MASA: Motif-Aware State Assignment in Noisy Time Series Data

Saachi Jain
saachi@stanford.edu
Stanford University

David Hallac
hallac@stanford.edu
Stanford University

Rok Sosic
rok@stanford.edu
Stanford University

Jure Leskovec
jure@stanford.edu
Stanford University

ABSTRACT

Complex systems, such as airplanes, cars, or financial markets, produce multivariate time series data consisting of a large number of system measurements over a period of time. Such data can be interpreted as a sequence of *states*, where each state represents a prototype of system behavior. An important problem in this domain is to identify repeated sequences of states, known as *motifs*. Such motifs correspond to complex behaviors that capture common sequences of state transitions. For example, in automotive data, a motif of "making a turn" might manifest as a sequence of states: slowing down, turning the wheel, and then speeding back up. However, discovering these motifs is challenging, because the individual states and state assignments are unknown, have different durations, and need to be jointly learned from the noisy time series. Here we develop *motif-aware state assignment* (MASA), a method to discover common motifs in noisy time series data and leverage those motifs to more robustly assign states to measurements. We formulate the problem of motif discovery as a large optimization problem, which we solve using an expectation-maximization type approach. MASA performs well in the presence of noise in the input data and is scalable to very large datasets. Experiments on synthetic data show that MASA outperforms state-of-the-art baselines by up to 38.2%, and two case studies demonstrate how our approach discovers insightful motifs in the presence of noise in real-world time series data.

CCS CONCEPTS

• **Computing methodologies** → **Cluster analysis**; **Motif discovery**; *Unsupervised learning*.

KEYWORDS

Noisy motif discovery, Temporal clustering, Multivariate time series

ACM Reference Format:
Saachi Jain, David Hallac, Rok Sosic, and Jure Leskovec. 2018. MASA: Motif-Aware State Assignment in Noisy Time Series Data. In *MileTS '19: 5th KDD Workshop on Mining and Learning from Time Series, August 5th, 2019, Anchorage, Alaska, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/1122445.1122456

MileTS '19, August 5th, 2019, Anchorage, Alaska, USA

ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

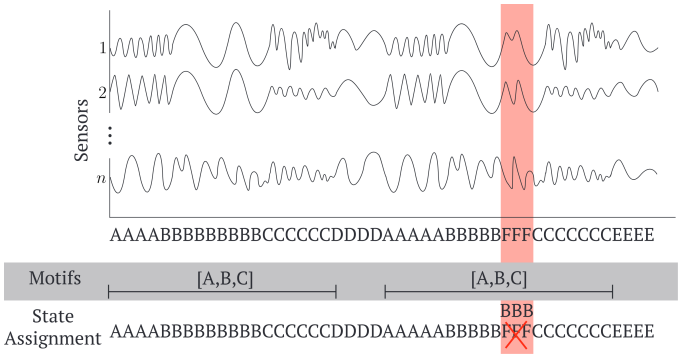

Figure 1: MASA assigns each measurement in a multivariate time series to a state. MASA further discovers motifs (repeated sequences of states), using them to improve state assignment in the presence of noise. Furthermore, the same motif ($[A, B, C]$) can spend different amount of time in each state (4 vs. 5 measurements in state A).

1 INTRODUCTION

Many domains and applications, ranging from automobiles [11], financial markets [8], and wearable sensors [28], generate large amounts of time series data. In most cases, this data is multivariate, where each timestamped measurement consists of a vector of readings from multiple entities, such as sensors. Thus, multivariate time series data captures system state via a sequence of sensor readings.

However, directly extracting meaningful insights from time series data is challenging, since the relationships between readings are often complex and mutated by noise. Not all sensors are important all the time, and key information may lie in the connection between readings across different timestamps rather than within individual measurements themselves. To understand this complex data, it is useful to label each measurement as one of K unique states. Each state is an interpretable template for system behavior which can repeat itself many times across the time series. These states distill the complexities of the multivariate dataset into a more accessible and interpretable symbolic representation. Moreover, this task allows us to partition the measurements in the time series into variable-length segments, where each segment is a sequence of measurements in the same state. Segmentation allows one to characterize large windows of time that exhibit a single behavior.

These states do not appear in isolation; the context in which a state occurs provides critical insights and can be just as useful as

the state itself. It is therefore important to discover *motifs*, which represent a sequence of state transitions. For example, in a car, a single state may indicate that the driver is slowing down, but the sequence of slowing down, turning the wheel, and then speeding up implies that the vehicle is in a motif known as a "turn". A motif is then an abstraction over a sequence of states in the time series. Motifs specify transitions but do not constrain duration: because there are many types of turns, a turning motif must not specify how long a driver slows down before turning, but only that deceleration occurred. Once learned, these motifs can be used to improve one's estimates of the states themselves. This context-aware motif-based assignment is especially useful for adjusting to noise in the data. For example, if a noisy measurement is (incorrectly) assigned to the wrong state, yet the sequence it is in is "close to" being an instance of a known motif, one can use this motif to re-assign the measurement to the correct state (Figure 1). Identifying an informative set of motifs is challenging because we cannot simply rank motifs by frequency: one seeks a set of motifs that are strong but are not redundant. Even ranking motifs is difficult, because one must consider both the significance of the motif pattern and the strength of the motif's instances in the dataset.

Discovering motifs in temporal datasets requires an unsupervised way of locating and labeling repeated behaviors. In general, as the states themselves are unknown, methods must simultaneously uncover the states, assign states to measurements, and identify the repeated motifs. Unlike standard time series segmentation techniques [10, 12], motifs repeat themselves many times across the time series. In contrast to clustering-only methods for state detection, many segments can be assigned to the same state [15, 27]. Additionally, existing model-based approaches for time-series motif discovery treat motifs as a final step and do not allow such motifs to influence and improve the robustness of the state assignment [18].

Here we introduce *Motif-Aware State Assignment* (MASA), a method for discovering noisy motifs in time series data which are used to robustly assign states to the underlying measurements. We optimize over three types of parameters: the first defines a model over K unique states, the second assigns each measurement to one of these states, and the third learns a set of motifs over the state assignment. Since these variables are combinatorial and coupled together in a highly non-convex manner, we solve the MASA optimization problem using an expectation-maximization (EM) type algorithm. Fixing our state model, we discover motifs and leverage them to robustly assign measurements to states. Then, fixing the state assignment, we update our state model. By iterating between these two steps, MASA uses motifs to shift the state models to include noisy measurements in order promote the repetition of global trends. During this process, MASA leverages novel techniques to scalably discover and curate a set of candidate motifs that are significant while minimizing redundancy.

We evaluate MASA on several synthetic and real-world datasets. First, we analyze how accurately MASA is able to label a time series with known ground-truth states. We compare our method with several state-of-the-art baselines, showing that MASA outperforms the best of the baselines by up to 38.2%. We further assess MASA's ability to identify planted motifs in the midst of irrelevant datapoints, and find that MASA can accurately isolate these motifs. We also examine how robust MASA is to the presence of noisy data, measuring how performance fluctuates with varying amounts of noise. Overall, we find that MASA outperforms existing methods in terms of accuracy in state assignment and robustness to noise in the dataset. We validate these results by repeating the experiment with real sensor data reported from subjects cycling on an exercise bike. Finally, we perform two real-world case studies by applying MASA to sensor data from both aircrafts and automobiles. We show that MASA discovers interesting and interpretable motifs that occur in flight and driving sessions.

Further Related Work. Recent approaches to time series clustering and segmentation often use distance-based metrics to identify different states [15]. Additionally, they apply dimensionality reduction [17, 18] or rule based approaches [7, 16] to identify symbolic representations of time series data. However, distance-based methods have been shown to be unreliable in certain cases [14], and lose the interpretability of multivariate data-points. Model-based methods, such as TICC [11], ARMA [29], Gaussian mixture models [3], and hidden Markov models [27], represent states as clusters using probabilistic models, and often can more accurately encode the relationships between sensors and the true underlying states. However, these existing methods have no way of incorporating motifs into their models.

Motif discovery is a common problem in time series data analysis [6]. Methods for finding motifs include random projection [4] and suffix arrays [26]. Some of these methods are event rather than numerically based and thus bypass the simultaneous problem of state assignment [23]. Most of these methods assume motifs of fixed length [9]. Some methods also use distance metrics [30]. The problem of finding repeated patterns also appears in the field of computational genomics. ACME uses a combinatorial approach to find super-maximal motifs in DNA [26]. Other bioinformatics models use edit-distance approaches to find motifs that vary slightly in appearance over the course of the sequence [22]. MASA departs from these methods in two respects. Firstly, MASA allows subsequences within a motif to have variable length while maintaining a given state. Moreover, unlike uniform scaling approaches [30], MASA allows each state within a motif to scale independently. Secondly, MASA iterates by using the motifs to re-assign the original measurement to the updated states, encouraging noisy sequences to match a given motif. This iteration allows for stronger motifs and a more robust state definition.

2 PROBLEM OVERVIEW

Our input is a sequence of T measurements, $\mathbf{X} = X_1, ..., X_T$, where each measurement $X_t \in \mathsf{R}^N$ is a vector of data values observed at time t. Our goal is to discover frequently occurring high level patterns, called motifs, in the input data. A key component of our approach are K state models which represent K different states, where each state captures the properties of similar measurements. We utilize these models to assign a state to each measurement and use the resulting state assignment to discover motifs.

Motifs. Our aim is to discover motifs that identify significant recurring and length-varying patterns in time series data. We assume that these patterns contain time consecutive measurements. Given a sequence of consecutive measurements, we define a *motif* as a sequence of corresponding state assignments, where all neighboring

occurrences of the same state are merged into one. To illustrate, a car turn can be viewed as three states: (A) slowing down, (B) turning, and (C) speeding up. Although in a given instance of the turn, each of these states might have a different duration, the "turn" motif is represented by $[A, B, C]$. Thus, states in the motif are ordered but the number of consecutive occurrences (*i.e.*, duration) of each state may vary between instances of the motif.

Since we are interested in commonly occurring motifs, we represent each motif by the pair (m, q), where m is the motif and q is an associated list of motif instances. A motif instance indicates one occurrence of the motif in the dataset.

We introduce the following motif constraints in our method:

(1) A motif m must contain at least 3 states: $|m| > 2$.
(2) A motif m must appear at least L times: $|q| \geq L$.
(3) Motif instances cannot overlap: each measurement can only belong to at most one motif.

We motivate the first constraint as motifs with two or fewer states are not very informative beyond the clustering itself. The second constraint aids our runtime: we do not want to spend time on investigating motifs that are not frequent. Since we are only interested in frequent patterns, we do not require that every measurement belongs to a motif:

MASA Problem Setup. Overall, MASA seeks to solve for a state model Θ, an assignment of states to measurements $\mathbf{S}$, and an assignment of motifs to measurements $\mathbf{M}$ to optimize the following objective (subject to motif constraints in the previous section):

$$\max_{\Theta, \mathbf{S}, \mathbf{M}} \sum_{i=1}^{T} \log P_\Theta(X_i \mid S_i) - \beta \mathbb{1}\{S_{i-1} \neq S_i\} + \log \gamma \mathbb{1}\{S_i < \mathbf{M}\} + \Psi(\mathbf{M}) - R(\Theta).$$

Here, X_i is the measurement at time i, S_i is its assigned state, and the probability $P_\Theta(X_i \mid S_i)$ is defined by our state model. The β term is a hyperparameter that encourages neighboring measurements to be assigned the same state. The γ parameter, $0 \leq \gamma \leq 1$, defines the cost of not assigning a measurement to a motif instance. Lower values of γ indicate a more aggressive penalty for a measurement not conforming to any motif. The term Ψ is a scoring metric quantifying the strength of our motifs based on their occurrences in the dataset. $R(\Theta)$ is a regularization penalty on our state model parameters Θ.

State model. MASA is agnostic to the specific parameterization Θ of the state model $P_\Theta(X_i \mid S_i)$, which models the likelihood of an observation given a state.

MASA requires the following operations:

- $P_\Theta(X_i \mid S_i)$: Distribution of how likely the measurement X_i is observed given that it belongs to the state S_i.
- UpdateStatesModel($\mathbf{S}$): Once the states are assigned measurements, the state model parameterized by Θ must be updated to maximize the likelihood of the measurements conditioned on the state assignment $\mathbf{S}$. UpdateStatesModel optimizes this objective: $\max_\Theta \sum_{i=1}^{T} \log P_\Theta(X_i \mid S_i) - R(\Theta)$.
- ProposeAssignment(Θ): Returns the state assignment $\mathbf{S}$ optimizing the non-motif objective: $\max_{\mathbf{S}} \sum_{i=1}^{T} \log P_\Theta(X_i \mid S_i) - \beta \mathbb{1}\{S_{i-1} \neq S_i\}$.

Given a state likelihood model P_Θ, we note that ProposeAssignment can typically be implemented via the Viterbi algorithm as in [19]. Thus, any potential state model can be used, which allows MASA to be applied to diverse types of data, from heterogeneous exponential families to categorical distributions [24].

In this paper specifically, we define our state model using the Toeplitz Inverse Covariance-based Clustering (TICC) model [11]. TICC defines each state by a block Toeplitz Gaussian inverse covariance matrix $\Theta_k \in \mathbb{R}^{N \times N}$ with empirical mean $\mu_k \in \mathbb{R}^N$. The probability $\log P(X_i \mid S_i)$ is then

$$\log P(X_i \mid S_i) = -(X_i - \mu_{S_i})^T \Theta_{S_i} (X_i - \mu_{S_i}) + \log \det \Theta_{S_i}, \quad (1)$$

and $R(\Theta)$ is an ℓ_1 penalty on the inverse covariance matrices Θ_k. UpdateStateModel is then simply a single iteration of TICC, and ProposeAssignment uses the Viterbi algorithm to find the most likely sequence of states.

Hyperparameters. Overall, our MASA algorithm contains four hyperparameters: K, the number of states, L, the minimum number of instances for a motif to be considered valid, β, the switching penalty, and γ, the aggressiveness of how strongly we encourage measurements to conform to a motif. All MASA hyperparameters can be pre-defined by the user, tuned by hand, or chosen more systematically using a method such as BIC. For our experiments, we chose L to be a set, consistent number ($L = 10$) which gives us a reasonable runtime, and picked K and β via BIC; we discuss hyperparameter tuning for our experiments in more detail in Appendix B. We discuss how γ can be chosen in Section 8.1. The values of each of the hyperparameters for each of our experiments can be found in the appendix.

3 MASA ALGORITHM

Our problem is non-convex, and solving for a global optimum is intractable. However, we solve this objective via an iterative EM-like alternating maximization approach. In the E-step, we assign states to measurements and discover motifs which we then leverage to robustly update state assignment. In the M-step, we then use the updated state assignment to recalculate the state probability model by updating Θ. Upon convergence, state assignments $\mathbf{S}$ and identified motifs $\mathbf{M}$ are provided as method results. MASA thus iteratively refines the state models, state assignments, and motifs, which allows for a robust motif discovery in the presence of noise.

The broad outline of our method is described below, and we give further details in the following sections:

Initialize Θ. Initialize the state model Θ in a reasonable manner. We initialize by alternately calling UpdateStateModel($\mathbf{S}$) and ProposeAssignment(Θ) until convergence.

E-Step: Compute $\mathbf{S}$ and $\mathbf{M}$. Use the state model to compute a motif-aware state assignment. This step proceeds in two phases.

E-Step A: Discover candidate motifs. Use ProposeAssignment(Θ) to compute a preliminary state assignment $\mathbf{S}$, then discover repeated patterns in $\mathbf{S}$ and generate a set of candidate motifs (Section 4).

E-Step B: Reassign states to measurements using motifs. Using the candidate motifs, reassign the states to measurements in order to match discovered motifs. If a measurement does not belong to a motif, a user-defined cost γ is imposed. We use a hidden Markov model (HMM) to model each motif separately, finding sequences of measurements that may conform to each motif. Since each measurement can only belong to a single motif, we aggregate the HMM output to obtain a state assignment $\mathbf{S}$ while maintaining the constraints

described in Section 2. This step gives us a new state assignment S as well as a set of motifs **M** (Section 5).

M-Step: Use S to recompute Θ. As the state assignment has changed, use UpdateStatesModel(S) to update the state model Θ and then jump back to the E-Step.

We can either repeat the E and M steps of MASA for a set number of iterations, or stop when MASA has converged. MASA converges if the state assignment S returned from E Step is the same as the state assignment returned by ProposeAssignment(Θ) on the updated Θ from the M Step. In that case, the optimal state assignment S according to the state model is the same as the one suggested by the motifs, which means that our model reflects our motif preferences.

Next we give more details about each of the step of our algorithm.

4 E-STEP A: DISCOVER CANDIDATE MOTIFS

Given Θ, we compute a preliminary assignment of states to measurements by setting S = ProposeAssignment(Θ). Using letters to indicate states, we can view S as a string, where each letter denotes a state assigned to a measurement. Because our motifs are independent of the number of adjacent equal states, our method operates on S′, which is obtained from S by collapsing consecutive duplicate letters into a single letter. For example, a state assignment S = [A,A,B,B,B,B,C,C,C,C] gets collapsed to S′ = [A,B,C].

Identify Repeated State Sequences. We first seek to find all maximal repeated subsequences of states in S′. A maximal subsequence is defined as a sequence that cannot be extended to either the left or right without changing the set of occurrences in S′ [26]. We require that each repeated subsequence has at least L non-overlapping instances in S′. We use a suffix array to solve this problem efficiently [25].

Select Candidate Motifs. Each repeated subsequence of states in S′ is a motif candidate. We need to select the relevant candidates for the final candidate set. We cannot simply select the motifs by frequency because an ideal motif candidate should also be a novel addition to the set. The motif $[A, B, C, D]$ may be largely useless if we already accepted $[A, B, C]$. We propose a dynamic way to select motif candidates by assessing each new candidate against a null model with knowledge of all previously accepted candidate motifs.

Motif occurrence probability. According to the null model, let the probability of observing an instance of the motif m be the probability of independently observing each of m's states m_i according to their empirical frequency in S′. If there are N_m instances of the candidate motif in S′, then the p-value of m is the probability of at least N_m occurrences of m appearing according to our null model under a binomial distribution [5]. We then select motifs that exceed a required threshold α. However, this method alone does not protect against redundancy: we want to consider a candidate motif relative to the other candidates we have accepted. To do so, we define a set $\mathcal{D}$ as the set of candidate motifs that the null model "already knows" about.

Dynamic candidate selection. We assess each candidate motif m in order of increasing length. If any candidates $d \in \mathcal{D}$ appear as a substring of m, we replace that substring in m with a single state that appears (according to the null model) with d's empirical probability in S′. We can then evaluate the probability according to the null model above. Intuitively, the null model will perform better given knowledge of the candidate sets versus a null model which still treated each state as independent. We elaborate on implementation details of this algorithm in Appendix A.

Let $P_\emptyset(m \mid \mathcal{D})$ be the probability of motif m according to the null model with knowledge of $\mathcal{D}$. If for candidate motif m

$$P\ \ \mathcal{B}(|\mathrm{S}'|, P_\emptyset(m \mid \mathcal{D}) \geq N_m\ \ \leq \alpha,$$

for threshold α, we accept m as part of our candidate set, and add m to $\mathcal{D}$ for evaluation of the next candidate. After this step, we have collected a novel and relevant set of candidate motifs, which we leverage in the next step to robustly assign states to measurements.

5 E-STEP B: USING MOTIFS TO ASSIGN STATES

Having collected a set of candidate motifs, we next seek to assign a state to each measurement in order to promote these candidate motifs. Simultaneously, we decide on a final set of motifs **M** that fit the constraints laid out in Section 2. This step proceeds in 4 phases.

Identify New Motif Instances. In this step, we identify possible instances of each candidate motif in our measurements. We specifically are looking for *noisy* instances of the motifs which could not be found from the symbolic output of the non-motif optimization ProposeAssignment(Θ). For example, if under our current state model X_i is most likely in state j, but assignment to state k would allow for completion of a motif with only a minimal effect on our likelihood objective, we should assign X_i to k instead of j.

Jointly considering all of our candidate motifs from E-Step A when re-assigning our measurements is both memory-expensive and slow. Instead, we consider a single candidate motif at a time, and re-assign all measurements according to this one motif to find a complete set of possible instances for this motif in the dataset. This approach simplifies our model and allows us to parallelize re-assignment by modeling every candidate motif concurrently.

For each candidate motif m, we define a time-varying hidden Markov model (HMM) to model the entire sequence of measurements **X**. Figure 2 shows an example of our HMM for $m = [A, B, C]$. The HMM has a hidden state z_i for each state m_i in m. Our emission probabilities for these hidden states are then $\log P(X_i \mid z_j) = \log P_\Theta(X_i \mid m_j)$. We further create a "non-motif" hidden state z_0, signifying that the measurement should not be assigned to this motif. We formalize z_0 as the original state assignment S_i, discounted by the non-motif penalty $\gamma < 1$. The emission probability for measurement X_i from the non-motif state is then $\log P(X_i \mid z_0) = \log P_\Theta(X_i \mid S_i) + \log(\gamma)$.

MASA prioritizes assignments of measurements to motif states even if a different assignment is more likely. γ encapsulates how much less likely the motif state for a measurement X_i can be while still being picked over the "most likely" sequence according to X_i's values. A $\gamma = 1$ would assign all measurements to z_0, since the non-motif state already represents an optimal sequence. A lower γ such as $\gamma = 0.6$ will be more aggressive, exploring the motif hidden states even when the likelihood might decrease.

Motif hidden states can transition only to themselves or into the next motif hidden state: In Figure 2, z_1 can transition to z_2 with penalty β, the state switching penalty. However, z_1 cannot directly transition to z_3. The non-motif hidden state can switch into itself

or start a motif instance by switching into z_1; however, once z_1 has been entered, the only way to return to the non-motif hidden state is by finishing the motif. The last hidden state (z_3) can either start another motif by switching to z_1 or enter the the non-motif z_0.

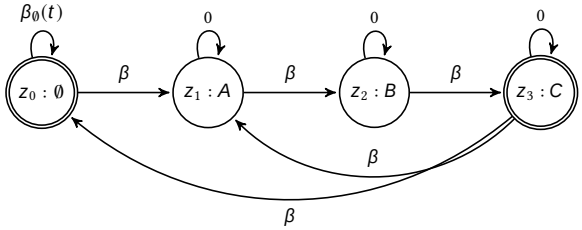

Figure 2: HMM for a motif $[A, B, C]$. Edges represent transition costs; z_1, z_2, z_3 are motif states, while z_0 represents measurements that are not part of a motif.

States do not incur a switching penalty when switching to themselves. The notable exception is the non-motif state, where we define the penalty of looping in the non-motif state using $\beta_\emptyset(t)$:

$$\beta_\emptyset(t) = \beta \mathbb{1}(t \ , \ 1 \wedge S_{X_{t-1}} \ , \ S_{X_t})$$

$\beta_\emptyset(t)$ thus adds a cost of β when transitioning from z_0 to itself if, in our original $\mathbf{S}$, this measurement also incurred a switching penalty. Both z_0 and the first motif hidden state (in our example in Figure 2, z_1) are valid starting states. z_0 and the last motif state (z_3) are the only valid end states.

With our HMM fully defined, we use the Viterbi algorithm, a dynamic programming method, to find the most likely sequence of hidden states for these measurements [19]. Any X_i which are in the motif states represent a possible motif instance for m. Thus, after performing this step, we have a list of possible instances for every candidate motif m.

Scoring Motif Instances. Since we build a separate HMM for each motif in the previous phase, it is possible for a single measurement to appear in multiple motifs, violating the restrictions in Section 2. Therefore, it is necessary to develop a way of ranking motif instances and assigning each measurement to only one motif. Each instance has two factors influencing its significance: the importance of the motif itself and the individual likelihood of the measurements.

Motif score. Let N_m be the number of instances of motif m identified via the HMM. We again define our null model under the assumption that each state in the motif occurs independently based on the empirical frequency counts from $\mathbf{S}'$ from Section 4. If $\mathsf{E}_\emptyset[N_m]$ is the expected number of instances of m according to the null model, then we define the motif score as the G-score [1]:

$$\Upsilon(m) = 2N_m \left(\log N_m - \log \mathsf{E}_\emptyset[N_m]\right).$$

Instance score. We now formalize the second component of our scoring metric, which evaluates an individual instance of a motif candidate. Let $X_i, \ldots, X_j$ be a consecutive sequence of measurements which were identified as a possible instance of the motif m by our HMM, where S_i is the state assignment of X_i according to $\mathbf{S}$ and $S_i^{(HMM)}$ is X_i's new assignment according to the motif model. Then we compute the instance score as the log ratio between the likelihoods of the two assignments [13]:

$$\Delta(m, (X_i, \ldots, X_j)) = \sum_{k=i}^{j} 2 \log P(X_i \mid S_k^{(HMM)}) - \log P_\Theta(X_i \mid S_k).$$

Then the total score for a motif instance is:

$$\Psi(m, (X_i, \ldots, X_j)) = \Upsilon(m) + \Delta(m, (X_i, \ldots, X_j)).$$

And our total motif score for $\mathbf{M}$ is:

$$\Psi(\mathbf{M}) = \sum_{(m;q) \in \mathbf{M}} \Upsilon(m) + \sum_{i=1}^{|\Theta|} \Delta(m, q_i).$$

Algorithm 1: Greedy Motif Assignment

```
Sort instances by Ψ; initialize X_1, ..., X_T open;
foreach motif instance (m, q) do
    if any X_i covered by q is locked then
        reject q and continue;
    if m is complete then
        Lock all measurements covered by q;
        Remove other bids on those measurements;
    else
        Place bid on all measurements covered by q;
        if # instances bid by m ≥ L then
            Mark m as complete;
            Lock all of m's current bids;
            Remove other bids on those measurements;
Retrieve motif instances that are permanently locked;
Construct and retrieve M;
```

Update State Assignment. Having curated a set of motif instances and developed a scoring methodology, we now update the state assignment $\mathbf{S}$. We do this by allocating each measurement X_i to either a single motif or no motif. If a measurement is assigned to motif m, then it takes its state according to m's HMM. If a measurement is not assigned to any motif, it keeps its old state from the old $\mathbf{S}$. While performing this process, we need to uphold the constraints in Section 2.

After sorting all of the found motif instances from all motifs according to Ψ, we greedily allocate measurements to motifs using a system of locks and bids (Algorithm 1). Each measurement can be either "locked" or "open". Initially, all measurements are open. A measurement becomes locked when it is permanently assigned to a motif. We enforce a constraint that multiple motifs can place a bid on an open measurement, but a locked measurement can only belong to a single motif. Only motifs that have at least L instances can lock measurements; we label such motifs as "complete".

After processing all motif instances in Algorithm 1, we only accept assignments of locked measurements to motifs. Since only complete motifs can lock, only motifs with at least L instances will appear in the final set of motifs. This lock/bid scheme further ensures that no X_i is assigned to multiple motif instances. Any measurement that does not belong to an accepted motif instance is set to its original assignment in $\mathbf{S}$.

We thus have a completely new assignment of measurements to states $\mathbf{S}$. We further construct $\mathbf{M}$ as the set of complete motifs.

6 M-STEP: RECOMPUTE STATE MODEL

By creating a new S, we have shifted the pool of measurements assigned to each state to include additional measurements based on motif assignment. We thus now use UpdateStatesModel(S) to recompute Θ.

We then check for convergence. Using our recomputed likelihood model, if ProposeAssignment(Θ) returns the same state assignment S that was returned from the E-Step, then our states have stabilized and MASA has converged. Otherwise we pass our updated Θ to the E-Step from the M-Step and repeat.

7 CONVERGENCE AND RUNTIME OF MASA

MASA converges when recomputing our state model Θ does not change our state assignment. MASA can alternatively be capped at a set number of iterations. The number of iterations that MASA needs to converge is dataset dependent, but typically is on the order of tens of iterations. The runtime of each iteration of MASA is:

E-Step A: Motif Discovery. For a time series of length T and minimum motif length L, we identify at most T/L patterns from the suffix array. Let C be the number of eventually accepted motif candidates. Since construction of the suffix array takes linear time, generation of motif candidates takes worst case time $O((T/L)C + T)$. However, $C \ll T$ since motif construction occurs from the collapsed form S' and is further constrained by the significance threshold α. We can optionally cap C by truncating the set according to p-score. This step thus takes linear time in T.

E-Step B: Motif assignment and scoring. Assuming any motif candidate has at most r states defining the motif sequence, each HMM takes $O(rT)$ time due to the chain structure of the HMM. We can identify instances for each motif candidate in parallel. Scoring the instances thus takes constant time per motif instance. In the worst case for Algorithm 1, each motif has instances that cover all T measurements. In such a scenario, each motif candidate can only bid on a measurement once, so the algorithm can take worst case $O(CT)$. Practically, however, this is a loose upper bound since such a scenario would indicate an extremely redundant candidate set and would not pass through the filter in E-Step A. Thus, this algorithm takes runtime linear in T.

M-Step: Recompute State Model. Finally, since UpdateStateModel is also linear in T, M-Step also takes linear time in T.

Thus, in total, each iteration of MASA takes *linear time* in T, which we further verify experimentally in Section 8.1.

8 EVALUATION

We performed an extensive evaluation of MASA's performance on a range of scenarios and different parameter values. Specifically, we seek to measure MASA's ability to identify common motifs in the presence of noise, as well as its accuracy in assigning points to the correct underlying states. For evaluations in this section, we initially use synthetic datasets, since they provide a known ground truth which we require for comparing MASA with state-of-the-art baseline methods. We then validate these results on real cycling data with planted motifs. Later, in Section 9, we apply MASA to two real-world case studies.

To run the experiments, we built a MASA solver that implements the algorithm described in Section 3. A link to our solver can be found in the footnote.[1]. Given a multivariate time series dataset our solver returns the ranked motifs, the motif assignments, and the state assignments for each measurement.

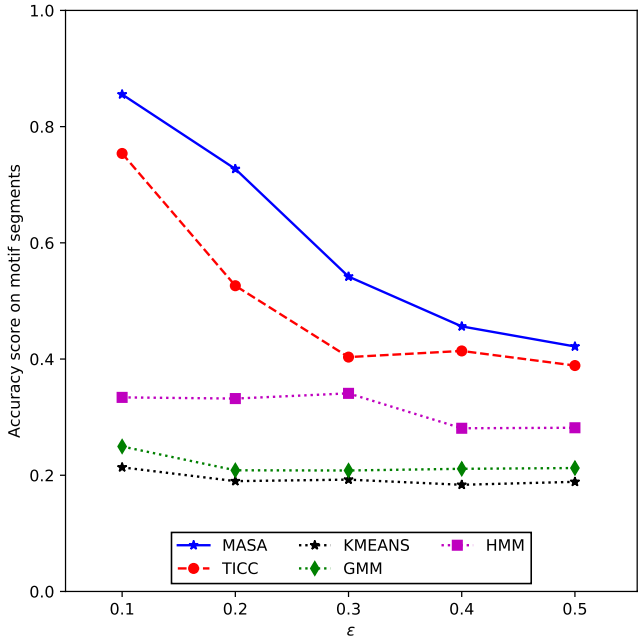

Figure 3: MASA and baseline accuracy scores on motif sections for synthetic data with varying levels of ϵ.

8.1 Experiments on Synthetic Data

Synthetic Dataset Generation. We generate our dataset as follows: Our ground-truth synthetic time series has a total of 150,000 measurements, each measurement being in $\mathbb{R}^5$. Measurements are taken from K states; we use $K = 10$ in our experiments. We first assign states to measurements and then generate specific data values for each measurement.

To create the assignment, we compose the time series from 1,000 "macro-segments", each containing 150 measurements. Each macro-segment begins with 6 "non-motif" segments of 15 measurements, where measurements within each segment are assigned randomly to one of K states. All measurements in one "non-motif" segment are assigned the same state. These non-motif segments are followed by 4 "motif" segments of 15 measurements each with states $A \rightarrow B \rightarrow C \rightarrow D$. Using these ground truth assignments, we form random ground truth covariances $\Sigma_1, ..., \Sigma_K$ for each state through the method described in [20]. Each data-point from ground truth state k is then drawn from a multivariate normal distribution with zero mean and covariance Σ_k as in [11].

We introduce noise into our segments as follows. For every segment, with probability ϵ, we perturb measurements in that segment by a random non-motif state. For example, suppose a segment with a ground truth state i is set to be perturbed. We pick a random state $j < \{A, B, C, D\}$ (the non-motif states). Rather than using Σ_i to generate data, we draw from a distribution with covariance

[1] Our MASA solver and all code from the experiments in this section are available for download at https://github.com/snap-stanford/masa.git.

Table 1: Weighted F_1 scores on motif states A, B, C and D as a function of noise ϵ. Notice that MASA performs best across all noise levels.

ϵ	0.1	0.2	0.3	0.4	0.5
MASA	**0.759**	**0.719**	**0.518**	**0.464**	**0.440**
TICC	0.736	0.599	0.419	0.462	0.436
KMEANS	0.217	0.208	0.198	0.192	0.202
GMM	0.234	0.185	0.190	0.191	0.219
HMM	0.252	0.269	0.283	0.256	0.263

$0.7\,\Sigma_j + 0.3\,\Sigma_i$, such that the new segment appears as if from the random state.

Robustness to Noise. The synthetic dataset plants motifs with noisy segments. While other algorithms might struggle with these noisy segments, MASA can leverage the motif structure to smooth out the state assignment of these perturbed segments.

To evaluate the effectiveness of MASA in noisy data, we create multiple datasets using the above method, each with a different ϵ value. A larger value of ϵ indicates greater noise in the data, since more measurement values will be perturbed. We run MASA at $\gamma = 0.8$ (later in this section, we evaluate the sensitivity of our results to the selection of γ) against TICC and the following baseline models: a Gaussian mixture model [3], K-means with Euclidean distance, and a hidden Markov model with Gaussian emissions.

Figure 3 shows the results from measuring the accuracy of these models on the motif sections of the dataset against the ground truth. The accuracy quantifies the probability that a given measurement was classified into the correct state. As shown, MASA classifies these measurements with higher accuracy than any of the other baselines at all values of ϵ, outperforming TICC (the second-best performing method) by up to 38.2%. Interestingly, as ϵ rises initially up to about 0.3, MASA outperforms TICC by an increasing margin, which means that MASA is able to successfully utilize context information to correct state assignments.

We further compare methods on their ability to assign measurements to ground-truth motif states. In Table 1, we evaluate the F_1 scores for states A, B, C, and D over the full dataset, including times where they are assigned in the non-motif sections. These scores are weighted by the support of these states within the ground truth dataset. MASA continues to significantly outperform TICC and other baselines, especially when the amount of noise is reasonable ($\epsilon < 0.4$). This shows how MASA can optimize accuracy on the motif sections while correctly ignoring the non-motif sections of the dataset.

γ-robustness. We evaluate robustness to the parameter γ, which encapsulates the aggressiveness of "forcing" measurements to follow motifs. Smaller γ's encourage sequences to conform to known motif patterns, even if they are not a perfect fit. Higher values of γ only encourage motifs to exist when there is a near-perfect alignment. We run MASA with $\epsilon = 0.2$ and plot the weighted F_1 score on the motif states A, B, C, and D (including times where they are assigned to non-motif sections, similar to Table 1) in Figure 4 (top). We see that MASA is robust to the selection of γ, obtaining a weighted F_1 score of above 0.69, the score that TICC achieves, for all $0.4 < \gamma \leq 0.99$. Because the initial motif filtering step only permits the algorithm to pursue plausible motifs, even low values of γ (which are overly aggressive) perform reasonably well because there are very few "incorrect" motifs that the dataset can be wrongly assigned to. Overall, this shows how MASA can discover accurate trends even if the aggressiveness parameter γ is not perfectly tuned to its optimal value.

Table 2: State assignment accuracy scores on motif sections for cycling data.

	MASA	TICC	KMEANS	GMM	HMM
Accuracy	**0.830**	0.741	0.614	0.812	0.710

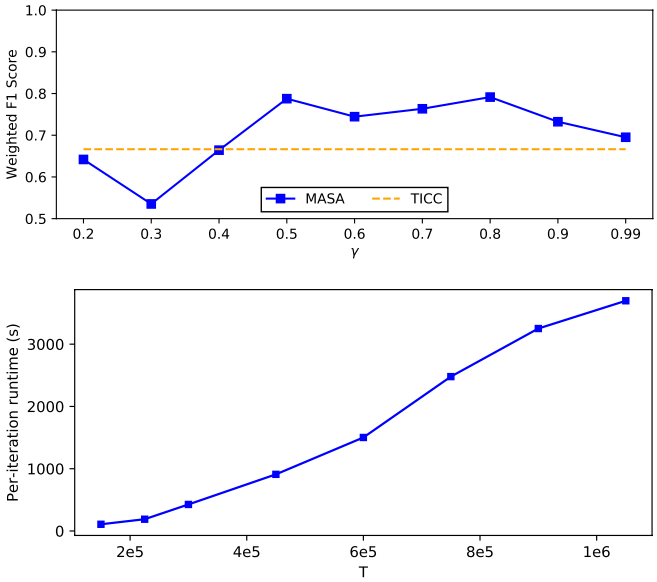

Figure 4: Top: Weighted F_1 scores on motif states for varying γ. Here, all values of γ above 0.4 outperform TICC's F_1 score. Bottom: MASA per-iteration runtime for a synthetic dataset of varying lengths. Our solver scales linearly for T, the number of timesteps.

Scalability. Since time series datasets can be extremely long, the number of measurements T dominates the runtime of MASA. We empirically measure the per-iteration runtime of MASA in Figure 4 (bottom) on synthetic data with $\epsilon = 0.2$ for increasing numbers of measurements. For consistency between measurements, we cap our number of motifs to 25 as described in Section 4. We find that the growth in runtime empirically increases linearly with respect to T, solving a dataset of 1.05 million measurements in 3,700 seconds.. Thus, MASA can scalably handle long time series datasets, and can solve an iteration for a dataset of over 1 million measurements in approximately one hour.

8.2 Experiments on Cycling Data

To conclude our evaluation, we evaluate MASA's performance on planted motifs on real cycling data. We use the publicly available Daily and Sports Activities dataset [2]. In this dataset, eight subjects cycled on an exercise bike, with 45 sensor values being recorded at 25 Hz. The dataset was broken down for each cyclist into five-second runs, or 125 sample long segments, where each of the eight

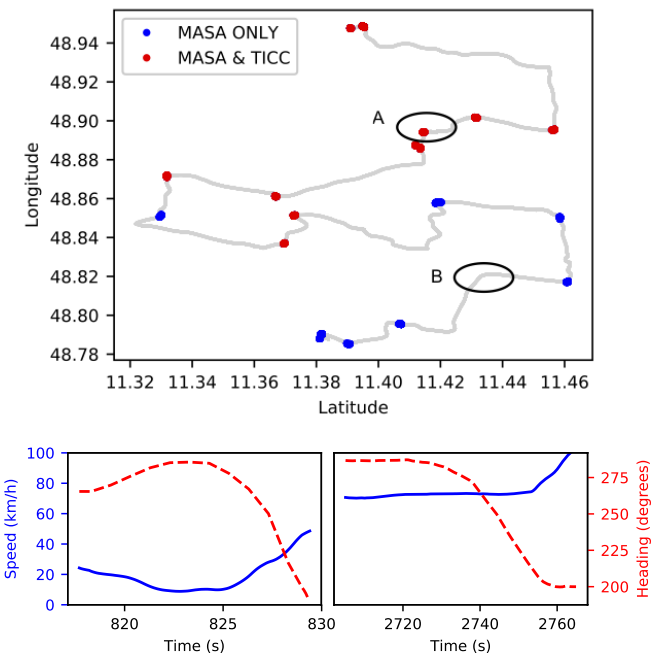

Figure 5: Top: Driver path over one session. Instances belonging to the "turn" motif are highlighted. Notice MASA identifies many more turns than TICC. Bottom: The speed/heading during Turn A (Left) and Turn B (Right).

cyclists generated 60 samples, for a total of 480 five-second segments in the dataset. We seek to evaluate MASA's performance on identifying the cyclist given a sample of data.

We plant motifs in a manner similar to the synthetic dataset above. We randomly sample six segments (each of length 125), choosing with equal probability any of the 480 samples, as the "non-motif" section. For the motif, we use the motif [A, B, C, D], with each state representing the first four cyclists in the dataset. To generate the motif, we draw one segment from each of these four cyclists in order (randomly picking one of that subject's 60 samples), and concatenate the 125-sample segments to form the motif [A, B, C, D]. We perform this procedure 100 times for a total of 125,000 datapoints. We then run MASA and evaluate accuracy scores on the motif sections, as in Figure 3. We find that MASA outperforms all other baselines (Table 2) with an accuracy score of 0.830 as compared with TICC's 0.741. We cover more specific details on this experiment in our supplementary section.

9 CASE STUDIES

Here, we run MASA on two real-world examples, using automobile and aircraft sensor data to demonstrate how our approach can be applied to discover insightful motifs.

Automobile Sensor Data. We first evaluate our algorithm on a multivariate car sensor dataset, provided by a large automobile company. Data from 7 sensors was collected every 0.1 seconds over an hour of driving data on real roads in Germany: brake pedal position, acceleration (X-direction), acceleration (Y-direction), steering wheel angle, speed, engine RPM, and gas pedal position.

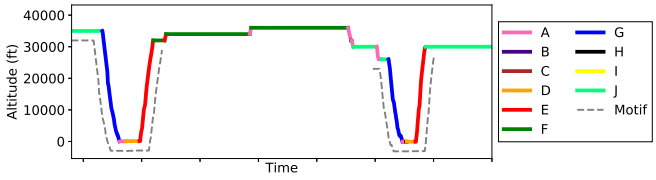

Figure 6: Airplane altitude over a 10 hour interval. The plot is colored according to its state assignment, and the specific motif instances are underlined.

We run MASA with 8 states on this dataset, and the results immediately identify a significant three stage motif that occurs 19 times during the one-hour session. After plotting the GPS coordinates of each sensor reading in Figure 5, we see that this motif corresponds to the driver turning the vehicle. Specifically, the three states appear to correspond to "slowing down", "turning the wheel", and "speeding up". We then searched for this sequence of states in the TICC state assignment: we found that only 11 of the 19 turns (marked in Figure 5) identified by MASA conformed to the motif pattern. Moreover, the turn pattern did not otherwise appear in TICC's assignment. In these other cases, some noise in the sensor readings led TICC to assign one (or more) of the measurements in the turn to an incorrect state. However, using the turn motif, MASA is able to correct for such noise and identify the sequence as a turn in a completely unsupervised way.

There are some bends in the driving path which MASA does not identify as a turn. These are due to significant deviations in the "typical turn" identified by the motif. Figure 5 depicts the speed and heading for the car during the section labeled A, which MASA identified as a turn. In this section, the car slows to 10 km/h during the turn before speeding back up. In contrast, during "turn B", the car maintained a speed of at least 70 km/h during the entirety of the turn. This stretch of road occurred on a highway, so even though the heading of the vehicle changed, this section of road did not conform to a classic three-stage "turn" motif, and thus was not classified as a turn by MASA.

MASA can thus identify directly interpretable segments of interest in an automobile dataset. Compared to state-of-the-art methods like TICC, MASA can intervene to more robustly assign individual measurements to states to make common behaviors, such as turns, appear uniform throughout the dataset in the presence of noise.

Airplane Sensor Data. We next analyze a dataset, provided by a large manufacturer, containing data from a single commercial aircraft over the course of several months. This multivariate dataset contained 1,459 sensors collected over 85 total flights, where data was sampled every 10 seconds. For computational scalability, we embed each 1,459-dimensional measurement in a low-dimensional vector using principal component analysis. Here, we pick a value where the eigenvalues store 99% of the cumulative energy, which yields a vector in $\mathbb{R}^{13}$. We then run MASA with 10 states.

Labeling each state from A through J, MASA identifies the top motif as $[J, G, A, H, D, E]$ (Figure 6). Plotting this motif against the airplane's altitude in Figure 6, we see that the motif encompasses the landing stages of each flight and the take-off stages of the next flight. This motif is extremely consistent, occurring almost every time the plane lands for one flight then takes off for the next (since

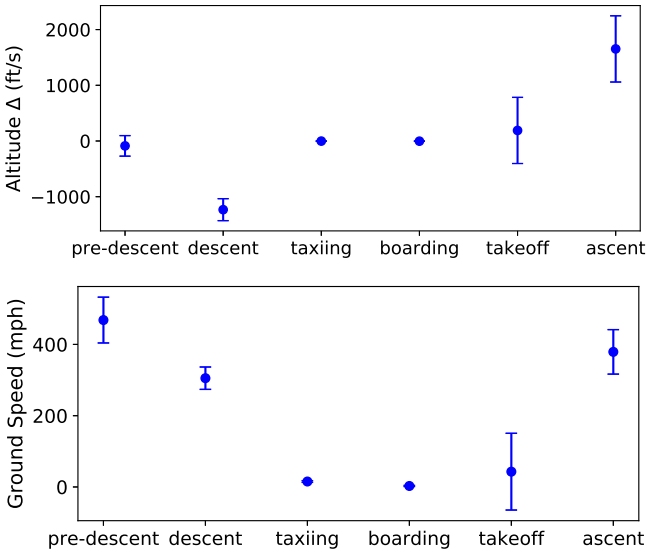

Figure 7: Average altitude change (top) and ground speed (bottom) of each segment of the top motif (J, G, A H, D, E) for all instances in the dataset. Error bars display one standard deviation.

we do not care about the length of each segment, the layover time does not effect the presence of this motif). We further characterize the plane's behavior by finding the average altitude change and ground speed for each of the six segments in the motif across all instances of the motif in the dataset. Using the velocities reported in Figure 7, we can interpret each stage of the identified motif as:

- **Pre-descent**: Equilibrium altitude, high ground speed.
- **Descent**: Slower ground speed with negative velocity indicating decrease in altitude.
- **Taxiing**: No vertical velocity and very slow ground speed.
- **Boarding**: Plane at rest.
- **Takeoff**: Increased positive vertical velocity and ground speed.
- **Ascent**: High positive vertical velocity and ground speed indicating ascent.

We note that MASA discovered this motif in a fully unsupervised manner. As shown, it was able to isolate an interesting repeated heterogeneous sequence of behaviors found across the time series. This has many practical benefits, as it can be used to auto-label the data with the different "stages" of a flight. Discovering motifs allows us to identify how these stages progress, and also lets us better label these stages when there is noise in the readings (due to turbulence, faulty sensors, or exogenous factors such as the weather).

10 CONCLUSION AND FUTURE WORK

In this paper, we have developed a novel method for discovering motifs in time series data. Our approach, Motif-Aware State Assignment (MASA), leverages these motifs to better characterize and assign states, discovering repeated segments and relevant trends. The promising results on both the synthetic experiments and case studies imply many potential directions for research. We leave for future work the analysis of MASA with different underlying likelihood models, rather than only TICC, the model we used in this paper. Furthermore, extending MASA to account for segment length, similar to hidden semi-Markov models, would open up this work to new applications and use cases.

REFERENCES

[1] Charu C Aggarwal and Jiawei Han. 2014. *Frequent pattern mining*. Springer.
[2] Kerem Altun, Billur Barshan, and Orkun Tunçel. 2010. Comparative study on classifying human activities with miniature inertial and magnetic sensors. *Pattern Recognition* 43, 10 (2010), 3605–3620.
[3] Jeffrey D Banfield and Adrian E Raftery. 1993. Model-based Gaussian and non-Gaussian clustering. *Biometrics* (1993), 803–821.
[4] Jeremy Buhler and Martin Tompa. 2002. Finding motifs using random projections. *Journal of Computational Biology* 9, 2 (2002), 225–242.
[5] Nuno Castro and Paulo J Azevedo. 2011. Time series motifs statistical significance. In *Proceedings of the 2011 SIAM International Conference on Data Mining*. SIAM, 687–698.
[6] Bill Chiu, Eamonn Keogh, and Stefano Lonardi. 2003. Probabilistic discovery of time series motifs. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 493–498.
[7] Gautam Das, King-Ip Lin, Heikki Mannila, Gopal Renganathan, and Padhraic Smyth. 1998. Rule Discovery from Time Series.. In *KDD*, Vol. 98. 16–22.
[8] Tak-chung Fu, Fu-lai Chung, Vincent Ng, and Robert Luk. 2001. Evolutionary segmentation of financial time series into subsequences. In *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, Vol. 1. IEEE, 426–430.
[9] Josif Grabocka, Nicolas Schilling, and Lars Schmidt-Thieme. 2016. Latent time-series motifs. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 11, 1 (2016), 6.
[10] David Hallac, Peter Nystrup, and Stephen Boyd. 2016. Greedy Gaussian Segmentation of Multivariate Time Series. *arXiv preprint arXiv:1610.07435* (2016).
[11] David Hallac, Sagar Vare, Stephen Boyd, and Jure Leskovec. 2017. Toeplitz inverse covariance-based clustering of multivariate time series data. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 215–223.
[12] Johan Himberg, Kalle Korpiaho, Heikki Mannila, Johanna Tikänmaki, and Hannu TT Toivonen. 2001. Time series segmentation for context recognition in mobile devices. In *ICDM*.
[13] Richard Hughey and Anders Krogh. 1996. Hidden Markov models for sequence analysis: extension and analysis of the basic method. *Bioinformatics* 12, 2 (1996), 95–107.
[14] Eamonn Keogh and Jessica Lin. 2005. Clustering of time-series subsequences is meaningless: implications for previous and future research. *Knowledge and information systems* 8, 2 (2005), 154–177.
[15] Eamonn J Keogh and Michael J Pazzani. 2000. Scaling up dynamic time warping for datamining applications. In *Proceedings of the sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 285–289.
[16] Yuan Li, Jessica Lin, and Tim Oates. 2012. Visualizing variable-length time series motifs. In *Proceedings of the 2012 SIAM international conference on data mining*. SIAM, 895–906.
[17] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Bill Chiu. 2003. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*. ACM, 2–11.
[18] Jessica Lin, Eamonn Keogh, Li Wei, and Stefano Lonardi. 2007. Experiencing SAX: a novel symbolic representation of time series. *Data Mining and Knowledge Discovery* 15, 2 (2007), 107–144.
[19] James H Martin and Daniel Jurafsky. 2009. *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*. Pearson/Prentice Hall.
[20] Karthik Mohan, Palma London, Maryam Fazel, Daniela Witten, and Su-In Lee. 2014. Node-based learning of multiple Gaussian graphical models. *The Journal of Machine Learning Research* 15, 1 (2014), 445–488.
[21] Matthew A Napierala. 2012. What is the Bonferroni correction. *AAOS Now* 6, 4 (2012), 40.
[22] Soumitra Pal, Peng Xiao, and Sanguthevar Rajasekaran. 2016. Efficient sequential and parallel algorithms for finding edit distance based motifs. *BMC genomics* 17, 4 (2016), 465.
[23] Panagiotis Papapetrou, George Kollios, Stan Sclaroff, and Dimitrios Gunopulos. 2005. Discovering frequent arrangements of temporal intervals. In *null*. IEEE, 354–361.
[24] Youngsuk Park, David Hallac, Stephen Boyd, and Jure Leskovec. 2017. Learning the Network Structure of Heterogeneous Data via Pairwise Exponential Markov Random Fields. In *Artificial Intelligence and Statistics*. 1302–1310.

[25] py-rstr max. 2011. Google Code Archive: py-rstr-max. https://code.google.com/archive/p/py-rstr-max/
[26] Majed Sahli, Essam Mansour, and Panos Kalnis. 2014. ACME: A scalable parallel system for extracting frequent patterns from a very long sequence. *The VLDB Journal* 23, 6 (2014), 871–893.
[27] Padhraic Smyth. 1997. Clustering sequences with hidden Markov models. In *Advances in Neural Information Processing Systems (NIPS)*. 648–654.
[28] Alireza Vahdatpour, Navid Amini, and Majid Sarrafzadeh. 2009. Toward Unsupervised Activity Discovery Using Multi-Dimensional Motif Detection in Time Series.. In *IJCAI*, Vol. 9. 1261–1266.
[29] Yimin Xiong and Dit-Yan Yeung. 2004. Time series clustering with ARMA mixtures. *Pattern Recognition* 37, 8 (2004), 1675–1689.
[30] Dragomir Yankov, Eamonn Keogh, Jose Medina, Bill Chiu, and Victor Zordan. 2007. Detecting time series motifs under uniform scaling. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 844–853.

Algorithm 2: Replacement of Known Candidates

Function *ReplaceKnownCands (m)*
  Sort $\mathcal{D}$ by decreasing length, breaking ties by decreasing empirical probability.;
  $m' \leftarrow m$ **foreach** $d \in \mathcal{D}$ **do**
    $m' \leftarrow$ Replace non-overlapping instances of d in m' from left to right with $\phi(d)$ **return** m'

Algorithm 3: Motif Candidate Set Generation

Data: C: the set of candidate motifs
Sort C by decreasing length.;
$\mathcal{D} \leftarrow$ The set of repeats of size 2.;
$\mathcal{A} \rightarrow \{\}$ **foreach** $m \in C$ **do**
  $m' \leftarrow$ `ReplaceKnownCands`(m);
  $p_\emptyset \leftarrow \prod_{m'_i \in m'} P_\emptyset(m'_i)$;
  if $P\ \mathcal{B}(|\mathbf{S}'|, p_\emptyset \geq N_m\ \leq \frac{\alpha}{|C|}$, **then**
    Add m to $\mathcal{A}$;
    Add m to $\mathcal{D}$;
return $\mathcal{A}$.

A IMPLEMENTATION DETAILS FOR MOTIF CANDIDATE SET GENERATION

Here we give a more detailed algorithm and implementation details for generating a candidate motif set. Each candidate motif m is a list of states. Given our candidate set of motifs generated from the suffix array, we seek output a non-redundant set of relevant, significant motif candidates.

The null model keeps track of a set $\mathcal{D}$, which contains repeated subsequences that have already been accepted as candidates. Initially, we set the null model to automatically know about repeats of size 2 that occurred in $\mathbf{S}$, which the null model assumes appear with probability according to their empirical frequency.

For a subsequence d, let the non-overlapping count of d in $\mathbf{S}$ be N_d. Denote $\phi(d)$ as a dummy "state" which occurs with probability $P_\emptyset(\phi(d)) = \frac{N_d}{|\mathbf{S}|}$ according to the null model. Given a candidate motif m, we define the sub-routine `ReplaceKnownCands` which returns a new m' which replaces known subsequences in m with their fake states (Algorithm 2). The null model then evaluates the probability of the new m' under the assumption that each state m'_i (which might be real or "dummy") occurs independently according to its empirical probability.

We then reject candidates who do not have a high enough threshold α. We use a Bonferroni corrected $\alpha = 0.001$ normalized by the number of motif candidates being evaluated [21]. The full algorithm is in Algorithm 3.

B REPRODUCIBILITY DETAILS FOR OUR EXPERIMENTAL EVALUATION

We performed a total of 2 experiments (synthetic and cycling data) and 2 case studies (automobile and airplane data). For the two experiments, we pick K (our number of states) to be the ground truth

Experiment	K	β	γ
Synthetic	10	25	0.8
Cycling	8	100	0.8
Airplane	10	50	0.8
Automobile	8	50	0.6

Table 3: Hyperparameters for experiments

number of states in the constructed dataset. We then chose β via BIC. For the case studies, we performed BIC to identify the cluster number. In both the cycling and airplane dataset, we performed PCA to reduce the dimensionality of each measurement to 10 and 13 features respectively. Scripts to run the experiments as well as the packaged results can be found in our linked source code.

While details of the hyperparameters can be found in our packaged software (see Section), below is a table of the hyperparameters used for each experiment.