An Inductive Logic Programming Approach for Feature-Range Discovery

Daniel Cyrus, Dany Varghese, and Alireza Tamaddoni-Nezhad

University of Surrey GU2 7XH, UK
{d.cyrus, dany.varghese, a.tamaddoni-nezhad}@surrey.ac.uk

Abstract. In this paper, we present NumLog, an Inductive Logic Programming (ILP) system designed for feature range discovery. NumLog generates quantitative rules with clear confidence bounds to discover feature-range values from examples. Our approach focuses on generating rules with minimal complexity from numerical values, ensuring the assessment of methods that could impact accuracy and comprehensibility. Traditional ILP systems, especially those intersecting with computer vision, struggle with numerical data. This convergence presents unique challenges, often hindering the generation of meaningful insights due to the limited capabilities of conventional ILP systems to handle numerical values. NumLog stands out by incorporating an advanced range discovery mechanism that generates low-complexity rules while maintaining high accuracy and comprehensibility. This enhancement significantly improves interpretability, promoting more effective human-machine learning collaboration. We compare NumLog with the state-of-the-art ILP systems such as NumSynth and Aleph and conduct comprehensive experiments on several datasets. We evaluated our approach by measuring accuracy, precision, F1 score, and rule complexity to demonstrate the effectiveness of the methodology.

Keywords: Human-Machine Learning · Inductive Logic Programming · Numerical Reasoning · Explainable AI.

1 Introduction

Inductive logic programming (ILP), as introduced by Muggleton [15], represents an explainable machine learning approach capable of deriving comprehensible rules from a set of examples. These rules are expressed as logical statements, symbolic and close to natural language, making them relatable and directly readable, which closely mirrors human reasoning processes. Additionally, ILP can seamlessly incorporate existing domain knowledge into its learning process, enhancing learning efficiency and the clarity of how new knowledge is derived. This integration is particularly useful in scenarios where data is scarce or expensive to collect, such as in domains with privacy concerns, enabling ILP to generalise effectively from smaller datasets [21]. The transparency and clarity of the models generated by ILP not only foster trust among users and stakeholders

- crucial in sensitive areas like healthcare, finance, and legal advising - but also support ethical AI practices. Moreover, ILP's capability to discover new patterns and regularities in data allows it to contribute to scientific and academic discovery [20], proposing new hypotheses and logical rules that may not have been previously considered.

In the field of Inductive Logic Programming (ILP), certain challenges arise when the system attempts to learn rules involving numerical values from continuous domains. ILP systems excel at handling discrete data and symbolic information, but often struggle with continuous or numerical data, especially from infinite domains [3, 4, 22]. This limitation stems from the difficulty in formulating generalisable logical rules that can accurately represent and make predictions based on such data.

To tackle the limitations inherent in existing ILP systems, particularly their struggle with handling numerical values from infinite domains, we introduce a novel approach that enhances their capacity to identify and process numeric value ranges. We implement this appraoch in a ILP system called NumLog[1]. Our methodology extends the conventional range of positive and negative examples in ILP systems, thereby enabling the inclusion of a broader spectrum of values from both positive and negative domains. Central to our strategy is the prioritisation of retaining the highest and lowest numerical values within binned data, a tactic aimed at preserving essential information about the extremities in the dataset. This is complemented by the integration of specialised predicates into the hypothesis space of the ILP system. These predicates are designed to capture intricate relationships and patterns in the data, significantly increasing the expressiveness and flexibility of the model. The enriched hypothesis space allows the system to engage more effectively with diverse numerical data distributions, thereby enhancing decision-making capabilities and ensuring robust performance across a variety of scenarios. Through this integrated approach, our ILP system is better equipped to manage and learn from complex numerical inputs, marking a significant advancement in the field.

1.1 Background

Inductive Logic Programming (ILP) [15] is a branch of artificial intelligence focused on deriving hypothesised predicate definitions. ILP uniquely employs logic programs to represent examples, background knowledge, and hypotheses uniformly. Unlike most other machine learning approaches, ILP stands out due to its use of a highly expressive representation language and its capability to utilise logically encoded background knowledge [16]. Finn et. al. [7] presented a case study of Pharmacophore Discovery. Their approach was to generate rules to identify potential pharmacophores. Although the method employs numerical reasoning to generate rules, it lacks the ability to identify the range of numerical values.

[1] NumLog:**Num**erical reasoning using **Log**ic programming

Range Discovery [5,10,14] is a method used to find the maximum and minimum values for each group of positive examples in relation to the nearest negative examples. Range discovery is mostly used to adjust a threshold between positive and negative examples.

2 Related Work

Quantitative Classification based on Associations (QCBA) [13] is a method that represents rules derived from pattern discovery. Its goal is to recover missing information in quantitative yet sparse data. While the approach includes pruning, the resulting rules can still become complex. Additionally, QCBA discretises continuous values, which can lead to a loss of information. In contrast, NumLog addresses both of these issues effectively. TILDE [2] discretises continuous values during the learning process by splitting numerical data based on an optimal criterion. However, this approach does not always align with the true underlying structure of the data. NumLog addresses this issue by retaining the best thresholds between data points for improved accuracy. Aleph Lazy evaluation procedure [17,18] is a technique on numerical reasoning. Lazy evaluation is a programming technique where an expression is not evaluated until its value is actually needed. In Aleph, every definition used in lazy evaluation operates independently, thereby hindering its ability to infer hypotheses involving multiple literals that necessitate shared variables for lazy evaluation, such as upper and lower bounds for a single variable. NUMSYNTH [12] employs satisfiability modulo theories solvers (SMT) to efficiently learn programs with numerical values. The method can identify numerical values in linear arithmetic fragments, such as real difference logic, and from infinite domains, such as real numbers or integers. NUMSYNTH is a library for Popper [4] that leverages the Popper system and lazy evaluation for numerical reasoning. NUMSYNTH's key distinction from NumLog is that it avoids discretisation, but this can lead to difficulties in generalizing to unseen data.

3 Overview of the Methodology

This section provides an overview of our methodology, highlighting the key techniques and processes used to achieve our research objectives. Our methodology focuses on threshold analysis and generalisation of numerical values.

FOL language First-Order logic (FOL) includes predicates, quantifiers, and logical connectives to express complex statements about objects and their relationships. Each predicate is associated a certain number of variables/terms. The number of variables/terms associated with a predicate is called $Arity(n)$. If a predicate $\mathcal{P}$ is associated with two entities, we will represent it as $\mathcal{P}/2$. We define a FOL language as an alphabet consisting of sets of predicate symbols $\mathcal{P} = \{P, Q, R, ...\}$, function symbols $\mathcal{F} = \{f, g, h, ...\}$, constants $\mathcal{C} \subseteq \mathcal{F} = \{a, b, c, ...\}$, variables $\mathcal{V} = \{x, y, z, ...\}$, logical connectors $\neg, \wedge, \vee, \rightarrow, \Leftrightarrow$, quantifiers $\exists, \forall$ and punctuation symbols . , (and).

Numerical Reasoning [6, 19] is an approach that allows individuals to analyse and draw conclusions from numerical data, facilitating informed decision-making. The NumLog system employs a dedicated learning algorithm that exclusively explores numerical values. The rules generated by this process are appended to the background knowledge for subsequent program searches using multi-class search. Algorithm 1 delineates the NumLog learning process and its focus on numerical search.

The final hypothesis produced by NumLog is a set of rules structured as $P \leftarrow Q \wedge R$. These rules are represented as

$$r(A) \leftarrow numerical\_feature(A, B), numerical\_range(B, C)$$

, where A is the input argument representing an instance with numerical feature B, and C is a constant that indicates the range of the numerical value B.

3.1 Data Preprocessing

In the context of data preprocessing, our method involves sorting values in sequence and removing redundant entries. The system then collects all numerical values and categorise by feature type from background knowledge across all positive (E^{+}) and negative (E^{-}) examples into a universal set, which we refer to as U_i.

$$\text{Let } E^{+} = \{p(x_1^{+}), p(x_2^{+}), ..., p(x_n^{+})\}$$
$$\text{Let } E^{-} = \{p(x_1^{-}), p(x_2^{-}), ..., p(x_n^{-})\}$$

$$\begin{gathered}\forall p(x^{+}) \exists \{\mathrm{f}_1(x^{+}, v_1^{+}), \mathrm{f}_2(x^{+}, v_2^{+}), ..., \mathrm{f}_n(x^{+}, v_n^{+})\} \\ \forall p(x^{-}) \exists \{\mathrm{f}_1(x^{-}, v_1^{-}), \mathrm{f}_2(x^{-}, v_2^{-}), ..., \mathrm{f}_n(x^{-}, v_n^{-})\}\end{gathered} \tag{1}$$

$$U_i = \{v_i^{1\pm}, v_i^{2\pm}, ..., v_i^{n\pm}\} \text{ such that } v_i^{1\pm} \leq v_i^{2\pm} \leq ... \leq v_i^{n\pm}$$

Each example p consists of a value x that is associated with a set of features f, where each feature has a numerical value v. All numerical values in U are defined by binning them across the entire range of negative values, effectively grouping positive feature values according to their proximity to negative feature values, resulting in the set we define as $\bar{U}$.

$$\begin{gathered}\text{Let } \bar{U_i} = \{G_i^1, G_i^2, ..., G_i^n\} \text{ such that } v^{-} \nexists G_i \text{ and } G_i^j \subset U_i \\ G_i = \{v_i^{j_1}, v_i^{j_2}, ..., v_i^{j_n}\} \text{ where } v_i^{j_n} < v_i^{j+1_1}\end{gathered} \tag{2}$$

Informal expression The equation 1 and 2 are utilised to process all values for each feature type. For instance, one feature type includes the positive set {1,2,3,6,7} and negative set {4,5}. These sets are combined and sorted into a set U, thus, $U = \{1, 2, 3, 4, 5, 6, 7\}$. The system then groups them by removing the negative values, resulting in $\bar{U} = \{\{1, 2, 3\}, \{6, 7\}\}$.

3.2 Threshold Analysis

The system supplements each group of feature values G_i by adding additional values. Discrete data can exhibit extensive distribution, leading to the possibility of encountering positive values during assessment. NumLog has the capacity to learn from data with the inclusion of a tolerance value.

$$\begin{aligned} \mu_i &= \frac{1}{n} \Sigma_{i=1}^{n} G_i \\ \sigma_i &= \sqrt{\frac{\Sigma(G_i^j - \mu_i)^2}{N_i}} \\ pdf_i(G_i) &= \frac{1}{\sqrt{2\pi\delta_i^2}} \exp^{(-\frac{(v_i^j-\mu_i)^2}{2\delta_i^2})} \end{aligned} \tag{3}$$

The equation 3 calculates the average μ, standard deviation σ and probability density function *pdf* for each group G. The initial value r_0 is the estimated threshold value between each group G, determined by calculating the average μ for each pair of groups:

$$r_0 = \frac{\mu_i + \mu_{i+1}}{2} \tag{4}$$

Final threshold value calculates by finding the root value R of pair functions *pdf* and initial value r_0, where $R = \{r \in \mathbb{R} | pdf(r) = 0\}$ and r is the value where two *pdf*s are equal:

$$\begin{aligned} r_k &= r_0 + k\epsilon \\ pdf_i(r_k) - pdf_{i+1}(r_k) &= 0 \end{aligned} \tag{5}$$

In the equation 5, the value k represents an incremental value that starts from 1 and continues until the root of the function reaches 0. The system determines the threshold point that separates each group G, see Figure 1.

Informal expression For instance, if a bin of positive data consists of numbers *1.2, 1.4, 1.6*, and negative examples *1.7,1.8* and *1.95*, the analysis introduces a new value, *1.65*, as a threshold. Consequently, the positive group now includes *1.2, 1.4, 1.6*, and *1.65*. thus a group of positive example G_1 consists of *{1.2,1.4,1.6,1.65}*. The system then concentrates exclusively on the upper and lower bounds of each bin, incorporating these bounds directly as literals into the new dataset. Thus the first group of positive feature value includes *{1.2,1.65}*,

3.3 Generalisation on Numerical Values

Generalizing numerical values entails identifying broader patterns or principles that apply to a dataset. This process relies on analyzing the dataset's patterns and distribution [1, 8]. We combine each group of distributions by assessing the overlap of their normal distributions and the quantity of data in each group [9, 11]. Calculating $x_i \sim \mathcal{N}(\mu, \delta^2)$ indicates that the data points x_i follow an approximate normal distribution $\mathcal{N}$ with mean μ and variance δ^2, implying that

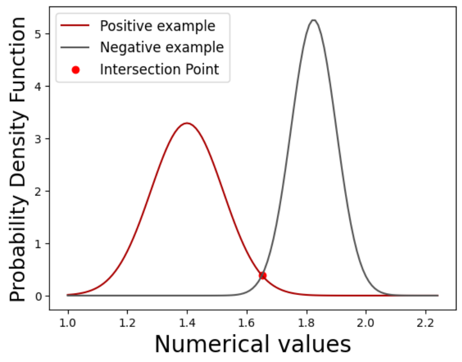

Fig. 1. The Figure illustrates an instance of a threshold separating two sets of positive and negative feature values. Positive values fall within the range of *1.2-1.6*, while negatives are within the range of *1.7-1.95*. The threshold value is *1.65514*.

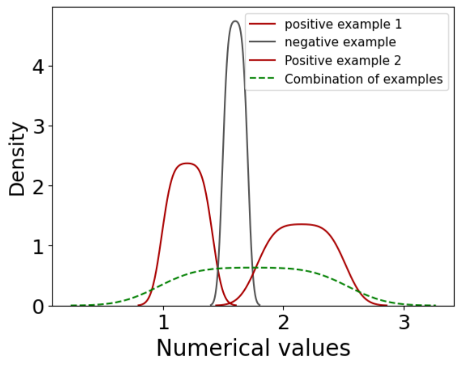

Fig. 2. Combining numerical feature values based on their distributions. The graph illustrates how closely clustered negative distribution can be merged with other positive distributions.

the data tends to cluster around the mean with a narrow range of values, see Figure 2. By measuring the overlap of normal distributions and the quantity of data in each group, we can effectively combine groups of distributions. This generalisation process reduces the number of groups by merging similar ones. The resulting consolidated group can then be utilised for ILP approaches, enhancing the efficiency and accuracy of the analysis.

$$\begin{aligned} \mathcal{T} &= \frac{(max(U_i) - min(U_i)}{\epsilon} \\ C' &= \frac{(\frac{Max(G_i^n) - min(G_i^n)}{\epsilon})}{\mathcal{T}} \\ l &= \frac{|U_i|}{|G_i^n|} \end{aligned} \tag{6}$$

Where $\mathcal{T}$ denotes the normalised size of all feature values and C' is the normalised feature size for specific group of values. We define the combination value $\mathcal{C}$ to determine if a group of values can be combined by verifying if $\mathcal{C} < C'$ and $\mathcal{C} \geq l$.

4 NumLog Algorithm

NumLog accepts a tuple as its input, $(E^+, E^-, \mathcal{M}, \mathcal{C})$, where E^+ and E^- are positive and negative examples respectively. The value $\mathcal{M}$ defines maximum number of clauses, generated in final hypothesis and $\mathcal{C}$ is the combination value (Refer 3.3). The epsilon value ϵ is used for three purposes: first, to determine the density of two points in a normal distribution; second, to set the minimum threshold between two groups of examples; and third, to represent the allowable error in the approximation of the root, defined as $|f(x)| < \epsilon$. The ϵ value is adjusted on each learning iteration using minimum gap δ between samples by the equation $\epsilon \leftarrow min(\epsilon, \delta_i)$. The learning process of NumLog is illustrated in Algorithm 1.

4.1 Background Knowledge

In this section, we define the background knowledge and hypothesis space used in NumLog. We utilise specific functions and notations to represent constraints and relationships between input values and constants. Let A denote the input value and B and C denote the constants. We define three primary functions to capture the relationships between parameters:

Table 1. Function Definitions

Function	**Mathematical definition**	**Description**
leq(A, B)	$A \leq B$	A is less than or equal to B
geq(A, B)	$A \geq B$	A is greater than or equal to B
inRange$(A, B \pm \mathrm{D})$	$B - D \leq A \leq B + D$	A is within the range

4.2 Complexity of Rules

The complexity of a rule can be defined by the user, depending on the specific requirements of the application. The complexity might be influenced by the

Algorithm 1: NumLog (E^+, E^-, $\mathcal{M}$, $\mathcal{C}$)

```
Data: Positive examples $E^+$, negative examples $E^-$, max number of clauses
      $\mathcal{M}$, and merging value $\mathcal{C}$
Result: Hypothesis ($\mathcal{H}$) to explain numerical ranges
$H \leftarrow \varnothing$;
sorted($E^+$), sorted($E^-$);
$U \leftarrow append(E^+, E^-)$;
$\epsilon \leftarrow 100$ ;                         // Initial value for epsilon
for $i = 0$ to $n - 1$ do
    $\delta_i \leftarrow U[i+1] - U[i]$;
    $\epsilon \leftarrow \min(\epsilon, \delta_i)$;
end
$\bar{U} \leftarrow U - (U \cap E^-)$ ;      // Group examples by removing negative values
foreach $G \in \bar{U}$ do
    $G_i \leftarrow$ Abduce values using Eq. 5 and $\epsilon$;
end
$\bar{U} \leftarrow$ Generalize values from $\bar{U}$ using Eq. 6 and $\mathcal{C}$;
$\bar{H} \leftarrow [BK, \bar{G}_i]$ ; // For each group $\bar{G}_i$ in $\bar{U}$, generate literal using BK
Generate all possible rules ($\mathcal{R}$) from $\bar{H}$ using Eq. 7;
foreach $\mathcal{R}$ do
    prove($\mathcal{R}_i$);
    if accuracy($\mathcal{R}_i$) is the best accuracy then
        Add $\mathcal{R}_i$ to $h$;
    end
end
Find the best hypothesis with respect to $\mathcal{C}$;
Evaluate each $\mathcal{R}_i$ from $h$;
Select the best hypothesis $\mathcal{H}$ from $h$;
return $\mathcal{H}$;
```

number of conditions, the types of relationships, and the logical operations used to combine these conditions.

$$\text{Combinations} = \bigcup_{N=0}^{\text{MaxLength}} \text{combinations}(N, \text{List}) \tag{7}$$

$\bigcup$ represents the union of all these sets for N ranging from 0 to MaxLength.

4.3 Hypothesis Space

The hypothesis space is defined by the combination of constraints and conditions that can be applied to the input values. We denote the hypothesis space by $\mathcal{H}$, which is a combination of the constraints defined above. The set of all possible hypotheses can be represented as:

$$\mathcal{H} = \{h \mid h = \text{feature}/2 \land \text{leq}/2 \lor \text{geq}/2 \lor \text{inRange}/2\} \tag{8}$$

The specific hypothesis chosen from this space can be tailored by adjusting the parameters as per the requirements of the problem. By defining the hypothesis space in this structured manner, we can systematically explore the possible rules and constraints that apply to our input values, ensuring a comprehensive analysis of the problem domain.

5 Experiments

In our study, we apply our methods to the breast cancer [23] and pharmacophores datasets [12]. The breast cancer dataset is a well-established benchmark for classification tasks. It includes a diverse array of numerical features obtained from real-world medical diagnoses, offering a solid basis for testing algorithms that emphasize numerical reasoning and classification. Moreover, the dataset's balanced distribution of positive and negative cases, combined with its straightforward binary outcome, makes it an excellent platform for assessing the performance of our system in processing medical data and generating interpretable hypotheses.

Pharmacophores are structural features in a molecule that interact with a receptor site, influencing biological activity. A pharmacophore model identifies the spatial arrangement of these critical features, aiding drug discovery by pinpointing compounds with similar structures and potential effects. In 3D, it highlights essential attributes like functional groups and hydrogen bond donors or acceptors, with precise distances between them crucial for effective binding.

Our approach improves Inductive Logic Programming (ILP) systems by enabling more effective processing and learning from numerical data in these interactions. The experiments are structured to assess the effectiveness of our proposed ILP-based method, as outlined in the previous section. To achieve this, we test several null hypotheses, which serve as baseline assumptions, asserting that our ILP-based approach yields no significant effects or improvements. By evaluating these null hypotheses, we aim to rigorously assess the validity and performance of our method, ensuring a thorough evaluation of the proposed approach.

Null Hypothesis 1 NumLog is unable to effectively learn from numerical values using ILP and generalisation approaches.

Null Hypothesis 2 Expanding numerical examples by incorporating new values through threshold analysis does not yield significant improvements in accuracy.

Null Hypothesis 3 The clause inRange/2, alongside leq/2 and geq/2, does not contribute to reducing the complexity of the rule.

In Our first experiment we usilised breast cancer dataset including *569* records. We select *100* random samples for the test set, comprising *50* positive examples and *50* negative examples. Our training set starts with *0* samples, increasing by *20* samples each time up to *100* examples, in addition to a final set with *200*

samples, see figure 3.
Listing 1.1 presents a generalised sample rule indicating that entity A is classified as having cancer if its worst fractal dimension falls within the range of approximately *0.0904* and its worst area is less than *1032.30447*.

Listing 1.1. Learned rule generated by NumLog from iris breast cancer dataset

```
cancer(A) ← worst_fractal_dimension(A,B),
                inRange(B,0.09048 ± 0.03544),
                worst_area(A,C),leq(C,1032.30447).
```

Table 2. We claimed that our approach generate lowest complexity on rule generating. The table shows the rule complexity for either number of literals and number of disjoint rules.

Number of Examples	Average Number of Rules		
	Aleph	**NumSynth**	**NumLog**
20	2	1.2	**1**
40	3.2	1.8	**1**
60	4	2.8	**1**
80	6.2	6	**1.6**
100	6.2	5.6	**1.6**
200	10.4	6.4	**1.8**

In our second experiment, we utilized the Pharmacophore dataset from the study by C. Hocquette et al. [12], which is explained and structured according to the work of P. Finn et al. [7]. We enriched the data with additional grounded background knowledge, particularly incorporating distances between each feature. This augmented dataset served as the basis for constructing pharmacophores, which are spatial arrangements of molecular features.

Listing 1.2 displays a generalised rule suggesting that atom A exhibits the trait of a hydrogen bond acceptor, in conjunction with zinc, at a distance of *14.2494* units, with a minimum and maximum range of *0.212* units.

Listing 1.2. Learned rule generated by NumLog from pharma dataset

```
active(A) ← hacc_zincsite_distance(A,B),
             inRange(B,  14.249 ± 0.212).
```

We assess the accuracy of our work using NumSynth and Aleph, following the same training set approach as outlined in [12]. Specifically, we use *60* negative and *60* positive examples. The results are presented in Table 3. As indicated in listing 1.2 and 1.1, NumLog is capable of learning from numerical values, and the inclusion of a single clause inRange aids in reducing the complexity of the rule. Consequently, Null Hypotheses 1 and 3 are refuted. Furthermore, Table 3 and Figure 3 illustrates that our system achieves commendable accuracy in learning. Therefore, Null Hypothesis 2 is rejected.

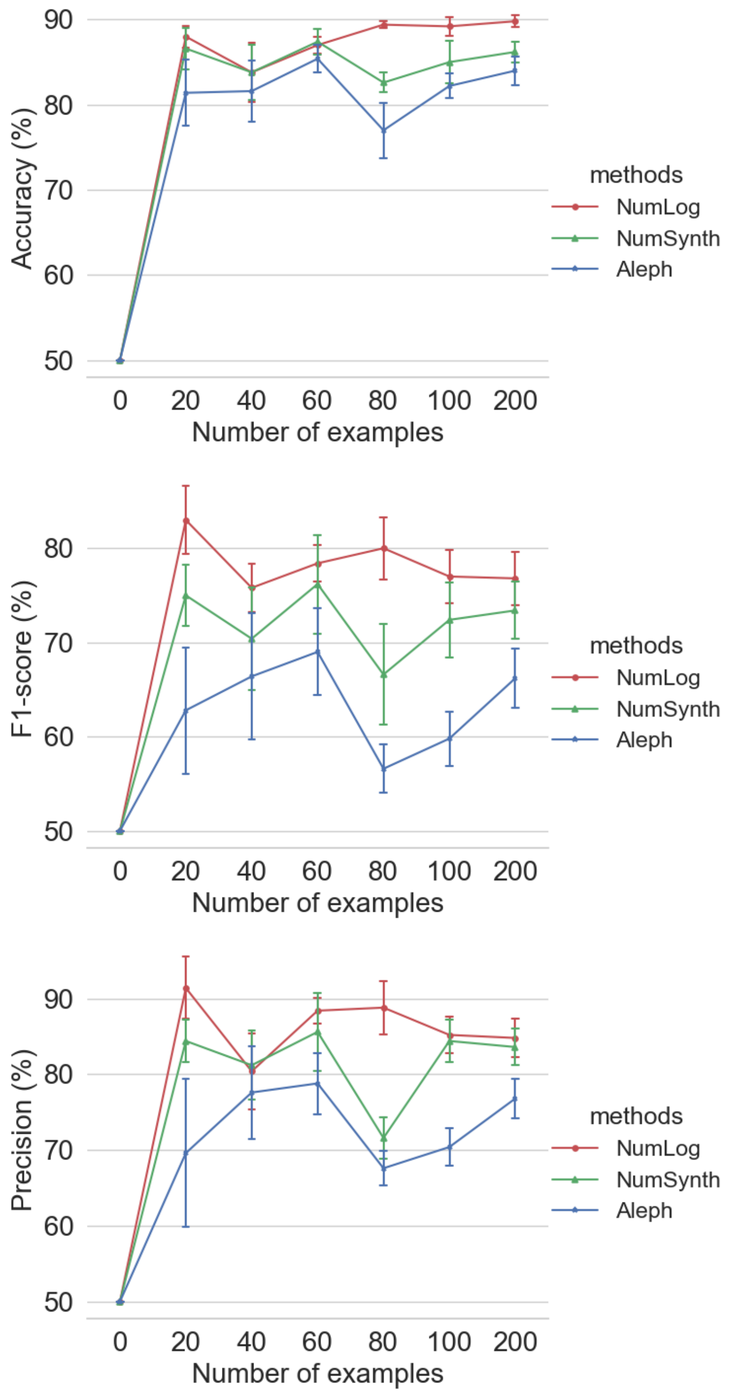

Fig. 3. Comparison of Accuracy, F1 score and Precision metrics across different models.

Table 3. Rounded accuracy to integer values with standard error. Aleph and Numsynth accuracy value given from [12]

Tasks	**Aleph**	**NumSynth**	**NumLog**
pharma1	82 ± 1	99 ± 0	99 ± 0
pharma2	83 ± 1	95 ± 1	**99 ± 0**
pharma3	81 ± 1	98 ± 1	**99 ± 0**
pharma4	76 ± 1	92 ± 1	**95 ± 2**

5.1 Results and Discussion

The ultimate goal of NumLog is to reduce the complexity of the rules it generates. This entails choosing the most straightforward set of rules that sufficiently describe the data, avoiding overfitting and unnecessary intricacies. Conversely, NumLog places a high value on making the generated rules easy to interpret and understand, see Table 2. Although NumLog demonstrates proficient learning capabilities with numerical values, even in the presence of noise and sparse data, its effectiveness may be compromised when dealing with datasets characterised by high levels of noise and sparsity. The current iteration of NumLog is specifically tailored for learning within first-order logic using numerical data. To extend its applicability to higher-order logic programming and datasets lacking grounded background knowledge, NumLog may need to be adapted and integrated with other ILP systems. This adaptation would enhance its robustness and effectiveness in a broader range of applications. By organising and abducting values within each group, the system generates all potential hypotheses. However, due to computational constraints, NumLog aims to identify the most accurate rule, which may result in a lack of completeness and consistency.

6 Conclusion

In conclusion, this paper has presented an innovative approach to numerical reasoning through the application of NumLog coupled with threshold analysis and diverse sets of positive and negative examples. By leveraging NumLog's capabilities, we have demonstrated the effectiveness of this methodology in handling numerical data and extracting meaningful insights. Through rigorous experimentation and analysis, we have showcased how the careful selection of thresholds and the incorporation of varied examples enrich the learning process and improve the accuracy of classification tasks. Furthermore, by addressing the challenges of rule complexity and redundant information, we have enhanced the efficiency and practicality of the proposed framework. Overall, this research contributes to the advancement of numerical reasoning techniques and underscores the potential of integrating symbolic and statistical methods for tackling real-world problems in diverse domains.

7 Future Work

In future work, we plan to enhance the NumLog system by incorporating Support Vector Machine Inductive Logic Programming (SVILP). This integration aims to leverage the strengths of SVILP in handling both classification and regression tasks, using declarative background knowledge to provide learning bias. Specifically, we will focus on comparing the performance of NumLog with SVILP against several established approaches. We will evaluate NumLog's performance against the histogram-based binning method introduced by D. Varghese [20, 22]. This method organises numerical data into bins or intervals, enabling the identification of patterns, outliers, and overall data distribution. By comparing these techniques, we aim to assess the effectiveness of NumLog in handling continuous numerical data. We will integrate SVILP, as described by Muggleton [16], into NumLog. SVILP combines SVM and ILP, providing a learning bias through declarative background knowledge. We will test NumLog's ability to handle numerical data within this framework and compare its performance to other SVM-based methods. The current version of NumLog generates deterministic rules. In the future, we plan to incorporate a feature that allows the system to learn from features with multiple numerical values, enabling it to produce non-deterministic rules.

Supplementary Material We have provided supplementary material on GitHub https://github.com/hmlr-lab/NumericalReasoning to accompany NumLog. Codes are implemented in Prolog, offering a valuable resource for researchers and practitioners interested in exploring and extending our work. This repository contains comprehensive documentation, including detailed explanations of NumLog's functionalities, usage examples, and guidelines for integration into other projects.

Acknowledgments. The first author acknowledges the PhD scholarship from the EPSRC and the University of Surrey. The authors acknowledge the EPSRC grant on human-machine learning of ambiguities. The third author acknowledges the EPSRC Network Plus grant on Human-Like Computing (HLC). We acknowledge Dr. Céline Hocquette for her comments and feedback on this work.

References

1. Bishop, C.M.: Pattern recognition and machine learning by Christopher M. Bishop. Springer Science+ Business Media, LLC (2006)
2. Blockeel, H., De Raedt, L.: Lookahead and discretization in ilp. In: International Conference on Inductive Logic Programming. pp. 77–84. Springer (1997)
3. Corapi, D., Russo, A., Lupu, E.: Inductive logic programming in answer set programming. In: International conference on inductive logic programming. pp. 91–97. Springer (2011)
4. Cropper, A., Morel, R.: Learning programs by learning from failures. Machine Learning **110**(4), 801–856 (2021)

5. Danbold, F., Unzueta, M.M.: Drawing the diversity line: Numerical thresholds of diversity vary by group status. Journal of Personality and Social Psychology **118**(2), 283 (2020)
6. Demšar, D., Gams, M.: Implementing numerical reasoning in ilp. The IPSI BgD Transactions on Internet Research p. 40 (2006)
7. Finn, P., Muggleton, S., Page, D., Srinivasan, A.: Pharmacophore discovery using the inductive logic programming system progol. Machine Learning **30**, 241–270 (1998)
8. Friedman, J.: The elements of statistical learning: Data mining, inference, and prediction. (No Title) (2009)
9. Grice, J.W., Barrett, P.T.: A note on cohen's overlapping proportions of normal distributions. Psychological Reports **115**(3), 741–747 (2014)
10. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer Science & Business Media (2009)
11. Hill, T.P., Miller, J.: How to combine independent data sets for the same quantity. Chaos: An Interdisciplinary Journal of Nonlinear Science **21**(3) (2011)
12. Hocquette, C., Cropper, A.: Relational program synthesis with numerical reasoning. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 37, pp. 6425–6433 (2023)
13. Kliegr, T., Izquierdo, E.: Qcba: improving rule classifiers learned from quantitative data by recovering information lost by discretisation. Applied Intelligence **53**(18), 20797–20827 (2023)
14. Morrison, J.A., Shamos, M.H.: Intersection of normal distributions: Theory and applications. Journal of Statistical Theory and Applications **3**(2), 123–145 (2005)
15. Muggleton, S.: Inductive logic programming. New generation computing **8**, 295–318 (1991)
16. Muggleton, S., Lodhi, H., Amini, A., Sternberg, M.J.: Support vector inductive logic programming. In: Discovery Science: 8th International Conference, DS 2005, Singapore, October 8–11, 2005. Proceedings 8. pp. 163–175. Springer (2005)
17. Srinivasan, A.: The aleph manual (2001)
18. Srinivasan, A., Camacho, R.: Experiments in numerical reasoning with inductive logic programming. Journal of Logic Programming (1997)
19. Srinivasan, A., Camacho, R.: Numerical reasoning with an ilp system capable of lazy evaluation and customised search. The Journal of Logic Programming **40**(2-3), 185–213 (1999)
20. Varghese, D., Barroso-Bergada, D., Bohan, D.A., Tamaddoni-Nezhad, A.: Efficient abductive learning of microbial interactions using meta inverse entailment. In: International Conference on Inductive Logic Programming. pp. 127–141. Springer (2022)
21. Varghese, D., Bauer, R., Baxter-Beard, D., Muggleton, S., Tamaddoni-Nezhad, A.: Human-like rule learning from images using one-shot hypothesis derivation. In: International Conference on Inductive Logic Programming. pp. 234–250 (2021)
22. Varghese, D., Bauer, R., Tamaddoni-Nezhad, A.: Few-shot learning of diagnostic rules for neurodegenerative diseases using inductive logic programming. In: International Conference on Inductive Logic Programming. pp. 109–123 (2023)
23. Wolberg, William, M.O.S.N., Street, W.: Breast Cancer Wisconsin (Diagnostic). UCI Machine Learning Repository (1995), DOI: https://doi.org/10.24432/C5DW2B