EXTRACT: Strong Examples from Weakly-Labeled Sensor Data

Davis W. Blalock, John V. Guttag
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA, USA
{dblalock, guttag}@mit.edu

Abstract—Thanks to the rise of wearable and connected devices, sensor-generated time series comprise a large and growing fraction of the world’s data. Unfortunately, extracting value from this data can be challenging, since sensors report low-level signals (e.g., acceleration), not the high-level events that are typically of interest (e.g., gestures). We introduce a technique to bridge this gap by automatically extracting examples of real-world events in low-level data, given only a rough estimate of when these events have taken place.

By identifying sets of features that repeat in the same temporal arrangement, we isolate examples of such diverse events as human actions, power consumption patterns, and spoken words with up to 96% precision and recall. Our method is fast enough to run in real time and assumes only minimal knowledge of which variables are relevant or the lengths of events. Our evaluation uses numerous publicly available datasets and over 1 million samples of manually labeled sensor data.

Index Terms—Sensor data; Semi-supervised learning

I. INTRODUCTION

The rise of wearable technology and connected devices has made available a vast amount of sensor data, and with it the promise of improvements in everything from human health [1] to user interfaces [2] to agriculture [3]. Unfortunately, the raw sequences of numbers comprising this data are often insufficient to offer value. For example, a smart watch user is not interested in their arm’s acceleration signal, but rather in having their gestures or actions recognized.

Spotting such high-level events using low-level signals is challenging. Given enough labeled examples of the events taking place, one could, in principle, train a classifier for this purpose. Unfortunately, obtaining labeled examples is an arduous task [4]–[7]. While data such as images and text can be culled at scale from the internet, most time series data cannot. Furthermore, the uninterpretability of raw sequences of numbers often makes time series difficult or impossible for humans to annotate [4].

It is, however, often possible to obtain approximate labels for particular stretches of time. The widely-used human action dataset of [8], for example, consists of streams of data in which a subject is known to have performed a particular action roughly a certain number of times, but the exact starts and ends of each action instance are unknown. Furthermore, the recordings include spans of time that do not correspond to any action instance. Similarly, the authors of the Gun-Point dataset obtained recordings containing different gestures, but had to expend considerable effort extracting each instance [6]. This issue of knowing that there are examples within a time series but not knowing where in the data they begin and end is common [4]–[6], [8], [9].

To leverage such weak labels, we developed an algorithm, EXTRACT, that efficiently isolates examples of an event given only a time series known to contain several occurrences of it. A simple illustration of the problem we consider is given in Figure 1. The various lines depict the normalized current, voltage, and other power measures of a home dishwasher. Shown are three instances of the dishwasher running, with idleness in between. With no prior knowledge or domain-specific tuning, our algorithm correctly determines not only what this repeating event looks like, but also where it begins and ends.

Fig. 1: a) True instances of the dishwasher running (shaded). b) Even when told the length and number of event instances, the recent algorithm of [10] returns intervals with only the beginning of the event. c) Our algorithm returns accurate intervals with no such prior knowledge.

This is a challenging task, since the variables affected by the event, as well as the number, lengths, and positions of event instances, are all unknowns. Further, it is not even clear what objective should be maximized to find an event. For example, finding the nearest subsequences using the Euclidean distance yields the incorrect event boundaries returned by [10] (Fig 1b).

To overcome these barriers, our technique leverages three observations:
1. Each subsequence of a time series can be seen as having representative features; for example, it may resemble different shapelets [11] or have a particular level of variance.
2. A repeating event will cause a disproportionate number of these features to occur together where the event happens. In Figure 1, for example, these features are a characteristic arrangement of spikes in the values of certain variables.
3. If we can identify these features, we can locate each instance of the event with high probability. This holds even in the presence of irrelevant variables (which merely fail to contribute useful features) and unknown instance lengths (which can be inferred based on the interval over which the features occur together).

Our contributions consist of:
- A formulation of semi-supervised event discovery in time series under assumptions consistent with real data. In particular, we allow multivariate time series, events that affect only subsets of variables, and instances of varying lengths.
- An $O(N \log(N))$ algorithm to discover event instances under this formulation. It requires less than 300 lines of code and is fast enough to run on batches of data in real time. It is also considerably faster, and often much more accurate, than similar existing algorithms [10], [12].
- Open source code and labeled time series that can be used to reproduce and extend our work. In particular, we believe that our annotation of the full dishwasher dataset [13] makes this the longest available sensor-generated time series with ground truth event start and end times.

II. DEFINITIONS AND PROBLEM

Definition II.1. Time Series. A $D$-variable time series $T$ of length $N$ is a sequence of real-valued vectors $t_1, \ldots, t_N, t_i \in \mathbb{R}^D$. If $D = 1$, we call $T$ “univariate”, and if $D > 1$, we call $T$ “multivariate.”

Definition II.2. Region. A region $R$ is a pair of indices $(a, b), a \leq b$. The value $b - a + 1$ is termed the length of the region, and the time series $t_a, \ldots, t_b$ is the subsequence for that region. If a region reflects an occurrence of the event, we term the region an event instance.

A. Problem Statement

We seek the set of regions that are most likely to have come from a shared “event” distribution rather than a “noise” distribution. This likelihood is assessed based on the subset of features maximizing how distinct these distributions are (using some fixed feature representation).

Formally, let $x_1, \ldots, x_K$ be binary feature representations of all $K$ possible regions in a given time series and $x_i^j$ denote feature $j$ in the region $i$. We seek the optimal set of regions $\mathcal{R}^*$, defined as:

$$\mathcal{R}^* = \arg \max_{\mathcal{R}} \prod_{j \in \mathcal{F}} \sum_{i \in \mathcal{R}} c_j \left( \log(\theta_{1j}) - \log(\theta_{0j}) \right)$$

(1)

where $\theta_{0j}$ and $\theta_{1j}$ are the empirical probabilities for each feature $j$ in the whole time series and the regions $\mathcal{R}$ respectively, and $c_j$ is the count of feature $j$. $\sum_{i \in \mathcal{R}} x_i^j, \mathcal{F}$ is the set of features that best separate the event. The prior $p(\mathcal{R})$ is 0 if regions overlap too heavily or violate certain length bounds (see below) and is otherwise uniform.

Equation 1 says that we would like to find regions $i$ and features $j$ such that $x_i^j$ happens both many times (so that $c_j$ is large) and much more often than would occur by chance (so that $\log(\theta_{1j}) - \log(\theta_{0j})$ is large). In other words, the best features $\mathcal{F}$ are the largest set that consistently occurs across the most regions, and $\mathcal{R}^*$ is these regions.

Given certain independencies, this objective is a MAP estimate of the regions and features. Because of space constraints, we defer the details to [14].

B. Assumptions

We do not make any of the following common assumptions:
- A known or constant length for instances, a known or regular spacing between instances, or a known number of instances.
- A known set of characteristics shared by instances. In particular, we do not assume that all instances have the same mean and variance, so we cannot bypass normalization when making similarity comparisons.
- That there is only one variable, or that all variables are affected by the event.
- Anything about variables not affected by the event.

So that the problem is well-defined, we do assume that:
- The time series contains instances of only one class of event. It may contain other transient phenomena, but we take our weak label to mean (only) that the primary structure in the time series comes from the events of the labeled class and that there are no other repeating events.
- There are at least two instances of the event, and each instance produces some characteristic but unknown pattern in the data.
- There exist bounds $M_{min}$ and $M_{max}$, $M_{min} > M_{max}/2$ on the lengths of instances. These bounds disambiguate the case where pairs of adjacent instances could be viewed as single instances of a longer event. Similarly, no two instances overlap by more than $M_{min} - 1$ time steps.
- We also do not consider datasets in which instances are rare [15]—all time series used in our experiments have instances that collectively comprise ~10% of the data or more (though this exact number is not significant).

C. Why the Task is Difficult

The lack of assumptions means that the number of possible sets of regions and relevant variables is intractably large. Suppose that we have a $D$-variable time series $T$ of length $N$ and $M_{min} \leq M \leq M_{max}$. There are up to $O(N/M)$ instances, which can collectively start at (at most) $\binom{N}{M}$ positions. Further, each can be of $O(M)$ different lengths. Finally, the event may affect any of $2^D - 1$ possible subsets of variables. Altogether, this means that there are roughly $O(N^N/M \cdot M^{N/M} \cdot 2^D)$ combinations of regions and variables.

Moreover, while there may be heuristics or engineered features that could allow isolation of any particular event in
any particular domain, we seek to develop a general-purpose tool that requires no coding or tuning by humans. We therefore do not use such event-specific knowledge. This generality is both a convenience for human practitioners and a necessity for real-world deployment of a system that learns new events at runtime. Lastly, because our aim is to extract examples for future use, we seek to locate full events, not merely the pieces that are easiest to find.

III. RELATED WORK

Several authors have built algorithms to address the difficulty of obtaining labeled time series for various tasks. The authors of [6] and [7] cluster univariate time series when much of the data in each time series is irrelevant. They do this by discovering informative shapelets [11] in an unsupervised manner. Their goal is to assign entire time series to various clusters. In contrast, we are interested in assigning a subset of the regions within a single time series to a particular “cluster.”

There is also a vast body of work on unsupervised discovery of repeating patterns in time series, typically termed “motif discovery.” Most of these works consider univariate time series and/or the task of finding only the closest pair of regions under some distance measure [19]. Others consider the task of finding multiple motifs and/or refining motif results produced by other algorithms [2], [9], [10], [12], both of which are orthogonal to our work in that they could employ our algorithm as the basic motif-finding subroutine.

A few motif discovery works seek to find all instances of a given event as we do, albeit under different assumptions. The techniques of [10], [16], and [17] do so by finding closest pairs of subsequences at different lengths and then extracting subsequences that are sufficiently similar under an entropy-based measure. Those of [12] and [9] do much the same, although with a distance-based generalization heuristic. All except [17] assume that event instances share a single length, and all but [9] assume that all variables are relevant. We discuss [10], [12], and [17] further in Section V.

IV. METHOD

The basic steps of our approach are given in Algorithm 1. In step 1, we create a representation of the time series that is invariant to instance length and enables rapid pruning of irrelevant variables. In step 2, we find sets of regions that may contain event instances. In step 3, we refine these sets to estimate the optimal instances \( R^* \). A more detailed description of the full algorithm and why it works can be found in the expanded version of this paper [18].

A. Feature Matrix Construction

We transform the data into a sparse binary feature matrix that encodes the presence of particular shapes at each position in the time series (Fig 2). We leave explanation of how these shapes are selected and their presence is determined to [18]. Using this feature matrix, we can compare windows of data without knowing the lengths of instances. This is because, even if there is extraneous data at the ends of the windows, there will still be more common features where the event happens (Fig 3a) than would be expected by chance. This allows us to

**Algorithm 1 EXTRACT**

1. Transform the time series \( T \) into a feature matrix \( \Phi \)
   a) Sample subsequences from the time series to use as shape features
   b) Transform \( T \) into \( \Phi \) by encoding the presence of these shapes across time
   c) Blur \( \Phi \) to achieve length and time warping invariance
2. Using \( \Phi \), generate sets of “candidate” windows that may contain event instances
   a) Find “seed” windows that are unlikely to have arisen by chance
   b) Find “candidate” windows that resemble each seed
   c) Rank candidates based on similarity to their seeds
3. Infer the true instances within these candidates
   a) Greedily construct subsets of candidate windows based on ranks
   b) Score these subsets and select the best one
   c) Infer exact instance boundaries within the selected windows

![Fig. 2: Feature matrix. Each row is the presence of a particular shape, and each column is a particular time (shown at reduced granularity). Similar regions of data contain the same shapes in roughly the same temporal arrangement.](image)

consider only windows of a fixed length \( M_{\text{max}} \) even though the instance lengths vary.

This representation also allows us to recover the precise starts and ends of instances once the windows containing them are identified. Specifically, if a start or end column does not contain a consistent set of 1s across these windows, it is probably not part of the event, and we prune it (Fig 3b).

![Fig. 3: Because the values in the feature matrix are independent of the window length, a window longer than the event can be used to search for instances.](image)
there will be many irrelevant shapes that are not indicative of instances.

To handle differences in the temporal arrangement of shapes, we “blur” the feature matrix in time. The effect is that a given shape is counted as being present over an interval, rather than at a single time step. This is shown in Figure 4, using the intersection of the features in two windows as a simplified illustration of how similar they are. Since the blurred features are no longer binary, we depict the “intersection” as the elementwise minimum of the windows.

![Features vary in exact position across instances](image)

**Fig. 4:** Blurring the feature matrix. Despite the second sine wave being longer and warped in time, the two windows still appear similar when blurred.

To deal with irrelevant features, no action is required beyond the application of our “intersection” operation. To see this, suppose that the probability of an irrelevant feature being present at a particular location in an instance-containing window is \( p_0 \), which is typically < 0.1 in our sparse representation. Then the probability of it being present by chance in \( k \) windows is \( p_0^k \approx 0 \) even for small \( k \).

Finally, to construct a feature matrix for a multivariate time series, we construct feature matrices for each of the variables and concatenate them row-wise. A variable may not be relevant, but this just means that it will add irrelevant features, which will be ignored with high probability.

### B. Finding Instances

Given the feature matrix, we locate event instances using Algorithm 2. The idea is that if we are given one “seed” window that contains an instance, we can generate a set of similar “candidate” windows and then determine which of these are likely to contain event instances. Since we cannot generate seeds that are certain to contain instances, we generate many seeds and try each. Seeds are selected via the biased sampling scheme described in [18], which returns subsequences that appear least similar to random walks.

The main loop iterates through all seeds \( s \) and generates sets of candidate windows for each. These candidates are the windows whose dot products with \( s \) are local maxima. To prevent excess overlap, a minimum spacing is enforced between the candidates by only taking the best relative maximum in any interval of width \( M_{\text{min}} \). If \( s \) contains an event instance, the resulting candidates should be (and typically are) a superset of the true instance-containing windows.

In the inner loop, we assess subsets of the candidates to determine which ones contain instances. Since there are \( 2^{|C|} = O(2N/M_{\text{min}}) \) possible subsets, we use a greedy approach that tries only \( |C| = O(N/M_{\text{min}}) \) subsets. Specifically, we rank the candidates based on their dot products with \( s \) and assess subsets that contain the \( k \) highest-ranking candidates for each possible \( k \). The scoring function is Eq 1 with the constraint that \( j \in F \implies (j \not\in c_{k+1} \land \theta_{ij} > .5) \). The final set returned is the highest-scoring subset of candidates for any seed.

### C. Complexity

For a \( D \)-variable time series of length \( N \), EXTRACT has complexity \( O(DM_{\text{max}} \log(M_{\text{max}})N \log(N)) \) using instance length upper bound \( M_{\text{max}} \). See [18] for details.

### V. RESULTS

We implemented our algorithm, along with baselines from the literature [10], [12], using SciPy [20]. All code, experiments, and raw results are publicly available at [14] and explained in more detail in the expanded paper at [18].

### A. Datasets

We used the following datasets, selected on the basis that they were both publicly available and contained repeated instances of some ground truth event, such as a repeated gesture or spoken word.

**MSRC-12** [8] consists of \((x,y,z)\) human joint positions captured by a Microsoft Kinect while subjects repeatedly performed specific motions. Each of the 594 time series in the dataset has 80 variables and contains 8-12 event instances.

**TIDIGITS** [21] is a large collection of human utterances of decimal digits. We use a subset of the data consisting of all recordings containing only one type of digit (e.g., only “9”s).

**Dishtwasher** [13] consists of energy consumption and related electricity metrics at a power meter connected to a residential dishwasher. It contains twelve variables and two years worth of data sampled once per minute, for a total of 12.6 million data points. We manually annotated and verified event instances across all 1 million+ of its samples.

**UCR** [22]. Following [10], we constructed synthetic datasets by planting examples from the UCR Time Series Archive.

---

**Algorithm 2** FindInstances(\( S, \mathcal{X} \))

1: **Input:** \( S \), “seed” windows; \( \mathcal{X} \), all blurred windows
2: \( \mathcal{I}_{\text{best}} \leftarrow \{\} \); score\(_{\text{best}} \leftarrow -\infty \)
3: for each seed window \( s \in S \) do
4: \( P \leftarrow [s \cdot \hat{x}_i, \hat{x}_i \in \mathcal{X}] \) // dot products with \( s \)
5: \( \mathcal{C} \leftarrow \text{spacedLocalMaxima}(P) \)
6: \( \mathcal{C} \leftarrow \text{sortByDescendingDotProd}(\mathcal{C}, P) \)
7: for \( k = 2, \ldots, |\mathcal{C}| \) do // assess subsets of \( \mathcal{C} \)
8: \( \mathcal{I} \leftarrow \{c_1, \ldots, c_k\} \) // \( k \) best candidates
9: score \leftarrow \text{computeScore}(\mathcal{I}, c_{k+1})
10: if score > score\(_{\text{best}} \) then
11: \( \mathcal{I}_{\text{best}} \leftarrow \mathcal{I} \); score\(_{\text{best}} \leftarrow \text{score} \)
12: return \( \mathcal{I}_{\text{best}} \)

---

802
in random walks. We took examples from the 20 smallest datasets, as measured by the lengths of their time series.

B. Evaluation Measures

Let \( \mathcal{R} \) be the ground truth set of instance regions and let \( \hat{\mathcal{R}} \) be the set of regions returned by the algorithm being evaluated. Further let \( r_1 = (a_1, b_1) \) and \( r_2 = (a_2, b_2) \) be two regions.

**Definition V.1.** IOU\((r_1, r_2)\). The Intersection-Over-Union (IOU) of \( r_1 \) and \( r_2 \) is given by \( \frac{|r_1 \cap r_2|}{|r_1 \cup r_2|} \), where \( r_1 \) and \( r_2 \) are treated as intervals.

**Definition V.2.** Match\((r_1, r_2, \tau)\). \( r_1 \) and \( r_2 \) are said to Match at a threshold of \( \tau \) if \( \text{IOU}(r_1, r_2) \geq \tau \).

**Definition V.3.** MatchCount\((\mathcal{R}, \mathcal{R}, \tau)\). The MatchCount of \( \mathcal{R} \) given \( \mathcal{R} \) and \( \tau \) is the greatest number of matches at threshold \( \tau \) that can be produced by pairing regions in \( \mathcal{R} \) with regions in \( \mathcal{R} \) such that no region in either set is present in more than one pair.

The F1 Scores we report use the MatchCount as the number of true positives.

C. Comparison Algorithms

While none of the techniques we reviewed both seek to solve our problem and operate under assumptions as relaxed as ours, we found that two existing algorithms solving the univariate version of the problem could be generalized to the multivariate case:

- **Dist**—finding the closest pair of subsequences under the \( L_2 \) distance, and deeming all subsequences within some threshold distance of this pair instances [9], [12]
- **MDL**—the single-motif-finding subroutine of [10], with distances and description lengths summed over variables; this amounts to closest-pair motif discovery to find seeds, candidate generation based on \( L_2 \) distance to these seeds, and instance selection using a Minimum Description Length (MDL) criterion.

D. Instance Discovery Accuracy

The core problem addressed by our work is the robust location of multiple event instances within a time series known to contain a small number of them. To assess our effectiveness in solving this problem, we evaluated the F1 score on each of the four datasets, varying the threshold \( \tau \) for how much ground truth and reported instances needed to overlap in order to count as matching. In all cases, \( M_{\text{min}} \) and \( M_{\text{max}} \) were set to 1/20 and 1/10 of the time series length. As shown in Figure 5, we outperform the comparison algorithms for virtually all match thresholds on all datasets.

Note that MSRC-12 values are constant because instance boundaries are not defined in this dataset (see [18]). Further, the dataset on which we perform the closest to the comparisons (UCR) is synthetic, univariate, and only contains instances that are the same length. These last two attributes are what Dist and MDL were designed for, so the similar F1 scores suggest that EXTRACT’s superiority on other datasets is due to its robustness to violation of these conditions. Visual examination of the errors on this dataset suggests that all algorithms have only modest accuracy because there are often regions of random walk data that are more similar in shape to one another than the instances are.

Our accuracy on real data is not only superior to the comparisons, but also high in absolute terms (Table 1). Suppose that we consider IOU thresholds of 0.25 or 0.5 to be “correct” for our application. The former might correspond to detecting a portion of a gesture, and the latter might correspond to detecting most of it, with a bit of extra data at one end. At each of these thresholds, our algorithm discovers event instances with an F1 score of over .9 on real data.

**Table 1: EXTRACT F1 Scores are High in Absolute Terms**

|               | Overlap ≥ .25 | Overlap ≥ .5 |
|---------------|--------------|-------------|
|               | Ours | MDL | Dist | Ours | MDL | Dist |
| Dishwasher    | 0.935 | 0.786 | 0.808 | 0.910 | 0.091 | 0.191 |
| TIDIGITS      | 0.955 | 0.779 | 0.670 | 0.915 | 0.140 | 0.174 |
| MSRC-12       | 0.947 | 0.943 | 0.714 | 0.947 | 0.943 | 0.714 |
| UCR           | 0.671 | 0.593 | 0.587 | 0.539 | 0.510 | 0.504 |

An example of our algorithm’s output on the TIDIGITS dataset is shown in Figure 6. The regions returned (shaded) closely bracket the individual utterances of the digit “0.” The “Learned Pattern” is the set of \( \left( \log(\theta_{j_1}) - \log(\theta_{0}) \right) I\{j \in \mathcal{F} \} \) values from the objective function (Eq 1).

**Fig. 5:** The proposed algorithm is more accurate for virtually all “match” thresholds on all datasets. Shading corresponds to 95% confidence intervals.

**Fig. 6:** (Top) Original time series, with instances inferred by EXTRACT in gray. (Bottom) The feature matrix. (Right) The learned feature weights. These resemble a “blurred” version of the features that occur when the word is spoken.
E. Speed

In addition to being accurate on both real and synthetic data, our algorithm is fast. To assess performance, we recorded the time it and the comparisons took to run on increasingly long sections of random walk data and the Dishwasher data.

In the first column of Fig 7, we vary only the length of the time series (N) and keep M_{min} and M_{max} fixed at 100 and 150. In the second column, we hold N constant at 5000 and vary M_{max}, with M_{min} fixed at M_{max} − 50 so that the number of lengths searched is constant. In the third column, we fix N at 5000 and set (M_{min}, M_{max}) to (150, 150), (140, 160), ..., (100, 200).

Our algorithm is at least an order of magnitude faster in virtually all experimental conditions. Further, it shows little or no increase in runtime as M_{min} and M_{max} are varied and increases only slowly with N. This is in line with what would be expected given our computational complexity.

Our approach is one to two orders of magnitude faster than comparisons.

Both Dist and MDL sometimes plateau in runtime thanks to their early-abandoning techniques. Dist even decreases because the lower bound it employs [12] to prune similarity comparisons is tighter for longer time series.

As with accuracy, our technique is fast not only relative to comparisons, but also in absolute terms—we are able to run the above experiments in minutes and search each time series in seconds (even with our simple Python implementation). Since these time series reflect phenomena spanning many seconds or hours, this means that our algorithm could be run in real time in many settings.

VI. DISCUSSION AND CONCLUSION

We have described an algorithm to efficiently and accurately locate instances of an event within a multivariate time series given virtually no prior information about the nature of this event. In particular, we assume no knowledge of how many times the event has occurred, what features distinguish it, or which variables it affects. Using a diverse group of publicly available datasets, we showed that this technique is fast and accurate both in absolute terms and compared to existing algorithms, despite its limited assumptions.

Moreover, while this work has focused on a feature matrix reflecting the presence of particular shapes in the data, our technique could be applied even when signals are not described well by shapes—our learning algorithm requires only a sparse feature matrix with entries between 0 and 1. In particular, one could one-hot encode categorical variables such as “day of week” or “user gender” and add these features with no change to the algorithm. We consider this adaptability a major strength of our approach, since mixed real and categorical variables are common in many domains.

In short, by applying our technique to low-level signals of various kinds, one can isolate segments of data produced by high-level events as diverse as spoken words, human actions, and household appliance usage.

VII. ACKNOWLEDGEMENTS

This work was supported by NSF Grant 020772-00001.

REFERENCES

[1] M. Moazeni, B. Mortazavi, and M. Sarrafzadeh, “Multi-dimensional signal search with applications in remote medical monitoring,” IEEE RSN 2013, pp. 1-6, 2013.
[2] D. Minnen, T. Starner, I. Essa, and C. Isbell, “Discovering Characteristic Actions from On-Body Sensor Data,” pp. 11-18, 2006.
[3] Y. Chen, A. Why, G. Batista, A. Mafra-Neto, and E. Keogh, “Flying Insect Classification with Inexpensive Sensors,” arXiv.org, Mar. 2014.
[4] H. T. Cheng, F. T. Sun, M. Griss, P. Davis, and J. Li, “Nuactiv: Recognizing unseen new activities using semantic attribute-based learning,” ICMS 2013, 2013.
[5] B. Hu, Y. Chen, and E. Keogh, “Time series classification under more realistic assumptions,” in SDM, 2013.
[6] J. Zakaria, A. Mucen, and E. Keogh, “Clustering time series using unsupervised-shapelets,” Data Mining (ICDM), 2012.
[7] L. Ulanova, N. Begum, and E. Keogh, “Scalable Clustering of Time Series with U-Shapelets,” SDM 2015, pp. 900-908, 2015.
[8] S. Fothergill, H. M. Mentis, P. Kohli, and S. Nowozin, “Instructing people for training gestural interactive systems,” in CHI, I. A. Konstan, E. H. Chi, and K. Hook, Eds. ACM, 2012, pp. 1737-1746.
[9] D. Minnen, C. Isbell, I. Essa, and T. Starner, “Detecting Subdimensional Motifs: An Efficient Algorithm for Generalized Multivariate Pattern Discovery,” in ICDM 2007, 2007.
[10] S. Yingchareonthawornchai and H. Sivarak, “Efficient proper length time series motif discovery,” Data Mining (ICDM), 2013.
[11] L. Ye and E. Keogh, “Time series shapelets: a new primitive for data mining,” in ACM SIGKDD. ACM Request Permissions, 2009.
[12] A. Mueen, “Enumeration of Time Series Motifs of All Lengths,” in ICDM 2013. IEEE, 2013, pp. 547-556.
[13] S. Makonin, F. Popowich, L. Bartram, B. Gill, and I. V. Bajic, “AMPds: A public dataset for load disaggregation and eco-feedback research,” in IEEE Electrical Power & Energy Conference (EPEC). IEEE, 2013.
[14] EXTRACT Homepage. [Online]. Available: http://smarturl.it/extract
[15] N. Begum and E. Keogh, “Rare time series motif discovery from unbounded streams,” VLDB, vol. 8, no. 2, Oct. 2014.
[16] P. Nunnathan, V. Niematrakul, and C. A. Ratanamahatana, Parameter-free motif discovery for time series data. IEEE, 2012.
[17] T. Rakthanmanon, E. J. Keogh, S. Lonardi, and S. Evans, “Time Series Epenthesis: Clustering Time Series Streams Requires Ignoring Some Data,” in ICDM 2011, 2011.
[18] D. W. Blalock and J. V. Guttag, “EXTRACT: Strong Examples from Weakly-Labeled Sensor Data,” arXiv.org, 2016.
[19] D. Yankov, E. Keogh, J. Medina, B. Chiu, and V. Zordan, “Detecting time series motifs under uniform scaling,” in ACM SIGKDD. ACM Request Permissions, 2007.
[20] E. Jones, T. Oliphant, and P. Peterson, “SciPy: Open source scientific tools for Python,” 2014.
[21] R. G. Leonard and G. Doddington, “Tidigits speech corpus,” Texas Instruments, Inc, 1993.
[22] E. J. Keogh, Q. Zhu, B. Hu, Y. Hao, X. Xi, L. Wei, and C. A. Ratanamahatana, (2011) The UCR Time Series Classification/Clustering Homepage. [Online]. Available: http://www.cs.ucr.edu/~eamonn/time_series_data/