Distance-based Evaluation Function for First-order Rule Construction

Nirattaya Khamsemanan[1], Cholwich Nattee[1], and Masayuki Numao[2]

[1]Sirindhorn International Institute of Technology, Thammasat University, Thailand
{nirattaya,cholwich}@siit.tu.ac.th
[2]The Institute of Scientific and Industrial Research, Osaka University, Japan
numao@ai.sanken.osaka-u.ac.jp

Abstract. Inductive Logic Programming (ILP) is a learning approach that allows us to learn from examples in form of First-Order Logic (FOL). Due its lack of properties, various existing techniques for the attribute-value learning cannot be applied. Evaluation functions used by various ILP systems are thus based on the numbers of positive and negative examples covered by a rule. We propose new evaluation functions that take into account the distances between covered positive examples. This allow us to guide the search to construct the rules that cover larger areas in the space. We also propose to use the eccentricity function to order the positive examples, so that an appropriate example can be considered first by the algorithm. The proposed technique yields higher accuracies that Aleph, the baseline ILP system on Mutagenesis and Alzheimer datasets.

1 Introduction

Inductive Logic Programming (ILP) [4] is a machine learning approach that unifies learning algorithms with first-order logic (FOL) representation. ILP allows training examples to be denoted in first-order logic form. It generates learning outcomes in the form of first-order rules. The rules show the generalization of the training examples. This approach has advantages over the attribute-value learning approach in both expressive power of the examples and the comprehensiveness of the output. Since the first-order data lack of properties of the Euclidean space, various techniques used in the attribute-value learning algorithms cannot be applied in order to build a set of hypotheses. Therefore, most of ILP systems construct output hypotheses by generating a set of candidate rules and evaluating them based on how the rules cover the positive and negative examples. A number of techniques have been proposed to apply existing techniques for the attribute-value learning to improve the performance of ILP's rule construction. For example, Landwehr et al. [2] propose an ILP system called 'kFOIL'. This ILP system uses functions generated by Support Vector Machines to evaluate and select the candidate rules.

Distance between objects has been used in a number of attribute-value learning algorithms to generate classification models. However, to the best of our knowledge, there has not been any study on the relationships between distance and generalization of the first-order logic objects. In this paper, we propose a modification of an evaluation function based on the distance between training examples. This modification allows us to obtain a set of rules that performs better that the rules constructed using the existing function. We also show that the eccentricity of positive examples also plays an important role in the performance of the learning outcome.

In Section 2, we describe the format of FOL datasets that are appropriate for our proposed functions. Section 3 provides the definition of the four-layer distance metric. Section 4 shows how to modify an evaluation function to include the information from distances. The results of the experiments are shown in Section 5. Finally, we conclude our work in Section 6.

2 Setting

We are working under the assumption that elements in a dataset is defined as follow:

Definition 1. *A (first-order logic) dataset $\mathcal{C}$ is a set of elements of the form*

$$ID^r = r(ID, x_1, ..., x_n),$$

where r is a predicate symbol. The first entry of an atom will be called rank 0, the $(i+1)^{th}$ entry will be call rank i of an atom. An argument in the rank 0 is called ID. An element of $\mathcal{C}$ is called an atom.

Definition 2. *An object X is an FOL object if its entire properties and identities are can be expressed by atoms in a (first-order logic) dataset $\mathcal{C}$. A set of all atoms with $ID = X$ is called the property set of X.*

A dataset $\mathcal{C}$ is then a union of property sets of FOL objects. A single-level structure object is a FOL object whose ID argument will not appear in rank 0 of other atoms. A multi-level structure object is a FOL object whose arguments in all entries of any atoms can be ID of other atoms.

3 The Four-Layer Distance Metric

Definition 3. *Suppose X and Y are two FOL objects whose properties are represented in a dataset $\mathcal{C}$ where both X and Y are main IDs. The four layer distance between X and Y is defined as follow:*

Layer 1: ***Distance function of arguments with respect to a predicate symbol r and rank i:*** *Suppose $X^r = r(X, x_1, x_2, \cdots, x_n)$ and $Y^r = r(Y, y_1, y_2, \cdots, y_n)$ are two atoms in $\mathcal{C}$ with the same predicate symbol r. The distance between x_i and y_i is defined as*

$$\Delta_{r,i}(x_i, y_i) = \begin{cases} 0 & \text{if } x_i = y_i, \\ 1 & \text{if } x_i \neq y_i, \text{ and } x_i \notin \mathbb{R} \text{ or } y_i \notin \mathbb{R}, \\ \dfrac{|x_i - y_i|}{\max(r,i)} & \text{if } x_i \neq y_i \text{ and } x_i, y_i \in \mathbb{R}. \end{cases}$$

where $\max(r,i)$ is the maximum difference of all pairs of arguments in the rank i of atoms with the predicate symbol r, ranging over $\mathcal{C}$.

Layer 2: ***Distance function of atoms with respect to a predicate symbol*** ***r:*** *Suppose $X^r = r(X, x_1, x_2, \cdots, x_n)$ and $Y^r = r(Y, y_1, y_2, \cdots, y_n)$ are two atoms in $\mathcal{C}$ with the same predicate symbol r. The distance function of two atoms with the same predicate symbol r is defined as*

$$d_r(X^r, Y^r) = \sqrt{\frac{\sum_{i=1}^{n} (\delta_{r,i}(x_i, y_i))^2}{n}}$$

where,

$$\delta_{r,i}(x_i, y_i) = \begin{cases} 0 & \text{if } x_i = y_i, \\ \varDelta_{r,i}(x_i, y_i) & \text{if at most one of } x_i, y_i \text{ is an ID}, \\ D(x_i, y_i) & \text{if both } x_i, y_i \text{ are ID's}. \end{cases}$$

Note that in the case where x_i and y_i are not equal and are both IDs of other atoms, then $\delta_{r,i}(x_i, y_i)$ is set to $D(x_i, y_i)$, ranging over $\varOmega'$, a set of all predicate symbols of atoms whose IDs are arguments in rank i of any atoms with the predicate symbol r and is defined recursively until x_i, y_i are single-level structure objects.

Layer 3: ***Distance between two FOL objects with respect to a predicate symbol r:*** *For a predicate symbol r, suppose there are p atoms with main $ID = X$ and q atoms with main $ID = Y$, then the r-distance between an object X and an object Y is*

$$D_r(X, Y) = \begin{cases} \max\{\max_{k=1}^{p} \min_{j=1}^{q} d_r(X^{r_k}, Y^{r_j}), \max_{j=1}^{q} \min_{k=1}^{p} d_r(X^{r_k}, Y^{r_j})\} & \text{if } p, q \neq 0 \\ 1 & \text{if } p \neq 0, q = 0, \\ & \text{or } \ p = 0, q \neq 0 \\ 0 & \text{if } p = q = 0 \end{cases}$$

Layer 4: ***The Four-layer Distance between two FOL objects:*** *For two objects main $ID = X$ and main $ID = Y$ whose properties are expressed as atoms in $\mathcal{C}$. The distance between X and Y is defined as*

$$D(X, Y) = \sqrt{\frac{\sum_{r \in \varOmega} (D_r(X, Y))^2}{|\varOmega|}}$$

where $\varOmega$ is the set of predicate symbols of atoms that contain main ID in $\mathcal{C}$. In the calculation of $D(x_i, y_i)$, if at some level, $D(x_i, y_i)$ is a function of itself, then $D(x_i, y_i)$ is obtained by solving that equation. This equation where $D(x_i, y_i)$ is a function of itself is call the distance equation of $D(x_i, y_i)$.

4 Distance Based Evaluation Function

We propose a modification of an evaluation function of an ILP system called 'Aleph' [6]. This ILP system applies the concept of inverse entailment [3] to construct bottom clauses, and searches for the most appropriate clauses according to a search strategy and an evaluation function. The default evaluation function of Aleph is a ‘coverage’ function. It evaluates each rule r by

$$\text{coverage}(r) = p - n \tag{1}$$

where p is the number of positive examples covered by r, and n is the number of negative examples covered by r. It can be seen from the evaluation function that only the numbers of covered examples are used to justify the quality of a rule. Two different rules may cover different set of positive and negative examples of the same numbers. Distances between the covered examples can be used to differentiate between good and bad rules.

Our idea is that a rule covering a larger area in the space of the examples performs better or yields a higher accuracy than a rule with smaller coverage area. To show a rough estimation, we find distances between all pairs of covered positive examples, and calculate the average distance. We first calculate $C(r)$ that finds the set of all possible pairs of positive examples by a rule r.

$$C(r) = \{(x, y) \mid x \in P(r),\ y \in P(r),\ x \neq y\}, \tag{2}$$

where $P(r)$ is the set of positive examples covered by r. Then, we calculate the average distance from the set $C(r)$:

$$\overline{d}(r) = \frac{1}{|C(r)|} \sum_{(x,y)\in Cr} d(x, y), \tag{3}$$

where $d(x, y)$ is a distance between two examples x and y. In this paper, we apply the four-layer distance function defined in the previous section.

We propose two modifications of the coverage function, i.e.

1. We weight the coverage value by the value of $\overline{d}(r)$. Thus, a rule covering the same numbers of positive examples with higher average distance will be evaluated higher.

$$\text{coverage}_{mul}(r) = \overline{d}(r)(p - n) \tag{4}$$

2. We adjust the coverage value by adding $\overline{d}(r)$. This yields the similar effect as the previous modification.

$$\text{coverage}_{add}(r) = p - n + \overline{d}(r) \tag{5}$$

As stated above, Aleph starts from constructing a bottom clause from a positive example in order to constrain the search for rule construction. However, using different positive example as a seed for the bottom clause may result in different set of rules and different predictive performance. Aleph is implemented to construct select positive examples for bottom clause construction based on their order in the input file. The distances between the examples can be utilized to appropriately order the positive examples. This allow us to maximize the predictive performance of the outcome without trying all combinations of the orders. Our idea is that starting from construct rules at the center of the positive set may yield different performance from starting from the border of the set. We can use the eccentricity function to approximate the location of each positive example. The eccentricity function measures the distance from an example to the center of the dataset. It is defined as follows:

$$E_p(x) = \left(\frac{1}{n} \sum_{y \in X} d(x,y)^p \right)^{\frac{1}{p}} , \tag{6}$$

where $1 \leq p < +\infty$, X is the dataset, and n is the cardinality of X.

Moreover, the distance to the center of negative examples may also be useful information for ordering the positive examples. We therefore compute two eccentricity values for each positive example i.e. the eccentricity against the positive set and the eccentricity against the negative set. We also rank the positive examples according these two values in ascending and descending orders.

5 Experiments and Discussions

We evaluate our proposed evaluation functions and positive example ordering techniques on real-world datasets, i.e. Mutagenesis dataset [5], and the Alzheimer dataset [1]. The experiments are conducted using 10-fold cross validation technique. We vary the following setting for the evaluation function and the order of positive examples: (1) The evaluation function is set to be either coverage$_{mul}$ or coverage$_{add}$, (2) The eccentricity values used to order the positive examples are computed against either the positive set or the negative set, (3) The eccentricity values are ordered in either ascending or descending order. The experimental results are also compared with the results from Aleph without modification and κFOIL. The results are shown in Table 1.

Table 1. Prediction accuracies on real-world datasets using 10-fold cross validation method (figures in boldface font indicate the best accuracy in each dataset; $E^{\oplus}$ indicates the eccentricity against the positive set, and $E^{\ominus}$ is the eccentricity against the negative set; ▲ and ▼ indicate the ascending and descending order, respectively)

Method	Mutagenesis	Alzheimer amine
Aleph	73.4 ± 11.8	70.2 ± 7.3
κFOIL	77.0 ± 4.5	89.8 ± 5.7
coverage$_{add}$, $E^{\oplus\blacktriangle}$	81.4 ± 7.5	75.8 ± 6.0
coverage$_{add}$, $E^{\oplus\blacktriangledown}$	81.9 ± 7.1	76.7 ± 6.3
coverage$_{add}$, $E^{\ominus\blacktriangle}$	81.4 ± 7.5	77.3 ± 7.1
coverage$_{add}$, $E^{\ominus\blacktriangledown}$	81.9 ± 7.1	77.4 ± 6.5
coverage$_{mul}$, $E^{\oplus\blacktriangle}$	78.2 ± 9.6	78.1 ± 7.3
coverage$_{mul}$, $E^{\oplus\blacktriangledown}$	80.3 ± 5.5	77.6 ± 6.0
coverage$_{mul}$, $E^{\ominus\blacktriangle}$	78.2 ± 9.6	77.7 ± 7.1
coverage$_{mul}$, $E^{\ominus\blacktriangledown}$	80.3 ± 5.5	77.4 ± 6.3

Compared to the conventional Aleph, all settings of the proposed technique can obtain the higher accuracies. The proposed technique performs better than κFOIL on the Mutagenesis dataset, but it still performs worse than κFOIL on the Alzheimer dataset. The coverage$_{add}$ function performs better than coverage$_{mul}$ on the Mutagenesis dataset, but both functions performs similarly on the Alzheimer dataset. Moreover, the eccentricity against the positive and negative sets and the order of positive examples do not affect the prediction accuracies. These issues need further investigation.

Fig 1 shows the rules with the highest positive example coverage constructed by the proposed technique and the conventional Aleph. This can be seen that the proposed technique can obtain the rules with higher coverage.

```
The Proposed Technique
[Rule 6] [Pos cover = 59 Neg cover = 5]
active(A) :- atm(A,B,c,27,C), lteq(C,0.005).
[Rule 1] [Pos cover = 55 Neg cover = 2]
active(A) :- atm(A,B,c,27,C), lteq(C,-0.076).
[Rule 5] [Pos cover = 41 Neg cover = 5]
active(A) :- atm(A,B,c,29,C), gteq(C,0.004).
```

```
Aleph
[Rule 3] [Pos cover = 56 Neg cover = 5]
active(A) :- atm(A,B,c,27,C), lteq(C,-0.054).
[Rule 4] [Pos cover = 32 Neg cover = 5]
active(A) :- atm(A,B,c,22,C), atm(A,D,c,10,E), bond(A,B,D,1).
[Rule 1] [Pos cover = 28 Neg cover = 0]
active(A) :- atm(A,B,c,27,C), lteq(C,-0.087).
```

Fig. 1. Top Rules obtained from the proposed technique and Aleph

6 Conclusion

We propose new evaluation functions for ILP systems based on distances between examples in the dataset. We also present a technique to order the positive examples based on their eccentricity values in order to obtain the rules with higher performance. This work is an attempt to investigate a link between generalization and distance between FOL objects.

7 Acknowledgement

This work was performed under the Cooperative Research Program of "Network Joint Research Center for Materials and Devices" at the Institute of Scientific and Industrial Research, Osaka University, Japan. It was also supported by the Thammasat University Research Fund, Thailand.

References

1. King, R., Srinivasan, A., Sternberg, M.: Relating chemical activity to structure: an examination of ilp successes. New Generation Computing 13(2, 4), 411–433
2. Landwehr, N., Passerini, A., Raedt, L.D., Frasconi, P.: kfoil: Learning simple relational kernels. In: Proceedings of the 21st National Conference on Artificial Intelligence. pp. 389–394 (2006)
3. Muggleton, S.: Inverse entailment and progol. New Generation Computing 13(3), 245–286 (1995), `http://dx.doi.org/10.1007/BF03037227`
4. Muggleton, S.: Inductive logic programming: issues, results and the challenge of learning language in logic. Artificial Intelligence 114(1), 283–296 (1999)
5. Srinivasan, A., Muggleton, S., King, R., Sternberg, M.: Theories for mutagenicity: a study of first-order and feature-based induction. Artificial Intelligence 85, 277–299
6. Srinivasan, A.: The Aleph Manual (2007), `http://www.cs.ox.ac.uk/activities/machinelearning/Aleph/aleph`