Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 132 (2018) 92–102

Procedia Computer Science

www.elsevier.com/locate/procedia

International Conference on Computational Intelligence and Data Science (ICCIDS 2018)

MRF: MapReduce based Forecasting Algorithm for Time Series Data

Ankita Sinha[a], Prasanta K. Jana[a]

[a]*Department of Computer Science and Engineering,IIT (ISM) Dhanbad, Dhanbad, Jharkhand 826004, India*

Abstract

Weighted moving average (WMA) is extensively used for short term forecasting to remove random fluctuations in cyclical data. However, it is susceptible to the number of previous observations (*window size*), used to predict the next element. A large value of *window size* increases the model risk whereas a smaller one poses parameter risk. For training data size of N, checking the *window size* ranging from 2 to N-1 needs O(N^3) time. In this paper, we propose MRF (MapReduce based forecasting algorithm), which checks the possible window values in parallel and reduces the time complexity to O(N^2). The algorithm meticulously designs $< key, value >$ for each *window size*, which distributes the job equally in the reduce phase. We have shown MRF to be the correct implementation of sequential algorithm. Extensive experiments were performed on MRF. The predicted values for real life meteorological and computing datasets generated using MRF are highly correlated to actual ones. Real and predicted are also validated using paired sample t tests. Results on synthetic data exhibits scalability of MRF algorithm.

Keywords: Time series prediction ; weighted moving average ; Apache Hadoop ; Big data ; MapReduce

1. Introduction

Business Intelligence (BI) [1] has always been fascinated by the analysis of historical data and prediction of probable outcomes for future profit generation. In addition to BI, forecasting techniques are used in weather forecasting [2], power load prediction [3], revenue forecasts [4], sensor networks [5] etc. The data generated in these applications are time series data, i.e., temporal data in which the interval between two consecutive observations is constant. Prediction of future events is critical input into many planning and decision making processes. One of the most widely used schemes for short term forecasting in statistics is moving averages(MA) [6] [7]. It is easy to understand and

* Corresponding author. Ankita Sinha
* Corresponding author. Tel.: +91-326-223-5273 ; fax: +91-326-2296563.
E-mail address: ankitasinha051@gmail.com

10.1016/j.procs.2018.05.167

practical in approach [9] [10]. However for data revolving about a consistent trend weighted moving average (WMA) techniques give more accurate results [8]. In WMA the most recent observation is assigned the largest weight which keeps on decreasing monotonically. One of the major constraints statisticians face while using MA/WMA is deciding upon the number of previous observations to be considered, refereed as *window size*. A smaller *window size* leads to ambiguities caused by wrong parameter selection. A larger *window size* increases the risks posed by the model. It also implies that process of data generation has changed over time period which is covered by the *window* and value of the moving average will change slowly as it is influenced by more number of past values [9].

Although, selection of *window size* is imperative for the quality of prediction [11]. The decision is generally made on the basis of expert judgment or analysts' heuristic [9]. This approach might lead to anomalies as stated above. Moreover, the choice of *window size* depends upon the specific application. A number of candidate solutions, w_1,w_1,...,w_r can be taken to calculate the predicted values. Error between the actual and predicted value is calculated. Objective is to minimize summation of mean squared error (MSE) for the training dataset. Value of w_i at which MSE is minimum corresponds to optimal solution. In the worst case for N observations in the training dataset, all values starting from 2 to N-1 needs to be checked. However, for one window w_i calculation of MSE requires $O(N^2)$ computations [8]. Hence, the time complexity for values ranging from 2 to N-1 will be $O(N^3)$. With the increase in data size affect of higher time complexity becomes more dominant. Data has grown at a tremendous rate in the past decade [12] [13] and will keep on increasing at a faster rate in forthcoming years. The amount of data generated everyday is 2.5 quintillion bytes and about 90% of the world's data has been generated in the last two years [15]. Due to such huge explosion in data generation, use of parallel techniques is inevitable. To store and process large scale data efficiently, Apache Hadoop has emerged as the de facto standard [14] [16]. Hadoop has a simple programming model MapReduce [18][17] and distributed storage solution HDFS (Hadoop Distributed File System) [19]. It provides a highly scalable, fault tolerant, fast, secure and flexible solution to process and store larger datasets.

The concept of optimal *window size* is extensively used in moving average (*MA*) filters [20]. MA filters selects a number of samples, say M samples from the input dataset. Deciding value of M is analogous to optimal window selection. Hence, the proposed algorithm can be applied to solve the problem of deciding M value in MA filters as well. Moving average filters have a variety of application in multiple fields such as in chemometrics [21], marketing and finance [22], and power system applications [23] etc. In spite of the fact that *window size* affects so many applications, the methods to select them has not received much attention [9]. Inoue et. al [9] proposed a technique to minimize the conditional Mean Squared Forecast Error (*MSFE*). However, they claimed that for accurate forecasting, window size selection method should be compared for discrete breaks. MapReduce based prediction for time series data is presented in [10]. They introduced a new framework on R programming environment and Hadoop is utilized for parallelization and fault tolerance support. They mentioned that MapReduce paradigm is not suitable for prediction as to predict any future event a series of observations is required. Hadoop distributes the data implicitly among the nodes in cluster, which makes it unsuitable. On the contrary, in this paper we present a parallel algorithm based on MapReduce for selection of optimal *window size*. We work on data elements in parallel for all possible values of window and meticulously design the *key* and respective *values*. We call the algorithm MapReduce based Forecasting (MRF) algorithm. Time complexity of MRF is $O(n^2)$ which is an improvement over sequential algorithm by a factor of n. The proposed is single pass and scans each element only once.

The remaining sections of this paper are organized as follows. Weighted moving average is discussed in Section 2. The proposed work with an illustrative example is described in Section 3 followed by experimental analysis and results in Section 4. The paper is concluded in Section 5.

2. Weighted moving average

Let n observations be d_1, d_2,..., d_n and w_k, w_{k-1},..., w_1 be the k positive weights assigned to k recent observations d_n, d_{n-1},..., d_{n-k+1} respectively. The *WMA* for this series is defined as follows.

$$WMA_k = \frac{d_n.w_k + d_{n-1}.w_{k-1} + ... + d_{n-k+1}.w_1}{w_k + w_{k-1} + ... + w_1} \tag{1}$$

here, $w_k \geq w_{k-1} \geq \ ... \ \geq w_1$
The value of WMA_k is the forecast value at time $d_{n+\tau}$, here τ is a small positive integer. For simplicity, value of τ is taken as 1. For n observations and k given weights, $k \leq n$. Mean Square Error(MSE) [6] for *n-k+1* predicted values is calculated by using Equation 2.

$$MSE = \sum_{t=k+1}^{n} \frac{\left(d_t - \hat{d}_t\right)^2}{n-k} \quad here \ \ \hat{d}_t = WMA_t \tag{2}$$

Window of size k keeps on sliding till all the observations n are covered in the given dataset. Objective is to choose the optimal value of k such that the *MSE* is minimal. Multiple value of window sizes ranging from $k_1, k_2, ..., k_r$ can be used, with a set of already established weights to find the *WMA* and hence the *MSE*. Value of k_i which gives the minimum *MSE* is chosen for forecasting. The sequential algorithm to find the optimal window size is presented in Algorithm 1.

Algorithm 1 : Sequential algorithm

Require: Time series data
Ensure: Optimal *window size* & *MSE*

1: **Start**
2: **for** $(i = 2 \ to \ w)$ **do**
3: $\Delta_i = \frac{(i \times (i+1))}{2}$
4: $MSE_i = 0$
5: **for** $(j = 2 \ to \ n-1)$ **do**
6: **for** $(r = 1 \ to \ i)$ **do**
7: temp=temp+ $\frac{data_{j+r} \times (r+1)}{\Delta_j}$
8: **end for**
9: $MSE_i = (data_{j+r} - temp)^2$
10: **end for**
11: **end for**
12: **Stop**

3. Proposed work

As stated earlier MapReduce based algorithm performs the computation in parallel fashion. In MRF algorithm, Mapper class reads data from the file system and performs operation on them for different *window size* ranging from 2 to *N*-1, where *N* is the size of training dataset. The *map*() method attaches appropriate *key*, *value* and generates intermediary result. To decrease the network cost, Combiner class is used. It merges output of one mapper based on *key* and decreases load on Reducer. The reduce phase then collect the outputs from Combiner class and finds the minimum *MSE* and corresponding optimal *window size*. An illustrative example of MRF algorithm is presented followed by pseudo code. MRF algorithm makes following assumptions:

- Univariate dataset is considered in the process.
- The dependent and independent variable are numeric.
- There are no missing values.

3.1. Illustrative example

Let, there are 5 data elements {(1:x),(2:y),(3:z),(4:w),(5:r)}. The first element specifies the time stamp and the second stands for record value at that instant. Let the window size range from 2 to 4. Then, for each record, Mapper assigns *key* and *value* as shown in Table 1, and passes them to next phase. Predicted value as well as actual record need to pass to next phase. Tag is attached to distinguish between the actual and predicted values. If the tag is 1, it is the actual record, otherwise the item is an addend of predicted value. Moreover, for w window size, w elements are added to get the predicted value. Hence, the parts of one predicted *value* are given the *key* accordingly, so that they get mapped to the same reducer. Reducer distinguishes between the different *window size* and addends for that *window size* based on the *key*. For example, if key is "2.1", first "2" indicates the *window size*, 2 and "1" specifies that it generates the first predicted item. Similarly, if the *key* is 2.2, it will provide the second predicted value of *window size* 2 and so on. An example is given in Table 2, to get the predicted value of 6th element for different window sizes. Table 3 explains the working of Reducer. A shared data structure *MSE* is used in the Reducer class to keep the track of MSE. It is an array of the size w, where w is max possible size of *window*. On getting each key and list of values it calculates the *MSE*. At the end it finds the minimum *MSE* and corresponding optimal *window size*. Combiner follows the same principle as Reducer. Hence, the working of Combiner is skipped in the illustrative example.

Table 1: Working of Mapper for window size ranging from 2 to 4

Time	Data item	Window size = 2		Window size = 3		Window size = 4	
		Key (Text)	Value (Text)	Key (Text)	Value (Text)	Key (Text)	Value (Text)
1	x	2.1	$\frac{x}{\Delta_2}$,0	3.1	$\frac{x}{\Delta_3}$,0	4.1	$\frac{x}{\Delta_4}$,0
2	y	2.1	$\frac{2y}{\Delta_2}$,0	3.1	$\frac{2y}{\Delta_3}$,0	4.1	$\frac{2y}{\Delta_4}$,0
		2.2	$\frac{y}{\Delta_2}$,0	3.2	$\frac{y}{\Delta_3}$,0	4.2	$\frac{y}{\Delta_4}$,0
3	z	2.1	z,1	3.1	$\frac{3z}{\Delta_3}$,0	4.1	$\frac{3z}{\Delta_4}$,0
		2.2	$\frac{2z}{\Delta_2}$,0	3.2	$\frac{2z}{\Delta_3}$,0	4.2	$\frac{2z}{\Delta_4}$,0
		2.3	$\frac{z}{\Delta_2}$,0	3.3	$\frac{z}{\Delta_3}$,0	4.3	$\frac{z}{\Delta_4}$,0
4	w	2.2	w,1	3.1	w,1	4.1	$\frac{4w}{\Delta_4}$,0
		2.3	$\frac{2w}{\Delta_2}$,0	3.2	$\frac{3w}{\Delta_3}$,0	4.2	$\frac{3w}{\Delta_4}$,0
		2.4	$\frac{w}{\Delta_2}$,0	3.3	$\frac{2w}{\Delta_3}$,0	4.3	$\frac{2w}{\Delta_4}$,0
				3.4	$\frac{w}{\Delta_3}$,0	4.4	$\frac{w}{\Delta_4}$,0
5	r	2.3	r,1	3.2	r,1	4.1	r,1
		2.4	$\frac{2r}{\Delta_2}$,0	3.3	$\frac{3r}{\Delta_3}$,0	4.2	$\frac{4r}{\Delta_4}$,0
		2.5	$\frac{r}{\Delta_2}$,0	3.4	$\frac{2r}{\Delta_3}$,0	4.3	$\frac{3r}{\Delta_4}$,0
				3.5	$\frac{r}{\Delta_3}$,0	4.4	$\frac{2r}{\Delta_4}$,0
						4.5	$\frac{r}{\Delta_4}$,0

Table 2: Predicted value of 6th element

Window size	Predicted value
2	$\frac{2r}{\Delta_2} + \frac{w}{\Delta_2}$
3	$\frac{3r}{\Delta_3} + \frac{w}{\Delta_3} + \frac{z}{\Delta_3}$
4	$\frac{4r}{\Delta_4} + \frac{3w}{\Delta_4} + \frac{2z}{\Delta_4} + \frac{y}{\Delta_4}$

3.2. MRF Algorithm

3.2.1. Weight calculation

Let k be the window size. Triangular summation to k is calculated as follows.

Table 3: Working of Reducer

Input Key	Input value	Processing	Output Key	Output value
2.1	$(\frac{x}{\Delta_2},0),(\frac{2y}{\Delta_2},0),(z,1)$	MSE[2]= MSE[2]+ $\{z-(\frac{x}{\Delta_2}+\frac{2y}{\Delta_2})\}^2$	Optimal *window size*	Minimum *MSE*
2.2	$(\frac{y}{\Delta_2},0),(\frac{2z}{\Delta_2},0),(w,1)$	MSE[2]= MSE[2]+ $\{w-(\frac{y}{\Delta_2}+\frac{2z}{\Delta_2})\}^2$		
3.1	$(\frac{3z}{\Delta_3},0),(\frac{2y}{\Delta_3},0),(\frac{x}{\Delta_3},0),(w,1)$	MSE[3]=MSE[3]+ $\{w-(\frac{x}{\Delta_3}+\frac{2y}{\Delta_3}+\frac{3z}{\Delta_3})\}^2$		
3.2	$(\frac{3w}{\Delta_3},0),(\frac{2z}{\Delta_3},0),(\frac{y}{\Delta_3},0),(r,1)$	MSE[3]= MSE[3]+ $\{r-(\frac{y}{\Delta_3}+\frac{2z}{\Delta_3}+\frac{3w}{\Delta_3})\}^2$		
4.1	$(\frac{4w}{\Delta_4},0),(\frac{3z}{\Delta_4},0),(\frac{y}{\Delta_4},0),[(\frac{x}{\Delta_4}],0),(r,1)$	MSE[4] = MSE[4]+ $\{r-(\frac{x}{\Delta_4}+\frac{2y}{\Delta_4}+\frac{3z}{\Delta_4}+\frac{4w}{\Delta_4})\}^2$		

$$\Delta_k = 1+2+...+k$$

Δ_k is denominator for weights in window k. Position of element is numerator. The most recent observation is assigned weight k which decreases by a factor of 1, till it reaches 1. The weights in decreasing order of magnitude for window size k are as follows:

$$\frac{k}{\Delta_k}, \frac{k\text{-}1}{\Delta_k}, ..., \frac{1}{\Delta_k}$$

This weight calculation is performed dynamically depending upon the window size k by MRF algorithm.

3.2.2. Map function

Algorithm 2 : *map(key,value)*

Require: *max window size* & input record
Ensure: Addends of predicted value

1: **Start**
2: **for** $(j = 2\ to\ w)$ **do**
3: count =0
4: $\Delta_j = \frac{(j\times(j+1))}{2}$
5: **while** $(j \geq count)$ **do**
6: $k' = j + '.' + (i - (count - 1))$
7: $v' = \frac{count}{\Delta_j} \times data$
8: ***context.write***(k', v')
9: $count$ ++
10: **end while**
11: **end for**
12: **Stop**

Map takes input from HDFS in form of $< key, value >$ pair. Input *key* is byte offset of record and input value is the actual observation. Time stamp starts from 1, in the presented work. However, the algorithm can be easily modified to work for real time occurrence. Algorithm 2 explains the working of Mapper class. On receiving input, Mapper calculates weighted addends by multiplying the observation with weight, as explained in previous section. Mapper also passes the original value to the next phase. In order to distinguish between the addends of predicted item and actual item, tag is attached to each value.

3.2.3. Combine function

Combiner combines the intermediate output of the each map task based on their *key*. Intermediate data are not written back to the file system, rather stored in local disk of the host. Hence, no communication cost is incurred by using the combiner. Combiner class uses the same method *reduce*() as Reducer class. For a unique *key* the combiner scans corresponding values. If tag bit is 1, data is passed without any computation to the Reducer. Otherwise, if tag bit is 0, Combiner computes summation of the values, and pass the *key* and summation to the next phase.

3.2.4. Reduce function

The input to Reducer is output from Combiner. Working of Reducer is almost same as Combiner. However, to keep track of the Mean Square Error, a shared variable *MSE* is used for each *window size*. The initial value of MSE is set to zero for each *window size*. The pseudo code for Reducer is given in Algorithm 3. On receiving a *key* and corresponding list of *values*, the Reducer checks tag bit, if the tag bit is 1, it identifies the content as actual value and the sums other values with tag bit 0 to find the predicted value. To find the error, predicted value is subtracted from actual value and added to the *MSE* value at that *window size*. After scanning the input, Reducer finds minimum of *MSE* and produces corresponding optimal *window size* with minimum *MSE* as output.

Algorithm 3 : *reduce* < *key*, *list*(*values*) >

Require: output of Combiners k'',list < v'' >
Ensure: Optimal *window size & minimum MSE*

A shared variable MSE is initialized to 0 for each window size from (2 *to W*)
predicted=0
while ($v''.hasNext()$) **do**
  if ($tag = 1$) **then**
    $actual = v''.value$
  else
    predicted=*predicted*+$v''.value$
  end if
end while
Split k'' into *window size* & *addend number*
w=*window size*
$MSE_w = MSE_w + (actual - predicted)^2$
minMSE=minimum(*MSE*)
$Optimal\_window\_size = IndexOf(minMSE)$
$keyFinal = Optimal\_window\_size$
$valueFinal = minMSE$
output.collect($keyFinal, valueFinal$)
Stop

3.3. Complexity analysis

The computational efficiency of MRF depends on the level of parallelism, which is dependent on the number of maps and reduce tasks. The programmer has no control over the number of map tasks as it depends on the data size and the block size in HDFS. The number of reduce tasks is governed by the number of unique *key* generated. For a chunk of data assigned to reduce phase, *reduce*() method runs for each unique *key* [17]. Hence, in some applications where many *values* gets mapped to the same *key*, the problem of data skew arises. This will cause variable running time of the reduce tasks leading to higher total execution time and lower the resource utilization [24]. However, in our proposed MRF algorithm, different keys are generated for all the window sizes. Therefore, problem of data skew does not arises. The number of *keys* generated is directly proportional to input data size and can be calculated as follows. For a particular window size *w* the number keys generated is (*N*-*w*+1). Here *N* is training data size and window size ranges from 2 to *N*-1. Total number of keys is given by equation 3.

$$Total\ number\ of\ keys = \sum_{i=2}^{N-1}(N - i + 1) \equiv \frac{N(N+1)}{2} \tag{3}$$

Hence, in MRF algorithm there is no problem of data skew and the execution time is directly proportional to number of reducers. Higher the number of reducers lesser will be the execution time. In an ideal situation number of map and reduce tasks are directly proportional to the number of nodes on the cluster, where map task is directly equal to number of nodes and reduce tasks are 0.95 or 1.75 times the number of nodes [18].

4. Experiments and results

We performed experiments of our proposed work on a heterogeneous fully distributed Hadoop cluster. Server machine, ML 350E with Gen 8 with Intel Xeon E5-24070 @2.20 GHz CPU and 24 GB of RAM was configured into 3 virtual machines. First VM was allocated 8GB RAM and 1 CPU core, and acted as master. Remaining two VMs were working as slaves machines had 6GB RAM and 1 CPU core. Other nodes in the cluster were laptop workstations with Intel *i*7 gen4 and Intel *i*3 gen 2 processors with 8GB RAM. Nodes were connected using a 100Mbps Ethernet switch. Hadoop is an open source software which can be downloaded from Apache website [16]. MapReduce codes are written in Java, version "1.7.0".

4.1. Datasets

Real life datasets from meteorology and Internet traffic are used to examine the accuracy of proposed MRF algorithm. The dataset were downloaded from Hyndman, R.J. Time Series Data Library [26]. The description of real life dataset is present in table 4. In addition to real life dataset synthetic dataset of various size were used to test the scalability of MRF algorithm. Size of synthetic dataset was varied from 100MB to 1.5GB.

Table 4: Dataset description

Dataset description	Category	Time period	Number of observations
Daily minimum temperatures in Melbourne, Australia	Meteorology	1981-1990	3650 fact values in 1 timeseries
Internet traffic data (in bits) from a private ISP with entries in 11 European cities	Computing	06:57 hours on 7 June to 11:17 hours on 31 July 2005. Hourly data.	1231 fact values in 1 timeseries

4.2. Experimental analysis

Table 5: Optimal *window size* for real life datasets

Dataset	MRF	Sequential Algorithm
Internet traffic data	2	2
Daily minimum temperatures in Australia	9	9

In this section, execution time of MRF algorithm with respect to sequential algorithm is evaluated. Table 5 shows the optimal *window size* for the real life datasets. It can be deducted from the results that MRF is correct implementation of the sequential algorithm. Further,it is also required analyse performance gain of parallel algorithm over sequential one. Table 6 shows execution time of both the algorithms. It is to be noted that for very small sized data, sequential algorithm outperforms the parallel one because of the overhead incurred in parallelization. Hadoop makes a trade off between the computation and communication. In case of large size data, the communication overhead is negligible in comparison to computation. However, in case of smaller files, communication cost will be higher than computation efficiency achieved. Hence, for very small sized data sequential algorithm is preferred over the parallel one. As the size of data increases it can be clearly observed from Table 6, that the execution time of sequential algorithm increases by a larger factor than MRF. Moreover, with increase in data size sequential algorithm encounters

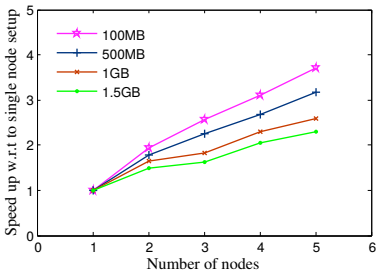


Fig. 1: Speed up of MRF

heap space error. To measure the speedup of MRF algorithm, the size of dataset was kept constant and the number of nodes in the cluster was increased up to 5. The baseline to calculate speedup was the execution time of MRF on a single node Hadoop cluster. It can be deduced from the Figure 2 that with addition of more nodes in cluster MRF algorithm shows good speed up with synthetic dataset.

Table 6: Execution time of Sequential algorithm and MRF

Number of observations	Execution time			
	Sequential algorithm	MRF on 2 node cluster	MRF on 3 node cluster	MRF on 4 node cluster
1000	3.66 sec	20 sec	24sec	31sec
10000	919 sec	32sec	27sec	22sec
100000	42min 27sec	38sec	29sec	25sec
200000	3 hour 37min	42sec	32sec	28sec

4.3. Statistical tests

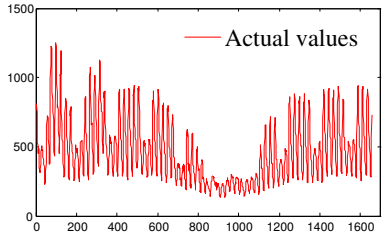
4.3.1. Paired samples correlation

The actual and predicted value for the real life datasets are shown in Figure 3. It is observed that predicted values follow same trend as actual ones. However, in order to statistically find closeness level of the values, correlation between them is calculated. The correlation coefficient r_{xy} between two variables x and y is defined as follows.

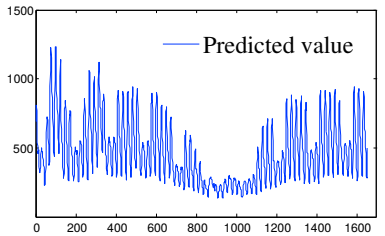
$$r_{xy} = \frac{cov(x, y)}{\sqrt{var(x)} \times \sqrt{var(y)}} \tag{4}$$

$$where \quad cov(x, y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x}) \times (y_i - \bar{y})}{n - 1} \tag{5}$$

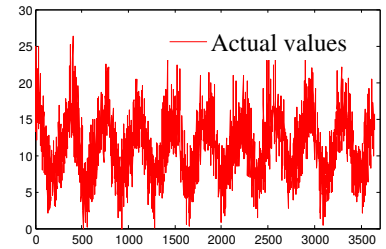
$$and \quad var(x) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n - 1} \tag{6}$$



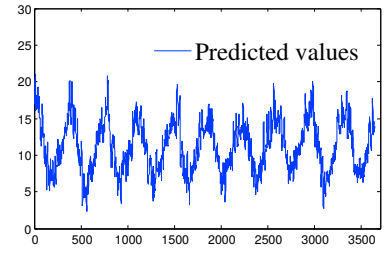
(a) Actual values for Internet traffic data in KB



(b) Predicted values for Internet traffic data in KB



(c) Actual values for daily maximum temperatures in Melbourne, Australia, 1981-1990 in degrees Celcius



(d) Predicted values for daily maximum temperatures in Melbourne, Australia, 1981-1990 in degrees Celcius

Fig. 2: Actual and predicted values for real life datasets

The value of r_{xy} lies in the range of -1 to +1 i.e.$r_{xy} \epsilon$ [-1,+1] . The sign specifies the direction of the relationship and the magnitude indicated how strong the relationship is. If r_{xy} is -1, it indicates that there is a perfectly negative linear relationship. If tr_{xy} is 0, there is no relationship and if r_{xy} is +1, then it specifies that there exists a perfectly positive linear relationship. However, if value of r_{xy} is greater than 0.7, then the relationship is considered to be a strong uphill (positive) linear relationship. Table 7 shows the paired sample correlations for the actual values and the predicted value generated by MRF. It is observed that r_{xy} for internet traffic and Melbourne minimum temperature datasets are 0.948 and 0.847 respectively. Therefore, prediction results generated by MRF are strongly positively correlated to the actual values.

Table 7: Paired samples correlation values

Dataset	No.of predicted observations	Correlation
Internet,traffic data	1655	0.948
Daily,minimum temperatures in, Australia	3641	0.847

4.3.2. Paired sample t test

To check mean difference between actual and predicted values using MRF, paired sample *t* test is applied [27] [25]. This test considers of two competing hypotheses. One is the null hypothesis and other is the alternate hypothesis. The null hypothesis assumes that the true mean difference between the paired samples are zero, i.e. the paired population means are equal. Conversely, the alternate hypothesis assumes that the true means difference between the two samples is not equal to zero and hence the paired sample means are unequal. The mathematical representation of the hypotheses are given as follows.

$$H0: \mu_1 = \mu_2$$
$$H1: \mu_1 \neq \mu_2$$

Table 8: Paired samples *t* test over real life datasets

Dataset	Paired Differences					*t*	*df*	Sig. 2-tail
	Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference				
				Lower	Upper			
internet-traffic-data -in-bits,	.00162	2.16377	.03586	-.06869	.07192	.045	3640	0.964
Daily minimum temp in Melbourne, Australia	-.5602	7148.78	175.7	-345.226	344.11	-.003	1654	.997

Here, μ_1 and μ_2 represents mean of actual and predicted values respectively. While calculation of paired sample *t* test, the actual values are the independent variable whereas the predicted values are dependent variable. Paired sample *t* has the following assumptions.

- Dependent variable must be continuous and normally distributed.
- The observations should be independent of each other.
- The dependent variable should not contain outliers.

The above mentioned assumptions are fulfilled by our subjected data. Hence, this test is applied to check the statistical equivalence of actual and the predicted values generated using MRF. The test statics for this test are described as follows.

$$t = \frac{\bar{d}}{s/\sqrt{n}} \tag{7}$$

Here, $\bar{d}$ represents the mean of the sample difference, n is the number of observations, s is the sample standard deviation of the difference. The two samples were analysed using IBM SPSS version 20. The confidence interval was maintained at 95%. The results in Table 8 shows output generated by comparing actual values of two real life datasets with the predicted values.

In order to verify the null hypothesis, the p value i.e. Sig. (2-tailed) value corresponding to the test statics is checked. If the value is greater than 0.05, the null hypothesis is accepted, otherwise rejected. The p value for the datasets is observed to be greater than 0.05, hence the null hypothesis is accepted in this case. Therefore, it is concluded that the actual and predicted values using MRF are not different statistically and mean difference between them is zero.

5. Conclusion

In this paper, we have presented MRF, a MapReduce based algorithm to select optimal *window size* for weighted moving average based prediction of time series data. The proposed technique works on each data element in parallel and is non iterative. It has been proved that optimal *window size* by MRF is same as the sequential version. It is also shown that actual and predicted values using MRF are highly correlated and true mean of the two values are equal. In future, the attempts would be to extend MapReduce parallel programming paradigm to other prediction techniques as well apart from WMA. Moreover, MapReduce programming paradigm is designed to process big data. Hence, the attempts would be to upgrade the algorithm, such that it can handle big time series data efficiently.

Acknowledgements

The authors would like to thank Council of Scientific and Industrial Research (CSIR), New Delhi, India for the financial support for this research work (File No.09/085(0111)/2014.EMR.1).

References

[1] Chen H, Chiang RH, Storey VC. Business intelligence and analytics: from big data to big impact. MIS quarterly. 2012 Dec 1:1165-88.

[2] Buehner M, McTaggart-Cowan R, Beaulne A, Charette C, Garand L, Heilliette S, Lapalme E, Laroche S, Macpherson SR, Morneau J, Zadra A. Implementation of deterministic weather forecasting systems based on ensemblevariational data assimilation at Environment Canada. Part I: The global system. Monthly Weather Review. 2015 Jul;143(7):2532-59.

[3] Quan H, Srinivasan D, Khosravi A. Short-term load and wind power forecasting using neural network-based prediction intervals. IEEE transactions on neural networks and learning systems. 2014 Feb;25(2):303-15.

[4] Krause GA, Douglas JW. Organizational structure and the optimal design of policymaking panels: Evidence from consensus group commissions revenue forecasts in the American states. American Journal of Political Science. 2013 Jan 1;57(1):135-49.

[5] Zhuang Y, Chen L, Wang XS, Lian J. A weighted moving average-based approach for cleaning sensor data. InDistributed Computing Systems, 2007. ICDCS'07. 27th International Conference on 2007 Jun 25 (pp. 38-38). IEEE.

[6] Montgomery DC, Jennings CL, Kulahci M. Introduction to time series analysis and forecasting. John Wiley & Sons; 2015 Apr 27.

[7] YYaffee RA, McGee M. An introduction to time series analysis and forecasting: with applications of SAS and SPSS. Elsevier; 2000 May 12.

[8] Jana PK, Sinha BP. Fast parallel algorithms for forecasting. Computers & Mathematics with Applications. 1997 Nov 1;34(9):39-49.

[9] Inoue, Atsushi, Lu Jin, and Barbara Rossi. "Rolling window selection for out-of-sample forecasting with time-varying parameters." Journal of Econometrics 196.1 (2017): 55-67.

[10] Li, L., Noorian, F., Moss, D. J., & Leong, P. H. (2014, August). Rolling window time series prediction using MapReduce. In Information Reuse and Integration (IRI), 2014 IEEE 15th International Conference on (pp. 757-764). IEEE.

[11] Pesaran MH, Pick A. Forecast combination across estimation windows. Journal of Business & Economic Statistics. 2011 Apr 1;29(2):307-18.

[12] Sinha A, Jana PK. Clustering Algorithms for Big Data: A Survey. The Human Element of Big Data: Issues, Analytics, and Performance. 2016 Oct 26:143.

[13] Chen CP, Zhang CY. Data-intensive applications, challenges, techniques and technologies: A survey on Big Data. Information Sciences. 2014 Aug 10;275:314-47.

[14] White T. Hadoop: The definitive guide. " O'Reilly Media, Inc."; 2012 May 19.

[15] IBM, Big Data and Analytics,*https* : //*www.ibm.com* Accessed on 20th January2018

[16] Apache Hadoop *https* : //*hadoop.apache.org*. Accessed on 10th October2017

[17] Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters. Communications of the ACM. 2008 Jan 1;51(1):107-13.

[18] MapReduce *https* : //*hadoop.apache.org*/*docs*/*r*1.2.1/*mapred_tutorial.html*. Accessed on 10th October2017

[19] Borthakur D. The hadoop distributed file system: Architecture and design. Hadoop Project Website. 2007 Aug;11(2007).

[20] Golestan S, Ramezani M, Guerrero JM, Freijedo FD, Monfared M. Moving average filter based phase-locked loops: Performance analysis and design guidelines. IEEE Transactions on Power Electronics. 2014 Jun;29(6):2750-63.

[21] Otto M. Chemometrics: statistics and computer application in analytical chemistry. John Wiley & Sons; 2016 Dec 27.

[22] Taylor MP, Allen H. The use of technical analysis in the foreign exchange market. Journal of international Money and Finance. 1992 Jun 1;11(3):304-14.

[23] Wang J, Liang J, Gao F, Zhang L, Wang Z. A method to improve the dynamic performance of moving average filter-based PLL. IEEE Transactions on Power Electronics. 2015 Oct;30(10):5978-90.

[24] Gufler B, Augsten N, Reiser A, Kemper A. Handling Data Skew in MapReduce. Closer. 2011 May 8;11:574-83.

[25] Mitchell, T. M. Machine learning. "Mc G raw Hill"; 1997.

[26] Hyndman, R.J. Time Series Data Library, http://data.is/TSDLdemo. Accessed on 10th October2017

[27] Morgan, George A., et al. IBM SPSS for introductory statistics: Use and interpretation. Routledge, 2012.