BEATLEX: Summarizing and Forecasting Time Series with Patterns

Bryan Hooi[1,2], Shenghua Liu[3], Asim Smailagic[1], and Christos Faloutsos[1]

[1] School of Computer Science, Carnegie Mellon University
[2] Department of Statistics, Carnegie Mellon University,
[3] CAS Key Laboratory of Network Data Science & Technology,
Institute of Computing Technology, Chinese Academy of Sciences
bhooi@andrew.cmu.edu, liushenghua@ict.ac.cn, asim@cs.cmu.edu,
christos@cs.cmu.edu

Abstract. Given time-series data such as electrocardiogram (ECG) readings, or motion capture data, how can we succintly summarize the data in a way that robustly identifies patterns that appear repeatedly? How can we then use such a summary to identify anomalies such as abnormal heartbeats, and also forecast future values of the time series? Our main idea is a vocabulary-based approach, which automatically learns a set of common patterns, or 'beat patterns,' which are used as building blocks to describe the time series in an intuitive and interpretable way. Our summarization algorithm, BEATLEX (BEAT LEXicons for Summarization) is: 1) fast and online, requiring *linear* time in the data size and *bounded* memory; 2) effective, outperforming competing algorithms in labelling accuracy by *5.3* times, and forecasting accuracy by *1.8* times; 3) principled and parameter-free, as it is based on the Minimum Description Length principle of summarizing the data by compressing it using as few bits as possible, and automatically tunes all its parameters; 4) general: it applies to any domain of time series data, and can make use of multidimensional (i.e. coevolving) time series.

1 Introduction

Consider a medical team who wishes to monitor patients in a large hospital. How can we design an algorithm that monitors multiple time series (blood pressure, ECG, etc.) for a patient, automatically learns and summarizes common patterns, and alerts a doctor when the patient's overall state deviates from the norm?

Time series data has attracted huge interest in countless domains, including medicine [**?**], social media [**?**], and sensor data [**?**]. But as the scale and complexity of time series data has exploded, the human capacity to process data has not changed. This has led to a growing need for scalable algorithms which automatically summarize high-level patterns in data, or alert a user's attention toward anomalies.

Figure 1 (top) shows an example of an ECG sequence. This ECG sequence contains two distinct types of patterns: indeed, it was manually labelled by cardiologists as shown in Figure 1 (bottom), who marked a *segmentation* at the start of each heartbeat (shown by the grey vertical lines), and labeled the heartbeats as 'normal beats' and 'premature ventricular contractions,' a type of abnormal heartbeat. A natural way to

summarize this sequence, then, would be to exploit patterns that occur multiple times: namely, the two types of heartbeat patterns.

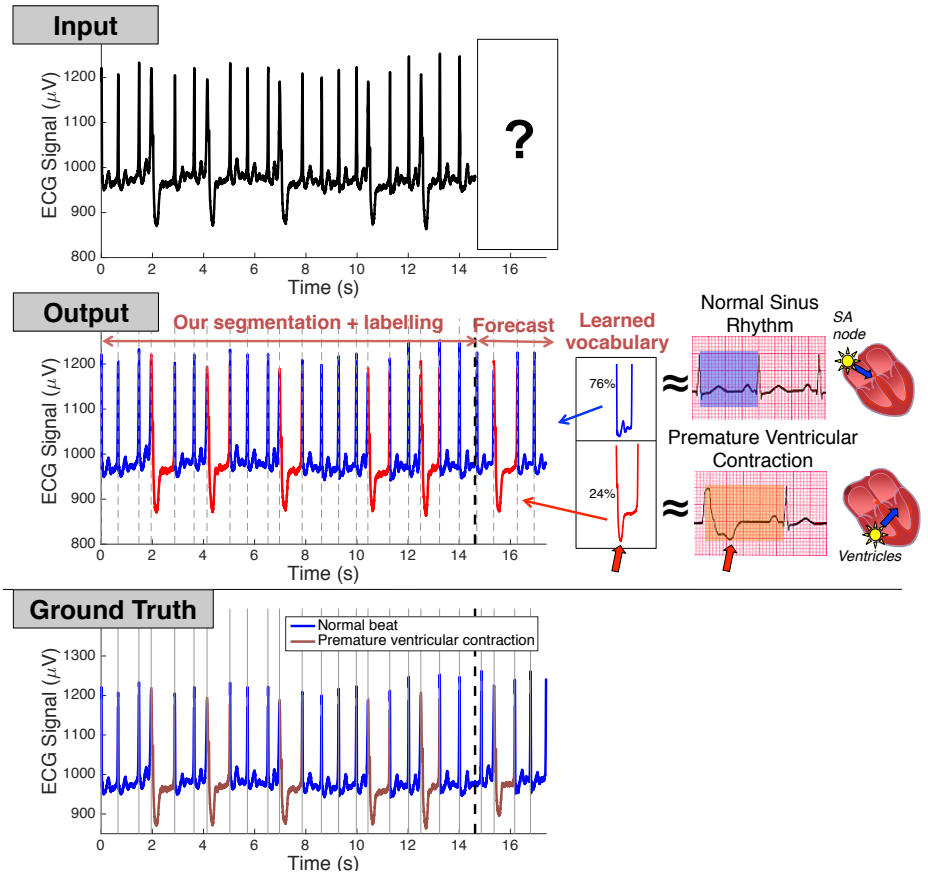


Fig. 1: **Accurate segmentation, labelling, and forecasting**: BEATLEX learns a vocabulary, segments the data (grey dotted vertical lines), labels heartbeat types based on the closest vocabulary term, and forecasts future data to the right of the black dotted line. Its output matches the ground truth almost exactly, and the vocabulary terms correspond closely to medically relevant patterns: 'normal sinus rhythm' and 'premature ventricular contraction.'

Thus, our goal is to summarize a time series using **patterns that occur multiple times**. Here patterns refers to any subsequences which are broadly similar to one another. This includes periodic time series, but applies much more generally to allow patterns whose shape or length gets distorted, or changes over time, or even multiple patterns interspersed with one another as in Figure 1.

Hence, the problem that we focus on is:

Informal Problem 1. *Given a time series X with patterns, find:*

- ***Summarization:*** *a model that succintly represents the common patterns in X.*
- ***Anomalies:*** *time periods in X during which anomalous events occurred (e.g. abnormal heartbeats).*
- ***Forecast:*** *forecast future time ticks of X.*

To robustly handle real-world data such as in Figure 1, there are several key challenges: 1) patterns can be highly complex and nonlinear, so simple parametric models do not work. 2) Patterns are often distorted in length and shape, so methods that assume that a pattern straightforwardly repeats itself do not work. 3) The correct segmentation of the data is unknown: domain-specific segmentation tools for ECG sequences are not enough as we want be able to handle any type of time series (e.g. motion-capture data in Figure 2).

To solve this problem, our algorithm adopts a **vocabulary**-based approach, as illustrated in Figure 1 (middle). It automatically and robustly learns a vocabulary containing the common patterns in the data. At the same time, it finds cut points to break up the data into segments, and describes each segment based on its closest vocabulary term, labelling each segment accordingly. Note in Figure 1 that both its segmentation and labelling are essentially identical to the ground truth annotation by cardiologists.

This vocabulary-based approach is intuitive and interpretable, since it describes the data exactly as a human would, in terms of its patterns and where they are located. In the ECG case, the learned vocabulary terms are also medically relevant, as shown in Figure 1 (right): the blue and red beats correspond to known heartbeat patterns.

Figure 1 also shows that BEATLEX also allows accurate forecasting: the part of the ECG sequence to the right of the black dotted line was forecasted by the algorithm, which also matches the ground truth. Note that the algorithm learns from past data that 3 normal (blue) beats tend to be followed by an abnormal (red) beat, then cycling back to normal beats, a known condition called 'quadrigeminy.'
Our method is:

- **Fast and online:** BEATLEX takes *linear* time in the length of the time series. Moreover, it requires *bounded* memory and constant update time per new data point, making it usable even with limited processing power, such as wearable computers or distributed sensor networks.
- **Effective:** BEATLEX outperforms existing algorithms in labelling accuracy by *5.3* times, and forecasting accuracy by *1.8* times.
- **Principled and parameter-free:** BEATLEX is fit using the Minimum Description Length principle of compressing the data into as few bits as possible, and automatically tunes all its parameters.
- **General:** BEATLEX applies to any type of time series data, and can make use of multidimensional time series, e.g. motion capture data in Figure 2.

2 Background and Related Work

Time Series Summarization. Methods for summarizing a time series can be divided into model-based and non-model-based methods. Model-based methods estimate a statisti-

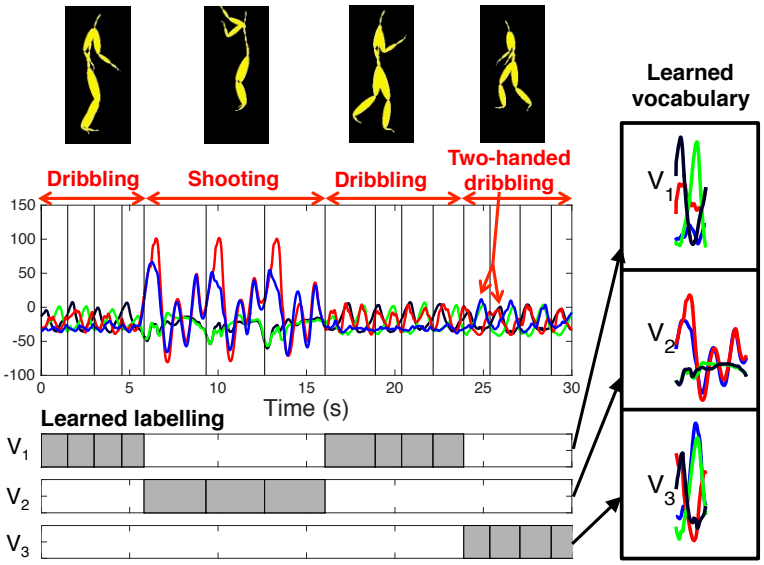


Fig. 2: **Generality:** BEATLEX accurately segments and labels action types in motion capture data of a basketball player. The 4 coloured lines correspond to the subject's left and right arms and legs.

cal model, and include classic methods such as autoregression (AR), ARIMA [**?**], and Hidden Markov Models (HMMs) [**?**]. More recent variants include DynaMMo [**?**], AutoPlait [**?**], and RegimeCast [**?**]. Non model-based methods summarize the data using approximations or feature representations, including SAX [**?**]. TBATS [**?**] is a forecasting approach allowing for complex seasonality. SAX [**?**] discretizes a time series into symbols, and has been used as a preprocessing step before time series clustering [**?**,**?**] and anomaly detection [**?**].

Distance-Based Methods. These methods extract subsequences using sliding windows, and measure distances between the subsequences, using either Euclidean distance or Dynamic Time Warping [**?**] (DTW). DTW is a distance-like measure that allows elastic shifting of the time axis, which has shown good empirical performance for time series classification [**?**]. Discord Detection methods [**?**,**?**] apply Euclidean distances or DTW between subsequences for anomaly detection, while subsequence clustering methods [**?**,**?**] find clusters of similar subsequences.

Table 1 summarizes existing work related to our problem. BEATLEX differs from existing methods as follows: 1) BEATLEX allows for patterns that change over time; 2) BEATLEX is an online algorithm; 3) BEATLEX uses a novel vocabulary-based approach; importantly, this difference allows it to robustly capture arbitrarily complex patterns to summarize the data.

Table 1: Comparison of related approaches. 'AR++' refers to AR and its extensions (ARIMA, etc). 'Changing Patterns' refer to modelling sequences with patterns that change over time. 'Non-linear' refers to sequences with non-linear dynamics.

	AR++ [?]	HMM++ [?,?,?]	Discord [?,?]	Clustering [?,?]	AutoPlait [?]	RegimeCast [?]	BEATLEX
Segmentation		✓			✓		✔
Anomaly Detection			✓	✓			✔
Forecasting	✓	✓				✓	✔
Pattern Discovery		✓		✓	✓		✔
Non-linear			✓	✓		✓	✔
Changing Patterns							✔
Online							✔

3 Problem Definition

Preliminaries Table 2 includes the main definitions and symbols used in this paper.

We first introduce the Minimum Description Length (MDL) principle, which will allow us to define what a good summarization is. The MDL principle states that the best representation for some data is the one that leads to the best compression of the data. Formally, given data X, the MDL principle states that we should find the model M to minimize the **description length** of X, which is defined as $\text{Cost}(M) + \text{Cost}(X|M)$, where $\text{Cost}(M)$ is the number of bits needed to encode the model M, and $\text{Cost}(X|M)$ is the number of bits needed to encode the data given model M. The full expression for these costs depends on the type of model used, and will be given later, in Eq. (1).

Based on this cost function, we can formally define our summarization problem:

Problem 1 (Summarization). Given $(X_i)_{i=1}^m$, a real-valued time series of length m, find a model M to minimize the description length, $\text{Cost}(M) + \text{Cost}(X|M)$.

We next define our model M, and explain how it is used to compress the data.

4 Model

Figure 3 illustrates our vocabulary-based model for a time series X. It consists of:

- **Vocabulary:** the 'vocabulary terms' $V_1, \ldots, V_k$ are short time series patterns which will be used to explain segments of the actual data. For example, one pattern may represent normal heartbeats, while another may represent abnormal heartbeats.
- **Segmentation:** this describes how X is split into continuous segments of time. We represent the segments using intervals $[a_1, b_1], \ldots, [a_n, b_n]$, where $a_1 = 1, b_n = m$, and $b_i + 1 = a_{i+1}$ for $i = 1, \ldots, n-1$.

Symbol	Interpretation
X	Real-valued input time series
$X_{a:b}$	Subsequence of X from index a to b (inclusive)
m	Length of time series X
V_i	ith vocabulary term
n_i	Length of V_i
k	Number of vocabulary terms
n	Number of segments
a_i	Start of ith segment
b_i	End of ith segment
X^i	ith segment, i.e. $X_{a_i:b_i}$
$z(i)$	Assignment variable for ith segment
N_i	Number of segments assigned to vocabulary term i
$C(\cdot)$	Description cost, i.e. no. of bits needed to describe a parameter
C_F	Number of bits for encoding a floating point number
MDTW	Modified DTW distance (see Definition 1)
s_{min}, s_{max}	Minimum and maximum width of a segment
k_{max}	Maximum vocabulary size
w	Width of Sakoe-Chiba band for DTW [**?**]

Table 2: Symbols and Definitions

– **Assignment variables:** these describe which vocabulary term is used to encode each segment. For each i, the assignment variable $z(i)$ means that the $z(i)$th vocabulary term (i.e. $V_{z(i)}$) is used to encode segment i.

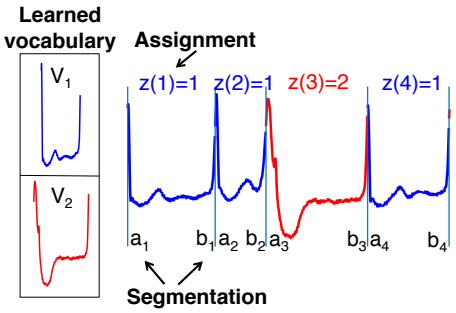


Fig. 3: Illustration of our summarization model. The data is broken into segments, and the assignment describes how each segment is described using the vocabulary.

5 Optimization Objective

In this section we explain our optimization objective, which is based on minimizing the description length of the model, and the data given the model.

5.1 Model Cost

From our model definition in Section 4, the parameters in the model are the vocabulary size k, the vocabulary sequences $V_1, \ldots, V_k$, the segmentation intervals $[a_1, b_1], \ldots, [a_n, b_n]$, and the assignment variables $z(1), \ldots, z(n)$. Let $C(\cdot)$ denote the number of bits required to store a parameter. As preliminaries: first, encoding an arbitrary integer requires $\log^* k$ bits.[4] Second, encoding a discrete variable taking N possible values requires $\log_2 N$ bits. In the rest of this paper, all logarithms will be base-2.

The model cost consists of:

- **Vocabulary size:** storing the positive integer k requires $\log^* k$ bits.
- **Vocabulary:** storing V_i requires $n_i \times C_F$ bits, where C_F is the number of bits needed to encode a floating point number.[5]
- **Segmentation:** for the segmentation $[a_1, b_1], \ldots, [a_n, b_n]$, it is sufficient to store $b_1, \ldots, b_{n-1}$, since these completely determine the segmentation. Each b_i takes m possible values. Hence, the total number of bits required is $(n-1)\log(m)$.
- **Assignment variables:** there are n such variables, each taking k possible values. Hence, the number of bits required is $n \log(k)$.

In total, the number of bits needed to store the model is:

$$\text{Cost}(M) = \underbrace{\log^*(k)}_{\text{cost of } k} + \underbrace{C_F \sum_{i=1}^{k} n_i}_{\text{vocab. cost}} + \underbrace{(n-1)\log(m)}_{\text{segmentation cost}} + \underbrace{n \log(k)}_{\text{assign. cost}}$$

5.2 Data Cost

How do we encode X given this model? Consider each segment X^i, and its corresponding vocabulary term $V = V_{z(i)}$. We want to encode X^i such that its encoding cost is low if it is similar to V, where this similarity should allow for slight distortions in shape. To achieve this, we encode X^i based on a modification of Dynamic Time Warping (DTW). Recall that given two sequences A and B, DTW aligns them while allowing for **expansions** on either sequence (e.g. if an entry of A is matched to two entries of B, we say that entry of A was 'expanded' once).

We modify DTW by adding penalties for each expansion. Similar penalized variants of DTW exist [**?**]; however, since we use an MDL framework, our choice of penalties has a natural interpretation as the number of bits needed to describe X^i in terms of V.

[4] Here, $\log^*$ is the universal code length for positive integers [**?**].

[5] We use $C_F = 8$, following [**?**].

Definition 1 (Modified DTW). *Given two sequences A and B, MDTW(A, B) modifies regular DTW by adding a penalty of $\log n_A$ for each expansion to A and $\log n_B$ for each expansion to B, where n_A and n_B are the lengths of A and B.*

The number of bits needed to describe X^i in terms of V is given by the MDTW cost: i.e. $C(X^i|V) = \text{MDTW}(X^i, V)$. To see this, note that the penalty, $\log n_A$ for expansions to A, exactly describes the number of bits needed to encode an expansion to A, since each expansion is of one of n_A possible entries. Thus the penalties capture the number of bits needed to encode expansions. In addition, we need to encode the remaining errors after warping; assuming these errors are independently Gaussian distributed, based on Huffman coding, their encoding cost in bits is their negative log likelihood under the Gaussian model [**?**], which is the standard squared-error DTW cost.

5.3 Final Cost Function

Combining our discussion so far, the description length cost under model M is:

$$\begin{aligned} f(M) &= \text{Cost}(M) + \text{Cost}(X|M) \\ &= \underbrace{\log^*(k)}_{\text{cost of } k} + \underbrace{C_F \sum_{i=1}^{k} n_i}_{\text{vocab. cost}} + \underbrace{(n-1)\log(m)}_{\text{segmentation cost}} + \underbrace{n \log(k)}_{\text{assign. cost}} + \underbrace{\sum_{i=1}^{n} \text{MDTW}(X^i, V_{z(i)})}_{\text{data cost}} \end{aligned} \quad (1)$$

6 BEATLEX Summarization Algorithm

6.1 Overview

In this section, we describe our algorithm for learning vocabulary terms, a segmentation, and an assignment, to minimize description length. From a high level, we first generate the initial vocabulary term (*New Vocabulary Term Generation*). Then, starting at the beginning of the time series, our algorithm repeatedly finds the closest match between any existing vocabulary term and a prefix of the time series (*Best Vocabulary-Prefix Match*). If a sufficiently good match is found, it assigns this segment to this vocabulary term (*Vocabulary Merge*). Otherwise, it creates a new vocabulary term (*New Vocabulary Term Generation*).

6.2 Best Vocabulary-Prefix Match

Assume that the algorithm has processed the time series up to time i so far, and the current vocabulary it has learned is $V_1, \ldots, V_k$. The next key subroutine the algorithm uses is to find the best match between any vocabulary term and any **prefix** of the current time series, i.e. $X_{i+1:i+s}$, for some s with $s_{min} \leq s \leq s_{max}$, where s_{min} and s_{max} are lower and upper bounds of the allowed segment length. We choose the best vocabulary term index j^* and the best prefix length s^* by minimizing **average encoding cost**:

$$j^*, s^* = \arg\min_{j,s} \frac{C(X_{i+1:i+s}|V_j)}{s} = \arg\min_{j,s} \frac{MDTW(X_{i+1:i+s}, V_j)}{s} \quad (2)$$

Algorithm 1: BEATLEX *summarization algorithm*

Input : Time series X
Output: Vocabulary $V_1, \ldots, V_k$, assignments z, segmentation S.

$k = 0$;
$i = 1$; // current position
while $i < m$ **do**
  ▷Find best vocabulary-prefix match:
  $j^*, s^* = \arg\min_{j,s} \frac{C(X_{i+1:i+s}|V_j)}{s}$; ▷see Section 6.2
  ▷If using existing vocab. has lower cost than creating new vocab. term:
  if $C(X_{i+1:i+s^*}|V_{j^*}) < C_F \cdot s^*$ *or* $k = k_{max}$ **then**
    ▷Use existing vocabulary term:
    V_{j^*} = VOCABMERGE$(X_{i+1:i+s^*}, V_{j^*})$); ▷see Section 6.3
    Append j^* to z;
  else
    ▷Create new vocabulary term:
    s^* = NEWVOCABLENGTH$(X, i, (V_1, \ldots, V_k))$; ▷see Section 6.4
    $V_{k+1} = X_{i+1:i+s^*}$;
    Append $k+1$ to z;
    $k = k + 1$;
  Append $[i+1 : i+s^*]$ to S;
  $i = i + s^*$;

Dividing by s allows us to compare fairly between different prefix lengths, by finding the prefix that can be encoded most efficiently per unit length.

How do we efficiently perform the minimization in (2)? Consider a single vocabulary term; we have to compare it against $s_{max} - s_{min} + 1$ choices of prefixes, and running a separate MDTW computation for each would be very slow. It turns out that we can minimize across all these subsequences optimally in a single MDTW computation. Across k vocabulary terms, this gives a total of k MDTW computations:

Theorem 1. *The best j^*, s^* minimizing* (2) *can be found in k MDTW computations, and requires $O(k \cdot w \cdot s_{max})$ time.*

Proof. For each j, consider computing the MDTW on V_j and $X_{i+1:i+s_{max}}$. Each entry of the DTW matrix computed by the dynamic programming DTW algorithm encodes the minimum DTW cost between each prefix of V_j and each prefix of $X_{i+1:i+s_{max}}$. Hence, the last $s_{max} - s_{min} + 1$ entries of the last row of the DTW matrix contains the minimum MDTW distance between V_j and each of the prefixes $X_{i+1:i+s}$ for $s_{min} \leq s \leq s_{max}$. Our algorithm can extract these values, divide them by their prefix lengths s, and choose the one minimizing average encoding cost. This requires k MDTW computations, one for each vocabulary term; each MDTW computation requires $O(w \cdot s_{max})$ time, using a Sakoe-Chiba band of width w [**?**]. ■

6.3 Vocabulary Merge (VOCABMERGE)

Now assume we have computed the best vocabulary-prefix match. If the resulting average encoding cost is less than C_F, then since generating a new vocabulary term would require C_F bits per unit length, the most efficient choice is to encode $X_{i+1:i+s^*}$ using the existing vocabulary V_{j^*}, so our algorithm makes this choice (Line 7). Otherwise, it creates a new vocabulary term (see Section 6.4).

After encoding $X_{i+1:i+s^*}$ using V_{j^*}, we would like to update V_{j^*} to make it an 'average' of the subsequences it encodes, to allow our algorithm to keep track of patterns that change over time. We use a *running average* approach, in which we keep track of how many subsequences have been assigned to V_{j^*} so far (call this number N_{j^*}). Intuitively, we should replace V_{j^*} by a weighted average of $X_{i+1:i+s^*}$ and itself, with weight of $\frac{1}{1+N_{j^*}}$ given to $X_{i+1:i+s^*}$, and $\frac{N_{j^*}}{1+N_{j^*}}$ given to the current value of V_{j^*}; this makes sense since the current value of V_{j^*} is an average of the N_{j^*} subsequences currently assigned to it. However, rather than using a straightforward average, we first run the MDTW algorithm in order to align $X_{i+1:i+s^*}$ to V_{j^*} before averaging it with V_{j^*}. Straightforward averaging would adversely affect the sequence shape, e.g. if both sequences have sharp peaks in slightly different locations, the average would have two peaks. Aligning the sequences first avoids this problem.

6.4 New Vocabulary Term Generation (NEWVOCABLENGTH)

In this case our algorithm chooses to generate a new vocabulary term V_{k+1}. To select the ideal length n_{k+1}, we iterate over all possible n_{k+1}. For each, we then use *Best Vocabulary-Prefix Match* to compute the average encoding cost of the *next* prefix after $X_{i+1:i+n_{k+1}}$, matched to any of the vocabulary terms (including V_{k+1}). The final length n_{k+1} chosen is the one that results in the lowest such average encoding cost.

6.5 Computation Time and Memory

BEATLEX is linear in the length m of the time series:

Lemma 1. BEATLEX *runs in* $O(m \cdot k \cdot w \cdot s_{max}/s_{min} + k^2 \cdot w \cdot s_{max}^2)$ *time.*

Proof. As shown in Theorem 1, each vocabulary-prefix match step takes $O(k \cdot w \cdot s_{max})$ time. We perform this step up to once per segment (not including vocabulary term generation), so at most once every s_{min} steps, taking $O(m \cdot k \cdot w \cdot s_{max}/s_{min})$ in total. The vocabulary term generation step runs k times, each trying at most s_{max} candidates taking $O(k \cdot w \cdot s_{max})$ each time, for a total of $O(k^2 \cdot w \cdot s_{max}^2)$. ■

Note that typically m greatly exceeds the other variables, the $O(k^2 \cdot w \cdot s_{max}^2)$ term is typically negligible. Figure 5c verifies the linear scalability of our algorithm in practice.

Lemma 2. BEATLEX *does not require the past history of the time series, and hence requires only bounded memory.*

Proof. We verify from Algorithm 1 that BEATLEX never needs to access the past history of the time series, and operates on incoming time series values while only keeping track of its vocabulary, storing at most its k_{max} vocabulary terms at any time, which requires bounded memory. ■

6.6 Extensions

Multidimensional Time Series: Figure 6 shows our algorithm applied on a 2-lead (i.e. 2-dimensional) ECG. To handle multidimensional time series, only a small change is needed: now, our vocabulary terms are likewise multidimensional. When encoding a subsequence using a vocabulary term, we use the dth dimension of the vocabulary term to encode the dth dimension of the subsequence. The description length of encoding a subsequence is then the sum of description lengths of encoding each individual dimension separately. The rest of our BEATLEX algorithm applies without change.

Anomaly Detection: Figure 1 reports the vocabulary terms with their frequencies in the data; PVCs are the rarer pattern. The most straightforward kind of anomaly are patterns that occur exactly once; we can easily detect these by selecting vocabulary terms that are only used to encode a single segment. More generally, we may also wish to detect *rare* patterns such as the PVCs in Figure 1. We can easily detect this kind of anomaly (as well as the former type) by ranking the vocabulary terms by how many segments they encode, and returning those encode the fewest.

Forecasting: How do we forecast future values of the time series, as done in Figure 1? Based on our learned segment labels, we learn Markov Models of orders $0, 1, \ldots, r_{max}$ using the usual maximum likelihood approach [**?**]. Here the 0th order model simply ignores the past and forecasts the most frequent label. To forecast the next segment label, we first try to use the highest (r_{max}) order model, but repeatedly drop to the next lower order if the sequence of the last r labels (where r is the current order) has not been seen in the past. This process continues to forecast as many segment labels as we need. Our forecasts for the actual data are the associated vocabulary terms.

Automatic Parameter Setting: our MDL objective (5) provides an easy way to choose parameters automatically, via grid search minimizing the description length cost. To keep the algorithm fast, we only do this once per type of data (e.g. on a single ECG sequence), then fix those values.

7 Experiments

We design experiments to answer the following questions:

- **Q1. Labelling Accuracy:** how accurate are the labels returned by BEATLEX?
- **Q2. Forecasting Accuracy:** how accurately does it forecast future data?
- **Q3. Scalability:** how does the algorithm scale with the data size?
- **Q4. Interpretability:** are the results of the algorithm interpretable by the user?

Our code, links to datasets, and experiments are publicly available at `www.andrew.cmu.edu/user/bhooi/beatlex`. Experiments were done on a 2.4 GHz Intel Core i5 Macbook Pro, 16 GB RAM running OS X 10.11.2.

7.1 Data

We evaluate our algorithm on ECG sequences from the MIT-DB Arrythmia Database[6] (MITDB) [**?**], as well as motion capture data from the CMU motion capture database[7].

MITDB Dataset The MITDB dataset contains 48 half-hour ECG recordings from test subjects from Beth Israel Hospital. Each recording consists of two ECG sequences, or 'leads,' at 360 time ticks per second, for a total of $650,000$ ticks. Ground-truth annotations by two or more independent cardiologists indicate the position and type of each heartbeat; disagreements were resolved to obtain the annotations.

CMU Motion Capture Dataset The dataset consists of motion-captured subjects performing various actions. Each recording is a 64 dimensional vector (representing the subject's body parts), of which we chose 4 dimensions (left-right arms and legs). The recording lasts 50 seconds with 120 ticks per second, for a total of 6000 time ticks.

7.2 Q1: Labelling Accuracy

Recall that BEATLEX labels each time tick based on which vocabulary term it was assigned to. We evaluate this by comparing it to the ground truth labelling using standard clustering metrics: Adjusted Rand Index [**?**] and Normalized Mutual Information [**?**]. Due to the fairly large number of ECG sequences and trials, we subset time series to 5000 time ticks, and do the same for our forecasting tests in the next subsection.

Baselines The baselines are Hidden Markov Models (HMMs) [**?**] and Autoplait [**?**], which identifies 'regimes' in time series data using a hierarchical HMM-based model.

Figure 4 shows the labelling accuracy of BEATLEX, HMMs and Autoplait, averaged across all 48 ECGs. Under both metrics, BEATLEX clearly outperforms both baselines. The key difference is that HMM-based methods (including Autoplait) have difficulty accurately representing the complex and nonlinear ECG patterns based on a model only using state transitions. In contrast, BEATLEX treats entire vocabulary terms as 'building blocks,' and hence has no particular difficulty handling complex patterns.

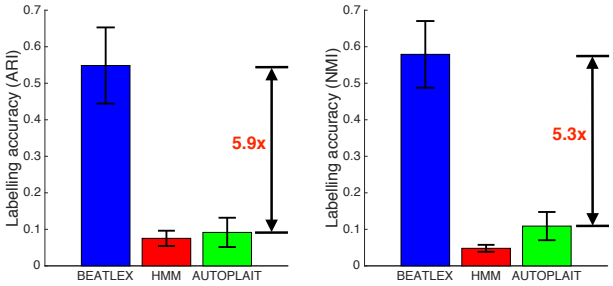
7.3 Q2: Forecasting Accuracy

We now evaluate BEATLEX in terms of forecasting future data. For each ECG sequence, the last 1000 time ticks (approximately 3 seconds) are hidden from the algorithm, and the algorithm forecasts this data. We compare the forecasts to the true values using Root Mean Squared Error (RMSE), as well as Dynamic Time Warping (DTW) distance.

Baselines As baselines, we use the classical ARIMA [**?**] algorithm, as well as the more recent TBATS [**?**] forecasting algorithm. We select the ARIMA order using AIC.

Figure 5 shows the forecasting error of BEATLEX compared to the baselines (lower is better). BEATLEX outperforms the baselines according to both metrics, but particularly under DTW distance. This difference between metrics is not surprising: RMSE is extremely sensitive to small temporal variations, e.g. of the spikes present in ECG data.

[6] https://physionet.org/cgi-bin/atm/ATM?database=mitdb
[7] http://mocap.cs.cmu.edu/



(a) Labelling accuracy (ARI) (b) Labelling accuracy (NMI)

Fig. 4: **Labelling accuracy (higher is better):** BEATLEX outperforms baselines, according to the Adjusted Rand Index (ARI) and Normalized Mutual Information (NMI) metrics, averaged across ECG sequences. Error bars indicate one standard deviation.

7.4 Q3: Scalability

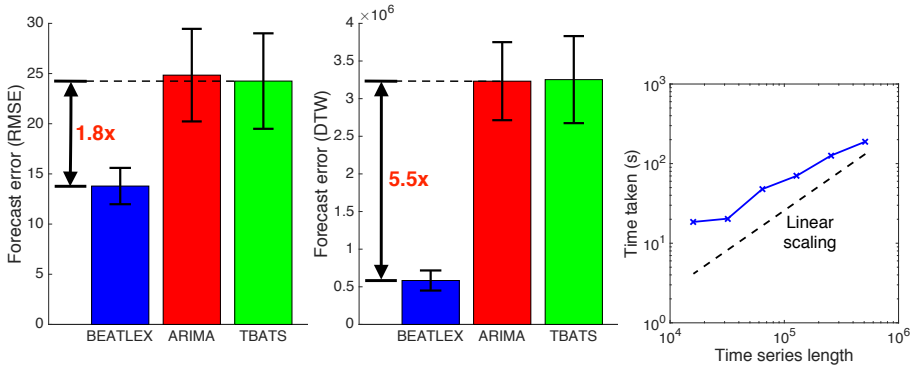
Figure 5c verifies the linear scalability of BEATLEX. We vary the length of a subset of an ECG sequence, and plot running time against length. The plot is around parallel to the main diagonal on a log-log plot, indicating linear growth. BEATLEX scales to large datasets, running on data of length $512,000$ in less than 200s.

7.5 Q4: Discoveries and Interpratability

In this section, we show that BEATLEX automatically discovers interpretable, medically relevant patterns. In Figure 1, the learned vocabulary terms can be easily and accurately matched with medically recognized patterns, 'normal sinus rhythm' and 'premature ventricular contraction' (PVC) respectively. Going beyond individual heartbeats, our algorithm also successfully learns a pattern present in Figure 1 known as 'quadrigeminy' in which PVC beats show up approximately every 4 beats. Medically, these patterns occur because the heart re-polarizes after a PVC, during which normal beats occur [**?**].

Multidimensionality: Section 6.6 explains that our algorithm handles multidimensional time series by learning multidimensional vocabulary terms. Figure 6 illustrates this. The two vocabulary terms still accurately correspond to normal sinus rhythm and PVCs respectively, except that each now has a multidimensional vocabulary term.

Motion Capture Dataset: our algorithm is generalizable to other time series datasets. Figure 2 shows its results on motion capture data of a basketball player. Despite significant variation in the width and shape of a pattern, our algorithm still successfully learns, segments and labels vocabulary terms corresponding to dribbling, shooting and two-handed dribbling.



(a) Forecast error with RMSE (b) Forecast error with DTW (c) Scalability

Fig. 5: **Forecast error (lower is better):** BEATLEX outperforms competing baselines in forecast accuracy averaged across all ECG time series, according to the standard RMSE and Dynamic Time Warping (DTW) distance metrics. **Scalability**: BEATLEX scales linearly, shown by growth parallel to the dotted diagonal on a log-log plot.

8 Conclusion

We presented the BEATLEX algorithm, a novel vocabulary-based approach designed to handle time series with patterns. It robustly learns interpretable vocabulary terms, in the face of possibly nonlinear patterns, distortions, and an unknown segmentation. BEATLEX is:

- **Fast and online:** BEATLEX takes *linear* time, *bounded* memory and constant update time per data point.
- **Effective:** BEATLEX outperforms existing algorithms in labelling accuracy by *5.3* times, and forecasting accuracy by *1.8* times.
- **Principled and parameter-free:** BEATLEX is fit using the Minimum Description Length (MDL) principle, and automatically tunes its parameters.
- **General:** BEATLEX applies to any domain of time series data, and multidimensional time series.

Reproducibility: Our code, links to datasets, and experiments are publicly available at `www.andrew.cmu.edu/user/bhooi/beatlex`.

Acknowledgments: This material is based upon work supported by the National Science Foundation under Grants No. CNS-1314632, IIS-1408924, and by the Army Research Laboratory under Cooperative Agreement Number W911NF-09-2-0053, by the Image Analysis and Machine Learning Platform -ERI/TIC/0028/14 grant, and by Beijing NSF No. 4172059. Shenghua Liu is also supported by the scholarship from China Scholarship Council. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, or other funding parties. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

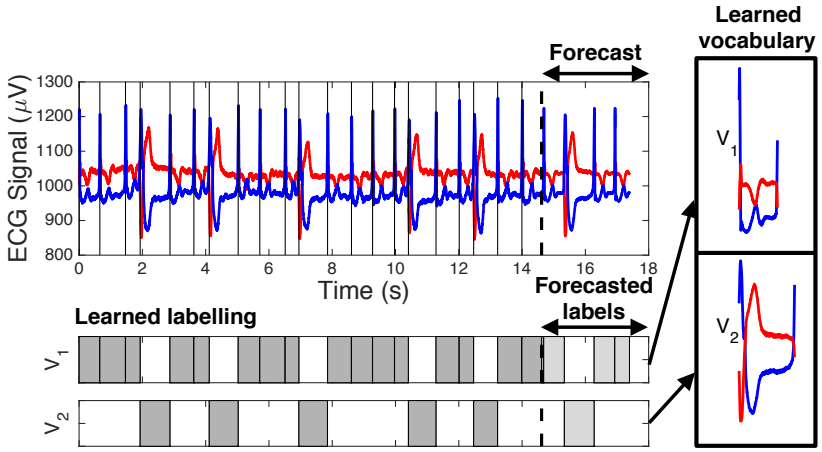


Fig. 6: **Accurate multidimensional summarization and forecasting:** an ECG sequence, with its two dimensions in red and blue. BEATLEX accurately segments the data, learns the two types of heartbeats, and forecasts future data.

References

1. Basu, S., Meckesheimer, M.: Automatic outlier detection for time series: an application to sensor data. Knowledge and Information Systems 11(2), 137–154 (2007)
2. Baum, L.E., Petrie, T.: Statistical inference for probabilistic functions of finite state markov chains. The annals of mathematical statistics 37(6), 1554–1563 (1966)
3. Box, G.E., Jenkins, G.M.: Time Series Analysis: Forecasting and Control. Holden-Day Inc., San Francisco, revised edn. (1976)
4. Cover, T.M., Thomas, J.A.: Elements of information theory. John Wiley & Sons (2012)
5. De Livera, A.M., Hyndman, R.J., Snyder, R.D.: Forecasting time series with complex seasonal patterns using exponential smoothing. Journal of the American Statistical Association 106(496), 1513–1527 (2011)
6. Keogh, E.: Exact indexing of dynamic time warping. In: Proceedings of the 28th international conference on Very Large Data Bases. pp. 406–417. VLDB Endowment (2002)
7. Keogh, E., Chu, S., Hart, D., Pazzani, M.: An online algorithm for segmenting time series. In: Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on. pp. 289–296. IEEE (2001)
8. Keogh, E., Lin, J., Fu, A.: Hot sax: Efficiently finding the most unusual time series subsequence. In: Data mining, fifth IEEE international conference on. pp. 8–pp. Ieee (2005)
9. Keogh, E., Lin, J., Truppel, W.: Clustering of time series subsequences is meaningless: Implications for previous and future research. In: Data Mining, 2003. ICDM 2003. Third IEEE International Conference on. pp. 115–122. IEEE (2003)
10. Letchner, J., Re, C., Balazinska, M., Philipose, M.: Access methods for markovian streams. In: Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on. pp. 246–257. IEEE (2009)
11. Li, L., McCann, J., Pollard, N.S., Faloutsos, C.: Dynammo: Mining and summarization of coevolving sequences with missing values. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 507–516. ACM (2009)

12. Mathioudakis, M., Koudas, N., Marbach, P.: Early online identification of attention gathering items in social media. In: Proceedings of the third ACM international conference on Web search and data mining. pp. 301–310. ACM (2010)
13. Matsubara, Y., Sakurai, Y.: Regime shifts in streams: Real-time forecasting of co-evolving time sequences. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 1045–1054. ACM (2016)
14. Matsubara, Y., Sakurai, Y., Faloutsos, C.: Autoplait: Automatic mining of co-evolving time sequences. In: Proceedings of the 2014 ACM SIGMOD international conference on Management of data. pp. 193–204. ACM (2014)
15. Matsubara, Y., Sakurai, Y., Faloutsos, C.: The web as a jungle: Non-linear dynamical systems for co-evolving online activities. In: Proceedings of the 24th International Conference on World Wide Web. pp. 721–731. ACM (2015)
16. Moody, G.B., Mark, R.G.: The impact of the mit-bih arrhythmia database. IEEE Engineering in Medicine and Biology Magazine 20(3), 45–50 (2001)
17. Rakthanmanon, T., Keogh, E.J., Lonardi, S., Evans, S.: Mdl-based time series clustering. Knowledge and information systems 33(2), 371–399 (2012)
18. Rand, W.M.: Objective criteria for the evaluation of clustering methods. Journal of the American Statistical association 66(336), 846–850 (1971)
19. Rissanen, J.: A universal prior for integers and estimation by minimum description length. The Annals of statistics pp. 416–431 (1983)
20. Sakoe, H., Chiba, S.: Dynamic programming algorithm optimization for spoken word recognition. IEEE transactions on acoustics, speech, and signal processing 26(1), 43–49 (1978)
21. Shieh, J., Keogh, E.: i sax: indexing and mining terabyte sized time series. In: Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 623–631. ACM (2008)
22. Strehl, A., Ghosh, J.: Cluster ensembles—a knowledge reuse framework for combining multiple partitions. Journal of machine learning research 3(Dec), 583–617 (2002)
23. Sun, H., Lui, J.C., Yau, D.K.: Distributed mechanism in detecting and defending against the low-rate tcp attack. Computer Networks 50(13), 2312–2330 (2006)
24. Wagner, G.S.: Marriott's practical electrocardiography. Lippincott Williams & Wilkins (2001)
25. Wang, P., Wang, H., Wang, W.: Finding semantics in time series. In: Proceedings of the 2011 ACM SIGMOD International Conference on Management of data. pp. 385–396. ACM (2011)
26. Wang, X., Mueen, A., Ding, H., Trajcevski, G., Scheuermann, P., Keogh, E.: Experimental comparison of representation methods and distance measures for time series data. Data Mining and Knowledge Discovery pp. 1–35 (2013)
27. Yankov, D., Keogh, E., Rebbapragada, U.: Disk aware discord discovery: finding unusual time series in terabyte sized datasets. In: Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on. pp. 381–390. IEEE (2007)