Prototypes Based Relational Learning

Rocío García-Durán, Fernando Fernández, and Daniel Borrajo

Universidad Carlos III de Madrid
29011 Leganes, Madrid, Spain
rgduran@inf.uc3m.es, ffernand@inf.uc3m.es, dborrajo@ia.uc3m.es
http://www.plg.inf.uc3m.es

Abstract. Relational instance-based learning (RIBL) algorithms offer high prediction capabilities. However, they do not scale up well, specially in domains where there is a time bound for classification. Nearest prototype approaches can alleviate this problem, by summarizing the data set in a reduced set of prototypes. In this paper we present an algorithm to build Relational Nearest Prototype Classifiers (RNPC). When compared with RIBL approaches, the algorithm is able to dramatically reduce the number of instances by selecting the most relevant prototypes, maintaining similar accuracy. The number of prototypes is obtained automatically by the algorithm, although it can be also bounded by the user. Empirical results on benchmark data sets demonstrate the utility of this approach compared to other instance based approaches.

Keywords: Inductive Logic Programming, Instance Based learning, Nearest Prototype.

1 Introduction

The main activity in relational data mining consists on finding patterns in data, described by multiple relations. This difference on the representation of data with respect to the propositional case increases the complexity of finding these patterns. The different relational techniques generate models using diverse representation schemes, such as equations, classification/regression trees/rules, association rules, or instance-based. In the last type of representation, we need to define a relational distance between two data entries (similarity measure). Usually, this distance measure in the relational case must be adapted and extended from the ones used for propositional learning. This technique to learn patterns is also named Instance-Based Learning (IBL).

IBL methods rank among the best options for classification tasks when dealing both with propositional and relational representations. Relational Instance-Based Learning (RIBL) [1] techniques have been able to obtain good classifiers in challenging domains, such as those composed of biological data.

The accuracy of IBL methods is comparable (sometimes even better than) to other relational techniques like inductive classification logic (ICL), logical decision trees (TILDE), relational regression trees (S-CART), relational rules

D. Dochev, M. Pistore, and P. Traverso (Eds.): AIMSA 2008, LNAI 5253, pp. 130–143, 2008.

induction (CPROGOL4.4) or relational association rules (WARMR) [2]. However, IBL have problems scaling up when the training set grows.

In most domains in the machine learning literature, there is no restriction on the time to perform classification. Furthermore, given that for IBL techniques classification time is usually proportional to the size of the classifier, the number of instances in the learned "model" does not represent a hard constrain. But, there are application domains where classification time is an important constrain, as it the case of applying relational learning to AI planning tasks. For instance, if we want to learn a model that would predict the heuristic value of every node of a search process, it should return a prediction (heuristic value) quite fast, so that the use of the learned heuristic outperforms using other heuristics, or even performing an extended blind search.

In this paper we introduce the first implementation of a Relational Nearest Prototype Classifier (RNPC). The goal is to obtain a reduced set of prototypes that generalizes the data set such that it can predict the class of a new instance faster than using RIBL and with an equivalent accuracy. Specifically, this solution is based on an existing nearest protoype algorithm for propositional data, the Evolutionary Nearest Prototype Classifier (ENPC) [3]. There are two main differences with that work: RNPC uses a relational representation; and the prototypes are extracted by selection as in [4]. The algorithm is based on an evolutionary process, where different operators eliminate or select new protoypes from the original data set. RNPC has three main properties: (i) an accuracy performance similar to other IBL approaches; (ii) the automated selection of the number of prototypes, that can be bounded by the user, and (iii) a reduction in time and memory in future classifications required in some domains.

In the next section we explain the RNPC algorithm. The third section shows the experiments performed over four different data sets, and comparative results with IBL methods and other classical relational algorithms. Finally, we draw some conclusions and suggest future work.

2 The RNPC Technique

As the rest of RIBL techniques, the RNPC algorithm uses a relational representation of the data (we use the MTARFF format [5]) and a relational distance measure to compute the similarity between instances. The RNPC algorithm is independent of the distance measure, and we have experimented with different relational distances. We provide first a set of definitions, and later describe the algorithm.

2.1 Definitions

A classifier, C, is composed of a set of N relational prototypes $C = \{p_1, \ldots, p_N\}$. Each prototype p_i is a selected instance of the training set and 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.

Table 1. RNPC Algorithm

Input: TrainingSet, $\boldsymbol{X}$; MaxNumberIterations, it
Initialize C_0 =choose-randomly($\boldsymbol{X}$), $i = -1$

REPEAT
  $i = i + 1$
  C_{i+1} = mutation($\boldsymbol{X}$, C_i)
  C_{i+1} = reproduction($\boldsymbol{X}$, C_{i+1})
  C_{i+1} = move($\boldsymbol{X}$, C_{i+1})
  C_{i+1} = fight($\boldsymbol{X}$, C_{i+1})
  C_{i+1} = die($\boldsymbol{X}$, C_{i+1})
UNTIL($C_i = C_{i+1}$) OR ($i = it$)

return C_{i+1}

Each prototype, p_i, is characterized by its quality, $quality(p_i)$, which is computed by equation 1.

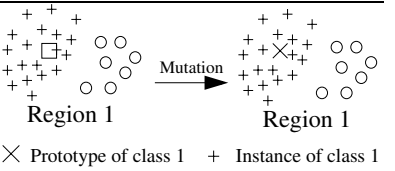
$$quality(p_i) = min(1, accuracy(p_i) \times coverage(p_i)) \tag{1}$$

The quality of a prototype is high if it classifies correctly, *accuracy*, and if it represents a sufficient number of instances, *coverage*. The value of the quality is limited to the range $[0, 1]$.

2.2 The Algorithm

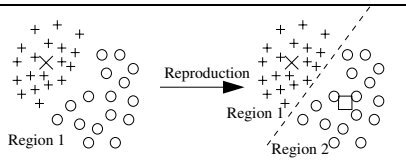
The RNPC algorithm is summarized in Algorithm 1. It receives as input a classifier of only one prototype randomly chosen from the set of instances. It also receives the maximum number of iterations, it, and, optionally (in case the user supplies this value), a bound on the number of prototypes, $MaxPrototypes$. It returns the last classifier (set of prototypes) obtained by iteratively applying some genetic-based operators described next. The loop stops when the maximum number of iterations has been completed.

Table 2. Algorithm of the Mutation operator



Function mutation:
for each $p_j \in C_i$
  $class(p_j) = mode(p_j, instances(p_j))$
return C_i

Table 3. Algorithm of the Reproduction operator



Function reproduction
for each $p_j \in C_i$
  $k = random(0, 1)$
  if $(k < P_{Repr}(p_j))$ *then*
    $class = roulette(instances(p_j))$
    if $(class <> class(p_j))$ *then*
      $p_{j'} = select(instances(p_j), class)$
      $C_i = \{p_{j'}\} \cup C_i$
return C_i
  where p_j is randomly selected from the new region

The **mutation** operator (Table 2) labels each prototype in the current classifier C_i with the most popular class in each region. Following the nearest neighbour rule, each prototype knows the number of instances of each class located in its region. Then, the prototype class becomes the same class as the most abundant class of instances in its region (the mode).

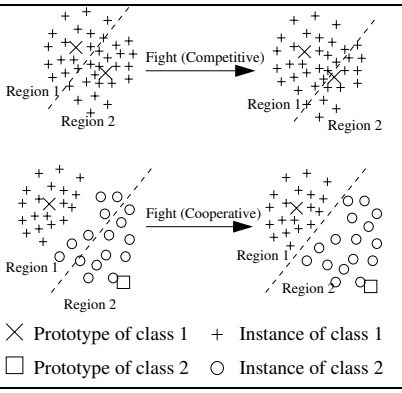
The **reproduction** operator (Table 3) introduces new prototypes into the current classifier C_i. The insertion of new prototypes is a decision that is taken by each prototype in order to increase its own quality. Thus, regions with instances belonging to different classes can create new regions of different classes. Equation 2 defines the probability of reproduction of a propotype, where p_j is a prototype, N is the current number of prototypes. In case the user supplies it, $MaxPrototypes$ is the maximum number of prototypes. So, the probability is inversely proportional to the accuracy (first factor): if the classification success of the prototype is high there is no need to reproduce. Also, when the current number of prototypes tends to $MaxPrototypes$, the probability tends to zero (second factor). By default $MaxPrototypes$ is the number of training instances.

$$P_{Repr}(p_j) = (1 - accuracy(p_j)) \times (1 - \frac{N}{MaxPrototypes}) \tag{2}$$

A roulette method decides the class of the new prototype. The roulette contains a slide per class with size proportional to the number of instances of each class in the region. If the selected class, j, is different from $class(p_i)$, the new region will have all instances belonging to class j and the new prototype will be chosen randomly among the instances of that class.

The **fight** operator (Table 4) allows the prototypes to capture instances from other regions. Each prototype chooses one rival prototype among its neighbours, adjacent regions, by a roulette mechanism. The probability of choosing one rival

Table 4. Algorithm of the Fight operator



Function fight
for each $p_j \in C_i$
  $p_r = roulette(neighbors(C_i, p_j))$
  $k = random(0, 1)$
  if $(k < P_{Fight}(p_j, p_r))$ *then*
    if $(class(p_j) = class(p_r))$ *then*
      $p_w, p_l = roulette(\{p_j, p_r\})$
      $stealGauss(p_w, instances(p_l), class(p_w))$
    else
      $stealClass(p_j, instances(p_r), class(p_j))$
return C_i
  where p_w and p_l are the winner and loser prototypes

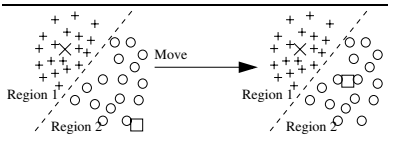
is proportional to the difference among the qualities of both prototypes. In a second step, the prototype decides whether to fight or not. This probability is also proportional to the difference of qualities between both rivals.

Finally, if they fight and both have the same class, the winner will be selected with a roulette mechanism using the quality measure. The winner prototype steals as many instances of its class as a Gaussian function dictates. However, if both rivals belong to different classes, they cooperate and the rival gives to the initial prototype all its instances belonging to the initial prototype class.

The **move** operator (Table 5) relocates the prototypes. It finds the medoid of a region (the relational instance with less distance to the rest) as was successfully applied in FORC [6]. This is the main difference with respect to ENPC, where the centroid (a new propositional instance placed in the middle of the region) was computed.

Finally, the **Die** operator (Table 6) eliminates weak prototypes. The probability of dying is proportional to the quality of the prototype and inversely proportional to the difference between the number of current prototypes and $MaxPrototypes$. Then, a prototype have more probability of dying if its quality is close to 0, and also if the current number of prototypes of the classifier is close to or greater than the expected one. The instances of a dead prototype are

Table 5. Algorithm of the Move operator



Function move
for each $p_j \in C_i$
$p_j = medoid(instances(p_j))$
return C_i

Table 6. Algorithm of the Die operator

Function die
for each $p_j \in C_i$
$k = random(0, 1)$
$if\ (k < P_{Die}(p_j))\ then C_i = C_i - \{p_j\}$
return C_i

moved to the nearest prototype. The resulting probability is given by equation 3, with the same notation as in equation 2.

$$P_{Die}(p_j) = 1 - min(1, \frac{MaxPrototypes}{N} \times quality(p_j)) \quad (3)$$

The prototypes change in each iteration, so a different classifier is generated at the end of each iteration. The algorithm collects only the next classifier with best training accuracy. If the training accuracy is similar, it selects the one with less number of prototypes.

3 Experiments

In this section we study the RNPC algorithm for three data sets:[1]

1. **Protein Fingerprints:** A protein fingerprint consists of a list of motifs where one motif is a set of sequences of amino-acids [7]. The total number of instances is 1842 and there are three possible diagnostics [8].
2. **Diterpenes:** Diterpenes are organic compounds of low molecular weight that are based on a skeleton of 20 carbon atoms. The goal is to identify the type of diterpenoid compound skeletons given their 13C-NMR spectra. There are 1503 instances and 23 different classes [9].

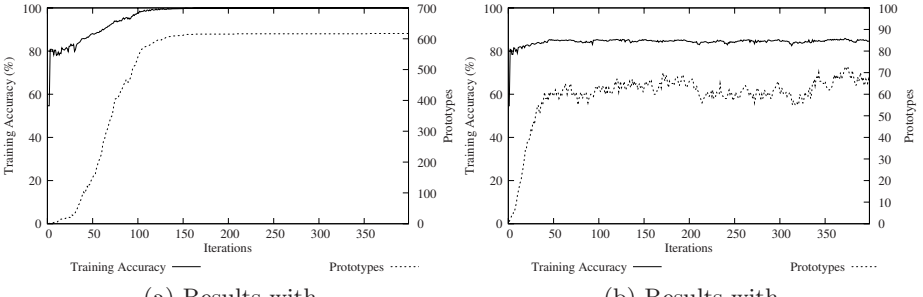
[1] A description of the data sets can be found in:
http : *//cui.unige.ch/* ∼ *woznica/rel_weka/*

3. **Mutagenesis:** The goal is to predict mutagenicity of a set of 188 aromatic and hetero-aromatic nitro-compounds. It has been described in two ways: in the first one, *Mutagenesis1*, the compounds are composed of atoms, which constitute bonds. In this case, the median of the number of atoms for each compound is 26 and the recursion is four. In the second one, *Mutagenesis2*, the compounds are composed of bonds, which consist of atoms. Now, the median of the number of bonds is 28 and the recursion level is limited to three.

3.1 Training in RNPC

First, we sample how RNPC behaves in training, showing the results of two executions with the Protein Fingerprints data set (Figure 1). They correspond to $MaxPrototypes = \#TrainingInstances$ and $MaxPrototypes = 92$ (5% of the training set) respectively. We refer as the *reduction factor* the percentage of reduction in the number of instances between the original data set and the one that RNPC returns. In both figures we can observe the accuracy over the training set (left Y axis) and the number of prototypes of the classifiers in each iteration (right Y axis).

In Figure 1(a) training accuracy rises to 100% in iteration 356, what means that there is overfitting in training. The algorithm selects the classifier with best training accuracy and, between these, the one with a lower number of prototypes: classifier resulting in iteration 356 with 617 prototypes. Then, the selected classifier is tested with the last fold of instances and its test accuracy decrease to 72.04%. However, in Figure 1 (b) where $MaxPrototypes = 92$, the best classifier is selected in iteration 372 with a training accuracy of 85.87% and 72 prototypes. This classifier is tested with the last fold of instances and obtains 84.41% test accuracy. We can see in this case how using a higher reduction in the number of prototypes, the test accuracy increases. Although we will show in the next section that this issue depends on the data set.



(a) Results with $MaxPrototypes = \#TrainingInstances$

(b) Results with $MaxPrototypes = 92$

Fig. 1. Training accuracy and number of prototypes in Fingerprints

3.2 Results in Biological Data Sets

The following experiment used the four data sets and the results are summarized from Table 7 to 10 respectively. The average number of prototypes are shown in Tables (a) and the accuracy results in Tables (b). We executed a 10-fold cross validation in each domain. For each fold, the experiments have been repeated 10 times for RNPC, since it is stochastic. In all cases, the algorithm was executed for a maximum of 400 iterations ($it = 400$). In all domains, the algorithm has been executed with different distance measures: the RIBL [6], AL (Average Linkage), CL (Maximum), Hausdorff (metric), SL (Minimum), SoMDM (Average sum of minimal), SoMDMNorm (Normalized SoMDM), and SymmetricDiff (Symmetric Difference) or Tanimoto (metric) distances [10,11,12].

In all cases we have run the RNPC algorithm with no reduction and with a 95% reduction ($\text{RNPC}_{95\%}$). To compare the generalization capability of the algorithm, we have also implemented the relational version of the IBL (RIBL) algorithm that processes instances incrementally [13]. It takes instance by instance selecting them as prototypes only if no prototype before covers it. The number of final prototypes tends to be similar to the number of prototypes in the RNPC algorithm. The RIBL

Table 7. Results in the Fingerprints data set, 1842 instances, 3 classes

(a) Average number of prototypes.

	RNPC	$\text{RNPC}_{95\%}$	RIBL
RIBL	620.38	69.09	507.1
AL	580.90	70.75	522.8
CL	412.98	64.35	581.7
Hausdorff	603.78	78.9	491.8
SL	649.45	76.37	514.9
SoMDM	642.64	76.94	513.6
SoMDMNo	2.85	2.97	955.4
SymDiff	705.97	71.62	556.2
Tanimoto	691.74	73.28	538.6

(b) Accuracy results.

	RNPC	$\text{RNPC}_{95\%}$	RIBL	IBk1	IBk3	IBk8
RIBL	73.87 ± 3	82.79 ± 2.1	69.71 ± 2.3	78.99 ± 3.7	83.66 ± 4	84.31 ± 3.4
AL	73.38 ± 9.4	78.70 ± 10.1	68.22 ± 10.5	79.04 ± 2.7	75.62 ± 3.3	85.02 ± 2.6
CL	73.59 ± 9.2	77.77 ± 10.1	65.12 ± 8.6	75.62 ± 3.3	81.11 ± 2.2	83.17 ± 2.4
Hausdorff	74.21 ± 2.8	82.07 ± 2.9	70.03 ± 2.4	79.48 ± 2.4	82.63 ± 2.5	84.04 ± 2.1
SL	71.66 ± 10.2	80.00 ± 10.2	67.94 ± 8.1	80.29 ± 2	84.09 ± 2.2	86.00 ± 2.5
SoMDM	72.55 ± 9.2	79.90 ± 10.6	67.62 ± 9.3	80.18 ± 2.4	83.82 ± 3.2	85.83 ± 2.4
SoMDMNo	54.17 ± 4.1	54.22 ± 4.1	43.49 ± 6.4	45.34 ± 6.5	43.87 ± 5.7	43.92 ± 5.9
SymDiff	70.97 ± 10.9	76.27 ± 9.5	68.49 ± 9.9	76.11 ± 1.8	76.87 ± 2.3	79.26 ± 1.6
Tanimoto	72.36 ± 9.4	77.32 ± 10.2	70.92 ± 7.6	78.61 ± 1.6	79.97 ± 2	82.30 ± 2.8

algorithm has two disadvantages: the resulting classifier depends on the order of the data; and it is not possible to reduce the number of final prototypes without deleting instances of the initial data set. For the RNPC and IBL algorithms we also show the average number of prototypes of the final classifiers.

Finally we compare against Relational IBk where k is 1, 3 and 8. This algorithm predicts the class of a new example, by selecting the majority class of the k nearest neighbours. The disadvantages in this algorithm are two: the need of trying with different values of k; and the test time cannot be reduced, because it must compute the distance with the complete data set.

Comparing distance metrics. First, we observe that not all distances are good for all the data sets. One example is the SoMDMNorm distance; its accuracies with all techniques is bad in the Fingerprints data set, while it behaves relatively well in the rest of data sets. In fact, it is the best in Diterpenes. However, the three algorithms behave similarly with the same distance.

Comparing each data set. Analyzing the results in the Fingerprints data set in Table 7 (b), we conclude that applying a reduction of 95% improves the

Table 8. Results of test accuracy in the Diterpenes data set, 1503 instances, 23 classes

(a) Average number of prototypes.

	RNPC	$\text{RNPC}_{95\%}$	**RIBL**
RIBL	311.52	64.77	327.5
AL	7.94	8.35	859.7
CL	9.38	11.02	1086.8
Hausdorff	478.28	54.18	442.9
SL	16.13	17.76	1012.7
SoMDM	238.81	67.27	256.8
SoMDMNo	213.41	69.17	227.9
SymDiff	3.75	4.17	996.8
Tanimoto	454.18	51.65	459.7

(b) Accuracy results.

	RNPC	$\text{RNPC}_{95\%}$	RIBL	IBk1	IBk3	IBk8
RIBL	86.03 ± 2.7	70.77 ± 4.2	87.87 ± 2.2	95.14 ± 2.9	92.68 ± 2	88.82 ± 2.5
AL	50.03 ± 11.5	49.47 ± 12	40.13 ± 6.9	39.39 ± 7.7	25.35 ± 3.7	26.01 ± 2.1
CL	30.45 ± 3.9	30.51 ± 4.2	19.87 ± 4.4	19.89 ± 4	19.23 ± 3.8	21.63 ± 4
Hausdorff	82.24 ± 3.1	61.38 ± 4.8	79.87 ± 3.2	84.90 ± 2.4	76.78 ± 2.3	68.12 ± 2.8
SL	36.48 ± 3.6	36.33 ± 3.6	24.53 ± 3.4	51.77 ± 2.9	51.70 ± 2.7	51.57 ± 2.8
SoMDM	91.32 ± 1.7	80.91 ± 2.8	91.2 ± 2.1	95.14 ± 1.5	92.22 ± 1.5	87.36 ± 2
SoMDMNo	92.93 ± 1.8	83.75 ± 3.8	91.73 ± 2.3	96.67 ± 1.1	94.15 ± 1.2	90.29 ± 1.8
SymDiff	35.21 ± 2.6	35.04 ± 2.4	26.4 ± 6.3	33.60 ± 2.2	33.60 ± 2.2	33.60 ± 2.2
Tanimoto	77.24 ± 4.3	57.02 ± 4.6	77.8 ± 2.3	88.02 ± 2.4	79.97 ± 2	80.11 ± 2.4

test accuracy in all cases with respect to not applying the reduction. These results are also better than the ones obtained in the relational IBL algorithm for all distances. However, the best classifier with respect to accuracy with all distances is IBk (k =8). The smallest difference between the test accuracies of the best two results for both algorithms is 1.5 points for the RIBL distance. The biggest difference is 6.5 points with the AL distance.

In the Diterpenes data set (Table 8), using the RNPC algorithm without reduction is better in most cases than with reduction (between 10 and 20 points). The explanation could be the high number of different classes in this data set (23). For the best distance, SoMDMNorm, it looks easier to predict instances of 23 different classes with 213 prototypes than with 69. The 95% reduction is too high for this domain. In fact, with a 90% reduction over the total set (or 88.9% over the training set) and with the best distance, we obtain a 87.58% in test accuracy. Globally, the IBk algorithm (k =1) is the best one, although its results are close to RNPC and relational IBL.

In both versions of Mutagenesis (Tables 9 and 10), the total number of instances is 188, which explains the high standard deviations. The best results in Mutagenesis1 are shared by RNPC algorithm in both versions and the IBk1. In

Table 9. Results of test accuracy in the Mutagenesis1 data set, 188 instances, 2 classes

(a) Average number of prototypes.

	RNPC	$\text{RNPC}_{95\%}$	**RIBL**
RIBL	70.41	7.39	57.9
AL	5.18	3.96	67.5
CL	8.77	5.07	67.3
Hausdorff	62.73	8.99	55.1
SL	25.92	8.51	65.4
SoMDM	50.32	9.4	43.5
SoMDMNo	47.71	9.61	46.9
SymDiff	4.58	3.18	66.2
Tanimoto	56.04	9.87	54.5

(b) Accuracy results.

	RNPC	$\text{RNPC}_{95\%}$	RIBL	IBk1	IBk3	IBk8
RIBL	73.16 ± 7.4	75.73 ± 8.4	65.82 ± 12.2	72.31 ± 9.4	67.95 ± 11.6	71.23 ± 7.3
AL	64.60 ± 9.4	67.21 ± 8.5	64.95 ± 9.7	55.29 ± 12.7	49.62 ± 14.5	68.10 ± 9.9
CL	63.31 ± 10.3	65.26 ± 10.6	63.31 ± 11.1	49.62 ± 14.5	65.41 ± 9.2	66.52 ± 6.4
Hausdorff	79.31 ± 11.5	70.67 ± 12.4	70.4 ± 14.1	76.52 ± 12	72.31 ± 11.6	76.61 ± 8.2
SL	75.30 ± 7.1	77.63 ± 7.6	60.62 ± 9	74.47 ± 9.2	76.08 ± 8.7	76.08 ± 8.3
SoMDM	81.76 ± 9.7	78.00 ± 10.5	80.93 ± 10.8	83.45 ± 9.4	78.22 ± 11.1	76.67 ± 9.9
SoMDMNo	83.85 ± 8.5	72.26 ± 10.5	81.33 ± 10.4	85.00 ± 9.8	82.40 ± 11.1	79.74 ± 9.4
SymDiff	64.93 ± 12.1	63.99 ± 11.2	62.26 ± 8.2	75.06 ± 9.9	75.06 ± 9.9	70.76 ± 7.1
Tanimoto	80.16 ± 7.3	71.89 ± 10.6	77.18 ± 8.5	78.25 ± 8.8	77.60 ± 9.9	69.15 ± 6.9

Table 10. Results of test accuracy in the Mutagenesis2 data set, 188 instances, 2 classes

(a) Average number of prototypes.

	RNPC	$\text{RNPC}_{95\%}$	**RIBL**
RIBL	53.99	8.24	75.5
AL	4.23	3.54	69.3
CL	3.93	3.48	72.5
Hausdorff	67.03	9.28	60.3
SL	25.44	9.03	65
SoMDM	44.75	9.61	47.6
SoMDMNo	25.13	5.94	42.9
SymDiff	3.36	2.56	63.1
Tanimoto	51.31	9.8	48.4

(b) Accuracy results.

	RNPC	$\text{RNPC}_{95\%}$	RIBL	IBk1	IBk3	IBk8
RIBL	76.21 ± 9.3	72.53 ± 10.3	59.91 ± 12.6	75.41 ± 11.1	67.98 ± 8.7	72.22 ± 10
AL	65.82 ± 12.5	66.13 ± 11.7	60.22 ± 11.9	55.29 ± 12.1	40.44 ± 10.2	74.39 ± 6.4
CL	57.29 ± 17.6	58.87 ± 16.3	50.87 ± 15.3	40.44 ± 10.2	62.81 ± 6.3	54.88 ± 15.5
Hausdorff	75.51 ± 11.6	63.06 ± 13.8	71.98 ± 12.2	76.55 ± 9.5	71.26 ± 11.8	66.02 ± 10.4
SL	77.16 ± 8.6	76.67 ± 8.6	57.59 ± 17.1	75.03 ± 9.6	76.08 ± 8.7	76.08 ± 9.3
SoMDM	85.03 ± 7.1	77.27 ± 8.7	78.76 ± 6.9	84.01 ± 8.8	78.27 ± 11.4	69.65 ± 8.5
SoMDMNo	71.75 ± 22.8	63.98 ± 21.6	81.46 ± 8.9	87.72 ± 8.7	81.84 ± 11.3	80.85 ± 11.2
SymDiff	66.54 ± 11	66.43 ± 11.3	60.09 ± 11.4	73.42 ± 10.7	73.42 ± 10.4	69.65 ± 8.5
Tanimoto	79.28 ± 7.8	76.46 ± 10.5	75.6 ± 6.4	82.46 ± 8.3	80.29 ± 9.4	73.45 ± 6.4

Mutagenesis2 all results are better using no reduction. Globally both algorithms, RNPC and IBk, obtain the best results.

In general terms, as almost always happens, the best algorithm depends on the data set. In the Fingerprints and Diterpenes data sets the IBk algorithm works better with k =8 and k =1. In Mutagenesis1 the best results are given by RNPC and in Mutagenesis2 the best results are good for both algortihms. RIBL is not so good as the others.

If we analyze the best results in the RNPC and IBk algorithms independently of the used distance and we compute the statistic significance tests, we can say that in Fingerprints and Diterpenes data sets, the differences are statistically significant with α =0.05. That is, IBk algorithm is the best one in Fingerprints and Diterpenes data sets. For both versions of Mutagenesis, it does not exist significant difference between both algorithms.

Comparing number of prototypes. In this subsection we study the reduction in the number of prototypes generated by RNPC and relational IBL with respect to the size of the training set. In general, and for all distances with

Table 11. Test accuracy results in Fingerprints data set using 10-fold cross validation

Algorithm	Test accuracy (%)
SVM-RBF algorithm	85.92
IBk 8 + RIBL distance	81.69
IBk 1 + RIBL distance	75.21
RNPC + RIBL distance	71.5 ± 1.41

Table 12. Test accuracy results in Mutagenesis1 data set using 10-fold cross validation

Algorithm	Test ac.(%)
CLAUDIEN [2]	90
RNPC + SoMDMNorm	82.1
ICL [2]	80.9
PROGOL [14]	79
TILDE [15]	75
MRDTL [16]	67
FOIL [15]	61

the best results, the number of protoypes is similar for the RNPC and IBL algorithms. However, observing the number of prototypes and the corresponding test accuracy, these two values are not directly proportional. This is the case of the Fingerprints domain (Table 7) where the test accuracy is better for all distances in the execution of $\text{RNPC}_{95\%}$, but the number of prototypes is much smaller than the resulting from the standard RNPC and the RIBL algorithms. That is, we can obtain better accuracy with a lower number of prototypes.

Comparing with other methods. In addition, we compared: i) the obtained results by RNPC; with ii) the ones reported in [7] using IBk (k =1 and k =8) and RIBL distance; and iii) the best result in [8] using the SVM-RBF algorithm. All these results refer only to the Fingerprints data set, where the instances belonging to the training and test sets of the 10-fold cross validation are the first 1487 instances used in the experiments explained before, and the final test was performed with the last 355 instances. The results are shown in Figure 11, where the test result of RNPC is 3.71 points worse than the results of IBk for k =1, 10.19 points worse than the IBk algorithm for k =8 with the RIBL distance, and 14.42 points worse than the SVM-RBF algorithm. The results of RNPC and IBk follow the same behavior as in Table 7 (b) for the RIBL distance.

More experiments with the Mutagenesis1 data set and other algorithms can be seen in Figure 12. We can see how RNPC with the SoMDMNorm distance (the best one selected in Table 9 (b)) improves the results of ICL, PROGOL, TILDE, MRDTL and FOIL. However, it does not improve CLAUDIEN.

4 Conclusions and Future Work

In this paper, we have presented RNPC as a first approach for nearest prototype classification of relational data. It is an efficient distance-based method that can use any relational distance, reducing dramatically the size of the database to compare new instances with. Furthermore, the number of prototypes can be either automatically generated by the algorithm, or bounded by the user. The reduction in the size of the data set permits to classify new instances faster than with the original data set.

RNPC obtains a kind of generalization of the training instances by generating prototypes. The algorithm has, as a secondary effect, the automatic clustering of the set of training instances such that the instances belonging to the same cluster (or prototype) are similar to each other. Thus, we can use the final prototypes as a generalized description of the clusters. This opens the possibility of human understanding of the resulting classifier and of using those reduced sets of prototypes.

We would like to extend this work in three ways. First, to decide the best distance metric for each domain automatically. Second, to improve the distance metrics used by RNPC, introducing feature selection or weighting methods following previous attribute-value approaches. Third, to extend the application to learning in planning, where the learning protoypes can be seen as policies to guide the search of a solution. This learning can substitute the planner or even it can be combinated with the base planner to act as a heuristic.

Acknowledgments

This work has been partially supported by the Spanish MEC project TIN2005-08945-C06-05, a grant from the Spanish MEC, and regional CAM-UC3M project CCG06-UC3M/TIC-0831.

References

1. Emde, W., Wettschereck, D.: Relational instance-based learning. In: Proceedings of the Thirteen International Conference on Machine Learning, pp. 122–130. Morgan Kaufmann, San Francisco (1996)
2. Dzeroski, S., Lavrac, N.: Relational Data Mining. Springer, Heidelberg (2001)
3. Fernández, F., Isasi, P.: Evolutionary design of nearest prototype classifiers. Journal of Heuristics 10(4), 431–454 (2004)
4. Kuncheva, L., Bezdek, J.: Nearest prototype classfication: Clustering, genetic algorithms, or random search? IEEE Transactions on Systems, Man, and Cybernetics (1998)
5. Witten, I., Frank, E.: Data Mining. Practical Machine Learning Tools and Techniques. Elsevier and Morgan Kaufmann (2005)
6. Kirsten, M., Wrobel, S., Horváth, T.: Distance Based Approaches to Relational Learning and Clustering. In: Relational Data Mining, pp. 213–232. Springer, Heidelberg (2001)

7. Woznica, A., Kalousis, A., Hilario, M.: Distance-based learning over extended relational algebra structures. In: Proceedings of the 15th International Conference of Inductive Logic Programming (2005)
8. Hilario, M., Kim, J., Bradley, P., Attwood, T.: Classifying protein fingerprints. In: Proceedings of the 8th Conference on Principles and Practice of Knowledge Discovery in Databases (2004)
9. Dzeroski, S., Schulze-Kreme, S., Heidtke, K., Siems, K., Wettschereck, D.: Intelligent Data Analysis in Medicine and Pharmacology. In: Diterpenes structure elucidation from 13C NMR spectra with machine learning, pp. 207–225 (1997)
10. Kalousis, A., Hilario, M.: Representational issues in meta-learning. In: Proceedings of the 20th International Conference on Machine Learning (2003)
11. Ramón, J., Bruynooghe, M.: A polinomial time computable metric between point sets. In: Acta Informática (2001)
12. Duda, P.H., Stork, D.: Nonparametric Techniques. In: Pattern Classification, 2nd edn. John Wiley & Sons, Chichester (2001)
13. Aha, D.W., Kibler, D., Albert, M.K.: Instance-based learning algorithms. Machine Learning 6, 37–66 (1991)
14. Srinivasan, A., King, R., Muggleton, S.: The role of background knowledge: using a problem from chemistry to examine the performance of an ILP program, Under review for Intelligent Data Analysis in Medicine and Pharmacology (1996)
15. Bolckeel, H.: Top-down induction of first order logical decision trees. PhD thesis, Departament of Computer Science, Katholieke Universiteit Leuven (1998)
16. Leiva, H., Atramentov, A., Honavar, V.: Experiments with MRDTL – a multirelational decision tree learning algorithm. In: Proceedings of the Workshop on Multi-Relational Decision Tree Learning (2002)