ESANN 2011 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2011, i6doc.com publ., ISBN 978-2-87419-044-5. Available from http://www.i6doc.com/en/livre/?GCOI=28001100817300.

A Similarity Function with Local Feature Weighting for Structured Data

Rubén Suárez, Rocío García-Durán and Fernando Fernández

Universidad Carlos III de Madrid - Computer Science Department
Avenida de la Universidad 30, 28911 Leganés, Madrid, Spain

Abstract. The application of learning approaches as Kernel or Instance Based methods to tree structured data requires the definition of similarity functions able to deal with such data. A new similarity function for nearest prototype classification in relational data that follows a tree structure is defined in this paper. Its main characteristic is its capability to weight the importance of the different data features in different areas of the feature space. This work is built over two previous ideas: a similarity function for Local Feature Weighting (LFW), and a Relational Nearest Prototype Classification algorithm (RNPC).

1 Introduction

The definition of accurate distance metrics or kernels is a key issue that may produce the same algorithm to behave very differently. For instance, in Nearest Prototype (NP) classification [1], Local Feature Weighting (LFW) permits to use a weighted euclidean distance, where the weight vector is different in different regions of the space, generating non-linear borders among the different classes [2]. LFW has shown its utility in different works [3], but only in euclidean spaces. In this work we propose to extend the ideas of LFW [2], and to integrate them into a relational version of a NP algorithm, Relational Nearest Prototype Classification (RNPC) [4]. RNPC uses evolutionary inspired operators to build a classifier in an iterative way. Instead of using the same set of weights in the relational distance, each prototype contributes its own weight vector. In each iteration of the algorithm, both the location and the local weights of each prototype are re-computed. The results in artificial datasets, as well as in a classical dataset in the relational literature, are promising when compared with previous approaches.

This paper is organized as follows. Section 2 defines the similarity function used and how LFW is applied. Section 3 describes LFW-RNPC (Local Feature Weighting for Relational Nearest Prototype Classification), which computes the location and the weight vector of each prototype at the same time. Section 4 reports the empirical evaluation performed and Section 5 describes the main conclusions obtained.

2 A Similarity Function for Local Feature Weighting in Nearest Prototype Classification

A Nearest Prototype Classifier, C, is composed of a set of N relational prototypes $C = \{p_1, \ldots, p_N\}$. Each prototype p_i is an instance of the training set (or a computed instance) which represents a region of the full data set. One instance I_j belongs to a specific region, or to a prototype p_i, when p_i is the closest prototype to I_j. All the instances in the same region are classified with the class of the corresponding prototype. The nearest prototype classification has two challenging problems. On the one hand, defining an accurate similarity function between instances and prototypes [5]. On the other hand, to define the number and location of the prototypes.

In this subsection we describe the distance measure used in LFW-RNPC, which is based on the Relational Instance-Based Learning distance (RIBL [6]), but introduces some changes and incorporates the local weighting to the distance calculation.

In a relational scheme, tuples of different relations are related through the definition of keys or correspondences of values of attributes in two relations, where the values of an attribute in the subordinated relation are restricted to existing values in the higher level relation. In LFW-RNPC, each prototype p has an associated set of weight vectors W_p. Each vector $\vec{w}_{p,h}$ corresponds to a single relation (h) in the relational schema, and contains the associated weight for each attribute in the relation (A_h is the set of attributes of relation h, and H the set of relations in the schema):

$$W_p = \{\vec{w}_{p,h} \in \Re^{|A_h|} | h \in H\} \tag{1}$$

The distance between two relational examples is computed recursively from the root relation, where a single tuple corresponds to a single relational example. The distance value between two tuples (I_1 and I_2) is the sum of the distance value associated to each of its attributes, d_a, considering the weight of the attribute w_a, divided by the number of attributes in the relation, $|A_h|$:

$$d(I_1, I_2) = \frac{\sum_{a \in A_h} w_a * d_a(I_1(a), I_2(a))}{|A_h|} \tag{2}$$

The contribution of each attribute depends on its type. For numeric attributes we follow an absolute distance, and for nominal values we use the Hamming distance. If the attribute is referenced by an attribute from other relation (a foreign key) the recursive step is made in the distance calculation. This step consists in the calculation of the distance between the sets of tuples that hold the same value as the tuples I_1 and I_2 in the foreign key, shown in Equation (3):

$$d_a(v_1, v_2) = \frac{\sum_{h \in H_a} D(s(h, a, v_1), s(h, a, v_2))}{|H_a|} \tag{3}$$

where D is a distance measure between sets, H_a is the set of relations that reference the attribute a, and $s(h, a, v)$ is the set of tuples in the relation h that hold the value v in the attribute that references the attribute a.

The way the sets are compared is the main difference between relational distances. In LFW-RNPC the RIBL [7] distance measure between sets is used. As it can be seen in equation (4), the distance between sets A and B is based on the distance of its elements (a_i and b_i), which are tuples of values obtained by using Equation (2).

$$D(A,B) = \begin{cases} \frac{\sum_{a_i} min_{b_j}(d(a_i,b_j))}{|B|} & \text{if } |A| < |B| \\ \frac{\sum_{b_i} min_{a_j}(d(a_i,b_j))}{|A|} & \text{if } |A| \geq |B| \end{cases} \tag{4}$$

3 Computing the Weights of the Similarity Function: The LFW-RNPC Algorithm

The LFW-RNPC algorithm is summarized in Figure 1. It receives as input the set of training instances. From this set, LFW-RNPC generates a first classifier with only one prototype randomly chosen. It also receives the maximum number of iterations, *it*, and, optionally, a bound to the number of prototypes, *MaxPrototypes*. It returns the classifier (set of prototypes) obtained by iteratively applying evolutionary-based operators: mutation, reproduction, fight, move and die. All those operators are the same as those used in the RNPC algorithm, and an extended description can be found in the literature [8, 4].

```
Function LFW-RNPC(InputSet, $X$; MaxNumberIterations, $it$): Classifier
Initialize $C_1$ =choose-randomly($X$);
Initialize $BestC$=$C_1$;
$i = 1$;

repeat{
    $C_{i+1}$ = mutation($X$, $C_i$);
    $C_{i+1}$ = reproduction($X$, $C_{i+1}$);
    $C_{i+1}$ = fight($X$, $C_{i+1}$);
    $C_{i+1}$ = move($X$, $C_{i+1}$);
    $C_{i+1}$ = die($C_{i+1}$);
    $C_{i+1}$ = weights-updating($C_{i+1}$);
    $C_{i+1}$ = instance-asignation($C_{i+1}$);
    $BestC$ = iteration-evaluation($X$,$C_{i+1}$);
    $i = i + 1$
}until($i > it$);

return $BestC$;
```

Fig. 1: LFW-RNPC Algorithm.

When the operators that modify the populations of prototypes are completed the weights are recomputed (**weights-updating**). The weight updating mechanism is based in the one used in the LFW-NPC algorithm [2].

Each attribute has an associated distortion measure (D^h_a), which represents the average difference between the attribute a in relation h of each relational example (I) and the prototype (p_i) of its region (R_i is the set of examples in region i):

$$D^h_a = \frac{1}{||R_i||} \sum_{I \in R_i} d_a(I, p_i) w_{p_i,a} \tag{5}$$

ESANN 2011 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2011, i6doc.com publ., ISBN 978-2-87419-044-5. Available from http://www.i6doc.com/en/livre/?GCOI=28001100817300.

The local weighting consists in the equiparation of all the distortion values in the same region:

$$\forall h \in H, \forall a \in A_h, D_a^h = 1 \tag{6}$$

To do this a weight related to each attribute is calculated ($w_{pi,a}$):

$$w_{pi,a} = \frac{D_a^h * ||R_i||}{\sum_{I \in R_i} d_a(I, p_i)} \tag{7}$$

As regions change through the algorithm execution, in each iteration weights are updated:

$$\Delta w_{p_i,a} = \frac{||R_i||}{\sum_{I \in R_i} d_a(I, p_i)} - w_{p_i,a} \tag{8}$$

A parameter $z \in [0, 1]$ is added to control the change of the weights:

$$w_{p_i,a}^{t+1} = w_{p_i,a}^{t} + z * \Delta w_{p_i,a}^{t+1} \tag{9}$$

Then, the **instance-assignation** is made. In this step each training instance is inserted in its nearest prototype region (its nearest prototype can be different from the past iteration because of the application of the evolutionary operators and the updating of the weights). To do this, the distance between each example and each prototype is computed, and the nearest prototype is the one that minimizes this measure.

Before finishing the iteration the current classifier is evaluated (**iteration-evaluation**). This evaluation is done by measuring the classification accuracy of the current classifier in the training set. If the resulting accuracy is better than the best classifier accuracy obtained in past iterations, the classifier is recorded as the best one. The algorithm returns the classifier with the best accuracy in the training set.

4 Evaluation

The algorithm evaluation was done in four different domains, three artificial ones, and another one used previously in literature.

The three artificial domains were created to test the efficiency of the local weighting on relational domains. In each domain distinct regions have been defined, so that each region is represented by a determined range of values, so that regions with different ranges in the same attributes can belong to the same class and the local weighting can be exploited. These domains are described briefly:

- Domain 1: a simple domain consisting of a single relation and two attributes. Two regions exist, each of which belongs to a class. This domain contains 200 relational examples.

ESANN 2011 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2011, i6doc.com publ., ISBN 978-2-87419-044-5. Available from http://www.i6doc.com/en/livre/?GCOI=28001100817300.

- Domain 2: a domain with 2 relations and 2 classes. In this domain 8 regions are defined, 4 for each class, and contains 200 relational examples.
- Domain 3: this domain consists of two relations and divides the space into two classes. It contains 800 relational examples.
- Musk: contains 92 relational examples divided into two classes, and using two different relations. Tuples of the root relation define the class and identifier of an example, and the other relation defines the set of tuples that describe it, with 166 attributes.

The experiments consisted of 4 cross-validations with 10 folds for each value of the parameter z. The results obtained from each domain are reported in figure 2, where accuracy obtained both for training and test is shown.

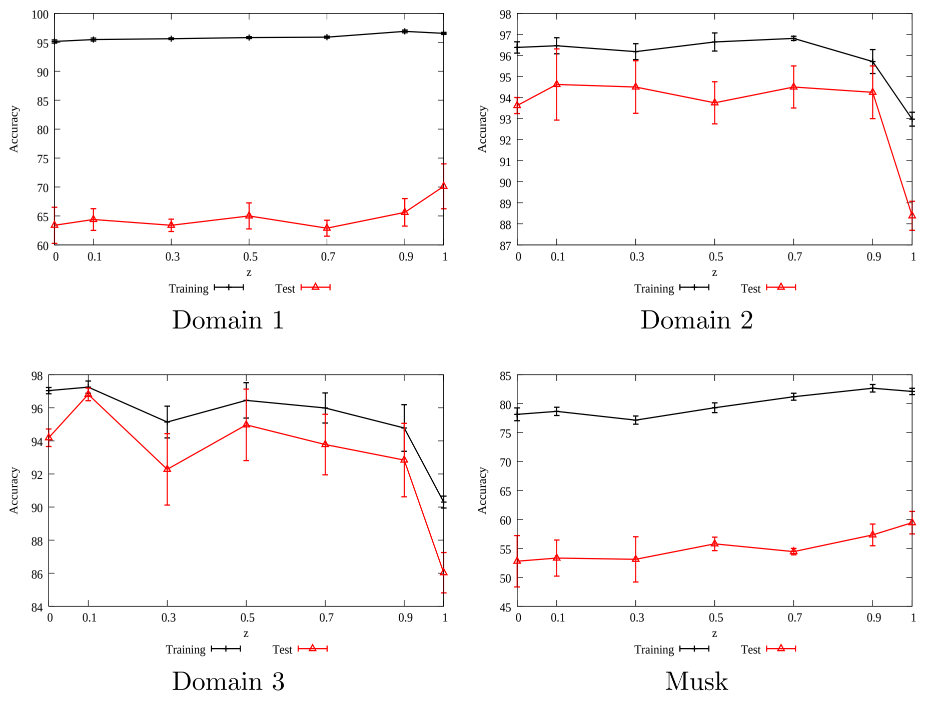

Fig. 2: Experimental results.

The purpose of the evaluation is to see the efficiency of the local weighting, comparing the case of no weighting ($z = 0$), where LFW-RNPC behaves as RNPC and the different values of weight updating.

The results in domains 1, 2 and 3 (artificials) show that using the local weighting improves the results obtained from the case of no updating. In Domain 1, the best results are obtained with $z = 1$. This could happen because the search performed in the space of classifiers and weight configurations with this value of z is completely greedy and can approach the classifier that maximizes the accuracy in this domain. In Domains 2 and 3, however, intermediate values of z generate better results, showing that the results depends on such parameter. In the Musk domain, the local feature weighting is also able to improve the results, obtaining almost a 60 percent of accuracy. Domains 1 and Musk show a strong overfitting to training data (differences among training and test results are around a 30%) which demonstrate that such overfitting should be avoided somehow.

5 Conclusions

In this paper we have described a new nearest prototype classification algorithm for tree structured data which is able to perform local feature weighting. The algorithm has been built using ideas previously developed in vectorial representations, and the empirical evaluation has shown promising results. In future we expect to compare different approaches and perform a wider evaluation of the method in more data sets.

Acknowledgments

This work has been partially supported by Spanish MICINN project number TIN2008-06701-C03-03 and the regional project CCG10-UC3M/TIC-5597.

References

[1] Ludmila I. Kuncheva and James C. Bezdek. Nearest prototype classification: Clustering, genetic algorithms, or random search? *IEEE Transactions on Systems, Man and Cybernetics*, 28(1):160–164, February 1998.

[2] Fernando Fernández and Pedro Isasi. Local feature weighting in nearest prototype classification. *IEEE Transactions on Neural Networks*, 19(1):40–53, 2008.

[3] Jae Heon Park, Kwang Hyuk Im, Chung-Kwan Shin, and Sang Chan Park. Mbnr: Case-based reasoning with local feature weighting by neural network. *Applied Intelligence*, 21:265–276, 2004. 10.1023/B:APIN.0000043559.83167.3d.

[4] Rocío García, Fernando Fenrnández, and Daniel Borrajo. A prototype-based method for classification with time constraints: A case study on automated planning. *Pattern Analysis and Applications*, 2011.

[5] M. Sebag. Distance induction in first order logic. In S. Džeroski and N. Lavrač, editors, *Proceedings of the 7th International Workshop on Inductive Logic Programming*, volume 1297, pages 264–272. Springer-Verlag, 1997.

[6] M. Kirsten, S. Wrobel, and T. Horváth. *Relational Data Mining*, chapter Distance Based Approaches to Relational Learning and Clustering, pages 213–232. Springer, 2001.

[7] W. Emde and D. Wettschereck. Relational instance-based learning. In *Proceedings of the Thirteen International Conference on Machine Learning (ICML'96)*, pages 122–130, 1996.

[8] Rocío García-Durán, Fernando Fernández, and Daniel Borrajo. Nearest prototype classification for relational learning. In *Conference on Inductive Logic Programming*, pages 89–91, Santiago de Compostela, Spain, 2006.