原著論文 Original Paper

Time Series Classification via Topological Data Analysis

Yuhei Umeda FUJITSU LABORATORIES LTD.
umeda.yuhei@jp.fujitsu.com

keywords: time series classification, dynamical system, topological data analysis, convolutional neural network

Summary

This paper focuses on a classification problem for volatile time series. One of the most popular approaches for time series classification is dynamic time warping and feature-based machine learning architectures. In many previous studies, these algorithms have performed satisfactorily on various datasets. However, most of these methods are not suitable for chaotic time series because the superficial changes in measured values are not essential for chaotic time series. In general, most time series datasets include both chaotic and non-chaotic time series; thus, it is necessary to extract the more essential features of a time series. In this paper, we propose a new approach for volatile time series classification. Our approach generates a novel feature by extracting the structure of the attractor using topological data analysis to represent the transition rules of the time series. As this feature represents the essential property of systems of the time series, our approach is effective for both chaotic and non-chaotic types. We applied a learning architecture inspired by a convolutional neural network to this feature and found that the proposed approach improves performance in a human activity recognition problem by 18.5% compared with conventional approaches.

1. Introduction

Time series classification is a significant problem in many fields such as engineering and the medicine [Aggarwal 14]. Several algorithms for time series classification have already been proposed and have been able to deliver good performances.

The most popular approach is dynamic time warping (DTW), which measures the similarity between two time series by warping the time axis [Rakthanmanon 12]. This approach is especially effective when combined with the *k*-Nearest Neighbor method (*k*-NN). However, the DTW algorithm is affected by gaps in measurement timing, and it is difficult to expand to a multi-channel algorithm without a priori knowledge about the correlation between the variables [Bankó 12].

In this context, the feature-based machine learning approach has been actively studied. This approach uses models that are fitted on the border of features extracted from the time series. Conventional machine learning algorithms such as support vector machines (SVMs) and artificial neural networks (ANNs) are extensively used for generating the classifier. Standard features include statistical features such as the maximum/minimum value, mean value, variance and frequency information. Altun *et al.* and Barshan *et al.* [Altun 10a, Barshan 14] examined various machine learning algorithms using these features for human activity recognition and identified the classifier that produced the best performance. Recently, new frameworks of features using local segment information such as bag-of-words and bag-of-features inspired by techniques in Native Language Processing have been proposed [Baydogan 13, Baydogan 15, Wanga 13]. These techniques can outperform the DTW approach in many time series datasets. Moreover, deep learning has emerged as a family of learning models that automatically extracts the features of a dataset. In deep learning, a deep architecture with multiple layers is built up for automating feature design. Convolutional neural networks (CNNs) have achieved particularly noteworthy performance in the image recognition field. Yang *et al.* [Yang 15] applied the CNN architecture to time series classification directly and was able to improve the performance compared to the DTW approach in various time series datasets. Wang *et al.* [Wang 15] proposed a method that converts time series into images for applying the CNN architecture to image classification. This algorithm also delivered good performance in some time series datasets.

The above features are based on the waveform of time series values: that is, time series that belong to the same class have a similar waveform and similar range. How-

ever, there are many time series datasets containing components that belong to the same class but have different waveforms and values ranges. For example, chaotic time series within the same system are sensitive to the initial conditions in which a small change in one state of a deterministic nonlinear system can result in large differences in a later state [Sprott 03]. As result of this sensitivity, the statistical features also have sensitivity; thus, the above features of these chaotic time series are not suitable for classification.

For chaotic time series classification, Ali *et al.* [Ali 07] proposed features based on chaos theory, and improved the performance compared to statistical features in the classification of motion-sensor signals. However, the features they used are not useful for non-chaotic datasets because most of these features are determinate values.

Chaotic time series are included in various time series datasets. For example, social, economic, medicine, international relations [Levy 94], brain waves [Korn 03], sensor signals of human activities [Ali 07], and time series in many areas [Lines 05] include chaotic time series. In most fields, chaotic and non-chaotic time series are mixed, thereby making it difficult to distinguish chaos from non-chaos. This is why it is necessary to generate a classification algorithm that is compatible with both chaotic and non-chaotic time series.

In this paper, we consider the classification of time series that perform up-and-down motion in a short interval (volatile time series) including chaotic time series. Specially, we propose a new feature that is efficient for both chaotic and non-chaotic time series. Our concept is based on the idea that features are generated on the basis of transition rules (time evolution equations). We use an attractor to realize our concept, which is a common approach to representing a transition rule by a figure [Kantz 03, Sprott 03] in dynamical system theory.

Applying the machine learning architecture to attractors first requires us to convert the attractors into a form that is suitable for the architecture. In terms of classification, the most important element of an attractor is the topological information of a figure. Therefore, we developed a conversion method based on topological data analysis (TDA) [Carlsson 09] by using a framework to extract topological information from a dataset. We use a persistent homology, the main TDA technique [Edelsbrunner 08], to extract this information, and generate new features for a time series classification that we named the Betti sequence. Perea *et al* [Perea 14, Perea 15] extracted the periodicity information of time-series by applying persistent homology to an analogue of the attractor.

As the Betti sequence has a few characteristics that are different from the standard vectors and standard time series in terms of time series classification, we introduce a learning architecture suitable for the Betti sequence based on a CNN. Our contribution involves extracting the feature of the time series by applying persistent homology to attractor and classifying the features using the CNN.

For the purpose of validation, we apply our algorithm to a human activity recognition problem and compare it with conventional algorithms.

The rest of this paper is organized as follows. First, we provide the outline of our algorithm in Chapter 2. Next, we introduce the main tools of our algorithm, namely, attractor theory, topological data analysis and CNN architecture in Chapter 3, Chapter 4 and Chapter 6, respectively. In Chapter 5, we confirm the effect of preprocessing using synthetic data. The experimental results and analysis are given in Chapter 7. We conclude in Chapter 8 with a brief summary and mention of future work.

2. Our Classification Algorithm

In this chapter, we sketch the outline of our time series classification algorithm, shown in Figure 1. Our classification algorithm comprises preprocessing and learning parts. The preprocessing part, in which we convert a time series dataset into a new dataset suitable for application to a machine learning algorithm consists of two steps first, we convert the time series into a quasi-attractor that represents the transition rules of the corresponding system (see Chapter 3), and second, we convert the quasi-attractor into a new form termed a Betti sequence that is generated by extracting the topological information of the quasi-attractor (see Chapter 4). In the learning part, we construct a classifier based on a one-dimensional CNN using the Betti sequence dataset (see Chapter 6).

3. Analysis of Dynamical Systems

Many previous studies have proposed various features for time series classification. The most popular features are statistical values such as the maximum/minimum value, mean value, and frequency information (see [Altun 10a, Barshan 14]). Some frameworks of features using local segment information have recently been proposed [Baydogan 13, Wanga 13]. These features are effective for time series that repeat specific patterns, especially non-chaotic time series, but the effect of these features on chaotic time series with random patterns is small. For chaotic time series classification, some features peculiar to chaos, e.g.,

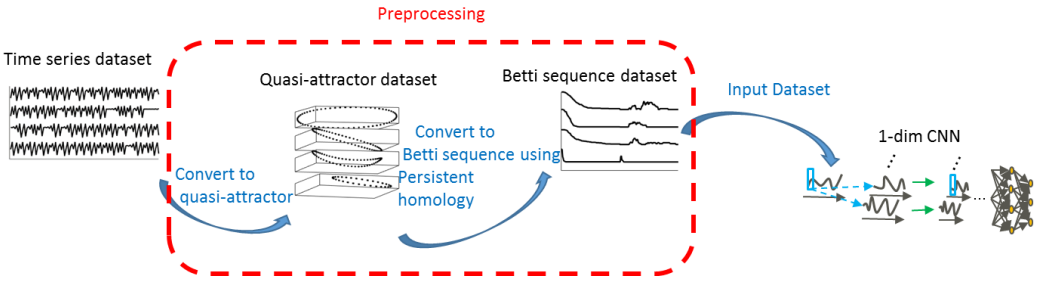


Fig. 1 Flow chart of proposed classification algorithm.

Lyapunov exponents, have been used with some success [Basharat 09]. However, these features do not generate a difference for non-chaotic time series.

Generally, distinguishing between chaos and non-chaos is difficult and observed time series datasets have mixed chaotic/non-chaotic time series. It is therefore desirable to eliminate the necessity to distinguish between chaos and non-chaos.

3·1 Difference Equation and Attractor

In this paper, we assume that an observed time series $\{x_1, \ldots, x_t\}$ $(x_i \in \mathbf{R})$ has a difference function that specifies its behavior

$$x_{k+1} = f(x_k, \ldots, x_1). \tag{1}$$

Generally, it would be possible to represent the transition rule of not only a chaotic but also many non-chaotic time series observed in nature by equation (1) [Basharat 09]. The original idea of our approach is to classify time series datasets based on this equation. However, it is very difficult to construct this equation from observed the time series data. Therefore we utilize an attractor, which has been used for the analysis of nonlinear dynamical systems [Basharat 09]. An attractor is a set of numerical values toward which a system described by (1) tends to evolve, for a wide variety of starting conditions of the system, embedded in d-dimensional space [Kantz 03]. Generally, many time series datasets include some time series that will have the same transition rule but different waveforms. In contrast, the attractors of a time series with similar transition rules resemble each other.

The idea of an attractor forms the foundation upon which the underlying chaotic system is modeled. However, because the attractor forms the shape of the difference equation (1) it represents the feature of not only chaotic but also non-chaotic systems with a difference equation. The attractor is therefore suitable for time series classification with the transition rules.

3·2 Quasi-attractor

As an attractor is the orbit to which the transition of system values converges, an attractor generally has infinite points. To generate an attractor from observed time series data with finite length is impossible. Moreover, it is difficult to find the dimension of the space in which the attractor is embedded.

The most popular approach for acquiring the information of an attractor from measured time series is a method using a quasi-attractor. The time series observations $\{x_0, x_1, \ldots, x_t\}$ are transformed into the phase space vectors $\mathbf{Z} = \{\mathbf{z_0}, \mathbf{z_1}, \ldots, \mathbf{z_{t'}}\}$ $(t' = t - (p-1)\tau + 1)$ through delay embedding. The delay vector is the vector generated by local information of the time series defined by

$$\mathbf{z_k} = [\mathbf{x_k}, \mathbf{x_{k+\tau}}, \ldots, \mathbf{x_{k+(p-1)\tau}}] \in \mathbf{R^p}, \tag{2}$$

where τ is the sampling lag and p is the embedding dimension. The quasi-attractor represents the set of delay vectors. An example of a quasi-attractor from a delay vector is given in Figure 2. The suitable embedding dimensions are different depending on the time series data. In terms of time series classification, however, it is unsuitable for transforming the information using different settings. Therefore we assign constant values to the parameters $\tau = 1$ and $p = 3$, in this paper.

A quasi-attractor is obtained as the point-cloud data. The key information of a quasi-attractor is the point arrangement.

4. Topological Data Analysis

The quasi-attractor has two primary characteristics: one, it is generated as a point cloud, thus it consists of sparse data, and two, the coordinate values of the point cloud do not have any meaning, which enhances the importance of the arrangement of the point cloud. Therefore, the format of phase space $\mathbf{Z}$ is not suitable to use as the input to the machine learning architecture.

Classification of the quasi-attractors, necessitiates the extraction of the topological information of the arrange-

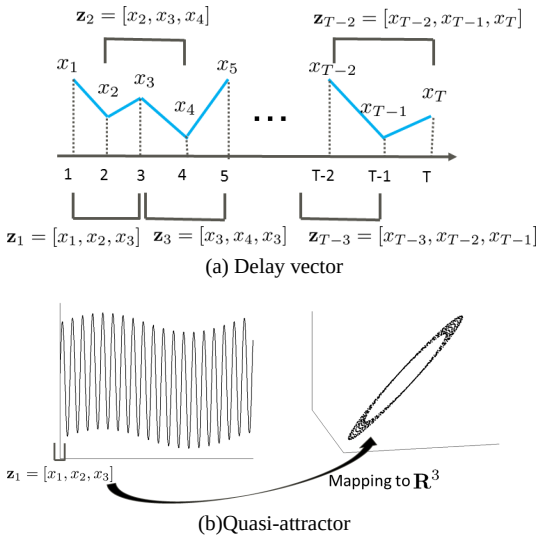


Fig. 2 (a)Delay vectors generated from observed time series. (b) Delay vectors map to three-dimensional Euclidean space. The set of points mapped from the delay vectors is the quasi-attractor.

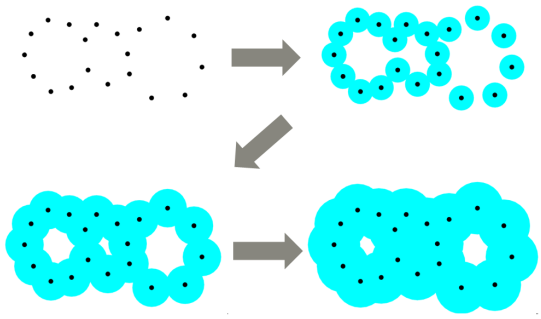


Fig. 3 Filtration of the expansion close ball. The first step includes the original point cloud, 20 zero-dimensional holes (connected components), and no holes of more than one-dimension. In the second step, there are 14 dead zero-dimensional holes and one alive one-dimensional hole. In the third step, there are five dead zero-dimensional holes and two alive one-dimensional holes. In the fourth step, there is one dead 1-dimensional hole (center hole).

ment of the point cloud. To this end, we propose a conversion method using persistent homology, the main technique of topological data analysis (TDA). TDA is a relatively new branch of applied mathematics that has been growing rapidly in the past decade. It provides a framework to extract the topological information from a dataset, being successful in coordinate-freeness, insensitive to particular metrics, dimension reduction and robustness against noise.

4·1 Persistent Homology

Persistent homology provides a strategy for constructing the topological information of point cloud data. In brief, homology finds "holes" in the point cloud structure. The fundamental idea of persistent homology is to construct a family of homology of the spaces $\mathbb{X}_\epsilon = \cup_{\mathbf{i}=\mathbf{1}}^{\mathbf{n}} \mathbb{B}(\mathbf{z_i}, \epsilon)$, where an expansion close ball $B(\mathbf{z_i}, \epsilon)$ with radius of $\epsilon > 0$ is centered around each point of the dataset $\mathbf{Z} = \{\mathbf{z_i}\}_{\mathbf{i=1}}^{\mathbf{m}}$.

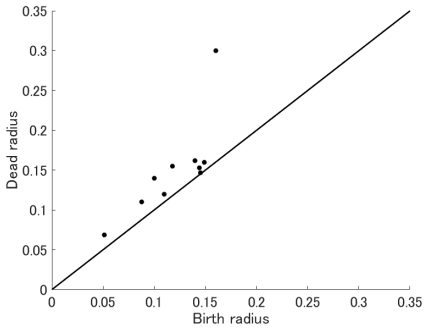
When the radius parameter ϵ is fixed, we can obtain the structure of $\mathbb{X}_\epsilon$ as the combination of zero-dimensional holes (= homeomorphic to a point/ connected element), one-dimensional holes (= homeomorphic to circle), two-dimensional holes (= homeomorphic to sphere) and higher dimensional holes. Persistent homology provides information on the birth and death of each dimensional hole in accordance with the radius ϵ. Here, we say the topological spaces X and Y are homeomorphic if there exists a function between X and Y that has the properties (1) bijection, (2) continuous, (3) inverse function is also continuous. Figure 3 shows an example of persistency of covering the space of a point cloud. Persistent homology is a method that extracts the feature of the shape of a point cloud by following up the change in the number of hole corresponding to the change of the radius parameter [Carlsson 09]. This enables us to extract information of the dynamics of time series by extracting the feature of the attractor using persistent homology. Refer to [Carlsson 09] for the mathematical definition of persistent homology.

Persistent homology has recently been applied to extensive areas such as sensor networks and protein classification. In the time series area, Perea *et al.* [Perea 14, Perea 15] applied it to extract the period of a time series. In this paper, we use persistent homology to extract features of the shape of the attractor that represent the transition rule of the time series.

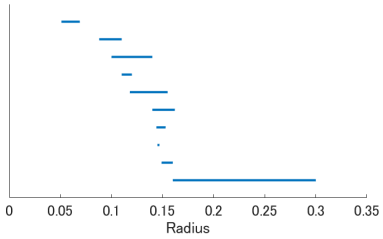
4·2 Feature of Time Series from Persistent Homology

The classical outputs of persistent homology are the persistent diagram and barcodes. The persistent diagram can be created by drawing a collection of points in the plane. Consider the extended plane $(\mathbf{R} \cup \{\infty\})^{\mathbf{2}}$ on which we represent the birth radius of a hole paired with the death of the hole radius as a point with two coordinates. Some of the classes may never die and are represented as points at infinity. The persistent barcode is essentially a multiset of intervals of the form $[a, b)$, where a and b are the birth and death radii, respectively. For one point cloud, the persistent diagrams and persistent barcodes of each dimensional hole are generated. Figure 4 shows an example of the persistent diagram and persistent barcodes of one-dimensional holes from a point cloud.

The persistent diagram and persistent barcodes provide crucial information. However, they are not suitable for



(a) Persistent diagram



(b) Barcode

Fig. 4 (a)Persistent diagram: the horizontal and vertical axis show the birth and death radii of one-dimensional holes, respectively. (b) Persistent barcodes: each line indicates the intervals from the birth radius to the death radius of each one-dimensional hole.

the input data of most machine learning techniques because the persistent diagram is a very sparse image and the number of components of the persistent barcode is not constant.

We therefore propose a new form of persistent homology output for the machine learning input. The important point of persistent homology is following the change of the number of holes corresponding to the change in the radius parameter. Thus, in this paper, we adopt the vectorization method that preserves the radius parameter by a setting that is unified for data. This new form corresponds to the radius of the expansion close ball and the number of holes of the structure at the radius (Betti number). We denote that $BN_d(r)$ is the number of d-dimensional holes (d-dimensional Betti numbers) of $\mathbb{X}_{\searrow}$. For generating finite length vector, we confine the radius parameter to $0 < r < E$, E is the finite value given as hyper parameter. This data is constructed by connecting vectors $BS_0, BS_1, \ldots, BS_n$ in ascending order, where n is the maximum value of the given usage dimension of homology. The i-th element of the respective vectors BS_d is the d-dimensional Betti number of $\mathbb{X}_{\beth * \mathbb{E}/\gg}$, that is $BS_d(i) = BN_d(i * E/m_d)$, where m_d is the given discretization mesh size (vector size) of BS_d. It is not necessary that $\{m_d\}_{d=0,\ldots,n}$ are same number in each dimension d. In this paper, we set a common value as $m_d = 300$ for all d. Figure 5 shows the conversion from the persistent barcodes to input data. In

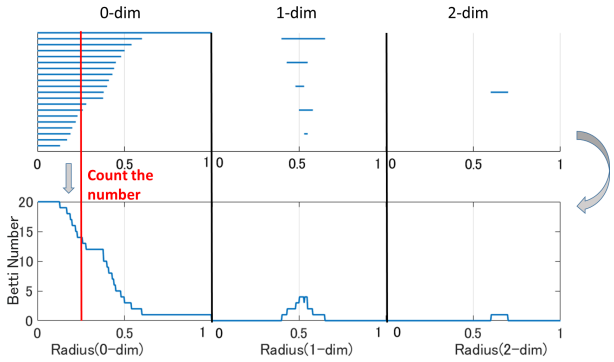


Fig. 5 Converting the persistent barcodes into Betti sequence.

this paper, we refer to this form as the Betti sequence. The vector length of the Betti sequence is $M = m_0 + \ldots + m_n$.

5. Synthetic Data

5·1 Synthetic Data

In this chapter, we confirm the effect of the proposed preprocessing algorithm using the following synthetic data.

Sin-I $\begin{cases} x_{k+1} = -0.56x_k - x_{k-1}, \\ x_0 = 0.5, x_1 = 0.9998, \end{cases}$

Sin-II $\begin{cases} x_{k+1} = -0.56x_k - x_{k-1}, \\ x_0 = 0.5, x_1 = 0.6999, \end{cases}$

Sin-III $\begin{cases} x_{k+1} = 0.56x_k - x_{k-1}, \\ x_0 = 0.5, x_1 = 0.966, \end{cases}$

Chaos-I $\begin{cases} y_{k+1} = 3.97y_k(1 - x_k), \ \ y_0 = 0.5, \\ x_k = 2.2y_k - 1.1, \end{cases}$

Chaos-II $\begin{cases} y_{k+1} = 3.97y_k(1 - x_k), \ \ y_0 = 0.2, \\ x_k = 2.2y_k - 1.1. \end{cases}$

The data Sin-I, Sin-II and Sin-III are prepared to confirm the influence on the amplitude and the period of the periodic wave. These data are the difference equations of the following functions:

Sin-I $x_k = 0.5\sin(1.6k) + 0.5,$

Sin-II $x_k = 0.2\sin(1.6k) + 0.5,$

Sin-III $x_k = 0.5\sin(1.2k) + 0.5.$

These difference equations are obtained by approximating the second derivatives of the corresponding differential equation, in case of Sin-I, $d^2x_k/dk^2 = -1.6^2x_k$, using the central difference with stride $\Delta k = 1$, that is, $d^2x_k/dk^2 = x_{k+1} - 2x_k + x_{k-1}$. We can confirm the influence of the amplitude by comparing Sin-I and Sin-II, whereas com-

paring Sin-I and Sin-III shows the influence of a period (frequency).

The data Chaos-I and Chaos-II are generated from the logistic map, one of the most well-known chaotic time series. These two time series have different waveforms due to the sensitivity of the initial condition, but they should be included in the same class in respect of classification based on time evolution equations.

5·2 Preprocessing Synthetic Data

Figure 6 shows the waveform (a), quasi-attractor (b) and Betti sequence (c) of the synthetic data.

In Figure 6(a), the figure on the left shows the waveform differences of the amplitude and period. These differences are difficult to distinguish by local segment information. The figure on the right shows the waveforms of chaotic time series. These waveforms are considerably different, thus it is difficult to classify Chaos-I and Chaos-II as being of the same class using local segment information and statistical information.

Figure 6(b) shows the quasi-attractor of the synthetic data. From Sin-I, Sin-II and Sin-III of Figure 6(b), we can observe that the difference in the amplitude and frequency of the time series are represented as the difference in the radius and angle of the ring. Moreover, the shapes of the chaotic data are almost the same. Therefore, we can classify these data based on appearance.

Figure 6(c) shows the Betti sequence of the synthetic data. The Betti sequences of Sin-I, Sin-II and Sin-III enable us to confirm that the Betti sequence can represent the differences in the amplitude and frequency of the time series. Moreover, we can observe that the Betti sequences from time series with the same evolution equation and different waveforms have forms that are similar to those of Chaos-I and Chaos-II in Figure 6(c).

6. Learning Architecture

In previous chapters, we discussed converting the problem of classification for time series into a problem of classification for the Betti sequence , which has the following two properties. The first property is that the radius parameters of the Betti sequence (cell numbers of the vector) do not depend on the point at which measurement of the time series started. In this respect, the Betti sequence differs from general time series, because the time parameters of time series generally depend on the starting point of measurements. For example, the vectors generated from the same time series with a different measurement starting time are different. This property is one of the reasons that validate our decision to carry out the conversion to the Betti sequence. The second property is that the difference in the Betti sequence of time series with different amplitudes represents the gap in the direction of the radius parameter.

⟨**Proposition 6·1**⟩ For two time series with different amplitude x_t and $\tilde{x}_t = ax_t (a > 0)$ and $d = 0, 1, \ldots, n$, the d-dimensional Betti numbers $BN_d(r)$ of closed ball space $\mathbb{X}_\diagdown$ from x_t and the Betti numbers $\tilde{BN}_d(r)$ of $\mathbb{X}_\diagdown$ from $\tilde{x}_t$ have the following relation:

$$\tilde{BN}_d(r) = BN_d(ar).$$

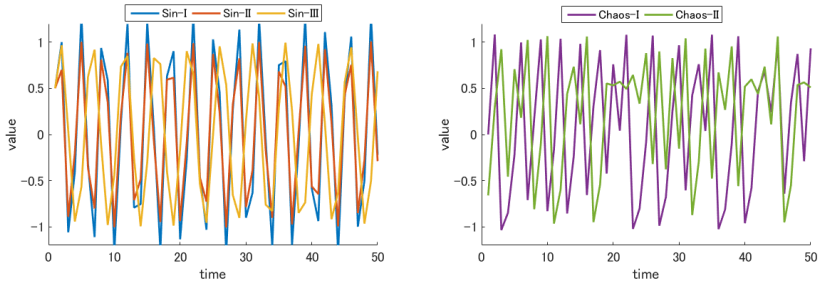
《**Proof**》 From equation (2), the phase space vectors $\mathbf{z_k}$ of x_t and $\tilde{\mathbf{z}}_k$ of $\tilde{x}_t = ax_t$ have the relation $\tilde{\mathbf{z}}_k = a\mathbf{z_k}$. Then the attractor of $\tilde{x}_t$ is the scaling image of the attractors of x_t, in which case the close ball space $\tilde{\mathbb{X}}_r$ of $\tilde{x}_t$ and $\mathbb{X}_{\supset\diagdown}$ of x_t are the scaling image. Thus, the Betti numbers of $\tilde{\mathbb{X}}_r$ and $\mathbb{X}_{\supset\diagdown}$ are the same.

From proposition 6·1, the Betti sequence BS of x_t and $\tilde{BS}$ of $\tilde{x}_t$ have the cell shift relation, that is $\tilde{BS}_d(i) = BS_d(\mathrm{ceil}(ai)) + \epsilon$, where ϵ is the discretization error. This sensitivity to scale of time series is very important property, because the scale of time series is very important information for classification. On the other hand, the cell shift relation is the difference of the classification problem of general vector data. If $a \approx 1$, it is very probable that both of the time series are classified in the same group. However, the Euclid distance of BS and $\tilde{BS}$ is relatively large. Hence, machine learning algorithms for simple vectors such as SVMs and artificial neural networks (ANNs) cannot provide sufficient performance. In image classification field, there is similar problem, that is the shift of the position of the subject. Therefore we consider applying convolutional neural network which is the very effective method for image classification to Betti sequences.

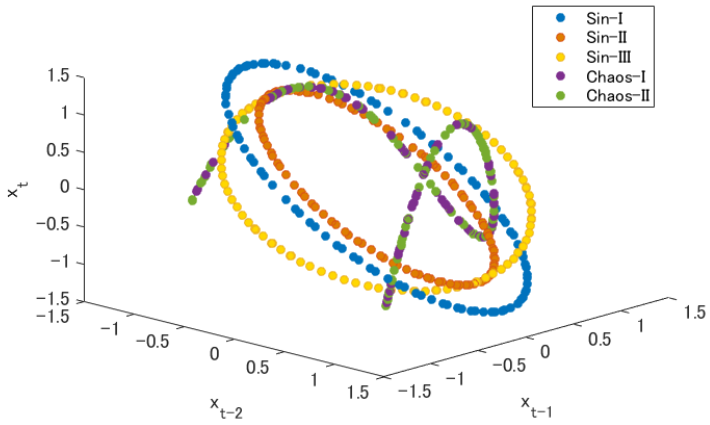
6·1 One-dimensional Convolutional Neural Network

The simple combination of *k*-nearest neighbor and dynamic time warping (DTW) could deliver a good classification performance in most domains because of the ability of this algorithm to resist the gap in the direction of the time parameters [Rakthanmanon 12]. However, in repect of applying this to the Betti sequence, the DTW distance between x_t and ax_t $(a >> 1)$ is also small. In case the scaling coefficient a is large, it is highly probable that the two time series belong to different groups. Therefore, DTW is not suitable for Betti sequence classification.

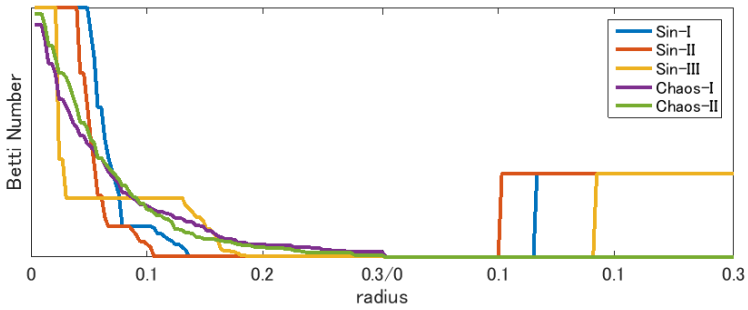
Convolutional neural networks (CNNs) comprise one or more convolutional layers (often with a subsampling step) followed by one or more fully connected layers as in a



(a) Waveforms of synthetic data



(b) Quasi-attractor of synthetic data



(c) Betti sequence of synthetic data

Fig. 6 Preprocessing of synthetic data.

standard multilayer neural network. The architecture of a CNN could achive the best performance for image classification , for which it is also important to preserve location information and resist the difference in the object size. We consider these properties to be similar to the gap of the radius parameter of the Betti sequence, so we consider CNN to be suitable for the classification of the Betti sequence.

One-dimensional CNN (1-CNN) architecture consists of a number of convolutional and subsampling layers optionally followed by fully connected layers. The input to a convolutional layer is M values, where M is the length of the Betti sequence. The convolutional layer will have k filters (or kernels) of size n, where n is smaller than the length of the Betti sequence. These filters emphasize the locally connected structure of the series to produce k feature maps of size $M - n + 1$. Each map is then subsampled typically with mean or max pooling over q contiguous cells, where q is given the subsampling unit size as hyperparameter. Either before or after the subsampling layer, an additive bias and sigmoidal nonlinearity is applied to each feature map. After the convolutional layers there may be any number of fully connected layers. The densely connected layers are identical to the layers in a standard multilayer neural network. Figure 7 illustrates a full layer in a CNN consisting of convolutional and subsampling sublayers.

6·2 Parallel One-dimensional CNN

When performing time series classification, we can also use multi-channel data, for example, sensors attached to the hands and legs. We can construct a multi-channel Betti sequence from multi-channel time series data by transforming the time series data of each channel to a Betti sequence.

Multi-channel Betti sequence classification requires us to modify the 1-CNN architecture. The simple extension involves connecting the multi-channel Betti sequence to the unit time series. In this case, the input layer has $M_1 + M_2 + \ldots + M_s$ units, where s is the number of channels and M_i is the length of the Betti sequence of the i-th channel.

Zheng *et al.* [Zheng 14] proposed a multi-channel deep CNN. Our similar concept architecture separates multivariate Betti sequences into univariate ones and performs feature learning on each univariate series individually. This architecture extracts the respective features of a multi-channel Betti sequence and then combines their features and calculates a score for each label. Figure 8 shows the parallel one-dimensional neural network architecture.

The difference between the above two architectures is that their filters for a multi-channel Betti sequence are either the same or different. Clearly, the parallel architecture can represent a more detailed model.

6·3 Learning Algorithm

The learning algorithm for 1-CNN and parallel 1-CNN is basically analogous to the learning algorithm for CNN for images [Simard 03]. In this paper, we adopted a back propagation algorithm with mini-batches. For parallel 1-CNN, we split the error of the first full-connected layer to respective channel errors and then adopted back propagation for each channel independently.

7. Experimental Results

7·1 Data Acquisition

§1 Gyro sensor

In this paper, we use the motion sensor data of daily and sports activities used in [Altun 10a, Altun 10b]. It is well known that motion sensor data includes chaotic time series [Ali 07]. This dataset has time series observed by a tri-axial accelerometer, a tri-axial gyroscope and a tri-axial magnetometer placed on five different points on the subject's body. We classify 19 activities using this dataset. Each of the 19 activities is performed by eight subjects and 60 signals are obtained for each activity by each subject.

Barshan *et al.* [Barshan 14] tested a variety of classification algorithms using several different combinations of the dataset. The features they used are the minimum and maximum values, the mean value, variance, skewness, kurtosis, autocorrelation sequence and the peaks of the discrete Fourier transform (DFT) of s with the corresponding frequencies.

In their results, only the case in which only gyro sensors were used had lower accuracy than the other combination data. We therefore tested our algorithm using only the gyro sensor data.

In [Barshan 14], they used the leave one subject out cross-validation techniques. The 7980 (= 60 vectors 19 activities 7 subjects) feature vectors of seven of the subjects were used for training and the 1140 feature vectors of the remaining subject were used in turn for validation. This was repeated eight times, removing a different subject from testing in each repetition. The eight correct classification rates were averaged to produce a single estimate. They pointed out that the leave one subject out scheme enables objective activity recognition as it treats the subjects as units, without having data samples from trials of the same subject contained within both training and testing portions. Therefore, they convincingly argue that the leave

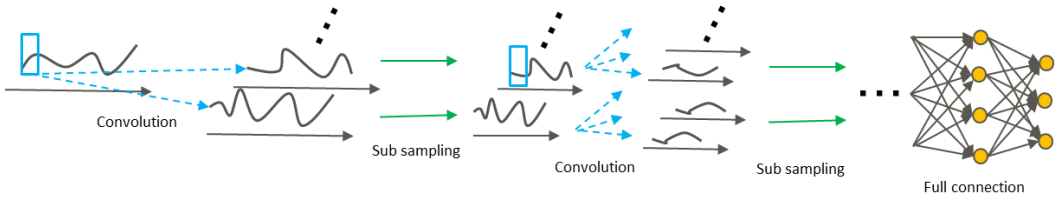


Fig. 7 Architecture of a one-dimensional CNN.

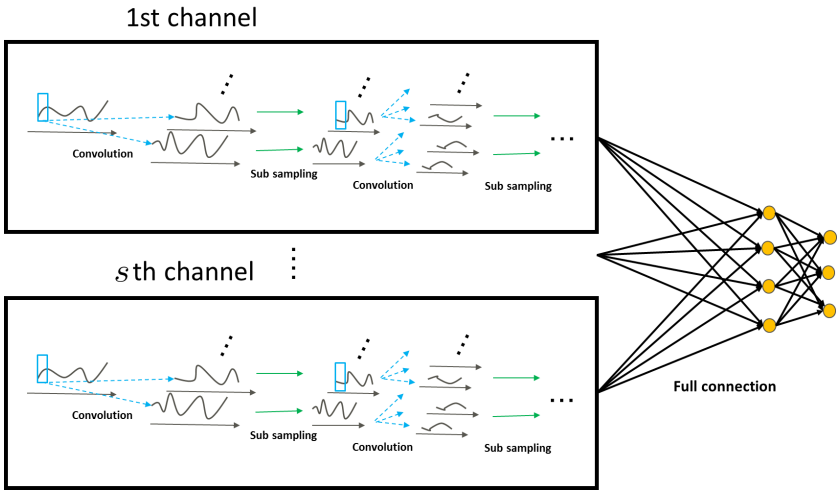
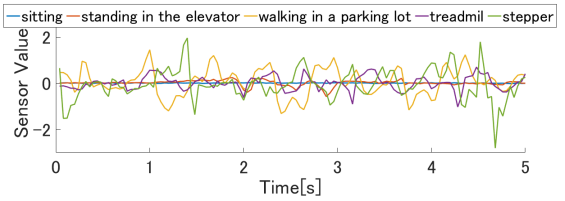
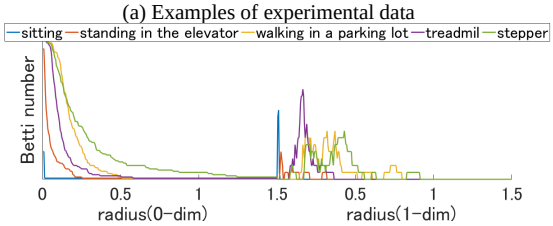


Fig. 8 Architecture of parallel one-dimensional convolutional neural network.



(a) Examples of experimental data



(b) Examples of Betti sequence of experimental data

Fig. 9 Experimental data

one subject out scheme results are the most meaningful hence, we selected to use this technique for the comparison.

Figure 9 shows an example of the experimental data and the Betti sequence of corresponding data. These data are the time series measured by a right arm x-axis gyro sensor when sitting, standing in the elevator, walking in a parking lot, running on the treadmill and exercising on a stepper.

§2 EEG dataset

We illustrated the generality of our algorithm by running additional experiments on different datasets. We use an existing electroencephalogram (EEG) dataset [Andrzejak], which contains two types of EEG datasets. We select datasets of the first type, and recorded while the volunteers were relaxed in an awake state with eyes open and eyes closed using a band-pass filterwith a 40Hz setting. We extract 600 signals of 2.5 seconds respectively, and run the binary classification problem using single-channel time-series data.

§3 EMG dataset

In addition to the EEG dataset, we conduct additional experiments on an electromyography(EMG) dataset. This EMG Physical Action Data Set [Lichman 13] includes EMG datasets of human activity collected from four subjects. We use the normal physical actions dataset, including 10 activities (Bowing, Clapping, Handshaking, Hugging, Jumping, Running, Seating, Standing, Walking and Waving) measured on eight muscles (8-channels). We extract 18 signals from each subject and each activity that contains 500 samples.

7·2 Comparing the Classification Algorithms

We compare the following classification algorithms: SVM using the statistical features which is the best method in [Barshan 14], SVM using chaotic features, imaging CNN and SVM, connected input 1-CNN and parallel 1-CNN using the Betti sequence. We use the Statistics and Machine

Learning Toolbox from MATLAB for the SVM algorithm, with the chaotic feature and Betti sequence. The 1-CNN algorithm and parallel 1-CNN algorithm are newly implemented in the MATLAB environment. The persistent homology is calculated by using an open source environment GUDHI [Clement 14].

In this work, we assign constant values to the hyper parameters for persistent homology and 1-CNN (Table 1). For parallel 1-CNN, we use the same hyper parameters for all channels.

7·3 Results and Discussion

Table 2 shows the cross-validation result of each algorithm and each problem.

These values represent the leave one subject out accuracy rate and the standard deviations of each algorithm in terms of the gyro sensor dataset and EMG dataset and the 10-fold accuracy rate in terms of the EEG dataset.

The Betti sequence was problematic in terms of gap of the radius parameter direction, thereby indicating that the SVM algorithm is slightly inappropriate for Betti sequence classification. However, combining the Betti sequence and 1-CNN algorithm achieved great performance. The leave one subject out cross-validation score was improved to $8.5 \sim 26.8\%$ compared to the SVM with Betti sequence. This improvement is attributable the ability of the Betti sequence to resist the gap of the radius parameter direction and reserving the location information. These results demonstrate that the 1-CNN architecture is suitable for Betti sequence classification. Moreover, this algorithm improves the cross-validation score by 12.2∼59.4% compared to the SVM algorithm with statistical features. Therefore, we can consider the proposed method combining Betti sequence and 1-CNN architecture to be a very effective algorithm for time series classification. The parallel 1-CNN algorithm improves the L1O score by 2.0∼6.3% compared to the connected input 1-CNN algorithm and by 18.5∼61.4% compared to the SVM algorithm with the statistic features. These results show that the suitable convolution kernels for each channel are different and that the parallel 1-CNN algorithm overcome the problem most appropriately. These results confirm that the proposed algorithm performs more accurately than the conventional algorithms.

8. Conclusion and Remarks

In this paper, we proposed a new algorithm for volatile time series classification. The proposed algorithm has new features, which we create for the time series, named the Betti sequence. The Betti sequence is converted from the transition rule of time series through the use of quasi-attractors, a key technique in dynamic systems theory. We also applied persistent homology, one of the main techniques of topological data analysis, to the quasi-attractor for extracting information on the arrangement of the quasi-attractor point cloud. In addition, we selected a suitable learning architecture for Betti sequence classification. This architecture mainly utilizes convolution and a subsampling operation to capture the relationship between the radius of a closed ball and the transition of the Betti sequence. We also modified the classification of multi-channel time series by selecting a parallel architecture to extract the features of the multichannel Betti sequence independently.

We validated the proposed algorithm by applying it to a human activity recognition task. The results showed that our proposed algorithm improved the recognition performance by 18.6% over that of conventional algorithms. This indicates that our proposed algorithm is highly effective for time series classification based on rules, for example, difference equations.

In our future work, we would need to shorten the computational time of the algorithm and validate its usefulness for various types of time series datasets.

♢ References ♢

[Adams 14] Adams, H., Tausz, A., and Vejdemo-Johansson, M.: javaPlex: A research software package for Persistent (Co)Homology, in *Mathematical Software – ICMS 2014*, pp. 129–136, Springer (2014)

[Aggarwal 14] Aggarwal, C. C.: *Data Classification: Algorithms and Application*, Champman and Hall/CRC, Boca Raton, FL, USA (2014)

[Ali 07] Ali, S., Basharat, A., and Shah, M.: Chaotic invariants for human action recognition, in *Proceedings of IEEE 11th International Conference on Computer Vision 2007 (ICCV 2007)*, pp. 1–8 (2007)

[Altun 10a] Altun, K. and Barshan, B.: Human activity recognition using inertial/magnetic sensor units, in *Human Behavior Understanding*, pp. 38–51, Springer (2010)

[Altun 10b] Altun, K., Barshan, B., and Tunçel, O.: Comparative study on classifying human activities with miniature inertial and magnetic sensors, *Pattern Recognition*, Vol. 43, No. 10, pp. 3605–3620 (2010)

[Andrzejak] Andrzejak, R. G., Lehnertz, K., Mormann, F., Rieke, C., David, P., and Elger, C. E.: Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state, *Physical Review E*, Vol. 64, 061907,

[Bankó 12] Bankó, Z. and Abonyi, J.: Correlation based dynamic time warping of multivariate time series, *Expert Systems with Applications*, Vol. 39, pp. 12814–12823 (2012)

[Barshan 14] Barshan, B. and Yuksek, M. C.: Recognizing daily and sports activities in two open source machine learning environments using body-worn sensor units, *The Computer Journal*, Vol. 57, No. 11, pp. 1649–1667 (2014)

[Basharat 09] Basharat, A. and Shah, M.: Time series prediction by chaotic modeling of nonlinear dynamical systems, in *Proceedings of IEEE 12th International Conference on Computer Vision 2009 (ICCV 2009)*, pp. 1941–1948 (2009)

Table 1 Hyper parameters for persistent homology and 1-CNN.

HyperParameter \Dataset	Gyro sensor		EEG dataset		EGM dataset	
Persistent homology						
embedding dimension	3		3		3	
maximum radius value	1.2		20		100	
Betti sequence length	300		300		300	
sampling radius width	0.004		0.067		0.33	
hole dimension	0 and 1		0 and 1		0 and 1	
1-CNN parameter	Connected	Parallel	Connected	Parallel	Connected	Parallel
convolution layer number	2	2	2	-	2	2
sub-sampling layer number	2	2	2	-	2	2
full connect hidden layer number	1	1	1	-	1	1
hidden layer unit size	19	19	2	-	10	10
1st convolution kernel size	9	5	6	-	6	7
1st convolution kernel number	8	4	7	-	7	7
1st sub-sampling unit size	8	4	7	-	5	3
2nd convolution kernel size	5	6	2	-	4	4
2nd convolution kernel number	8	4	7	-	10	7
2nd sub-sampling unit size	10	3	3	-	4	5

Table 2 Cross-validation results of each classification algorithm.

Datasets	Gyro sensor	EEG dataset	EMG dataset
	Accuracy		
method\validation	Leave one subject out [%]	10-fold[%]	Leave one subject out[%]
SVM + statistical feature	67.6 ± 4.7	44.4 ± 19.8	15.0 ± 10.0
SVM+Chaos feature	53.3 ± 7.1	55.2 ± 9.6	41.5 ± 25.9
DTW + 1-NN	6.4 ± 5.1	72.4 ± 6.1	15.0 ± 10.0
imaging CNN	18.9 ± 5.2	48.9 ± 4.2	10.0 ± 0
SVM+Betti sequence	63.5 ± 11.3	66.7 ± 5.6	49.6 ± 18.2
connected input 1-CNN+Betti sequence	79.8 ± 5.0	75.38 ± 5.7	74.4 ± 10.6
parallel 1-CNN+Betti sequence	86.1 ± 7.2	-	76.4 ± 7.2

[Bauer 14] Bauer, U., Kerber, M., Reininghaus, J., and Wagner, H.: PHAT–Persistent Homology Algorithm Toolbox, in *Proceedings of Mathematical Software, ICMS 2014 -4th International Congress*, pp. 137–143 (2014)

[Baydogan 13] Baydogan, M. G., Runger, G., and Tuv, E.: A bag-of-features framework to classify time series, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 35, No. 11, pp. 2796–2802 (2013)

[Baydogan 15] Baydogan, M. G. and Runger, G.: Learning a symbolic representation for multivariate time series classification, *Data Mining and Knowledge Discovery*, Vol. 29, No. 2, pp. 400–422 (2015)

[Carlsson 09] Carlsson, G.: Topology and data, *Bulletin of the American Mathematical Society*, Vol. 46, pp. 255–308 (2009)

[Clement 14] Clement, M., Jean-Daniel, B., Marc, G., and Mariette, Y.: The Gudhi library: Simplical complexes and persistent homology, in *Proceedings of Mathematical Software, ICMS 2014 -4th International Congress*, pp. 167–174 (2014)

[Edelsbrunner 08] Edelsbrunner, H. and Harer, J.: Persistent homology - A survey, in *Surveys on Discrete and Computational Geometry: Twenty Years Later: AMS-IMS-SIAM Joint Summer Research Conference*, pp. 257–282, American Mathematical Societ (2008)

[Fraser 86] Fraser, A. M. and Swinney, H. L.: Independent coordinates for strange attractors from mutual information, *Physical Review A*, Vol. 33, No. 2, pp. 1134–1140 (1986)

[Fulcher 14] Fulcher, B. D. and Jones, N. S.: Highly comparative feature-based time-series classification, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 26, No. 12, pp. 3026–3037 (2014)

[Huerta 12] Huerta, R., Vembu, S., Muezzinoglu, M. K., and Vergara, A.: Dynamical SVM for time series classification, in *Pattern Recognition*, pp. 216–225, Springer, Berlin, Heidelberg (2012)

[Kantz 03] Kantz, H. and Schreiber, T.: *Nonlinear Time Series Analysis*, Cambridge University Press, Cambridge, Massachusetts (2003)

[Korn 03] Korn, H. and Faure, P.: Is there chaos in the brain? II. Experimental evidence and related models, *C. R. Biologies*, Vol. 326, pp. 787–840 (2003)

[Levy 94] Levy, D.: Chaos theory and strategy: Theory, application, and managerial implications, *Strategic Management Journal*, Vol. 15, pp. 167–178 (1994)

[Lichman 13] Lichman, M.: UCI Machine Learning Repository, University of California, Irvine, School of Information and Computer Sciences (2013), http://archive.ics.uci.edu/ml

[Lines 05] Lines, M.: *Nonlinear Dynamical Systems in Economics*, Springer-Verlag Wien, Wien, Austria (2005)

[Otter] Otter, N., Porter, M. A., Tillmann, U., Grindrod, P., and Harrington, H. A.: A roadmap for the computation of persistent homology, arXiv:1506.08903[math.AT]

[Perea 14] Perea, J. A. and Harer, J.: Sliding windows and persistence: An application of topological methods to signal analysis, *Foundations of Computational Mathematics*, Vol. 15, No. 3, pp. 799–838 (2014)

[Perea 15] Perea, J. A., Deckard, A., Haase, S. B., and Harer, J.: SW1PerS: Sliding windows and 1-persistence scoring; discovering periodicity in gene expression time series data, *BMC Bioinformatics*, Vol. 16, No. 257 (2015)

[Rakthanmanon 12] Rakthanmanon, T., Campana, B., Mueen, A., Batista, G., Westover, B., Zhu, Q., Zakaria, J., and Keogh, E.: Searching and mining trillions of time series subsequences under dynamic time warping, in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2012)*, pp. 262–270 (2012)

[Simard 03] Simard, P. Y., Steinkraus, D., and Platt, J. C.: Best practices for convolutional neural networks applied to visual document

analysis, in *Proceedings of 7th International Conference on Document Analysis and Recognition 2003*, pp. 958–963 (2003)

[Sprott 03] Sprott, J. C.: *Chaos and Time-Series Analysis*, Oxford Univ. Press, Oxford, United Kingdom (2003)

[Wang 15] Wang, Z. and Oates, T.: Imaging time-series to improve classification and imputation, in *Proceedings of 24th International Joint Conference on Artificial Intelligence (IJCAI 2015)*, pp. 3939–3945, Buenos Aires, Argentina (2015)

[Wanga 13] Wanga, J., Liub, P., Shea, M. F., Nahavandia, S., and Kouzanid, A.: Bag-of-words representation for biomedical time series classification, *Biomedical Signal Processing and Control*, Vol. 8, No. 6, pp. 634–644 (2013)

[Yang 15] Yang, J. B., Nguyen, M. N., San, P. P., Li, X. L., and Krishnaswamy, S.: Deep convolutional neural networks on multichannel time series For Human Activity Recognition, in *Proceedings of 24th International Joint Conference on Artificial Intelligence (IJCAI 2015)*, pp. 3996–4001, Buenos Aires, Argentina (2015)

[Zheng 14] Zheng, Y., Liu, Q., Chen, E., Ge, Y., and Zhao, J. L.: Time series classification using multi-channels deep convolutional neural networks, *Web-Age Information Management (Proceedings of 15th International Conference Web-Age Information Management (WIAM 2014))*, pp. 298–310 (2014)

〔担当委員：高橋 恒一〕

Received July 25, 2016.

♢ Appendix ♢

A. Computational Time

Table A.1 Computational time of each classification algorithms.

method	preprocessing per 1 unit (15 time series)[s]	learning using 7 subjects dataset [s]	test per 1 unit [ms]
SVM +statistical feature	4.5	8	0.32
SVM +chaotic feature	7.3	140	0.25
imaging CNN	0.1	$>$ 600000	235
Proposed with javaPlex	67	56000	1.6
Proposed with PHAT	28.1	56000	1.6
Proposed with GUDHI	3.3	56000	1.6

Table A.1 shows the computational time required for the preprocessing part, learning part and validation test in an environment consisting a Xeon 3.6GHz CPU, 32GB RAM and runnning Windows 7 OS. The conventional algorithms are implemented in a MATLAB environment using the MATLAB toolboxes. The preprocessing part of our algorithm is implemented in MATLAB except for calculating the persistent homology, and the other parts are implemented in the MATLAB environment. The table lists the averaged computational time results for one unit of data (from 15 time series data observed at the same time) for both the preprocessing and test parts. The computational time results for learning by using the seven-subject dataset (7980 units) are listed in the learning part. Most of the computational cost of our preprocessing part arise from calculating the persistent homology. In terms of a suitable environment for calculating the persistent homology, some open sources are known, for example javaPlex [Adams 14], PHAT [Bauer 14] and GUDHI [Clement 14].

The calculation cost using various environments for calculating the persistent homology has previously been researched [Otter], and GUDHI is found to be faster than other environments for many datasets. In fact, GUDHI is also the fastest in these environments when calculating our datasets. When using the GUDHI environment, the computational efficiency of preprocessing is determined to be about 15 times higher than with javaPlex and five times more efficient than PHAT.

On the other hand, most of the computational costs to compute the statistical and chaotic feature are attributable to calculating the FFT and Lyapunov exponents, respectively. Table A.1 shows that, in our work, preprocessing using GUDHI require less time than preprocessing the statistical and chaotic features.

Conversely, for the learning part, our algorithm require much more time than the conventional algorithms. This is a general problem for deep learning architecture. However, this result was obtained by using a naive implementation in thes MATLAB environment and using only CPUs, so there is some possibility of shortening the computational time for learning by optimizing implementation and using GPUs.

Author's Profile

Yuhei Umeda (Member)

He received the M.S. and Ph.D. degrees in Mathematics from Kyushu University, Japan. Presently, he is a researcher at FUJITSU LABORATORIES LTD. Dr. Umeda is a member of IEEE and the Society of Instrument and Control Engineers.