Time series forecasting on distributed systems

Xiaoqian Wang

Oct 11, 2019

Time series decomposition and predictive analytics using MapReduce framework

Expert Systems With Applications

https://www.sciencedirect.com/science/article/pii/S0957417418305943

Outline

- **Introduction**
- **Methodology & Application**
- **Conclusion**

Introduction

Introduction

Background

- Precision agriculture
- Weather forecasting → e.g., rainfall predictions, temperature change, humidity analysis, identifying frost conditions
- Data: sensing and Information & Communications Technology (ICT) based equipment
- Challenge: volume, velocity, variety, real-time operations capability, large storage and computational processing power
- Solution: MapReduce-based approach

Introduction

This paper

- Focus on temperature data analysis, decomposition, and forecasting.
- Propose a method based on the MapReduce framework.
- Propose a new decomposition approach on big data platform.
- M-ARIMA: to forecast temperature and decompose components.
- M-KNN & M-HM: to increase the forecasting accuracy and handle both linear-nonlinear components easily.
- The proposed approach are better in terms of accuracy, speed-up and size up.

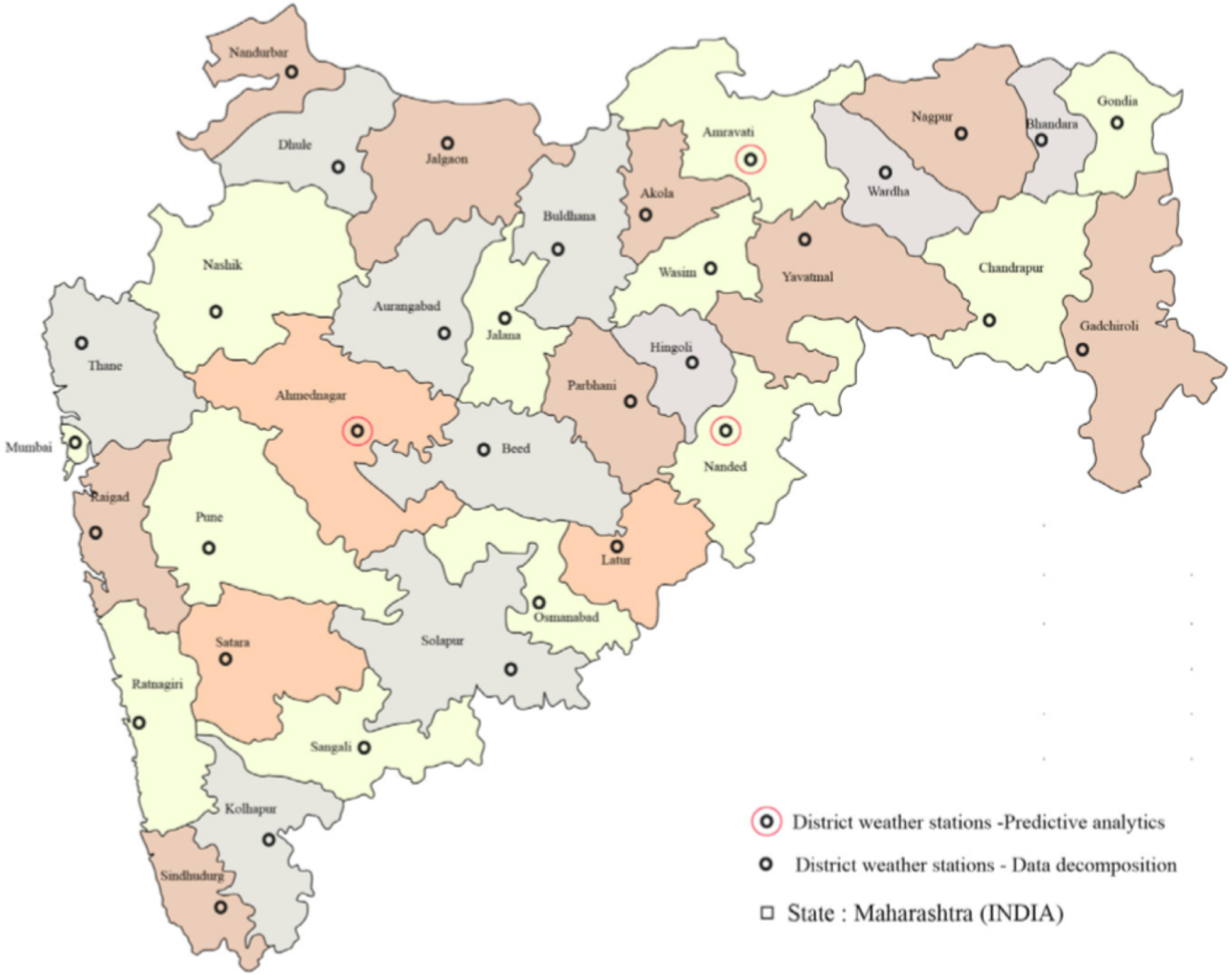
Methodology & Application

Methodology & Application

Data

- Weather data:
  - generated by site-specific stations located in various district areas of Maharashtra state.
  - 39 weather stations, 100 years old historical minimum, average and maximum temperature, humidity and more variables data.
- Data for analytics:
  - 39 weather stations monthly average maximum temperature data.
- Data for testing the performance:
  - 3 different stations monthly maximum temperature data.

Methodology & Application

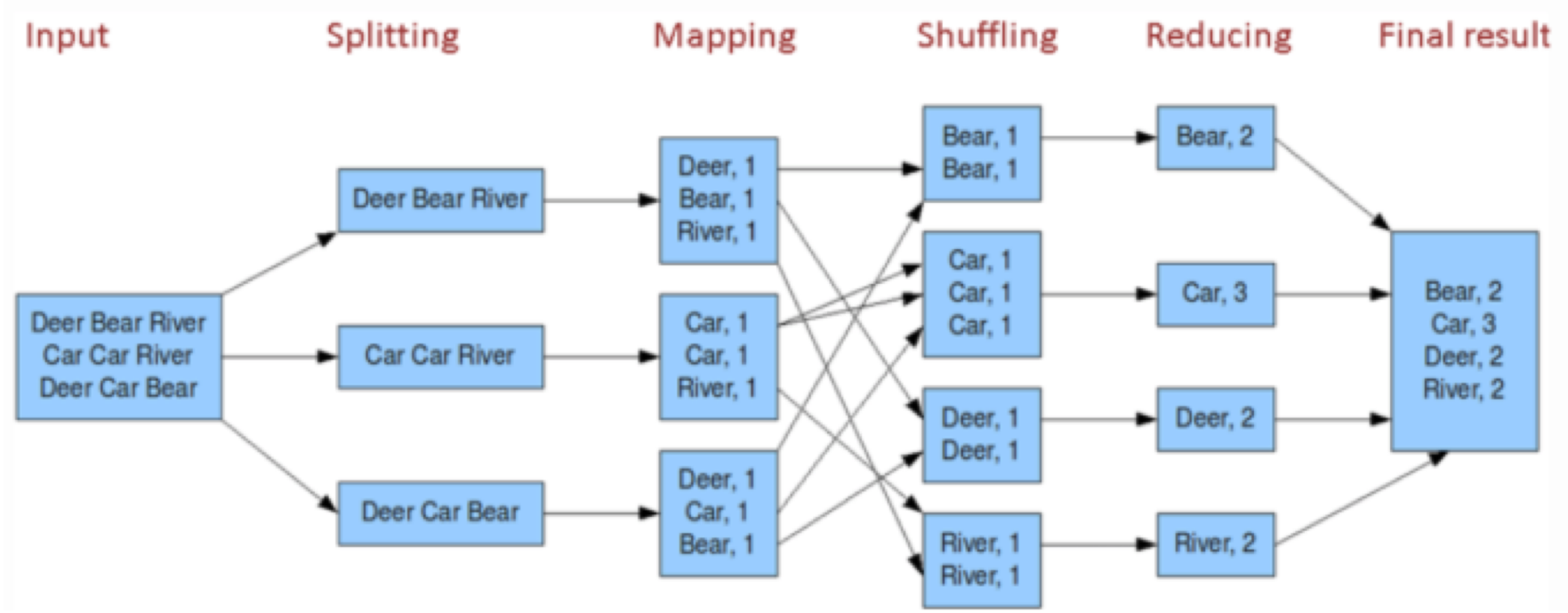


Methodology & Application

- Pre-processing:
  - **temperature** (−100 to 100°C)
  - humidity (1 to 100%)
  - rainfall (0 to 1000mm)
  - values exceeds the conditions are replaced with averaging method
- Input (historical period): 1901–1992
- Forecasting period: 1993–2002

Methodology & Application

MapReduce example



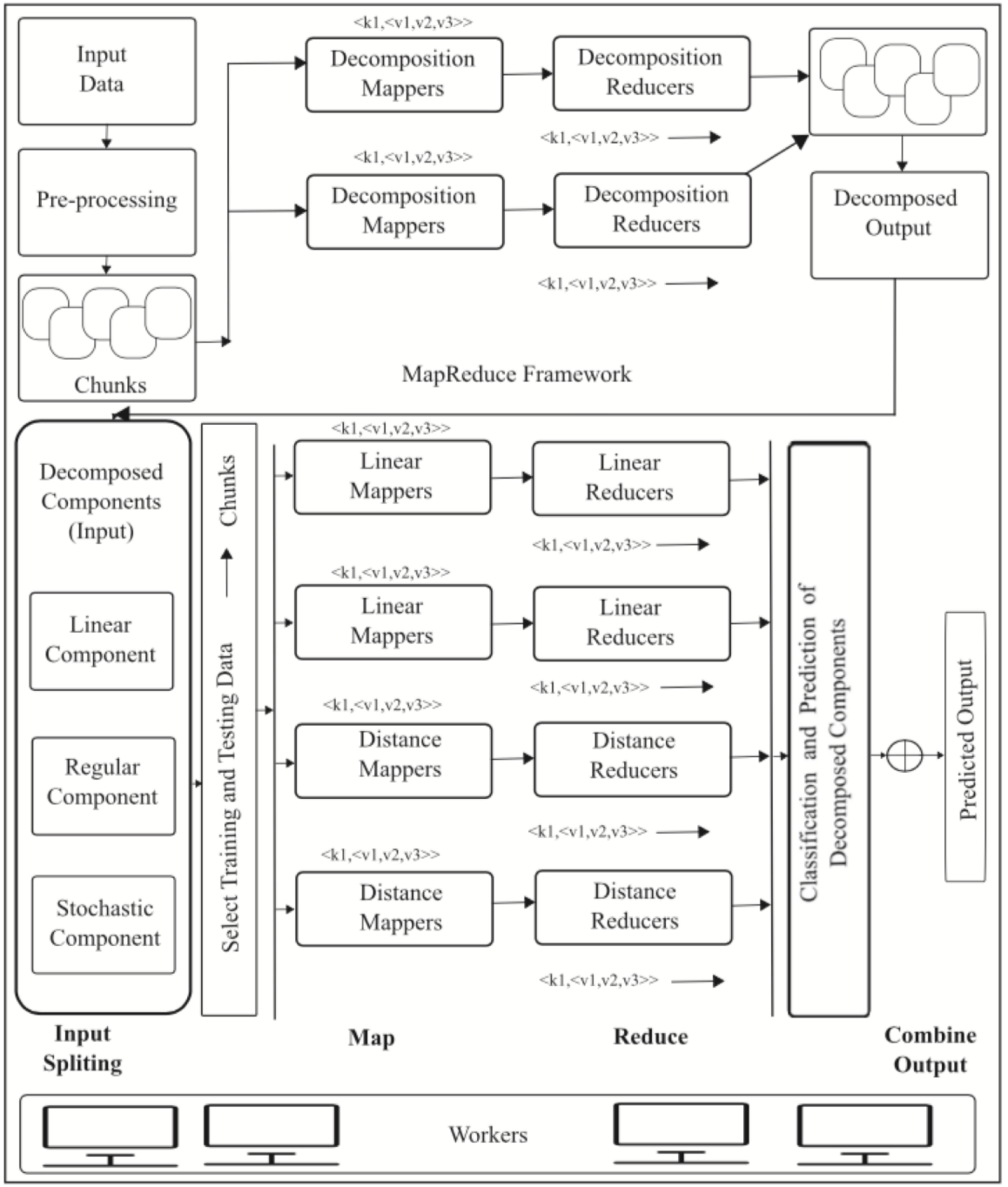
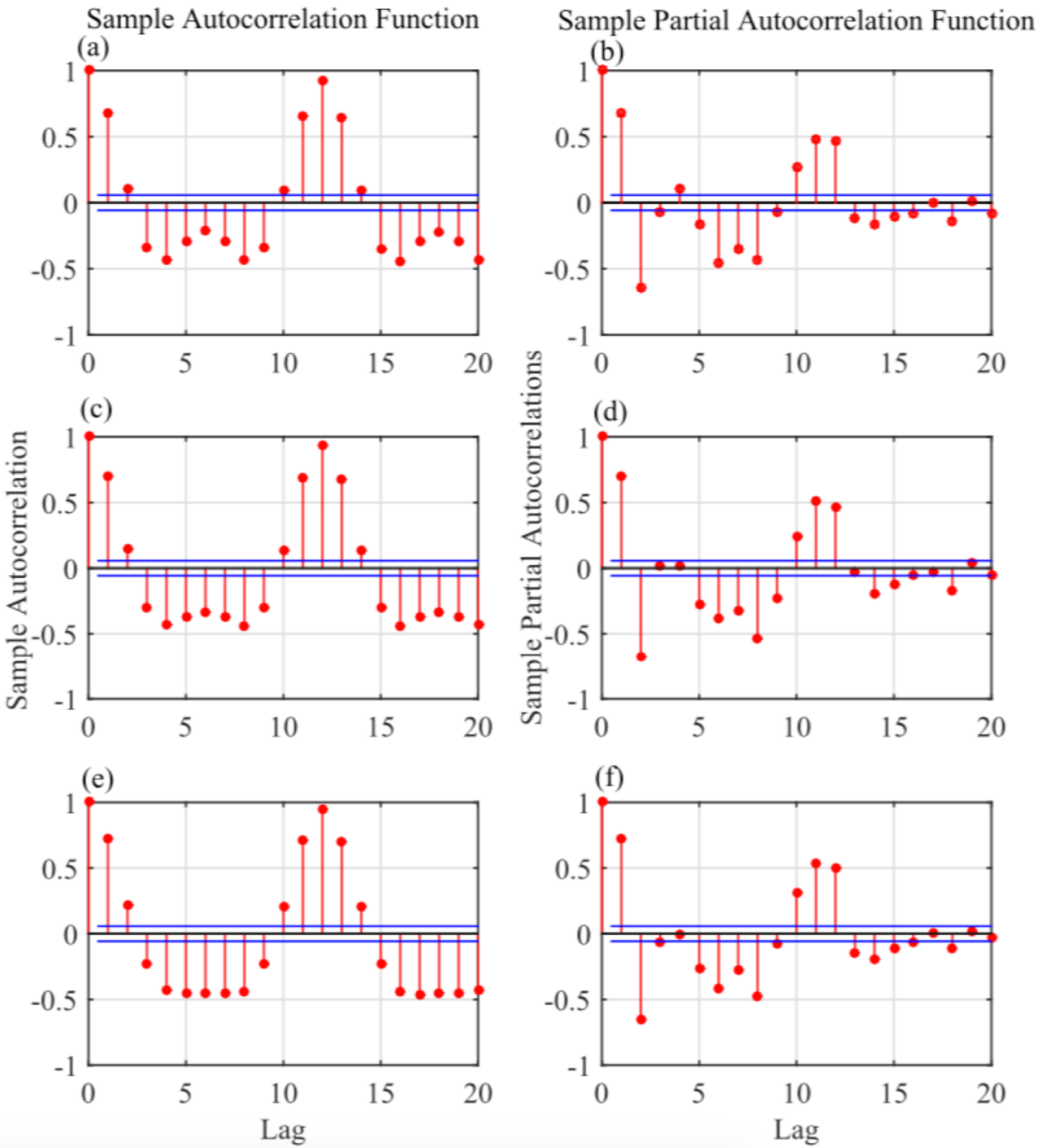


Fig. 3. MapReduce approach and its data flow.

Methodology & Application

Data analysis

- 3 different locations weather station data
- ACF & PACF
- ACF is a tool for determining stationarity.
  - If a time series is nonstationary, the ACF will not die out quickly.
  - If the ACF does not die out even for large lags then autocorrelation for the first difference is computed.


Sample Autocorrelation Function
Sample Partial Autocorrelation Function
(a)
(b)
(c)
(d)
(e)
(f)
Sample Autocorrelation
Sample Partial Autocorrelations
Lag
Lag
1
0.5
0
-0.5
-1
0
5
10
15
20

Methodology & Application

- The ACF of these plots shows linear decay (nonstationarity) in the data.
- Seasonal pattern present.
- If seasonality is present, then it removed by backward difference e.g. $\nabla\nabla^{12} x_t$.
- Conclusions:
  - ACF and PACF plot help to decide p, d, q.
  - But the strong component is seasonality which is present in the repeated form in the time series. Then... we have to decompose time series and find patterns in data for forecasting.

Methodology & Application

Decomposition approach

- $x_t = \underbrace{t_t + p_t}_{\text{seasonal/regular component}} + s_t, t = 1,2,3, \dots, n.$

- Trend (moving average)

$$\widehat{m_t} = \frac{x_{t-q} + x_{t-q+1} + \cdots + x_{t+q} + x_{t+q+1}}{d}, q < t \leq n - q.$$

- Periodic pattern

$$\widehat{p_k} = \begin{cases} w_k - d^{-1} \sum_{i=1}^{d} w_i, k = 1, \dots, d, \\ p_{k-d}, \qquad\qquad k > d. \end{cases}$$

Algorithm 1 MapReduce based decomposition approach.

Map Class
Input: (*key* ← `name of the input;` *value* ← `value of the input`)
Output: (*key*, *value*)
Map (*key*, *value*)
$t_s \leftarrow$ `time series signal to decompose`, $n \leftarrow$ `lenght of` t_s, $d \leftarrow$ `time window`
$trend \leftarrow \hat{m}_t = (x_{t-q} + x_{t-q+1} + \ldots + x_{t+q} + x_{t+q+1})/d$, $q < t \leq n - q$, # $\hat{m}_t$ moving average trend and $q = d/2$ (Eq.4)
$\hat{p}_k = \begin{cases} w_k - d^{-1} \sum_{i=1}^{d} w_i, & k = 1, \ldots, d, \\ \hat{p}_{k-d}, & k > d, \end{cases}$ # periodic pattern (Eq.5)
`ramaining data` $\leftarrow t_s$ `- periodic pattern`
$r_t \leftarrow m_t + p_t$ # r_t regular component or seasonal component
$s_t \leftarrow t_s - r_t$ # s_t stochastic component
Intermediate: ('*key*', *value*) # add multiple values such as m_t, p_t, r_t and s_t
Reduce Class
Input: (*key* ← `name of the mapped data;` *value* ← `list of all map data`)
Output: `Key of the mapped data into row and column`
Reducer (*key*, *value*)
while values.hasNext( ) **do**
data = getnext(value)
Output: (*key*, *value*) # return the final output

Methodology & Application

M-ARIMA approach

- The data have a seasonal component.
- Seasonal ARIMA model: $\mathrm{ARIMA}(p,d,q)\times(P,D,Q)_m$.
- Box-Jenkins time series approach.
  - Identification: differencing & configuring AR and MA.
  - Estimation.
  - Diagnostic checking: overfitting & residual errors.

Algorithm 2 M-ARIMA forecasting algorithm.

1: **Map Class**
2: **Input:** (*key* ← `name of the input;` *value* ← `value of the input)`
3: **Output:** (*key*, *value*)
4: **Map** (*key*, *value*)
5: `Initialize p, d and q values from the ACF and PACF analys`
6: *seasonality* ← 12 # seasonality monthly average input values of the year
7: `ARIMA(p,d,q)`
8: `forecast(mdl, data)`
9: **Intermediate:** (*key*, *value*)
10: **Reduce Class**
11: **Input:** (*key* ← `name of the mapped data;` *value* ← `list of all map data)`
12: **Output:** `Key of the mapped data into row and column`
13: **Reducer** (*key*, *value*)
14: **while** values.hasNext() **do**
15: data = getnext(value)
16: **Output:** (*key*, *value*) # return the final output

Methodology & Application

M-KNN approach

- Time series forecasting with KNN in R: the **tsfknn** package.
  - https://github.com/franciscomartinezdelrio/tsfknn

```
library(tsfknn)
pred <- knn_forecasting(ts(1:8), h = 1, lags = 1:2, k = 2)
knn_examples(pred)
```

```
##      Lag2 Lag1 H1
## [1,]    1    2  3
## [2,]    2    3  4
## [3,]    3    4  5
## [4,]    4    5  6
## [5,]    5    6  7
## [6,]    6    7  8
```

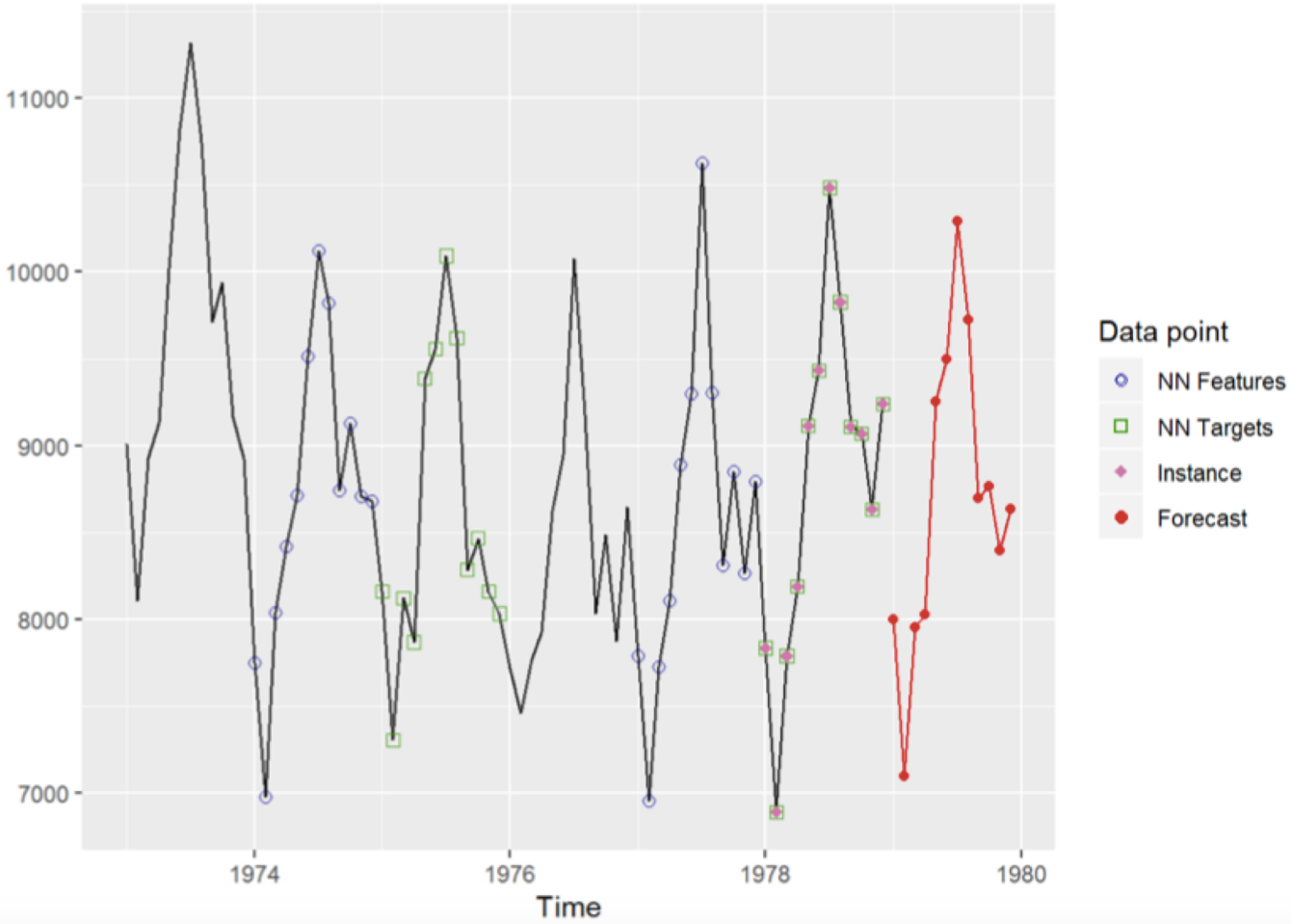
```
pred$prediction
```

```
## Time Series:
## Start = 9
## End = 9
## Frequency = 1
## [1] 7.5
```

Methodology & Application

- Multi-step ahead strategies.

```
pred <- knn_forecasting(USAccDeaths, h = 12, lags = 1:12, k = 2, msas = "MIMO")
autoplot(pred, highlight = "neighbors", faceting = FALSE)
```



Methodology & Application

M-KNN approach

- In this paper ...
  - The whole time series is distributed into training (T_{train}) and testing (T_{test}) samples. (90% & 10%)
  - (x, y) training sample & (x', y') testing sample.
  - The basic Euclidean distance (d) function is used evaluate similarity as:

$$d_i(x, y) = \sum_{j=1}^{n} \sqrt{(x_j - y_j)^2}$$

  - The y' is evaluated as:

$$y' = \text{argmax} \sum_{(x_i, y_i) \in T_{train}} C(c = c_{y_i})$$

Algorithm 3 M-HM model with M-KNN forecasting algorithm.

Map Class
Input: (*key* ← `name of the input;` *value* ← `value of the input`)
Output: (*key*, *value*)
Map (*key*, *value*) # Distance mapper
$T \leftarrow$ `time series`
stringlen ← `length of time series`
$k \leftarrow$ `number of nearest neighbors`# where, $k = 3$ used in this study
$T_{test} \leftarrow$ `testing sample`
$T_{train} \leftarrow$ `training sample`
if $T_{train} \neq T_{test} > \max(T_{train})$ **then**
    $T_{test} = T_{train}$ # labeling unlabeled objects
for `each testing sample` $T_{test} = (x', y')$ **do**
    `Compute` $d(x', x)$, `the distance between` T_{test} (Eq.8)
    `and each sample,` $(x, y) \epsilon T_{train}$
    `Select` $T_{trainT_{test}} \subseteq T_{train}$,
    `the set of k closest training samples to` T_{test}
Intermediate: (*key*, *value*)
Reduce Class
Input: (*key* ← `name of the mapped data;` *value* ← `list of all map data`)
Output: `Key of the mapped data into row and column`
Reducer (*key*, *value*) # Distance reducer
while values.hasNext() **do**
data = getnext(value)
Output: (*key*, *value*) # return the output

If the element of testing set is out of range then it is labeled as new class and the algorithm re-trains the training set.

Methodology & Application

M-HM approach

- M-HM: hybrid approach based on ARIMA and KNN algorithm.
  - ARIMA → linear component
  - KNN → nonlinear component
- The seasonal and periodic components are easily handled by the ARIMA model.
- The nonlinear components are handled by KNN model.

25: **Map Class**
26: **Input:** (*key* ← `name of the input;` *value* ← `value of the input)`
27: **Output:** (*key*, *value*)
28: **Map** (*key*, *value*) # Prediction mapper
29: **if** *neighbor*(*j*, *i*) == *k* **then**
30: $pred(j, i) \leftarrow T_{test}(k)$ (Eq.9)
31: **goto** *loop*.
32: **close**;
33: **Intermediate:** (′*key*′, *value*) # add multiple values such as m_t, p_t, r_t and s_t
34: **Reduce Class**
35: **Input:** (*key* ← `name of the mapped data;` *value* ← `list of all map data)`
36: **Output:** `Key of the mapped data into row and column`
37: **Reducer** (*key*, *value*) # Prediction reducer
38: **while** values.hasNext() **do**
39: data = getnext(value)
40: **Output:** (*key*, *value*) # return the final output

Conclusion

Conclusion

Performance measure

Table 1
Performance measures.

Method	M-ARIMA				M-KNN				M-HM			
Parameter	MAD	MSE	RMSE	MAPE	MAD	MSE	RMSE	MAPE	MAD	MSE	RMSE	MAPE
Station-1 Data	0.8376	1.0864	1.0423	2.6097	0.01913	0.0010	0.0329	0.0001	0.0123	0.0005	0.0239	0.0364
Regular Component	0.0743	0.0096	0.0980	10.1247	0.0088	0.0009	0.0300	0.0234	0.0066	0.0007	0.0268	0.8001
Stochastic Component	0.8955	1.3853	1.1770	2.9767	0.0326	0.0243	0.1566	0.0001	0.0186	0.0099	0.0995	0.0569
Station-2 Data	1.0633	1.8610	1.3641	3.1503	0.0224	0.0011	0.0343	0.0001	0.0151	0.0005	0.2425	0.0442
Regular Component	0.0867	0.0137	0.1173	4.8002	0.0064	0.0001	0.0105	0.0080	0.0039	0.0001	0.0071	−0.2765
Stochastic Component	1.0700	1.9830	1.4082	3.4336	0.0131	0.0015	0.0390	0.0000	0.0089	0.0011	0.0336	0.0296
Station-3 Data	0.9027	1.3086	1.1439	2.7078	0.0234	0.0010	0.0322	0.0000	0.0153	0.0005	0.0227	0.0441
Regular Component	0.0802	0.0112	0.1060	9.8425	0.0069	0.0001	0.0120	0.0434	0.0036	0.0000	0.0072	−0.2307
Stochastic Component	0.9615	1.6187	1.2523	3.1132	0.0099	0.0002	0.0166	−0.0002	0.0055	0.0001	0.0116	0.0176

So, M-HM method in this study are able to give better predictions on temperature.

Conclusion

Performance measure

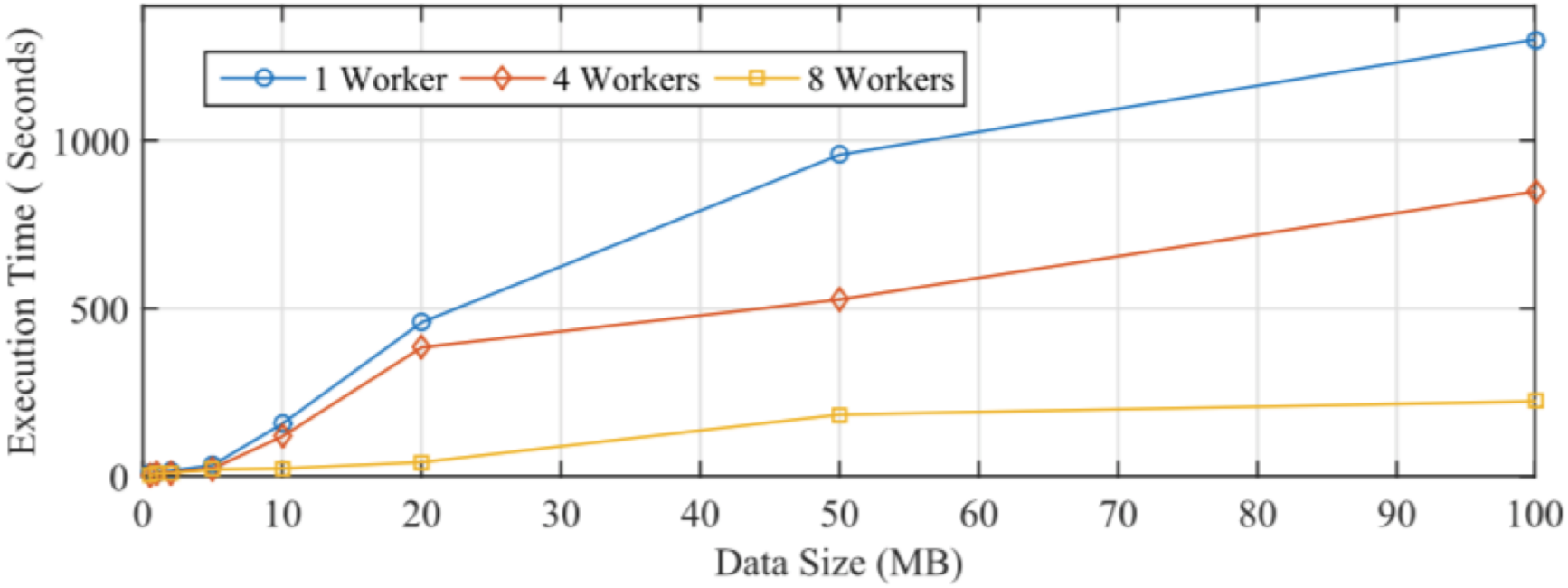


Fig. 10. Performance of decomposed approach over 1, 4, and 8 workers.

So, the performance of the system with parallel execution is approximately 90% greater than the single-node system.

Multi-step forecasting for big data time series based on ensemble learning

Knowledge-Based Systems

https://www.sciencedirect.com/science/article/pii/S0950705118304957

Outline

- **Introduction**
- **Methodology**
- **Application**

Introduction

Introduction

Background

- Big data: increasing generation and storage of massive data in recent years.
- Big data processing framework: MapReduce; Hadoop; Spark.
- Specific modules for mining big data: MLlib.
- A set of **promising** time series forecasting methods in a big data environment was studied.

Introduction

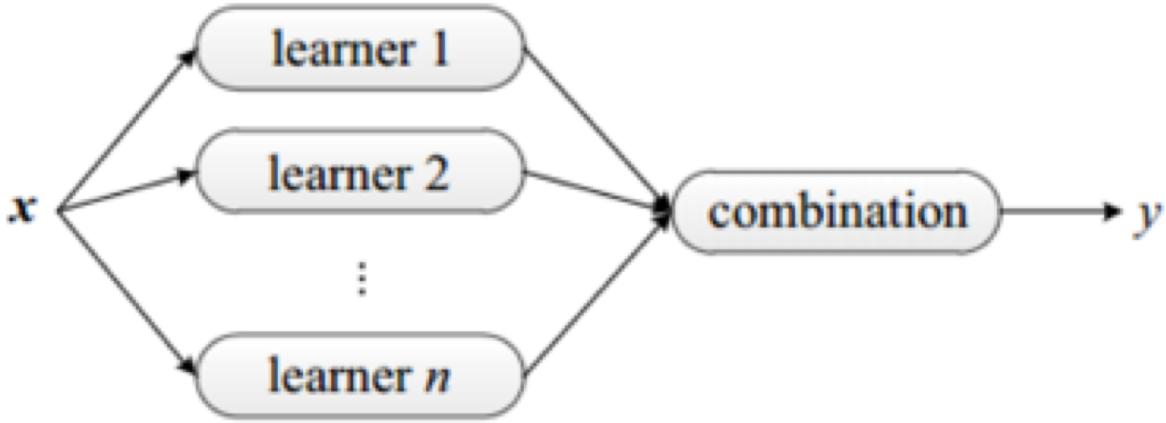
This paper...

- Combining decision trees, gradient boosted trees and random forest into ensembles.
- The ensemble approach assigns different weights to every method by using a weighted square least method.
- Two different strategies for training the ensemble models: static and dynamic.
- The handling of two limitations in Spark:
  - One step ahead prediction.
  - Not designed to guarantee the order of data.

Methodology

Methodology

Ensemble learning



Methodology

- Decision tree (DTs)
  - Classification and regression tasks.
  - Be able to model nonlinear relations.
  - Built through a recursive binary partition (with the highest information gain)
- Gradient boosted trees (GBT)
  - An ensemble of DTs (iteratively).
  - The errors made by the first tree are taken into account when adding the second tree and so on.
- Random forests (RF)
  - An ensemble of DTs (parallel).
- MLlib supports both GBT and RF.

Methodology

Multi-step forecasting

- The forecasting problem:

$$[x_{t+1}, x_{t+2}, \dots, x_{t+h}] = f(x_t, x_{t-1}, \dots, x_{t-(w-1)})$$

- The existing regression techniques in MLlib do not support this multi-step forecasting.
- Then, this paper splits the problem into h sub-problems:

$$x_{t+1} = f_1(x_t, x_{t-1}, \dots, x_{t-(w-1)})$$
$$x_{t+2} = f_2(x_t, x_{t-1}, \dots, x_{t-(w-1)})$$
$$\dots$$
$$x_{t+h} = f_h(x_t, x_{t-1}, \dots, x_{t-(w-1)})$$

  - The existing possible relations between the h consecutive values are not taken into consideration.

Methodology

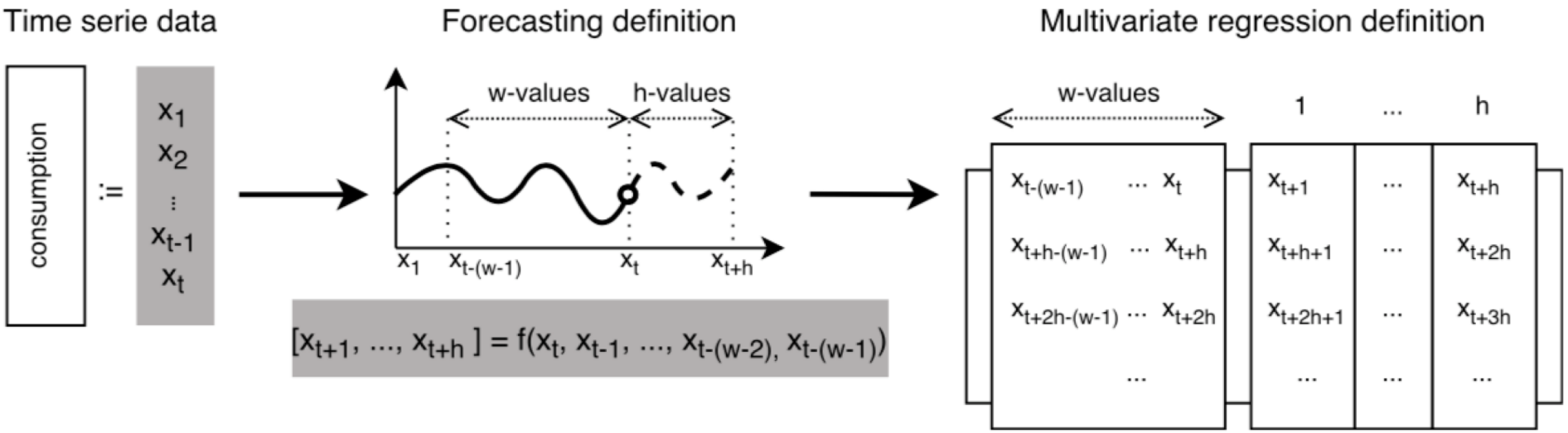


Fig. 3. Illustration of the multivariate problem.

Methodology

Proposed ensemble model – static

- A weighted least squares method is applied to obtain the weights for each algorithm.
  - Minimize the squared error between the predictions of the *K* algorithms and the actual values for the *N* instances of the validation set.
- For each *j*th value of the prediction horizon:

$$\widehat{P^j}\,\alpha^j = b^j$$

  - $\widehat{P^j}$: N×K matrix containing the prediction for the *j*th value of the prediction horizon for the validation set for each algorithm.
  - α^j: a vector of *K* elements corresponding to the weights.
  - b^j: a vector composed of the N actual values.

Methodology

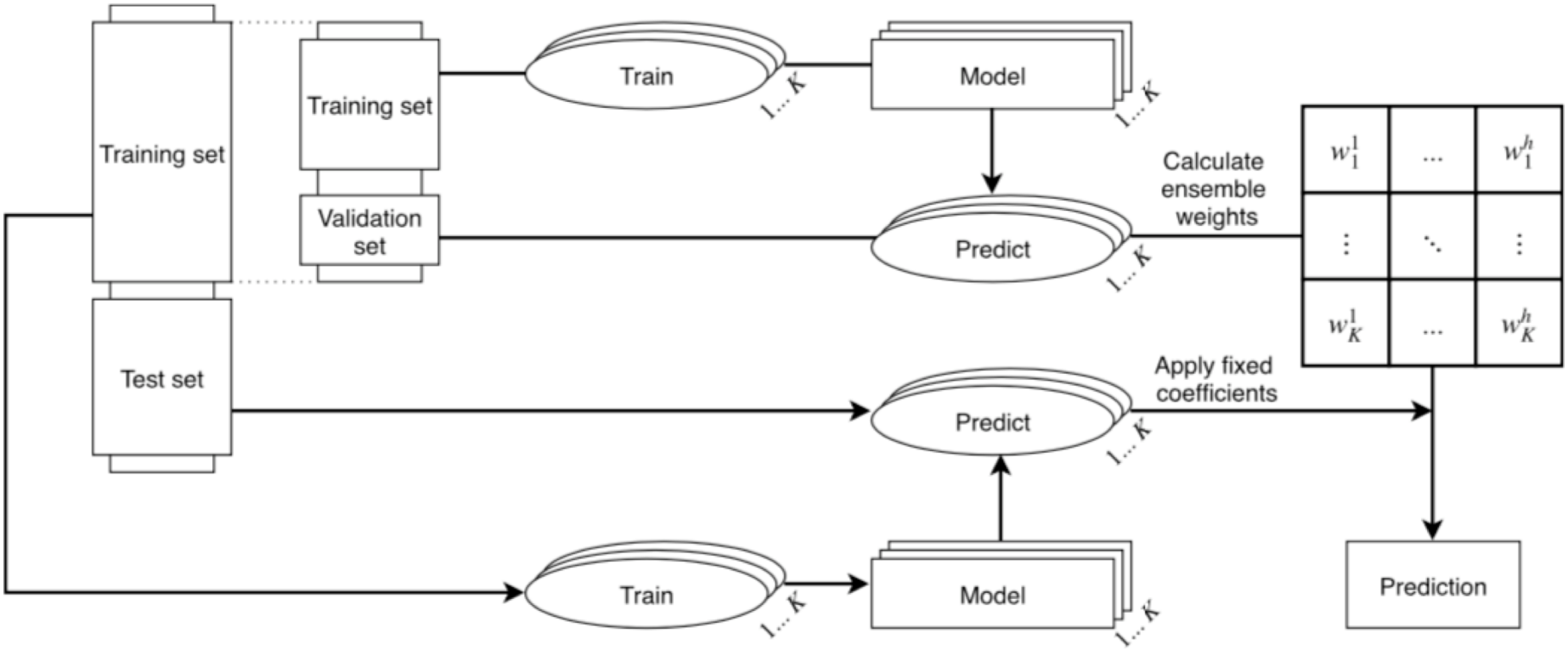


Fig. 1. Workflow of the proposed static ensemble.

Methodology

Proposed ensemble model – dynamic

- The test set *TS* can be divided into subsets as follows:

$$TS = \bigcup_{t=1}^{R} TS^t$$

- For each *j*th value of the prediction horizon, **weights** are found by solving:

$$\widehat{P^j}(t)\alpha^j(t) = b^j(t) \quad t = 1, \dots, R$$

- Training set (*TRS*) are updated as follows:

$$TRS \leftarrow TRS^t \cup \left(\bigcup_{l=1}^{t-1} TS^l \right) \quad t = 1, \dots, R$$

Methodology

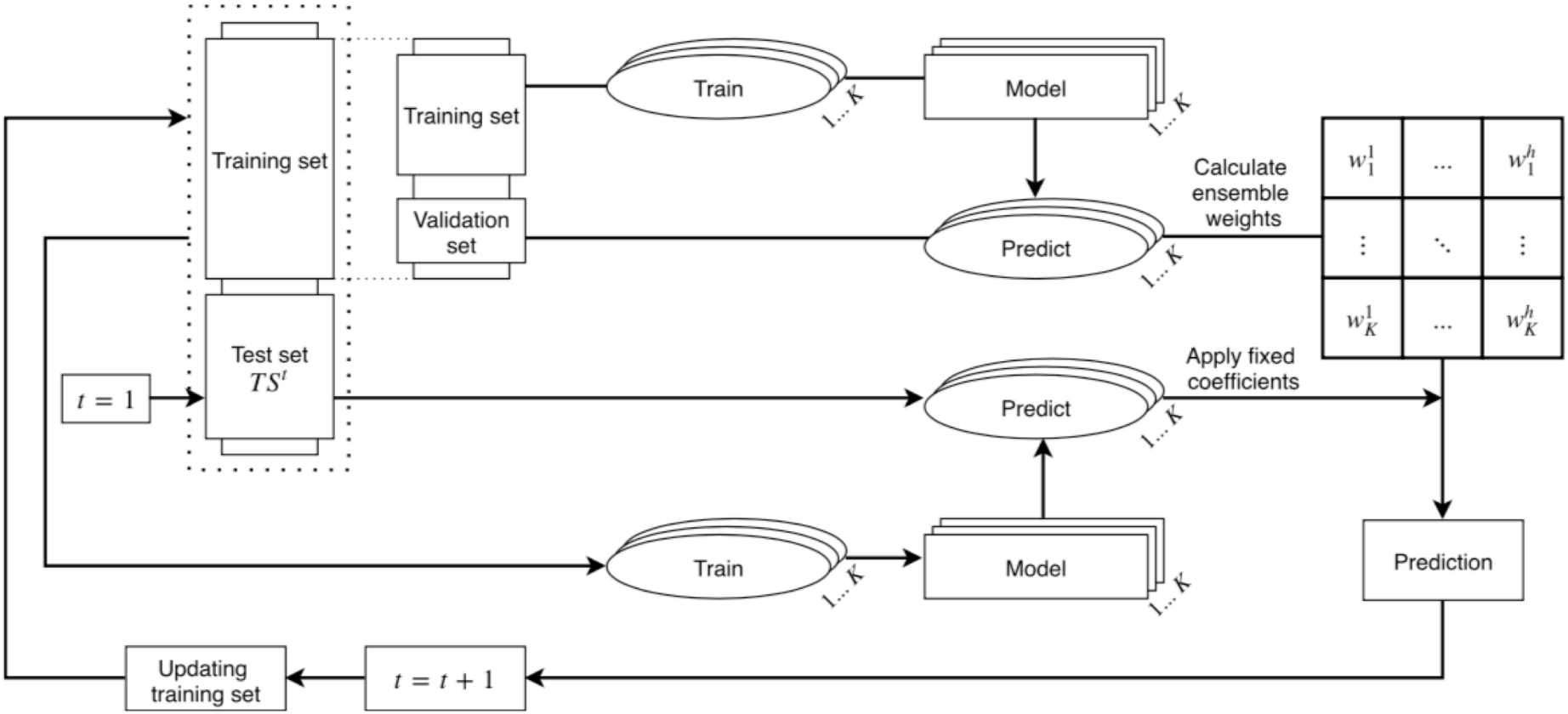


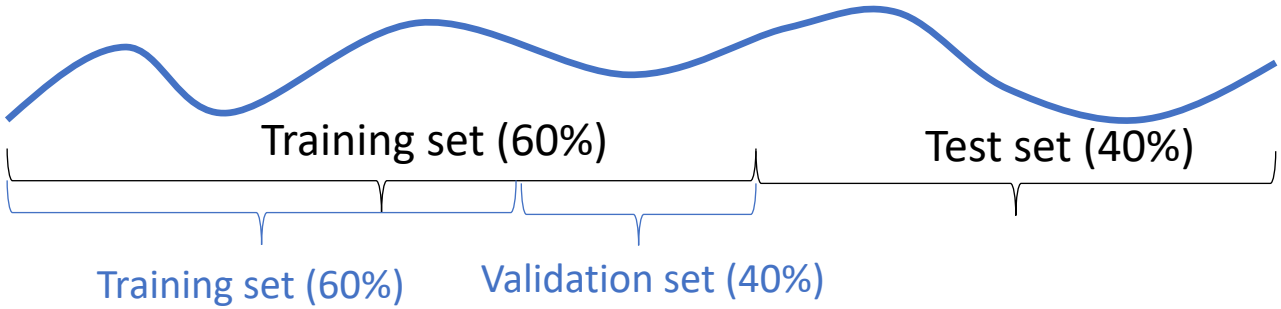
Fig. 2. Workflow of the proposed dynamic ensemble.

Application

Application

Electricity consumption data

- The total electrical energy consumption in Spain, from January 1st 2007 at midnight to June 21st 2016 at 11:40 pm.
- A time series of nine and a half years with a high sampling frequency, namely 10 min intervals, including 49,832 measurements in total.
- *h* is set to 24 values and window *w* is set to 144 past values.
- Static ensemble:



- Dynamic ensemble: updated every 13,104 predicted values.

Application

The mean relative error (MRE)

$$\mathrm{MRE} = \frac{1}{n}\sum_{i=1}^{n}\frac{|p_i - a_i|}{a_i}$$

Application

The size of historical window

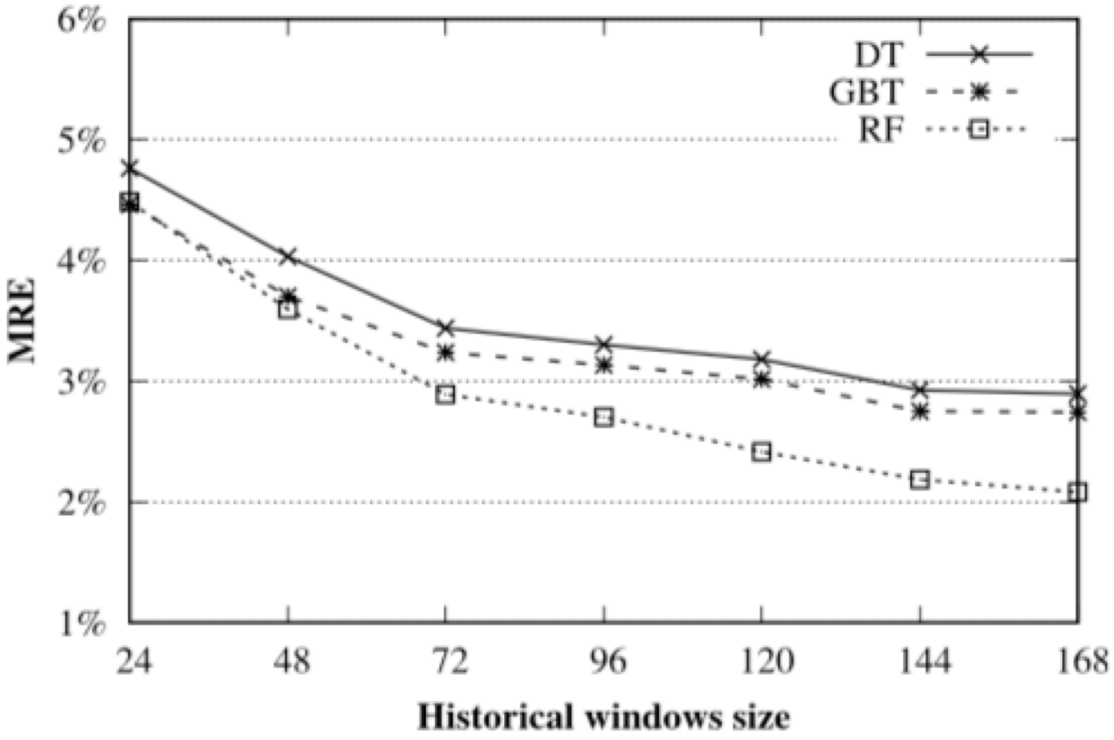


Fig. 4. MRE evolution for different historical window sizes.

- Increasing the window size from 144 to 168 does not lead to a significant improvement for DT and GBT. So *w*=144 is selected.

Application

Performance

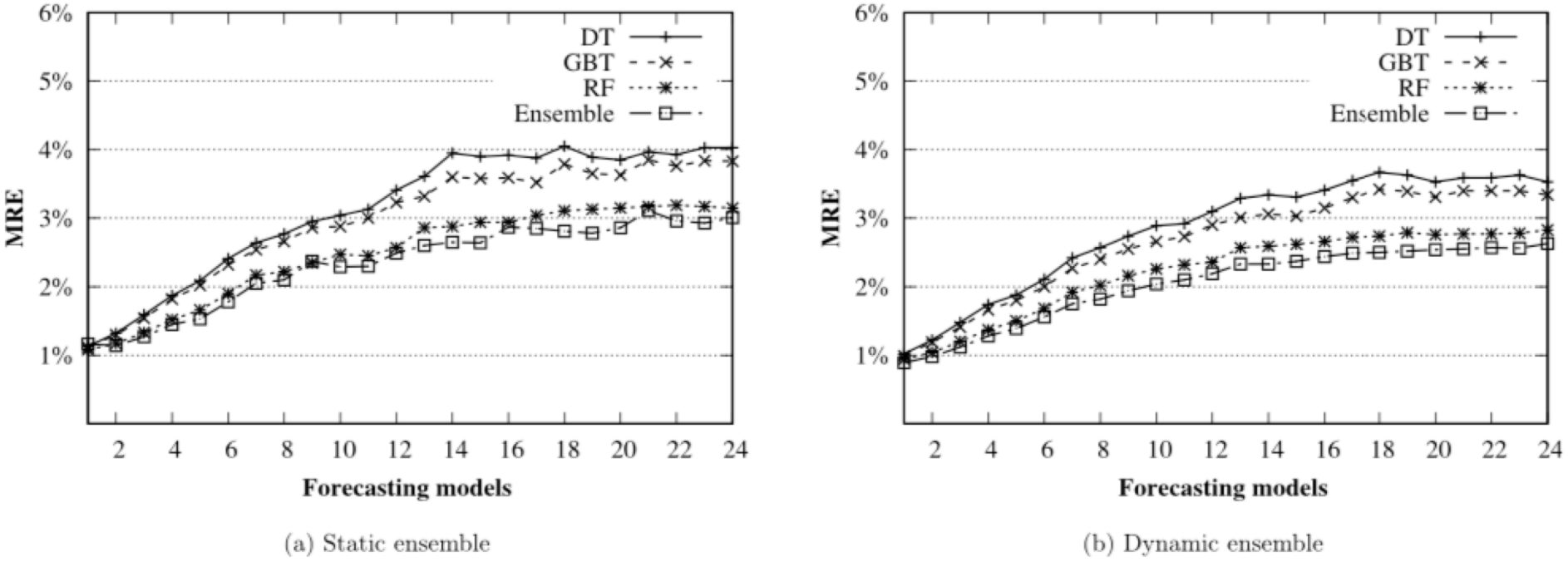


Fig. 5. MRE for each model, for each time point of the prediction horizon.

- Initially (1–2 h) all methods perform similarly.
- For hours 3–7, RF and the ensemble show similar performance and start outperforming the other methods.
- The ensemble method outperforms RF.

Application

Table 3
MRE (mean and variance) on the test set, for the worst and best predicted days.

Model	Algorithm	Worst (%)	Mean (%)	Variance (%)	Best (%)
Static	DT	10.2102	3.1400	0.014	1.2401
	GBT	10.1950	2.9680	0.012	1.2074
	RF	8.8475	2.4838	0.011	0.7621
	Ensemble	**9.3207**	**2.3320**	**0.009**	**0.8230**
Dynamic	DT	9.5022	2.8395	0.015	1.1500
	GBT	9.1633	2.6569	0.014	1.0782
	RF	8.5162	2.2243	0.012	0.6530
	Ensemble	**8.6016**	**2.0362**	**0.010**	**0.7189**

- The most accurate prediction model is the dynamic ensemble.

Application

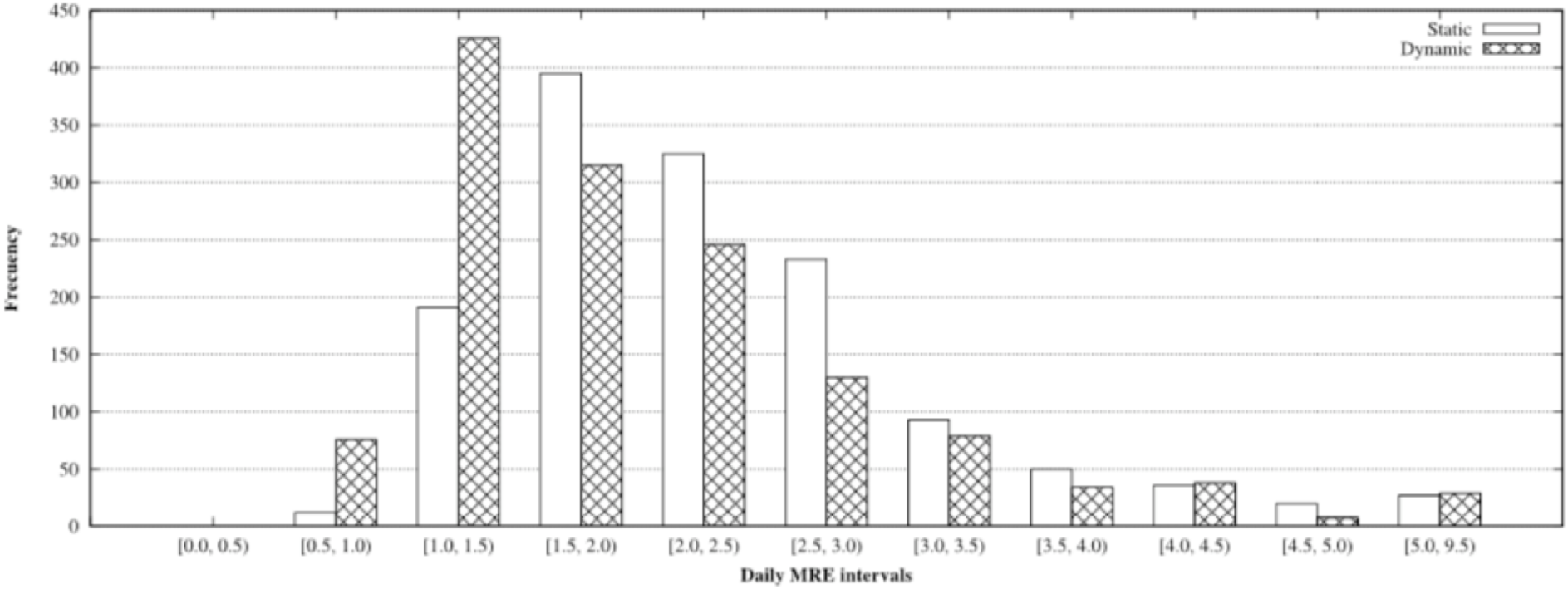


Fig. 6. Histogram of daily errors for static and dynamic ensemble models.

Least Squares Approximation for a Distributed System

Xuening Zhu, Feng Li, Hansheng Wang

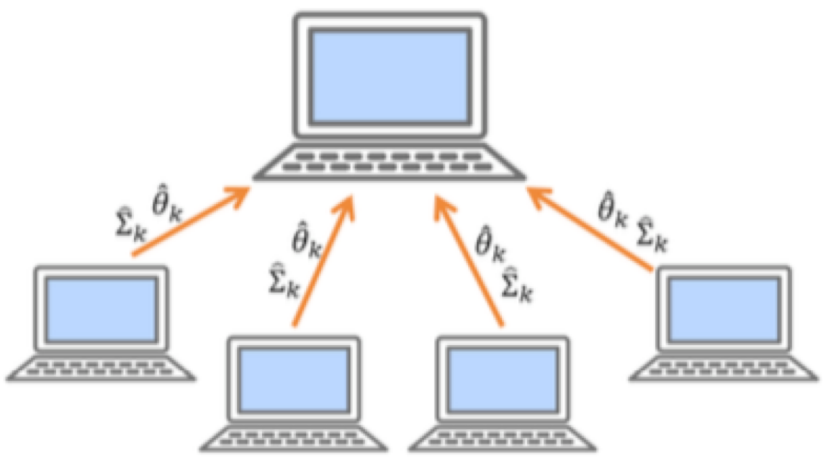
https://arxiv.org/pdf/1908.04904.pdf

Methodology

- Distributed least squares approximation (DLSA).
- A twice-differentiable loss function (Negative log-likelihood function).
- Decomposing the global loss function using Taylor's expansion techniques.
- This leads to a weighted least squares estimator (WLSE).
- An analytical form estimator that can be easily computed on the distributed system:

$$\tilde{\theta} = \underset{\theta}{\operatorname{argmin}}\, \tilde{\mathcal{L}}(\theta) = \left(\sum_k \alpha_k \widehat{\Sigma_k}^{-1}\right)^{-1}\left(\sum_k \alpha_k \widehat{\Sigma_k}^{-1}\, \widehat{\theta_k}\right)$$

Methodology



Step 1:

- The local workers conduct estimation based on local data;
- Then they send local estimators $\hat{\theta}_k$ and asymptotic covariance $\hat{\Sigma}_k$ to the master.

Step 2:

- The master constructs a weighted least squares type objective function and conduct estimation.

Figure 1: Illustration of the DLSA method.

Methodology

- DLSA estimator is as **statistically optimal** as the global estimator.
- Computationally efficient.
- Also develop the corresponding shrinkage estimation and prove the DBIC measure to be selection consistent.

- For time series forecasting:
  - DLSA is designed for the independent data, time series analysis need to take account the dependency structure.