Matrix Profile Index Prediction for Streaming Time Series

Maryam Shahcheraghi, Trevor Cappon, Samet Oymak, Evangelos Papalexakis, Eamonn Keogh, Zachary Zimmerman, and Philip Brisk

University of California Riverside
{sshah073, tcapp001, zzimm001}@ucr.edu, oymak@ece.ucr.edu
{epapalex, eamonn, philip}@cs.ucr.edu

Abstract

Discovery and classification of motifs (repeated patterns) and discords (anomalies) in time series is fundamental to many scientific fields. These and related problems have effectively been solved for offline analysis of time series; however, these approaches are computationally intensive and do not lend themselves to streaming time series, such as those produced by IoT sensors, where the sampling rate imposes real-time constraints on computation and there is strong desire to locate computation as close as possible to the sensor. One promising solution is to use low-cost machine learning models to provide approximate answers to these problems. For example, prior work has trained models to predict the similarity of the most recently sampled window of data points to the time series used for training. This work addresses a more challenging problem, which is to predict not only the "strength" of the match, but also the relative location in the representative time series where the strongest matching subsequences occur. We evaluate our approach on two different real world datasets; we demonstrate speedups as high as about 30× compared to exact computations, with predictive accuracy as high as 87.95%, depending on the granularity of the prediction.

1 Introduction

This paper describes a machine learning system that predicts similarities between a streaming time series and a representative time series used to train the model. The system approximates the locations in the representative time series of subsequences that are most similar to the most recent window of sampled data points from the streaming time series. Prior work has computed this information exactly at great cost (1; 2; 3; 4; 5; 6), but cannot meet the real-time constraints on computation that are needed to process streaming data. One existing technique can predict similarities to a representative time series, but cannot approximate indices (7). For example, if applied to earthquake data, this method predicts if a recently sampled window of seismograph readings is similar to *something* in the historical record; whereas, the system presented in this paper predicts similarity *and* the approximate temporal location of similar subsequences, which may or may not be in the vicinity of a recorded seismic event. The method is not specific to seismic data and can be applied to any time series.

In this paper, we solve the index prediction problem using a tree of machine learning models of varying granularity. Using seismic activity prediction as an example, suppose that our window contains the most recent minute of sampled datapoints. An exact search would compute the similarity of this window to each minute in the historical record. Assuming that we limit the historical record to a year's worth of data, the tree would first predict the month(s) during which similar subsequences occur; then, within each predicted month, it would then predict the weeks, days, hours, and finally

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

minutes, leveraging the tree structure to limit the search to the most promising regions of the historical record. This approach readily generalizes to other domains.

2 Background Material & Related Work

A Time Series T is a sequence of n datapoints ordered in time. A *subsequence of T* starting at position i with length $m \ll n$ is denoted as $T_{i,m}$ or T_i if m can be inferred from context.

Time series analysis to solve problems such as classification, clustering, motif discovery, and anomaly detection (1; 8; 9; 10). Among these techniques, the *Matrix Profile* and its extensions can solve these problems offline once the time series has been fully collected (1; 2; 3; 4; 5; 6); the *Learned Approximate Matrix Profile (LAMP)* can solve them online through prediction (7).

The Matrix Profile is an array of the maximum Pearson correlations between each $T_i \in T$ and all the other subsequences $T_j \in T$; The subsequences with maximum correlation to T_i are called the *Nearest Neighbors*. A separate array, the the Matrix Profile Index, contains the indices (locations) of the nearest neighbor of each subsequence $T_i \in T$ The pair comprising T_i and its nearest neighbor in T is called T_i*'s self-join*; The pair of $T_i \in$ one time series (T_A) and its nearest neighbor in another time series (T_B) is called the T_i*'s AB-join*. The *AB-Matrix Profile* is a similar extension that stores correlations between two time series.

The Matrix Profile is expensive to compute. As a lower-cost alternative, LAMP (7) is a function $L_{T_R} : T_S \rightarrow [0, 1)$ that predicts the AB-Matrix Profile between a *representative* time series T_R and a much shorter window of sampled datapoints T_S from a streaming time series. In principle, any machine learning model can be used; the original LAMP paper employed a small 1D convolutional neural network (1D-CNN) with residual connections; we use the same 1D-CNN here. LAMP predictions run orders of magnitude faster than computing AB-Matrix Profiles directly. This makes LAMP a suitable option for real-time applications deployed on embedded systems.

The work presented here extends LAMP for index prediction. It exhibits similarities to time series indexing methods such as iSAX (11), iSAX 2.0 (12), iSAX2+ (13) and ADS+ (14), among others. Specifically, our technique constructs a tree of LAMP models of varying granularity; this is similar to TARDIS (15), which is a tree-structured indexing method built upon iSAX. All these methods index a time series, not its Matrix Profile. The approach presented here is also influenced by a recent learned indexing methods for database structures, such as B-Trees (16).

3 Proposed Method

To address the problem of Matrix Profile index prediction we propose a tree of LAMP models, in which each vertex corresponds to a LAMP model trained on different portions of a long time series. We refer to it as *LAMP-Tree*. For example, Figure 1 shows an example in which the training data is divided into 48 segments w_i. The 48 leaves are each trained on one week of data; each internal node is trained on a four-week period of data; and the root node is trained on the full 48-week worth of data; the root represents the original LAMP model (7).

3.1 LAMP-Tree Queries (Inference)

The input to the LAMP-Tree is a short time series T_Q called a *query* (i.e., the most recent window of sampled datapoints) along with a user-specified threshold. A match occurs if the LAMP model at each node predicts a correlation value above the threshold: assuming that the LAMP models are reasonable accurate, at least one subsequence in leaf reachable from the current node will match the query. This yields simple top-down querying procedure, starting at the root: if a match is predicted at a non-leaf node, all of its children are search recursively; if a non-match is predicted, the search is pruned at that node. If a match is predicted at a leaf node, then that leaf is added to the set of matching leaves returned by the query.

Figure 2 depicts the LAMP-Tree inference process. Given the window W of recently-sampled datapoints, the LAMP model at the root reports a strong match, indicating that a similar subsequence occurred at least once in the 48 weeks' worth of data. Next, we run the LAMP models at the twelve internal nodes: a strong match is reported at w_{13-16}, but not at other internal nodes. This means that

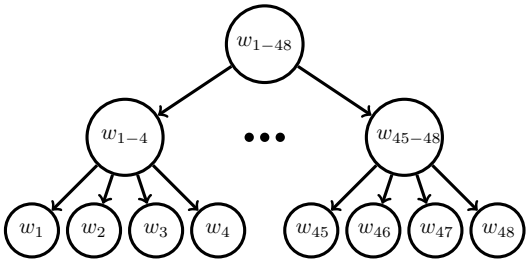

Figure 1: A 48-week LAMP-Tree: leaf node w_i represents the i^{th} week, and corresponds to subsequence of training data T_{w_i} with LAMP model L_{w_i}; internal node w_{i-j} represents the i through j^{th} week and corresponds to subsequence of training data $T_{w_{i-j}}$ and LAMP model $L_{w_{i-j}}$.

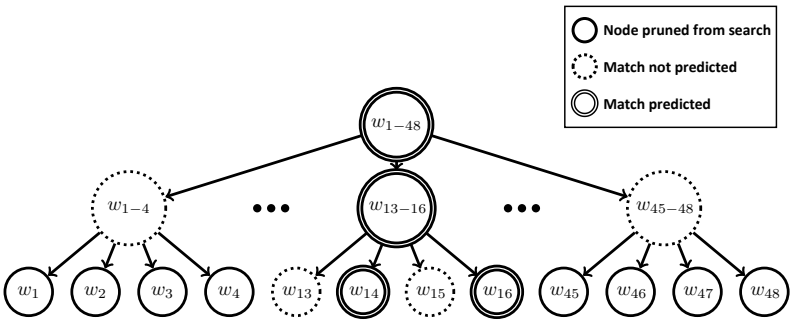

Figure 2: Querying the LAMP-Tree in Figure 1. The query returns w_{14} and w_{16} as matching leaves.

any predicted matches occurred between w_{13} and w_{16}; the search continues exclusively in the subtree rooted at w_{13-16}, and positive matches are predicted for w_{14} and w_{16}. In this example, weeks are the finest granularity, but in principle the tree could extend to days, hours, etc.

3.2 LAMP-Tree Training

We summarize the LAMP-Tree training process using the 48-week time series in Figure 1. The model developer specifies the arity of each level of the tree; we assume that the representative time series has sufficient length to admit a complete tree. Figure 1 has an arity of 12 at the root and 4 at internal nodes. We introduce overlap between consecutive training segments T_{w_i}, $T_{w_{i+1}}$ to ensure that no correlations are lost at subsequence boundaries. Training proceeds bottom-up, starting at the leaves. Let w_i be a leaf node. We construct a LAMP model L_{w_i} trained on T_{w_i}'s Self-Join (7). The user specifies a parameter p, which throttles the amount of additional training data. If there are N leaf nodes in total, we randomly select $\sim \lceil pN \rceil$ additional leaf nodes and retrain their LAMP models using subsequence T_i to improve accuracy. We then recursively train a LAMP model $L_{w_{i-k}}$ for each non-leaf node w_{i-k} using its children's LAMP models. For each training segment, we train $L_{w_{i-k}}$ to predict the maximum correlation among its children's LAMP models; by induction, this implies that we are training $L_{w_{i-k}}$ to predict the maximum correlation between the training segment and the best matching nearest neighbor among the leaf nodes $w_j, i \leq j \leq k$. This process terminates at the root.

4 Experimental Results

Experimental Setup: We evaluated LAMP-Tree using an Intel Core i9-9900 CPU running at 3.1 GHz with 32 GB RAM and running Ubuntu 18.04.4 LTS. We trained the 1D-CNNs employed in our LAMP models and computed the Matrix Profile (for comparative purposes) using the SCAMP algorithm (4) on one core of an Nvidia TU102 GPU.

Datasets: We used a publicly available dataset of Seismic data, collected near Parkfield, CA in 2004 that contains several catalogued earthquakes, consisting of 140M datapoints for 112 days (starting from August, 15'th) (17).

Table 1: Summary of the LAMP-trees used for evaluation

STRUCTURE	LEAVES	INTERNAL
3L-4M	30×4M	5×24M
3L-10M	12×10M	3×40M
2L-20M	6×20M	-
2L-24M	5×24M	-
2L-40M	3×40M	-

Table 2: Indexing speedup of LAMP-Tree compared to exact Matrix Profile Index computation.

STRUCTURE	SPEEDUP
3L-4M	10.70
3L-10M	20.76
2L-20M	27.44
2L-24M	28.06
2L-40M	26.99

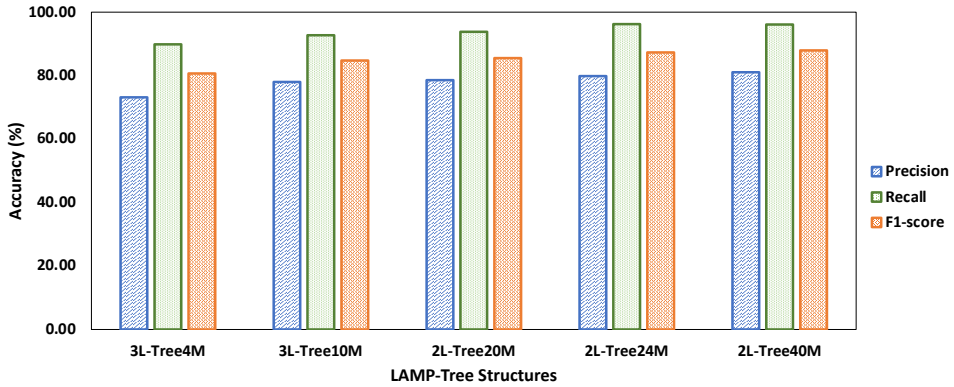

Figure 3: Indexing accuracy in terms of Precision, Recall, and F1-score for LAMP-Trees with p=0.3 for 4NN Matrix Profile.

LAMP-Trees: We trained five LAMP-Trees for the Seismic dataset. We use naming convention **aL-b** for each tree, where **a** is the number of levels and **b** is the number of datapoints per leaf. For example, a **3L-4M** LAMP-Tree has three levels and four million datapoints per leaf. The first 85% of each time series is used for training; the remaining 15% is used for evaluation.

Table 1 summarizes the LAMP-Trees that we trained, including the number of leaves and internal nodes, as well as the number of datapoints per leaf and internal node. For example 30×4M leaves means that the tree contains thirty leaves and that each leaf encompasses four million datapoints.

4.1 Indexing Time

We compare the indexing time of five LAMP-Trees to computing the AB-Matrix Profile Index directly. We set the threshold for a match to a correlation value of 0.75, with $p = 0.3$ for retraining. Table 2 reports the speedup for the target dataset. LAMP-Trees with smaller leaf node training segments and more overall nodes (e.g. 3L-4M) tend to show less speedup than the ones with larger leaf node training segments and fewer overall nodes (e.g. 2L-40M). Moreover, if no match is found by a parent node the search stops and its subtree is pruned. So, the indexing time is also affected by parent nodes' prediction. For instance, in our experiment 2L-24M shows more speedup than 2L-40M, even though it has more overall nodes. All speedups reported are in the 10-30× range.

4.2 Indexing Precision & Recall

We evaluate the accuracy of the LAMP-Tree (p = 0.3, threshold of 0.75) in terms of *precision*, *recall*, and F_1 *score* (the harmonic mean of precision and recall). We evaluated the LAMP-tree predictions by comparing to the 4 highest correlated subsequences to the given query. Figure 3 reports the precision, recall, and F1-score results for these experiments. Although there isn't a significant difference in accuracy between the LAMP-Trees, the general trend is that those with larger leaf node training segments and fewer leaf nodes are the most accurate; this is because each leaf has a larger training segmnent, leading to more accurate LAMP models. On the other hand, LAMP-Trees with smaller leaf node segments lead to faster exact computation when a match occurs but they are less accurate due to the less training data.

Acknowledgment

This work was supported in part by NSF Awards #1528181, #1763795, #1901379, and #1932254. P. Brisk has a small equity stake in Shapelets, a company providing decision-support software for time series. The authors declare no other competing interests.

References

[1] C.-C. M. Yeh, Y. Zhu, L. Ulanova, N. Begum, Y. Ding, H. A. Dau, D. F. Silva, A. Mueen, and E. Keogh, "Matrix profile i: all pairs similarity joins for time series: a unifying view that includes motifs, discords and shapelets," in *2016 IEEE 16th international conference on data mining (ICDM)*, pp. 1317–1322, Ieee, 2016.

[2] Y. Zhu, Z. Zimmerman, N. S. Senobari, C.-C. M. Yeh, G. Funning, A. Mueen, P. Brisk, and E. Keogh, "Matrix profile ii: Exploiting a novel algorithm and gpus to break the one hundred million barrier for time series motifs and joins," in *2016 IEEE 16th international conference on data mining (ICDM)*, pp. 739–748, IEEE, 2016.

[3] C.-C. M. Yeh, Y. Zhu, L. Ulanova, N. Begum, Y. Ding, H. A. Dau, Z. Zimmerman, D. F. Silva, A. Mueen, and E. Keogh, "Time series joins, motifs, discords and shapelets: a unifying view that exploits the matrix profile," *Data Mining and Knowledge Discovery*, vol. 32, no. 1, pp. 83–123, 2018.

[4] Z. Zimmerman, K. Kamgar, N. S. Senobari, B. Crites, G. Funning, P. Brisk, and E. Keogh, "Matrix profile xiv: Scaling time series motif discovery with gpus to break a quintillion pairwise comparisons a day and beyond," in *Proceedings of the ACM Symposium on Cloud Computing*, pp. 74–86, 2019.

[5] F. Madrid, S. Imani, R. Mercer, Z. Zimmerman, N. Shakibay, and E. Keogh, "Matrix profile xx: Finding and visualizing time series motifs of all lengths using the matrix profile," in *2019 IEEE International Conference on Big Knowledge (ICBK)*, pp. 175–182, IEEE, 2019.

[6] Y. Zhu, C.-C. M. Yeh, Z. Zimmerman, and E. Keogh, "Matrix profile xvii: Indexing the matrix profile to allow arbitrary range queries,"

[7] Z. Zimmerman, N. Shakibay Senobari, G. Funning, E. Papalexakis, S. Oymak, P. Brisk, and E. Keogh, "Matrix profile xviii: Time series mining in the face of fast moving streams using a learned approximate matrix profile," in *2019 IEEE International Conference on Data Mining (ICDM)*, pp. 936–945, 2019.

[8] M. Karimi, A. Jahanshahi, A. Mazloumi, and H. Z. Sabzi, "Border gateway protocol anomaly detection using neural network," in *2019 IEEE International Conference on Big Data (Big Data)*, pp. 6092–6094, IEEE, 2019.

[9] M. Linardi, Y. Zhu, T. Palpanas, and E. Keogh, "Matrix profile x: Valmod-scalable discovery of variable-length motifs in data series," in *Proceedings of the 2018 International Conference on Management of Data*, pp. 1053–1066, 2018.

[10] N. S. Senobari, G. J. Funning, E. Keogh, Y. Zhu, C.-C. M. Yeh, Z. Zimmerman, and A. Mueen, "Super-efficient cross-correlation (sec-c): A fast matched filtering code suitable for desktop computers," *Seismological Research Letters*, vol. 90, no. 1, pp. 322–334, 2019.

[11] J. Shieh and E. Keogh, "i sax: indexing and mining terabyte sized time series," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 623–631, 2008.

[12] A. Camerra, T. Palpanas, J. Shieh, and E. Keogh, "isax 2.0: Indexing and mining one billion time series," in *2010 IEEE International Conference on Data Mining*, pp. 58–67, IEEE, 2010.

[13] A. Camerra, J. Shieh, T. Palpanas, T. Rakthanmanon, and E. Keogh, "Beyond one billion time series: indexing and mining very large time series collections with isax2+," *Knowledge and information systems*, vol. 39, no. 1, pp. 123–151, 2014.

[14] K. Zoumpatianos, S. Idreos, and T. Palpanas, "Indexing for interactive exploration of big data series," in *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pp. 1555–1566, 2014.

[15] L. Zhang, N. Alghamdi, M. Y. Eltabakh, and E. A. Rundensteiner, "Tardis: Distributed indexing framework for big time series data," in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pp. 1202–1213, IEEE, 2019.

[16] T. Kraska, A. Beutel, E. H. Chi, J. Dean, and N. Polyzotis, "The case for learned index structures," in *Proceedings of the 2018 International Conference on Management of Data*, pp. 489–504, 2018.

[17] N. C. E. D. Center, "Northern california earthquake data center." HRSN (2014), High Resolution Seismic Network. UC Berkeley Seismological Laboratory. Dataset. doi:10.7932/HRSN., 2014.