COMPRESSING TIME SERIES DATA

Xenophon Kitsios[1], Panagiotis Liakos[1], Katia Papakonstantinopoulou[1], Yannis Kotidis[1]

[1]Athens University of Economics and Business, Greece

Keywords: Time series, Compression, Internet-of-Things

Introduction

With the emergence of the Internet of Things (IoT) enormous amounts of streaming, timestamped datasets are being generated. Sensors from smart wearables, smart cities, autonomous cars, agricultural facilities etc., produce time-series data which needs to be collected centrally in order to be later analyzed. Time-series data is any data recording associated with a specific time stamp. Beyond IoT applications, time series are encountered in other domains, such as finance (stock prices, commodity prices, exchange rates), e-commerce (website traffic, sales volume, conversion rates), health care (electrocardiogram, blood pressure monitoring), social networks (user engagement, post frequency, sentiment analysis) and more.

Traditional databases can be used to store and query time-series data. However, they lack specific optimizations when processing consecutive data values that are often related. Essentially, each input data and its timestamp are processed separately, resulting in bottlenecks in data ingestion rates and storage requirements. As a result, Time Series Databases (TSDBs) have been introduced to facilitate storage and querying of large time series datasets. TSDBs support time-based queries and analytics, such as filtering, aggregation, and statistical analysis of time-series data. Moreover, TSDBs offer specialized data compression algorithms, which allow them to handle large and complex time-series datasets. Based on the independent website DB-Engines the top-5 ranked TSDBs are: 1) InfluxDB, 2) Kdb+, 3) Prometheus, 4) Graphite, and 5) TimescaleDB.

Due to the popularity of time-series data, many compression techniques have been proposed over the years. Depending on the application requirements, a lossless or a lossy compression technique can be used. The former reduces the size of data without loss of information, whereas the latter aims for much larger space savings while tolerating a bounded maximum error. Lossless time series compression techniques are used in applications where exact, detailed data values are needed. On the other hand, lossy techniques are more suitable for applications that are not particularly interested in individual exact values but rather seek to find and analyze the overall shape, trends, and characteristics of the data. Examples include seasonality detection, clustering, forecasting and similarity search. Traditional compression algorithms, i.e., LZ4, LZSS, ZStd etc., are not generally used for compressing time series data due to their slow running times that affect data ingestion rates and query performance. Specific compression algorithms oriented on time-series data include lossless techniques like Gorilla (Pelkonen et al., 2015), Chimp (Liakos, Papakonstantinopoulou and Kotidis, 2022) and Snappy (Google, 2011), or lossy algorithms such as Piecewise Aggregate Approximation (PAA) (Keogh et al., 2001) and Piecewise Linear Approximation (PLA) (Hakimi and Schmeichel, 1991).

Sim-Piece Algorithm

Piecewise Linear Approximation (PLA) is a fundamental data compression technique dating back to the 1960s, commonly used to approximate time-series data. PLA algorithms represent time-series measurements using a sequence of line segments, while keeping the approximation error within a predetermined acceptable threshold. Clearly, these algorithms are associated with a trade-off between space efficiency and precision loss, i.e., the space savings grow with the value of the error threshold.

The proposed Sim-Piece (Kitsios et al., 2023) is a novel approach for PLA that seeks to exploit similarities among the line segments produced. Unlike previous PLA techniques, Sim-Piece comes up with lossy compressed representations that provide impressive space savings, even in cases where the acceptable error threshold is very small.

When constructing a line segment, Sim-Piece fixes its starting point to a quantized value that is within a user input error of the original value. Then, we add subsequent data points to the segment by maintaining a pair of slopes, i.e., the extreme upper and lower slopes that satisfy the required approximation guarantees, until a point falls out of the area between them. As any of the lines between the two slopes can approximate the data points of the segment, we can find groups of segments with intersecting sets of candidate lines and represent them jointly, to reduce the overall space requirements.

Sim-Piece computes the optimal solution to this problem, coming up with the minimum possible number of groups to represent the PLA segments. As a result, our technique achieves impressive compression ratios that vastly improve the current state-of-the-art approaches.

Evaluation of Sim-Piece against state-of-the-art PLA methods

Through extensive experiments on various datasets from diverse sources, we have demonstrated that Sim-Piece manages to achieve outstanding results compared to existing compression algorithms for time series data. In a lossy compression algorithm, the decompressed data values are within a maximum error threshold from the original values. This error threshold is defined by the application and is used to balance the tradeoff between the size of the data representation and the required accuracy of the computations. As an example, Table 1 shows the compression ratio achieved by Sim-Piece and three state-of-the-art PLA techniques, i.e., Mixed (Luo et al, 2015), Swing (Elmeleegy et al., 2009) and Slide (Elmeleegy et al., 2009), on five real datasets (Dau Hoang Anh et al, 2018 and NEON, 2022). The maximum error threshold used was 5% of each dataset range of values. Clearly Sim-Piece stands out as the most space-efficient algorithm, offering compression ratios that are far superior to other leading techniques.

		PLA Compression Algorithm			
		Sim-Piece	**Mixed**	**Slide**	**Swing**
Dataset	**Cricket**	74.8	45.2	38.3	22.8
	FaceFour	20.9	15.6	11.9	10.2
	Lightning	115.4	83.8	67.9	36.6
	Wind Speed	40.8	27.2	24.3	9.8
	Wind Direction	15.0	7.6	6.7	4.4

Table 1: Compression Ratio of various PLA techniques.

Furthermore, we explored the quality of the approximation obtained by each method. We used two popular metrics to measure quality (i) Mean Absolute Error (MAE) and (ii) Root Mean Square Error (RMSE). Compared to the other techniques, Sim-Piece provides the second-best average MAE and RMSE values at a significantly higher compression ratio. Moreover, for the same space, Sim-Piece clearly outperforms all existing PLA techniques in terms of both MAE and RMSE.

Sim-Piece against Lossless Compression Techniques

When using lossless compression algorithms, the decompressed data are exactly identical as the original data. This strict requirement comes at a cost, as it limits opportunities for compressing the data further. As a result, lossy compression can achieve significantly higher compression ratios as some information is sacrificed, in a controllable manner. In some cases, sacrificing accuracy is indeed beneficial, as a form of data denoising.

In a TSDB, the reduced size of a lossy data representation reduces disk access times, as more relevant data can be kept in memory when processing a user query over voluminous datasets. The size of the data representation also plays a significant role in IoT applications, where remote sensory data needs to be transmitted to a central location. An IoT sensor that transmits its data over a wireless network by encoding them first with a lossy compression algorithm, will save network bandwidth and reduce energy transmission cost. Thus, lossy compression improves the scalability of IoT systems by reducing the amount of data that needs to be processed and stored (Deligiannakis, Kotidis and Roussopoulos, 2007; Deligiannakis and Kotidis, 2005). It can enable more IoT devices to be connected to a network without overloading the network or data storage resources. In addition, IoT devices that are powered by batteries can minimize energy consumption as they can keep their radios in a low-power state for longer intervals (Giatrakos et al., 2013). Additionally, lossy compression can also reduce the computational resources needed to process and analyze streaming time-series data in a decentralized system, which can further conserve energy.

In the field of lossless time series compression, Gorilla is a compression algorithm particularly suited for floating-point time series. Due to its effectiveness, Gorilla is the preferred algorithm used in most existing TSDBs. A more recent Chimp lossless compression algorithm has been shown to outperform Gorilla and be very competitive in terms of space requirements with lossy PLA approaches. Our Sim-Piece algorithm is motivated by the impressively low space requirements of Chimp and achieves significant improvements with regards to the state-of-the-art in PLA algorithms, offering better compression than Chimp, even when the precision error threshold is set extremely low.

Table 2 presents the maximum error needed so that Sim-Piece produces an equivalent compression ratio to that of Chimp. The maximum error is expressed as the percentage of the range of the dataset. For example, the range of Wind Direction is 360° and maximum error is 360° × 0,44% = 1,584°. The data from this table suggest that Sim-Piece, unlike existing lossless algorithms, can provide highly accurate data representations while matching the performance of the best lossless streaming algorithm. Increasing the maximum error threshold beyond the depicted numbers results in significantly higher compression ratios for Sim-Piece, often by two orders of magnitude, compared to the lossless algorithm.

Dataset	Maximum Error
Cricket	0,15%
FaceFour	1.25%
Lightning	0.12%
Wind Speed	0.43%
Wind Direction	0.44%

Table 2: Sim-Piece Maximum Error per Dataset that produces an equivalent compression ratio to that of Chimp.

References

Kitsios, X., Liakos, P., Papakonstantinopoulou, K. and Kotidis, Y. (2023), 'Sim-Piece: Highly Accurate Piecewise Linear Approximation through Similar Segment Merging', *Proceedings of the VLDB Endowment*, 16(8), pp. 1910-1922.

Pelkonen, T., Franklin, S., Teller, J., Cavallaro, P., Huang, Q., Meza, J. and Veeraraghavan, K. (2015), 'Gorilla: a fast, scalable, in-memory time series database', *Proceedings of the VLDB Endowment*, 8(12), pp. 1816–1827.

Liakos, P., Papakonstantinopoulou, K. and Kotidis, Y. (2022), 'Chimp: efficient lossless floating point compression for time series databases', *Proceedings of the VLDB Endowment*, 15(11), pp. 3058–3070.

Google (2011) Snappy. Available at: https://google.github.io/snappy/ (Accessed: 10 March 2023).

Keogh, E., Chakrabarti, K., Pazzani, M. and Mehrotra, S. (2001), 'Dimensionality reduction for fast similarity search in large time series databases', *Knowledge and Information Systems*, 3(3), pp. 263–286.

Hakimi, S.L. and Schmeichel, E.F. (1991), 'Fitting polygonal functions to a set of points in the plane', *CVGIP: Graphical Models and Image Processing*, 53(2), pp. 132–136.

Luo, G., Yi, K., Cheng, S.-W., Li, Z., Fan, W., He, C. and Mu, Y. (2015), 'Piecewise linear approximation of streaming time series data with max-error guarantees'. *2015 IEEE 31st International Conference on Data Engineering*.

Elmeleegy, H., Elmagarmid, A.K., Cecchet, E., Aref, W.G. and Zwaenepoel, W. (2009), 'Online piece-wise linear approximation of numerical streams with precision guarantees', *Proceedings of the VLDB Endowment*, 2(1), pp. 145–156.

Deligiannakis, A., Kotidis, Y. and Roussopoulos, N. (2007), 'Dissemination of compressed historical information in sensor networks' *The VLDB Journal*, 16(4), pp. 439–461.

Deligiannakis, A. and Kotidis, Y. (2005), 'Data reduction techniques in sensor networks', *IEEE Data Engineering Bulletin, Volume 28(1)*, March 2005.

Giatrakos, N., Kotidis, Y., Deligiannakis, A., Vassalos, V. and Theodoridis, Y. (2013), 'In-network approximate computation of outliers with quality guarantees', *Information Systems*, 38(8), pp. 1285–1308.

Dau Hoang Anh, Keogh Eamonn, Kamgar Kaveh, Yeh Chin-Chia Michael, Zhu Yan, Gharghabi Shaghayegh, Ratanamahatana Chotirat Ann, Yanping, Hu Bing, Begum Nurjahan, Bagnall Anthony, Mueen Abdullah, Batista Gustavo, and Hexagon-ML. 2018. The UCR Time Series Classification Archive. https://www.cs.ucr.edu/~eamonn/time_series_data_2018.

National Ecological Observatory Network (NEON). 2022. Barometric pressure above water on-buoy (DP1.20004.001). https://doi.org/10.48443/V4AP-NY05.

National Ecological Observatory Network (NEON). 2022. Windspeed and direction above water on-buoy (DP1.20059.001). https://doi.org/10.48443/E16C-8686.