection : ML algorithms try to improve the ML models with respect to some performance measure. ML algorithms narrow down their candidates of performance measure. The developers need to choose the performance measure to use for ML applications and to determine the criteria for accepting the developing models.

Computing resource acquirement : ML algorithms demand considerable amount of computing resources because they may handle large volume of data and/or they conduct heavy computations. As mentioned, the ML application development is an engineering work in which the developers cut, add, modify, and/or tune the system so that the expected performance of the model could be achieved. Hence, there is some trial-and-test nature because multiple or many candidates are tried to be trained and evaluated until a satisfactory model is identified. One of major application domains of high performance computing is the ML applications. A private section may not afford to acquire sufficient computing resources for ML application development. The cloud services for ML application development are one of the easiest enabling technology and one of the most accessible computing infrastructure.

Algorithm implementation and Public tool usage :

Once an ML algorithm is selected for use, it should be implemented as a code. The code may be implemented by the developer. Various excellent ML tools and frameworks have been developed and published in public use. Some ML tools are commercially available. The developers need to decide which tools to use or to develop by themselves. The self-development takes considerable cost and time, but provides much flexibility. The public and commercial tools have their own way to use them, and hence the developer need to learn how to use them and to build the execution environment of the employed tools.

Execution of ML algorithms: Some ML algorithm takes certain amount of time to execute. In ML application development, various ML models are usually tried to find out an excellent model. Parallel and/or execution of candidate models are widely practiced in real development. The developers need to manage the deployment and monitoring of the model training tasks over the computing resources. When a model seems to be destined to be a failure, it is better to terminate the execution of an algorithm.

Evaluation of trained models: Multiple ML models are trained with different configurations and hyperparameters. To select the most effective model, they need to be evaluated with respect to evaluation criteria and test data. The developers need to prepare the procedure and/or schedule for the ML model evaluation.

Model update for data distribution change in the problem domain: Sometimes, the data distribution in the problem domain may change over the time. As time goes, more data are collected. The ML model need be improved as more data are collected and used for learning. The developers need to take actions for model improvement as new and more data are collected.

Even an expert developer is devoted to develop an ML application with due consideration of the above mentioned design choices, it also takes considerable computing resources for multiple trials in finding a model with satisfactory performance. Hence, it will be great to have a platform to help ML applications without an ML expert and to take care of computing resources required to search for a best model and its hyperparameters. We call such a system as an autonomic ML platform because it works with least human involvement.

3.2 Requirements of Autonomic Machine Learning

An autonomic ML platform aims to provide an environment in which a non-expert on ML develops an ML-based application in his/her problem. There are some requirements that such a ML platform meets:

First, the platform needs to provide the mechanism which can use the existing public ML frameworks and tools. There are excellent ML tools some of which are even open source software supported by major companies or well-known open source software communities. Representative ML tools are SparkML, TensorFlow, Mahout, and Weka. There are also some language environments such as Spark and Python which allow easy development of massive data processing and complicated computation. It is important for the developers to make their familiar tools use when they use a new development environment. Various well-known and rapidly developing tools are efficient and effective for ML experts to use. The autonomic ML platform hence supports such tools on its environment.

Second, the platform should enable non-experts to build their ML applications even though they do not have enough understanding of ML algorithms and skills to implement their complicated logics by themselves. We pay attention to the level of autonomicity which tells how much a system takes care of ML tasks in an autonomous way. We categorize the autonomicity into six levels.

The autonomicity levels are specified in terms of the existences of specified data set, of specified

input attributes, of specified ML task, of specified ML algorithms, and of hyperparameters. At the autonomicity level 0, all pieces of the above-mentioned information are supposed to be provided by the developer. Level 0 autonomicity is the current level of ML application development. Practically level 0 autonomicity does not provide any autonomicity.

At the autonomicity level 1, the developers are free from choosing the proper input attributes in the development of ML applications. In ML applications, it is important to determine the relevant attributes to the output attribute, especially in supervised learning tasks. For each type of ML tasks, there are some techniques which can be applied for feature selection and feature extraction. The platform of autonomicity level 1 should be equipped with the functionality for relevant feature extraction and selection which are exercised with no developer's involvement.

At the autonomicity level 2, the platform has the functionality of automatically determining the hyperparameters for the given algorithm, which has the capability of level 1. The hyperparameters are one of major factors to affect the performance of a specific ML algorithm for the given data set. The hyperparameters settings are usually made by the developers based on their experience and understanding of the algorithm. It is usually time-consuming because it is somewhat trial-and-test task with no golden law for best hyperparameter values even though there have been actively studied for the hyperparameter selection in ML algorithms.

At the autonomicity level 3, only the training data set and its ML task are given, and all other ML works are done by the platform. The platform takes care of which ML algorithm to use, the hyperparameter setting, and relevant input attribute selection. It needs to evaluate candidate ML algorithms for the given t ask and to determine the hyperparameters for each ML algorithm under consideration. The brute-force approach is not a good approach in searching for a good ML model due to excessive number of possible models. There needs some remedies to reduce the search space. One strategy is to use the expert knowledge about which ML algorithm\s are effective under which situation. This requires to build up knowledge base by extracting expert knowledge and literature surveys. This knowledge-based approach is not sufficient to find a good model. While the platform is utilized for many ML application development, it meets many problem cases and may get knowledge from its experience. Hence the platform need to accumulate the profile data about its ML learning tasks and to build up the ML model construction knowledge from the profile data.

At the autonomicity level 4, the ML application is developed when the training data is only provided. All other decision choices are made by the platform. At this level, the platform searches for which task can be conducted for the given data set. It performs unsupervised learning on the data set. The unsupervised learning is rather broader than the unsupervised learning approach conventionally mentioned in ML literature because the platform is concerned with not only supervised tasks but also supervised tasks.

The autonomicity level 5 is the ultimate level at which nothing is made by the developers. The platform tries to find some patterns from all the available things without any involvement of the developers.

Third, the platform needs to have the capability of using the external computing resources and executing the learning and inference tasks with them. The platform should have the mechanism for registering, locating, monitoring the ML computing resources and ML tools, and assigning ML tasks to them.

Fourth, the platform needs to provide a script language with which the autonomic ML tasks are expressed in a distributed and parallel manner. The script languages can be either coarse grained or fine grained. Coarse grained script languages allow the programmers to express the tasks and processes in a highly abstract way. Fine grained script languages allow them to specify the details of the tasks and processes. Some existing ML tools is equipped with a fine-grained script language in which the developers design their ML models and implement the logics of ML algorithms. Such script languages sometimes express the ML models and the operations of ML algorithms in directed graphs such data flow graph in TensorFlow and RDD lineage graph in Apache Spark. The fine-grained script languages are tightly bound to their execution platform on which they are compiled, deployed and executed. Coarse grained script languages deal with the functional modules which carry out some designated tasks, and express the process of operations with the functional modules. Once an ML applications are written in a coarse-grained script language, it is more convenient to execute them on a distributed and parallel environment

because scheduling and management can be done on the functional module basis.

3.3 Architecture of the Proposed Autonomic Machine Learning Platform

To realize an autonomic ML platform, we designed an architecture that meets the above-mentioned requirements as shown in Figure 1. The architecture supports both the black box and white box modes. In the black box mode, the ML platform does not ask the developers to write some codes of ML tasks, but just asks them to specify the components corresponding to the autonomicity level employed at the moment. The remaining details on ML tasks are taken care of by the platform which uses the existing ML platforms and tools integrated. In the black box approach, the platform contains various functional modules for both ML algorithms and design choices mentioned in Section 3.1. The management works for task deployment, execution, and monitoring are coordinated by the policy taken by the platform. The developers of ML applications are not strongly engaged in ML model development except the design choices required for the developer to provide. Hence, the developers cannot make fine control on the ML application development in terms of ML model construction and computing resource utilization.

In the white box mode, the developers write some codes for ML tasks according to their own logic and design choices, using the provided script language. The ML models and the algorithms for training and inference are encoded in a directed computation graph which is later compiled and executed. The deployment of operations in the directed computation graph can be more complicated than coarse-grained scripts because the granularity of operations may not be large enough or the data communication between operations happens heavily. When operations are tightly bound, it is better to wrap them into a module which runs a single machine.

In the architecture, a developer interacts with the platform through the ML client. The ML client delivers the developer' request to the Autonomic ML Coordinator, monitors the progress of the deployed tasks, and receives the results of learning and inference. It maintains the connection between the database system and the platform so that the data sets in the database can be used in ML tasks and the results of learning and inference are stored and maintained in the database. The Autonomic ML coordinator takes charge of coordinating the autonomic ML tasks by generating ML task plans for plausible configurations, executing them with the available computing platforms, and selects the best models and patterns from the results of the ML tasks. Due to enormous number of candidate ML configurations, the demands on computing resources are soaring up as the autonomicity level grows high. Expert knowledge can be helpful to set up the initial configurations and to narrow down the ranges of candidate hyperparameters under consideration.

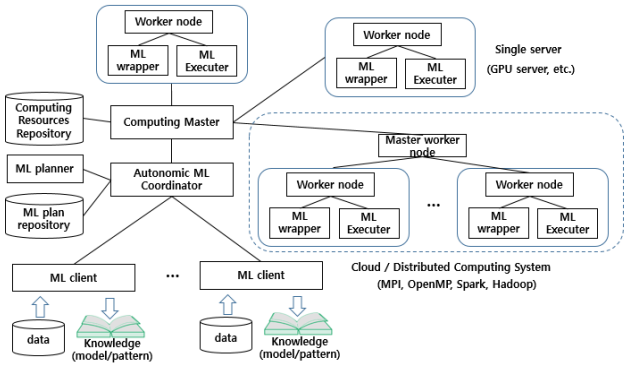

Fig. 1. The architecture of the proposed autonomic ML platform

Such knowledge is maintained in the ML plan repository to which the execution profiles including configurations and their performance are maintained to incrementally enhance the knowledge. The ML plan repository also includes the cost model for configurations and data characteristics. The cost model is updated to reflect the execution profiles collected during the ML tasks conducted on the platform. The ML planner is in charge of determining the model configuration, the hyperparameters, and/or feature selection and extraction, as well as ML tasks deployment.

The Computing Master receives the ML tasks from the Autonomic ML coordinator, deploys them over the computing resources, monitors their progresses, reconfigures the deployments of ML tasks, if needed, and reports the status and the results to the Autonomic ML coordinator. The computing resources include the single servers like a GPU server and a multi-core server, the cloud services and distributed computing systems. Each computing resource has a master worker node which communicates with the Computing Master and manages its own computing resources to conduct the ML tasks. The proposed platform supports to use the existing ML platform and tools. Hence, the ML wrappers are implemented to invoke the ML

tools to execute the corresponding ML tasks, to report their progresses and results, and to terminate the deployed tasks, if needed.

When the ML tasks are expressed in the script language, the codes are executed by the ML Executer which launches the corresponding ML platform and conducts the execution of the codes on the platform according to the corresponding configuration. When a cloud service or a distributed computing system is used as a computing resource, the master worker node is installed on it.

The master worker node communicates with the deployed ML platform or framework which takes care of the ML tasks. If the ML codes in the script language are delivered, the master worker node distributes the ML codes to the Worker nodes which next invokes the ML Executers. The master worker nodes of the computing resources that support the proposed ML platform to register themselves into the Computing Resource Registry with their access method, functionality, capability, workload, and billing policy. The Registry checks the registered computing resources on a regular basis, and maintains the current states. The Computing Master refers to the Computing Resource Registry when distributing the ML tasks and monitoring their progress.

4 Implementation of the Proposed ML Platform

The proposed autonomic ML platform is a huge system to integrate the existing ML platforms and frameworks and to use many external computing resources. In addition, it is an ambitious approach to try to realize the full range of the autonomicity levels from the primitive level to the completely autonomic level. Under both the black box approach and the coarse grained script support, the architecture and the functional modules are designed and specified.

To evaluate the feasibility of the proposed architecture, we have developed an initial stage prototype which realizes a preliminary platform of the proposed architecture. The Single server node has been implemented to provide the GPU-based ML framework to execute the Tensorflow programs.

The prototypes of the ML client, the Autonomic ML coordinator, the ML planner, the ML plan repository, the Computing Resource Registry, and the Computing Master have been implemented. The autonomicity levels 0 and 1 have been supported by the ML planner. The prototype system has been tested to execute the deep learning algorithms like CNN and RNN algorithms.

5 Conclusions

ML application development takes an expert developer to be successful and also takes computation resources to find an excellent model from training data. The autonomic ML platform can be an enabler for non-experts of ML to employ ML techniques in solving their real domain problems. We defined the autonomicity levels from the primitive level to the ultimate level. To handle the vast demands on computing resources in autonomic ML, we proposed an architecture to realize the ML platform which uses the external computing resources and the existing ML platforms and frameworks. To evaluate the feasibility of the proposed platform, we have developed a primitive prototype of the proposed architecture for the autonomic ML platform.

The proposed ML architecture can be an important enabling technique to convert a database into an intelligent database. With the advances and widespread deployment of sensor technology like IoT, large volumes of data have been accumulated with the expectation of finding business opportunities from them. It is, however, not easy to transform such data into valuable knowledge. The autonomic ML platform can help extract valuable and confident knowledge from data. There yet remains much work in realizing the functionalities of the autonomic computing platform. Once it is implemented, the ML technology can be easily adopted in many domains without ML experts. Especially, the databases can be evolved into intelligent databases with the help of the autonomic ML platform. Such intelligent databases allow the dream that once data are stored, the knowledge are developed automatically.

Acknowledgments: This research was supported by Next-Generation Information Computing Development Program through the National Research Foundation (NRF) of Korea (Grant no.: NRF-2017M3C4A7069432), by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program(IITP-2017-2013-0-00881) supervised by the IITP(Institute for Information & communications Technology Promotion).

References:

[1] Kraska, T., Talwalkar, A., Duchi, J., Griffith, R., Franklin, M. J., Jordan, M.: MLbase: A Distributed Machine-learning System. CIDR, Vol. 1, 2013.

[2] Bello-Orgaz, G., Jung, J. J., Camacho, D.: Social big data: Recent achievements and new challenges. Information Fusion, Vol. 28, pp. 45-59, 2016.

[3] Jha, S., Qiu, J., Luckow, A., Mantha, P., & Fox, G. C.: A tale of two data-intensive paradigms: Applications, abstractions, and architectures. Proceedings of 2014 IEEE International Congress on Big Data (BigData Congress), (pp. 645-652), 2014.

[4] Cai, Z., Gao, Z. J., Luo, S., Perez, L. L., Vagena, Z., Jermaine, C.: A comparison of platforms for implementing and running very large scale machine learning algorithms. In Proceedings of the 2014 ACM SIGMOD international conference on Management of data, pp. 1371-1382, 2014.

[5] Lee, K.M., Lee, S.Y., Lee, K.M., Lee, S.H.: Document Density and Frequency-Aware Cluster Identification for Spatio-Temporal Sequence Data, Vol. 93, No. 1, 2017.

[6] Brochu, E., Cora, V. M., & De Freitas, N.: A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. arXiv preprint arXiv:1012.2599, 2010.

[7] Bergstra, J. S., Bardenet, R., Bengio, Y., Kégl, B.: Algorithms for hyper-parameter optimization. In Advances in Neural Information Processing Systems, pp. 2546-2554, 2011.

[8] Johnson, V. E., Wong, W. H., Hu, X., Chen, C. T.: Image restoration using Gibbs priors: Boundary modeling, treatment of blurring, and selection of hyperparameter. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 13, No. 5, pp.413-425, 1991.

[9] Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. Journal of Machine Learning Research, 13(Feb), pp.281-305, 2012.

[10] Bergstra, J., Yamins, D., Cox, D.: Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. Proceedings of International Conference on Machine Learning, pp. 115-123, 2013, February.

[11] Thornton, C., Hutter, F., Hoos, H. H., Leyton-Brown, K.: 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, 2013, August.

[12] Larochelle, H., Erhan, D., Courville, A., Bergstra, J., & Bengio, Y.: An empirical evaluation of deep architectures on problems with many factors of variation. Proceedings of the 24th international conference on Machine learning, pp. 473-480, 2007.

[13] Kang, S. J., Lee, S. Y., & Lee, K. M.: Performance comparison of OpenMP, MPI, and mapreduce in practical problems. Advances in Multimedia, Vol. 7, 2015.

[14] Lee, K., Lam, M., Pedarsani, R., Papailiopoulos, D., Ramchandran, K.: Speeding up distributed machine learning using codes. Proceedings of 2016 IEEE International Symposium on Information Theory (ISIT), pp. 1143-1147, 2016, July.

[15] Lee, K.M., Jeong, Y.-S., Lee, S.H., Lee, K.M.: Bucket-size balancing locality sensitive hashing using the map reduce paradigm, Cluster Computing, 2017.

[16] Li, M., Andersen, D. G., Park, J. W., Smola, A. J., Ahmed, A., Josifovski, V., Su, B. Y.: Scaling Distributed Machine Learning with the Parameter Server. OSDI, Vol. 1, No. 10.4, p. 3, 2014.

[17] Sparks, E. R., Talwalkar, A., Smith, V., Kottalam, J., Pan, X., Gonzalez, J., Kraska, T.: MLI: An API for distributed machine learning. Proceedings of 2013 IEEE 13th International Conference on Data Mining (ICDM), pp. 1187-1192, 2013, December.

[18] Singh, D., Reddy, C. K.: A survey on platforms for big data analytics. Journal of Big Data, Vol. 2, No. 24, 2015.