Towards Scalable Prototype Selection by Genetic Algorithms with Fast Criteria

Yenisel Plasencia-Calaña^{1,2}, Mauricio Orozco-Alzate³,
Heydi Méndez-Vázquez¹, Edel García-Reyes¹, and Robert P.W. Duin²

¹ Advanced Technologies Application Center, 7ma A # 21406, Playa, Havana, Cuba
{yplasencia, egarcia, hmendez}@cenatav.co.cu

² Faculty of Electrical Engineering, Mathematics and Computer Sciences,
Delft University of Technology, The Netherlands
r.duin@ieee.org

³ Departamento de Informática y Computación, Universidad Nacional de Colombia -
Sede Manizales. Kilómetro 7 vía al Aeropuerto, Campus La Nubia – Bloque Q,
Piso 2, Manizales, Colombia
morozca@unal.edu.co

Abstract. How to select the prototypes for classification in the dissimilarity space remains an open and interesting problem. Especially, achieving scalability of the methods is desirable due to enormous amounts of information arising in many fields. In this paper we pose the question: are genetic algorithms good for scalable prototype selection? We propose two methods based on genetic algorithms, one supervised and the other unsupervised, whose analyses provide an answer to the question. Results on dissimilarity datasets show the effectiveness of the proposals.

Keywords: dissimilarity space, scalable prototype selection, genetic algorithm.

1 Introduction

The vector space representation is a common option to represent the data for learning tasks since many statistical techniques are applicable for this kind of representation. However, there is an increasing number of real-world problems which are not vectorial. Instead, the data are given in terms of pairwise dissimilarities which may be non-Euclidean and even non-metric. In [1] several approaches were presented to learn from dissimilarity data, where the dissimilarity space (DS) has several advantages over the other approaches. In the DS approach, the dissimilarities of the training objects to the representation set are interpreted as coordinates in the space. Class separability is a property that one wants to maintain after the mapping which can be accomplished by a careful selection of the prototypes. However, a random selection was found to perform well for large numbers of prototypes [2]. Since this is the fastest method, the selection of good prototypes by more dedicated methods is of interest only for small numbers of prototypes. A good method must be able to find a minimal set without a significant decrease in accuracy of classifiers in the DS.

Several methods have been proposed [2,3] to find small representation sets by supervised or unsupervised strategies. Supervised methods have the advantage of maintaining a high accuracy, however this is achieved at high computational costs; besides, they might suffer from overfitting. Unsupervised methods have the advantages of being fast, generalizing well and avoiding overfitting. However, as a disadvantage, they are not always good in maintaining class separability since class labels are not taken into account. When the purpose is to learn from large datasets, the scalability of the method must also be considered since it is known that the prototype selection problem is *NP*-complete. This has been overlooked so far and only some studies such as the one in [4] copes with the problem. Large datasets arise in several situations and being able to deal with them is of interest. Some of the issues that cause the existence of large datasets are: vast amounts of data due to dropping costs for capturing, transmitting, processing and storing; modalities that have millions of classes such as biometrics; and modalities that have hundreds of thousands samples such as brain tractography data in [4].

This paper is aimed to study if genetic algorithms (GAs) are good for scalable prototype selection and if a fast clustering replacing the random initialization can help to improve the results further. It is usually assumed in the literature that linear-time algorithms are acceptable for scaling up to large datasets [5]. We also adopt this assumption. The parameter that dominates the complexity in our problem is the number of samples, since we assume that the other parameter, the number of selected prototypes, will always be small and that the full dissimilarity matrix is already computed. Some strategies to cope with scalability include parallelism, techniques to work with data that do not fit into memory, stochastic methods, etc. However in this study we focus on the time complexity and, in case the full dissimilarity matrix do not fit into memory, the developed approaches are easily adaptable by loading the dissimilarities on demand.

In [6], a GA was proposed for prototype selection but it is not able to cope with scalability issues. Here, we develop two versions of a scalable GA that optimize two different criteria for prototype selection. The remaining part of the paper is organized as follows: Section 2 presents the dissimilarity representation and prototype selection, Section 3 presents the two proposed methods. Experimental results are reported and discussed in Section 4 and conclusions are drawn in Section 5.

2 Dissimilarity Space and Prototype Selection

The dissimilarity space was proposed by Pekalska and Duin [1]. It was postulated as a Euclidean vector space, which allows the use of several classifiers. Let X be the space of objects which may not be vectorial, let $R = \{r_1, r_2, \dots, r_k\}$ be the set of prototypes such that $R \in X$, and let $d : X \times X \mapsto \mathbb{R}^+$ be a suitable dissimilarity measure for the problem. The prototypes may be chosen based on some criterion or even at random; however, the goal is that they have good representation capabilities specially when pursuing small representation sets. For a finite training set $T = \{x_1, x_2, \dots, x_n\}$ such that $T \in X$, the dissimilarity space is created by the data dependent mapping $\phi_R^d : X \mapsto \mathbb{R}^k$ where:

$$\phi_R^d(x_i) = [d(x_i, r_1) \ d(x_i, r_2) \ \dots \ d(x_i, r_k)]. \quad (1)$$

For dissimilarity representations, the adaptation of prototype selection techniques for improving the k -nearest neighbour (k -NN) classifier as well as the adaptation of clustering techniques have been investigated showing good results [2]. Other approaches use the geometry and the distribution of the objects to find the prototypes [7]. In [2], various techniques were compared such as Kcentres, mode seeking, forward selection (FS), linear programming, editing and condensing, and a mixture of Kcentres with linear programming. These techniques showed good performances. However, some of them are computationally expensive for very large datasets such as the FS, which runtime is quadratic in the number of samples.

3 Proposed Methods

We propose two different variants of GA with different scalable criteria for prototype selection. The two methods receive as parameter the desired number of prototypes. Finding an appropriate number of prototypes for each particular problem is out of the scope of this paper, however for this purpose some methods can be applied. For example, a good practice is to find the intrinsic dimensionality and select the number of prototypes accordingly. We assume we have a square dissimilarity matrix D among all training samples; the prototypes will be selected from this set of samples. Note that our goal is not to generate new prototypes as combinations of the original ones, but to select ones that already exist.

The GA is a biologically motivated search method which explores individuals (chromosomes or solutions) created after each generation by the best fitted ones. This property of GAs makes them much better scalable than using a full search. In our problem, each individual is a set of prototypes of fixed cardinality k codified in a k - *vector* containing in each position the index of the potential prototype. For example the 5 - *vector* (65, 30, 7, 19, 87) codifies an individual representing a set of 5 potential prototypes which can be accessed in some data structure by the indexes 65, 30 and so on. The GA starts the search in an initial population of randomly generated individuals.

Before executing the GA, we propose to perform a 1-step K-centres clustering in the space of candidates, where the number of clusters equals the desired number of prototypes. The candidates are clustered in order to guide the GA search in such a way that it has a faster convergence. The clustering runtime is $O(nk)$, where $n = |T|$, being T the training/validation set and $k = |R|$. The GA is slightly modified since its initial population is now generated by randomly sampling one potential prototype per cluster. Therefore, each element of an individual (also named gene) is linked to a particular cluster since only objects from that cluster are allowed in the corresponding position of the individual. In each generation, the best solution according to the fitness function is found and reproduced with each member of the population with a preset probability by

Algorithm 1. Genetic Algorithm

Input: D : dissimilarity matrix among samples and candidates to prototypes; k : desired number of prototypes, S : number of individuals in the population, rp : reproduction probability, mp : mutation probability, $iter$: number of generations

Output: $bestindividual$: set of prototypes indexes

```

// perform a 1-step Kcentres clustering in the candidates to
// prototypes to find  $k$  clusters
1  $cluslabs \leftarrow Kcentres(D, k, 1)$ ;
// randomly generate the population ensuring that, in the  $j$ -th
// position of the individual, only objects belonging to the  $j$ -th
// cluster are allowed
2  $P \leftarrow GenerateInitialPopulation(cluslabs, D, k, S)$ ;
3 while number of generations < iter do
    // find the best solution from the population and assign it to
    //  $bestindividual$ 
4   foreach currentindividual in  $P$  do
5     if  $Fitness(currentindividual, D) > Fitness(bestindividual, D)$ 
6       then
7          $bestindividual \leftarrow currentindividual$ ;
8     end
    // Evolution cycle
9   foreach currentindividual in  $P$  do
    // Reproduction, replace a gene of currentindividual with
    // probability  $rp$  by a gene of the best
10     $Reproduce(bestindividual, currentindividual, rp)$ ;
    // Mutation, change a gene of currentindividual with
    // probability  $mp$ 
11     $Mutate(currentindividual, mp)$ ;
12  end
13 end

```

gene using uniform reproduction or crossover. Elitist selection is performed since the best fitted individual is retained for the next generation without undergoing mutation; in addition, only the best fitted individual is selected as parent of the next population of individuals. The rest of the population undergoes gene mutation with a preset probability which is usually small but also with the constrain that the new index codified in a gene must belong to the related cluster. The pseudo-code is presented in Algorithm 1.

To achieve full scalability these methods should be able to handle: (1) large sets of candidates to prototypes, (2) large number of individuals in the search space of the GA and (3) large number of samples to be used (if needed) to compute the fitness function. In our proposal, the GA handles well (1) large sets of candidates for prototypes since we discarded the standard binary codification of individuals that demands vectors of length equal to n where $n \gg k$. Instead,

we resorted to vectors of length equal to the number of prototypes k since we codify only the indexes of the prototypes to be evaluated. Scalability in the number of individuals to analyze in the search space (2) is achieved since the stopping condition is a small predefined number of GA generations that does not depend on the number of individuals in the search space. A small number of generations is sufficient for GA's convergence thanks to its guided sampling since not all the possible combinations of prototypes are explored but only the best ones which arise after each generation. In addition, the initial clustering helps to avoid redundant prototypes in the same individual. Scalability in the number of samples to be used to compute the fitness (3) will be explained in the next subsections.

3.1 Minimum Spanning Tree Based Unsupervised Criterion

In the fitness function computation, the set of prototypes being evaluated and the training samples are usually involved. However, note that the proper number of prototypes k depends on the intrinsic dimension of the data which is usually small, thereby, $n \gg k$. For large datasets this implies that the dominant term for the fitness computation is the total number of samples n . To achieve scalability in the fitness function it must scale well to a large n . This highly depends on how the criterion to be optimized in the fitness function by the GA is conceived. Our first proposal for GA criterion is based on the minimum spanning tree (MST) of a set of prototypes. The prototypes are interpreted as nodes in a graph and the dissimilarity values between prototypes correspond to edge weights. The sum of edge weights (named tree weight) is used as criterion to be maximized, thereby improving the coverage over the DS. As we used the Prim's algorithm to find the MST and the graph is complete, the computation of this criterion has a runtime of $O(k^2 \log(k))$. Therefore, it is completely independent on the large number of samples n and as a consequence highly scalable for very large problems. The pseudo-code is presented in Algorithm 2.

The total runtime of the proposed GA with this criterion is as follows. For computing the initial clustering of the prototypes the runtime is $O(nk)$, for the fitness function $O(k^2 \log(k))$, and $O(k)$ for mutation and reproduction since each position of the vector representing an individual has to be analyzed. The dominant term in the whole procedure is $O(nk)$, as we assume that the desired number of prototypes is small and fixed, therefore the total runtime is $O(nk)$. However, if the initial clustering step is discarded, the complexity is only $O(k^2 \log(k))$, which in case of needing sub-linear (in n) methods is more appropriate.

3.2 Supervised Criterion Based on Counting Matching Labels

Our second criterion proposal is a linear-time supervised criterion that is different from previous expensive supervised ones [8] since it does not compute a classification error in the DS or an intra-class distance, which are usually quadratic. Our method, instead, considers each candidate to prototype as a representative of a cluster and every object in T is assigned to the cluster represented by its

Algorithm 2. Unsupervised Fitness function by MST

Input: w : vector of prototypes indexes; D : dissimilarity matrix
Output: res: fitness value
 // Interpret the prototypes indexed in w as nodes and
 dissimilarities among them as edges weights of a complete
 graph $G = (V, E)$
 // compute minimum spanning tree by Prim's algorithm
 1 $V' \leftarrow V[1];$
 2 $k \leftarrow |V|;$
 3 $E' \leftarrow \emptyset;$
 4 **while** $|E'| < k - 1$ **do**
 // select an edge of minimum weight which connects one node in
 V' with a node which is not in V'
 // add the new edge to E' , add the new node to V'
 5 **end**
 6 $T \leftarrow (V', E');$
 // sum all the weights of edges in E'
 7 $res \leftarrow \text{SumWeights}(E');$

Algorithm 3. Supervised Fitness function

Input: w : vector of prototypes indexes; D : dissimilarity matrix
Output: res: fitness value
 // interpret the prototypes r_j indexed in w as centers of
 clusters
 1 $res \leftarrow 0;$
 2 **foreach** $x \in T$ **do**
 // find the nearest prototype of x
 3 $r' \leftarrow \text{argmin}(D[x, r_j]);$
 4 **if** $\text{getclasslabel}(x) = \text{getclasslabel}(r')$ **then**
 5 $res \leftarrow res + 1;$
 6 **end**
 7 **end**

nearest prototype. The proposed criterion counts the number of assigned objects whose labels match their representative label. The best solution is the one that maximizes this value. This has the smallest runtime for a supervised method that uses all the samples ($O(nk)$). The pseudo-code is presented in Algorithm 3.

The runtime of the whole supervised GA is as follows. For computing the clustering the runtime is $O(nk)$, for the fitness function $O(nk)$, and for mutation and reproduction $O(k)$. The total runtime is $O(nk)$. However, in practice, this is higher than the unsupervised procedure since it is multiplied by a high constant due to the cost for comparing the labels. In general these times are better than or comparable to other linear (in n) algorithms compared in our experiments such as the Kcentres which is $O(nk)$ and the farthest first transversal (FFT) which is also $O(nk)$.

4 Experiments

4.1 Datasets and Experimental Setup

Four different datasets of moderate size were used for the experiments: the Zongker data [9] computed by deformable template matching, Pendigits [10,11] computed by edit distances, XM2VTS [12] computed by chi square distances on LBP histograms, and Diabetes [13] computed using Euclidean distances on features. The characteristics of the datasets as well as the cardinality of the training sets used are summarized in Table 1.

Table 1. Characteristics of the datasets used in this study, the $|T|$ column refers to the training/validation set cardinality used for the experiments

Datasets	# Classes	# Obj	Metric	$ T $
Zongker	10	200×10	no	1000
Diabetes	2	500/268	yes	384
Pendigits	10	10992	no	5000
XM2VTS	295	12×295	yes	1770

The datasets were randomly divided 30 times into training/validation set and test set. The validation set is used to optimize the criteria. The candidates to prototypes also belong to the validation set. The best performing classifier per dataset between the linear discriminant classifier (LDC) and the 1-NN classifier was used to report the classification errors for the different prototype selection methods compared, which are: random selection, FS [2] optimizing the supervised criterion; FFT [4], Kcentres [2], GA in the space of clustered prototypes with the proposed unsupervised fitness function based on MST (GA (clust) span), GA with the proposed unsupervised fitness function based on MST without clustering the prototypes (GA span), GA in the space of clustered prototypes with the proposed supervised fitness function (GA (clust) sup), and GA with the proposed supervised fitness function (GA sup).

The parameters used for the GA are: 20 individuals for the initial population, 0.5 for reproduction probability per gene, 0.02 for mutation probability per gene, the stopping condition is 20 generations reached for the GA with initial clustering in the space of prototypes, and 25 for the GA without the clustering.

4.2 Results and Discussion

From test set errors, for an increasing number of GA generations, we observed that 20 generations provide a good compromise for acceptable classification error at acceptable runtime. We used this number in all the experiments. Figure 1 presents the average errors over 30 experiments for 10, 15, 20, 30, 40 and 50 prototypes. It can be seen in Fig. 1(a) that, for the Zongker dataset, the best results are obtained with the FS and the supervised criterion, also the proposed

Fig. 1. Average errors for different numbers of prototypes for the best performing classifier in each dataset

GA with this criterion outperforms the other unsupervised methods in accuracy with a comparable efficiency. The FS outperformed the GA in this dataset because it does not present a significant class overlap. However, when there is a significant overlap among the classes as in XM2VTS (see Fig. 1(d)), the GA outperforms the FS since it takes the relation among all the prototypes into account. One problem of the FS that causes its lower performance is that when an object is selected to be a prototype, it cannot be discarded afterwards.

From results on Diabetes dataset in Fig. 1(c), it can be seen that the proposed GA with the unsupervised criterion outperforms the other methods except for the FFT which has a similar performance but at a higher computational cost. From results for Pendigits in Fig. 1(b) we find again that this method is among the best performing ones both in speed and accuracy (see Fig. 2(b)); in addition, the computation of times for some of the datasets reported in Fig. 2 shows that the unsupervised GA is the fastest method. The supervised proposal is comparable to other unsupervised methods in execution times. This analysis, together with the computational complexity analysis, indicates that the presented GAs are able to scale well to large datasets when the final goal is the selection of

Fig. 2. Average times in seconds plotted in log scale

prototypes. However, the fitness function must also be designed scalable. As an extra bonus, GAs are embarrassingly parallelizable. Regarding our second question whether cluster analysis may be helpful, we see that results with initial clustering are usually equal to or better than those without clustering the prototypes before executing the GA, except for the XM2VTS complicated dataset that presents a significant class overlap.

In the XM2VTS we find that the best method is the proposed supervised GA but without the clustering. A clear explanation of why this happens was derived from the data exploration by multidimensional scaling (MDS) and the knowledge of the dataset characteristics. This is a dataset with strong illumination changes: frontal, right, and left illuminations of the faces. The three different illuminations create three large clusters of objects, so they are determined by noise instead of by the identities. Due to this, the unsupervised clustering before the GA does not find suitable clusters. However, our supervised criterion handles the class overlap well. Thereby, the supervised clustering that we evaluate in our criterion is not linked to the initial unsupervised clustering of the space of prototypes. We analyzed relations between performance of methods and data distribution by inspecting the MDS plots. We found that the supervised method handles well datasets with homogeneous distributions or with class overlap. In contrast, the unsupervised criterion copes better with inhomogeneous distributions where we can find inside the same class densely populated regions as well as sparse ones. In addition, the MST-based unsupervised method handles well elongated classes as in the Diabetes dataset.

5 Conclusions

The selection of prototypes is a crucial step for classification in the dissimilarity space. In this paper we proposed two different prototype selection methods by a GA and two different supervised and unsupervised criteria. Our work focuses on achieving low computational costs by finding approximate but sufficiently good solutions by powerful search heuristics, and maintaining low asymptotic complexities in the fitness function. Experimental results showed the validity of

the proposals for selecting good prototypes. The runtime analysis showed that the proposed methods are able to scale well to large datasets. Other general approaches include parallelism, stochastic methods etc. The proposed unsupervised method is the fastest since the evaluation of its criterion does not depend on the size of the dataset but on the number of prototypes. Besides, the linear time supervised criterion is also very fast compared to other supervised ones, which are generally quadratic and thereby do not scale well, and is comparable to unsupervised methods.

References

1. Pekalska, E., Duin, R.P.W.: The Dissimilarity Representation for Pattern Recognition: Foundations and Applications (Machine Perception and Artificial Intelligence). World Scientific Publishing Co., Inc., River Edge (2005)
2. Pekalska, E., Duin, R.P.W., Paclík, P.: Prototype selection for dissimilarity-based classifiers. *Pattern Recogn.* 39(2), 189–208 (2006)
3. Bunke, H., Riesen, K.: Graph classification based on dissimilarity space embedding. In: da Vitoria Lobo, N., Kasparis, T., Roli, F., Kwok, J.T., Georgiopoulos, M., Anagnostopoulos, G.C., Loog, M. (eds.) *S+SSPR 2008*. LNCS, vol. 5342, pp. 996–1007. Springer, Heidelberg (2008)
4. Olivetti, E., Nguyen, T.B., Garyfallidis, E.: The approximation of the dissimilarity projection. In: *Second Workshop on Pattern Recognition in NeuroImaging, PRNI 2012*, pp. 85–88. IEEE Computer Society, Washington, DC (2012)
5. García-Pedrajas, N., Haro-García, A.: Scaling up data mining algorithms: review and taxonomy. *Progress in Artificial Intelligence* 1(1), 71–87 (2012)
6. Plasencia-Calaña, Y., García-Reyes, E., Orozco-Alzate, M., Duin, R.P.W.: Prototype selection for dissimilarity representation by a genetic algorithm. In: *Proceedings of the 20th International Conference on Pattern Recognition, ICPR 2010*, pp. 177–180. IEEE Computer Society, Washington, DC (2010)
7. Lozano, M., Sotoca, J.M., Sánchez, J.S., Pla, F., Pekalska, E., Duin, R.P.W.: Experimental study on prototype optimisation algorithms for prototype-based classification in vector spaces. *Pattern Recogn.* 39(10), 1827–1838 (2006)
8. Zare Borzeshi, E., Piccardi, M., Riesen, K., Bunke, H.: Discriminative prototype selection methods for graph embedding. *Pattern Recogn.* 46(6), 1648–1657 (2013)
9. Jain, A.K., Zongker, D.: Representation and recognition of handwritten digits using deformable templates. *IEEE Trans. Pattern Anal. Mach. Intell.* 19, 1386–1391 (1997)
10. Spillmann, B., Neuhaus, M., Bunke, H., Pekalska, E., Duin, R.P.W.: Transforming strings to vector spaces using prototype selection. In: Yeung, D.-Y., Kwok, J.T., Fred, A., Roli, F., de Ridder, D. (eds.) *SSPR 2006 and SPR 2006*. LNCS, vol. 4109, pp. 287–296. Springer, Heidelberg (2006)
11. Alimoglu, F., Alpaydin, E.: Methods of combining multiple classifiers based on different representations for pen-based handwritten digit recognition. In: *Fifth Turkish Artificial Intelligence and Artificial Neural Networks Symposium* (1996)
12. Messer, K., Matas, J., Kittler, J., Jonsson, K.: XM2VTSDB: The extended M2VTS database. In: *Second International Conference on Audio and Video-based Biometric Person Authentication*, pp. 72–77 (1999)
13. Frank, A., Asuncion, A.: UCI machine learning repository (2010)