Tackling Temporal Pattern Recognition by Vector Space Embedding

Brian Iwana*, Seiichi Uchida*, Kaspar Riesen†, Volkmar Frinken*
*Kyushu University, Fukuoka, Japan
†University of Applied Sciences and Arts Northwestern Switzerland, Olten, Switzerland

***Abstract*—This paper introduces a novel method of reducing the number of prototype patterns necessary for accurate recognition of temporal patterns. The nearest neighbor (NN) method is an effective tool in pattern recognition, but the downside is it can be computationally costly when using large quantities of data. To solve this problem, we propose a method of representing the temporal patterns by embedding dynamic time warping (DTW) distance based dissimilarities in vector space. Adaptive boosting (AdaBoost) is then applied for classifier training and feature selection to reduce the number of prototype patterns required for accurate recognition. With a data set of handwritten digits provided by the International Unipen Foundation (iUF), we successfully show that a large quantity of temporal data can be efficiently classified produce similar results to the established NN method while performing at a much smaller cost.**

I. Introduction

It is a promising trend to use massive patterns for better pattern recognition accuracy. A famous work of 80 million tiny images [1] has shown that even a simple nearest neighbor classification performs sufficiently if we prepare a huge number of image instances. Similar trials with massive patterns has been made for image segmentation [2], video image recognition [3], [4], and handwritten character images [5] and has proved the potential of massive patterns.

On the other hand, massive patterns always encounter a problem of huge computations. There are three main remedies for this problem. The first remedy is to reduce the size of each pattern. For example, tiny images [1] and tiny videos [3] tried to avoid the problem by reducing image resolution to be very low. Low-dimensional representation of patterns is also a common technique for the size reduction. The second remedy is to use efficient nearest neighbor search schemes, such as approximate nearest neighbor and vocabulary tree search. The third remedy is to select meaningful patterns for a specific application. Prototype selection and edition and condensation are classical techniques [6] for this remedy.

This paper tackles temporal pattern recognition with massive patterns. An important characteristic of temporal patterns is that we must deal with their temporal fluctuation and variation in length. Especially, the variation in length is a problematic for massive pattern recognition because each temporal pattern will have a different dimensionality. Consequently, the common remedies for reducing computations in a fixed-dimensional Euclidean pattern space are not applicable anymore.

The main contribution of this paper is to propose a new discriminative prototype selection method for massive temporal pattern recognition using *dissimilarity space embedding*. Dissimilarity space embedding, or dissimilarity representation, emerged as a novel approach in pattern recognition [7]–[9]. The basic idea is to represent patterns (visual objects or even temporal patterns) by vectors of dissimilarities. The dissimilarity representation of patterns is based on pairwise comparisons, and is therefore also referred to as *relative representation*. The intuition of this approach is that the notion of proximity, i.e. similarity or dissimilarity, is more fundamental than that of a feature or a class. Similarity groups patterns together and should thus play a crucial role in the constitution of a class [10]. Using the notion of similarity, rather than features, renews the area of pattern recognition in one of its foundations, namely the representation of objects [11], [12].

In recent years, dissimilarity space embedding emerged as one of the fastest growing subfields in pattern recognition [13]. This can also be confirmed by the high number of recent publications which are concerned with various aspects of dissimilarity based pattern recognition [14]–[18]. Moreover, the idea of dissimilarity representation has been generalized to string based object representation [19] and eventually to the domain of graphs [20], [21].

In the proposed method, we first embed a temporal pattern $\mathbf{s}$ into a dissimilarity space by using dynamic time warping (DTW) with a large number of prototypes. Specifically, given n prototypes $\mathbf{p}_1, \ldots, \mathbf{p}_n$, we calculate n DTW distances (i.e., dissimilarities) then form an n-dimensional feature vector v_s whose i-th element is a DTW distance between $\mathbf{s}$ and $\mathbf{p}_i$. This vector representation suggests that any temporal pattern $\mathbf{s}$ is embedded into a fixed dimensional vector space regardless of length variations.

We then select discriminative prototypes by applying AdaBoost, which is a well-known machine learning method based on an ensemble of classifiers called weak-learners. Considering that the i-th feature element of v_s corresponds to a prototype, our prototype selection task is equivalent to the feature selection tasks in the dissimilarity space. On the other hand, as in [22], AdaBoost can be used for not only designing a strong classifier but also selecting discriminative features, if each weak learner only depends on a single feature element. We, thus, use AdaBoost for selecting discriminative prototypes, which span the dissimilarity space.

The remaining of this paper is organized as follows. Section II elaborates on dissimilarity space embedding and the application of DTW. In Section III, we introduce the application of AdaBoost for prototype selection. Section IV provides a more detailed description of the method proposed. In Section V, we confirm that the proposed method can select discriminative prototypes through a large scale experiment

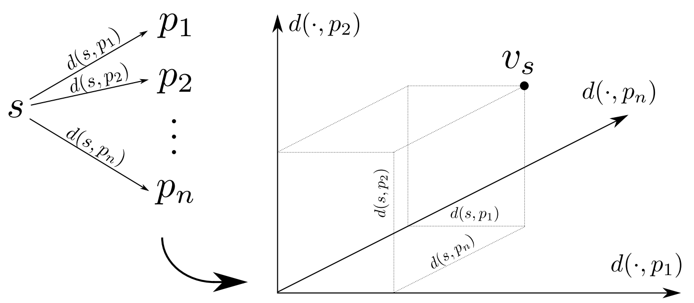

Fig. 1. A visualization of embedding an element $\mathbf{s}$ into an N-dimensional vector space using pairwise dissimilarities to prototypes.

on handwriting digit pattern recognition. Finally, Section VI draws a conclusion and describes future work.

II. Dissimilarity Space Embedding

A. Definition

Dissimilarity space embedding [7] is a general scheme to transform any data, on which a dissimilarity measure $d(\cdot,\cdot)$ is defined, into a N-dimensional vector space over the real numbers. As illustrated in Fig. 1, the key idea is to select n prototypes $\mathbf{p}_n$ for $n = 1,\ldots,N$ and represent any sample $\mathbf{s}$ as a vector whose entries are the distances $v_s = (d(\mathbf{s},\mathbf{p}_1), d(\mathbf{s},\mathbf{p}_2),\ldots,d(\mathbf{s},\mathbf{p}_N))^T$.

By means of this embedding, we can greatly enhance the possibilities to handle various data types. Sequential data, e.g., as in this paper, is not endowed with mathematically well defined operations such as addition or multiplication. Yet, after embedding a sequence into this vector space, we can make use of the full potential of existing algorithms that work on vectors.

Furthermore, instead of absolute values, features are now represented only in relation to the other features. The dissimilarity measure will capture the important differences between patterns. Another point of view is to see the embedding as a form of unsupervised learning. A meaningful similarity measure encodes already information about the membership likelihood of the prototypes' classes. Thus the position of the sample in the embedded space is strongly correlated with its class membership.

B. DTW-based Distances

The only operation needed for this embedding is a function on two data points that returns a real number. This number does not need to be a metric or a true distance function, but it should reflect a form of similarity between the data points.

Working on on-line trajectories of handwritten digits, dynamic time warping (DTW) is a natural choice for this distance measure. Given two sequences $\mathbf{s_1} = s^1_1 s^1_2 \cdots s^1_i \cdots s^1_I$ and $\mathbf{s_2} = s^2_1 s^2_2 \cdots s^2_j \cdots s^2_J$, DTW computes the optimal alignment between the sequence elements, with respect to the sum of pairwise sequence elements, as shown in Fig. 2. Note that several different ways on how the sequence element alignments exist in the literature. In this work, we chose the following rules. If s^1_i is aligned to s^2_j then s^1_{i+1} can only be aligned to s^2_j, s^2_{j+1}, or s^2_{j+2}. This ensures that first elements are aligned

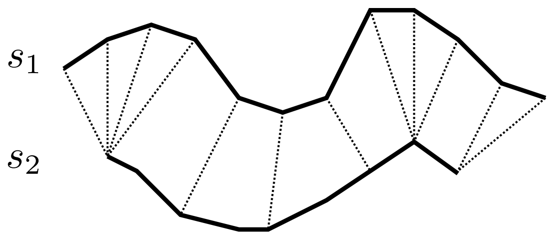

Fig. 2. DTW as the sum of optimally aligned pairwise element distances.

in order, since j is increasing monotonically, and second, that the number of alignments is always m, the length of sequence $\mathbf{s_1}$, regardless of $\mathbf{s_2}$. Hence, we do not need to care about normalization and can directly use the DTW distance to compare the dissimilarity distance between $\mathbf{s_1}$ and several other sequences. Note, that this distance is also not symmetric, yet still be able to be used for dissimilarity space embedding.

III. AdaBoost: General Definition and its Application to Prototype Selection

A. Definition

The adaptive boosting (AdaBoost) algorithm, introduced by Freund and Schapire [23], attempts to improve the accuracy of a learning algorithm by combining a set of weak learners to create an ensemble classifier. The algorithm works by iterating through multiple rounds of weak learner application on a set of weighted training patterns until a desired low training error has been achieved [23], [24]. To effectively adapt to difficult training patterns, the weights are designed to emphasize the patterns of incorrect classifications and understate the patterns of successful classifications. The advantage of AdaBoost is its speed and ability to improve the accuracy of less accurate classifiers. The algorithm has very little tuning and only requires the weak learners only need to be *better than chance* classification. This removes the need for each individual classifier to be effective over the entire space and allows for more simple classifiers to be trained.

The algorithm, shown in Fig. 3, creates a strong classifier out of a training set $(x_1, c_1), ..., (x_m, c_m), ..., (x_M, c_M)$ where x_i belongs to some domain space X and c_i is the labels which each x_i can be classified and M is the maximum number of m patterns. The idea is to apply a weak component classifier to the set of training patterns weighted by $W_k(m)$ over a number of AdaBoost rounds $k = 1, ..., k_{max}$. The final classifier $F(x)$, Fig. 3 Line 10, is calculated by using a weighted majority vote where α_k is the weights assigned to the weak learners $f_k(x)$ [25].

B. Prototype Selection with AdaBoost

The motivation behind using AdaBoost is to use it as a discriminative feature selector for the dissimilarity space. By determining which prototypes are relevant to an accurate classification of temporal patterns, we can reduce the computational requirement of using exhaustive classification algorithms. A typical k-NN classification method achieves accurate results by using all of the available prototypes to use as in the comparison. Because AdaBoost only requires the prototypes beneficial to its final composite classifier, we

```
1: function AdaBoost
2:     Training set $\{(x_1, c_1), ..., (x_m, c_m), ..., (x_M, c_M)\}$
3:     Initialize weights: $W_1(m) \leftarrow \frac{1}{M}$ for $m = 1, ..., M$
4:     for $k = 1, ..., k_{max}$ do
5:         Train weak classifier using $W_k(m)$.
6:         Select classifier with the lowest error:
               $$f_k(x) \leftarrow \arg\min_{f_j} \epsilon_j = \sum_{i=1}^{m} W_k(m)[f_k(x_m) \neq c_m]$$
7:         Calculate confidence:
               $$\alpha_k \leftarrow \frac{1}{2}\ln(\frac{1-\epsilon_k}{\epsilon_k})$$
8:         Update weights:
               $$W_{k+1}(m) \leftarrow \frac{W_k(m)}{Z_k} \times \begin{cases} e^{-\alpha_k}, & \text{if } f_k(x_m) = c_m \\ e^{\alpha_k}, & \text{otherwise} \end{cases}$$
9:     end for
10:    return composite classifier: $F(x) \leftarrow \sum_{k=1}^{k_{max}} \alpha_k f_k(x)$
11: end function
```

Fig. 3. AdaBoost algorithm.

can use it for feature selection by relying on the algorithm to discriminatively chose the relevant prototypes, minimizing the necessary calculations when used for test pattern classification. The number of prototype patterns required is reduced to $1 \leq n \leq k_{max}$, where n is the number of prototypes $\mathbf{p}_n$ and k is the number of iterations of weak learners determined by the algorithm. This can be a significant improvement over using the entire prototype collection.

IV. Prototype Selection Using AdaBoost with Dissimilarity Space Embedding

A. Dissimilarity Space Embedded with DTW

A dissimilarity space is created by using the DTW-distance between entire prototype and sample patterns as the dissimilarity measure. In Section II and Fig. 1, the samples within a dissimilarity space was described as $v_s = (d(\mathbf{s}, \mathbf{p}_1), d(\mathbf{s}, \mathbf{p}_2), \ldots, d(\mathbf{s}, \mathbf{p}_N))^T$. Using DTW-distance as the dissimilarity measure creates a resulting dissimilarity space with each sample m and the prototypes n represented as X_m, which is defined by:

$$X_m = (\text{DTW}(\mathbf{s}_m, \mathbf{p}_1), \text{DTW}(\mathbf{s}_m, \mathbf{p}_2), ..., \text{DTW}(\mathbf{s}_m, \mathbf{p}_N))^T$$

B. AdaBoost in Dissimilarity Space Embedded with DTW

To use AdaBoost, we selected a weighted nearest prototype classifier to be the weak component classifier. We chose two classes of handwritten on-line digits to be compared one at a time, class A and class B, to become the two-class classifier. More specifically, the weak classifier in Line 5 of Fig. 3, is defined by the comparison between the two classes, A and B, with each sample's DTW in $X_m(n)$, where $X_m(n)$ is the samples m in respect to the prototypes n. For each weak classifier, or prototypes-axis, in $n = 1, ..., N$, we determine the classifier with the lowest error for the step in Fig. 3 Line 5.

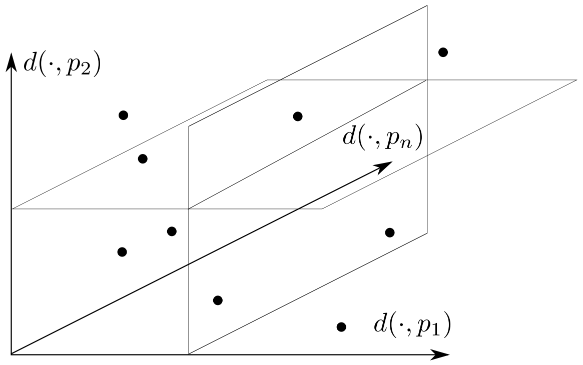

Fig. 4. A visualization of AdaBoost applied to a dissimilarity space. The points represent samples in the N-dimensional space. The planes depict weak classifiers in relation to a prototype axis that can be selected by AdaBoost.

This process is illustrated in Fig. 4, which demonstrates the comparison of two component classifiers that can be selected. These discriminatively selected classifiers are represented as the prototypes in relation to the training samples that are the most beneficial for the ensemble classifier produced by AdaBoost.

V. Experimental Results

A. Data Set

The UNIPEN on-line handwritten data set, a collection of handwritten characters managed by the International Unipen Foundation (iUF), was used for the experiment. The iUF provides benchmark data sets for handwriting recognition and writer identification algorithms [26]. The portion of the data set used for the experiments contains about 15,000 patterns made of time ordered sets of coordinates split into ten classes of isolated handwritten digits. The on-line handwritten numerical digit paths provides us a method of demonstrating our described method on real temporal patterns with the benefit of being able to be classified into distinct categories.

The data set was divided into three subsets, a prototype set, a training set, and a test set. For the experiment, we used a two-class classifier comparing to individual classes and repeated the experiment for all possible combinations of classes. Each prototype set was made of 260 patterns, 130 patterns of each class. The size of each test set was similarly defined at 260 patterns. Finally, the training sets for the dissimilarity space embedding and AdaBoost contained 2080 patterns. This process was repeated using different partitions of the same data set for 9-fold cross-validation.

B. Evaluations

In this section, we consider the following three evaluations:

- *Dissimilarity space embedding with AdaBoost (DSE AdaBoost Classified)*: This is the method described in this paper. The testing set of data was evaluated against the strong classifier trained by AdaBoost.
- *1-NN on random prototypes (1-NN Randomly Selected)*: To serve as our baseline evaluation, a standard 1-NN method was used on a set of random prototypes

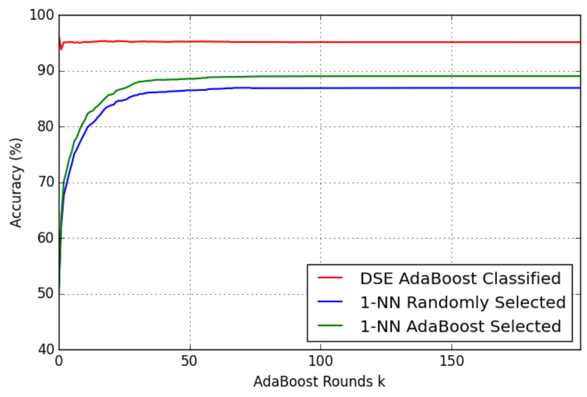

Fig. 5. A comparison of a sample fold between the dissimilarity space embedding with AdaBoost method, the 1-NN with randomly selected prototypes baseline method, and the 1-NN with AdaBoost selected prototypes method.

with a size exactly equal to the number of prototypes selected by the AdaBoost algorithm. The baseline 1-NN method is also given a two-class classification problem and is repeated for all possible combinations.

- *1-NN on AdaBoost selected prototypes (1-NN AdaBoost Selected)*: This is similar to the 1-NN on random prototypes method, but instead of randomly selecting prototypes, this method uses the prototypes selected by AdaBoost in dissimilarity space. This shows the discriminative potential of prototype selection with AdaBoost and provides a demonstrative stepping stone to the results obtained by dissimilarity space embedding with AdaBoost.

C. Results

A comparison of the results of the test evaluations are shown in Fig. 5. Each line is an average of all of the classifications and all of the cross-validation folds over the course of each round of AdaBoost.

Using dissimilarity space embedding with AdaBoost with a two-class classification, an average of 95.15% accuracy was obtained after 200 rounds. Fig. 6 shows a map of the matrix with each column representing the tested two-class combinations and the error rate of each respective round of AdaBoost indicated by a shade. Due to the averaging, it is less evident on Fig. 5, but the results displayed on Fig. 6(b) shows that for many of the comparisons, the test accuracy exhibited the expected improvements and behavior of AdaBoost. It also shows that for some of the classifications, for example the comparison between the class "1" and class "5", had a neutral or adverse affect from AdaBoost.

The dissimilarity space embedding with AdaBoost method had a dramatic increase in accuracy compared to 1-NN with comparatively reduced prototypes. The evaluation of 1-NN with randomly selected prototypes and the evaluation of 1-NN with AdaBoost selected prototypes had an average accuracy of 86.94% and 89.06% respectively. Even without using dissimilarity space embedding and without the learned classifier, the AdaBoost selected prototype 1-NN was able to out-perform the randomly selected method. This out-performance of shows

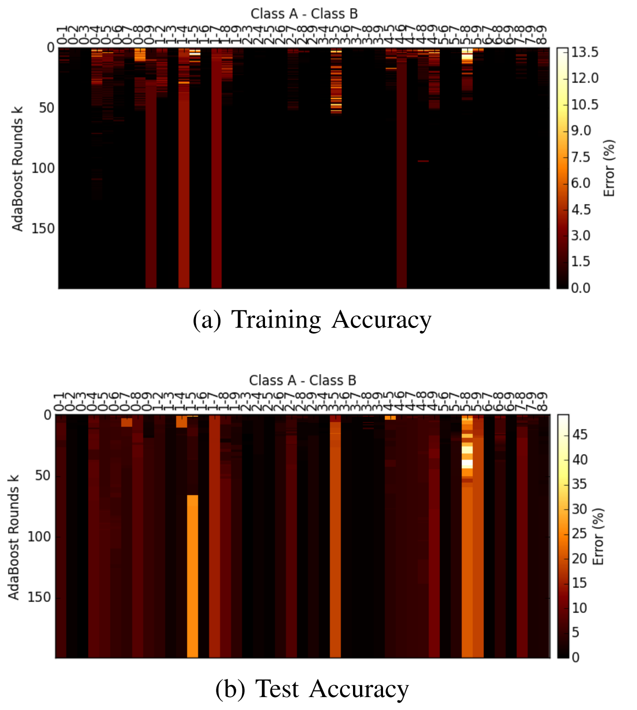

Fig. 6. An overview of the tested error rates of each combination of classes using dissimilarity space embedding with AdaBoost for a sample fold. (a) shows the error rates calculated in the training step and (b) is the observed testing error rate of the ensemble classifier.

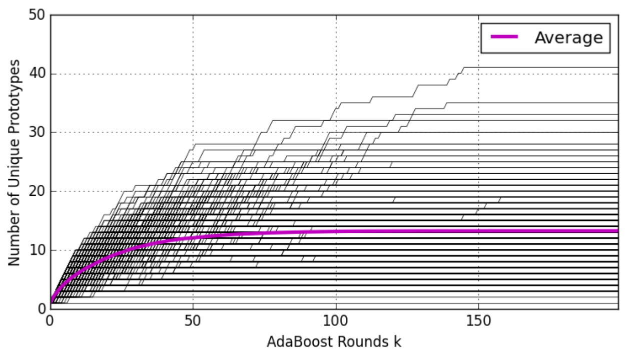

Fig. 7. This graph shows the the number of unique prototypes selected by AdaBoost over the number rounds of weak learner decisions.

the discriminative ability of the AdaBoost algorithm. It also illustrates the contributing factor that just the prototype selection has on the final outcome of the ensemble classifier.

When the weak learner is applied to each round of AdaBoost across the prototype vector space, only the weighted classifier with the lowest error rate is selected. Because the algorithm can repeat the use of a past selection, after 200 rounds, only an average 6.63% of the original prototypes were needed for the final composite classifier. Fig. 7 shows a graph of the reduction in prototypes selected when using AdaBoost. It shows the diversity in number of unique prototypes selected for each of the comparisons and displays the average. As an example case, the prototypes selected by AdaBoost in the comparison of class "4" and class "6" for a sample fold is

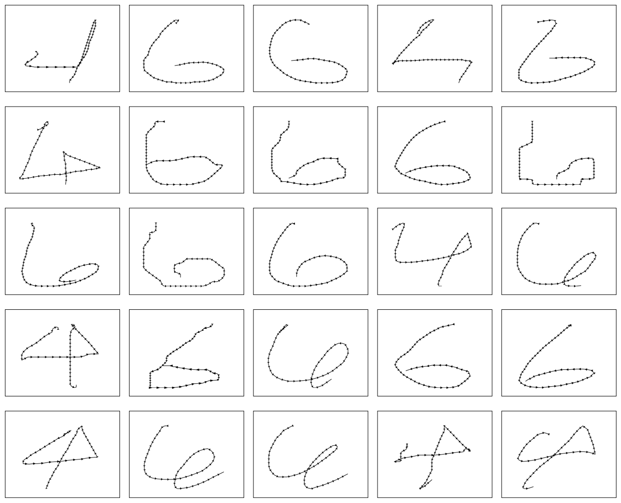

Fig. 8. The unique prototype patterns discrimitely selected by AdaBoost between class "4" and class "6' in order of appearence.

shown in Fig. 8. The example case 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.

VI. Conclusion

We employed 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 feature 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, was able to maintain an accuracy of 95.15% while reducing the number of prototypes to an average of 6.63%. The proposed method has potential to improve the availability and scalability of pattern recognition on a massive data set.

This paper lays a groundwork for temporal pattern vector space with machine learning algorithms and prototype selection. Future work will be done in the analysis of the possible trends or specific prototype patterns selected by AdaBoost. Further research also lies in the exploration of dissimilarity embedding based on different distance measures as well as different machine learning methods for the purpose of pattern recognition with a consideration to massive data by prototype selection.

References

[1] A. Torralba, R. Fergus, and W. T. Freeman, "80 Million Tiny Images: A Large Data Set for Nonparametric Object and Scene Recognition," *TPAMI*, 30(11), 2008.

[2] C. Liu, J. Yuen, and A. Torralba, "Nonparametric Scene Parsing via Label Transfer," *TPAMI*, 33(12), 2011.

[3] A. Karpenko and P. Aarabi, "Tiny Videos: A Large Data Set for Nonparametric Video Retrieval and Frame Classification," *TPAMI*, 33(3), 2011

[4] J. Yuen, B. C. Russell, C. Liu, and A. Torralba. "LabelMe Video: Building a Video Database with Human Annotations," *ICCV*, 2009.

[5] M. Goto, R. Ishida, Y. Feng and S.=Uchida, "Analyzing the Distribution of a Large-scale Character Pattern Set Using Relative Neighborhood Graph", *ICDAR* pp.3-7, 2013.

[6] S. Garcia, J. Derrac, J. R. Cano, and F. Herrera, "Prototype Selection for Nearest Neighbor Classification: Taxonomy and Empirical Study", *TPAMI*, 34(3), 2012

[7] E. Pekalska, R. Duin, *The Dissimilarity Representation for Pattern Recognition: Foundations and Applications*, River Edge: World Scientific Publishing Co., Inc., NJ, 2005.

[8] E. Pekalska, R. Duin, "Dissimilarity representations allow for building good classifiers," *Pattern Recognition Letters*, vol. 23, no. 8, pp. 943-956, Jun. 2002.

[9] E. Pekalska, R. Duin, P. Paclik, "Prototype selection for dissimilarity-based classifiers", *Pattern Recognition*, vol. 29, no. 2, pp. 189-208, Feb. 2006.

[10] S. Edelman, *Representation and Recognition in Vision*, Brandford Books, 1999.

[11] V. Mottl, S. Dvoenko, O. Seredin, C. Kulikowski, I. Muchnik, "Featureless regularized recognition of protein fold classes in a hilbert space of pairwise alignment scores as inner products of amino acid sequences," *Pattern Recognition and Image Analysis: Advances in Mathematical Theory and Applications*, vol. 11, no. 3, pp. 597-615, 2001.

[12] R. Duin, D. de Ridder, D. Tax, "Featureless pattern classification", *Kybernetika*, vol. 34, no. 4, pp. 399-404, 1998.

[13] M. E. Pelillo, ed, *Similarity-Based Pattern Analysis and Recognition* (Advances in Computer Vision and Pattern Recognition No. XIV), London, England: Springer-Verlag, 2013

[14] A. Fred, A. Lourenço, H. Aidos, S. Bulò, N. Rebagliati, M. Figueiredo, M. Pelillo, "Learning Similarities from Examples Under the Evidence Accumulation Clustering Paradigm", in *Similarity-Based Pattern Analysis and Recognition* (Advances in Computer Vision and Pattern Recognition No. II), London, England: Springer-Verlag, 2013, pp. 85-117

[15] R. Wilson, R. Duin, E. Pekalska, E. Hancock, "Spherical and Hyperbolic Embeddings of Data," TPAMI, vol. 36, no. 11, pp. 2255-2269, 2014

[16] M. San Biagio, S. Martelli, M. Crocco, M. Cristiani, M., V. Murino, "Encoding structural similarity by cross-covariance tensors for image classification," *Int. Journal of Pattern Recognition and Artificial Intelligence*, vol. 28, no. 7, 2014

[17] Y. Calana, M. Orozco-Alzate, H. Mendez-Vazquez, E. García-Reyes, R. Duin, "Towards scalable prototype selection by genetic algorithms with fast criteria," in *Structural, Syntactic, and Statistical Pattern Recognition*, Springer, 2014, LNCS vol. 8621, pp. 343-352.

[18] R. Duin, M. Bicego, M. Orozco-Alzate, S. Kim, M. Loog, "Metric learning in dissimilarity space for improved nearest neighbor performance," in *Structural, Syntactic, and Statistical Pattern Recognition*, Springer, 2014, LNCS vol. 8621, pp. 183-192.

[19] B. Spillmann, M. Neuhaus, H. Bunke, E. Pekalska, R. Duin, "Transforming strings to vector spaces using prototype selection," in *Structural, Syntactic, and Statistical Pattern Recognition*, Springer, 2006, LNCS vol. 4109, pp. 287-296.

[20] 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, LNCS vol. 4538, pp. 383-393.

[21] K. Riesen, H. Bunke, "Graph classification based on vector space embedding," *Int. Journal of Pattern Recognition and Artificial Intelligence*, vol. 23, no. 6, pp. 1053-1081, Sept. 2009.

[22] P. Viola and M. Jones, "Robust Real-Time Face Detection", *IJCV*, 57(2), 2004.

[23] Y. Freund and R. E. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119-139, 1997.

[24] R. O. Duda, P. E. Hart and D. G. Stork, "Boosting," in *Pattern Classification*, 2nd ed., John Wiley & Sons, Inc., 2001, pp. 476-480.

[25] Y. Freund and R. E. Schapire, "A Short Introduction to Boosting," *Journal of Japanese Society for Artificial Intelligence*, vol. 14, no. 5, pp. 771-780, Sept. 1999.

[26] International Unipen Foundation, "Int. Unipen Foundation - iUF," [Online]. Available: http://www.unipen.org/home.html. [Accessed 16 January 2015]