Prototypes Generation from Multi-label Datasets Based on Granular Computing

Marilyn Bello[1,2](✉), Gonzalo Nápoles[2], Koen Vanhoof[2], and Rafael Bello[1]

[1] Computer Science Department, Universidad Central de Las Villas, Santa Clara, Cuba
mbgarcia@uclv.cu

[2] Faculty of Business Economics, Hasselt University, Hasselt, Belgium

Abstract. Data reduction techniques play a key role in instance-based classification to lower the amount of data to be processed. Prototype generation aims to obtain a reduced training set in order to obtain accurate results with less effort. This translates into a significant reduction in both algorithms' spatial and temporal burden. This issue is particularly relevant in multi-label classification, which is a generalization of multiclass classification that allows objects to belong to several classes simultaneously. Although this field is quite active in terms of learning algorithms, there is a lack of prototype generation methods. In this research, we propose three prototype generation methods from multi-label datasets based on Granular Computing. The experimental results show that these methods reduce the number of examples into a set of prototypes without affecting the overall performance.

Keywords: Multi-label classification · Prototype generation · Granular Computing · Rough Set Theory

1 Introduction

Classification is one of the most popular Data Mining topics. Its aim is to learn from labeled patterns a model able to predict the decision class for future, never seen before, data samples [1]. The best way to solve a classification problem is usually to have as much information as possible. In practice, however, this is not always the case. The performance of learning algorithms may decrease due to the abundance of information, because many examples may be very irrelevant to the resolution of the problem or may provide the same information [14,17].

On the other hand, the abundance of information could increase the computational complexity of the method, particularly in the case of instance-based learners such as the kNN (k Nearest Neighbors) [10] algorithm. However, it is possible to reduce or modify the datasets without affecting the learning process, improving process performance by reducing computational cost.

One approach to doing this is the classification based on the Nearest Prototype (NP) [6,12]. It is an approach in which the decision class of a new object is calculated by analyzing its proximity to a set of prototypes selected or generated from the initial set of objects. Strategies are needed to reduce the number of examples of input data into a set of representative prototypes. It is possible to say that the data reduction methods with respect to the instances are divided into two categories: *selection of prototypes* [11] and *generation of prototypes* [25]. Prototype selection algorithms select a set of representative objects according to a well-defined criterion, while prototype generation algorithms are capable of generating a set of new objects in the application domain from the initial objects.

On the other, Multi-Label Classification (MLC) is a type of classification where each of the objects in the data has associated a vector of outputs, instead of being associated with a single value [26,31]. ML-kNN is the first learning method that uses the kNN rule in multi-label prediction [30]. This method finds the k nearest neighbours in the datasets using the maximum a posteriori principle in order to determine the label set of the test object. The solution is based on the prior and posterior probabilities of the frequency of each label within the k nearest neighbours. Consequently, this method has the same drawbacks of kNN, because as the datasets increases, so does the computational cost of the algorithm, since for each test object its distance to all existing objects in the training set is calculated.

Despite extensive work on multi-label learning, as far as we know, only in [7] a method of prototypes selection is proposed. In this paper, we develop three methods of prototype generation from MLC datasets. Unlike the method proposed in [7], the three methods proposed are independent of the learning algorithm to be used. By doing so, we rely on Granular Computing [3,4,21], and two different ways of the granularity of the information. Two classical granulations are condition granulation and decision granulation, to name the granularity of the universe according to conditional attributes and decision class, respectively.

In the case of the first two methods proposed, the granulation of a universe is performed using a similarity relation that builds similarity classes (or granules) of objects in the universe from conditional attributes. By using similarity relations, methods can be used in the presence of mixed data, i.e., when there are both numerical and nominal attributes. On the other hand, the third method performs a granulation of the universe from an equivalence relation, and taking into account the different labels existing in the universe of discourse. From this, an equivalence class (or granule) is built for each label, and a prototype is generated for each granule.

The paper is organized as follows. Section 2 motivates our research, while Sect. 3 presents the theoretical background on Granular Computing. Section 4 introduces the three prototype generation methods from MLC datasets, and Sect. 5 is dedicated to evaluating the performance of the ML-kNN algorithm on the set of prototypes generated with our methods. Finally, in Sect. 6 we provide some concluding remarks and research directions.

2 Motivation

Some classification algorithms, such as the ones founded on examples-based learning, use a training set to estimate the class label, which causes the scalability problems when the size of the training set increases. In this case, the number of training objects affects the computational cost of a method [13,15]. The nearest neighbour rule is an example of a high computational cost method when the number of examples is large [2].

The most popular algorithm in this category is kNN. The computational complexity of kNN is $O(nm)$, where n and m are the size of dataset and the dimensionality of embedding space. Thus, these methods are computationally very expensive on large-scale datasets. The purpose of the NP approach is to reduce storage costs and learning technique processes based on examples. In the literature, several papers [2,15,25] on this issue have been proposed in the context of single-label learning.

Nevertheless, to the best of our knowledge, the only relevant work relating to NP in the field of multi-label classification is the *kNNc* method described in [7]. It works in two stages, by combining prototype selection techniques with example-based classification. First, a reduced set of objects is obtained by prototype selection techniques used in classical classification [18]. The goal of this stage is to determine the set of labels which are nearest to the ones in the object to be classified. Then, the full set of samples is used, but limiting the prediction to the labels inferred in the previous step.

Unlike that study, this research proposes three new proposals for the generation of prototypes using different alternatives to granulate the datasets.

3 Granular Computing

Basic issues of Granular Computing may be studied from two related aspects, the construction of granules and computation with granules. The former deals with the formation, representation, and interpretation of granules, while the latter deals with the utilization of granules in problem solving [22,29].

In the construction of granules, it is necessary to study a criteria for deciding if two elements should be put into the same granule, based on available information. Typically, elements in a granule are drawn together by indistinguishability, similarity, proximity, or functionality [29].

With the granulation of universe, one considers elements within a granule as a whole rather than individually. The loss of information through granulation implies that some subsets of the universe can only be approximately described. The Rough Set Theory (RST) [19,23] is one of the most representative theories within Granular Computing. It deals mainly with the approximation aspect of information granulation. It uses two main components: an information system and an indiscernibility relation. The former is defined as $IS = (U, A)$, where U is a non-empty finite set of objects, and A is a non-empty finite set of attributes that describe each object. A particular case are the decision systems where $DS = (U, A \cup \{d\})$, whereas $d \notin A$ is the decision class.

In the classical RST, the relation of indiscernibility (R) is defined as an equivalence relation [20]. From this point on, $[x]_R$ defines an equivalence class of an element $x \in U$ under R, where $[x]_R = \{y \in U : yRx\}$, i.e. the equivalence class of an element includes all objects in the universe indiscernible from x. Each equivalence class may be viewed as a granule consisting of indistinguishable elements. Two objects are equivalent if they have exactly the same value with respect to a set of attributes. It means that two inseparable objects could incorrectly be labeled as separable, making the relationship excessively strict [28].

This problems can be alleviated in some extent by extending the concept of inseparability relation [24] and replacing the equivalence relation with a weaker binary relation. Equation (1) shows an indiscernibility relation,

$$R : xRy \Longleftrightarrow \delta(x, y) \geq \xi \tag{1}$$

where $0 \leq \delta(x, y) \leq 1$ is a similarity function. This weak binary relation states that objects x and y are inseparable as long as their similarity degree $\delta(x, y)$ exceeds a similarity threshold $0 \leq \xi \leq 1$. This relation actually defines a similarity class $\overline{R}(x) = \{y \in U : yRx\}$ that replaces the equivalence class.

The similarity function could be formulated in a variety of ways, for example, $\delta(x, y) = 1 - \varphi(x, y)$ with $\varphi(x, y)$ being the distance between objects x and y. In reference [27] the authors studied the properties of several distance functions which allow comparing heterogeneous instances, i.e., objects comprising both numerical and nominal attributes.

4 Methods for the Generation of Prototypes from MLC Datasets Based on Granular Computing

As mentioned, in MLC scenarios an object may be associated with multiple labels. Let $mlDS = (U, A \cup L)$ be a multi-label decision system, where the set U is a non-empty finite set of objects, A is a non-empty finite set of attributes that describe each observation, and $L = \{L_1, L_2, \ldots, L_k\}$ is a non-empty finite set of labels such that the label domain is $L_i = \{0, 1\}$.

Prototype-based classification determines the value of the decision class of a new object by analyzing its similarity to a set of prototypes generated from the initial set of objects. By doing so, we must define what is considered to be a decision class in the MLC context. For example,

- Each combination C_i of labels represents a decision value. For example, let $L = \{L_1, L_2, L_3\}$ denote the set of labels, a combination of labels could be “101”, pointing out that the object belongs to the labels L_1 and L_3, then “101” defines a decision class, so that all objects associated with labels L_1 and L_3 belong to that decision class.
- Each label (L_i) is considered a decision value, so that all the objects associated that label belong to this decision class. According with this definition, in the example above there are three decision classes.

The basic idea of the *first two proposed algorithms* below is similar. Both are iterative algorithms in which a similarity class is built using the similarity relationship defined in Eq. (1). An object may belong to several similarity classes at the same time. However, when an object is included in a similarity class, it is not taken into account to build a new similarity class from it. Each similarity class consists of a granule that is used to build a prototype.

From this, a prototype or centroid is built for a set of similar objects. Each prototype is composed of both conditional and label attributes. To add their information both by condition (attribute values) and by decision (labels values) an aggregation operator is used. In the case of conditional attributes, the average can be used as the aggregation operator if the attribute value is numeric, or the mode if the attribute value is nominal.

The way in which the part of the prototype related to the labels is built differs between the two algorithms, exactly based on what is considered a decision class. In the case of Algorithm 1 each combination of labels represents a decision value, however Algorithm 2 considers each label independently as a decision value. In this way, the first algorithm builds its decision class from the most common combination of labels in the granule, while the second algorithm does it taking into account the labels independently. The resulting prototype will have as decision values the most common labels of the objects in the granule.

Algorithm 1. GP1mlTS

```
1: Initialize objects' counter
        Used [i] = 0 i=1,..., n
        PrototypeSet = $\emptyset$
2: While $\exists i$ : Used [i] = 0
        j = i
        Construct the similarity class $\overline{R}(O_j)$
        Construct a vector $P = [P_{cond}, P_{dec}]$ from all objects in $\overline{R}(O_j)$
            $P_{cond}$ is calculated from the set of values of the attributes $(A)$ of all objects
            in $\overline{R}(O_j)$ and using an aggregation operator
            $P_{dec}$ is calculated from the most common label combination $(C)$ among
            all existing label combinations in the objects in $\overline{R}(O_j)$
        PrototypeSet = PrototypeSet $\cup$ P
        Used [j] = 1 for all the objects in $\overline{R}(O_j)$
3: Return PrototypeSet
```

In contrast to the first two algorithms, the *third algorithm* performs a different granulation of the universe. In this case, the decision class of the objects is taking into account to build the granulation of the data. The basic idea is to perform a granulation of the universe taking into account the labels instead of the condition attributes. Therefore, a granule is built for each label, so it will include all objects that are labeled with that decision label.

The partition and the covering of an information space are two common types of granulation of the universe [9]. The granulation obtained by this algorithm is

Algorithm 2. GP2mlTS

```
1: Initialize objects' counter
       Used [i] = 0 i=1,..., n
       PrototypeSet = $\emptyset$
2: While $\exists i$ : Used [i] = 0
       j = i
       Construct the similarity class $\overline{R}(O_j)$
       Construct a vector $P = [P_{cond}, P_{dec}]$ from all objects in $\overline{R}(O_j)$
           $P_{cond}$ is calculated from the set of values of the attributes ($A$) of all objects
           in $\overline{R}(O_j)$ and using an aggregation operator
           $P_{dec} = \{L_1, L_2, \ldots, L_k\}$, where $L_k = 1$ if most of the objects in $\overline{R}(O_j)$
           are labeled with that label, otherwise $L_k = 0$
       PrototypeSet = PrototypeSet $\cup$ P
       Used [j] = 1 for all the objects in $\overline{R}(O_j)$
3: Return PrototypeSet
```

a covering, since any objects could belong to two or more information granules. A prototype is then generated for each granule similar to the Algorithm 2. This procedure is formalized in Algorithm 3.

Algorithm 3. GP3mlTS

```
1: PrototypeSet = $\emptyset$
2: For each $Li \in L$
       Construct the equivalence class $[L_i]_R$
       Construct a vector $P = [P_{cond}, P_{dec}]$ from all objects in $[L_i]_R$
           $P_{cond}$ is calculated from the set of values of the attributes ($A$) of all objects
           in $[L_i]_R$ and using an aggregation operator
           $P_{dec} = \{L_1, L_2, \ldots, L_k\}$, where $L_k = 1$ if most of the objects in $[L_i]_R$
           are labeled with that label, otherwise $L_k = 0$
       PrototypeSet = PrototypeSet $\cup$ P
3: Return PrototypeSet
```

5 Results and Discussion

In this section, we explore the performance of our prototype generation methods when coupled with the ML-kNN classification algorithm. To accomplish that, we use *Hamming Loss* (HL) metric which is a well-known performance measure in MLC scenarios [16]. This metric is defined as follows,

$$HL = \frac{1}{n}\frac{1}{k}\sum_{i=1}^{n}|Y_i \Delta Z_i| \tag{2}$$

where Δ operator returns the symmetric difference between Y_i (the real label set of the ith instance) and Z_i (the predicted one).

To perform the simulations, we rely on 12 multi-label datasets taken from the well-known RUMDR [8] repository. Table 1 summarizes the number of instances, attributes, and labels for each dataset. In the adopted datasets, the number of instances ranges from 1,675 to 10,491, the number of attributes from 294 to 1,836, and the number of labels from 6 to 400.

Table 1. Characterization of the MCL datasets used in our study.

	Domain	Instances	Attributes	Labels
bibtex(D1)	Text	7395	1836	159
corel5k(D2)	Images	5000	499	374
enron(D3)	Text	1702	1001	53
scene(D4)	Images	2407	294	6
stackex_chemistry(D5)	Text	6961	540	175
stackex_chess(D6)	Text	1675	585	227
stackex_cooking(D7)	Text	10491	577	400
stackex_cs(D8)	Text	9270	635	274
stackex_philosophy(D9)	Text	3971	842	233

We also studied the reduction coefficient, $Red(.)$ [5]. This measure, in Eq. (3) indicates by how much the number of objects is reduced, that is, the proportion between the size of the set of prototypes (P) and the universe (U),

$$Red(.) = \frac{|U| - |P|}{|U|} * 100 \tag{3}$$

Figure 1 displays the reduction coefficient achieved once the proposed prototype generation methods are used on each dataset. In this experiment, we have adopted the Heterogeneous Euclidean-Overlap Metric (HEOM), which computes the normalized Euclidean distance between numerical attributes and an overlap metric for nominal attributes [27]. The similarity threshold ξ used in Eq. (1) ranges from 0.85 to 0.95.

It is worth mentioning that, for each datasets, we have estimated the HL value by using a 10-fold cross validation scheme. For each fold, this procedure splits the whole training set into two data pieces, namely, the training set and the test set. It should be highlighted that, while the training set is used to generate the set of prototypes. The test set is never modified so that it only serves to compute the HL associated with the current fold.

From the results in Fig. 1 we can conclude that our methods achieve a reduction rate higher than 20% in most problems. The $GP1mlTS$ and $GP2mlTS$ methods have a similar behaviour. However, the $GP3mlTS$ method reports reduction coefficients even higher than 90%. On the other hand, Fig. 2 displays the HL values achieved by the ML-kNN method with the original multi-label datasets, and the results obtained after using the set of prototypes generated by each of the methods proposed in this paper.

The results show that the prototypes generated for each dataset leads to HL values similar to those obtained with the original dataset. Only in the case of the *scene* dataset there is a significant difference in the LH value, especially when we

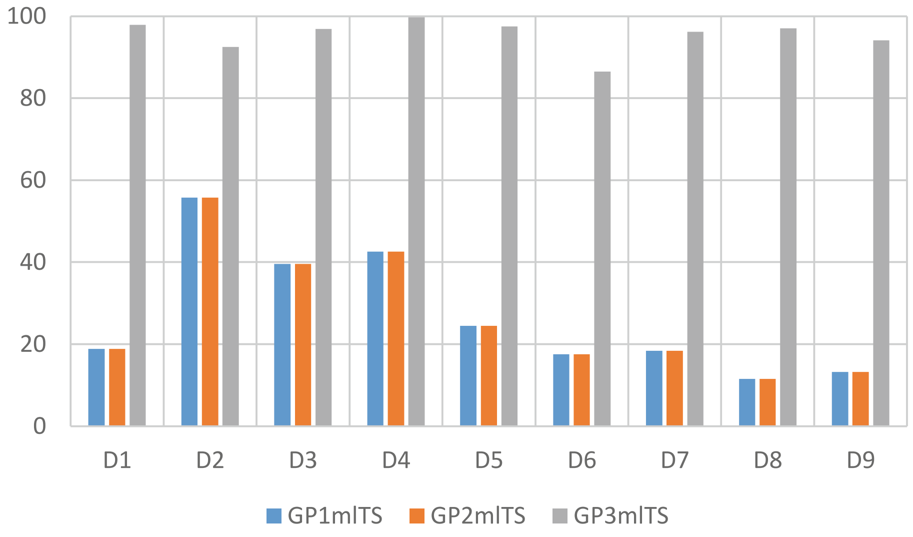

Fig. 1. Reduction percent achieved by each edition method.

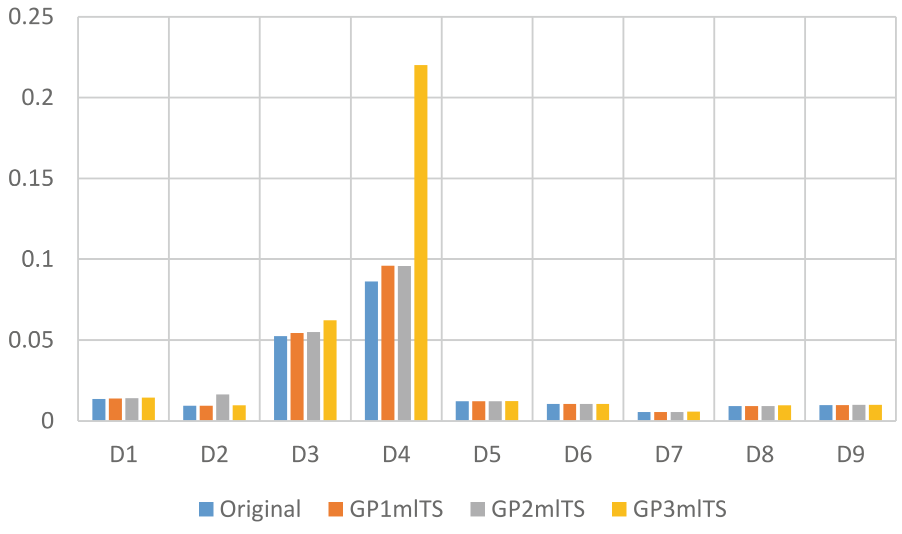

Fig. 2. HL values achieved by the ML-kNN method.

use the set of prototypes generated by the *GP3mlTS* method with respect to the original dataset. This is due to the fact that this dataset has few labels (exactly 6 labels), thus only a few prototypes are generated. In short, the results showed that our proposal provides a suitable trade-off between algorithm's performance and the number of training examples in the dataset.

6 Concluding Remarks

The *Prototype Generation* algorithms have proved their usefulness by improving some kNN issues such as computational time, noise elimination or memory use. Although the extensive work in multi label classification, as far as we know, the topic of prototype generation has not received any attention so far. This paper proposes three methods based on Granular Computing for the generation of prototypes.

After analyzing the reduction coefficient, it could be concluded that the proposed methods achieve a significant reduction of the datasets from the resulting prototypes, while preserving the efficacy of the ML-kNN method in most case studies.

The set of prototypes generated by these methods could be used as a learning set for other learning algorithms, even those not intended for example-based learning.

References

1. Aggarwal, C.C.: Data Classification: Algorithms and Applications. CRC Press, New York (2014)
2. Barandela, R., Cortés, N., Palacios, A.: The nearest neighbor rule and the reduction of the training sample size. In: Proceedings 9th Symposium on Pattern Recognition and Image Analysis, vol. 1, pp. 103–108 (2001)
3. Bargiela, A., Pedrycz, W.: Granular Computing: An Introduction, vol. 717. Springer, Berlin (2012)
4. Bello, R., Falcón, R., Pedrycz, W.: Granular Computing: At the Junction of Rough Sets and Fuzzy Sets, vol. 224. Springer, Berlin (2007)
5. Bermejo, S., Cabestany, J.: A batch learning vector quantization algorithm for nearest neighbour classification. Neural Process. Lett. **11**(3), 173–184 (2000)
6. Bezdek, J.C., Kuncheva, L.I.: Nearest prototype classifier designs: an experimental study. Int. J. Intell. Syst. **16**(12), 1445–1473 (2001)
7. Calvo-Zaragoza, J., Valero-Mas, J.J., Rico-Juan, J.R.: Improving knn multi-label classification in prototype selection scenarios using class proposals. Pattern Recogn. **48**(5), 1608–1622 (2015)
8. Charte, F., Charte, D., Rivera, A., del Jesus, M.J., Herrera, F.: R ultimate multilabel dataset repository. In: Martínez-Álvarez, F., Troncoso, A., Quintián, H., Corchado, E. (eds.) HAIS 2016. LNCS (LNAI), vol. 9648, pp. 487–499. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-32034-2_41
9. Chen, B., Sun, M., Zhou, M.: Granular rough theory: a representation semantics oriented theory of roughness. Appl. Soft Comput. **9**(2), 786–805 (2009)

10. Cover, T.M., Hart, P.E., et al.: Nearest neighbor pattern classification. IEEE Trans. Inf. Theory **13**(1), 21–27 (1967)
11. García, S., Cano, J.R., Herrera, F.: A memetic algorithm for evolutionary prototype selection: a scaling up approach. Pattern Recogn. **41**(8), 2693–2709 (2008)
12. García, S., Luengo, J., Herrera, F.: Data Preprocessing in Data Mining. Springer, Berlin (2015)
13. García-Durán, R., Fernández, F., Borrajo, D.: A prototype-based method for classification with time constraints: a case study on automated planning. Pattern Anal. Appl. **15**(3), 261–277 (2012)
14. Guan, D., Yuan, W., Lee, Y.K., Lee, S.: Nearest neighbor editing aided by unlabeled data. Inf. Sci. **179**(13), 2273–2282 (2009)
15. Hernández, F., et al.: An approach for prototype generation based on similarity relations for problems of classification. Computación y Sistemas **19**(1), 109–118 (2015)
16. Herrera, F., Charte, F., Rivera, A.J., del Jesus, M.J.: Multilabel classification. Multilabel Classification, pp. 17–31. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-41111-8_2
17. Kim, S.W., Oommen, B.J.: A brief taxonomy and ranking of creative prototype reduction schemes. Pattern Anal. Appl. **6**(3), 232–244 (2003)
18. Nanni, L., Lumini, A.: Prototype reduction techniques: a comparison among different approaches. Expert Syst. Appl. **38**(9), 11820–11828 (2011)
19. Pawlak, Z.: Rough sets. Int. J. Comput. Inf. Sci. **11**(5), 341–356 (1982)
20. Pawlak, Z., Skowron, A.: Rough sets: some extensions. Inf. Sci. **177**(1), 28–40 (2007)
21. Pedrycz, W.: Granular Computing: Analysis and Design of Intelligent Systems. CRC Press, New York (2016)
22. Pedrycz, W., Homenda, W.: Building the fundamentals of granular computing: a principle of justifiable granularity. Appl. Soft Comput. **13**(10), 4209–4218 (2013)
23. Pedrycz, W., Skowron, A., Kreinovich, V.: Handbook of Granular Computing. Wiley, Hoboken (2008)
24. Slowinski, R., Vanderpooten, D.: A generalized definition of rough approximations based on similarity. IEEE Trans. Knowl. Data Eng. **12**(2), 331–336 (2000)
25. Triguero, I., Derrac, J., Garcia, S., Herrera, F.: A taxonomy and experimental study on prototype generation for nearest neighbor classification. IEEE Trans. Syst. Man Cybern. Part C Appl. Rev. **42**(1), 86–100 (2012)
26. Tsoumakas, G., Katakis, I., Vlahavas, I.: Mining multi-label data. In: Maimon, O., Rokach, L. (eds.) Data Mining and Knowledge Discovery Handbook, pp. 667–685. Springer, Boston (2009). https://doi.org/10.1007/978-0-387-09823-4_34
27. Wilson, D.R., Martinez, T.R.: Improved heterogeneous distance functions. J. Artif. Intell. Res. **6**, 1–34 (1997)
28. Yao, Y., Zhong, N.: Granular computing using information tables. In: Lin, T.Y., Yao, Y.Y., Zadeh, L.A. (eds.) Data Mining, Rough Sets and Granular Computing. STUDFUZZ, vol. 95, pp. 102–124. Springer, Heidelberg (2002). https://doi.org/10.1007/978-3-7908-1791-1_5
29. Zadeh, L.A.: Toward a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic. Fuzzy Sets Syst. **90**(2), 111–127 (1997)
30. Zhang, M.L., Zhou, Z.H.: Ml-knn: a lazy learning approach to multi-label learning. Pattern Recogn. **40**(7), 2038–2048 (2007)
31. Zhang, M.L., Zhou, Z.H.: A review on multi-label learning algorithms. IEEE Trans. Knowl. Data Eng. **26**(8), 1819–1837 (2014)