Prototype-based classification by fuzzification of cases

Parisa KordJamshidi
Dep.Telecommunications
and Information Processing
Ghent university
pkord@telin.ugent.be

Bernard De Baets
Dep. Applied Mathematics
Biometrics and Process Control
Ghent university
bernard.debaets@UGent.be

Guy De Tre
Dep.Telecommunications
and Information Processing
Ghent university
guy.detre@telin.ugent.be

Abstract

A fuzzy prototype-based method is introduced for learning from examples. A special kind of prototype vector with fuzzy attributes is derived for each class from aggregating fuzzified cases for the purpose of concept description. The fuzzified cases are derived by defining a fuzzy membership function for each attribute of the sample cases. In a first method, for the classification of a new case, the membership degrees of its crisp attributes to fuzzy aggregated prototypes are measured. In a second method, after fuzzifying the new case, fuzzy set comparison methods are applied for measuring the similarity. The methods are compared to case-based ones like POSSIBL and kNN using UCI machine learning repository. We also make comparisons by using various transformation methods from probabilities to possibilities instead of defining membership functions.

Keywords: Fuzzy prototype, classification, prototype-based learning

1 Introduction

One of the main challenges for machine learning methods, in particular of model-based ones, is the extraction of a model from observed data for concept description. For example in statistical methods usually the frequencies are considered and the probability of occurrence of the attribute values in sample cases is the most important criterion to classify new cases. In these methods the complex part is the model building.

On the other hand in case-based learning methods it is not necessary to build a model because, as these methods are called lazy learners, the learning does not happen, indeed. The reasoning process starts when a new case arrives and the main criterion is the similarity between the new case and the observed stored cases, like kNN (k-nearest neighbor). Therefore this kind of methods are very slow when they process a new case.

This work concerns the classification framework. In the suggested method, a fuzzy model is constructed according to the stored cases. In the first step we assign a membership function (fuzzy set) to each attribute of a stored case. We propose a subjective method for the fuzzification of attribute values and then we aggregate the fuzzy sets for each attribute in every class. The result is a fuzzy vector representing each class. Hence, we call it a fuzzy prototype. The new cases are compared to the fuzzy prototypes for classification purpose.

For comparing the new case, we applied two different methods from fuzzy pattern matching [8, 6]. In the first one, the membership degree of the attribute values of the new case to the fuzzy prototype vectors is measured. In the second method, we fuzzify the new case in the same way as done for the sample cases. Hence, we use a fuzzy comparison method for measuring the similarity between the new case and the fuzzy prototypes. Finally, in both methods the maximum aggregated similarity determines the class of the new case.

Since we do not use a kind of averaging but use fuzzy aggregation of our sample cases for making our prototypes, the main advantage of the method is that it is very flexible compared to prototype-based methods like k-means (in clustering) [7] and Rocchio (in text classification) [17], which make prototypes by some kind of averaging on sample cases. So the accuracy is much better than those kinds of approaches, with nearly the same efficiency. On the other hand, the classification process is very efficient compared to case-based methods like kNN or Possibilistic Instance-Based learning, which are among the outperforming methods in many applications [11], while its accuracy is nearly the same.

In Section 2 we explain our suggested fuzzy prototype-based learning, including some subsections to clarify how we fuzzify the attributes, how we build the prototypes and finally how we classify the new cases. The experimental results and the comparisons with related works are presented in Section 3 including comparisons with kNN and PossIBL and also with using transformations from probabilities to possibilities. Finally, Section 4 presents the conclusion of this work.

2 Fuzzy prototype-based learning

The most important difficulty of case-based methods like kNN and also possibilistic case-based method is the high cost (in terms of execution time) of classifying new cases. Since they are lazy learners, all the computations take place at classification time rather than when the sample set is first encountered. This characteristic prohibits these approaches in many applications.

One way to improve the efficiency is to find some reduced representatives or prototypes to represent the whole training data. In case-based methods like kNN each case is a prototype, so we have many prototypes for each class to be compared to new cases [2]. If we reduce the number of prototypes and use those representatives for classification, the efficiency would raise.

The simplest and most efficient approach for building such a model is deriving one prototype vector for each class by some kind of weighted averaging like the Rocchio method in text classification and k-means and fuzzy c-means in clustering. In these methods training a classifier comes down to building a generalized case for each category. A well-known disadvantage of these methods is their characteristic of dividing the data space linearly. This leads to a lower accuracy. Here, we have a predefined finite set of labels $L = \{\lambda_1, ..., \lambda_m\}$. Each class C_k has a label λ_k and for each X, $L(X_i)$ determines the class to which X belongs.

In the suggested method, the main idea is to derive one fuzzy prototype for each class. We have a set of n samples with known nominal labels and J relevant attributes. We represent each sample X_i as a vector of fuzzy attributes. This means that each attribute value is represented by a fuzzy set f_i^j which explains the jth attribute of the ith sample. If the attributes have crisp values we fuzzify them according to the nature of the data. We use a triangular membership function which is explained in the following section.

2.1 Fuzzifying attributes

In the first step, we fuzzify each attribute of every case in the sample or training set by defining a membership function for each one.

In fact, all problems arising in the theory of fuzzy sets are due to the lack of our knowledge of the interpretations of fuzzy. There are no guidelines or rules that can be used to choose the appropriate membership generation technique [15].

There are many subjective elicitation methods for membership functions like: polling, direct rating, reverse rating, interval estimation, membership function exemplification and pairwise comparison [5]. However, a particularly important concern in practice are approaches to construct membership functions from a given set of data. In this case fuzzy clustering techniques, neural network-based methods and also fuzzy nearest neighbor techniques has been suggested [5, 15]. Transferring the probabilities to possibilities using data histograms is also an interesting way for elicitation of possibilities and assigning them to membership grades [8, 15, 16].

In our method a membership function measures the degree of similarity of an element to the set in question.we also have a subjective view and we determine the parameters of our predefined membership function which comes from our interpretation of the fuzziness in this problem. However, the fuzzy methods are insensitive to a precise measurement of membership functions and the qualitative properties of the phenomena modeled using fuzzy sets, is more important [14].

Hereby, when we come across $v \in domain(f_i^j)$ the value of attribute f_i^j for a case X_i in our training set, we consider that v has the possibility of 1 in the class to which this case belongs. On the other hand we derive the boundaries of other possible values using the training set. If we look for the samples in the related class, then we get two boundaries a,b for which $v \in [a, b]$, in this class. We call them class boundaries. If we consider all values in the training set for this attribute we get c,d for which $v \in [c, d]$ and we call them training set boundaries.

According to our interpretation of fuzziness the closer the values are to the observed one the more similar they are to it and so they are more possible in the class domain. Therefore by defining a normalized triangular membership function on (a,b,v), parameters a and b locate the support of the triangle and the parameter v locates the core, we assign a degree of possibility to the values in between the class boundaries. This possibility would depend on the distance from v reversely. We can also use a wider support like training set boundaries for the triangle like (e,f,v) where $c \leq e \leq a$ and $b \leq f \leq d$. Specially while in real data we have class overlaps it would provide better results. Afterward we have cases with fuzzy attributes.

2.2 Building fuzzy prototypes

In the second step our concern is to make a prototype for each class. By a prototype we mean a representative model for each class, so the next comparisons will be performed by these prototypes and there is no more need to keep cases like case-based reasoning methods to compare one by one.

As we have fuzzy vectors from the previous step, our idea here is to aggregate the vectors(cases) in each class to make a prototype vector, by means of aggregation operations. Actually, we aggregate membership functions of the same attributes of all samples which have the same label. Consequently we will have one membership function (the result of the aggregation) for each attribute and so we derive a prototype vector with fuzzy attributes for each class. In a formal way for each class λ_k, $1 \leq k \leq m$, $prototype(\lambda_k) =$

$$\{Ag(\{X_i\})\,|L(X_i) = \lambda_k \ \ and \ \ 1 \leq i \leq n\}$$

where Ag is an aggregation function from $X^n \to X$ and $prototype(\lambda_k)$ is the fuzzy prototype for the class λ_k of the training set.
As we mentioned Ag works on each attribute separately. If we present each case X by its fuzzy attributes like $X = (f_1, ..., f_J)$ then we define the $Ag(\{X_i\}), 1 \leq i \leq n$ as

$$Ag(\{X_i\}) = (ag(\{f_i^1\}), ..., ag(\{f_i^J\})), 1 \leq i \leq n$$

where f_i^j is a fuzzy set related to the jth attribute of the ith case and ag is an aggregation operator. Actually, we do aggregation for the cases with the same label. Consequently we have a fuzzy vector $prototype(\lambda_k)$ for each class λ_k at the end of this phase. We denote each prototype vector as:

$$prototype(\lambda_k) = (p_k^1, ..., p_k^J)$$

where $1 \leq k \leq m$ and m is the number of classes.

2.3 Classification of new cases

In this phase we use the prototype fuzzy vectors to make a prediction for new cases.

Here we use the related ideas in fuzzy pattern matching methods [8]. The new case must be compared with all the class prototypes and the most similar prototype determines the class label of the new case.

Alternatively, we may fuzzify the new case also or work with the vector with crisp attribute values. In both situations we must compare the vectors, attribute by attribute. However, the method of comparison is completely different in these two situations.

Crisp new case In this approach we calculate the membership degree of each attribute value of the new case to the fuzzy set of the related attribute, in each prototype. In a more formal way, we present the new case by its crisp attribute values as:

$$X_{new} = (v_1, ..., v_J)$$

$p_k^j(v_j)$ is the membership degree of v_j to the p_k^j fuzzy set. Now we derive a vector D_k containing all the membership of the attribute values of the new case to kth class prototype.

$$D_k = (p_k^1(v_1), ..., p_k^J(v_J))$$

Fuzzy new case In this approach we fuzzify our new case. We can do this in the same way as with the sample cases by using the training set boundaries. Each attribute value converts to a fuzzy set with triangular membership function. This means that the values near to the present value are also accepted but to a lower possibility depending on the distance from the actual value. Here, the fuzzy set comparison methods can be used for comparing fuzzy attributes. In this case we may use overlap, inclusion or equality of two fuzzy sets [5]. The counting based comparisons can also be suitable choices for comparing two fuzzy sets[3, 5]. There is a lot of flexibility in choosing the comparison method and the choice depends on the application and the data. Here we use fuzzy overlap which is defined by the following formula, for two fuzzy sets F and G: $O(F, G) = ht(F \cap G) = \sup \min_{u \in U}(F(u), G(u))$
By using the same notation as for the crisp case, here the v_j's of the new case are fuzzy sets. If we use fuzzy overlap for comparing two fuzzy sets then $p_k^j(v_j)$ are the degree of overlap(similarity) between v_j and p_k^j fuzzy sets.

As the result of both methods, we will obtain the local similarities between the new case attributes and the prototype's. It is necessary to aggregate these degrees to derive the total measure as the membership degree of the new case to each class. Here again, different aggregation operators can be used, for example if there are some priorities among attributes, weighted aggregation operators can be used. For example OWA operators or if we want to include dependencies between attributes then some setting of Choquet integrals would be helpful [10, 8].

$$C_k(X_{new}) = W(p_k^1(v_1), ..., p_k^J(v_J))$$

where W is an aggregation operator and $C_k(X_{new})$ is the membership degree of new case X_{new} to class C_k. and then

$$L(X_{new}) = arg_{\lambda_k \in L} \max\{C_k(X_{new}) \,|L(C_k) = \lambda_k\}$$

It means that the maximum of the aggregated membership degrees determines the class label of X_new.

3 Related works

Fuzzy notions have been widely used in machine learning approaches in both model-based and case-based methods [12]. For example in case-based methods like kNN, the fuzzy version assigns a degree of membership to each new case instead of a unique label [13, 10]. A fuzzy version of model based kNN proposed in [7] and performed well.

Another case-based method which does possibilistic prediction is developed in [4]. Actually, in this work we compared our results to POSSIBL [11] and the other results of this work. It uses the similarity measures as membership degrees in an Instance-based method and it has a noticeable accuracy on UCI machine learning repository data.

There are also many model-based approaches that apply fuzzy and/or possibilistic notions for classification and clustering purposes. In the machine learning literature, almost all the existing methods have been combined with fuzzy notions in some way.

For example in clustering, k-means is a simple method which partitions the sample set into k clusters. The aim of this algorithm is to find cluster centroids for each group. A prototype-based approach has been used in classification of text documents known as Rocchio [17]. It makes a crisp and usually centroid vectors for each class.

k-Means, in its fuzzy version called fuzzy c-means [7], employs fuzzy partitioning such that an instance can belong to all clusters with different degrees of membership between 0 and 1. In addition, in graphical models like Bayesian networks [1] and Markov networks fuzzy and possibilistic concepts has been investigated. In decision trees, fuzzy rule-based methods and fuzzy neural networks, there also exist many research works in this area [12]. In some related works possibilities have been extracted from transformation of the probabilities and almost always the probabilities derived from frequencies in training data. In [8] this approach has been used for classification and feature extraction by fuzzy integrals through which the dependencies between attributes are taken into account. Variable transformation of probabilities to possibilities has been used in diagnosis by pattern recognition [16]. In the same work various characteristics of the different transformation methods has been compared.

An investigation on prototype-based methods has been developed well in [2] and compared with instance-based methods like kNN.

4 Experimental results

4.1 Experimental setup

The experiments were performed with the same setting as Hüllermeier [11, 10], for comparison purposes. He compared, his suggested possibilistic instance based classifier, PossIBL with kNN for different k's. We added our results to his tables.

In that setting, five public datasets were chosen from the UCI machine learning repository, Table 1. In our first method F-P-B1, we compared the crisp new case with the fuzzy prototypes of our prototype based method. In the second method F-P-B2, the new case has been fuzzified beforehand and then compared to fuzzy prototypes. For aggregating training cases and making prototypes we used the max operator in both experiments. At the time of classification, for the aggregation of membership degrees or similarities, we used the mean operator because here we ignore dependencies and priorities between attributes.

We also used the idea of fuzzy pattern recognition [8]. We used different transformation methods like Variable Transformation (VT), Normalized Transformation (NT), Asymmetric, Transformation of Dubois and Prade (AT) and Symmetric Transformation of Dubois and Prade (ST) [16]. We made our prototypes according to the obtained possibilities to compare it to our proposed prototype derived from aggregating the fuzzified cases.

The results showed that symmetric transformation worked better than other transformations on these data sets so, for saving the space we compare only the results of ST with other methods.

In a single simulation run, the data set divided at random into a training set and a test set, Table 1. Results are summarized by means of statistics for the percentage of correct classifications (mean, minimum, maximum and standard deviation). The experiments on four data sets are represented in Tables 1-4. The experiments on another data set named Balance Scale Database will be discussed in section 4.2.3 .

4.2 Analysis of the results

4.2.1 Accuracy

As can be seen in the tables, for three data sets, the F-P-B in both cases is nearly the same and is comparable with kNN and POSSIBL. On the Wine Recognition data set F-P-B worked even much better. Since these

Data set	#of observ.	#of predictive att.	#of classes	Training set size
GLASS IDENTIFICATION	214	9	7	100
WINE RECOGNITION	178	13	3	89
IRIS PLANT	150	4	3	75
PIMA INDIANS DIABETES	768	8	2	380
BALANCE SCALE	625	4	3	300

Table 1: description of Data sets

Algorithm	mean	min	max	std
F-P-B1	0.9102	0.7733	0.9867	0.029
F-P-B2	0.9123	0.800	1.000	0.027
PossIBL	0.9574	0.8400	1.000	0.020
1NN	0.9492	0.8400	1.000	0.019
3NN	0.9554	0.8666	1.000	0.017
5NN	0.9586	0.8533	1.000	0.018
w5NN	0.9561	0.8400	1.000	0.018
ST	0.8583	0.64	0.9733	0.038

Table 2: Iris Plant Database (10000 simulation runs)

Algorithm	mean	min	max	std
F-P-B1	0.6179	0.4386	0.7719	0.043
F-P-B2	0.6283	0.4211	0.7632	0.042
PossIBL	0.6841	0.5300	0.8400	0.041
1NN	0.6870	0.5200	0.8200	0.041
3NN	0.6441	0.4800	0.8100	0.042
5NN	0.6277	0.4800	0.7800	0.041
w5NN	0.6777	0.5000	0.8300	0.041
ST	0.4677	0.2807	0.6754	0.057

Table 3: Glass Identification Database (10000 simulation runs)

Algorithm	mean	min	max	std
F-P-B1	0.9465	0.8202	1.0000	0.028
F-P-B2	0.9570	0.8427	1.0000	0.025
PossIBL	0.7148	0.5506	0.8652	0.040
1NN	0.7163	0.5843	0.8652	0.040
3NN	0.6884	0.5506	0.8315	0.040
5NN	0.6940	0.5730	0.8090	0.039
w5NN	0.7031	0.5730	0.8315	0.040
ST	0.7668	0.4157	0.9325	0.076

Table 4: Wine Recognition Database (1000 simulation runs)

Algorithm	mean	min	max	std
F-P-B1	0.6711	0.6030	0.7448	0.020
F-P-B2	0.6598	0.5927	0.7345	0.019
PossIBL	0.7096	0.6421	0.7711	0.019
1NN	0.6707	0.6132	0.7289	0.019
3NN	0.6999	0.6447	0.7500	0.018
5NN	0.7190	0.6553	0.7684	0.018
w5NN	0.6948	0.6421	0.7474	0.018
ST	0.6865	0.5902	0.7706	0.024

Table 5: Pima Indians Diabetes Database (1000 simulation runs)

methods are case-based and are among the outperforming methods in machine learning, the results are promising. F-P-B's are much better on almost all data sets, Compared to ST which uses symmetric transformation of probabilities to possibilities. It means when we need possibilities, defining fuzzy subjective membership functions works better than using probabilities as initial data. However, while the method is very efficient compared to case-based methods and very accurate compared to using probability transformations, so the results would be satisfactory in many applications. Fuzzification of the new cases in F-P-B2 could make very small improvements but did not have any sharp effect on the results. It is worthy to mention that our results are without any enhancement and noise reduction.

4.2.2 Complexity

From the complexity point of view, in case-based methods like POSSIBL and kNN, at classification time, each case must be compared with all cases in the training set. So, the complexity is of order $O(n)$ where n is the size of training set. In our suggested method

F-P-B each case must only be compared with prototypes. Because we have one prototype for each class, the complexity order decreases to $O(m)$ where m is the number of classes. Since always $m << n$, we conclude that from the complexity point of view F-P-B is very efficient and comparable to other prototype-based methods.

However, in case based methods we have the process of the comparison of two cases. Its complexity depends on the number of attributes and the similarly measurement process. Here, we have a different approach to compare a case with fuzzy prototypes. This process includes the complexity of deriving membership degrees from membership functions. The complexity of this process depends to some extent on the implementation of membership functions. For example, in some way by means of indexing we can decrease the order of finding membership degrees to $O(1)$. In this case the comparison with one prototype would only depend on the number of attributes, similar to case-based methods. Therefore the approach has a total complexity of a simple prototype-based method which is comparatively very efficient.

Finally, in our proposed method we have the complexity of making prototypes, but this is only at the training step and does not effect the efficiency of classification.

4.2.3 Dependency ignorance

On the fifth data set, Balance-scale, our method did not work properly. If we consider on this data set, it has only four attributes for each case. The class label would be determined according to the comparison of $a_1 \times a_2$ and $a_3 \times a_4$ (a_1, a_2, a_3 and a_4 show the attributes). If these are equal, the case belongs to class 1. In the other two situations (greater or less than), the case belongs to class 2 or 3. So a strong dependency between attributes is observable which our method in its simple form is not able to deal with. However, aggregation operators like fuzzy integrals [8] can provide a better situation by handling the dependencies, but with more complexity.

Defining membership functions also can be one critical source of wrong classifications in some kind of data sets. Since we give a possibility 1 to every observed attribute value, by analysis of the Balance-scale data we see that our fuzzification method is not suitable for this data because the possibility distribution of one attribute completely depends on the value of other attributes. For example if we have a value for a_1 in class 1, then some values are impossible for a_2 because the above mentioned equation must hold. Thus we need a more sophisticated membership generation method which includes these kind of conditions and dependencies.

Hence, in case of dependency between attributes we must be careful with defining membership functions or choosing aggregation operators. In fact, in machine learning methods for choosing a suitable approach, we always must know the characteristics of our data to some extent.

5 Conclusions

In this paper, we investigated fuzzy and possibilistic reasoning methods, we tried to deal with the shortcomings of the lazy case-based methods and overcome the problem of low efficiency of kNN and POSSIBL. We went toward making a fuzzy prototype for each class.

For the classification of a new case the comparisons are done with fuzzy prototypes instead of with every case in our training set. This solution decreases the complexity of our classifier from O(number of cases), to O(number of classes). So, this method is from efficiency point of view, comparable with prototype-based methods which do a kind of simple averaging or weighted averaging on each class like the idea of Rocchio in text classification and k-means and fuzzy c-means in clustering.

On the other hand the experimental results on the UCI repository show that the accuracy of our suggested method is comparable to that of case-based methods while they are usually in between the best methods in many applications of machine learning. In addition ac-

curacy is usually very much better than using transformations from probability to possibilities.
As our method neglects the dependency between attributes of the cases this will be our next research interest to include these dependencies when we define the membership functions or when we aggregate the membership degrees, by means of more sophisticated aggregation operators.
However coming to fuzzy and possibilistic approaches would have more advantages while we are interested in measuring the reliability of our classifier and the uncertainty of final decisions. These are also our future research interests.

Acknowledgements

This research has been partially supported by Ilam university, Iran.

References

[1] C.Borgelt, R.Kruse. Graphical Models: Methods for Data Analysis and Mining. *Wiley*,2001.

[2] F.Chang, C.C.Lin, C.J.Lu. Adaptive Prototype Learning Algorithms: Theoretical and Experimental Studies. In *J. Machine Learning Research*, vol 7,p 2125-2148, 2006.

[3] B.De Baets, H.De Meyer, H.Naessens. A class of rational cardinality-based similarity measures. In *J. Comput. Appl. Math.*, vol 132, p 51-69, 2001.

[4] D.Dubois, E.Hüllermeier, H.Prade. Fuzzy set-based methods in instance-based reasoning. In *IEEE Transactions on Fuzzy Syst.*, vol 10, no 3, p 322-332, 2002.

[5] D.Dubois, H.Prade. Fundamentals of Fuzzy Sets. In *Kluwer Academic Publishers*, 2000.

[6] D.Dubois, H.Prade, C.Testemale. Weighted fuzzy pattern matching. In *Fuzzy Sets Syst.*,vol 28, no 3, p 313-331, 1988.

[7] G.Guo, D.Neagu. Fuzzy *k*NNModel Applied to Predictive Toxicology Data Mining. In *Int.J.of Computational Intelligence and Applications* vol 5, no 3, p 321-333, 2005.

[8] M.Grabisch. Fuzzy Integral for Classification and Feature Extraction. In *Fuzzy measures and integrals: Theory and applications, Physica-Verlag*, p 415-434, 2000.

[9] F.Hoppner, F.Klawonn, P.Eklund. Learning indistinguishability from data. In *Soft computing*, vol 6, p 6-13, 2002.

[10] E.Hüllermeier. Case-based Approximate Reasoning. *Springer*,2007.

[11] E.Hüllermeier. Possibilistic instance-based learning. In *Artificial Intelligence*, vol 148, p 335-383, 2003.

[12] E.Hüllermeier. Fuzzy methods in machine learning and data mining: Status and prospects. In *Fuzzy Sets and Syst.* , vol 156, p 387-406, 2005.

[13] J.M.Keller, R.Gray, J.A.J.R. Givens. A Fuzzy *k*-nearest Neighbor Algorithm. In *IEEE Trans. syst. Man Cybernet* vol 15(4), p 580-585, 1985.

[14] R.Kruse, J.Gebhardt, F.Klawonn. Foundations of fuzzy Systems.*John Wiley and Sons,New York*, 1994.

[15] S.Medasani, J.Kim, R.Krishnapuram. An overview of membership function generation techniques for pattern recognition. In *Int.J.of Approximate Reasoning* vol 19, p 391-417, 1998.

[16] M.Sayed Mouchaweh, P.Billaudel. Variable Probability-Possibility Transformation for the Diagnosis by Pattern Recognition. *Int.J.of Computational Intelligence: Theory and Practice*, vol 1(1),p 9-25, 2006.

[17] F.Sebastiani. Machine Learning in Automated Text Categorization. In*ACM Computing Surveys*,vol 34, p 1-47, 2002.