Transactions on Computational and Scientific Methods | Vo. 4, No. 08, 2024
ISSN: 2998-8780
https://pspress.org/index.php/tcsm
Pinnacle Science Press

AI-Enhanced Distributed Time Series Modeling: Incremental Learning for Evolving Streaming Data

Yuxiao Wang
University of Pennsylvania, Philadelphia, USA
yx99.wang@gmail.com

Abstract: This study proposes an incremental learning framework for time series modeling in distributed systems to address the challenges of dynamic modeling and continuous learning in real-time streaming environments. The framework integrates a distributed parallel architecture with an incremental optimization mechanism to achieve online feature extraction and adaptive parameter updates for non-stationary streaming data. The system consists of four layers: data stream sampling, distributed time series modeling, incremental updating, and global coordination. It maintains model consistency and convergence stability under multi-node asynchronous communication conditions. By introducing drift detection and weight constraint mechanisms at the node level, the model dynamically adjusts the learning rate according to data distribution changes, effectively mitigating concept drift and forgetting problems. To verify the effectiveness of the proposed algorithm, sensitivity experiments were conducted to analyze the impact of learning rate, optimizer, weight decay coefficient, and communication delay on model performance. The results show that the proposed distributed incremental learning method achieves low prediction error and high robustness under various complex streaming data conditions, maintaining stable convergence as data continuously evolves. This research provides a scalable, high-precision, and low-latency solution for real-time streaming data modeling and holds important significance for dynamic optimization and continuous learning in distributed intelligent systems.

Keywords: Distributed systems; streaming data modeling; incremental learning; concept drift

1. Introduction

In today's data-driven era, the explosive growth of real-time streaming data has made traditional static batch processing systems increasingly inadequate for dynamic intelligent computing scenarios. With the rapid advancement of the Internet of Things, cloud computing, edge computing, and 5G technologies, massive heterogeneous data are continuously generated and transmitted in streaming form. These data exhibit characteristics such as high frequency, high dimensionality, and high noise[1]. Traditional centralized processing architectures face limitations such as transmission latency, computational bottlenecks, and storage pressure when dealing with such continuous data streams, making it difficult to achieve fast response and adaptive learning[2]. Therefore, building distributed systems for real-time streaming data has become a core foundation for supporting modern intelligent applications. Distributed computing frameworks improve system throughput and fault tolerance through multi-node parallel processing and task decomposition mechanisms, providing scalable computational support for dynamic data modeling. This architecture enables distributed data acquisition, transmission, and storage, while also providing an efficient execution environment for time series models and learning algorithms.

In real-time streaming data scenarios, time series modeling plays a crucial role in capturing dynamic patterns, forecasting trends, and detecting anomalies. Unlike static data, streaming data is characterized by continuous arrival, concept drift, and distributional shifts, which require models to possess online learning and rapid adaptability[3]. Although traditional deep time series models perform well on fixed datasets, they often suffer from catastrophic forgetting and model degradation when exposed to continuously updated data, making it difficult to maintain adaptability. Incorporating incremental learning mechanisms into the time series modeling process has become an effective way to address this issue. Incremental learning allows the model to update parameters and feature representations dynamically without retraining from scratch, enabling it to adapt to distribution changes, reduce computational costs, and maintain stable performance. This mechanism aligns naturally with distributed systems, providing efficient, flexible, and scalable intelligent learning capabilities for real-time applications[4].

At the same time, distributed systems also introduce new challenges under multi-node parallel processing environments, such as data consistency, model synchronization delay, and communication overhead[5]. In the context of continuously updating streaming data, maintaining system stability while achieving incremental model optimization becomes a critical concern. Incremental learning strategies designed for single-node systems cannot be directly applied to distributed settings, as different nodes may receive data from different time periods or sources, leading to asynchronous updates and conflicts. Therefore, developing incremental learning algorithms tailored to distributed architectures requires a balance between local updates and global coordination[6]. Through distributed parameter synchronization, asynchronous update strategies, and adaptive local optimization, it is possible to achieve low-latency online learning while ensuring global convergence, allowing the system to maintain robust time series prediction and self-evolution capabilities under high-concurrency data streams.

From an application perspective, distributed incremental learning for real-time streaming time series modeling holds great practical value. In domains such as financial risk control for detecting abnormal transactions, industrial IoT for equipment monitoring, and smart city management for traffic flow forecasting and energy scheduling, the dynamic nature of data requires models capable of online learning and rapid adaptation[7]. Distributed incremental learning models can maintain consistent predictive performance as data evolves, responding to environmental changes and sudden events in real time. Moreover, these algorithms reduce dependence on central nodes, avoiding privacy risks and bandwidth burdens associated with data transmission, and enabling fast learning and local decision-making at the edge. This capability is crucial for building autonomous, scalable, and low-latency intelligent distributed systems, driving the paradigm shift from "centralized training-offline inference" to "continuous learning-online decision-making".

In summary, research on incremental learning algorithms for time series modeling in distributed systems with real-time streaming data not only promotes the theoretical integration of streaming intelligence and distributed learning but also provides new directions for practical system optimization. By deeply integrating distributed computing architectures with incremental learning mechanisms, such systems can retain adaptability and learning ability even under uncertain, dynamic, and non-stationary environments. This research direction contributes to the development of intelligent infrastructures with autonomous learning and decision-making capabilities, enhancing the real-time performance and accuracy of streaming data analysis. It lays a solid foundation for building efficient, secure, and self-evolving distributed intelligent systems to support intelligent transformation in areas such as energy scheduling, financial risk control, and smart manufacturing.

2. Proposed Framework

2.1 General Introduction

This study proposes an incremental learning algorithm framework for distributed system time-series models oriented towards real-time streaming data. This framework aims to achieve dynamic feature modeling and adaptive parameter updating of continuously arriving data streams in a multi-node distributed computing environment. The overall system structure consists of four main modules: a data stream sampling layer, a distributed time-series modeling layer, an incremental update layer, and a global coordination layer. The data stream sampling layer receives data from heterogeneous source nodes and performs time windowing processing to form local sequences suitable for training. The time-series modeling layer performs feature encoding and sequence dependency modeling in parallel based on multi-node computing resources. The incremental update layer implements real-time correction of model weights to ensure a balance between new and old knowledge. The global coordination layer ensures the stable convergence of the system in large-scale streaming environments through parameter aggregation and consistency maintenance. The overall optimization objective of the model can be formalized as:

$$\min_{\theta} E_{(xt,yt)\sim D(t)} L(f_{\theta}(x_t), y_t)$$

Where $D(t)$ represents the data distribution at time t, θ represents the model parameters at time t, and L represents the loss function. Unlike traditional static training, this framework requires that the model parameters be continuously optimized over time in a distributed environment, and that synchronous or quasi-synchronous updates be achieved between nodes. Its overall model architecture is shown in Figure 1.

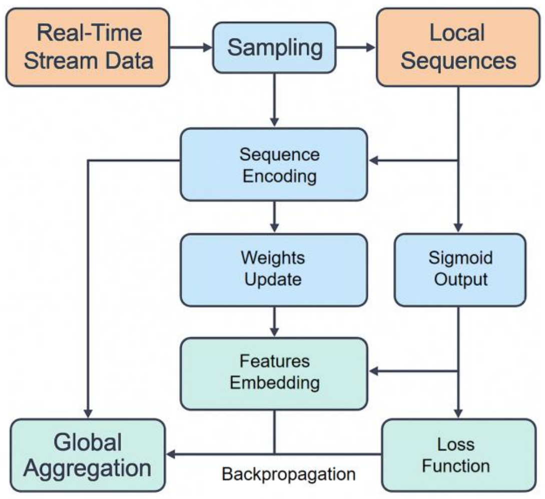

Figure 1. Overall model architecture

2.2 Distributed time series modeling mechanism

In a distributed environment, time series modeling employs a parallel multi-node architecture to distribute computational load. Assuming the system has $x_t^{(i)}$ computing nodes, each receiving a subset t of data streams, the time series modeling function for node i can be expressed as:

$$h_t^{(i)} = f_{enc}(x_t^{(i)}, h_{t-1}^{(i)}; \theta)$$

Here, f_{enc} is a time-dependent encoder, which can be in the form of a recurrent neural network or a state-space model. Each node obtains a state representation $h_{t-1}^{(i)}$ through local modeling, and then performs local loss minimization within a time window W_t.

$$L_t^{(i)} = \frac{1}{|W_t|} \sum_{x_i \in W_t} \|f_{\theta}(x_t) - y_t\|^2$$

Nodes perform feature aggregation and gradient synchronization through an asynchronous communication mechanism, defining a global gradient aggregation function:

$$g_t = \frac{1}{N}\sum_{I=1}^{N} \nabla \theta_i L_t^{(i)}$$

This ensures local autonomy while maintaining consistency in the global optimization direction. This distributed time-series modeling mechanism effectively reduces latency and improves the system's robustness under non-stationary data flows.

2.3 Incremental Updates and Adaptive Concept Drift

To address the dynamic changes and concept drift of streaming data, this study introduces an incremental learning mechanism based on time-series modeling. The model parameters are updated according to the following recursive form:

$$\theta_{t+1} = \theta_t - \eta_t \bullet \nabla \theta_t L_t + \lambda \bullet \Delta t$$

Where η_t is the time-adaptive learning rate, λ is the regularization balancing factor, and Δt represents the compensation term based on drift detection. When a change in data distribution is detected, the system uses the drift discrimination function:

$$Drift(t) = l[KL(D(t)//D(t-\tau)) > \epsilon]$$

This triggers local adjustments to the model structure or parameters to achieve adaptive updates. Here, KL represents the Kullback-Leibler divergence, used to measure the degree of data distribution drift. When concept drift is detected, the system automatically updates the weight decay coefficient and feature importance weights, enabling rapid response to new patterns and gradual forgetting of old knowledge, thereby maintaining the long-term stability of the model.

2.4 Global coordination and parameter consistency maintenance

In the context of multi-node incremental learning, ensuring the consistency and efficient convergence of the global model is a core issue. This study employs an improved distributed parameter aggregation mechanism, using a global control variable z_t to coordinate the synchronization process of parameters across nodes.

$$z_t = \alpha \bullet \frac{1}{N}\sum_{i=1}^{N} \theta_t^{(i)} + (1-\alpha) \bullet z_{t-1}$$

Where α is the synchronization coefficient, used to balance local independence and global consistency. Subsequently, each node updates its own parameters through a global feedback mechanism:

$$\theta_{t+1}^{(i)} = \theta_t^{(i)} + \gamma(z_t - \theta_t^{(i)})$$

γ represents the coordination step size, used to control the convergence speed of parameters towards the global equilibrium. This strategy achieves adaptive adjustment of parameter synchronization and model consistency while maintaining system parallelism. By establishing a mathematical synergy between asynchronous communication and incremental learning, this method effectively reduces the communication load and synchronization cost of distributed systems, providing a stable, scalable, and continuously optimizing intelligent learning mechanism for time-series modeling of real-time streaming data.

3. Experimental Analysis

3.1 Dataset

This study uses the Yahoo Streaming Benchmark Dataset (YSB) as the experimental foundation. The dataset consists of high-frequency real-time streaming logs mainly derived from online advertising clickstreams and system monitoring data. It reflects the characteristics of distributed systems with high concurrency and low latency. The purpose of the YSB dataset is to evaluate the overall performance of streaming systems in terms of throughput, latency, and accuracy. Its core feature lies in the continuous generation of data with strong temporal dependencies. Each record includes a timestamp, event type, advertisement identifier, user behavior, and contextual attributes. On average, it can generate hundreds of thousands of streaming events per second, making it suitable for incremental learning and time series modeling tasks.

The structure of this dataset represents a typical distributed real-time data scenario. Data are continuously ingested into the system through message middleware such as Kafka and are partitioned into independent shards for processing in a multi-node environment. Each data stream shard corresponds to different user groups or geographical regions, providing naturally heterogeneous and multi-source inputs for distributed incremental learning algorithms. The dynamic and bursty nature of the data requires models to adapt quickly to distributional shifts. When node updates occur asynchronously, the dataset further highlights the algorithm's advantages in stability and consistency. These characteristics make YSB a highly suitable environment for validating the proposed distributed incremental learning framework.

In addition, the YSB dataset offers significant advantages in scalability and reproducibility. Researchers can adjust data rates and time window lengths according to experimental needs to simulate real-time streaming scenarios of various scales. The dataset provides both event time and processing time semantics, allowing flexible adaptation between latency-sensitive and throughput-prioritized tasks. With its high dimensionality, multi-source inputs, and continuous updating characteristics, the YSB dataset not only enables effective evaluation of distributed time series modeling algorithms but also provides a solid data foundation for future research on system scalability and model adaptability.

3.2 Experimental Results

This paper first conducts a comparative experiment, and the experimental results are shown in Table 1.

Table 1: Comparative experimental results

Method	MSE	MAE	MAPE	RMSE
LSTM[8]	0.0248	0.1092	6.21%	0.1576
BILSTM[9]	0.0225	0.1028	5.79%	0.1501
MLP[10]	0.0281	0.1184	6.83%	0.1677
Transformer[11]	0.0209	0.0973	5.42%	0.1445
FedFormer[12]	0.0196	0.0948	5.10%	0.1403
Informer[13]	0.0189	0.0927	4.96%	0.1375
Ours	0.0163	0.0865	4.53%	0.1277

As shown in Table 1, under real-time streaming data scenarios, different time series modeling methods exhibit significant variations in performance across error metrics. Traditional LSTM and BILSTM models can capture temporal dependencies, but their stability and generalization are limited when handling high-concurrency and asynchronously updated streaming data in distributed systems. This limitation results in relatively high error metrics such as MSE and MAE. The MLP model, lacking sequential modeling capability, is highly sensitive to dynamic changes in data streams, which further increases prediction bias. These findings indicate that traditional static learning architectures are insufficient for continuously evolving data stream environments. Under conditions of concept drift and temporal instability, such models struggle to maintain consistent long-term predictive performance.

In contrast, Transformer-based models with attention mechanisms, including Transformer, FedFormer, and Informer, perform better in multi-scale feature extraction and long-range dependency modeling, reducing overall error metrics by approximately 10% to 20%. FedFormer maintains stability in feature aggregation and global modeling, while Informer improves both efficiency and accuracy in handling long sequences through sparse attention mechanisms. However, these methods still rely mainly on static training and cannot adaptively optimize for incoming real-time data. They also suffer from parameter synchronization delays and local inconsistencies in distributed environments. Therefore, there remains substantial room for improvement in their application to streaming data prediction tasks.

The distributed incremental learning algorithm proposed in this study achieves the best results across all metrics. The MSE decreases to 0.0163, and the RMSE is only 0.1277, demonstrating the model's strong adaptability and robustness under real-time data streams. The proposed method leverages multi-node collaborative optimization and incremental update mechanisms to achieve rapid response to concept drift and maintain global parameter consistency. As a result, the system maintains low error and high accuracy even under high-frequency input and asynchronous distribution conditions. These results validate the effectiveness of the proposed framework in enabling continuous learning and efficient prediction within distributed systems and highlight the essential role of incremental learning mechanisms in intelligent modeling of streaming data.

Furthermore, the impact of the learning rate on the experimental results is presented, and the results are shown in Table 2.

Table 2: Hyperparameter sensitivity experimental results

(Learning Rate)

Learning Rate	**MSE**	**MAE**	**MAPE**	**RMSE**
0.0004	0.0218	0.1013	5.48%	0.1476
0.0003	0.0192	0.0958	5.02%	0.1386
0.0002	0.0174	0.0902	4.71%	0.1320
0.0001	0.0163	0.0865	4.53%	0.1277

As shown in Table 2, different learning rates have a significant impact on model prediction performance in streaming data scenarios. When the learning rate is large (for example, 0.0004), the update step becomes too large, causing parameter oscillations near local extrema and preventing stable convergence. In this case, the MSE and MAE reach 0.0218 and 0.1013, respectively, representing the worst overall performance. This instability is particularly evident in real-time streaming environments because continuously arriving data amplifies the uncertainty of parameter updates. As a result, the model exhibits high volatility under concept drift or sudden distribution shifts, making it difficult to maintain prediction accuracy.

As the learning rate gradually decreases, the prediction error of the model significantly improves. When the learning rate is 0.0003 and 0.0002, the MSE and MAE decrease by approximately 10% to 20%, indicating that an appropriate learning rate enhances convergence stability and the ability to capture temporal dependencies. In distributed environments, learning rate control affects not only the training speed of individual nodes but also the efficiency and consistency of global synchronization. An excessively high update rate can cause parameter drift across nodes, while a learning rate that is too small may result in delayed global aggregation. Both scenarios can negatively affect the system's responsiveness under real-time data streams.

When the learning rate is set to 0.0001, the model achieves optimal performance with an MSE of 0.0163 and an RMSE of 0.1277. This indicates that the configuration achieves a balance between global convergence stability and high prediction accuracy. It demonstrates that in a distributed incremental learning framework, an appropriate learning rate not only reduces gradient fluctuations but also improves parameter aggregation consistency. The model can thus maintain high robustness when handling dynamic variations and asynchronous updates across multiple nodes. Overall, these results confirm the adaptive learning characteristics and the effectiveness of the distributed optimization mechanism in real-time streaming data processing. This paper also presents experimental results on the hyperparameter sensitivity of the optimizer, as shown in Table 3.

Table 3: Hyperparameter sensitivity experimental results

(Optimizer)

Optimizer	MSE	MAE	MAPE	RMSE
AdaGrad	0.0207	0.0975	5.39	0.1438
SGD	0.0221	0.1012	5.64	0.1487
Adam	0.0178	0.0906	4.79	0.1334
AdamW	0.0163	0.0865	4.53%	0.1277

As shown in Table 3, different optimization algorithms have a significant impact on model performance in distributed streaming data scenarios. Traditional Stochastic Gradient Descent (SGD) converges slowly in continuous data streams and is highly sensitive to learning rate changes. This often leads to oscillations and local overfitting during real-time updates, with MSE and MAE reaching 0.0221 and 0.1012, respectively, representing the worst performance. In contrast, AdaGrad adapts the step size by accumulating historical gradient information, allowing faster convergence in the early stages. However, because its learning rate continuously decays, it tends to suffer from a "learning rate saturation" problem during long-term streaming data training, resulting in insufficient updates and limited prediction accuracy. This trend indicates that traditional optimizers struggle to maintain continuous learning and dynamic adaptability in non-stationary streaming environments.

With improvements in optimization algorithms, Adam introduces first and second moment estimates during parameter updates, which significantly enhance the model's adaptability to streaming data. It performs better than SGD and AdaGrad across all metrics, especially in MAE and MAPE, which decrease by about 10% and 12%, respectively. This result shows that Adam has a stronger capability for adaptive gradient adjustment in non-independent and non-identically distributed data streams. However, Adam still faces convergence shift issues in distributed environments. When parameters are updated asynchronously across nodes, slight instability may occur. This problem can lead to inconsistent convergence paths in high-frequency streaming tasks, affecting the overall parameter aggregation performance.

The AdamW optimizer adopted in this study further incorporates a weight decay mechanism, which significantly improves long-term stability and generalization in distributed incremental learning. Its MSE is only 0.0163, and RMSE reaches 0.1277, representing the best overall performance. In dynamic streaming data environments, AdamW balances parameter update magnitude with weight regularization, preventing gradient explosion and weight drift. This enhances model robustness under multi-node asynchronous updates. These results confirm the necessity of introducing structured weight constraints in distributed time series modeling, ensuring that the model maintains high predictive accuracy and system stability during continuous learning and real-time response.

This paper further illustrates the impact of the weight decay coefficient on the experimental results, as shown in Figure 2.

As shown in Figure 2, the weight decay coefficient has a significant impact on model performance in streaming data scenarios. When the weight decay coefficient is small (for example, 0.0001), the model tends to overfit local patterns in the real-time data stream, leading to reduced generalization performance. Both MSE and MAE remain at high levels. This indicates that in environments with continuous data arrival and frequent concept drift, insufficient regularization causes parameters to respond too strongly to transient features, resulting in performance fluctuations when the data distribution changes. This effect is more pronounced in high-dimensional streaming data, where parameter updates rely heavily on recent samples, weakening the model's long-term stability.

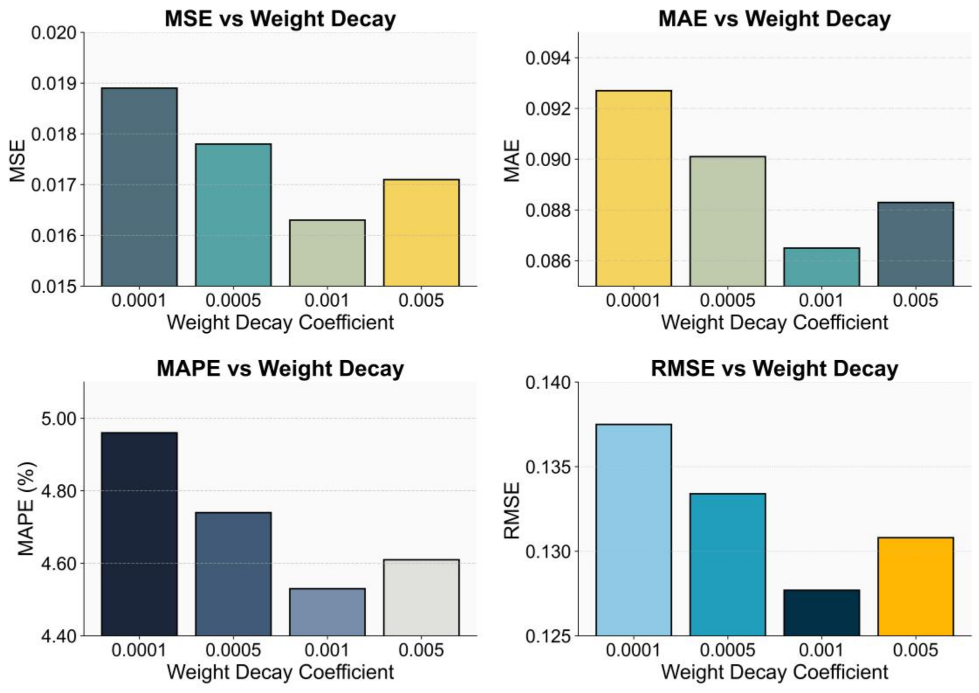

Figure 2. The impact of the weight decay coefficient on experimental results

As the weight decay coefficient increases to 0.0005, the model error decreases significantly, suggesting that moderate weight constraints can effectively suppress overfitting and enhance robustness to temporal variations. At this stage, parameter updates become smoother, and synchronization among distributed nodes becomes more stable, allowing for greater consistency during global aggregation. This trend demonstrates that in a distributed incremental learning framework, moderate parameter decay not only improves local node learning stability but also mitigates drift accumulation caused by asynchronous updates, thereby promoting overall system balance.

When the weight decay coefficient increases to 0.001, all evaluation metrics (MSE = 0.0163, MAE = 0.0865, MAPE = 4.53%, RMSE = 0.1277) reach their optimal values. This indicates that this configuration achieves the best balance between model complexity and generalization capability. Under this condition, the model can quickly respond to changes in real-time streaming data while avoiding performance oscillations caused by excessive adjustment. These results verify the stable convergence of the proposed distributed incremental learning algorithm under the weight regularization mechanism and demonstrate its strong adaptability to dynamic feature variations in non-stationary environments.

When the weight decay coefficient further increases to 0.005, the performance slightly declines. The main reason is that excessive regularization reduces the model's sensitivity to new features, limiting its learning ability during concept drift phases. Although the increase in error is minor, it can lead to slower short-term responses in real-time tasks. Therefore, it can be concluded that in this research scenario, the weight decay coefficient should be maintained around 0.001 to balance stability and adaptability. This finding provides a useful reference for adaptive hyperparameter tuning in distributed systems and further confirms that appropriate control of regularization strength is essential for achieving continuous optimization and robust prediction in streaming data environments.

This paper also presents the impact of communication delay on the experimental results, which are shown in Figure 3.

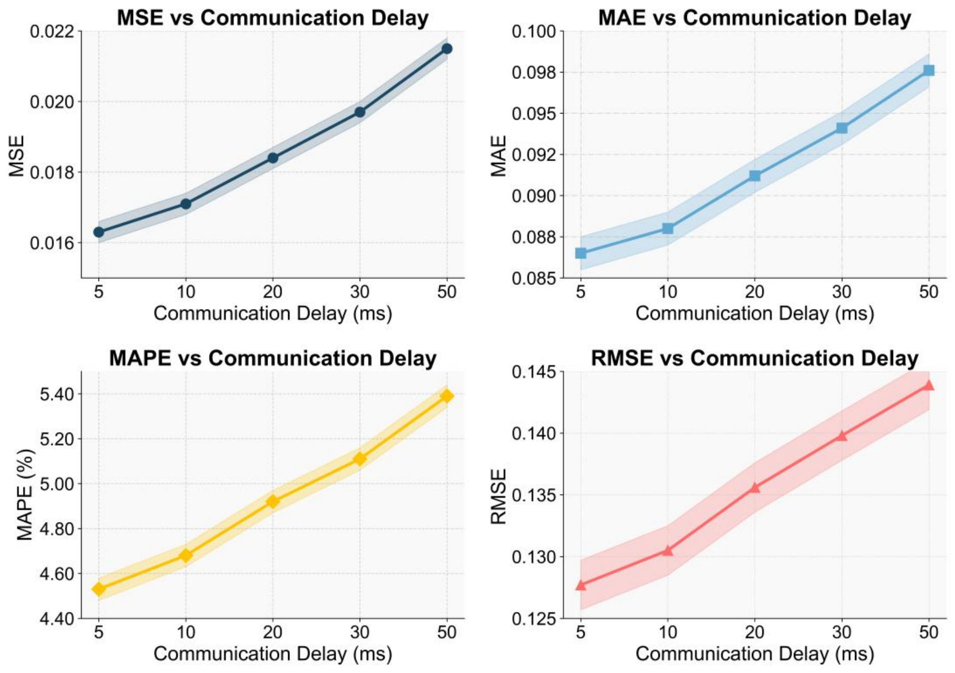

Figure 3. The impact of communication delay on experimental results

As shown in Figure 3, communication delay has a clear impact on model prediction performance in distributed real-time streaming environments. As the delay increases from 5 ms to 50 ms, all error metrics (MSE, MAE, MAPE, RMSE) show an upward trend, indicating that longer communication latency weakens the model's global synchronization ability. When the delay is low, each node can quickly complete parameter exchange and gradient aggregation, maintaining consistent learning states across nodes and achieving lower prediction errors. However, as the delay grows, the timeliness of parameter updates decreases, and differences among node models accumulate, slowing down the global convergence process and leading to continuous error growth.

Under low-delay conditions (such as 5 ms to 10 ms), the model still maintains strong time series modeling capability, with only slight variations in MSE and MAE. This indicates that the proposed incremental learning mechanism adapts well to mild communication instability. In this phase, local nodes can rapidly adjust parameters, while the global synchronization strategy ensures distributed consistency. This feature is particularly important for high-concurrency streaming data scenarios, as it allows the model to continue learning and optimizing dynamically under low-latency feedback between edge nodes and the central server, ensuring stable real-time responses.

When the communication delay increases to 30 ms or more, the rise in error metrics becomes more pronounced, especially for MAPE and RMSE, indicating higher sensitivity of prediction results to latency. The main reason is the delayed gradient feedback, which causes local nodes to update based on outdated information, thereby affecting the overall convergence direction. Since the proposed algorithm relies on asynchronous aggregation across multiple nodes in distributed settings, communication delays not only slow model synchronization but also amplify parameter drift among nodes, leading to dynamic instability in the learning process. This phenomenon reveals the performance bottleneck of distributed incremental learning in high-latency network environments.

Overall, the experimental results show that communication delay is one of the key factors affecting the performance of distributed incremental learning systems. Moderate delays do not significantly impair model performance, but excessive delays negatively affect global convergence and prediction stability. The proposed framework demonstrates a certain level of robustness within the delay range due to its parameter buffering and asynchronous update mechanisms, allowing the model to maintain low error under partial delay conditions. However, to achieve better streaming data modeling performance, future system designs should further optimize communication scheduling and gradient synchronization strategies at the architectural level to enhance timeliness and distributed coordination efficiency.

4. Conclusion

This paper proposes an incremental learning framework for time series modeling in distributed systems to address the challenges of real-time streaming data modeling and learning. The framework integrates distributed parallel computing with online learning mechanisms, enabling dynamic feature modeling and adaptive parameter updating as data continuously arrives. It maintains high prediction accuracy and robust performance in non-stationary environments. By introducing a multi-node asynchronous synchronization mechanism and a drift detection module, the model can respond quickly to distribution changes while preventing global parameter imbalance and catastrophic forgetting. This design highlights the intelligence and self-learning capabilities of distributed systems under high-concurrency data scenarios, providing a scalable, low-latency, and continuously optimized solution for real-time data processing.

The experimental results show that the proposed incremental learning mechanism maintains stable performance under various conditions of latency, noise, and dynamic data distributions. This validates the generality of the distributed incremental optimization framework for complex streaming tasks. Compared with traditional static training models, the proposed approach demonstrates stronger convergence and adaptability in long-term online learning tasks, offering a solid theoretical foundation for distributed intelligent decision-making systems. More importantly, this study breaks away from the traditional "offline training and online inference" paradigm. It enables continuous learning even under limited computational and communication resources, supporting real-time decision-making in areas such as edge intelligence, industrial IoT, financial risk control, and intelligent transportation.

From a system perspective, the contribution of this research lies not only in performance improvement but also in the optimization of the distributed learning architecture. By incorporating structured memory and regularized update strategies at the algorithmic level, the system maintains information consistency across nodes while achieving a dynamic balance between computation and communication efficiency. The proposed framework can be widely applied to real-time response and continuous learning scenarios, such as energy scheduling, network security, online recommendation, and anomaly detection. Its modular design also provides strong scalability for cross-platform migration and deployment in heterogeneous environments, demonstrating high practical and engineering value.

Future work can be extended in several directions, including communication optimization, enhanced model adaptability, and multimodal data fusion. One direction is to integrate adaptive communication compression and parameter quantization strategies into distributed incremental learning to reduce bandwidth consumption and improve system responsiveness. Another direction is to explore joint modeling methods for multi-source heterogeneous streaming data, enabling the system to handle more complex spatiotemporal dependencies and multimodal features. Furthermore, combining federated learning with self-supervised mechanisms can help build a more intelligent distributed learning ecosystem that unifies data privacy protection and collaborative intelligence. Overall, this study enriches the theoretical foundation of real-time streaming data learning and lays a solid basis for the evolution and application of distributed intelligent systems in highly dynamic environments.

References

[1] Melgar-García L, Gutiérrez-Avilés D, Rubio-Escudero C, et al. A novel distributed forecasting method based on information fusion and incremental learning for streaming time series[J]. Information Fusion, 2023, 95: 163-173.

[2] Wang J, Shi M, Zhang X, et al. Incremental data stream classification with adaptive multi-task multi-view learning[J]. Big Data Mining and Analytics, 2023, 7(1): 87-106.

[3] Hu L, Li W, Lu Y, et al. Scalable concept drift adaptation for stream data mining[J]. Complex & Intelligent Systems, 2024, 10(5): 6725-6743.

[4] Lora, A. T., Santos, J. M. R., Expósito, A. G., Ramos, J. L. M. and Santos, J. C. R., "Electricity market price forecasting based on weighted nearest neighbors techniques", IEEE Transactions on Power Systems, vol. 22, no. 3, pp. 1294-1301, 2007.

[5] Melgar-García L, Gutierrez-Aviles D, Rubio-Escudero C, et al. Identifying novelties and anomalies for incremental learning in streaming time series forecasting[J]. Engineering Applications of Artificial Intelligence, 2023, 123: 106326.

[6] Lin X, Chang L, Nie X, et al. Temporal attention for few-shot concept drift detection in streaming data[J]. Electronics, 2024, 13(11): 2183.

[7] Ziffer G, Bernardo A, Della Valle E, et al. Towards time-evolving analytics: Online learning for time-dependent evolving data streams[J]. Data Science, 2023, 6(1-2): 1-16.

[8] Sarmas, E., Strompolas, S., Marinakis, V., Santori, F., Bucarelli, M. A. and Doukas, H., "An incremental learning framework for photovoltaic production and load forecasting in energy microgrids", Electronics, vol. 11, no. 23, p. 3962, 2022.

[9] Wang, K., Qi, X. and Liu, H., "Photovoltaic power forecasting based LSTM-Convolutional Network", Energy, vol. 189, p. 116225, 2019.

[10]Yao, L. and Ge, Z., "Online updating soft sensor modeling and industrial application based on selectively integrated moving window approach", IEEE Transactions on Instrumentation and Measurement, vol. 66, no. 8, pp. 1985-1993, 2017.

[11]Nayak G H H, Alam M W, Avinash G, et al. Transformer-based deep learning architecture for time series forecasting[J]. Software Impacts, 2024, 22: 100716.

[12]Zhou T, Ma Z, Wen Q, et al. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting[C]//International conference on machine learning. PMLR, 2022: 27268-27286.

[13]Zhou H, Zhang S, Peng J, et al. Informer: Beyond efficient transformer for long sequence time-series forecasting[C]//Proceedings of the AAAI conference on artificial intelligence. 2021, 35(12): 11106-11115.