Pattern Recognition 64 (2017) 268–276

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

# Efficient temporal pattern recognition by means of dissimilarity space embedding with discriminative prototypes

Brian Kenji Iwana[a,b,*], Volkmar Frinken[a,c,d], Kaspar Riesen[e], Seiichi Uchida[a]

[a] Department of Advanced Information Technology, Kyushu University, Fukuoka, Japan
[b] Institute of Decision Science for a Sustainable Society, Kyushu University, Fukuoka, Japan
[c] Electrical and Computer Engineering, UC Davis, Davis, CA, USA
[d] Onai Technology, San Jose, CA, USA
[e] University of Applied Sciences and Arts Northwestern Switzerland, Olten, Switzerland

ARTICLE INFO

*Keywords:*
Temporal patterns
Online digit classification
Dissimilarity representation
Ensemble classification
Dissimilarity space embedding

ABSTRACT

Dissimilarity space embedding (DSE) presents a method of representing data as vectors of dissimilarities. This representation is interesting for its ability to use a dissimilarity measure to embed various patterns (e.g. graph patterns with different topology and temporal patterns with different lengths) into a vector space. The method proposed in this paper uses a dynamic time warping (DTW) based DSE for the purpose of the classification of massive sets of temporal patterns. However, using large data sets introduces the problem of requiring a high computational cost. To address this, we consider a prototype selection approach. A vector space created by DSE offers us the ability to treat its independent dimensions as features allowing for the use of feature selection. The proposed method exploits this and reduces the number of prototypes required for accurate classification. To validate the proposed method we use two-class classification on a data set of handwritten on-line numerical digits. We show that by using DSE with ensemble classification, high accuracy classification is possible with very few prototypes.

## 1. Introduction

An important issue in pattern recognition is the method in which the data is represented. Using a representation that best describes the data or that captures the discriminating features is one of the most important factors in a successful machine learning model [1]. For that reason, much research is dedicated to the preprocessing, feature design, and transformation of data [2,3]. In general, there are two major approaches to this problem, statistical and structural. A statistical approach uses feature vectors as basic representation formalism, while the latter employs symbolic data structures (e.g. strings, trees, or graphs). Whether we use feature vectors or symbolic data structures for pattern representation, the goal is to find a good representation of the data with the foundation to support machine learning.

Dissimilarity representation is a novel approach in pattern recognition [4] in which the data is represented as the difference, or *dissimilarity*, to other objects or patterns. Instead of using a pattern's features, data representation is done based on the patterns relations to other patterns. In this manner, we represent the data as vectors of dissimilarities rather than feature vectors and are able to embed the dissimilarity representation into vector space, or dissimilarity space embedding (DSE). The intuition of this approach is the notion that distance between patterns of similar categories is smaller than patterns of different categories. This fundamental view of patterns as distances also grants the to use complex patterns, like symbolic data structures, with the mathematical and machine learning foundation of vector space [5]. By embedding the patterns into dissimilarity space, we can bridge the gap between complex pattern recognition and feature vector recognition.

This paper specifically tackles temporal pattern recognition. Temporal patterns present an aforementioned complex pattern due to the interconnectivity and dependency on sequence order. Also, when learning temporal patterns, temporal distortions and length variations must be considered. These obstacles are not addressed by classical fixed-dimensional Euclidean pattern recognition. Rather than using a complex model to represent temporal patterns, we can accuracy represent them using DSE and still use the tools granted by vector space. To accomplish this, we embed the temporal patterns into a dissimilarity space by using dynamic time warping (DTW) as a dissimilarity measure. DSE is an $N$-dimensional feature vector who elements are the DTW distance between prototype patterns and sample patterns. The $N$-dimensional space is now dimensionally fixed to the

* Corresponding author at: Department of Advanced Information Technology, Kyushu University, Fukuoka, Japan.
*E-mail addresses:* brian@human.ait.kyushu-u.ac.jp (B.K. Iwana), kaspar.riesen@fhnw.ch (K. Riesen), uchida@ait.kyushu-u.ac.jp (S. Uchida).

http://dx.doi.org/10.1016/j.patcog.2016.11.013
Received 9 January 2016; Received in revised form 17 September 2016; Accepted 16 November 2016
Available online 21 November 2016
0031-3203/

number of prototypes $N$ and can be used for the analysis of temporal patterns.

Recently, it has been a trend in pattern recognition that by increasing the data set size, a higher accuracy could be achieved. Banko and Brill [6] demonstrated that for natural language processing, the number of words was more important than the complexity of the classifier. The work of 80 million tiny images [7] has shown that even with a simple nearest neighbor classification performs sufficiently with a massive number of image patterns. Similar trials with massive patterns has been made for image segmentation [8], video image recognition [9,10], and handwritten character images [11], proving the value of using massive pattern numbers. Therefore, intuitively, to exploit DSE to its potential, we need to use large data.

However, by increasing the size of the data set, you also increase the computational requirement. When using massive pattern numbers, you can encounter the problem of having a huge computational requirement. Typically, there are three primary ways to address this issue: reducing the size of the patterns or features, using efficient search schemes, and selecting meaningful patterns. Tiny images [7] and tiny videos [9] overcome the computational requirement by reducing the image resolution. Other solutions include creating low-dimensional representations via feature selection [12,13]. Another solution to the computational problem is to use optimized search schemes such as estimation, approximate nearest neighbor [14], and vocabulary tree search [15]. A third solution is to select only the meaningful patterns or prototypes useful for the classifier. Classical methods include prototype selection, edition, and condensation [16].

Besides the problem of computational time when using a massive data set and a large set of prototypes, we are also creating a high dimensional vector space that is subject to the *curse of dimensionality* [17]. Therefore, we need to reduce the dimensionality of the DSE. AdaBoost offers two advantages for this purpose: (1) building a strong ensemble classifier from weak learners and (2) being used for selecting discriminative features when each weak learner depends on single features [18]. To solve the computational challenge, we apply AdaBoost to select the meaningful features and in this the case of DSE, prototypes as references to dissimilarities to samples. The end result is a novel, optimized prototype selector scheme for DSE.

The contribution of this paper is to address the massive pattern recognition of temporal patterns. Firstly, we embed temporal patterns into a vector space using a DTW-based dissimilarity measure. This allows us to represent temporal patterns as vectors of dissimilarities to prototypes. Secondly, we develop a new method of prototype selection by using AdaBoost as a feature selector for the resulting DSE. We can increase the computational efficiency by only using the prototypes within the ensemble classifier created by AdaBoost. Finally, we show that the accuracy of the classifier constructed by AdaBoost can be improved by preparing the prototype pool with patterns of classes external to a two-class classification.

The remaining of this paper is organized as follows. Section 2 reviews the related work in prototype selection and DSE. Section 3 elaborates on dissimilarity representation and the relation to vector space embedding. Section 4 provides a more detailed description of the method proposed. In Section 5, we confirm that the proposed method and discuss the usability of patterns of classes external to the two-class classifier. Finally, Section 6 draws a conclusion and describes future work.

## 2. Related works

In this section, we will briefly discuss prototype selection and provide the previous works related to DSE.

### 2.1. Prototype selection

The goal of prototype selection is to reduce an initial set of prototypes to a subset while retaining as much information as possible. There are many aspects and types of prototype selection methods, but in general, the mechanism for selecting prototypes can be broken down into three main categories [16]. The *condensation* methods find the points close to the decision boundaries and remove the internal points. The purpose of this method is to save the classifier's accuracy, but remove the redundancy. *Edition* methods attempt to reduce the noise by removing prototypes near decision boundaries which conflict with its surroundings. Finally, *hybrid* methods combine the two previous methods.

### 2.2. Dissimilarity representation

Dissimilarity representation is a fast growing field [19]. There have been many recent publications which concern different aspects of dissimilarity based pattern recognition [20–23] as well as extrapolating dissimilarity representation to graphs [24], string based object representation [25], and signature verification [26,27].

There have also been attempts to increase the efficiency of classification and the applications to different patterns. Pekalska et al. [28] proposed using normal density-based classifiers in DSE to overcome the limitations of $k$-NN methods. Pekalska et al. [29] also analyzed prototype selection techniques, such as mode seeking, $k$-Centers, an editing and condensing algorithm, linear programming, and feature selection, within DSE. Riesen and Bunke [30] also used prototype clustering in addition to prototype selection techniques on dissimilarity based graph classification. Other trials for prototype selection in DSE include using genetic algorithms [31] and using it for the application of signature verification [32].

## 3. Dissimilarity space embedding

Instead of characterizing patterns by its features, it is possible to represent patterns as dissimilarities to other patterns [4]. This dissimilarity representation measures the distance between patterns with the consideration that patterns of the same class are more likely to be similar than patters of a different class. Due to variations of patterns, there exist instances where a category of patterns does not share a common feature, but it is still possible to classify the patterns based on the distance to a prototype [33]. By using the distance measure between patterns as features, we can emphasize the fundamental relationship between patterns and de-emphasize the features of the patterns. Dissimilarity representation allows for the embedding of information on how the patterns exist on the system as a whole.

Using this dissimilarity representation, we can take a vector space embedding approach by representing the data as vectors of dissimilarities. An $N$-dimensional vector space is created from the dissimilarities between each prototype $\mathbf{p}$ in $\mathbf{P}$ and each sample $\mathbf{s}$ in $\mathbf{S}$ based on a distance measure $d(\mathbf{p}, \mathbf{s})$. As illustrated in Fig. 1, the key idea is to select $N$ number prototypes $\mathbf{P}$ and represent any sample as a vector $\mathbf{v}_s$ whose entries are the distances

$$\mathbf{v}_s = (d(\mathbf{p}_1, \mathbf{s}), d(\mathbf{p}_2, \mathbf{s}), \ldots, d(\mathbf{p}_N, \mathbf{s}))^T. \quad (1)$$

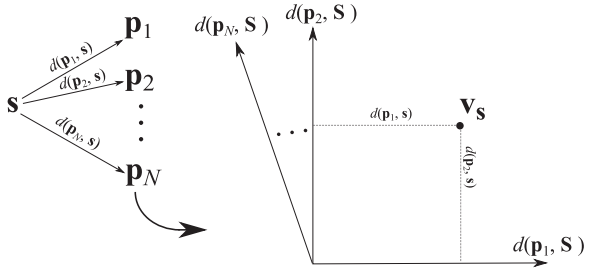


**Fig. 1.** A visualization of embedding a sample $\mathbf{s}$ into an $N$-dimensional vector space using the dissimilarities to prototypes $\mathbf{p}_1, \ldots, \mathbf{p}_N$. $\mathbf{v}_s$ is the vector whose entries are the distances to the respective prototypes.

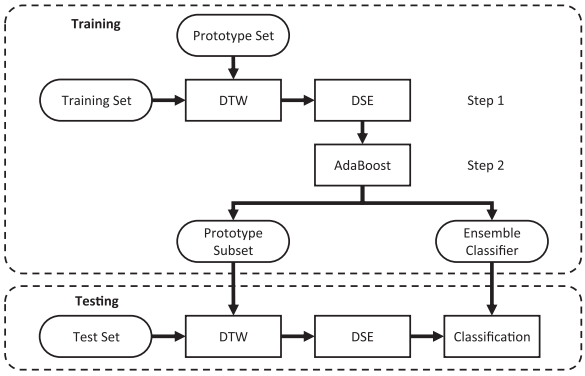


**Fig. 2.** Diagram outlining the proposed method. The training consists of creating a DSE with a DTW-based distance measure and learning with AdaBoost. The resulting ensemble classifier and prototype subset are carried to the testing phase.

The resulting DSE preserves some of the characteristics of the underlying patterns.

DSE was originally proposed for dissimilarities directly computed between objects [4], but the distance measure is not limited Euclidean feature vectors and can be used on patterns not endowed with mathematically well defined operations such as graphs [30] and sequences [34]. The embedding into this vector space, allows us to use the full potential of existing algorithms that work on vectors but on additional data types.

## 4. Prototype selection with AdaBoost in DSE

The proposed method for prototype selection of temporal patterns, as illustrated in Fig. 2, consists of two primary steps. First, we create an $|\mathbf{P}|$-dimensional DSE with a DTW-based distance measure. Next, we apply AdaBoost to discriminatingly select the relevant prototypes. In doing so, we select a subspace of the original DSE to those prototypes used in the final ensemble classifier created by AdaBoost by preserving the dimensions associated with the selected prototypes.

### 4.1. DTW-based distance measure

Embedding into vector space requires a distance measure that returns a value that represents the dissimilarity between patterns. To accurately portray the dissimilarities, the chosen distance measure should be an effective measurement for the patterns.

DTW is a popular dynamic programming algorithm of finding the optimal alignment and measuring the distance between temporal patterns. Given two sequences $\mathbf{a} = a_1, \ldots, a_i, \ldots, a_I$ of length $I$, and $\mathbf{b} = b_1, \ldots, b_j, \ldots, b_J$ of length $J$, where $a_i$ and $b_j$ are elements with a well-defined distance measure, such as coordinates or vectors, DTW computes the optimal matching of sequence elements given constraints. The goal is to determine the optimal match between pairwise elements so that the global distance is minimized. This can be seen in Fig. 3, which illustrates a sample DTW-distance calculation and the optimal matches as determined by the warping path from DTW.

There are several different constraints to which sequence elements must conform in the literature. Our method uses the asymmetric slope constraint given by

$$D(i, j) = c(a_i, b_j) + \min_{j' \in \{j, j-1, j-2\}} D(i-1, j') \tag{2}$$

over the cost matrix $c(a_i, b_j)$ where $c$ is the Euclidean distance between the points at $i$ and $j$ of sequences $\mathbf{a}$ and $\mathbf{b}$ respectively. The DTW algorithm estimates the optimal distance on the cost matrix between the $D(0, 0)$ and $D(I, J)$, therefore, the DTW-distance is determined by the value at $D(I, J)$. In other words, where $Q$ is the set of all matched elements of $\mathbf{b}$ with respective elements in $\mathbf{a}$, the DTW-distance is defined as

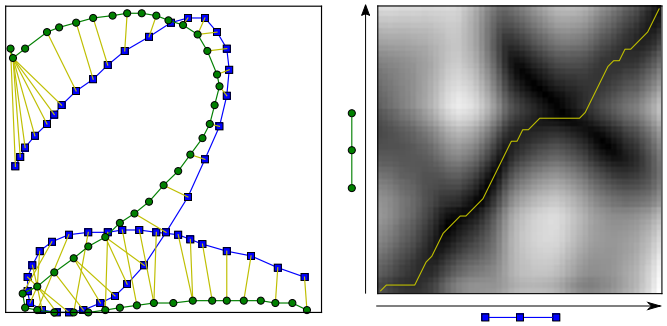

**Fig. 3.** An example of a DTW calculation. (left) Two superimposed sample patterns. The yellow connections are the optimized matches between the elements in $\mathbf{a}$, the blue square markers, and the elements in $\mathbf{b}$, the green circle markers. The length of the connections are the distances between matched elements. (right) An rendering of the cost matrix between the patterns over time. The values are determined by the local distance between every element where dark areas as low cost and light areas as high cost. The central yellow line represents the warping path determined by DTW. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article).

$$d_{\mathrm{DTW}}(\mathbf{a}, \mathbf{b}) = D(I, J) = \sum_{b_q \in Q} c(a_i, b_q). \tag{3}$$

The asymmetric slope constraint of Eq. (2) aligns matching features so that the path of the distance measure is monotonic and that the number of matches is always equal to $I$, the length of each sequence $\mathbf{a}$. The fixed length of $\mathbf{a}$ and varying length of $\mathbf{b}$ allows for the DTW to be calculated without consideration of the normalization of pattern sizes and variable rates, while the monotonicity maintains elastic matching continuity. These characteristics allow for a desirable dissimilarity measure for non-uniform temporal patterns.

We embed temporal patterns into a vector space, in the manner described by Eq. (1), by using the DTW-distance $d_{\mathrm{DTW}}$ as the dissimilarity measure. The resulting dissimilarity space for each sample $\mathbf{s}_m$ is now represented as a feature vector,

$$\mathbf{v}_{\mathbf{s}_m} = (d_{\mathrm{DTW}}(\mathbf{p}_1, \mathbf{s}_m), d_{\mathrm{DTW}}(\mathbf{p}_2, \mathbf{s}_m), \ldots, d_{\mathrm{DTW}}(\mathbf{p}_N, \mathbf{s}_m))^T, \tag{4}$$

where each feature is the DTW-distance to each prototype $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_N$. The DTW-distance measure allows the embedding of patterns of varying sizes into a fixed $N$-dimensional vector space.

### 4.2. Prototype selection with AdaBoost

The AdaBoost algorithm, described in Algorithm 1, is a machine learning algorithm that combines a set of weak learners to create a nonlinear ensemble classifier [35]. The algorithm performs by iterating through many rounds of weighted weak learners and combines them into a single strong classifier after a desired low training error has been reached [35,36]. During each round of AdaBoost, the weak learners is applied with cumulative weights emphasizing the incorrect patterns and deemphasizing the correct classifications from the previous rounds. The weak learners typically consist of fast, inaccurate, or linear classifiers with the only requirement of being a *better than chance* classifier. The final ensemble classifier combines the weak learners using a weighted vote allowing for the adaptation to difficult and nonlinear data sets.

**Algorithm 1.** AdaBoost algorithm.

If we define the weak component learner to be linear classifiers based on a single feature, the decision boundaries are dependent on single dimensional axes. By representing the dissimilarities between prototypes and samples in vector space as features, each axis is determined by a prototype. Each prototype-axis has a potential weak learner to be selected by AdaBoost. This consideration allows AdaBoost

1: **function** AdaBoost
2: Training set $\{(\mathbf{s}_1, \mathbf{l}_1), \ldots, (\mathbf{s}_m, \mathbf{l}_m), \ldots, (\mathbf{s}_M, \mathbf{l}_M)\}$
3: Initialize weights: $\mathbf{W}_1(m) \leftarrow \frac{1}{M}$ **for** $m = 1, \ldots, M$
4: **for** $k = 1, \ldots, k_{max}$ **do**
5: Train weak learner using $\mathbf{W}_k(m)$.
6: Select classifier with the lowest error:
$h_k(\mathbf{s}) \leftarrow \underset{h_n}{\text{argmin}} \epsilon_n = |\{(s, 1) \in T | w_k(m)\{s \neq 1\}\}|$
7: Calculate confidence:
$\alpha_k \leftarrow \frac{1}{2}\ln\left(\frac{1-\epsilon_k}{\epsilon_k}\right)$
8: Update weights:
$\mathbf{W}_{k+1}(m) \leftarrow \mathbf{W}_k(m) \times \begin{cases} \mathbf{e}^{-\alpha_k}, & \text{if } h_k(\mathbf{s}_m) = l_m \\ \mathbf{e}^{\alpha_k}, & \text{otherwise} \end{cases}$
9: Normalize
$\mathbf{W}_{k+1}(m)$ such that $\sum_{m'} \mathbf{W}_{k+1}(m') = 1$
10: **end for**
11: **return** composite classifier:
$H(\mathbf{s}) \leftarrow \text{sign}(\sum_{k=1}^{k_{max}} \alpha_k h_k(\mathbf{s}))$
12: **end function**

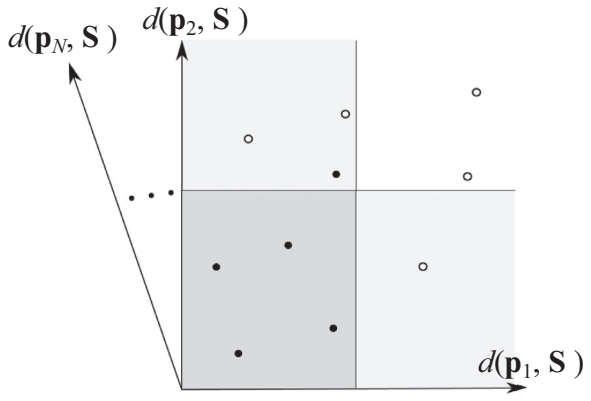


**Fig. 4.** A visualization of AdaBoost applied to a dissimilarity space. The points represent samples in the $N$-dimensional space of two classes, white and black. The mid-lines depict weak learners based on each of the prototype dissimilarity-based axis with the shaded areas representing a classification of black. Only classifiers for the dissimilarity-based axes of $\mathbf{p}_1$ and $\mathbf{p}_2$ are shown. The eventual end ensemble classifier would be a weighted vote based on all dimensions of the DSE that are chosen by AdaBoost.

to be used as a discriminative prototype selector due to its ability to choose a subset of classifiers to build its ensemble classifier. This is only possible if the weak learner is dependent on a single feature. If the weak learner uses all features, AdaBoost cannot be used in this manner. Unlike exhaustive classification methods, only prototypes useful to AdaBoost need to be calculated and irrelevant prototypes can be discarded. This ability to select features, or dissimilarities to prototypes, allows AdaBoost to reduce the dimensionality of DSE and the computational requirement of classifying massive data sets that would typically require large amounts of prototypes.

To compare each of the two classes in the ensemble classifier training, we use a nearest prototype classifier as the weak learner in Line 5 of Algorithm 1 within the DTW-based DSE created from $\mathbf{v}_{s_m}(\mathbf{p}_n)$. The weights of the classifiers, as determined by AdaBoost, are applied to the samples of each prototype-axis within the DSE. Within each round of AdaBoost, the classifier with the lowest error in Line 5 of Algorithm 1 is selected as the basis of the weight updates for the subsequent round. Fig. 4 illustrates the application of AdaBoost on a dissimilarity space. The resulting training produces a discriminatively selected subset of prototype-axis classifiers and confidence values used to calculate the final classifier.

AdaBoost provides the ability to discriminately select prototypes as well as offering us a powerful nonlinear classifier. With ability to reduce the dimensionality of independent dimensions, we can overcome the challenges of computationally costly classifiers, such as $k$-NN and Support Vector Machines (SVM), of related DSE works. Combined with being able to learn very complex decision boundaries, AdaBoost is an ideal classifier for DSE.

### 4.3. Preparation of the initial prototype set

The assumption of using dissimilarities in two-class classification is that patterns can be grouped into classes based on their similarity to the prototypes. Thus, for a two-class classification problem, it is natural to prepare the sets by using patterns from those two classes. Usually, this indicates that patterns more similar to the prototype are members of the prototype's class and patterns that are less similar belong to the compared class. We start by preparing the initial prototype set with patterns of class $A$ and class $B$. However, note that the selected prototypes might not be balanced; prototypes of one class may be selected over the other class. In the most extreme case, all of the selected prototypes would be of a single class.

Although, DSE only requires the dissimilarity representation to show a separation of the two classes for accurate classification. It does not necessarily need to know the class membership of the prototypes. Based on this idea, it is possible that in two-class classification, there exist prototypes of a third, external class that is more discriminative than the pairwise classes. Fig. 5 demonstrates an example where the prototype-axis of external class $C$ has a higher distinction in the dissimilarities of the samples than the pairwise classes that are being compared.

## 5. Experimental results

### 5.1. Data set

The UNIPEN on-line handwritten digit data set was used for the experiment. The data set is provided by the International Unipen Foundation (iUF). The iUF provides benchmark data sets for writer identification and handwriting recognition [37]. The data set used for the experiment consists of about 15,000 time ordered sets of coordinates split into ten digit classes. Real on-line handwritten numerical digits is an ideal source of data for temporal pattern classification because of the numerous variations of patterns while maintaining distinct categories.

For the experiment, we used a two-class classifier comparing to individual classes and repeated the experiment for all possible combinations of classes. While the two-class classifier employed by AdaBoost can be extended by using a 1-vs-other classification or a multi-class AdaBoost, we adhere to the use of the two-class classification to better analyze the experimental results. This also allows the experiment to determine the effects proposed by Section 4.3, namely, the inclusion of prototypes of classes not found in the two compared classes.

The data set was divided into three subsets, an initial prototype set, a training set, and a test set. Each prototype set was made of 130 patterns of each class. The size of each test set was defined at 260 patterns. Finally, the training sets for the dissimilarity space embedding and AdaBoost contained 2080 patterns. This process was repeated for both experiments using the same partitions of the data set for 9-fold cross-validation.[1]

[1] 9-fold cross-validation was used because the data set was split into 10 sets: 8 sets for training patterns, and two left out, 1 for test patterns and 1 for prototype patterns.

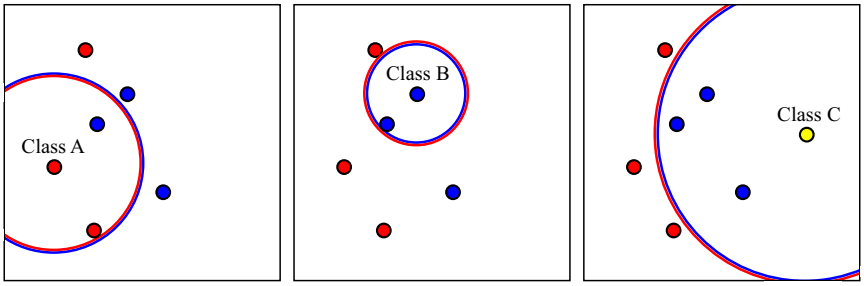


**Fig. 5.** This figure shows an idealized example of an external class, Class *C*, performing better than any pattern in within the pairwise classification of Class *A* and Class *B*. The rings represent decision boundaries for a classifier based on the distance to the respective centers.

### 5.2. Evaluation of the proposed method

In this section, we evaluate the viability of two-class classification in DSE with AdaBoost in accuracy and efficiency. We consider the following five evaluations:

- *Proposed Method*: The test sample set is classified using the strong ensemble classifier that is constructed by AdaBoost in DSE. This serves as the evaluation of the proposed method when using prototypes of classes within the pairwise classes.
- *Proposed Method+External Classes*: Similar to the Proposed Method, this evaluation uses the strong ensemble classifier constructed by AdaBoost in DSE. Unlike the Proposed Method, the initial prototype set of this method was prepared with prototypes of any of the ten possible classes. The extra patterns for the external classes were taken from the unused prototypes that would normally be left out for two-class classification. There is an equal representation of every class in the initial prototype set.
- *DSE Random*: DSE using randomly selected prototypes equal to the number of prototypes selected by Proposed Method and the same nearest prototype classifier used in the proposed methods.
- *DSE k-Centers*: *k*-Centers selects prototypes in a manner that the prototypes are evenly distributed, as described in [30]. The prototypes are selected through a *k*-medians clustering method. Like DSE Random, classification is done in a DSE created from the selected prototypes and with the same classifier as Proposed Method.
- *Simple 1-NN*: To serve as the baseline evaluation of classic non-DSE space, random prototypes were used with a 1-NN method based on the DTW-distance between prototypes and test samples. The number of prototypes selected was exactly equal to the number of prototypes selected by Proposed Method.

These evaluations are done considering the effectiveness of the proposed method to reduce the number of prototypes. With exception to Proposed Method+External Classes, the evaluations use a number of prototypes equal to the Proposed Method. The accuracies are calculated by averaging all folds and the computational time cost is determined on a system with a Intel Xeon E5 2.6 GHz processor.

In addition, we compare our method to other two-class classification methods without considering prototype reduction. This includes using 1-NN with the entire training set as prototypes (Full Training Set 1-NN), using a Support Vector Machine with a DTW based kernel (DAG-SVM-GDTW) [38], and representing each time-series as a sum of piecewise polynomial basis functions through a kernel interpolation (Interpolation Kernel) [39]. The computational time cost of Full Training Set 1-NN is done on the same system as the previous methods. The other comparisons were reported using the same handwritten digits from the UNIPEN data set using a reduced training set size for speed purposes.

#### 5.2.1. Analysis of proposed method

The proposed method of using AdaBoost in DSE has a large increase in accuracy compared to 1-NN with comparatively reduced prototypes. As shown in Table 1, averaging all two-class classifications across all folds of cross-validation, AdaBoost in DSE achieves a 95.15 ± 5.74% accuracy after 200 rounds while Simple 1-NN with randomly selected prototypes of the same amount has a lower accuracy of 86.94 ± 17.2%. Where 1-NN in feature space is highly dependent on the prototype set has a low tolerance to noise [40], the weak learners selected by AdaBoost in DSE tend to have high accuracies due to the weighted error minimization of the selection process. This high

**Table 1**
A table comparing the evaluated methods and methods from literature in recognition rate, data set size required at execution, and average per digit classification time.

| Evaluation | Recognition rate | Data set size | Per digit time |
|---|---|---|---|
| Proposed Method | 95.15 ± 5.74% | 13.25 ± 6.02 prototypes | 0.088 s |
| Proposed Method +External Classes | 96.67 ± 4.38% | 14.66 ± 6.97 prototypes | 0.094 s |
| Simple 1-NN | 86.94 ± 17.2% | Same as Proposed Method | 0.120 s |
| DSE Random | 83.19 ± 7.89% | Same as Proposed Method | 0.092 s |
| DSE *k*-Centers | 80.75 ± 7.54% | Same as Proposed Method | 0.090 s |
| Full Training Set 1-NN | 99.03 ± 0.92% | 90% training set | 14.33 s |
| DAG-SVM-GDTW [38] | 96.2% | 40% training set | – |
| Interpolation Kernel [39] | 97.1 | 50% training set | – |

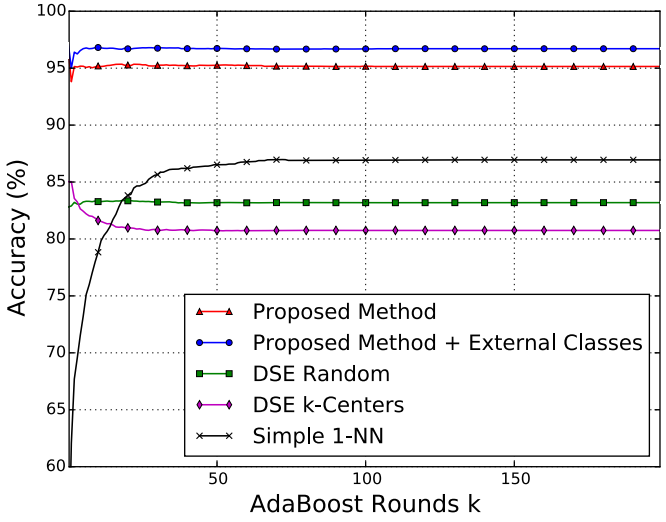


**Fig. 6.** The average recognition accuracy for 45 class pairs over 9 folds between the DSE with AdaBoost method (Proposed Method and Proposed Method+External Classes), the DSE baseline methods using prototypes selected randomly (DSE Random) and prototypes selected through *k*-medians clustering (DSE *k*-Centers), and 1-NN with randomly selected prototypes (Simple 1-NN).

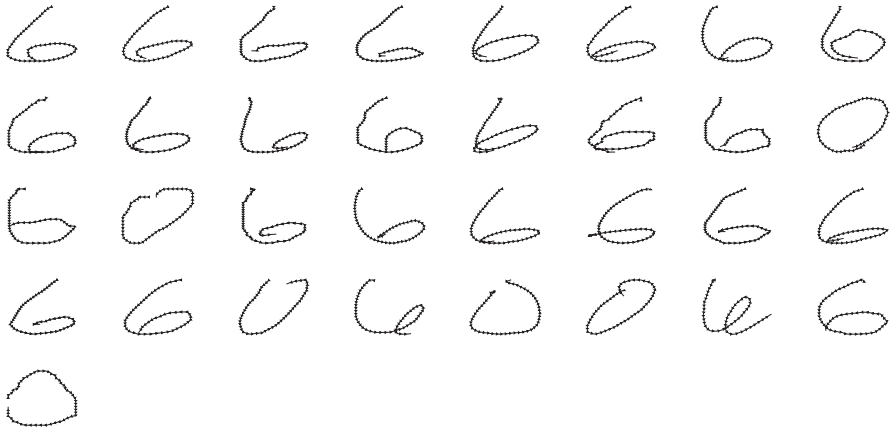

**Fig. 7.** The unique prototype patterns selected by AdaBoost between class "0" and class "6" when limited to the compared classes (Proposed Method), in order of selection (top-left to bottom-right).

accuracy of the weak learners provides a high overall accuracy of the ensemble classifier, even in the initial rounds.

When compared to DSE Random and DSE *k*-Centers, Proposed Method had a much higher accuracy using the same number of prototypes. DSE Random and DSE *k*-Centers both had the lower accuracy of 83.19 ± 7.89% and 80.75 ± 7.54% respectively. The accuracy improvement is due to the nonlinear ability of the ensemble classifier in addition to the weak learner error minimization. Fig. 6 also shows that the simple clustering of prototypes does not necessarily improve classification of dissimilarity space. The overfitting in Fig. 6 from the DSE *k*-Centers evaluation is due to prototypes from outlier clusters being selected and having a negative impact on the accuracy.

An example of an improvement of DSE with AdaBoost is the case of the comparison between class "0" and class "6." This example reveals which patterns the AdaBoost algorithm judged as the best prototypes for the classifier as well as the progression to the prototypes it thought were more difficult but still beneficial to the ensemble classifier, as shown in Fig. 7. The selected patterns are visually and intuitively closely representative to their ground truth classes and distant from their opposing classes. The improvement in the abilities to distinguish classes provided by AdaBoost in DSE can be seen in Figs. 8 (a) and (b). Compared to the randomly selected prototypes of Fig. 8 (a), the DSE created by the prototypes selected by AdaBoost have much less noise and a better distinction between classes.

In addition, Fig. 6 shows a quick convergence in accuracy from the Proposed Method when compared to the baselines. Freund and Schapire [41] proved that the error of the final classifier drops exponentially and is bounded by $\mathbf{e}^{-2r(1/2-\epsilon_k)^2}$, where $k$ is the round number and $\epsilon_k$ is the error of the $k$th learner. AdaBoost has been also shown to be resilient to overfitting [42–44] which is consistent with the observations from the experiment.

A convergence was not only observed in the accuracy but the prototypes selected by AdaBoost, as shown in Fig. 9. We found that

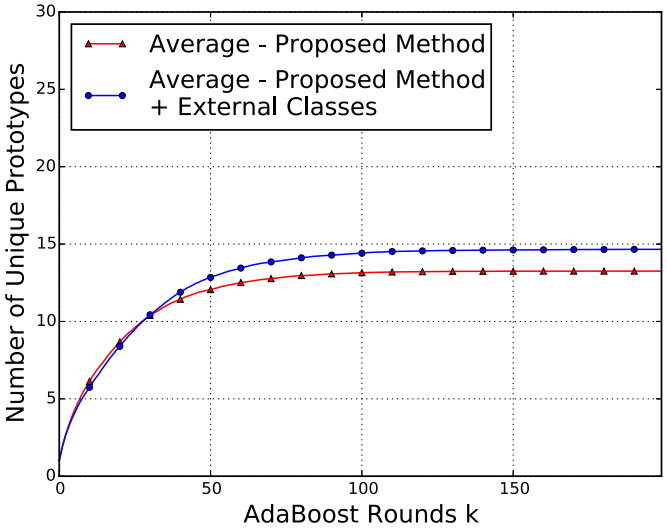


**Fig. 9.** A comparison of the reduction in prototypes between limiting the selection of prototypes to the pairwise classes (Proposed Method) and including external classes (Proposed Method+External Classes). The graph indicates the average number of unique prototypes selected by Adaboost over the number of rounds of weak learner decisions.

after the initial rounds, AdaBoost tends to repeat the use of previously selected weak learners. The initial weak learners are selected for their ability for separating the general patterns, but as the weights of the difficult patterns get emphasized, the latter iterations are selected for their ability with the outliers and noise. This caused the prototype selection process to become cyclic in that the algorithm will compensate the weights for the difficult patterns only to change it back for either other difficult patterns or the general patterns. The repeated use of past prototypes reduced the original prototype set of 260 patterns

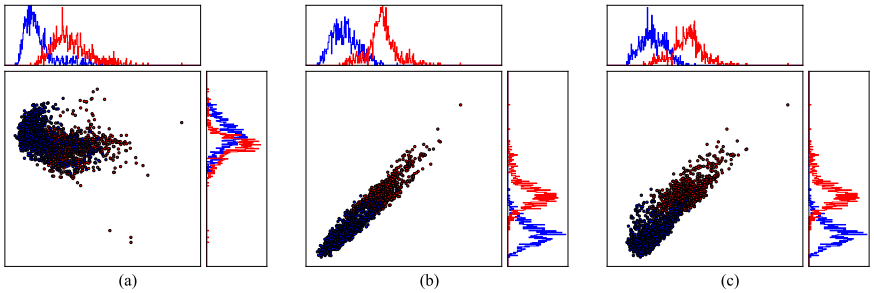


**Fig. 8.** An example of two prototypes in DSE comparing Class "0" (red) and Class "6" (blue). (a) shows the first two prototypes selected by DSE Random, (b) shows the first two prototypes selected by Proposed Method, and (c) shows the first two prototypes selected by Proposed Method+External Classes. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article).

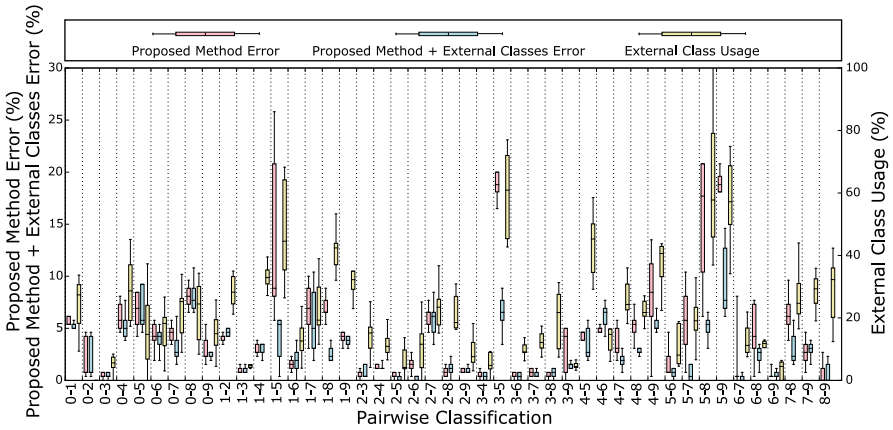


**Fig. 10.** A box plot of the pairwise classifications across all folds with the top and bottom of the boxes as the third and first quantile respectively. The Proposed Method and Proposed Method+External Classes boxes represent the error rates. The External class usage boxes are calculated by $N_E/N_T$ where $N_E$ is the number of unique prototypes that belong an external class and $N_T$ is the total number of unique prototypes used to build the strong classifier. Note, the External Class Usage boxes are scaled for the visualization. This shows relationship between the frequency at which external prototypes were selected and the accuracies of the proposed methods.

per classification to an average of 13.25 ± 6.02 prototypes after 200 rounds.

The 94.90% reduction in prototypes results in a much faster test classification time due to the dramatic reduction in required DTW calculations. Using an Intel Xeon E5 2.6 GHz processor, the Proposed Method had an average time cost of 0.088 s for every classification, which includes repeating the DTW calculations for each prototype in the ensemble classifier. In comparison, Full Training Set 1-NN had an average time cost of 14.33 s for every classification due to requiring a DTW calculation with each training prototype. While the accuracy of Full Training Set 1-NN is the highest of all compared methods, a time of 14.33 s is unsuitable for real-time applications and larger data sets. However, the results of the proposed method confirm that a high level of accuracy can be maintained even after reducing the number of prototypes significantly. It is important to note that while the accuracy stabilizes quickly, 200 rounds were calculated to give the prototype selection process to stabilize.

### 5.2.2. Analysis of proposed method+external classes

To evaluate the effectiveness of including prototypes of classes external to the two-class classification, the proposed method is expanded to allow the selection of prototypes of any class. The addition of prototypes of external classes to the pairwise classification increased the accuracy of the ensemble classifier constructed by AdaBoost. The results in Table 1 show an improvement with the previous results of an average of 95.15 ± 5.74% after 200 rounds in Section 5.2 increased to 96.67 ± 4.38%. As portrayed in Fig. 9, the total number of unique prototypes used for the ensemble classifier slightly increased from 13.25 ± 6.02 to 14.66 ± 6.97 and takes an average of 0.094 s for every classification with an Intel Xeon E5 2.6 GHz processor. Overall, there was a significant increase in accuracy and only a nominal increase in number of prototypes and computational time.

The distribution of percentages of external class prototypes selected by AdaBoost for each pairwise classification, shown in Fig. 10, reveals that using the additional external class prototypes is common, especially when the ensemble classifier failed to achieve a high accuracy before they were included. Generally, for the more difficult classifications of Proposed Method, AdaBoost selected more prototypes of the external classes during Proposed Method+External Classes. A large performance gain is seen on the initially difficult classifications when a large percentage of external class prototypes were introduced. Only rarely were the external class prototypes a detriment to the classification. It can be inferred that the class is not an important factor in selecting prototypes, but the shape of the pattern.

Two example cases of this include:

**Case 1: An increase in choices of similar shapes.** In general, a good classifier would expect the distance between samples in the same class to be smaller than the samples from a different class [36]. Intuitively, by increasing the pool of prototypes, AdaBoost will be able to find those prototypes which are more similar to one class than the comparison class. Because only the shape and not the class membership of the prototypes is taken into consideration, its possible for there to be prototypes of an external class to resemble one of the two classes in question. An example of this would be in the comparison of class “0” and class “6′ in Fig. 11, the digits which have the ground truth of ”4″ are visually similar to be number “6”s.' These prototypes were selected because their similarity to class “6” greatly exceeds their dissimilarity to class “0.”

**Case 2: A better separation in dissimilarity between the test classes despite being different to both.** In contrast to Example 1, there are instances where there are few successful classifiers within the two classes being compared. This means that there exist patterns that are dissimilar to both classes, but are selected because they are more dissimilar to one class than the other. The DSE of these external class prototypes can provide a higher distinction between the compared classes better than the compared classes can themselves. The comparison of class “5” and class “8” is a good example of this phenomenon. The unique selected prototypes are shown in Fig. 13 and a comparison of the dissimilarity representation of the first two prototypes selected by AdaBoost with and without external classes can be seen in Fig. 12. From Fig. 12 (a) and (b), it is observed that class “5” makes poor linear classifiers due to the large difference in writing styles . Likewise, class “8” has a high variation in styles causing for inconsistent classifier performance. However, from the viewpoint of class “0,” both classes in question are dissimilar, but class “5” is even more dissimilar than class “8.”

## 6. Conclusion

In this paper, we presented a method of temporal pattern recognition by transforming the data into a vector space by means of dissimilarity embedding. Using AdaBoost for the purpose of prototype selection in the DTW-distance vector space proved to be a viable method of pattern classification. The experiment performed on a portion of the UNIPEN on-line handwriting data set. It was able to maintain an average accuracy of 95.15 ± 5.74% for two-class classifiers over 9-fold validation. We were able to increase the accuracy to an average of 96.67 ± 4.38% by introducing prototypes of classes external to the two-class classifiers into the initial prototype set. The number of prototypes were reduced to an average of 13.25 ± 6.02 for Proposed

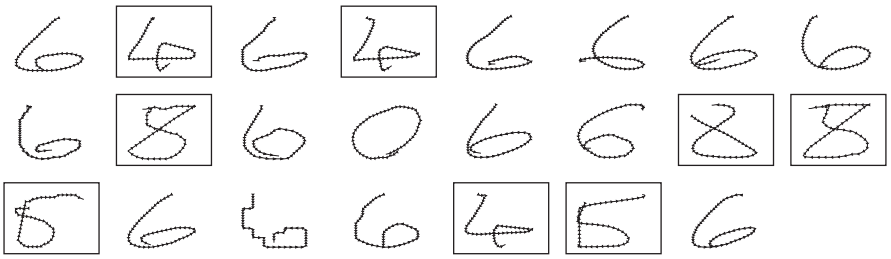

**Fig. 11.** The unique prototype patterns selected by AdaBoost between class “0” and class “6” when allowed to choose any class (Proposed Method+External Classes), in order of selection (top-left to bottom-right). The boxed patterns are instances of AdaBoost supplementing the selection with prototypes of external classes.

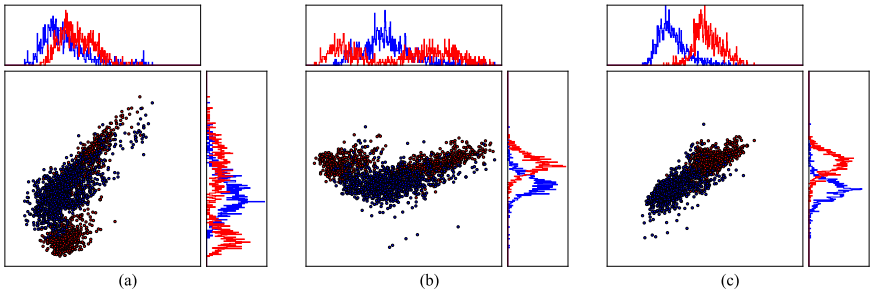


**Fig. 12.** An example of two prototypes in DSE comparing Class “5” (red) and Class “8” (blue). (a) shows the first two prototypes selected by DSE Random, (b) shows the first two prototypes selected by Proposed Method, and (c) shows the first two prototypes selected by Proposed Method+External Classes. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article).

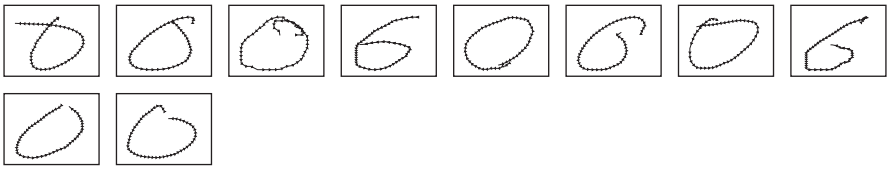

**Fig. 13.** The unique prototype patterns selected by AdaBoost between class “5” and class “8” when allowed to choose any class (Proposed Method+External Classes), in order of selection (top-left to bottom-right). The boxed patterns are instances of AdaBoost supplementing the selection with prototypes of external classes.

Method and 14.66 ± 6.97 for Proposed Method+External Classes. The experiment demonstrates:

- AdaBoost can be used as an effective prototype selection method in DSE. Temporal patterns with a DTW-based distance measure can be classified with a high accuracy and low prototype count by using the strong classifier constructed by AdaBoost. Compared to exhaustive methods such as 1-NN, it is possible to reduce the computational requirement by using an ensemble classifier comprised of fast weak learners with a very small number of prototypes.
- The overall performance of AdaBoost in DSE can be improved by introducing classes external to the two-class classifier while maintaining a substantial decrease in prototypes.
- There exist patterns that are more similar to one class than another class but do not necessarily belong to either. Some of these patterns from third classes can distinguish between the two classes better than patterns from the two classes themselves.

In the future, we hope to expand the ability of pattern recognition in DSE to other data sets and applications as well as explore the characteristics of temporal patterns in DSE. The large reduction in computational requirement opens DSE to be used with massive data sets and real time applications. Also, because the proposed method demonstrates the usefulness of shape over class membership, future work can be done by using DSE with generated, classless, and unlabeled prototypes. Optimized prototypes can lead to more efficient and accurate dissimilarity representation.

## Conflict of interest

None declared.

## References

[1] P. Domingos, A few useful things to know about machine learning, Commun. ACM 55 (10) (2012) 78–87. http://dx.doi.org/10.1145/2347736.2347755.
[2] H. Liu, H. Motoda, Feature Extraction, Construction and Selection: A Data Mining Perspective, Kluwer Academic Publishers, Norwell, MA, USA, 1998. http://dx.doi.org/10.1016/s0898-1221(99)90021-4.
[3] J. Han, M. Kamber, J. Pei, Data Mining: Concepts and Techniques. ⟨http://dx.doi.org/10.1145/565117.565130⟩.
[4] E. Pekalska, R.P. Duin, The Dissimilarity Representation for Pattern Recognition: Foundations and Applications, World Scientific, Singapore, 2005. http://dx.doi.org/10.1142/9789812703170.
[5] R.P. Duin, E. Pkalska, The dissimilarity representation for structural pattern recognition, in: Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications, Springer, 2011, pp. 1–24. ⟨http://dx.doi.org/10.1007/978-3-642-25085-9_1⟩.
[6] M. Banko, E. Brill, Scaling to very very large corpora for natural language disambiguation, in: Proceedings of the 39th Annual Meeting on Association for Computational Linguistics, Association for Computational Linguistics, 2001, pp. 26–33. ⟨http://dx.doi.org/10.3115/1073012.1073017⟩.
[7] A. Torralba, R. Fergus, W.T. Freeman, 80 million tiny images: a large data set for nonparametric object and scene recognition, IEEE Trans. Pattern Anal. Mach. Intell. 30 (11) (2008) 1958–1970. http://dx.doi.org/10.1109/tpami.2008.128.
[8] C. Liu, J. Yuen, A. Torralba, Nonparametric scene parsing via label transfer, IEEE Trans. Pattern Anal. Mach. Intell. 33 (12) (2011) 2368–2382. http://dx.doi.org/10.1109/tpami.2011.131.
[9] A. Karpenko, P. Aarabi, Tiny videos: a large data set for nonparamet495-ric video retrieval and frame classification, IEEE Trans. Pattern Anal. Mach. Intell. 33 (3) (2011) 618–630. http://dx.doi.org/10.1109/tpami.2010.118.

[10] J. Yuen, B. Russell, C. Liu, A. Torralba, Labelme video: Building a video database with human annotations, in: Proceedings of the 2009 IEEE 12th International Conference on Computer Vision, IEEE, 2009, pp. 1451–1458. ⟨http://dx.doi.org/10.1109/iccv.2009.5459289⟩.
[11] M. Goto, R. Ishida, Y. Feng, S. Uchida, Analyzing the distribution of a large-scale character pattern set using relative neighborhood graph, in: Proceedings of the 2013 12th International Conference on Document Analysis and Recognition (ICDAR), IEEE, 2013, pp. 3–7. ⟨http://dx.doi.org/10.1109/icdar.2013.10⟩.
[12] D.L. Martin, Feature extraction: a survey, in: Proceedings of the IEEE, vol. 57(8), 1969, p. 1391. ⟨http://dx.doi.org/10.1109/proc.1969.7277⟩.
[13] G. Chandrashekar, F. Sahin, A survey on feature selection methods, Comput. Electr. Eng. 40 (1) (2014) 16–28. http://dx.doi.org/10.1016/j.compeleceng.2013.11.024.
[14] S. Arya, D.M. Mount, N.S. Netanyahu, R. Silverman, A.Y. Wu, An optimal algorithm for approximate nearest neighbor searching fixed dimensions, J. ACM (JACM) 45 (6) (1998) 891–923. http://dx.doi.org/10.1145/293347.293348.
[15] D. Nister, H. Stewenius, Scalable recognition with a vocabulary tree, in: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, IEEE, vol. 2, 2006, pp. 2161–2168. ⟨http://dx.doi.org/10.1109/cvpr.2006.264⟩.
[16] S. Garcia, J. Derrac, J.R. Cano, F. Herrera, Prototype selection for nearest neighbor classification taxonomy and empirical study, IEEE Trans. Pattern Anal. Mach. Intell. 34 (3) (2012) 417–435. http://dx.doi.org/10.1109/tpami.2011.142.
[17] R. Bellman, Adaptive Control Processes: a Guided Tour, vol. 4. ⟨http://dx.doi.org/10.2307/1909901⟩.
[18] P. Viola, M.J. Jones, Robust real-time face detection, Int. J. Comput. Vis. 57 (2) (2004) 137–154. http://dx.doi.org/10.1023/b:visi.0000013087.49260.fb.
[19] M. Pelillo, Similarity-Based Pattern Analysis and Recognition, Springer, London, UK, 2013. http://dx.doi.org/10.1007/978-1-4471-5628-4.
[20] A.L. Fred, A. Lourenço, H. Aidos, S.R. Bulò, N. Rebagliati, M.A. Figueiredo, M. Pelillo, Learning similarities from examples under the evidence accumulation clustering paradigm, in: Similarity-Based Pattern Analysis and Recognition, Springer, 2013, pp. 85–117. ⟨http://dx.doi.org/10.1007/978-1-4471-5628-4_5⟩.
[21] R.C. Wilson, E.R. Hancock, E. Pekalska, R.P. Duin, Spherical and hyperbolic embeddings of data, IEEE Trans. Pattern Anal. Mach. Intell. 36 (11) (2014) 2255–2269. http://dx.doi.org/10.1109/tpami.2014.2316836.
[22] M. San Biagio, S. Martelli, M. Crocco, M. Cristani, V. Murino, Encoding structural similarity by cross-covariance tensors for image classification, in: International Journal of Pattern Recognition and Artificial Intelligence, vol. 28(07). ⟨http://dx.doi.org/10.1142/s0218001414600088⟩.
[23] R.P. Duin, M. Bicego, M. Orozco-Alzate, S.-W. Kim, M. Loog, Metric learning in dissimilarity space for improved nearest neighbor performance, in: Structural, Syntactic, and Statistical Pattern Recognition, Springer, 2014, pp. 183–192. ⟨http://dx.doi.org/10.1007/978-3-662-44415-3_19⟩.
[24] K. Riesen, M. Neuhaus, H. Bunke, Graph embedding in vector spaces by means of prototype selection, in: Graph-Based Representations in Pattern Recognition, Springer, 2007, pp. 383–393. ⟨http://dx.doi.org/10.1007/978-3-540-72903-7_35⟩.
[25] B. Spillmann, M. Neuhaus, H. Bunke, E. Pekalska, R.P. Duin, Transforming strings to vector spaces using prototype selection, in: Structural, Syntactic, and Statistical Pattern Recognition, Springer, 2006, pp. 287–296. ⟨http://dx.doi.org/10.1007/11815921_31⟩.
[26] L. Batista, E. Granger, R. Sabourin, Applying dissimilarity representation to off-line signature verification, in: Proceedings of the 2010 20th International Conference on Pattern Recognition (ICPR), IEEE, 2010, pp. 1293–1297. ⟨http://dx.doi.org/10.1109/icpr.2010.322⟩.
[27] G.S. Eskander, R. Sabourin, E. Granger, Dissimilarity representation for handwritten signature verification., in: AFHA, Citeseer, 2013, pp. 26–30.
[28] E. Pekalska, R.P. Duin, Dissimilarity representations allow for building good classifiers, Pattern Recognit. Lett. 23 (8) (2002) 943–956. http://dx.doi.org/10.1016/s0167-8655(02)00024-7.
[29] E. Pekalska, R.P. Duin, P. Paclík, Prototype selection for dissimilarity-based classifiers, Pattern Recognit. 39 (2) (2006) 189–208. http://dx.doi.org/10.1016/j.patcog.2005.06.012.
[30] K. Riesen, H. Bunke, Graph Classification Based on Vector Space Embedding 23, World Scientific, Singapore, 2009, pp. 1053–1081. http://dx.doi.org/10.1142/s021800140900748x.
[31] Y. Plasencia-Calana, M. Orozco-Alzate, H. Méndez-Vázquez, E. García-Reyes, R.P. Duin, Towards scalable prototype selection by genetic algorithms with fast criteria, in: Structural, Syntactic, and Statistical Pattern Recognition, Springer, 2014, pp. 343–352. ⟨http://dx.doi.org/10.1007/978-3-662-44415-3_35⟩.
[32] G.S. Eskander, R. Sabourin, E. Granger, On the dissimilarity representation and prototype selection for signature-based bio-cryptographic systems, in: Similarity-Based Pattern Recognition, Springer, 2013, pp. 265–280. ⟨http://dx.doi.org/10.1007/978-3-642-39140-8_18⟩.
[33] S.K. Reed, Pattern recognition and categorization, Cogn. Psychol. 3 (3) (1972) 382–407. http://dx.doi.org/10.1016/0010-0285(72)90014-x.
[34] B. Iwana, S. Uchida, K. Riesen, V. Frinken, Tackling temporal pattern recognition by vector space embedding, in: Proceedings of the 2015 13th International Conference on Document Analysis and Recognition (ICDAR), 2015, pp. 816–820. ⟨http://dx.doi.org/10.1109/icdar.2015.7333875⟩.
[35] Y. Freund, R.E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, J. Comput. Syst. Sci. 55 (1) (1997) 119–139. http://dx.doi.org/10.1006/jcss.1997.1504.
[36] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classif. (2012). http://dx.doi.org/10.1007/s00357-007-0015-9.
[37] International Unipen Foundation, International Unipen Foundation –iUF. ⟨http://www.unipen.org/home.html⟩ (accessed 19.10.15).
[38] C. Bahlmann, B. Haasdonk, H. Burkhardt, Online handwriting recognition with support vector machines-a kernel approach, in: Proceedings of the Eighth International Workshop on Frontiers in Handwriting Recognition, IEEE, 2002, pp. 49–54. ⟨http://dx.doi.org/10.1109/iwfhr.2002.1030883⟩
[39] K. Sivaramakrishnan, K. Karthik, C. Bhattacharyya, Kernels for large margin time-series classification, in: Proceedings of the International Joint Conference on Neural Networks, IJCNN 2007 IEEE, 2007, pp. 2746–2751. ⟨http://dx.doi.org/10.1109/ijcnn.2007.4371393⟩.
[40] I. Kononenko, M. Kukar, Machine Learning and Data Mining: Introduction to Principles and Algorithms, Horwood Publishing, Chichester, UK, 2007. http://dx.doi.org/10.5860/choice.45-3834.
[41] Y. Freund, R.E. Schapire, A desicion-theoretic generalization of on-line learning and an application to boosting, in: Lecture Notes in Computer Science, Springer Science+Business Media, 1995, pp. 23–37. ⟨http://dx.doi.org/10.1007/3-540-59119-2_166⟩.
[42] A.J. Grove, D. Schuurmans, Boosting in the limit: Maximizing the margin of learned ensembles, in: Proceedings of the Fifteenth National Conference on Artificial Intelligence, 1998, pp. 692–699.
[43] H. Drucker, C. Cortes, Boosting decision trees, in: Advances in Neural Information Processing Systems, 1996, pp. 479–485.
[44] J.R. Quinlan, Bagging, boosting, and c4.5, in: Proceedings of the Thirteenth National Conference on Artificial Intelligence, AAAI Press, 1996, pp. 725–730.

**Brian Kenji Iwana** received a B.S. from the University of California, Irvine, United States of America in 2005. He is currently pursuing a Ph.D. degree from the Department Advanced Information Technology at Kyushu University, Japan. His research interests include pattern recognition, temporal patterns, and dissimilarity space embedding.

**Volkmar Frinken** is a researcher at Onai Technology and was formerly a Research Assistant Professor at Kyushu University, Japan. He obtained his M.S. in Computer Science in 2007 from the University of Dortmund, Germany and in 2011 his Ph.D. from the University of Bern, Switzerland. His research interests include handwriting recognition and keyword spotting, semisupervised learning, language models and document analysis.

**Kaspar Riesen** received his M.S. and Ph.D. degrees in Computer Science from the University of Bern in 2006 and 2009, respectively. He is interested in graph based representations in pattern recognition and related fields. Kaspar Riesen has more than 40 publications which are currently cited by 158 authors in more than 200 publications. Since 2009 he works at the University of Applied Sciences Northwestern Switzerland.

**Seiichi Uchida** received B.E. and M.E. and Dr. Eng. degrees from Kyushu University in 1990, 1992 and 1999, respectively. From 1992–1996, he joined SECOM Co., Ltd., Japan. Currently, he is a professor at Kyushu University. His research interests include pattern recognition and image processing. He received 2007 IAPR/ICDAR Best Paper Award, 2010, ICFHR Best Paper Award, and many domestic awards. Dr. Uchida is a member of IEEE, IEICE and IPSJ.