Parallel dynamic data-driven model for concept drift detection and prediction

Lin, S-Y. , Chiu, Y-C. , Lewandowski, J. and Chao, K-M.

Parallel Dynamic Data-Driven Model for Concept Drift Detection and Prediction

Szu-Yin Lin[a], Yao-Ching Chiu[b], Jacek Lewandowski[c,d], Kuo-Ming Chao[c]
[a] *Department of Information Management, Chung Yuan Christian University, Taoyuan, Taiwan*
[b] *Institute of Information Management, National Chiao Tung University, Hsin-Chu, Taiwan*
[c] *School of Computing, Electronics and Mathematics, Coventry University, UK*
[d] *Department of Genetics, Wroclaw University of Environmental and Life Sciences, Poland*

Abstract. The traditional data analysis and prediction method assumes that data distribution is normal and will not change. Therefore, it can predict unlabeled data by analyzing the static and historical data. However, in today's big-data environment, which is changing frequently, the traditional approaches can no longer be effective, as they cannot handle concept drift problems in a Dynamic Data Driven Application System (DDDAS). This study proposes a parallel detection and prediction method for concept drift problems in DDDAS. The proposed method can detect dynamic and changing data, and then feedback to the prediction model to revise for better subsequent predictions. Furthermore, this method computes a global prediction result by aggregating local predictions in the resource bounded environment. Therefore, the prediction accuracy increases, and the computation time decreases. In the simulation, the Map-Reduce technology is used for parallel processing. The simulation results show that the prediction accuracy is raised by 14%, and the execution time is improved by almost 45%.

Keywords: Dynamic Data-Driven Application System, Concept Drift, Map-Reduce

1. Introduction

In recent years, the Ubiquitous Sensor Networks [1], Internet of Things [2], Cloud Computing [3] and Big Data [4] have been rapidly developed and advanced to meet the changing application's environments and demands. These information technologies are so popular that Gartner Inc. listed them in the top 10 Information Technology Strategies of the Year 2016 [5]. Fast development of wireless and cellular network infrastructures, which took place in recent years, enabled mobile devices with computing power to communicate with each other with a cheaper rate and higher availability. This allowed us to collect a wide variety of data through sensors, smart phones or wearable devices. Furthermore, with the higher efficiency in data storage and the lower cost of storage devices, we can handle larger amounts of data in a shorter time. The amount of such data accumulates, however, quickly. In recent years, big data analysis has become one of the hottest research fields. Strong interests in the collected data grow from the big data's ability to derive new useful information, which can be further used to generate new knowledge and applications. These big data (data driven) applications could be applied in many fields of social networks, cybercrime, energy, or e-health [6]. However, due to the volume, velocity and variety of big data, to increase the precision of predicting the future trend or events in such dynamic, uncertain, and complicated environments is still a challenge. Most the existing prediction methods are designed to focus on analysis of static historical data. However, in the dynamic environment, new data can be generated at any time which violates the assumption of most traditional prediction models which is built upon that idea that the population is distributed in a stationary manner. Therefore, such prediction models may lose accuracy if it relies on the historical data only without up-to-date data, which may possess new features with new distribution. This phenomenon is called Concept Drift [7]. Concept drift means that the focus on the topic moves from one concept to another. The definition of concept drift in the traditional context differs from the one in the context of data mining. Once concept drift happens, the prediction model, which does not include the new factors, may have the high probability of making the wrong predictions. Emerging topic detection [8] and the typhoon rain falls predictions are the examples of Concept drift.

This research paper proposes a method to detect concept drift and adjust the prediction model to retain the precision in a dynamic data stream environment. The proposed method leverages the concept of Dynamic Data Driven Application System(DDDAS) [9] to detect changing concepts, and then adopts the available adjustment strategies to increase the prediction accuracy. Due to the limited resources (e.g. computation, storage), it is not possible to try various combinations of datasets and parameter settings of a model. In order to allow varied prediction models with varied parameter settings to apply to different datasets, different models have been deployed to a distributed computing architecture and computed in parallel to try different combinations in order to optimize the model prediction output before the designated deadline. The proposed method will include the detection and adjustment of these two components.

2. Related Works

2.1. Dynamic Data Driven Application System (DDDAS)

A Dynamic Data Driven Application Systems (DDDAS) is a real time feedback and control system. It is a new paradigm supported by the National Science Foundation (NSF) in order to solve the inefficient problems in traditional simulations, predictions and measurements [10]. DDDAS provides a model which is characterized by a more reliable outcome, stable data process and accurate prediction or analysis results. It allows a model to dynamically receive and respond [11] to the changing environments. The DDDAS concept describes the dynamic capability of the system to process and control data. DDDAS implements real-time data coordination in a runtime environment. Hence, DDDAS does not only provide more just in-time statistics, but also offers the feedback method to enhance the model and improve experimental results [11-14]. Recent improvements of computing power and models (e.g. Cloud Computing, Grid Computing) as well as data technologies(e.g. Sensor Network, data storage techniques, data communication techniques), speeds up the promotion of DDDAS architecture [14]. These components are integrated with automatic feedback, measurement, simulation and a control method to work in a dynamic way. Users could use the DDDAS model via the dynamic visualization module to interact with other components: (1) real-time dynamic data gathering from others modules; (2) time series data gathered from measurement instruments (e.g. sensors, database, GIS, open data); (3) dynamic computation modules dealing with mathematical models and prediction computation.

Due to these functions, DDDAS is able to provide efficiency and stability to handle real-time situations in the real world. As Figure 1 shows, the DDDAS architecture includes: the user's controller, dynamic visualization interface, dynamic computation and real-time dynamic data gathering modules [14].

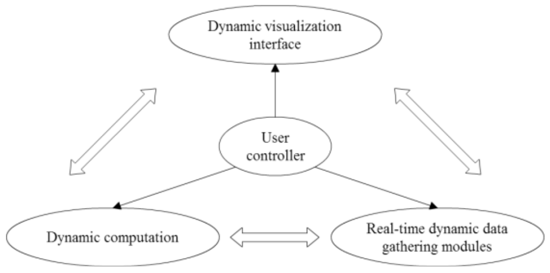

Dynamic visualization interface
User controller
Dynamic computation
Real-time dynamic data gathering modules

Figure 1. Basic concept and components of DDDAS.

In the past, when facing challenges in weather prediction, agricultural forecast or contaminant tracking, the historical data was used as the input into a prediction system. However, when a model relies on historical data only, it cannot reflect the real-time events or provide real-time feedback. DDDAS proposed a real-time feedback method to transfer data to a computing model, thereby enhancing the predication accuracy. In the field of environmental and agriculture sciences, (e.g., Greenhouse Gas emission, River Pollution monitoring) DDDAS offers the ability to adjust and change parameters. This feature makes the model scalable [15]. Moreover, it can be used in predicting hurricanes [16]. DDDAS is not only able to process numerical data, but it can also analyze graphs or diagrams to increase the prediction success rate [17].

Frederica Darema who proposed the DDDAS concept, pointed out that it encompasses more than real-time control. However, there are some challenges which require further work. One of the challenges is the uncertainty of concept drift dynamics, which causes the prediction error. The key to solving this problem is to find out how to establish data correlation. In this research paper, we will try to solve the correlation problem and increase the predication model accuracy in an efficient way.

2.2. Concept Drift

"Concept" is a central structure that is used to describe the common properties of a set of objects. The states of the world continue to change, and the corresponding concepts also change rapidly. A concept is mapped to different objects in different places or times [18]. "Concept Drift" is a phenomenon that happens when the distribution of data shift from its original patterns to different ones. This "Drift" will possibly cause the prediction model, which was trained on historical data, to make the erroneous predictions. The tastes of users are often determined by Hidden Contexts [19]. It is a challenge to give a clear and definite formula to model the tastes of users. An example such as: a customer's shopping behavior changes over time due to a number of hidden reasons, which can make the modeling process difficult and complicated. Furthermore, data generated in a dynamic environment may contain lots of noise data to prevent learning methods to distinguish real concept drifts from false ones. For example, some learning algorithms may be too sensitive with data accompanied by noise, while some of them may be characterized by a slow response to the changes, this results in the algorithm being too slow to adapt to the latest concepts [20].

The two types of concept drift are: (1) gradual concept drift and (2) abrupt concept drift. An example of gradual concept drift is the data set that uses spam assassin data collection (Katakis et al., 2010), and an example of abrupt concept drift is the SEA dataset which represents the streaming ensemble algorithm [21]. An ideal concept drift detection system should be able to (1) adapt quickly to a drifted concept, (2) effectively identify noise data, and (3) recognize recurrent context and model them as internal rules. The machine learning and analysis prediction approach of concept drift is to detect the statistical characteristic of variables of target objects. The pre-built model is used to analyze the patterns of the target variable and predict its possible changes over time. However, such a model is too reliant on historical data to maintain a high accuracy, which may be suitable for the dynamic and real-time applications. In order to build an accurate and effective learning machine, the designer should consider the following factors in the modeling process:

(1) Future assumption: the model being able to accommodate future unstable data distribution.

(2) Classification changes: the model being able to find the possible variation rules.

(3) Adaptive learning: a learning machine being able to adapt to emerging contextual concepts via (1) and (2).

(4) Model selection: being able to identify rules from experiences to choose the appropriate parameters and model accordingly over time [22].

When over-fitting phenomenon happens, adaptive approaches should be adopted to adjust the pre-built model to avoid incorrect prediction results for different distributions of data [18, 23, 24].

2.3. Dynamic Weighted Majority

The Dynamic Weighted Majority (DWM) based on a Weighted Majority Algorithm [25], proposed by JZ Kotler and MA Malo in 2003, is an algorithm which takes dynamic weights adjustment as the core concept. DWM is an ensemble method that uses an 'expert' group to determine the weights, so each member in the group assigns a weight to each possible approach for the algorithm to try. DWM greatly benefits from the concept drift detection because of the dynamic weight adjustment. The algorithm changes the weights to the approaches. If some of the approaches' weights decrease and the results start to change, a concept drift taking place can be concluded. Later in this paper we will discuss the core concept of DWM and advantages of DWM over Weight Majority Algorithm along with the integration of DWM in a dynamic environment.

The Weighted Majority Algorithm (WMA) is one of the components in the pool of machine learning methods. WMA presents a pool of prediction algorithms (e.g. a group of classifiers, a group of the same or different approaches) without any prior knowledge. Although it assumes that we have no prior knowledge about the accuracy of the algorithms in the pool, there are adequate reasons to believe that one or more algorithms will perform well [25]. Unlike common prediction methods, the WMA makes decisions by group voting. It makes fewer mistakes than a single prediction approach because it employs a collection of prediction approaches. The steps of the algorithm making predictions via WMA are described below:

- ➢ STEP 1: Each prediction method in the WMA pool makes the prediction individually.
- ➢ STEP 2: WMA concludes the result of a class from its highest total.

- STEP 3: Compare the prediction results with the actual outcome.
- STEP 4: Increase the *weight (ω)* of experts who make the prediction correctly, and decrease the *weight (ω)* of experts who make the wrong prediction.

Assume that the problem is a binary decision problem. The process of the algorithm is presented as follows in each trial: To construct the compound algorithm, a positive weighted value is given to each of the algorithms in the pool. Each of prediction algorithms is executed to make the prediction. The compound algorithm then collects weighted votes from all the algorithms in the pool, and gives the final prediction that has the highest vote. If the compound algorithm makes a mistake, the algorithms in the pool that contributed to the wrong prediction will be discounted by a constant ratio β where $0<\beta<1$. If Weight Majority Algorithm is applied to a pool of functions with $\beta = 0$ then the initial weights are equal. If $\beta > 0$, then Weight Majority Algorithm gradually decreases the influence of functions that make a large number of mistakes and gives the inconsistency can eliminate a function. Assume that the Weight Majority Algorithm is applied to a pool, *F*, of functions and that the sequence of trials has *m* mistakes with respect to *F*. For the general case where Weight Majority Algorithm is applied to a pool *F* of algorithms we show the following upper bounds on the number of mistakes made in a given sequence of trials:

- $O(\log |F| + m)$, if one algorithm of F makes at most *m* mistakes.
- $O(\log |F| / k + m)$, if each of a subpool of k algorithms of F makes at most m mistakes.
- $O(\log |F| / k + m / k)$, if the total number of mistakes of a subpool of k algorithms of F is at most *m*.

As discussed above, the WMA presents weighted voting based on the ensemble method. It combines a group of prediction approaches and takes each approach as an 'expert' with its own weight. The superiority of this algorithm is the use of group decision, as it can provide a more stable and accurate output. The Dynamic Weighted Majority (DWM) is based on the Weighted Majority Algorithm (WMA). It extends the advantage of WMA by adding a threshold (Θ) in the algorithm to allow weight change in the runtime, and its weight is reduced by the multiplicative constant, β, (β is from 0 to 1) when the errors occur in the prediction process. Therefore, the best algorithm with the highest weight can be found in the pool of WMA dynamically.

In this study, we take Naïve Bayes classifiers as the example to explain the process of the DWM. Naive Bayes is a simple technique for constructing classifiers. Models assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. In the initial stage, each expert makes the prediction for collecting the results and summarizing the weight of experts. Secondly, the DWM takes the highest weight of the results as a global result. Thirdly, it compares the global result with the actual answer Furthermore, if the global result makes the wrong prediction, it decreases the weight of the expert who proposed the wrong predication, multiplying it by the constant β. Lastly, it checks if any expert's weight is below the threshold Θ; it then deletes the expert and adds the new one with normalized weight. Based on the knowledge of concepts and processes we described above, it makes sense that the model of DWM fits the requirements of DDDAS which includes (1) Dynamic data environment, (2) Prediction methods (experts), and (3) feedback (weighting update) and β is the adjustment value for the learning experience. Thus, this research will take the concept of DWM to support our designed method to solve the issues of concept drift.

3. Parallel Detection and Prediction Method for Concept Drift in DDDAS

In this section, a parallel detection and prediction method for concept drift in dynamic data driven application systems is presented in detail. Characteristics of big data include dynamics, uncertainty, variety, complication, and correlation. Thus, using the traditional prediction methods or machine learning approaches, which are used to predict and analyze the static or stationary data, may not be sufficient enough to satisfy new requirements of big data, as they also cannot adjust themselves dynamically to meet new phenomenon when the concept drift occurs. The proposed method can detect the concept drift phenomenon in a dynamic environment, and efficiently adjust the parameters in the prediction model according to the emerging context. Furthermore, different configurations of models for the applications can be simulated and tested efficiently via the use of distributed parallel computational nodes. There are three key problems in this study:

(1) Dynamic concept drift: The first one is to detect the occurrence of concept drift. Concept drift means that a prediction model makes a prediction, but the correlation or characteristic of the data has been changed over time in unforeseen ways that causes the predictions to become less accurate over time. The phenomenon is defined as dynamic concept drift. So, how to detect and perceive the concept changes in dynamic environment, is the key problem in this study.
(2) Model adjustment: Model selection (adjustment) and validation is to choose the most appropriate model for the data. In a dynamic environment, the values of model parameters should be adjusted repeatedly according to dynamic data in order to ensure that the model behavior is appropriate. Moreover, determining which segment of the data and the parameters should be selected for adjustment is also a problem.
(3) Resource balance: The last problem is to balance the computing resources. Validating models (experts) in DWM is a time-consuming task. If all the required resources of the concept detection and adjustment model are computed in one machine, it will take too much time to produce the prediction results. In a real-time application, the slightest difference in time to produce the outcome for the predication model could lead to a huge difference in its usefulness.

3.1. Definition and Notations

3.1.1. Definition

Concept drift is a phenomenon in which patterns of data continuously change over time and lead to the pre-built model not being able to make accurate predictions. This study uses three levels of concept drift and two metrics to compute the prediction results. One of our goals is to detect concept drift. We use the error-rate (p_i) and the standard deviation (s_i) of the prediction model to obtain the confidence level, which determines the possibility of concept drift occurrence [26, 27]. The binomial distribution gives the general form of the probability or to observe an error, so the error-rate (p_i) is a random variable from a sequence of Bernoulli trials. For each record i in the data stream and the number of miss-classifications until, noted as $error_i$, the error-rate is $p_i = (error_i / i)$ and the standard deviation is $s_i = \frac{\sqrt{p_i(1-p_i)}}{i}$. Moreover, p_{min} and s_{min} represent the current minimum p_i and s_i respectively in all of the prediction models in distributed nodes. Three levels of confidences are listed below:

– Normal Level (NL):
Occurs if $p_i + s_i < p_{min} + 2s_{min}$
– Warning Level (WL):
Occurs if $p_i + s_i \geq p_{min} + 2s_{min}$
– Drift Level (DL):
Occurs if $p_i + s_i \geq p_{min} + 3s_{min}$

Besides this, this study also uses the following two metrics to build the prediction mechanism in the proposed parallel dynamic data-driven model:

– Local Prediction (LPred):
In distributed nodes, each node (worker) will predict its own local result (*LPred*) with the built-in model held by it. All local prediction results will be sent to the driver, as described in the next section, to do further prediction and detection work.

– Global Prediction (GPred):
The global prediction result (*GPred*) is calculated from a group of prediction models produced by the drivers in order to get a reliable result.

3.1.2. Notations

Table 1. Notation table.

Notation	Definition
α	The factor of raising the weight when the algorithm's predictor makes the wrong prediction
β	The factor of decreasing the weight when the algorithm's predictor makes the wrong prediction
$LPred_{t,i}$	Local prediction result of worker i in time t
p_i	Error-rate of instances until i
s_i	Standard deviation of instances until i
p_{min}	Current minimum value of p_i
s_{min}	Current minimum value of s_i

3.2. The proposed method

As shown in Figure 2, the architecture of the proposed method has two major roles: the driver and the worker. These two major roles control the flow of our programming model.
Driver: The driver is a central controller that receives streaming data and passes the data to workers to dispatch tasks. After the worker finishes the task, the driver will receive the prediction result from the worker, and it can then select the best prediction result for the user.

Worker: The worker is a computational device that will maintain the prediction model and wait for the driver to call and provide input data. Once the worker is triggered, it will make the prediction based on the input data, and then passes the local prediction result (*LRep*) back to the driver. While this happens, it simultaneously invokes another function to evaluate and determine if a concept drift occurs. These two actions asynchronously take place in order to improve the performance. We will now introduce each node in the architecture in more details.

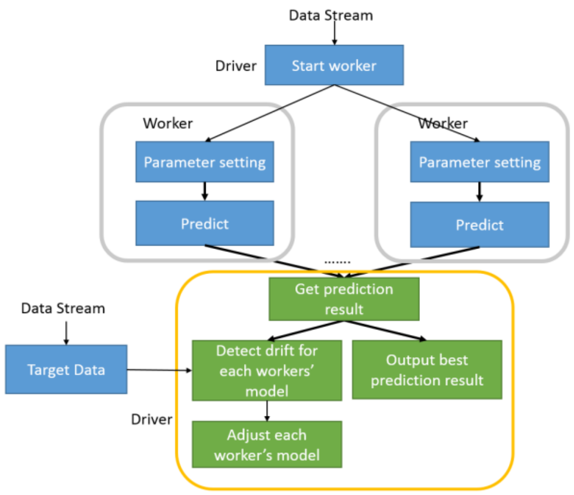


Figure 2. Overall architecture of the proposed method.

– Start workers

Once the driver receives the data stream, it will transmit the data to workers and ask them to start working in parallel.

– Parameters setting

After being awoken by the driver, the worker will set the parameters and concept drift detection strategy. Each worker may have different parameters (such as the time range of training datasets) and adjustment strategies (see the detailed explanation in Section 3.2.1). In other words, each worker has different model parameters and adjustment strategies. Through the dynamic learning and adjustment process of these large varieties by DWM, we expect to get better results in a dynamic process.

– Predict

At this stage, the worker will first partially train the prediction model with pre-defined parameter settings and data range. The worker will then apply the received data to the prediction model to obtain the local prediction result (*LPred*). After finishing the prediction, the worker will forward the result in two directions. The first one is to give feedbacks to the driver. The other is to invoke the drift detection function to analyze if the concept drift, based on the recently received data, did take place.

– Get prediction results

While receiving the *LPred* made by the worker, the driver will invoke the drift detection function and best result selection function respectively. The key point is that, these two functions are running in the parallel processes or threads to prevent the driver from being blocked, so it can continue receiving new input data streams or other prediction results.

– Output the best results

Because the driver has the weight states of all the workers, it can determine which *LPred* is suitable as the *GPred* (generally, selecting the worker with the highest weight). Finally, the driver sends the output of the *GPred* to the user.

– Detect drift for each worker's model

At this stage, the driver will store the *LPred* of the data stream, and then each worker waits for the actual labels of that data stream (target data) at time *t+1* to evaluate the prediction results of worker's model at time *t*. After the evaluation, the driver will update the weight edvalue and calculate the confidence of concept drift for each worker's model.

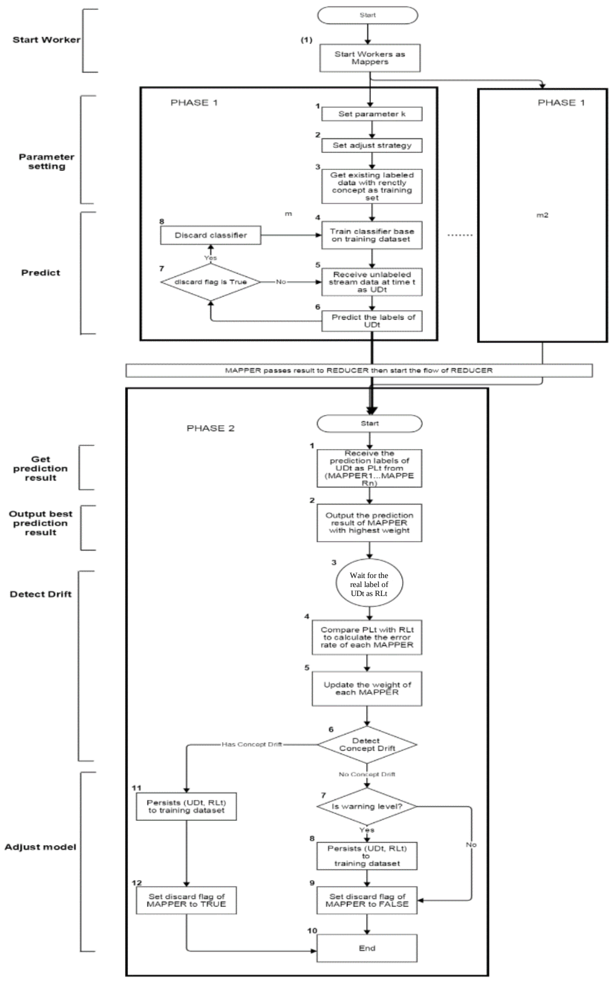
– Adjust each workers' model

At this stage, if the concept drift phenomenon has been detected, the driver will adjust the worker's prediction model according to different adjusting strategies.

Figure 3 is the flow of the proposed method. It is composed of two major phases, corresponding to the prediction and the detection respectively. We will introduce these in detail in the following sections.

3.2.1. Phase 1 – Prediction

In this phase, the main objective is to predict the unlabeled data coming from the data streams. In order to analyze huge amounts of data with multiple models trained with different parameter settings and referenced training data, our method will distribute the computational tasks to different computing nodes as workers. Each worker will work on the data it receives and use the built-in model to make the prediction. Also, each worker will check the discard flag to determine whether it should discard the trained prediction model or not.



Start Worker
Parameter setting
Predict
Get prediction result
Output best prediction result
Detect Drift
Adjust model
Start
(1)
Start Workers as Mappers
PHASE 1
1 Set parameter k
2 Set adjust strategy
3 Get existing labeled data with renctly concept as training set
m
4 Train classifier base on training dataset
8 Discard classifier
Yes
7 discard flag is True
No
5 Receive unlabeled stream data at time t as UDt
6 Predict the labels of UDt
....
PHASE 1
m2
MAPPER passes result to REDUCER then start the flow of REDUCER
PHASE 2
Start
1 Receive the prediction labels of UDt as PLt from (MAPPER1...MAPPERn)
2 Output the prediction result of MAPPER with highest weight
3 Wait for the real label of UDt as RLt
4 Compare PLt with RLt to calculate the error rate of each MAPPER
5 Update the weight of each MAPPER
6 Detect Concept Drift
Has Concept Drift
No Concept Drift
11 Persists (UDt, RLt) to training dataset
7 Is warning level?
Yes
No
8 Persists (UDt, RLt) to training dataset
12 Set discard flag of MAPPER to TRUE
9 Set discard flag of MAPPER to FALSE
10 End

Figure 3. Flow of the Map-Reduce distributed nodes.

When the driver receives the input data stream, it will ask all of the workers in the cluster that is controlled by the driver to start working. We use yarn [28] to cluster computers together in order to start the jobs in the Map-Reduce paradigm. The overview of this distributed computing architecture and the relationship between driver and works is shown in Figure 4.

– Steps 1-3:

After being triggered by the driver, the worker will determine the parameters for the model and select the drift detection strategy. The worker will then take the existing labeled data as the training set in order to build the prediction model.

– Steps 4-6:

The worker will first train the prediction model with the input value and parameters obtained from the previous stage. Then the worker will get the received unlabeled data at time *t* (UD_t) to predict (classify) their label. After the worker finishes the above task, it does two actions simultaneously; transmits the prediction results to the driver that will immediately do the reduce task and checks whether or not the discard flag of its model has been set to True in Step 7. If the flag is set to True, the worker will discard its model in Step 8 and train a new one. If the flag is False, the worker will continue receiving the unlabeled data stream for prediction in Step 5.

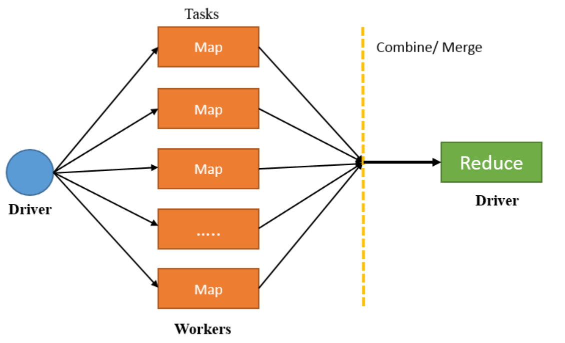


Figure 4. Architecture of Driver and Worker.

3.2.2. Phase 2 - Detection

In this phase, the aim is to detect the concept drift phenomenon by comparing the prediction results with the actual ones. By collecting each prediction result from each worker, the driver will evaluate the occurrence of concept drift. If the concept drift happens, it then stores the data with new concepts for the worker to build the new prediction model.

– Step 1:

The driver receives the prediction labels of the data stream at time *t* from worker i as$LPred_{\mathrm{t,i}}$. The driver will then check whether the weight of worker i is the highest of all. If it is True, then the driver will mark the $LPred_{\mathrm{t,i}}$ as *GPred* and then output to the user in Step 2.

– Steps 3-6:

The driver will wait for the actual labels of UD_t as RL_t and then compare the $LPred_{\mathrm{t}}$ of each workers' model with $\mathrm{RL_t}$ to calculate error-rate (p_i) and standard deviation (S_i) as well as analyze the possible concept drift phenomenon. Also, the driver will update the weight of each worker using the following equations:

$$Weight_i = Weight_i + Weight_i * \alpha \qquad (3.1)$$

$$Weight_i = Weight_i - Weight_i * \beta \qquad (3.2)$$

α is a coefficient that represents the speed of increasing weight when the model makes a correct prediction. β is a coefficient that represents the speed of decreasing weight when the model makes an incorrect prediction. In this method, we use binomial distribution to calculate p_i which is equivalent to the probability of k miss-classifications of n instances:

$$p_i = (error_i/i) \tag{3.3}$$

We then use p_i to calculate standard deviation:

$$s_i = \frac{\sqrt{p_i(1 - p_i)}}{i} \tag{3.4}$$

The standard deviation represents the degree of dispersion of these instances, and then applies the confidence interval presented in Figure 5, to determine the probability or confidence of concept drift happening. It assumes that p_i will decrease while i increases if the data has a stationary distribution. A significant increase in p_i indicates that the distribution of the data is changing. The minimum values (p_{min}, s_{min}) of p_i and s_i are recorded when $p_i + s_i$ reaches its minimum value. Here are the different levels (normal, warning, and drift level) with different confidences in the confidence interval for concept drift detection.

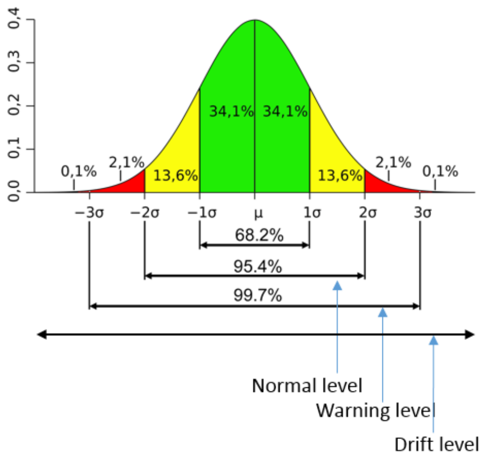


Figure 5. Confidence Interval with three levels.

- At level NL, concept drift is considered not happening.
- At level WL, it is possible for concept drift to happen. The method will store the labeled data in the corresponding time frame.
- At level DL, we are 99% confidence that concept drift happened. The method will set the discard flag of the workers' model to True and pass the labeled data stored from the warning level as a training set for workers to train the model for the new concept.

– Step 7-12:

In order to adjust prediction model when concept drift happens, we first check the confidence level described previously:

- If the level is "Drift": The driver will store the current evaluated data with its real label (UD_t, RL_t) as a training set and set the worker's discard flag to True. So, in the next round, when the worker, based on new data stream, starts its prediction, it will discard the old prediction model and use the stored training set to build a new prediction model.
- If the level is "Warning": The driver will store the current evaluated data with its real label (UD_t, RL_t) from the time slot *t*.
- If the level is "Normal": The evaluated result is good and no adjustment is needed in this level.
- The additional consideration is that, because the data stream may contain noises, it is necessary to prevent the worker from mistaking noise as a concept drift. If the method wrongly determines too many concept changes as noise data, the model will not be adjusted enough to model the new concept. Hence, the following four adjustment strategies can apply:

(1) Assume that the last warning detection signal is an exception. If the present status of concept detection falls in the area of normal level, and the last status fell in the warning level, the last warning detection signal is an exception. Therefore, the data instances which persisted from time *t-1* to *t* are discarded.

(2) Assume that the last warning detection signal is NOT an exception. If the present status of concept drift detection falls in the area of normal level, and the last status fell in the warning level, the last warning detection signal is NOT an exception. Therefore, the instances which persisted from time *t-1* to *t* are continued to be stored for training the next prediction model when the next concept drift happens.

(3) Assume that the present drift detection signal is an exception. Because the last detection signal fell in the normal level, and the present status of concept drift detection falls in the normal level, it skipped over the warning level. Therefore, the discard flag does not reset to TRUE to keep the current prediction model.

(4) Assume that the present drift detection signal is NOT an exception. The last detection signal fell in the

normal level, and then the present status of concept drift detection really falls in the normal level suddenly. Therefore, the discard flag is resetting to TRUE to retrain the new prediction model in the same worker by the persisted training dataset.

3.3. Discussions on concept drift

The proposed method uses a DDDAS with Map-Reduce distributed architecture, a statistic method to calculate $\mathrm{p_i}$ and $\mathrm{s_i}$ and confidence interval to detect concept drift. Furthermore, we use *LPred* and *GPred* to get the best prediction result in the current context in a dynamic environment. Compared to traditional machine learning that focuses on stationary distribution data, the proposed method can detect the concept drift phenomenon and efficiently adjust the prediction model. In a dynamic environment, training a model with historical data is a problem because of the noise data and unpredictable changes over different contexts. In the past, handling multiple prediction models was difficult due to the limited computational resources. The proposed method uses a Map-Reduce distributed computing framework, which can easily cluster multiple computer resources to simultaneously run the prediction model with different parameters and related data to analyze concept drifts and detect them. Finally, the metric of *GPred* is assigned by the *LPred* that is produced by each distributed worker, but it uses the highest weight to get the best prediction result.

4. Simulation and Analyses of Results

In this section, the manually generated dataset is applied to verify the proposed method.

4.1. Simulation

In this section, we present the simulation scenarios and environment in order to verify the proposed method: the simulation parameters and evaluation metrics are also shown.

4.1.1. Simulation Assumption

In this experiment, we assume that the unlabeled data arriving in time t can be verified in time: $t+1$. I.e. we will know the labeled data shown at time $t+1$ which was generated at time t.

4.1.2. Simulation Environment

Apache Spark [28] is an open-source Map-Reduce implementation platform. Its core is written in Scala, which is also executed by JVM and has integration with Java. Spark provides wide APIs for different programming languages, including Scala, Java and Python. The key point in using Spark is that it provides a resilient distributed dataset (RDD), which is a collection of elements partitioned across the nodes of the cluster that can be operated in parallel. RDD can be transformed from various sources, such as HDFS, normal text file or other existing programming language collections in the driver program. Furthermore, users can ask Spark to place an RDD in memory and this allows it to be reused efficiently across parallel operations. The other point of using Spark is that it has high integration with Hadoop and it has a better performance. Also, Spark will guarantee quality of service, i.e. it will automatically recover from node failures. We have implemented our method in the dynamic environment which is a time flow concept. In this environment, new data will be coming through and the model generated by the selected algorithms will do the prediction.

4.1.3. Spark Setup and Parameters Settings

The simulation is required to setup Spark first. In this experiment, we clustered three Ubuntu computers as a computation cluster. We set one computer as the Driver, but also as a Worker. The other two computers are set only as the Workers. Hence, we have a total of 3 cores and 6 GBs of memory resources in our cluster. In this experiment, we chose Naïve Bayes Classifiers to build the prediction model. The types of Naïve Bayes Classifier, parameter k of binomial distribution is the result of the amount of received instances at one time unit to divide 3, and the weighting coefficients (α, β) are set to 0.01 and 0.1 respectively.

4.1.4. Case

In this section, the SEA data are generated based on the rule introduced in [21]. The generated SEA data exhibit the concept drift phenomenon. A group of data sets are generated via a specific rule. The data sets are designed to change concepts over time, and to test whether or not the proposed method works.

- **Initial data setting**

In this section, the rule for generating data will be explained. In this experiment, we generate SEA data to validate the proposed method. The basic rule introduced by Street and Kim [21] is considered a concept drift benchmark. The dataset has two classes $c1, c2$ and three features with values between 0 and 10. But only the first two features f_1, f_2 are relevant. The target concept function is (4.1). The concepts have been changed at positions 100, 200 and 300:

$$\begin{cases} c1: f_1 + f_2 > \theta \\ c2: f_1 + f_2 \leq \theta \end{cases} \tag{4.1}$$

We use this formula to generate 500 instances, and set θ in this way:

0 – 100 instances: $\theta = 3$
101 – 200 instances: $\theta = 5$
201 – 300 instances: $\theta = 7$
301 – 500 instances: $\theta = 9$

After generating, the distribution of dataset is shown in Figure 6. We get 10 instances in each time unit to simulate the data stream. If you are interested in this dataset, please contact me by mail for more information and further details of this dataset.

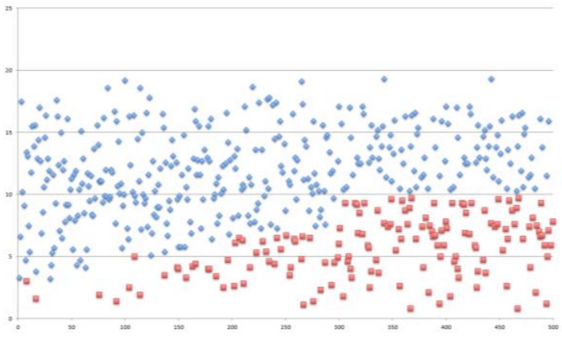


Figure 6. Distribution of dataset with SEA concept.

4.1.5. Evaluation Metrics

In order to evaluate whether or not the proposed method can detect the concept drift phenomenon and improve the prediction accuracy, we use accuracy rate (AR) (4.2) to evaluate each prediction model and (4.3) to evaluate the drift detection rate (DDR). Here, PCi represents the correct prediction instance, *i*, in sequential data, Pi represents the instance *i* that was predicted in the sequential data, DDi represents drift that was detected in sequential data, *i*, and Di represents the drift really happened in sequential data, *i*. Furthermore, we record the total prediction time with processing the specific amount of data in a data stream to evaluate the efficiency.

$$AR = \frac{\sum PCi}{\sum Pi} \tag{4.2}$$

$$DDR = \frac{\sum DDi}{\sum Di} \tag{4.3}$$

4.2. Results and Analyses

In this section, we evaluate the performance of the proposed method by using the metrics: accuracy rate (AR) (4.2), drift detection rate (DDR) (4.3) and execution time. The simulation case result is analyzed in this case, we run three workers and each of them has its own prediction model. The parameter k is set to (total prediction instances in a unit time) / 3. Because there are four adjustment rules (in Section 3.3.2) to apply, each worker's parameter and adjustment setting are listed in Table 2.

Table 2. Each worker's parameter and adjustment setting.

Worker	*k*	Adjustment strategy
Worker 1	(total prediction instances in a unit time) / 3	(1) (3)
Worker 2	(total prediction instances in a unit time) / 3	(2) (3)
Worker 3	(total prediction instances in a unit time) / 3	(1) (4)

After running the experiments, the result shows that Worker 1 has accuracy 0.85 on average, Worker 2 has an average accuracy rate of 0.83 and Worker 3 on average has an accuracy rate of 0.86. Furthermore, the *GPred* held by driver has accuracy 0.88 on average. The entire concept drift that happened in the SEA dataset has been detected, so the drift detection rate is 100%. Figure 7 shows the results of each worker with the proposed method. The accuracy rate with the proposed method will drop down just after concept drift occurs, but it will respond to the new concept and then improve the accuracy rate again.

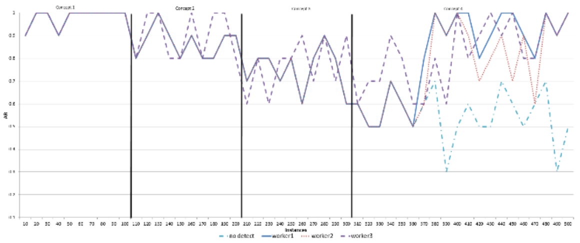


Figure 7. Comparison of each worker and no drift detection.

In Figure 7, we can also see the comparison between the results of each worker with and without the proposed methods. Figure 8 shows the weight of each worker in each state, and Figure 9 shows the result of GPred selected from each prediction based on the weight of each workers.

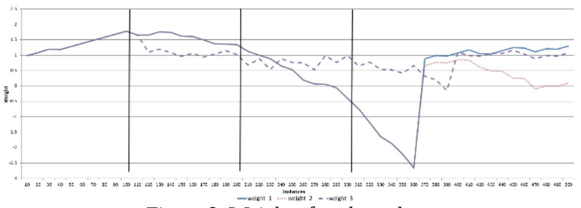


Figure 8. Weight of each worker.

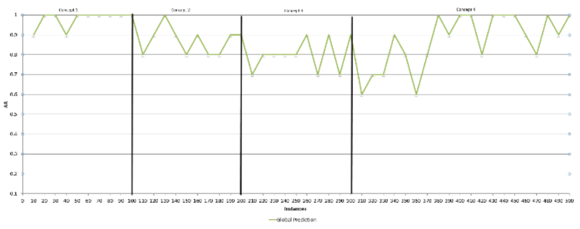


Figure 9. Result of Global Prediction

4.3. Discussions

Table 3 shows the prediction accuracy of all workers in all cases. The worker that did not apply the proposed method decreased its predication accuracy when the concept drift phenomenon happened. Although the accuracy of our method still decreased when concept drift phenomenon happened, it has quickly responded to the new concept and adjusted the model to fit the current context. Hence the other three workers have a higher accuracy compared to workers without the proposed method. Furthermore, according to weighting and selection strategy, the *GPred* gets the highest accuracy on average. The proposed method increases by 14% in accuracy in all cases.

Table 3. Accuracy of model for each worker.

	AR in concept 1	AR in Concept 2	AR in Concept 3	AR in concept 4	Average AR
Without method	0.98	0.87	0.75	0.55	**<u>0.74</u>**
Worker 1	0.98	0.87	0.75	0.82	0.85
Worker 2	0.98	0.87	0.75	0.765	0.83
Worker 3	0.98	0.9	0.77	0.82	0.86

Also, the proposed method uses distributed computing to improve the efficiency. Table 4 shows the execution time with/without distributed computing. It increases by 45% in time efficiency while applying the same amount of dataset to the system. This feature makes the proposed method more suitable for real-time prediction and dynamic environments.

Table 4. Execution time.

	Total execution time	Amount of instances
Local computing	1.1s	500
Distributed computing	0.6s	500

In this research, the proposed method shows the capability of doing on-line predictions and responding to the current contexts by adjusting the prediction model in a dynamic environment. Also, the proposed method is able to select the most reliable prediction result as the final output to the user.

5. Conclusions

Data analysis and prediction problems in a dynamic environment are very important and attractive because the information hidden within the data may be very valuable. Dynamic data driven prediction is a popular research are a due to reported difficulties to use a normal approach to deal with this problem. Hence, this paper proposed a method based on a dynamic data driven application system with parallel concept drift detection and model adjustment. DDDAS can dynamically inject data into a system (stream) and gain their feedback. We use Map-Reduce distributed computing architecture and a statistic method to detect the concept drift phenomenon via a probability model. Furthermore, we use *LPred* and *GPred* to get the best prediction results in the current context in a dynamic environment. Compared to traditional machine learning that focuses on stationary distribution data, the proposed method can detect the concept drift phenomenon and efficiently adjust the prediction model. The experimental results show that the proposed method can detect concept drift in dynamic concept changing environment. This method improves the average predication accuracy about 14%. Furthermore, due to the distributed computing, the proposed method saves almost 10 seconds for 581,012 instances in total processing time which is a significant improvement in the real-time prediction world. In the future, we will look for more realistic datasets as experiment data for testing the proposed approach. More workers are also used in the Map-Reduce distributed architecture to enhance the computing efficiency.

Acknowledgement

This research work was supported by the Ministry of Science and Technology, Taiwan under the grant 104-2410-H-033-025 and 105-2410-H-033-027. We would like to express our gratitude to Po-Chien Chao for his invaluable comments on the early version of this paper.

References

[1] I.-T. T. W. B. R. Series, "Ubiquitous Sensor Networks (USN)," 2008.
[2] K. Ashton, "That 'internet of things' thing," *RFiD Journal,* vol. 22, pp. 97-114, 2009.
[3] P. Mell and T. Grance, "The NIST definition of cloud computing," *National Institute of Standards and Technology,* vol. 53, p. 50, 2009.
[4] A. McAfee and E. Brynjolfsson. (2012). *Big Data: The Management Revolution*.
[5] I. Gartner. (2015). *Gartner Identifies the Top 10 Strategic Technology Trends for 2016*. Available: http://www.gartner.com/newsroom/id/3143521
[6] Gema Bello Orgaz, et.al.,"Social big data: Recent achievements and new challenges," *Information Fusion*, vol. 28, pp. 45-59, 2016.
[7] A. Tsymbal, "The problem of concept drift: definitions and related work," *Computer Science Department, Trinity College Dublin,* vol. 106, 2004.
[8] Y. Chen, H. Amiri, Z. Li, and T.-S. Chua, "Emerging topic detection for organizations from microblogs," in *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, 2013, pp. 43-52.
[9] F. Darema, "Dynamic Data Driven Applications Systems: A New Paradigm for Application Simulations and Measurements," in *Computational Science-ICCS 2004*, ed: Springer, 2004, pp. 662-669.
[10] S.-Y. Lin, "Reinforcement learning-based prediction approach for distributed Dynamic Data-Driven Application Systems," *Information Technology and Management,* vol. 16, pp. 313-326, 2015.
[11] F. Darema, "Introduction to the ICCS 2007 workshop on dynamic data driven applications systems," in *Computational Science–ICCS 2007*, ed: Springer, 2007, pp. 955-962.
[12] C. C. Douglas, R. Loader, J. D. Beezley, J. Mandel, R. E. Ewing, Y. Efendiev, G. Qin, M. Iskandarani, J. Coen, and A. Vodacek, "DDDAS approaches to wildland fire modeling and contaminant tracking," in *Proceedings of the Winter Simulation Conference*, 2006, pp. 2117-2124.
[13] R. Rodríguez, A. Cortés, and T. Margalef, "Data Injection at Execution Time in Grid Environments Using Dynamic Data Driven Application System for Wildland Fire Spread Prediction," in *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, 2010, pp. 565-568.
[14] C. C. Douglas and Y. Efendiev, "A dynamic data-driven application simulation framework for contaminant transport problems," *Computers & Mathematics with Applications,* vol. 51, pp. 1633-1646, 2006.
[15] C. C. Douglas, D. Bansal, J. D. Beezley, L. S. Bennethum, S. Chakraborty, J. L. Coen, Y. Efendiev, R. E. Ewing, J. Hatcher, and M. Iskandarani, "Dynamic Data-Driven Application Systems for empty houses, contaminat tracking, and wildland fireline prediction," in *Grid-Based Problem Solving Environments*, ed: Springer, 2007, pp. 255-272.
[16] G. Allen, "Building a dynamic data driven application system for hurricane forecasting," in *Computational Science–ICCS 2007*, ed: Springer, 2007, pp. 1034-1041.
[17] R. Hirschfeld and K. Kawamura, "Dynamic service adaptation," in *Proceedings of the 24th International Conference on Distributed Computing Systems Workshops*, 2004, pp. 290-297.
[18] S. Wang, S. Schlobach, and M. Klein, "What is concept drift and how to measure it?," in *Knowledge Engineering and Management by the Masses*, ed: Springer, 2010, pp. 241-256.
[19] M. B. Harries, C. Sammut, and K. Horn, "Extracting hidden context," *Machine learning,* vol. 32, pp. 101-126, 1998.
[20] G. Widmer and M. Kubat, "Learning in the presence of concept drift and hidden contexts," *Machine learning,* vol. 23, pp. 69-101, 1996.
[21] W. N. Street and Y. Kim, "A streaming ensemble algorithm (SEA) for large-scale classification," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, 2001, pp. 377-382.
[22] I. Zliobaite, "Learning under concept drift: an overview," Overview", Technical report, Vilnius University, 2009 techniques, related areas, applications Subjects: Artificial Intelligence2009.
[23] S. R. Rangari, S. Dongre, and L. Malik, "A new classifier for handling concept drifting data stream," *International Jour-nal of Science and Research,* vol. 2, pp. 441-444, 2013.
[24] Agustín Ortíz Díaz, José del Campo-Ávila, Gonzalo Ramos-Jiménez, et al., "Fast Adapting Ensemble: A New Algorithm for Mining Data Streams with Concept Drift," The Scientific World Journal, vol. 2015, Article ID 235810, 14 pages, 2015.
[25] N. Littlestone and M. K. Warmuth, "The weighted majority algorithm," *Information and computation,* vol. 108, pp. 212-261, 1994.
[26] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Advances in Artificial Intelligence–SBIA 2004*, ed: Springer, 2004, pp. 286-295.
[27] A. Andrzejak and J. B. Gomes, "Parallel Concept Drift Detection with Online Map-Reduce," in *Proceedings of the IEEE 12th International Conference on Data Mining Workshops (ICDMW)*, 2012, pp. 402-407.
[28] A. Murthy. (2012). *Apache Hadoop YARN – Background and an Overview*. Available: http://hortonworks.com/blog/apache-hadoop-yarn-background-and-an-overview/
[29] A. foundation and G. contributors. (2010). *Apache Spark*. Available: https://github.com/apache/spark