Chapter 1

Leveraging Nearest Neighbors for Time Series Forecasting with Matrix Profile

1.1 Background

Humans have made predictions since ancient times. In ancient societies, accurate predictions were important to the success of subsistence activities such as hunting, planting, and harvesting. There was a need to predict weather dynamics, such as rainfall and temperature. For example, it is crucial to plant during the period with sufficient rainfall and appropriate temperatures. Our ancestors used divination tools such as turtle shells, wooden blocks (moon blocks), or bones to make predictions. Without doubt, the accuracy was not guaranteed. Even in modern societies, prediction is still essential. Predicting traffic-jam patterns only a few hours ahead can save time by enabling the selection of an alternative route [1]. A wealth could be created by forecasting stock market trends. Predicting the future, also known as time series forecasting, is crucial.

With advances in hardware technologies, we collect enormous amounts of data from diverse sources, such as smart sensors and social media platforms, continuously in the form of time series data. A time series is an ordered sequence of measures, represented in real-valued numbers, at discrete equal-interval timestamps [2]. The vast data collections have created the era of "Big Data", which provides a wealth of datasets for developing and deploying reliable, robust, data-driven forecasting techniques to discover patterns and extract valuable information [3]. Applications can be found in the financial sector, such as predicting

business cycles and stock market movements [4, 1, 5, 6] and the medical field, such as the status of critical patients according to their vital signs [7, 8] and the propagation of diseases such as influenza [6, 9] and COVID-19 [7, 10].

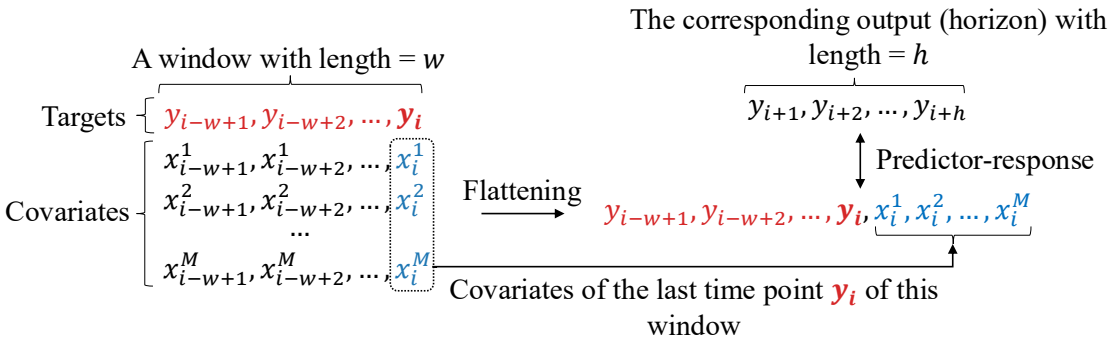

Figure 1.1: Flattening a 2D window of size $(1 + M) \times w$ to a 1D vector of length $w + M$. The resulting vector serves as the *predictor* for the regressor. Its expected *response* is the vector of length h. The regressor learns from this predictor-response pair.

A recent study [11] shows that, in time series classification, a non-parametric, instance-based method, namely nearest-neighbor classifiers (1-NN) and its generalized form k-NN, with appropriate distance measures, such as Dynamic Time Warping (DTW), despite their simplicity, perform well and are therefore commonly used as benchmarks. In detail, when a new instance is to be classified, k-NN finds its k nearest neighbors in the training set and returns their majority label among them. k-NN is considered a lazy learner because the training steps involve only memorizing all the instances verbatim; no higher-level concepts have been learned. In addition, in time series forecasting, a recent study demonstrates that a well-known machine learning baseline, Gradient Boosting Regression Tree (GBRT), such as XGBoost, equipped with an appropriate data engineering of the data, can achieve competitive or even superior performance than the deep learning method. In detail, they transform the time series forecasting task into a window-based regression problem, as shown in Figure 1.1. For each training window of length w with the last time point $\mathbf{y_i}$, and its lagged values $y_{i-1}, y_{i-2}, \ldots, y_{i-w+1}$ are concatenated with covariates $x_i^1, x_i^2, \ldots, x_i^M$ to form a *predictor* for a multi-output GBRT. This transformation is called flattening. The corresponding response is the following h points of $\mathbf{y_i}$. It provides a simple, more efficient yet accurate method for time series forecasting.

Moreover, k-NN has also shown to be a promising method for time series forecasting [12]. The k-NN uses the lagged values of the last time point to form a query Q. It identifies the k previous similar subsequences to Q and uses their

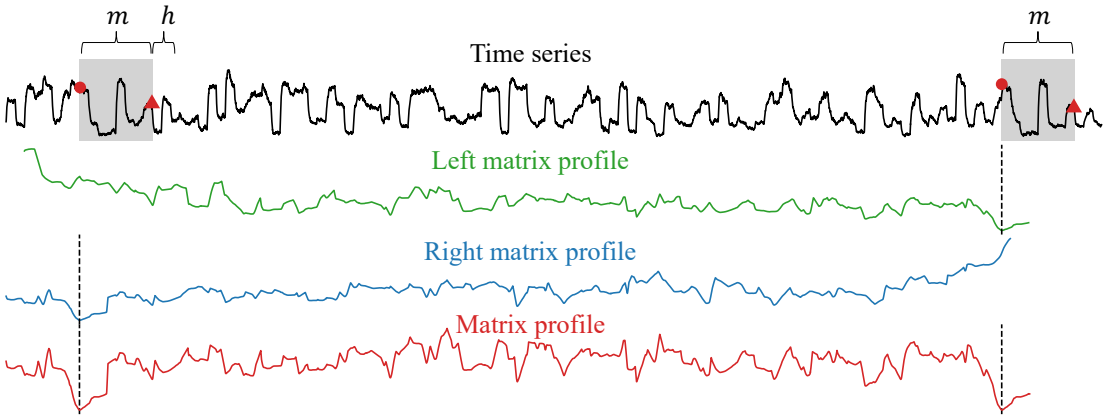

Figure 1.2: The left (right) matrix profile and the matrix profile of a time series. The matrix profile shows the distances between each m-subsequence and its nearest neighbor, where m is a user-given value. The left (right) matrix profile shows the same information but is limited to its left (right) nearest neighbor. For a particular m-subsequence shown by the right gray box, its nearest neighbor is the left gray box, as indicated by the dashed line in the left matrix profile and the matrix profile. Similarly, the nearest neighbor of the left gray box is the right gray box, as indicated by the dashed line in the right matrix profile and the matrix profile. The first (last) point in each box is denoted by a red circle (triangle). h denotes the length of the immediate subsequence of the nearest neighbor. Intuitively, this subsequence should be similar to the immediate subsequence (i.e., future) of the right gray box. To note, the left (right) matrix profile starts (ends) at a later (earlier) index because the corresponding nearest neighbor with length m does not exist.

Algorithm 1 The brute force approach to compute Matrix Profile

```
1: for i ← 1 to n − m + 1 do
2:     for j ← 1 to n − m + 1 do
3:         Compute the z-normalized Euclidean distance between t_{i,m} annd t_{j,m}.
```

immediate subsequences to predict the immediate subsequence, which is the forecasting window, of Q. The intuition is that history repeats itself. The previous (historical) subsequences that are similar to Q can provide a hint about the future of Q. They are similar, and so are their immediate subsequences. Figure 1.2 depicts this idea. Observe that the immediate subsequence of the right gray box is similar to that of the left gray box. The left gray box is the nearest neighbor of the right gray in the "past".

Based on these findings, this study proposes a method to improve the performance of existing forecasters by leveraging information from the nearest neighbors of each subsequence in the target variable. For each time point y_i of the target variable Y, a window of length w is constructed with y_i as the last point, then

we retrieve the window's historical nearest neighbors and use their information to create new covariates for the window. The information includes the similarities between the window and its nearest neighbors, as well as the immediate subsequences of them. The similarity can be interpreted as a measure of confidence or weight in using the information from the corresponding nearest neighbor. The intuition is that, if the similarity of the window and a neighbor is high, then the future (i.e., immediate subsequence) of the neighbor should also be similar to the future of the window. The fundamental difference between this study and previous approaches [13, 12, 14] is that they directly use the subsequent points for prediction, whereas we use the nearest neighbor information for each subsequence as covariates, which are used as primitives for other forecasters. We explain this subtle difference by Figure 1.2. The previous approaches simply use the information of the nearest neighbors of the last look-back window, which consisted of the last time point and its lagged values (i.e., the right gray box), for prediction. In contrast, we use the information of the nearest neighbors of **all** of the windows.

We use the Matrix Profile [15, 16] to annotate the nearest neighbor for each m-subsequence of a time series T of length n. The distance is the z-normalized Euclidean distance. It may seem computationally expensive to perform this annotation at first glance. Algorithm 1 shows the brute force approach to compute the matrix profile. The two for-loops and the computation of z-normalized Euclidean distance, which takes $\mathcal{O}(m)$, indicate that the computational complexity is $\mathcal{O}(n^2m)$. The space complexity is $\mathcal{O}(n^2)$ because of the pairwise distance of each subsequence with the other subsequence. However, the matrix profile can be computed in $\mathcal{O}(n^2)$ using an exact method, namely STOMP [15] or its community-open-sourced version, STUMP [17]. Besides, the running time can be further sped up by parallelization for a single machine with multiple computation units, such as CPUs or GPUs. The tool also allows us to compute the left matrix profile to find the left nearest neighbor of each window. To note, the matrix profile annotates a time series with information about the nearest neighbor of each subsequence, including the similarity with its nearest neighbor and its location, as shown in Figure 1.2.

In this study, we make the following contributions:

- We are the first to propose leveraging the matrix profile to create meaningful covariates that improve forecaster performance.

- We have proposed two methods namely, GBRT-NN, which uses the immediate subsequence as the covariates, and GBRT-NN-S, which also involves the similarity on top of GBRT-NN, as the covariates.
- Experimental results show that our approach can improve the GBRT forecaster in most cases, with a very small cost to compute the matrix profile.

The rest of this study is organized as follows. Section 1.2 presents the related-work. In Section 1.3, we introduce the necessary background knowledge, then introduce our method. Section 1.4 contains an empirical evaluation. Finally, we conclude this study and provide future work in Section 1.5.

1.2 Related Work

Many methods have been developed for time series forecasting. Traditional methods include rolling averages (RA), vector auto-regression (VAR) [1, 18], and auto-regressive integrated moving averages (ARIMA) [19, 20, 18]. Because of their rigorous statistical properties, they have long been the standard. The shortcomings of ARIMA and its variants include their high computational cost [1]. In contrast, VAR is arguably the most widely used method, particularly in multivariate time series analysis, owing to its simplicity. However, most of these traditional approaches have certain limitations. They perform well when the data meet specific statistical assumptions, such as stationarity [21], which means that the mean and variance of the time series remain constant over time. It motivates the community to develop machine learning methods, particularly deep learning methods for time series forecasting. Many deep learning models have been proposed, including RNN-based models, CNN-based models, GNN-based models, Transformer-based models, and compound models that incorporate different base models mentioned before [6]. The compound models are promising. For example, RNNs are well suited to capturing long-term dependencies, whereas CNNs are well suited to capturing short-term dependencies. A good way to improve performance is to compound them. For example, LSTnet [1] integrates CNN, RNN, and autoregressive [22] techniques to extract both short-term and long-term patterns. Using the occupancy rate of a freeway as an example [1] to explain these two patterns, the "short-term" patterns refer to the morning peaks against evening peaks, while the "long-term" patterns refer to the workday patterns against weekend patterns. Clearly, a good forecaster needs to capture and distinguish both kinds of patterns. Despite the superior performance deep learning methods have achieved, they tend

to be overly complex, opaque, and incur high computational costs compared to traditional techniques.

1.3 Method

In this section, we first formulate the time series forecasting problem, followed by the evaluation method. We then explain how to use a Gradient Boosting Regression Tree (GBRT) for forecasting. Subsequently, we discuss how to leverage the nearest neighbors' information of each subsequence to improve GBRT's performance. Finally, we discuss how to compute those nearest neighbors using the Matrix Profile.

To begin, we define the data type of interest: time series.

Definition 1 (Time series)**.** A *time series* $T = t_1, t_2, \ldots, t_n$ is a sequence of real-valued numbers with length $= n$.

In Definition 1, T is a univariate time series where each entry is a scalar number. If each entry is a vector consisting of scalar numbers with size > 1, T is a multivariate time series. A multivariate time series can be regarded as a sequence of vectors. It can also be represented as a vector of univariate time series, where each univariate time series is referred to as a channel. In a dataset with more than one time series, we use T_i to denote a time series in a dataset with N time series, where $1 \leq i \leq N$.

The local properties of T can be analyzed through its subsequences.

Definition 2 (Subsequence)**.** A *subsequence* $T_{i,m} = t_i, t_{i+1}, \ldots, t_{i+m-1} = t_{i:i+m-1}$ of a T is a sequence that consists of a continuous subset of the entries from T of length m starting from i.

1.3.1 Problem Formulation

Time series forecasting is the task of predicting h-future values $y_{t+1}, y_{t+2}, \ldots, y_{t+h}$ of a target Y at the current time point t. In this study, there is only one target variable Y. h is the number of steps we want to predict in the future. The simplest case is one-step-ahead forecasting, where $h = 1$. The predicted value is denoted as $\hat{y}_{t+1}$ where the actual value is y_{t+1}. It is preferable to predict multiple points in the future. It is called multi-horizon (multi-step) forecasting, where $h > 1$. The task of forecasting is encoded in Equation 1.1.

$$\hat{y}_{t+\tau} = f(y_{t-w+1:t}, x_{t-w+1:t}, u_{t-w+1:t+\tau}, \tau) \tag{1.1}$$

where

- $\hat{y}_{t+\tau}$ is a prediction of the target value at $t+\tau$, where $\tau \in \{1, 2, \ldots, h\}$.
- $y_{t-w+1:t}$ are the actual values consisting of the current value y_t and the lag values before it. y_{t-i} is called the lag of i or i-lag.
- x_t are inputs that can only be known historically at time t. x_{t+1} is not known at t.
- u_t are kwown inputs for all time. For example, the date information such as the day of the week or month [21]. Even at t, u_{t+i} where $1 \leq i \leq \infty$ are known.

x_t and u_t are called covariates of y_t. The input of Equation 1.1 is a look-back window w.

We explain the task of forecasting in terms of Equation 1.1. The forecasting process estimates the value of $y_{t+\tau}$, denoted by $\hat{y}_{t+\tau}$ with the aim to minimize the error function, typically represented as a function of $y_{t+\tau} - \hat{y}_{t+\tau}$ for each τ. It is obvious that date information is useful when the target variable depends on when the measurement is taken. For example, if the target variable is the electricity consumption rate, there is a clear pattern by month: consumption is higher during the winter and summer months, when air conditioners and heaters are used. x_t provides additional information about the state of y_t. For example, if the target variable is the body temperature, and x_t tells us the severity of the sore throat, we may guess the body temperature will raise tomorrow.

1.3.2 Evaluation Method

Given a dataset D of N time series T_i, where $1 \leq i \leq N$, we explain how to evaluate the performance of a forecaster on T_i. The error made by the forecaster on D is simply the summation of errors made by the forecaster on each T_i. We now focus on a single time series; hence, we drop the index i. T is divided into two subsequences, namely training subsequence T_{train} and test subsequence T_{test}.

In our study, the forecaster is only allowed to train on T_{train}. Recall that we are predicting the h future values from the w values just before them. Hence, we use a sliding window of length $w+h$ to generate the w-predictor-h-response pairs in T_{train}, enabling the forecaster to train on them. During the test (evaluation) phase, we do rolling forecasts. We predict based on the ground truth, not results generated from the model. It is used to prevent the accumulation of errors.

This concept is called "Teacher forcing" [23] because the teacher's values are "force fed" into the forecaster when we "roll" the forecaster on the T_{test} [24]. The intuition is that, suppose each question (except the first one) in an exam depends on the answers to the previous questions, rather than simply grading every answer in the end, a teacher would grade (evaluate) the answer once it is given by the student, and provide the correct answer to the student so he can answer the next question based on the correct answer.

To evaluate the forecaster, we used the following three evaluation metrics defined as:

- Root Mean Square Error (RMSE)

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{1.2}$$

- Weighted Absolute Percentage Error (WAPE)

$$\text{WAPE} = \frac{\sum_{i=1}^{n}|y_i - \hat{y}_i|}{\sum_{i=1}^{n}|y_i|} \tag{1.3}$$

- Mean Absolute Error (MAE)

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{1.4}$$

where n is the length of the time series, y_i, $\hat{y}_i$ is ground true value and predicted value, respectively. RMSE and MAE are widely used metrics. MAE can better reflect the actual error situation than RMSE [4]. WAPE was introduced by [25]. By rewriting Equation 1.3 to Equation 1.5, it is more obvious that it is a weighted absolute percentage error.

$$\text{WAPE} = \sum_{i=1}^{n} w_i \frac{|y_i - \hat{y}_i|}{|y_i|} \tag{1.5}$$

where the weights are given by

$$w_i = \frac{|y_i|}{\sum_{i=1}^{n}|y_i|} \tag{1.6}$$

For all of them, a lower value is better.

1.3.3 Gradient Boosting Regression Tree (GBRT)

Gradient boosting [26] is a boosting algorithm that ensembles a group of weak learners (usually decision trees) to make predictions. It sequentially adds learners to an ensemble, with each learner connecting its predecessor. It constructs weak learners in a way that each learner strategically corrects the predecessor's mistakes by fitting the new learner to the residual errors made by the predecessor [27, 28]. It can be used in classification and regression. In this study, we focus on its use in regression. For the usage in regression, the model is called "Gradient Boosting Regression Tree (GBRT)". Some popular optimized implementations of gradient boosting are XGBoost [29], CatBoost [30], and LightGBM [31]. In this study, we use XGBoost.

In order to apply GBRT into time series forecasting problem, we need to cast the input into an appropriate format to input into GBRT. The casting approach is similar to successful time-series forecasting models, which reconfigure the time series into windowed inputs [19]. Figure 1.1 presents the reconfiguration. For each entry of the target variable y_i, we retrieve its u_i, such as the day information from the calendar. Hence, for each y_i, it is associated with x_i and u_i. To simplify the notation, we absorb u_i into x_i, and it is called the covariates of y_i. The 2D window, as shown on the left-hand side in Figure 1.1, with size $w \times M$, where M is the total number of covariates, is flattened into a 1D array on the right-hand side with length $w + M$. To note, as suggested in the literature [19], only the covariates of the last time-point i are kept and appended to the final vector. By reconfiguration, we obtain the predictor-response pairs for training. In detail, the predictor is $y_{i-w+1}, y_{i-w+2}, \ldots, \mathbf{y_i}, x_i^1, x_i^2, \ldots, x_i^M$ where the red part refers to the current target value $\mathbf{y_i}$ and its lag values, and the blue part refers to the covariates of $\mathbf{y_i}$. The corresponding output is $y_{i+1}, y_{i+2}, \ldots, y_{i+h}$, with length $= h$. We predict the h-horzon from the w-look back window (i.e, $w + M$-flattened predictor). With this predictor-response formulation, the forecasting problem becomes a multi-output regression problem. Standard XGBoost cannot return a sequence of predicted values; it only returns a single number [19]. To note, a multi-output regression problem is simply a group of single-output regression problems. In other words, XGBoost internally simply treats the prediction of h-steps as h individual problems. Hence, the final output is produced by the h regressors rather than by a single model. One may argue that the h regressors operate individually and hence the temporal relationship in the output sequence is lost. However, as the

individual regressors are trained on the same flattened input, the prediction would still preserve the temporal relationship [19].

1.3.4 Matrix Profile

Our proposed method has a simple intuition. We would like to create covariates for each time point in the time series, as in the case of using the date information to create the covaraites. For example, given the timepoint date (2026-02-01, 14:38), we can deduce the minute (38), hour (14), day (1), month (2), and year (2026). Besides, by reading the calendar, the day of the week (Sunday), the day of the year (31+1=32), the week of the year (the firth week in 2026) can also be obtained. Given a time point t_p in T, and a user-given m, we want to use the m-subsequence with t_p as the ending point as a query Q, and to find its left m-subsequence that is the nearest neighbor of Q. Hence, the immediate subsequence of the nearest neighbor informs us of the dynamics of the immediate subsequence of Q. In Figure 1.2, the red triangle in the right gray box is t_p and the right gray box is Q. We seek its leftmost nearest neighbor, which turns out to be the left gray box. Note that we would like to perform this operation for each point t_p in the time series T, $m \leq t_p \leq |T|$.

We discuss how to use a data primitive called Matrix Profile [15, 16] for this operation. In brief, a matrix profile P is a time series that annotates an input time series T, storing the z-normalized Euclidean distance between each m-subsequence and its nearest neighbor in T. Besides, another supplementary time series, namely the Matrix Profile Index I, stores the location of the corresponding nearest neighbor. To begin with, we first define the distance profile of T.

Definition 3 (Distance profile)**.** A *distance profile* $D_i = d_{i,1}, d_{i,2}, \ldots, d_{i,n-m+1}$ of a T is a vector of the Euclidean distances between a given subsequence $T_{i,m}$ and each subsequences in T, where $d_{i,j}$ is the distance between $T_{i,m}$ and $T_{j,m}$, $1 \leq i, j \leq n - m + 1$.

The distances are measured between z-normalized time series. Equipped with the distance profile, we know all the Euclidean distances between a given subsequence $T_{i,m}$ and each subsequences in T. From this, we define the matrix profile P and its supplementary matrix profile index I as follows.

Definition 4 (Matrix profile)**.** A *matrix profile* $P = \min(D_1), \min(D_2), \ldots,$ $\min(D_{n-m+1})$ of T is a vector of Euclidean distances between every subsequence $T_{i,m}$ of T and its nearest neighbor in T.

Definition 5 (Matrix profile index)**.** A *matrix profile index* $I = I_1, I_2, \ldots, I_{n-m+1}$ of T is a vector of integers, where $I_i = j$ if $d_{i,j} = \min D_i$.

In this study, instead of the general matrix profile (index), we are interested in the left version of it. The left version of the matrix profile informs us of the information about the left nearest neighbor of any m-subsequences in T. It is shown in Figure 1.2. The left matrix profile has high values in the beginning because they can only find the left neighbor, and there are few left subsequences. In contrast, right matrix profile has low values in the beginning because there are many right subsequences To note, each entry i in Matrix Profile P_i is simply the min of P_i^L and P_i^R (i.e., $P_i = \min(P_i^L, P_i^R)$).

Definition 6 (Left distance profile)**.** A *left distance profile* $D_i^L = d_{i,1}, d_{i,2}, \ldots, d_{i,i-\lceil m/4 \rceil - 1}$ of T is a vector of Euclidean distances between a given subsequence $T_{i,m}$ and each subsequence that appears before $T_{i,m}$. To note, $i - \lceil m/4 \rceil - 1$ is the index location of the last eligible subsequence before $T_{i,m}$ because of the exclusion zone.

We discuss the idea of an exclusion zone. For a given m-subsequence Q, the distance profile is zero at the location of Q (because Q must be a nearest neighbor of Q) and close to zero just before and after it. But the corresponding m-subsequences are trivial matches of Q. We need to define a zone, namely an exclusion zone, for this region. It is defined as $m/2$ before and after the location of Q [15]. The community implementation uses $m/4$ instead [17].

Definition 7 (Left matrix profile)**.** A *left matrix profile* $P^L = \min(D_1^L), \min(D_2^L), \ldots, \min(D_{n-m+1}^L)$ of T is a vector of Euclidean distances between every subsequence $T_{i,m}$ of T and its nearest neighbor in T before it.

Definition 8 (Left matrix profile index)**.** A *left matrix profile index* I^L is a vector of integers $I_1^L, I_2^L, \ldots, I_{n-m+1}^L$ of T, where $I_i^L = j$ if $d_{i,j} = \min D_i^L$.

Note that the original indexing system in Matrix Profile uses the start point rather than the end point to define the subsequence. In Figure 1.2, the matrix profile is shown in the original definition. The gray boxes are associated with the start point. Hence, the red Matrix Profile starts at $t = 1$ rather than $t = m$. To facilitate our discussion, from now on, we refer to Matrix Profile and its variants using the index defined on the end point instead of the start point. Formally, we use the entries of Matrix Profile at $i - m + 1$ to be the matrix profile at i in our indexing system. Visually speaking, we shift the matrix profile to the right by $m - 1$.

To conclude this part, given the left matrix profile (index), we can compute, for each m-subsequence in T, the information on its left nearest neighbor, including its similarity and location. Besides, with the location (it now refers to the end point of the subsequence), we can retrieve the points after it and form the immediate subsequence with length $= h$ of the nearest neighbor.

A natural setting for m and the size of the immediate subsequence would be the w and h as defined in Figure 1.1, respectively.

1.4 Experiments

Table 1.1: Dataset Statistics. N is the number of time series in the dataset, while $|T| = n$ is the length of each time series. "rate" refers to the measuring rate. w is the size of the look-back window. h is the size of the forecasting window, also known as the forecasting horizon. T_{train} is the training subsequence of T. T_{test} is the test subsequence of T. To note, $|T_{\text{train}}| + |T_{\text{test}}| = n$.

	Data			**Forecasting Task**		
Dataset	N	n	rate	w, h	$\lvert T_{\text{train}}\rvert$	$\lvert T_{\text{test}}\rvert$
Electricity [32]	70	26,136	hourly	24	25,968	168
Traffic [32]	90	10,560	hourly	24	10,392	168
PeMSD7 [32]	228	12,672	/5 mins	9	11,232	1,440
Exchange-Rate [1]	8	7,536	daily	24	6,048	1,488

The code and data are available at `https://github.com/colemanyu/matrix-profile-motif-forecasting`. In this study, we focus on univariate time series forecasting. There is only one single target variable Y. We predict the future of Y only based on the past of it. The dataset used is listed in Table 1.1. For each time series T in the dataset, it is split into two contiguous time series, namely $T_{\text{train}} = T(1 : \text{split})$ and $T_{\text{test}} = T(\text{split} + 1 : |\text{T}|)$, where split defines the training-test split, which is shown in the column 6 in Table 1.1. To note, the N time series in a dataset are considered as multiple, independent unvariate time series instead of a multivaraite time series with channels $= N$. The original covariates are constructed from the timestamp information. The settings of w and h are adopted from the original corresponding papers [19]. It is not necessary for them to be the same.

GBRT-NN refers to a model that incorporates immediate sequence information. GBRT-NN-S refers to a model that incorporates immediate sequence infor-

mation with the similarity information. The experimental results are shown in Table 1.2 and Table 1.3.

Table 1.2: Experimental Results without original covariates (bold represents the best result and underlined represents the second best) (* refers to the results reported from [19]) (For the GBRT-related methods, we round to the nearest 4 decimal places for better comparison, for the others, we round to the 3 decimal places).

Dataset	Metric	LSTNet* [1]	TRMF* [33]	DARNN* [34]	ARIMA* [35]	GBRT [19]	GBRT-NN	GBRT-NN-S
Electricity [32]	RMSE	1095.309	136.400	404.056	181.210	**136.254**	142.035	138.354
	WAPE	0.997	**0.095**	0.343	0.310	0.103	0.101	0.100
	MAE	474.845	**53.250**	194.449	154.390	57.929	56.464	55.972
Traffic [32]	RMSE	0.042	0.023	0.015	0.044	0.012	0.016	**0.008**
	WAPE	0.102	0.161	0.132	0.594	0.108	0.079	**0.037**
	MAE	0.014	0.009	0.007	0.032	0.006	0.004	**0.002**
PeMSD7 [32]	RMSE	55.405	**5.462**	5.983	15.357	5.610	5.575	5.557
	WAPE	0.981	0.057	0.060	0.183	**0.051**	**0.051**	**0.051**
	MAE	53.336	3.329	3.526	10.304	2.984	2.965	**2.957**
Exchange-Rate [1]	RMSE	**0.018**	**0.018**	0.025	0.123	0.020	0.019	0.019
	WAPE	0.017	**0.015**	0.022	0.170	0.016	**0.015**	**0.015**
	MAE	0.013	**0.011**	0.016	0.101	0.012	0.011	0.012

Table 1.3: Experimental Results with original covariates (bold represents the best result and underlined represents the second best) (* refers to the results reported from [19]).

Dataset	Metric	DeepGlo* [32]	GBRT [19]	GBRT-NN	GBRT-NN-S
Electricity [32]	RMSE	141.285	132.669	**123.591**	124.215
	WAPE	0.094	0.0929	**0.089**	**0.089**
	MAE	53.036	52.0232	**49.606**	49.739
Traffic [32]	RMSE	0.026	0.014	0.018	**0.009**
	WAPE	0.239	0.109	0.101	**0.062**
	MAE	0.013	0.006	0.006	**0.003**
PeMSD7 [32]	RMSE	6.490	5.191	5.163	**5.156**
	WAPE	0.070	**0.048**	**0.048**	**0.048**
	MAE	3.530	2.796	2.779	**2.778**
Exchange-Rate [1]	RMSE	0.038	0.020	**0.019**	**0.019**
	WAPE	0.038	0.016	**0.015**	**0.015**
	MAE	0.029	**0.012**	**0.012**	**0.012**

1.5 Concluding Remarks

1.5.1 Future Work

Leveraging nearest neighbors' location information: It would be beneficial to retrieve the nearest neighbors for each subsequence in a specific range with respect to it. Real-world applications often require the separation of information

of short-term and long-term repeating patterns for making accurate predictions [1]. Notably, the matrix profile also provides the locations of the nearest neighbors from the matrix profile index. Using this location (index) information, we can retrieve the nearest neighbors of each subsequence in a specified range with respect to it to find those "short-term" and "long-term" patterns.

Extend to multidimensional case: This study focuses on univariate time series forecasting, where there is a single target and no exogenous inputs. It only requires us to find one-dimensional nearest neighbors. When there are multiple targets or a single target with multiple exogenous inputs, we need to identify multidimensional nearest neighbors [36] and leverage their information for forecasting.

Top-k neighbors and motifs: A simple extension is to consider the information not just from the nearest neighbor but from the k-nearest neighbors. Besides, we can use motifs instead of neighbors to obtain more stable "future" information for each window. Recall that a time series motif is a repeated pattern that consists of at least two occurrences. A motif can be considered as a family of nearest neighbors A nearest neighbor is a historical occurrence that instantiates this motif. The motif captures the ideal behavior. By finding the occurrences of a motif and considering their immediate subsequences, we can make a more confident guess about this motif. We outline the approach for finding members of a motif [1] Given a subsequence A in a time series T, we denote the left-hand side of A in T as T_L. We find A's nearest neighbor in T_L, denoted as B. They are the two members of a motif M. We want to identify other subsequences in T_L that belong to M. We define a threshold $\theta = r \times \mathrm{ED}(A, B)$, where $r > 1$. The center M_C of M is defined as the average of A and B. Then, we compute the distance profile between M_C and T_L. Any part of the distance profile that is smarter than θ points to a member of M in T_L. These members can then be added to M. After identifying all the members of M and excluding these members in the next consideration, we can find the next left nearest neighbor of A in T_L, and repeat the same process for finding the next motif. Given a set of immediate subsequences of members (occurrences) of M, we can compute a more stable immediate subsequence (future) associated with M by excluding the outliers among them or using the ensemble value, such as the mean or median of them, to cancel the noise.

Identify outliers of immediate subsequences: When we have a set of immediate subsequences, we can determine whether an immediate subsequence is an outlier or not by comparing it with others. We provide a heuristic to identify an outlier as follows. Recall that the length of a nearest neighbor and its immediate

[1]The idea has been mentioned in `https://www.cs.ucr.edu/~eamonn/TimeSeriesMotifs/`.

subsequence is m and h, respectively. We concatenate all nearest neighbors into a single long time series T'. To establish a clear boundary, we append a NaN value after each of them. It ensures that all matrix profile computations do not consider subsequences that span multiple neighbors. Then, we compute a distance profile of T' to find the nearest neighbor distance d_i of each neighbor i. Let S_i be the sequence consisting of neighbor i of length m and its immediate subsequence of length h. The expected nearest neighbor distance of S_i (found within the set of all extended sequences) should be proportional to the increase in length: $d_i \times \frac{m+h}{m}$. If the actual nearest neighbor distance of S_i is greater than $r' \times (d_i \times (m+h)/m)$ among the others, where r' is a user-given value, the immediate subsequence in S_i is considered as an outlier.

Bibliography

[1] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, "Modeling long- and short-term temporal patterns with deep neural networks," in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, ser. SIGIR '18. New York, NY, USA: Association for Computing Machinery, Jun. 2018, pp. 95–104.

[2] C. Chatfield, *The Analysis of Time Series: An Introduction, Sixth Edition*, 6th ed. New York: Chapman and Hall/CRC, Jul. 2003.

[3] O. Y. Al-Jarrah, P. D. Yoo, S. Muhaidat, G. K. Karagiannidis, and K. Taha, "Efficient machine learning for big data: A review," *Big Data Research*, vol. 2, no. 3, pp. 87–93, Sep. 2015.

[4] W. Li and K. L. E. Law, "Deep learning models for time series forecasting: A review," *IEEE Access*, vol. 12, pp. 92 306–92 327, 2024.

[5] N. I. Sapankevych and R. Sankar, "Time series prediction using support vector machines: A survey," *IEEE Computational Intelligence Magazine*, vol. 4, no. 2, pp. 24–38, May 2009.

[6] Z. Chen, M. Ma, T. Li, H. Wang, and C. Li, "Long sequence time-series forecasting with deep learning: A survey," *Information Fusion*, vol. 97, p. 101819, Sep. 2023.

[7] J. F. Torres, D. Hadjout, A. Sebaa, F. Martínez-Álvarez, and A. Troncoso, "Deep learning for time series forecasting: A survey," *Big Data*, vol. 9, no. 1, pp. 3–21, Feb. 2021.

[8] D. B. da Silva, D. Schmidt, C. A. da Costa, R. da Rosa Righi, and B. Eskofier, "Deepsigns: A predictive model based on deep learning for the early detection of patient health deterioration," *Expert Systems with Applications*, vol. 165, p. 113905, Mar. 2021.

[9] N. Wu, B. Green, X. Ben, and S. O'Banion, "Deep transformer models for time series forecasting: The influenza prevalence case," Jan. 2020.

[10] F. Martínez-Álvarez, G. Asencio-Cortés, J. F. Torres, D. Gutiérrez-Avilés, L. Melgar-García, R. Pérez-Chacón, C. Rubio-Escudero, J. C. Riquelme, and A. Troncoso, "Coronavirus optimization algorithm: A bioinspired metaheuristic based on the **COVID-19** propagation model," *Big Data*, vol. 8, no. 4, pp. 308–322, Aug. 2020.

[11] M. Middlehurst, P. Schäfer, and A. Bagnall, "Bake off redux: A review and experimental evaluation of recent time series classification algorithms," *Data Mining and Knowledge Discovery*, vol. 38, no. 4, pp. 1958–2031, Jul. 2024.

[12] F. Martínez, M. P. Frías, M. D. Pérez, and A. J. Rivera, "A methodology for applying k-nearest neighbor to time series forecasting," *Artificial Intelligence Review*, vol. 52, no. 3, pp. 2019–2037, Oct. 2019.

[13] S. Tajmouati, B. E. L. Wahbi, A. Bedoui, A. Abarda, and M. Dakkon, "Applying k-nearest neighbors to time series forecasting: Two new approaches," *Journal of Forecasting*, vol. 43, no. 5, pp. 1559–1574, 2024.

[14] D. T. Thi, N. T. Phong, and N. D. Bui, "Forecast timeseries based on matrix profile," *Journal of Informatics and Innovative Technologies (JIIT)*, 2021.

[15] 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)*. Barcelona, Spain: IEEE, Dec. 2016, pp. 1317–1322.

[16] Y. Zhu, M. Imamura, D. Nikovski, and E. Keogh, "Matrix profile vii: Time series chains: A new primitive for time series data mining," in *2017 IEEE International Conference on Data Mining (ICDM)*, Nov. 2017, pp. 695–704.

[17] S. M. Law, "**STUMPY**: A powerful and scalable python library for time series data mining," *Journal of Open Source Software*, vol. 4, no. 39, p. 1504, Jul. 2019.

[18] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time Series Analysis: Forecasting and Control*. John Wiley & Sons, May 2015.

[19] S. Elsayed, D. Thyssens, A. Rashed, H. S. Jomaa, and L. Schmidt-Thieme, "Do we really need deep learning models for time series forecasting?" Oct. 2021.

[20] G. E. P. Box and D. A. Pierce, "Distribution of residual autocorrelations in autoregressive-integrated moving average time series models," *Journal of the American Statistical Association*, vol. 65, no. 332, pp. 1509–1526, 1970.

[21] B. Lim and S. Zohren, "Time-series forecasting with deep learning: A survey," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 379, no. 2194, p. 20200209, Feb. 2021.

[22] G. U. Yule, "On a method of investigating periodicities in disturbed series, with special reference to Wolfer's sunspot numbers," *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, vol. 226, pp. 267–298, 1927.

[23] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Computation*, vol. 1, no. 2, pp. 270–280, Jun. 1989.

[24] K. P. Murphy, *Probabilistic Machine Learning: An Introduction*, ser. Adaptive Computation and Machine Learning. Cambridge, Massachusetts London, England: The MIT Press, 2022.

[25] S. Kolassa and W. Schütz, "Advantages of the **MAD**/mean ratio over the **MAPE**," *Foresight: The International Journal of Applied Forecasting*, vol. 6, pp. 40–43, 2007.

[26] J. H. Friedman, "Greedy function approximation: A gradient boosting machine." *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, Oct. 2001.

[27] L. G. Serrano and S. Thrun, *Grokking Machine Learning.* Shelter Island: Manning, 2021.

[28] A. Géron, *Hands-On Machine Learning with Scikit-Learn and PyTorch: Concepts, Tools, and Techniques to Build Intelligent Systems.* US: O'Reilly Media, 2025.

[29] T. Chen and C. Guestrin, "**XGBoost**: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowl-*

edge Discovery and Data Mining, ser. KDD '16. New York, NY, USA: Association for Computing Machinery, Aug. 2016, pp. 785–794.

[30] A. V. Dorogush, V. Ershov, and A. Gulin, "**CatBoost**: Gradient boosting with categorical features support," in *Workshop on ML Systems at NIPS 2017*, 2017.

[31] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "**LightGBM**: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., 2017.

[32] R. Sen, H.-F. Yu, and I. S. Dhillon, "Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting," in *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019.

[33] H.-F. Yu, N. Rao, and I. S. Dhillon, "Temporal regularized matrix factorization for high-dimensional time series prediction," in *Advances in Neural Information Processing Systems*, vol. 29. Curran Associates, Inc., 2016.

[34] Y. Qin, D. Song, H. Cheng, W. Cheng, G. Jiang, and G. W. Cottrell, "A dual-stage attention-based recurrent neural network for time series prediction," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, ser. IJCAI'17. Melbourne, Australia: AAAI Press, Aug. 2017, pp. 2627–2633.

[35] S. Makridakis and M. Hibon, "**ARMA** models and the **Box–Jenkins** methodology," *Journal of Forecasting*, vol. 16, no. 3, pp. 147–163, 1997.

[36] C.-C. M. Yeh, N. Kavantzas, and E. Keogh, "Matrix profile vi: Meaningful multidimensional motif discovery," in *2017 IEEE International Conference on Data Mining (ICDM)*, Nov. 2017, pp. 565–574.