TPLP: Page 1–49. © The Author(s), 2026. Published by Cambridge University Press. This is an Open Access article, distributed under the terms of the Creative Commons Attribution licence (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted re-use, distribution and reproduction, provided the original article is properly cited.

doi:10.1017/S1471068426100441

Explaining Neural Networks in Preference Learning: A Post Hoc Inductive Logic Programming Approach

DANIELE FOSSEMÒ
Department of Information Engineering, Computer Science and Mathematics, University of L'Aquila, Italy
(*e-mail:* daniele.fossemo@graduate.univaq.it)

FILIPPO MIGNOSI
Department of Information Engineering, Computer Science and Mathematics, University of L'Aquila, Italy
(*e-mail:* filippo.mignosi@univaq.it)

GIUSEPPE PLACIDI
Department of Life, Health and Environmental Sciences, University of L'Aquila, Italy
(*e-mail:* giuseppe.placidi@univaq.it)

LUCA RAGGIOLI
Department of Electrical Engineering and Information Technologies, University of Naples Federico II, Italy
(*e-mail:* luca.raggioli@unina.it)

MATTEO SPEZIALETTI
Department of Information Engineering, Computer Science and Mathematics, University of L'Aquila, Italy
(*e-mail:* matteo.spezialetti@univaq.it)

FABIO AURELIO D'ASARO
Department of Human Sciences, University of Verona, Italy
(*e-mail:* fabioaurelio.dasaro@univr.it)

submitted 13 November 2024 UTC; revised 5 December 2025 UTC; accepted 15 April 2026 UTC

Abstract

In this paper, we propose using Learning from Answer Sets to approximate black-box models, such as Neural Networks (NN), in the specific case of learning user preferences. We specifically explore the use of ILASP (Inductive Learning of Answer Set Programs) to approximate preference learning systems through weak constraints. We have created a dataset on user preferences over a set of recipes, which is used to train the NNs that we aim to approximate with ILASP. Our experiments investigate ILASP both as a global and a local approximator of the NNs. These experiments address the challenge of approximating NNs working on increasingly

https://doi.org/10.1017/S1471068426100441 Published online by Cambridge University Press

high-dimensional feature spaces while achieving appropriate fidelity on the target model and limiting the increase in computational time. To handle this challenge, we propose a preprocessing step that exploits Principal Component Analysis to reduce the dataset's dimensionality while keeping our explanations transparent.

Under consideration for publication in Theory and Practice of Logic Programming (TPLP).

KEYWORDS: explainable AI, preference learning, learning from answer sets, ILASP, principal component analysis

1 Introduction

In recent years *Machine Learning* (ML) algorithms have become pervasive in our daily life. This is thanks to the extensive research and results carried out over time on these methods, covering very diverse application areas, such as robotic and drone navigation (Soori *et al.* 2023), medical imaging (Rajpurkar and Lungren 2023), legal reasoning (Zhang and Meng 2024), and many other domains. However, its recent developments have highlighted the need to better understand and explain black-box ML models (Cath *et al.* 2018; Bostrom and Yudkowsky 2018). In fact, although these models prove very accurate in many application domains, their internal structures are exceedingly hard to understand and interpret, even for the developers themselves. For this reason, they are nowadays often referred to as *black-box* (BB) models. Recent regulations (such as the General Data Protection Regulation[1] and the recent AI Act[2]) have stressed the need for automated systems decisions to be explainable, especially when these may affect individuals in high-risk domains.

To address these concerns about BB, *Explainable AI* (XAI) was proposed as a direction to create AI systems whose internal decision-making can be understood and appropriately trusted by end "lay" users (Gunning and Aha 2019). This field spotlights the need for *transparent systems* that are able to provide a human-understandable rationale for their decision, as well as *post hoc methods*, which formulate explanations from BB outputs (see (Guidotti *et al.* 2018) for a survey of such methods). An advantage of this latter approach is that they do not require to alter the black-box structure, thus not impacting the performance of these models. Transparent systems, on the other hand, often impose a trade-off between accuracy and explainability.

1.1 Motivation

Preference learning represents a particularly relevant domain for explainability, as it directly models human choices and decision processes. It tackles everyday decision processes of users in several contexts, ranging from recommendation systems to search tools. Additionally, this domain, formalized in Section 3.1, naturally accommodates decision-making under uncertainty, for instance, by reflecting the noisy labels common in real-world tasks, thus providing an interesting and realistic application domain for XAI techniques.

[1] https://eur-lex.europa.eu/eli/reg/2016/679/oj
[2] https://www.europarl.europa.eu/doceo/document/TA-9-2024-0138˙EN.pdf

https://doi.org/10.1017/S1471068426100441 Published online by Cambridge University Press

1.2 Problem statement

In this work, we address the problem of explaining black-box models for preference learning by means of symbolic, interpretable theories. We work in a post hoc setting, where a neural network has already been trained to implement a preference model, and we seek to approximate its behavior with logical theories learned by an Inductive Logic Programming (ILP) framework. Following the formal perspective adopted later in Section 3, an *explanation* is a transparent (i.e., low-complexity) model that approximates the behavior of a black box with high fidelity on a suitable region of the input space, in the sense of (Ribeiro *et al.* 2016).

We propose a post hoc method to explain black-box models for preference learning using Answer Set Programming (ASP) (Calimeri *et al.* 2020) and the Inductive Logic Programming (ILP) framework ILASP (Law *et al.* 2014, 2015, 2018), building upon (D'asaro *et al.* 2020; Fossemò *et al.* 2022). These systems can output logical theories that easily translate into natural language and, therefore, can be understood even by non-experts. For this reason, they are easier to check (when compared to non-transparent ML models) for the presence of biases and artifacts resulting from training on large collections of data (see, e.g., (Caliskan *et al.* 2017; Schramowski *et al.* 2020)).

Following (D'asaro *et al.* 2020), we deepen our exploration of ILASP as a method to analyze and produce theories from target black boxes, which may prove useful in situations where the data the black boxes were trained on is not available. This method may help identify systematic biases, such as, for example decisions influenced by prejudices or unfair principles.

In (Fossemò *et al.* 2022) we presented a preliminary implementation and workflow to use ILASP to approximate black-box models in the context of preference learning systems, by presenting methods to tackle practical limitations of ILASP when dealing with high-dimensional feature spaces. Building on that work, the present study extends both the experimental analysis and the evaluation framework. In particular, while the previous study focused on preliminary results limited to the global approximation of deep neural networks, this paper (i) refines the formal notion of approximation, distinguishing explicitly between local and global explanations, and (ii) provides a broader empirical evaluation of global approximation. Moreover, we include a new ground-truth-based metric that leverages the transparency of Inductive Logic Programming to assess the quality of the learned theories.

1.3 Scope beyond preference learning

It is worth noting that our method is not limited to preference learning. As discussed in Section 3.1, our framework operates in the general context of ranking problems: we approximate a black-box model that orders items according to some (possibly latent) criterion. Ranking tasks involving black-box models are not restricted to user preferences: examples include information retrieval, where documents are ranked by relevance (Severyn and Moschitti 2015), and scheduling problems such as cluster scheduling (Li *et al.* 2022), where tasks are ordered to optimize system performance. Our choice of preference learning is therefore motivated both by its intrinsic relevance and by its natural fit with weak constraints in ASP and ILASP.

https://doi.org/10.1017/S1471068426100441 Published online by Cambridge University Press

Our approach integrates Principal Component Analysis (PCA) to balance feature space reduction with the clarity and fidelity of the resulting theories to the intended models. To support this methodology, we developed a dataset capturing user preferences for recipes sourced from GialloZafferano,[3] a popular Italian culinary website. This dataset comprises two main sections: a recipes section, created using a crawler to collect the top recipes and their ingredients categorized into classes, and a user preferences dataset, assembled from surveys designed to representatively sample user tastes across the recipe spectrum. Both datasets underwent preprocessing, including normalization and standardization for recipes, and PCA and k-means clustering for user preferences. We trained the black-box model on user preferences and approximated it with ILASP. The result of this approximation is a theory composed of weak constraints, which represents the user preference system deduced from the black-box model. The code to check and reproduce our experiments can be accessed through Github.[4]

1.4 Research questions

The research questions addressed in this work are the following:

- **Q1**: Is it possible to make a black-box model transparent (globally and locally) using an Inductive Logic Programming framework?
- **Q2**: What level of fidelity can be achieved with such an approach?
- **Q3**: Is the method scalable as the amount of data increases?

1.5 Contributions

The main contributions of this paper are as follows:

1. We propose a framework to explain the decision-making process of black-box models in preference learning through Inductive Logic Programming, approximating the model both globally and locally;
2. We address the exponential growth in execution time with respect to dataset size by introducing Principal Component Analysis in the pipeline;
3. We introduce a novel ground-truth-based metric that leverages ILASP's transparency to better assess the quality of learned logical theories;
4. We present a curated dataset of recipes and user preferences, designed to evaluate explainable preference learning systems under realistic conditions.

1.6 Results and generality

As shown in Section 6, our proposed methods to tackle scalability issues are effective both in terms of execution time and fidelity. In fact, even in the worst cases, these methods manage to increase fidelity while reducing execution time. Furthermore, as described in Section 5, our approach is *model-independent* in the sense that it treats the neural network as an oracle, querying it on suitably sampled instances, rather than relying on direct access to the original training data. In practice, the ILASP mode bias and the sampling

[3] https://www.giallozafferano.it/
[4] https://github.com/DanieleF198/ILASP-as-post-hoc-method-in-a-preference-system

https://doi.org/10.1017/S1471068426100441 Published online by Cambridge University Press

procedures are instantiated from the feature representation of the available data, but the explanation pipeline itself does not require retraining or modifying the black-box model. Thus, the same methodology can be applied to contexts beyond preference learning by appropriately modifying the sampling technique, then processing the resulting instances with PCA (if needed), and adapting ILASP's mode bias to match the dataset features. Although this may look non-trivial, this pipeline can be automated, and we plan to do so in future work.

The remainder of this paper is structured as follows. Section 2 reviews the main contributions in the literature that are related to our work. Section 3 introduces the necessary background concepts, including preference learning, Answer Set Programming (ASP), ILASP, and the process of approximating a neural network through ILASP. Section 4 describes the datasets used in our experiments, namely the *Recipes Dataset* and the *Users Preferences Dataset*. Section 5 details the proposed methodology, including the experimental setup for global and local approximation, the use of *indirect* and *direct* PCA, preliminary analyses, and the definition of truth scores. Section 6 presents and discusses the experimental results for both global and local approximations. Finally, Section 7 concludes the paper and outlines possible directions for future work.

2 Related Work

2.1 Explainable artificial intelligence

XAI is a research field that aims to make AI systems more understandable to humans (Adadi and Berrada 2018). This field of study has gained increasing attention in recent years, together with the success of machine learning and deep learning in many application areas. A key driver of this interest is the need to ensure transparency and accountability in AI systems, especially in critical domains such as healthcare, finance, and law enforcement. Recent taxonomies (Arrieta *et al.* 2020; Weber *et al.* 2023; Fan *et al.* 2021) organize XAI methods along several dimensions. A useful distinction is between *transparent-by-design*, *model-specific*, and *post hoc* approaches. Transparent-by-design methods are simple enough to be interpretable (e.g., decision trees or sparse linear models). Model-specific methods exploit the internal structure of complex models and try to reconstruct the reasoning that led to specific outputs. Post hoc methods, by contrast, aim to explain the results of a black-box model without making strong assumptions about its internal structure (Guidotti *et al.* 2018). Another widely used distinction is between *global* and *local* methods: global techniques seek to capture the overall decision logic of the black box, whereas local techniques focus on explaining the prediction for a particular input or a small neighborhood of it. Our work fits into this landscape as a post hoc, rule-based approach that can be instantiated both globally and locally, in the sense made precise in Section 3.4.

2.2 Previous work and ILP-based proposed approach

This paper builds and extends upon (D'asaro *et al.* 2020) and (Fossemò *et al.* 2022). In the former, ILASP was applied as a post hoc XAI method to local and global approximation of *Support Vector Machines* (SVMs), a set of simple classical ML models that have

https://doi.org/10.1017/S1471068426100441 Published online by Cambridge University Press

often been used to benchmark explainable AI technologies (see e.g., (Kamishima 2003)). The present work addresses the need for enlarging the scope of black boxes taken under consideration to more fashionable (and complex) ML models such as NNs. In (Fossemò *et al.* 2022), we continued the work with the creation of a dataset in order to have larger data and a more complex neural network to conduct studies. In that preliminary study, the focus was on global approximation of a neural network trained on a first version of the recipes dataset, and on showing that ILASP could, in principle, approximate the black box with weak-constraint theories. In this paper, we present the final version of the dataset and a substantially extended set of experiments on both global and local approximation. We systematically investigate the behavior of ILASP under variations of (i) the sampled dataset size, (ii) the parameters of the *indirect* and *direct* PCA methods, and (iii) the `#maxp` ILASP hyperparameter. For the evaluation, in addition to the estimators described in Section 6.1, such as the classical fidelity indicator (Guidotti *et al.* 2018), we also consider execution time, the length of the returned theories, and ground-truth scores based on the weak-constraint definition. This more extensive analysis clarifies in which regimes ILASP can provide faithful and compact explanations of neural preference models.

Deep neural network architectures currently represent the state of the art in preference learning and recommendation, as they effectively capture complex patterns of user: item relevance and ranking relations. For this reason, a DNN model was adopted in our experiments as the black-box to be approximated. Representative examples of such architectures include DNNs combined with matrix factorization (Zhu *et al.* 2018), dual adversarial networks (Zhang *et al.* 2021), cross-domain DNNs (Hong *et al.* 2020), and esthetic cross-domain networks (Liu *et al.* 2022). Our goal is not to propose a new recommendation architecture, but to study how an ILP-based surrogate can approximate and explain a reasonably powerful neural preference model. The framework proposed in this paper can in principle be applied on top of more sophisticated recommendation architectures, providing a symbolic approximation layer that makes their learned preference structures explicit and interpretable.

2.3 Alternative ILP frameworks

Our work is not the only one using ILP to approximate black-boxes. For instance, (Rabold *et al.* 2018, 2020) and (Shakerin and Gupta 2019) use the Prolog-based system Aleph (Srinivasan 2019). We use ILASP instead, which was shown to outperform Aleph as well as decision trees (De Ville 2013) when applied to preference learning dataset (Law 2018). Even if not used in similar task, a mention must be made to FastLAS (Law *et al.* 2020). FastLAS is an ILP system based on the ILASP framework that solves a restricted version of the context-dependent learning from answer set task. This method received attention thanks to its faster performance on large-scale ILP benchmarks. However, FastLAS in its current form does not yet support weak constraint learning using ordering examples, which is central to our use case. In our setting, the ability to learn weak constraints and explicit rules is not only a technical requirement but also a central ingredient of transparency: the learned weak constraints make explicit the trade-offs and relative penalties that drive the underlying preference model. This justifies the use of ILASP, despite its

https://doi.org/10.1017/S1471068426100441 Published online by Cambridge University Press

higher computational cost compared to FastLAS, because it offers the expressivity needed to capture fine-grained preference structures and to present them in an interpretable, rule-based form. This limitation underscores the novelty and technical challenge of our work, which leverages ILASP's unique capabilities for learning preferences in a logical framework.

2.4 Neurosymbolic ILP approaches

Integrated neurosymbolic methods have recently received considerable attention for their ability to combine neural networks, which handle non-symbolic data such as images, audio, and raw text, with symbolic reasoning. For instance, Collevati et al., (Collevati *et al.* 2024) propose a neurosymbolic pipeline for slice discovery that extracts logical rules describing model behaviors and then uses those rules to generate new data to amend the underlying classifier. Other lines of work pursue similar goals with different design choices, such as Hillerström et al., (Hillerström and Burghout 2024), which aims to handle probabilistic background knowledge by using neural components to predict the probability that a predicate holds (e.g., in an object-detection scenario). Our method, by contrast, has a different objective: it is intended purely as a *post hoc* explanation mechanism for black-box models, rather than as a way to improve their predictive performance or correct them through additional training. Nonetheless, the integration of neurosymbolic components could be explored in future work to extend our approach to non-symbolic input domains.

2.5 Non-ILP approaches to black-box explanation

ILP is not the only possible approach to explaining black-box systems: for instance, Exceptional Preferences Mining (Rebelo *et al.* 2016) and LIME (Ribeiro *et al.* 2016) are other techniques whose relationship to ILP and ILASP may be the subject of future work. Specifically, (Brunot *et al.* 2022) proposes LIRE on the MovieLens (Harper and Konstan 2015) dataset, a local surrogate model tailored for recommender system which extends LIME. Another work in the same domain is presented in (Hu *et al.* 2022), where Pref-SHAP is introduces, a Shapely value-based model explanation framework for pairwise comparison data, applied to a variety of synthetic and real-world dataset. These methods provide feature-attribution style explanations around specific predictions of recommendation models and have been influential in the XAI literature for recommender systems. Despite these methods are promising, they fundamentally differ from our approach. LIRE focuses on rating prediction rather than on learning global, symbolic preference theories, and both LIRE and Pref-SHAP provide local importance scores but do not aim at inducing an explicit, human-readable rule set that captures the overall preference structure. Indeed, LIRE focuses on classification rather than preference learning, and both LIRE and Pref-SHAP do not account for uncertainty. Furthermore, our method allows to get interpretable results which can be easily translated in natural language so that are easy to understand even to non-experts. We previously mentioned that our technique is a *post hoc* one, namely it applies to any black-box whose training was performed before the explanation procedure takes place. Other powerful techniques can be deployed if

https://doi.org/10.1017/S1471068426100441 Published online by Cambridge University Press

we relax this constraint and perform model-specific operations throughout the training phase, as in (Ferreira *et al.* 2022) that is specific to NNs and requires some additional labeling effort. These systems are potentially more accurate than ours. Nonetheless, they are not post hoc and thus are less generally applicable than our approach.

3 Background

3.1 Preference learning

Preference Learning (Furnkranz 2010; Gurrieri *et al.* 2014; Zhou *et al.* 2014) is a subfield of Artificial Intelligence (AI) whose purpose is profiling user preferences. This field deals with several scenarios: for instance, a search engine that must return the results according to the user past searches; a streaming platform that needs to order videos according to the user's preferences. In both examples, Preference Learning aims to build a system that collects user data and returns the results that are likely to be preferred by the user.

The problem we have informally described here is to order a set of elements based on user preferences, that is a *ranking* problem. Among the various instantiations of ranking, we mainly focus on *label ranking*. Given an instance space $\boldsymbol{X}$ and a set of labels $L = \{\lambda_1, \lambda_2, \ldots, \lambda_k\}$, an ordering for an instance $\boldsymbol{x}$ can be described as ordering of the labels by means of a relation $\succ_{\mathbf{x}}$ such that $\lambda_i \succ_{\mathbf{x}} \lambda_j$ indicates that λ_i is preferred over λ_j by instance $\boldsymbol{x}$. So the ordering induced by $\succ_{\mathbf{x}}$ can be identified by a permutation $\pi_{\mathbf{x}}$ such that $\pi_{\mathbf{x}}(i) = \pi_{\mathbf{x}}(\lambda_i)$ indicates the position of the label λ_i in the ordering. So, a complete ordering of the labels in L can be described as following:

$$\lambda_{\pi_{\mathbf{x}}^{-1}(1)} \succ_{\mathbf{x}} \lambda_{\pi_{\mathbf{x}}^{-1}(2)} \succ_{\mathbf{x}} \ldots \succ_{\mathbf{x}} \lambda_{\pi_{\mathbf{x}}^{-1}(k)} \tag{1}$$

where $\pi_{\mathbf{x}}^{-1}(j)$ indicates the index of the label which occupies the j-th position in the ordering (Zhou *et al.* 2014). An ordering of elements can be decomposed in the form of *pairwise comparison*. The idea behind this decomposition is that the ordering of a set of elements can be structured as a succession of consecutive questions "do you prefer λ_i over λ_j?" (with $i \neq j$) in which the answers could be either "yes" or "no." Thus, knowing the position of each element, we can define the preference ordering as follows:

$$f(\lambda_i, \lambda_j) = \begin{cases} \lambda_i \succ_{\mathbf{x}} \lambda_j, & \text{if } \lambda_i \text{ precedes } \lambda_j \text{ in the ordering} \\ \lambda_j \succ_{\mathbf{x}} \lambda_i, & \text{otherwise} \end{cases} \tag{2}$$

where the number of pairs (λ_i, λ_j), if k indicates the number of elements in the collection, is equal to $k(k-1)/2$. This decomposition allows us to decompose the original problem into a set of simpler sub-problems, which is useful from a machine learning perspective (Furnkranz 2010). In our work we deal with the case where the relationship between any two elements may be unknown or uncertain, i.e., for an instance $\boldsymbol{x}$ and indices i, j we are unsure as to whether $\lambda_i \succ_{\mathbf{x}} \lambda_j$ or $\lambda_j \succ_{\mathbf{x}} \lambda_i$. For pairs of elements for which the preference relationship is unknown, we label them as *uncertain* pairs (denoted with $\lambda_i \sim_{\mathbf{x}} \lambda_j$). This makes our problem a *ternary* classification problem.

$$p_x(\lambda_i, \lambda_j) = \begin{cases} 1 & \text{if } \lambda_i \succ_x \lambda_j \\ 0 & \text{if } \lambda_i \sim_x \lambda_j \\ -1 & \text{if } \lambda_j \succ_x \lambda_i \end{cases} \tag{3}$$

https://doi.org/10.1017/S1471068426100441 Published online by Cambridge University Press

In the rest of the paper, we will often view $p_x(\lambda_i, \lambda_j)$ as the (symbolic or neural) output of a pairwise comparison model f_x for user $\mathbf{x}$, which will later be approximated by ILASP theories.

3.2 Answer set programming

Answer Set Programming (Calimeri *et al.* 2020) is a declarative logic programming language based on the stable model semantics, whose output is a model of the input theory called *stable model* or *answer set*. This language is at the base of our work thanks to the possibility to define *weak constraints*. In fact, weak constraints give us a suitable and natural way to represent the preference relation $\succ_{\mathbf{x}}$ in ASP, inducing a (partial) ordering over the stable models of a theory. Given n, $m \geq 0$, a weak constraint has the form:

$$:\sim \texttt{b1}, \ldots, \texttt{bn}[\texttt{w@l}, \texttt{t1}, \ldots, \texttt{tm}] \tag{4}$$

where `b1`, ..., `bn` are *literals*, w is a *weight* associated with the constraint, l is a *priority level*, and `t1`, ..., `tm` are used to handle independence among weak constraints. We are not going to discuss this syntax here, but the interested reader can find the full syntax and semantics in (Calimeri *et al.* 2020). The important difference with hard constraints is that while these affect the answer set of a theory by excluding some solution at prior, weak constraint induce a preference relation among them, exactly as we want to naturally reflect a user preference system. To describe this preference relation, we need to introduce the notion of *cost* of an answer set at some priority level l. This is defined as the sum of weights at priority level l for all the weak constraints such that their bodies are satisfied by the answer set.

Example 1.
Consider the ASP program P consisting of the following axioms and weak constraint:

```
p(a). p(b). p(c).
0{ q(X) }1 :- p(X).
:~ q(a). [1@2, a]
:~ q(b). [3@1, b]
:~ q(c). [-1@2, c]
```

This theory has 8 answer sets, namely: $\{p(a), p(b), p(c)\}$, $\{p(a), p(b), p(c), q(b)\}$, $\{p(a), p(b), p(c), q(c)\}$, $\{p(a), p(b), p(c), q(b), q(c)\}$, $\{p(a), p(b), p(c), q(a)\}$, $\{p(a), p(b), p(c), q(a), q(c)\}$, $\{p(a), p(b), p(c), q(a), q(b)\}$ and $\{p(a), p(b), p(c), q(a), q(b), q(c)\}$. These eight answer sets are obtained by *ignoring* the weak constraints, that is, by computing all stable models of the hard part of the program and then evaluating their costs afterwards. Weak constraints do not change which answer sets exist, but they induce a preference ordering over them. When weak constraints are taken into account, answer sets are ranked lexicographically by their total cost at each priority level: one first minimizes the sum of weights at level 1, and, among those tied, minimizes the sum at level 2, and so on. Under this criterion, the optimal (i.e., most preferred) answer set is $\{p(a), p(b), p(c), q(c)\}$, which has total cost 0 at level 1 and -1 at level 2 (the lowest at each level). Moreover, consider $A = \{p(a), p(b), p(c), q(a), q(b), q(c)\}$ and $B = \{p(a), p(b), p(c), q(a), q(b)\}$. Then

https://doi.org/10.1017/S1471068426100441 Published online by Cambridge University Press

A is preferred over B with respect to weak constraints, since even though both have cost 3 at level 1, A has a lower cost than B at level 2 (i.e., 0 vs 1). In general, we say that an answer set A is preferred to an answer set B (according to theory T) if the cost of A is strictly smaller than that of B at the highest level for which their costs differ.

3.3 ILASP

ILASP (short for *Inductive Learning of Answer Set Programs*) (Law *et al.* 2014, 2015, 2018) is an Inductive Logic Programming language which can learn ASP programs, including weak constraints. This means that we can represent preferences using weak constraints in ASP and make ILASP learn them. Moreover, ILASP's output can be expressed in natural language, although this translation may require some effort and may vary from case to case. Nonetheless, the structured and human-readable form of ILASP rules supports its effectiveness as an XAI method by providing clear and transparent explanations that can be made accessible to non-expert end users. In the remainder of this section, we focus on the abstract ILASP formalization rather than on concrete syntax. Full syntactic details and complete code listings for the examples are reported in Appendix A.

The three main components of ILASP are the *background knowledge*, the *language bias* and the *examples*. The background knowledge B is an ASP program and represents the prior knowledge ILASP has prior to rule induction. The language bias L (often called a *mode bias*) is used to specify the so-called *hypothesis space* or *search space*, which is the set of rules that ILASP can learn. Language bias can be defined both explicitly and through the *mode declarations*. Since in many situations it is not practical to define the language bias explicitly, it is preferable to use mode declarations, which are ground atoms whose arguments are the so-called *placeholders*. Intuitively, mode declarations specify which predicates may appear in the head or body of learned rules, which arguments are variables or constants of given types, and which atoms may occur inside weak constraints. In this way, L compactly characterizes the space S_L of candidate hypotheses without committing to a particular syntax.

Lastly, the examples are observations guiding the learning process. Examples are divided into positive and negative examples (denoted E^+ and E^-, respectively), indicating which situations should or should not be entailed by the target theory. However, using only the examples, it is not possible to make ILASP learn the weak constraints, as this subdivision does not give any information on how learnable answer sets should be ordered. For this reason, ILASP allows a second type of example called an *ordering example*, which serves to define a preference ordering over a set of answer sets. These ordering examples are crucial in our work, since they give us the point of contact between preference learning, ASP and ILP systems. Cautious and brave ordering examples constrain, respectively, all or at least one of the answer sets extending one example to be preferred to those extending another, according to the weak-constraint optimization. In our setting, brave orderings are used to encode pairwise user (or neural) preferences between items.

Finally, we can define what an ILASP task is:

https://doi.org/10.1017/S1471068426100441 Published online by Cambridge University Press

Definition 1.
An ILP task is a tuple $T = \langle B, S_L, E^+, E^- \rangle$. *A hypothesis* H *is an inductive solution of* T *(written* $H \in ILP_{LAS}(T)$*) if and only if:*

1. $H \subseteq S_L$
2. $\forall e^+ \in E^+ \exists A \in AS(B \cup H)$ such that A extends e^+
3. $\forall e^- \in E^- \nexists A \in AS(B \cup H)$ such that A extends e^-

where S_L *is the search space induced by the language bias* L *and* $AS(\cdot)$ *is the set of all answer sets of an ASP program* $(\cdot)$.

Example 2.
Consider the following code:

```
#modeo(1, value(const(val), var(val))).
#modeo(1, category(const(mg)), (positive)).
#weight(val).
#weight(1).
#weight(-1).
#constant(mg, 1).
#constant(mg, 2).
#constant(val, p).

#pos(item0,{},{},{category(2). value(p, 1).}).
#pos(item1,{},{},{category(1). value(p, 2).}).
#pos(item2,{},{},{category(2). value(p, 3).}).

#brave_ordering(o0@2,item2,item1,>).
#brave_ordering(o1@2,item2,item0,>).
#brave_ordering(o2@1,item1,item0,<).
```

Here the first block of code is used to define the language bias L which defines the search space. the declaration #`modeo`(1, `value`(`const`(`val`), `var`(`val`))) puts the following constraints in the search space:

```
:~ value(p,V1).[1@2, V1]
:~ value(p,V1).[-1@2, V1]
:~ value(p,V1).[V1@2, V1]
:~ value(p,V1).[-V1@2, V1]
:~ value(p,V1).[V1@1, V1]
:~ value(p,V1).[-1@1, V1]
:~ value(p,V1).[1@1, V1]
:~ value(p,V1).[-V1@1, V1]
```

Listing 1: constraints generated by the first line in Example 2

https://doi.org/10.1017/S1471068426100441 Published online by Cambridge University Press

Brave orderings allow us to express a preference relationship between items. In this example, `item2` is not preferred to `item1` and `item0`, while `item1` is preferred to `item0`. Note that in the first two `#brave_ordering` declarations the penalty is set to 2, and so ILASP will give a penalty score equal to 2 to those theories that do not cover those orderings. Jointly with the hypothesis space defined by the `#modeo` declaration, the program will learn a theory consisting of weak constraints expressing the order preferences exploited by brave orderings. In this case the returned theory is:

```
:~ value(p,V1).[V1@1, V1]
```

Listing 2: Theory returned by Example 2

meaning that, given the defined cost, the higher the `value` of `p`, the better. Abstractly, this example illustrates a typical ILASP setup: background knowledge encodes the space of possible items, the language bias specifies which weak-constraint patterns can be used to express preferences, and brave orderings encode training preferences that guide the search towards a hypothesis H capturing a monotonic relationship between an attribute (`value/2`) and preference. The concrete syntax above is only one possible realization of this abstract schema.

3.4 Explaining black boxes

In this section we clarify what it means for a black-box model (such as a Neural Network) to be *approximated* and *explained*. Following Ribeiro *et al.* (2016), an explanation is a *transparent* (i.e., low-complexity) model that approximates the behavior of a black box with high fidelity on a suitable region of the input space.[5]

3.4.1 Preliminaries

Formally, let $f : X \to Y$ be the function computed by a fixed black-box model, and let $\mathcal{T}$ be a class of candidate white-box models with the same input space X (for us, typically ILASP theories). We assume:

- an *error* (or *unfaithfulness*) measure
$$L(T, f, A) \in \mathbb{R}_{\geq 0}$$
that quantifies how poorly $T \in \mathcal{T}$ approximates f on a non-empty region $A \subseteq X$;
- a *complexity* measure
$$\Omega : \mathcal{T} \to \mathbb{R}_{\geq 0}$$
that penalizes non-interpretable models (e.g., number of rules, depth of a tree, number of non-zero weights). Small values of $\Omega(T)$ correspond to *transparent* or *low-complexity* models.

Intuitively, Ribeiro *et al.* (2016) first isolates fidelity to the black box, and only then adds a complexity constraint: a model that nearly minimizes L is a good *approximation*

[5] In Ribeiro *et al.* (2016), this is instantiated by minimizing, for each prediction, a locality-aware unfaithfulness term plus a complexity penalty. Our presentation abstracts this setup to generic regions of the input space and generic error and complexity measures.

https://doi.org/10.1017/S1471068426100441 Published online by Cambridge University Press

of f; among those, the models with low Ω are taken as *explanations*. This leads to the following.

Definition 2 (Approximation on a region).
Fix a region $A \subseteq X$ and a tolerance $\varepsilon \geq 0$. A model $T \in \mathcal{T}$ is an approximation *of f on A if*

$$L(T, f, A) \leq \inf_{T' \in \mathcal{T}} L(T', f, A) + \varepsilon.$$

That is, T almost minimizes the error of approximating f on A.

Definition 3 (Explanation on a region).
Fix in addition a complexity threshold $\kappa \geq 0$. A model $T \in \mathcal{T}$ is an explanation *of f on A if:*

1. *T is an approximation of f on A (as above), and*
2. *$\Omega(T) \leq \kappa$ (i.e., T is transparent).*

Intuitively, an explanation is a low-complexity model that approximates f with high fidelity on A.

3.4.2 Local and global explanations

We now specialize the choice of the region A to distinguish global and local explanations, again in the spirit of Ribeiro *et al.* (2016).

Definition 4 (Global explanation).
Let $X_{\text{rel}} \subseteq X$ be a task-relevant region of the input space (e.g., the subset of X on which f is intended to be used). A model $T \in \mathcal{T}$ is a global explanation *of f (relative to X_{rel}) if it is an explanation of f on X_{rel}, that is it approximates f well on X_{rel} and has low complexity.*

To capture locality, we endow X with a metric d that induces neighborhoods around each input.

Definition 5 (Local explanation).
Let (X, d) be a metric space, and let $x^\star \in X$ be a specific query instance. For $r > 0$, write

$$B_r(x^\star) = \{\, x \in X \mid d(x, x^\star) \leq r \,\}$$

for the closed ball of radius r around $x^\star$. A model $T_{x^\star} \in \mathcal{T}$ is a local explanation *of f at $x^\star$ if it is an explanation of f on $B_r(x^\star)$, i.e., it achieves low error $L(T_{x^\star}, f, B_r(x^\star))$ while having low complexity $\Omega(T_{x^\star})$.*

3.4.3 ILASP approximation of neural networks

With respect to the notation introduced in Section 3.1, let $\mathbf{x}$ be a user, let $L = \{\lambda_1, \lambda_2, \ldots, \lambda_k\}$ be the set of elements that we aim to order according to the preference system of $\mathbf{x}$, and let $T_{\mathbf{x}}$ be the ILASP theory learned from data about $\mathbf{x}$. Then, we

https://doi.org/10.1017/S1471068426100441 Published online by Cambridge University Press

model the pairwise preference relations $\succ_{\mathbf{x}}$ and $\sim_{\mathbf{x}}$ induced by $T_{\mathbf{x}}$ as follows:

$$T_{\mathbf{x}}(\lambda_i, \lambda_j) = \begin{cases} \lambda_i \succ_{\mathbf{x}} \lambda_j, & \text{if } \lambda_i \text{ has lower cost than } \lambda_j \text{ w.r.t. } T_{\mathbf{x}}, \\ \lambda_j \succ_{\mathbf{x}} \lambda_i, & \text{if } \lambda_j \text{ has lower cost than } \lambda_i \text{ w.r.t. } T_{\mathbf{x}}, \\ \lambda_i \sim_{\mathbf{x}} \lambda_j, & \text{if } \lambda_i \text{ and } \lambda_j \text{ have the same cost w.r.t. } T_{\mathbf{x}}. \end{cases} \tag{5}$$

Thus, starting from the pairwise comparison problem, we can establish (partial) orderings of the elements in L by using the ranking formulation introduced in (1).

In practice, $T_{\mathbf{x}}$ is learned from a finite *ground-truth* set of already ordered pairs for user $\mathbf{x}$. Following the code example in Appendix D, the ILASP *brave ordering* encodes this ground truth, and the returned theory $T_{\mathbf{x}}$ is the corresponding preference system learned by ILASP.[6]

From the point of view of the general framework introduced above, we can regard the (unknown) user preference system as a target function

$$f_{\mathbf{x}} : X \to Y,$$

where X is the space of ordered pairs (e.g., feature representations of (λ_i, λ_j)) and Y encodes the outcome of the pairwise comparison (e.g., "λ_i preferred," "λ_j preferred," or "tie"). The ILASP theory $T_{\mathbf{x}}$ is an interpretable model in a hypothesis class $\mathcal{T}$ that approximates $f_{\mathbf{x}}$ on some region $A \subseteq X$ in the sense that it nearly minimizes the error $L(T_{\mathbf{x}}, f_{\mathbf{x}}, A)$ introduced in the previous subsection. Since we only observe a finite ground-truth set of pairs, we cannot determine whether $T_{\mathbf{x}}$ perfectly aligns with the true preference system of $\mathbf{x}$ on unseen pairs; we therefore treat $T_{\mathbf{x}}$ as an *approximation* of $f_{\mathbf{x}}$ on the relevant region A.

In the black-box explanation setting considered in this paper, the role of the target function is played by a neural network $f : X \to Y$ trained to model the preference system of user $\mathbf{x}$. Instead of using human-labeled pairs as ground truth, we use the predictions of f as labels for training ILASP. The resulting theory, which we denote by T_{NN}, is then an interpretable surrogate that approximates the neural network on a region $A \subseteq X$ by (approximately) minimizing $L(T_{NN}, f, A)$. Whenever T_{NN} achieves low error and has low complexity (e.g., few rules, shallow structure), it qualifies as an *explanation* of the neural network: it is a transparent model that closely approximates the black box on A.

Following standard terminology on explanation properties Guidotti *et al.* (2018), we use *fidelity* to describe how closely such an interpretable model approximates the neural network. In our pairwise setting, and in line with (5), we estimate fidelity by comparing the outcome $T(\lambda_i, \lambda_j)$ with the prediction $NN([\lambda_i, \lambda_j])$ over a sample of pairs, and we derive quantitative scores from these agreements and disagreements. The specific fidelity scores used in our experiments are described in Sections 6.1 and 5.3, while the neural network architecture is reported in Appendix B.

In our specific setting, the black-box model f is a neural network defined on the feature space F_{NN} of pairs (λ_i, λ_j), so that $X = F_{NN}$, each instance $x^\star \in X$ corresponds to one such pair, and the surrogates T (for global approximation) and $T_{x^\star}$ (for local

[6] Note that there is a discrepancy in notation between ILASP and preference learning. In ILASP, *brave orderings* use $>$ to state that the *second* item is preferred over the first, whereas in preference learning the symbol $\succ$ denotes that the *first* item is preferred over the second. Although this may be slightly confusing, we keep this convention to remain consistent with the respective literatures.

https://doi.org/10.1017/S1471068426100441 Published online by Cambridge University Press

approximation) are instantiated as symbolic ILASP theories learned from f according to the general local/global definitions given above.

4 The dataset

In this section, we discuss the dataset we built to carry out our experiments on. The interested reader can find the dataset online.[7] The dataset concerns users' gastronomic preferences. We collected recipes, their main features, and users' preferences over these recipes. The public availability of this resource is itself one of the contributions of this work: the dataset contains 100 curated recipes, 277 distinct ingredients organized into 36 ingredient classes and 12 meta-classes, and explicit preference data from 54 users. For each user we derive 210 labeled pairwise comparisons, yielding $54 \times 210 = 11\,340$ annotated pairs in total. This combination of rich structured features and dense pairwise preference information is, to the best of our knowledge, not available in existing open datasets for food recommendation and preference learning. We carefully considered existing benchmarks for preference learning and recommendation, both food-oriented (e.g., sushi preference datasets (Kamishima 2003)) and generic recommendation datasets such as MovieLens (Harper and Konstan 2015). However, these datasets either lack fine-grained, interpretable ingredient-level features or do not expose a structured taxonomy of attributes that can be naturally mapped to logical predicates and weak constraints. In this work, we explicitly leverage different levels of granularity in the feature space (ingredient, class, meta-class), which are crucial both for ILASP-based rule learning and for human-understandable explanations. For this reason, we rely on the dataset presented in (Fossemò *et al.* 2022) and extend it here as a reusable resource that can serve as a benchmark for interpretable preference learning, rule extraction, and XAI studies.

4.1 Recipes dataset

This dataset was automatically collected by means of a crawler that scraped data from the popular Italian recipe database GialloZafferano and then manually verified and processed to ensure data quality.[8] From the 4000 recipes in the database, we chose the 100 most voted ones and gathered 277 ingredients, which we grouped into 36 *classes* and 12 *meta-classes*, obtaining three levels of granularity for the ingredients. For example, the ingredient "spaghetti" belongs to the class "dry pasta" which in turn belongs to the meta-class "pasta." In Appendix C we report ingredients, classes, and meta-classes.

In order to deal with the challenge of balancing specificity and dataset size, we organized the ingredients into a hierarchical classification. With 277 ingredients, it would be impractical for participants to evaluate each one in a single survey. To address this, we utilized meta-classes and classes instead of individual elements. On the other hand, since ILASP struggles with high-dimensional feature spaces, we performed ILASP experiments with the 12 ingredients meta-classes, which makes the execution time computationally feasible in our application. The dataset structure is outlined in Table 1. Before analysis,

[7] https://doi.org/10.5281/zenodo.7135196
[8] https://www.giallozafferano.it/

https://doi.org/10.1017/S1471068426100441 Published online by Cambridge University Press

Table 1. *Recipes dataset features*

Feature	Description
ID	A unique integer assigned to the recipe
Name	The name of the recipe
Category	An integer from 1 to 5, where: 1 = starter; 2 = complete meal; 3 = first course; 4 = second course; 5 = savory cake
Cost	An integer from 1 to 5, where: 1 = very low; 2 = low; 3 = medium; 4 = high; 5 = very high
Difficulty	An integer from 1 to 4, where: 1 = very easy; 2 = easy; 3 = medium; 4 = difficult
Preparation time	A positive integer, expressed in minutes
Ingredients	A vector of 36 elements representing the classes of ingredients. The element in position i is equal to 0 if the corresponding ingredient is not used in the recipe; otherwise, it specifies how important the ingredient is in the recipe.
Preparations	A vector of 9 elements. Encoding is similar to the ingredients vector.
Link	A link pointing to the recipe on GialloZafferano

each recipe's ingredient vector and preparation vector were normalized, and each numerical feature was standardized. We manually assigned a level of importance to each recipe and each of the ingredients associated with it. To ensure the importance of ingredients is represented consistently at different granularities, we aggregate the importance values for each class. This means that for each class, we add up the importance values of the ingredients that belong to that class (or meta-class). For example, if a recipe includes "pancetta" (bacon) with an importance value equal to 4, and "prosciutto" (ham) with an importance value equal to 2, both of which are part of the "pork meat" class, then the "pork meat" class will have an importance value equal to 6.

The resulting Recipes Dataset thus provides a mid-sized but carefully curated testbed: the number of recipes is large enough to support meaningful clustering, PCA, and neural-network training, while the ingredient taxonomy and manually assigned importance scores remain compact and interpretable. This is precisely the regime in which ILASP can be applied without prohibitive runtimes, and where the learned weak constraints can be understood in terms of intuitive culinary categories (e.g., "pasta," "pork meat," specific preparation methods).

4.2 Users preferences dataset

A series of data regarding 54 user preferences over the recipes dataset were collected by means of a survey. The most important criteria followed for the realization of this survey was the best trade-off between number of recipes and representativeness of the full distribution of the full set of 100 recipes. To achieve the best results, we created 10 different surveys aiming to be as representative as possible of the distribution of the full set of 100 recipes, ensuring that all recipes appeared in at least some of the surveys.

https://doi.org/10.1017/S1471068426100441 Published online by Cambridge University Press

To do so we used PCA for dimensionality reduction following *Kaiser rule* (taking into account components with eigenvalue ≥ 1) reducing the 47 features to 17 components. Subsequently, we used the k-means clustering to partition the dataset, considering $k = 2$, 3, 4, 5, 20. k was finally set to 3 as it allowed to maximize the number of clusters, while minimizing the inter-group variance, excluding results in which we have group with small cardinality. From each of the three partitions, we draw randomly 7 recipes, which are combined to create a representative subset of 21 recipes. The 10 representative subsets obtained at the end of this process cover the entire recipes' dataset. Recipes can be present in multiple representative subsets, but not more than once in the same one. Each representative subset is used to create a survey, which is described in Table 2.

The survey was administered to 54 participants, each of whom completed exactly one of the ten questionnaires. This design yields a moderate number of users but a relatively dense preference signal for each user, which is adequate for our goal of studying ILASP-based explanations of per-user neural models rather than building a large-scale industrial recommender system.

It is important to note that the section *Sorting recipes* is divided in 3 subsections. Specifically, users are asked to sort 21 recipes in the survey, by effectively sorting 10 elements at a time. This allows on one side to avoid asking users to order too many items at once, on the other to introduce the uncertainty class described in Section 3.1. These subsections are organized to maximize the number of common recipes among the ordering. This would give us as much information as possible about the preferences and uncertainties of the user. In fact, elements present in more than one subsection allow us to combine the three orderings ranked by the user, and create pairwise comparison between the 21 recipes, finding inconsistencies in the middle. This process, which is deeply described in (Fossemò *et al.* 2022), allow us to obtain uncertain pairs, and so passing from the classic pairwise binary classification problem, to the ternary one. It's worth noting that this process would have been impossible to perform if we had used only one sort order, preventing us from inferring user uncertainties even if they are actually present.

Starting from the *sorting recipes* section, we created the user preferences dataset. For each entry (and so, for each user) there are 210 pairs, in which the elements are the ID of the recipes (pairs with elements with same ID were discarded). Each pair has label 1 if the first element is preferred over the second, -1 if the second is preferred over the first, 0 in case of uncertain relationship among the two element. The remaining sections were instead used as ground-truth in the experiment, as fully described in next chapters. In the released dataset, all responses are stored in anonymised form, and the mapping from raw survey answers to ternary pairwise labels is documented, so that other researchers can directly reuse the 11 340 labeled pairs or reconstruct alternative target variables (e.g., from ratings or meta-class evaluations) for related studies on interpretable preference learning and recommendation.

5 Methods

In this section, we introduce methods used to approximate the underlying theory of a Neural Network (Fossemò *et al.* 2022) using ILASP with optimal efficiency. The procedure is summarized in the Figure 1. Conceptually, our method instantiates the abstract

https://doi.org/10.1017/S1471068426100441 Published online by Cambridge University Press

Table 2. *Survey structure*

Section	Description
Personal data	gender, age range and Italian region of residence
Recipes evaluation	each recipe presented is rated on a scale from 1 to 10, where 1 means "I would never eat it" and 10 "I like it very much."
Sorting recipes	where the user has to sort the recipes presented according to her/his personal preferences.
Ingredient classes evaluation	for each ingredient class c, we ask to rate from 1 to 10 the following question: "Do you like recipes where there is c?" where 1 means "absolutely not" and 10 "absolutely yes."
Ingredient meta-classes evaluation	for each ingredient meta-class mc, we ask to rate from 1 to 10 the following question: "Do you like recipes where there is mc?" where 1 means "absolutely not" and 10 "absolutely yes."
Preparations evaluation	for each preparations p, we ask to rate from 1 to 10 the following question: "Do you like recipes which require p?" where 1 means "absolutely not" and 10 "absolutely yes."
Difficulty, preparation time and cost	we ask to rate from 1 to 10, where 1 means "absolutely not" and 10 "absolutely yes," the following questions: "Do you prefer simple cuisine to elaborate cuisine?," "Are you willing to wait for a dish with a long preparation time?," and "Are you willing to spend for a good dish?".
Ingredient/Preparation combination preferences	we ask the user if there are particular combinations of ingredients (class or meta-class) and preparations for which his preferences diverges from the scores assigned to the individual elements (e.g., the user has given a high score to the fish ingredient meta-class and "frying" preparation method, but does not like fried fish). Each combination generated by the user also needs to be rated (as in Recipes evaluation). We limited each combination to a maximum of 3 items between ingredients and preparations, and we collected for each user a maximum of 4 combinations.

approximation framework of Section 3: the neural network B plays the role of the black-box function f, ILASP theories $T \in \mathcal{T}$ are candidate white-box models, and our sampling procedures determine the regions $A \subseteq X$ on which T is required to approximate B (globally or locally). Once the feature space X and the logical vocabulary (predicates, types, and constants) are fixed for a given domain, the overall pipeline – sampling, labeling with B, ILASP training, and evaluation – depends only on access to the trained model B and to unlabeled inputs in X. We do not require access to the original training set of B, which can be beneficial in scenarios where training data are sensitive or unavailable. However, ILASP's background knowledge and language bias are necessarily instantiated

https://doi.org/10.1017/S1471068426100441 Published online by Cambridge University Press

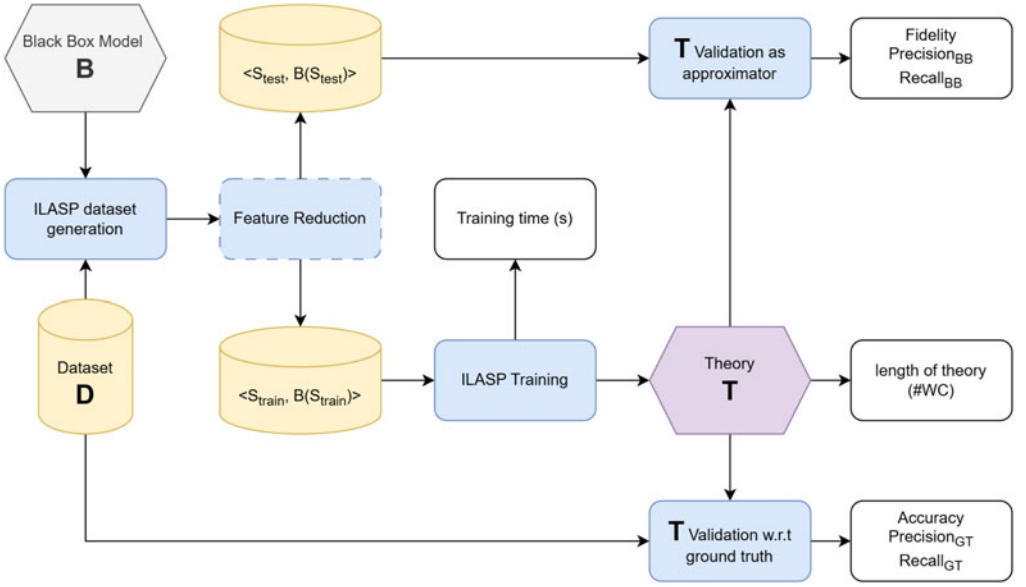


Fig. 1. Conceptual diagram of the overall approach, instantiating the approximation framework of Section 3. Given a dataset D and a pre-trained black box $B : X \to Y$ (in our case, a neural network), we first sample a training set $S_{\text{train}} \subseteq X$ and a test set $S_{\text{test}} \subseteq X$ uniformly at random from the feature space. These instances are then labeled with the corresponding black-box predictions $B(S_{\text{train}})$ and $B(S_{\text{test}})$, respectively. Dataset generation can follow either a *global* strategy (Figure 2a), where S_{train} and S_{test} are drawn from a task-relevant region $X_{\text{rel}} \subseteq X$, or a *local* strategy (Figure 2b), where S_{train} is built by perturbing a neighborhood around specific query instances (cf. Definition of local explanation in Section 3). On these labeled data we optionally apply feature reduction via direct or indirect PCA (Section 5.1) to control the size of the ILASP search space. ILASP is then run on S_{train} with an appropriate language bias L, producing a symbolic theory T consisting of weak constraints. At this stage we record the execution time and the length of the theory (i.e., the number of weak constraints in T) as proxies for computational cost and complexity $\Omega(T)$. Finally, T is evaluated on S_{test} to obtain fidelity, precision_{BB} and recall_{BB} with respect to B, and on user ground truth (Section 5.3) to obtain $\text{accuracy}_{\text{GT}}$, $\text{precision}_{\text{GT}}$ and $\text{recall}_{\text{GT}}$, thus jointly assessing approximation quality and human alignment.

from domain-specific information (e.g., the recipe attributes in Section 4); in this sense, the ILASP components are *tailored to the domain*, even though the methodological steps for approximation are independent of any particular dataset. When no real data are available but the structure of X is known, one can still generate or collect a surrogate dataset of inputs (Angwin *et al.* 2016) to query B and drive ILASP.

For the sake of completeness the neural Network architecture and training is reported in Appendix B. We tested two main approaches: *global approximation* and *local approximation*. Given that ILASP is not fast for highly dimensional datasets, we explored various methods to balance performance, execution time, and complexity of the resulting theories.

https://doi.org/10.1017/S1471068426100441 Published online by Cambridge University Press



Fig. 2. Generation of sampled datasets in the global (Figure 2a) and local (Figure 2b) approximation settings. In the *global* case, we sample N pairs (i_1, i_2) uniformly at random from the feature space X of ordered recipe pairs (cf. Section 3.1) to form S_{train}, and we compute the corresponding labels $B(i_1, i_2)$ from the neural network B, obtaining $B(S_{\text{train}})$. A disjoint test set S_{test} of size K is generated analogously. This realizes a global explanation in the sense of Section 3, with region $A = X_{\text{rel}} \subseteq X$. In the *local* case, we first sample N query pairs $Q_n = (i_1, i_2)_n$ that we wish to explain. For each query Q_n, we generate a local training set $S^{(n)}_{\text{train}}$ of M perturbed pairs $(j_1, j_2)_m = (i_1, i_2)_n + R_m$ by adding Gaussian noise R_m to the feature representation of Q_n (Section 5.1.2). Each perturbed pair is labeled with $B(j_1, j_2)_m$, and the original query Q_n together with its prediction $B(Q_n)$ is used as the corresponding test instance. This implements local explanations on metric balls $B_r(Q_n)$ around each query, as in the formal definition of local approximation.

Our approach involved optimizing `#maxp` ILASP hyperparameter to enhance efficiency. Additionally, we integrated Principal Component Analysis (PCA) to reduce dimensionality and further improve performance. To evaluate the degree of generalization of our explanation, we defined three bespoke scores based on Table 2 data: $\text{Accuracy}_{\text{GT}}$, $\text{precision}_{\text{GT}}$ and $\text{recall}_{\text{GT}}$. These scores assess how well the theories align with the user's preferences obtained through a survey.

https://doi.org/10.1017/S1471068426100441 Published online by Cambridge University Press

5.1 Experimental setup

As we already mentioned, the main objective of this work is to approximate a Neural network (and, more in general, a black-box model) both globally and locally. Recall that we deal with a user-dependent scenario, that is when we talk about approximating a Neural Network, we are talking about a Neural Network trained on the data from a specific user. Formally, for each user $\mathbf{x}$ we train a neural network $B_{\mathbf{x}} : X \to Y$ that approximates the pairwise preference function $p_{\mathbf{x}}(\lambda_i, \lambda_j)$ introduced in Section 3.1. In our experiments, X is the feature space of ordered recipe pairs (Section 4), and $Y = \{-1, 0, 1\}$ encodes whether the first recipe is less preferred, equally preferred, or more preferred than the second.

Concretely, $B_{\mathbf{x}}$ is instantiated as a feed-forward Deep Neural Network with two dense hidden layers of 64 nodes each. A dropout layer with rate 0.1 is applied after every dense layer, and batch normalization is included between the two hidden layers. The network is trained with stochastic gradient descent using a learning rate of 0.0005. We use a tanh activation at the input layer, ReLU in the first hidden layer, a linear activation in the second hidden layer, and a softmax activation in the output layer to model the ternary distribution over $\{-1, 0, 1\}$. Additional architectural and training details (e.g., early stopping, class balancing, and train/validation splits) are reported in Appendix B. On held-out data, these models reach on average 82.72% accuracy, 83.26% precision and 82.43% recall across users, which we take as a reasonable performance baseline for the subsequent approximation by ILASP.

In the remainder of this section, we describe the *general* global and local approximation procedures, deliberately abstracting away from specific training-set sizes N, K and M or from particular PCA configurations. The concrete hyperparameter choices for our experiments are reported in Section 6.1.

5.1.1 Global approximation

In the global case, we are interested in extrapolating the general logic that the Neural Network uses to predict user preferences. The procedure is summarized in Figures 1 and 2a. Specifically, we sample N pairs (i_1, i_2) uniformly at random from the feature space, and label these pairs according to predictions $B(i_1, i_2)$ of the Neural Network. We repeated the same procedure to sample and label a test set of K samples Then, we used ILASP on the training set using an appropriate language bias L in order to obtain an explanation theory. Finally, we test the learned theory on the test set.

In terms of the approximation framework of Section 3, global approximation corresponds to choosing a task-relevant region $X_{\text{rel}} \subseteq X$ (here, the support of the recipe-pair distribution) and seeking an ILASP theory $T \in \mathcal{T}$ that approximately minimizes $L(T, B, X_{\text{rel}})$ while keeping the complexity $\Omega(T)$ (e.g., number of weak constraints) below a threshold. The training set S_{train} provides a finite sample from X_{rel} on which ILASP can observe the behavior of B, while S_{test} is used to estimate fidelity, Precision $_{BB}$ and Recall $_{BB}$ as empirical proxies for $L(T, B, X_{\text{rel}})$.

At the ILASP level, the background knowledge B encodes the feature representation of each recipe (Section 4), while the language bias L specifies admissible weak-constraint

https://doi.org/10.1017/S1471068426100441 Published online by Cambridge University Press

patterns over these features (cf. Section 3.3). Brave ordering examples encode the pairwise preferences predicted by the neural network: for each sampled pair (i_1, i_2), an ordering example asserts that the answer sets extending i_1 should be preferred, tied, or less preferred than those extending i_2, depending on whether $B(i_1, i_2) = 1, 0, -1$ (with the ILASP-specific convention between $>$, $<$ and $\sim$, see equation (5)). The concrete ILASP encoding for the global case, including mode declarations and ordering examples, is reported in Appendix D.

A typical theory learned by ILASP in the global setting may look as follows.

Example 3.

```
:~ value(vegetables, V1).[-V1@1, V1]
:~ value(meat, V1).[-V1@2, V1]
:~ value(difficulty, V1).[-V1@3, V1]
:~ value(stewing, V1).[V1@4, V1]
:~ value(dairies, V1), category(3).[V1@5, V1]
```

Abstractly, this theory consists of weak constraints whose bodies mention simple conjunctions of feature atoms (such as `value(vegetables,V1)` or `value(dairies,V1), category(3)`) and whose weights and priority levels encode how these features contribute to the preference relation $\succ_{\mathtt{x}}$ defined in equation (5). Negative weights (`-V1`) reward higher values of the corresponding feature at a given priority level, while positive weights (`V1`) penalize them. In this example, the presence of vegetables and meat, as well as higher difficulty, tends to increase preference (at different priority levels), whereas the use of stewing and dairies in first-course recipes (category 3) is penalized.

From the perspective of preference learning (Section 3.1), the learned theory T induces an ordering over recipes by comparing the costs of their associated answer sets. When restricted to pairwise comparisons $T(\lambda_i, \lambda_j)$ as in equation (5), this yields a symbolic surrogate of the neural network's pairwise predictions. As discussed at the end of Section 3.3, the weak constraints in such theories can be systematically translated into natural-language statements, providing human-understandable explanations of the ranking logic captured by the black-box.

5.1.2 Local approximation

In the local case, we are interested in the reason that led the Neural Network to make a specific prediction for a query Q. The procedure is summarized in Figures 1 and 2b. As before, we sample N pairs (i_1, i_2) uniformly at random from the feature space, and we label them with the Neural Network predictions $B(i_1, i_2)$. For each generated pair Q, we also generate M associated pairs $(j_1, j_2)_m$ *sampled around* Q, and label them with Neural Network predictions $B(j_1, j_2)_m$. We chose to implement *sampling around* a pair $(j_1, j_2)_m$ as adding Gaussian noise to i_1 and i_2, but one can also use other noise models. Note that depending on the chosen noise model and feature type (categorical vs continuous), some post-processing may be necessary to project perturbed points back into the valid feature space, as discussed in Section 6.

https://doi.org/10.1017/S1471068426100441 Published online by Cambridge University Press

From the viewpoint of Section 3, local approximation corresponds to choosing, for each query Q, a metric d on X and a radius $r > 0$ such that the perturbations $(j_1, j_2)_m$ lie in the ball $B_r(Q) = \{x \in X \mid d(x, Q) \leq r\}$. ILASP is then asked to find a low-complexity theory T_Q that approximates B *restricted to* this local region, that is that nearly minimizes $L(T_Q, B, B_r(Q))$ under the complexity constraint $\Omega(T_Q) \leq \kappa$. In our implementation, the metric d is induced by π_Q defined below, and the brave ordering penalties are modulated by π_Q so that ILASP preferentially fits pairs closer to Q.

Let π_Q be a metric distance associated to the query Q, then for any (j_1, j_2) there is an associated $\pi_Q(j_1, j_2)$. Formally, we define $\pi_Q(j_1, j_2)$ as follows:

$$\pi_Q(j_1, j_2) = \sum_{k=1,2} \sqrt{\sum_f (d(j_k(f), i_k(f)))^2} \tag{6}$$

where $j_k(f)$ is the value of feature f for the item j_k, and $d(j_k(f), i_k(f))$ evaluates to $j_k(f) - i_k(f)$ when f is a continuous feature, whereas it evaluates to

$$d(j_k(f), i_k(f)) = \begin{cases} 0, & \text{if } j_k(f) = i_k(f) \\ 3, & \text{otherwise} \end{cases} \tag{7}$$

when f is a categorical feature. We chose 3 as an arbitrary value for the distance between categorical features, as ILASP works with integer weights, and 3 was a reasonable choice given the ranges of the dataset's continuous features. For each of the N queries, we use ILASP on the corresponding generated M pairs using an appropriate language bias L, by making ILASP prioritize theories that cover the pairs which are closer to the query (i.e., by putting `o_n` $= \pi_Q(j_1, j_2)$ on the corresponding `brave_ordering` of the pair (j_1, j_2), as detailed in Appendix D). Note that the original query is not included in the training set, but it is always included in the test set. Thus, differently from the global approximation, the cardinality of the test set is $K = N$, and the test samples are exactly the N pairs sampled at the start of the local sampling procedure. On the other hands, the training set size is identified by M, which is the number of pairs generated through Gaussian noise.

In summary, the global and local variants differ only in how the region A is instantiated (whole task region vs. neighborhood of a query) and how training examples are weighted or selected within ILASP. The underlying ILASP formalism (background knowledge, language bias, and use of brave orderings to encode preferences) remains the same, which makes the method amenable to other pairwise preference domains beyond recipes, provided that a structured feature representation similar to that in Section 4 is available.

5.2 Principal component analysis

We use PCA as the main method to reduce execution time while maintaining high performance (Abdi and Williams 2010). We make typical use of PCA to decrease the feature space size with little information loss. In fact, PCA is designed to store in the first Principal Components as much variance (of original data) as possible, and so to capturing as much information as possible. Considering the use of *Kaiser rule* for PC selection, this lead to a negligible loss of information. Note that this is also reflected by the results in Section 6, where we get an overall increase in estimators when using PCA rather than

https://doi.org/10.1017/S1471068426100441 Published online by Cambridge University Press

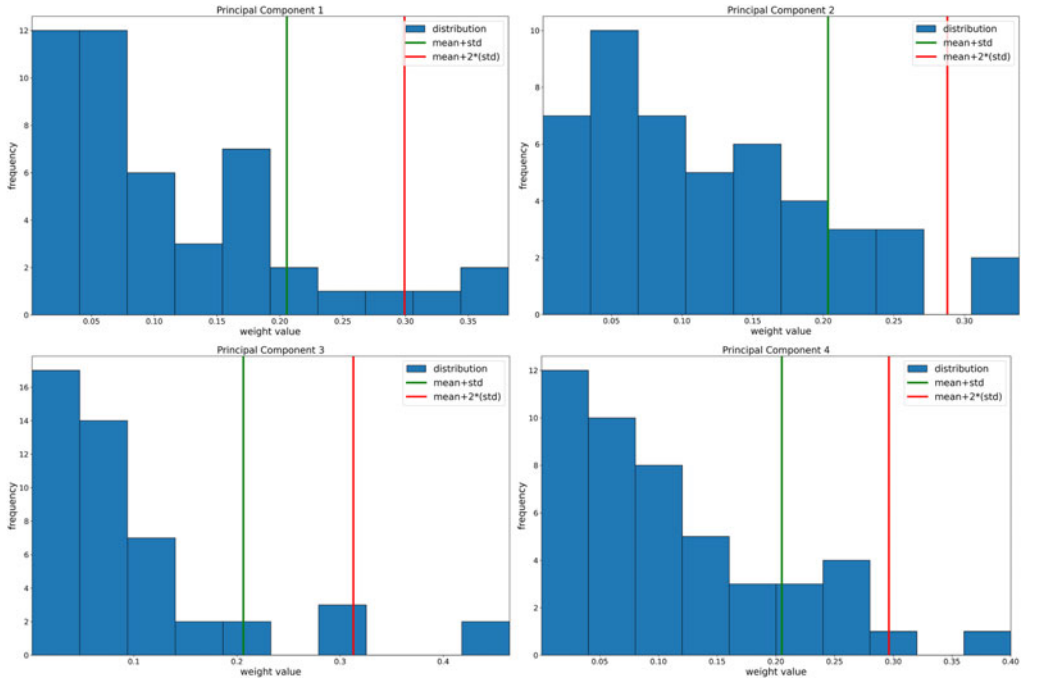


Fig. 3. Histogram of the absolute weights of the features for the first four principal components. When using *indirect PCA*, we select the features with weights in the i^{th} PC that are to the right of the red line.

not using at all. We explored two different techniques, which we refer to as *indirect* and *direct*.

In what we call *indirect PCA*, we determine the most important features among the first n Principal Components (PCs), so that we can reduce the number of features in the dataset accordingly. Let us consider the first n principal components obtained by applying the PCA on the dataset, and the weights w_{ij} that the feature i assigns to component j, then we only kept the features such that:

$$|w_{ij}| \geq \mu'_j + 2\sigma'_j \quad j = 1, \ldots, n \tag{8}$$

Where μ'_j and σ'_j are respectively the mean and the standard deviation of $|w_j|$. This means that, the features whose absolute weights are greater than the mean of the features' absolute weights plus two times the standard deviation, among the first n PCs, are those that we keep in the dataset.

In Figure 3 we show the histograms of the features' absolute weights for the first four PCs. Distributions of the PCs fit well with a $Gamma(k, \theta)$ distribution with shape parameter $k \approx 1.3$ and scale parameter $\theta \approx 0.1$ ($p < 0.05$ according to the Kolmogorov–Smirnov test for most PCs). This indicates that (8) selects approximately 5% of the features.

In the *direct* approach, the dataset is passed on to ILASP directly after the PCA, that is we pass the PCs as features. Differently from the *indirect* approach, the direct approach does not exclude any feature since, by the very definition of PCA, they all contribute

https://doi.org/10.1017/S1471068426100441 Published online by Cambridge University Press

towards each of the PC. It follows that the effectiveness of this technique is influenced by how and how many features are retro-projected from the PC involved in the weak constraint of the obtained theories. There are several ways to perform retro-projection; for instance, to remain consistent with the *indirect* approach and other experiments, we extrapolate features satisfying (8) with $n = 8$, that is we first select the principal components whose eigenvalues are at least 1 according to the Kaiser rule (the same criterion used in Section 4), and then, for each of these 8 PCs, we retain the original features whose weights are more than two standard deviations above the mean in absolute value. This threshold reflects the empirical Gamma-like distribution of weights in Figure 3 and typically results in a sparse subset of highly influential features per PC. In other words, indirect PCA drops low-loading features *before* ILASP training, shrinking the search space S_L and thus the optimization problem; direct PCA compresses the feature space into PCs for ILASP training and then uses retro-projection only at the *explanation* stage to map PCs back to a small set of original interpretable features. In both cases, the goal is to keep the resulting ILASP theory T both computationally tractable (short runtimes) and interpretable (small $\Omega(T)$ expressed in terms of domain-level attributes such as ingredient meta-classes and preparations).

It is worth noting that, while in the *indirect* approach we drop a subset of the original features before using ILASP, the retro-projection excludes features after ILASP application, that is when generating explanations. In fact, the explanation should include only the essential features needed to capture the variance in the data. This is necessary for two reasons: we want to keep the explanation concise, and we want to ensure that it is easily understandable to humans.

Finally, assuming that the trained black-box model is on hand and that, in the case of *direct* or *indirect* PCA, the corresponding mixing matrix or feature-selection mask has been retained, one can reply the overall proposed approach. In fact, one can generate arbitrary synthetic samples, apply either *direct* or *indirect* PCA, and query model for predictions. In turn, these data can be then used to generate a theory using ILASP, following the methodology described in Section 5.1.

5.3 User ground-truth scores

Symbolic artificial intelligence, such as ILASP, has a well-known advantage over black-box models. It can reason over the theory it generates, rather than opaquely making predictions on new data points. We use the transparent output theory to better evaluate ILASP performance.

As discussed in Section 3 (see in particular equation (5) and the subsequent discussion on fidelity (Guidotti *et al.* 2018)), we use *fidelity* to quantify how closely an interpretable model T mimics the predictions of the black box B on a given test region A. Operationally, fidelity is computed like an accuracy score, but with respect to B's outputs rather than ground-truth labels. While this makes the notion similar in form to standard accuracy, the underlying semantics differ: accuracy measures predictive performance against the true preference function $p_{\mathbf{x}}$, whereas fidelity measures agreement with B, regardless of whether B is itself accurate. Because there is no widely adopted naming convention for precision and recall in this context, we use Precision_{BB} and Recall_{BB} to denote precision and recall scores computed on ILASP's predictions against the outputs of the black box.

https://doi.org/10.1017/S1471068426100441 Published online by Cambridge University Press

Furthermore, as described in Section 5.1, each test set is sampled from the feature space. For this reason, we do not have ground-truth labels of test sets, except for those provided by the black-box model. This scenario is quite common in real-world use cases, where data may often be unlabeled. Thus, we propose the use of ground-truth data described in Table 2 to define $\text{Accuracy}_{\text{GT}}$, $\text{precision}_{\text{GT}}$, and $\text{recall}_{\text{GT}}$, that are score which help us to better understand the degree of generalization of our explanation. In fact, Fidelity, $\text{precision}_{\text{BB}}$ and $\text{recall}_{\text{BB}}$ tell us only how well the explanation is approximating the BB (which remain the main task of the paper), while these new score tell us how much this theory is in line with the preferences expressed by the users in the survey in general. We do this by comparing the labels assigned by the explanation returned by ILASP with those that would be assigned if ground-truth scores were used instead of the weights and the priority levels provided by ILASP in the explanation. It should be noted that there is no single canonical way to define how labels would be derived from ground truth in such a symbolic setting; the procedure we propose below is one principled way of mapping user scores to weak-constraint priorities so as to obtain a reference theory T_{GT} for comparison. Alternative mappings would be possible if different assumptions about priority scales or score ranges were adopted. Recall from Table 2 that recipe features are ranked by users in the integer range $[1, 10]$. For a user x and a feature f, we denote the ground-truth evaluation of f according to x by $G_x(f)$. Then, given a theory T returned by ILASP after its application, which is composed of only weak constraints, for each feature $f \in T$ we define a weak constraint $wc_{GT}(f)$ as:

$$:\sim \texttt{value}(f,\ \texttt{V1})\,\texttt{[V1@}|m(\bar{G}_x(f))|\texttt{,\ V1]}. \qquad \text{if } m(\bar{G}_x(f)) > 0$$
$$:\sim \texttt{value}(f,\ \texttt{V1})\,\texttt{[-V1@}|m(\bar{G}_x(f))|\texttt{,\ V1]}. \qquad \text{if } m(\bar{G}_x(f)) < 0$$

where:

$$\bar{G}_x(f) = (G_x(f) - 5)(-1) \tag{9}$$

$$m(\bar{G}_x(f)) = \begin{cases} \bar{G}_x(f) & if\ \ \bar{G}_x(f) > 0 \\ \bar{G}_x(f) - 1 & otherwise \end{cases} \tag{10}$$

the mapping function $m(\bar{G}_x(f))$ for $G_x(f)$ is used to align with ILASP's weak constraint semantics. Specifically, $G_x(f)$ belongs to the range [1, 10], where a value of 1 means that the user x absolutely dislikes f, while a value of 10 indicates a strong preference for f. Such score is used to define weight sign and priority level of the new weak constraints, assigning progressively higher priorities to the weak constraint if $m(\bar{G}_x(f)) > 0$ and vice versa, with the rationale that the score given by the user implicitly define a priority level among features. To define the priority level of the weak constraint, we use the mapping function $m(\cdot)$. Since $G_x(f) \in [1, 10]$ and we need a priority level in $[1, 5]$(since $\texttt{\#maxp} = 5$), discerning between positive and negative preferences (i.e., negative and positive weight sign with regard to Section 3.2) we define $m(\cdot)$ on $\bar{G}_x(f)$. $\bar{G}_x(f)$ is obtained centering $G_x(f)$ around zero and inverting the sign, so that $\bar{G}_x(f) \geq 6$ results in negative weights in weak constraint, and vice versa. In our case, working with $\texttt{\#maxp} = 5$, we simply set $m(\bar{G}_x(f) = \bar{G}_x(f))$ if $\bar{G}_x(f) > 0$. However, since it wouldn't make sense to assign a priority level of zero when $\bar{G}_x(f) = 0$ (thus $G_x(f) = 5$), we shift by one whenever $\bar{G}_x(f) \leq 0$. Note

https://doi.org/10.1017/S1471068426100441 Published online by Cambridge University Press

that the resulting priority is set in absolute value to avoid negative priority levels. It's important to note that the choice of $m(\cdot)$ is not unique and can be adapted depending on the context (e.g., with a different value of `#maxp`).

Finally, given a theory T, we define $T_{GT} = GT(T)$, where:

$$GT(T) = \{wc_{GT}(f) | f \in T\} \tag{11}$$

So, accuracy$_{\text{GT}}$, precision$_{\text{GT}}$ and recall$_{\text{GT}}$ are calculated comparing the label assigned by the explanation T given by ILASP after its application with the label assigned by $T_{GT} = GT(T)$, following the cost concept described in Section 3.2.

To better understand the process, consider the following example: Let $T = \{:\sim \texttt{value(browning, V1)[V1@1, V1].}\}$ (where *browning* is a preparation of a recipes, as described in Table 1) be the theory returned by ILASP, then $\mathcal{F} = \{f | f \in T\} = \{browning\}$. Let be $G_x(browning) = 3$, then $\bar{G}_x(browning) = (3-5)(-1) = 2$, and so $m(\bar{G}_x(browning)) = 2$. So, $T_{GT} = GT(T) = \{:\sim \texttt{value(browning, V1)[V1@2, V1].}\}$. Note that while in T the feature *browning* is placed at the first level of priority, in T_{GT} it appears at the second level. As we will show later in this example, this does not necessarily mean that T and T_{GT} will produce different results. Consider the sample, $s = (r_1, r_2)$ which is a pair of recipes we intend to classify with regard to user x preferences. Since our theories are composed only of the feature "browning," let's focus only on that feature. Let's say that $r_1(browning) = 3$, while $r_2(browning) = 1$ and consider the theory T, then we get the cost vectors $p_T(r_1) = [0\ 3]$ and $p_T(r_2) = [0\ 1]$. So, with regard to the theory T, the user x prefers r_2 over r_1 (and so assign the label 1). Consider now the theory T_{GT}, then we get $p_{T_{GT}}(r_1) = [3\ 0]$ and $p_T(r_2) = [1\ 0]$, which return the same label, confirming that theory T weight and priority level assignments are in line with the ground-truth available data.

Consider the previous example, but now suppose that $G_x(browning) = 7$, then $\bar{G}_x(browning) = (7-5)(-1) = -2$ and so $m(\bar{G}_x(browning)) = -3$. Finally we obtain that $GT(T) = \{:\sim \texttt{value(browning, -V1)[V1@2, V1].}\}$. In this case $p_{T_{GT}}(r_1) = [-3\ 0]$ and $p_T(r_2) = [-1\ 0]$, and so returned label would be -1, disconfirming that theory T weight and priority level assignments are in line with the ground-truth available data.

6 Results

In this section, we answer to the research question made in Section 1, and present the results of our proposed methods. Out of all the users, we focused on 10 of them whose corresponding neural networks had the best accuracies. The results involve the use of ILASP both as a global and local approximator and the PCA as a *direct* and *indirect* method.

To further decrease dimensionality, we considered different ingredient granularities, that is standard classes for neural networks and meta-classes for ILASP as illustrated in Table C1. The results report on average fidelity, precision $_{\text{BB}}$, recall $_{\text{BB}}$, and execution time for each case, as well as the average number of weak constraints returned by ILASP after its application and the average accuracy$_{\text{GT}}$, precision$_{\text{GT}}$, recall$_{\text{GT}}$.

https://doi.org/10.1017/S1471068426100441 Published online by Cambridge University Press

The dataset is unbalanced, with an average of 17.44% of pairs labeled as *uncertain* for each user. This affects the neural network's performance, reducing precision and recall, while accuracy is less impacted as it accounts for both true positives and true negatives.

Experiments have been made on machine with AMD Ryzer 9 5900X CPU, 32 GB RAM, GeForce RTX 3080 GPU.

Global approximation. In the global approximation approach we considered training sets of different sizes, namely with $N = 45$, 105 and 190 pairs; on the other hand, the test set (which is the same for all three training sets) has $K = 105$ pairs.

Local approximation. In the local approximation approach, we uniformly sampled a set of $N = 100$ pairs and then generated a training set for each pair in this set, that is $M = 45$, 105 and 190. The results are calculated as average performance on pairs for each user. As described in 5.1.2, the generated training set are obtained by adding Gaussian noise. To examine how ILASP behaves under various degrees of locality, we carried out experiments using different standard deviation values for the Gaussian noise. Specifically, we tested for σ values of 1, 0.1, 0.01, and 0.001.

It must be noted that in our dataset the recipe features are non-negative float numbers (see Table 1). However, since `clingo` only supports integers, we must preprocess data. We first shift the range of the values so that 0 becomes the minimum. This keeps data consistent with the initial dataset, as Gaussian noise can introduce negative values. Next, we generate the Gaussian noise with a standard deviation equal to 0.001, 0.01, 0.1, or 1, as displayed in Tables 7, 8, and 9 in Section 6.3. To reason over integers as required by `clingo`, we must round up the obtained values. However, due to the magnitude of noise (for standard deviation equal to 0.001, 0.01, and 0.1), rounding up typically results in discarding the changes made. For this reason, we multiplied the values by a factor inversely proportional to the standard deviation, so that the system is sensitive to all variations. For instance, for standard deviation 0.1, we multiply by 10; for standard deviation 0.01, we multiply by 100, and so forth.

Indirect PCA. In the *indirect* approach, we experimented with n equal to 8 and 17. We chose two different n to conduct more complete study, comparing *indirect* approach with different value of n. We specifically chose 17 as it is the number of Principal Components involved during the training of neural networks, and 8 as it is its rounded half. Note that 17 PCs explain about 76% of the dataset and preserve 34 features, while 8 PCs explain about 49% of the dataset and preserve 15 out of 48 original features.

Direct PCA. In the *direct* approach, we tested the case with 5, 10, 15 and 20 PCs. Finally, following discussion from Section 6.1, `#maxp` was set to 5.

6.1 Preliminary analysis

To benchmark ILASP performance, we conducted a study to understand the role of ILASP hyperparameters in terms of *accuracy*, *precision*, *recall* and *execution time*. Since these experiments usually require a long execution time, we focused only on `#maxp` (maximum number of weak constraints in a theory), setting `#maxv = 1` (maximum number of variables for weak constraints in a theory). To further fix environmental conditions, we ran experiments in the hard case of global approximation, indirect PCA, and $n = 8$.

https://doi.org/10.1017/S1471068426100441 Published online by Cambridge University Press

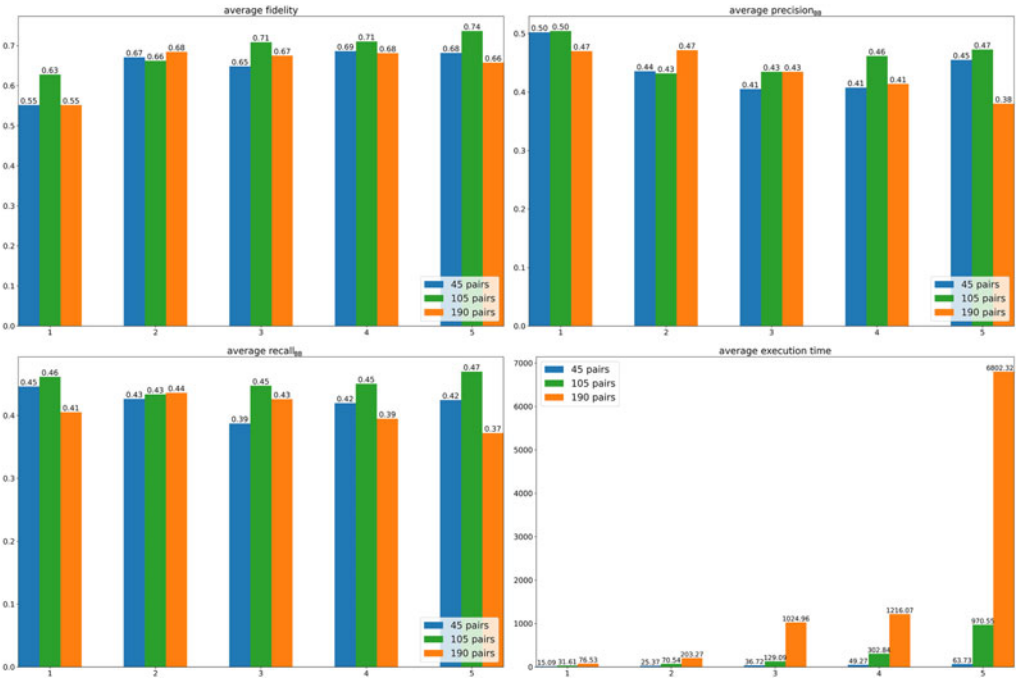


Fig. 4. Global fidelity, precision $_{BB}$, recall $_{BB}$, and execution time results for `#maxp` between 1 and 5, *indirect* PCA.

As it can be seen in Figures 4, the execution time grows exponentially both in the size of the dataset and `#maxp`. As can be seen from these figures, `#maxp` also affects fidelity, precision $_{BB}$, and recall $_{BB}$. With the increase of `#maxp`, fidelity rises by approximately 11 to 13%, while precision$_{BB}$ and recall$_{BB}$ generally decline by around 2 to 9%. However, overall, we did not notice a significant trend in the observed metrics. It is worth noting that better results were generally obtained with the training set consisting of 105 pairs, compared to the one with 190 pairs. To understand the reasons behind these results, we analyzed the similarity between the sampled training sets used for ILASP and the dataset on which the black-box model was trained, using the Maximum Mean Discrepancy (MMD) metric (Gretton *et al.* 2012). This analysis revealed that the training set with 105 pairs is more similar to the neural network's training data, whereas the set with 190 pairs is less similar. This suggests that ILASP may perform less effectively when applied on the set with 190 pairs, possibly due to inconsistencies in the ground truth derived from the neural network, which could be more prone to errors when classifying unfamiliar pairs. Importantly, this outcome seems to be a coincidence, as the training sets were randomly sampled. Considering the differences in execution time, particularly with the dataset containing 190 pairs, the best choice for the experiments would be to set $\#\texttt{maxp} = 4$, as it offers the optimal balance between execution time and performance. However, in the remainder of this paper we set $\#\texttt{maxp} = 5$, as we were interested in complex theories; nonetheless, other choices for `#maxp` are possible, and Figures 4 serves as a useful guide to help the reader select the most appropriate value depending on the specific application.

https://doi.org/10.1017/S1471068426100441 Published online by Cambridge University Press

6.2 Global approximation

General results. The results of ILASP globally approximating the neural network on the three different training sets are reported in Tables 3, 4 and 5. Tables show that it is possible to make the black-box model globally transparent, achieving a fidelity of approximately 70%, thereby **Q1** and **Q2**. As expected, using PCA decreases the computational time. What is most interesting is the fact that both *direct* and *indirect* PCA approaches lead to an overall increase in estimators compared to the scenario without PCA. However, in terms of user ground-truth scores, we generally achieve superior results only in *indirect* PCA approach, while *direct* remain stable only on $\text{accuracy}_{\text{GT}}$.

Scalability of the method. In terms of execution time, as visible in Figure 5, there is a significant difference between the different approaches: *indirect* PCA reduce the execution time but maintain the exponential growth between datasets (similarly to the case without PCA), while in the *direct* PCA the time increase is more like to be linear. Additionally, the *direct* PCA approach generally provides better estimators. Although counterintuitive, the difference in results between the *direct* approach and other cases cannot be attributed to a greater decrease in feature space, as *indirect* PCA involves only 15 features (when $n = 8$), whereas the *direct* approach involves 20 features. This outcome results directly from PCA's definition as a change of basis: it re-expresses the original feature space in a new orthonormal basis optimized to capture variance Abdi and Williams (2010). When ILASP is applied using the *direct* PCA case, it has a simpler feature space compared to the starting one. This assumption does not hold in the case of *indirect* PCA, where dimensionality reduction simply involves removing features that do not meet the criteria defined by (8).

Overall, the reported results indicate that both the indirect and direct approaches successfully enhance the scalability of the method, thus answering **Q3**.

Performances of PCA approaches. In the *direct* PCA approach, we generally observe a gradual improvement in fidelity, $\text{precision}_{\text{BB}}$, and $\text{recall}_{\text{BB}}$ up to 10 principal components (PC), after which they decrease. This could be a form of early overfitting caused by the fact that the combined use of the *direct* PCA method and the predictions of the neural network as labels oversimplifies the feature space and ground-truth. Moreover, the theory returned in *direct* cases have less complexity, reaching the average max number of weak constraint of 1.9. As it is visible from the tables, we obtain the worse ground-truth scores, with only $\text{accuracy}_{\text{GT}}$ being comparable to other cases. This result could be attributed to a combination of the two earlier suppositions: on one hand, the simplified feature space encourages the development of streamlined theories (with fewer weak constraints), and on the other, the early overfitting leads to theories that are highly tailored to the training set, potentially limiting their ability to fully capture the ground-truth.

On the other hand, *Indirect* PCA approach obtains more complex theory and better ground-truth scores, both generally better than the results obtained without the use of the PCA. Furthermore, even if we do not get the same time reduction as with the *direct* approach, it remains significant, halving the execution time or more.

Lastly, it can be seen that the *indirect* PCA approach yields better estimators when we set $n = 8$ instead of 17, as well as compared to the case in which PCA is not involved at all. On the other hand, $\text{Accuracy}_{\text{GT}}$ is generally better when $n = 17$, indicating that

https://doi.org/10.1017/S1471068426100441 Published online by Cambridge University Press

https://doi.org/10.1017/S1471068426100441 Published online by Cambridge University Press

Table 3. *ILASP as global approximator on training set of 45 pairs*

	Fidelity	Precision_{BB}	Recall_{BB}	Time(s)	#WC	accuracy_{GT}	precision_{GT}	recall_{GT}
No PCA	66.60%	37.74%	42.62%	72.27	3.5	58.79%	50.04%	48.56%
Indirect (8 PC)	68.19% (+1.59%)	**45.49% (+7.75%)**	42.45% (−0.17%)	63.73 (−8.54)	**3.8 (+0.3)**	**70.98% (+12.19%)**	**64.09% (+14.05%)**	51.78% (+3.22%)
Indirect (17 PC)	67.68% (+1.08%)	37.87% (+0.13%)	37.86% (−4.76%)	68.55 (−3.72)	3.5 (–)	65.65% (+6.86%)	59.99% (+9.95%)	**56.56% (+8.00%)**
Direct (5 PC)	**70.98% (+4.38%)**	44.55% (+6.81%)	**49.25% (+6.63%)**	**1.83 (−70.44)**	1.3 (−2.2)	51.49% (−7.30%)	22.39% (−27.65%)	26.35% (−22.21%)
Direct (10 PC)	70.22% (+3.62%)	41.96% (+4.22%)	41.99% (−0.63%)	4.50 (−67.77)	1.6 (−1.9)	57.59% (−1.20%)	28.09% (−21.95%)	28.29% (−20.27%)
Direct (15 PC)	68.95% (+2.35%)	41.96% (+4.22%)	44.35% (+1.73%)	6.62 (−65.65)	1.4 (−2.1)	58.98% (+0.19%)	31.46% (−18.58%)	34.17% (−14.39%)
Direct (20 PC)	69.02% (+2.42%)	41.35% (+3.61%)	40.49% (−2.13%)	8.91 (−63.36)	1.4 (−2.1)	58.41% (−0.38%)	31.05% (−18.99%)	34.25% (−14.31%)

https://doi.org/10.1017/S1471068426100441 Published online by Cambridge University Press

Table 4. *ILASP as global approximator on training set of 105 pairs*

	Fidelity	Precision$_{BB}$	Recall$_{BB}$	Time(s)	#WC	accuracy$_{GT}$	precision$_{GT}$	recall$_{GT}$
No PCA	71.62%	41.39%	42.11%	6102.15	5.0	57.90%	47.66%	35.27%
Indirect (8 PC)	**73.65% (+2.03%)**	42.26% (+0.87%)	46.97% (+4.86%)	970.54 (−5131.61)	4.9 (−0.1)	60.95% (+3.05%)	**50.54% (+2.88%)**	39.60% (+4.33%)
Indirect (17 PC)	71.81% (+0.19%)	44.67% (+3.28%)	42.97% (+0.86%)	2617.40 (−3484.75)	**5.0 (−)**	**61.71% (+3.81%)**	46.84% (−0.82%)	**40.70% (+5.43%)**
Direct (5 PC)	71.81% (+0.19%)	44.49% (+3.10%)	48.89% (+6.78%)	**3.67 (−6098.48)**	1.4 (−3.6)	50.29% (−7.61%)	19.64% (−28.02%)	27.64% (−7.63%)
Direct (10 PC)	72.44% (+0.82%)	45.12% (+3.73%)	**49.14% (+7.03%)**	12.83 (−6089.32)	1.6 (−3.4)	57.21% (−0.69%)	23.61% (−24.05%)	27.61% (−7.66%)
Direct (15 PC)	70.22% (−1.40%)	43.35% (+1.96%)	46.97% (+4.86%)	23.88 (−6078.27)	1.5 (−3.5)	59.43% (+1.53%)	26.18% (−21.48%)	31.05% (−4.22%)
Direct (20 PC)	71.11% (−0.51%)	**47.63% (+6.24%)**	43.61% (+1.50%)	30.85 (−6071.30)	1.5 (−3.5)	58.29% (+0.39%)	24.89% (−22.77%)	29.85% (−5.42%)

https://doi.org/10.1017/S1471068426100441 Published online by Cambridge University Press

Table 5. *ILASP as global approximator on training set of 190 pairs*

	Fidelity	Precision_{BB}	Recall_{BB}	Time(s)	#WC	accuracy_{GT}	precision_{GT}	recall_{GT}
No PCA	64.95%	35.81%	35.47%	22210.65	5.0	65.78%	44.93%	42.98%
Indirect (8 PC)	65.78% (+0.83%)	38.00% (+2.19%)	37.20% (+1.73%)	6802.32 (−15408.33)	**5.0** **(–)**	63.56% (−2.22%)	**54.70%** **(+9.77%)**	**49.53%** **(+6.55%)**
Indirect (17 PC)	65.59% (+0.64%)	37.31% (+1.50%)	37.15% (+1.68%)	12106.05 (−10104.60)	**5.0** **(–)**	**65.65%** **(−0.13%)**	53.96% (+9.03%)	44.86% (+1.88%)
Direct (5 PC)	67.17% (+2.22%)	37.79% (+1.98%)	38.44% (+2.97%)	**8.74** **(−22201.91)**	1.6 (−3.4)	52.13% (−13.65%)	25.23% (−19.70%)	30.38% (−12.60%)
Direct (10 PC)	**69.33%** **(+4.38%)**	**42.03%** **(+6.22%)**	**40.32%** **(+4.85%)**	25.03 (−22185.62)	1.8 (−3.2)	59.49% (−6.29%)	28.49% (−16.44%)	32.13% (−10.85%)
Direct (15 PC)	66.92% (+1.97%)	36.64% (+0.83%)	36.87% (+1.40%)	49.58 (−22161.07)	1.9 (−3.1)	58.98% (−6.80%)	25.89% (−19.04%)	34.54% (−8.44%)
Direct (20 PC)	65.71% (+0.76%)	35.44% (−0.37%)	35.33% (−0.14%)	60.06 (−22150.59)	1.9 (−3.1)	63.68% (−2.10%)	30.52% (−14.41%)	36.52% (−6.46%)

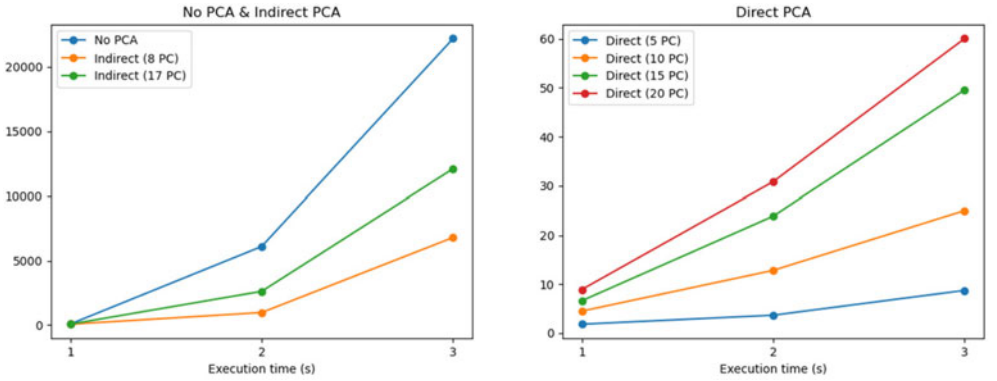


Fig. 5. Execution time comparison for global approximation across No PCA, Indirect PCA, and Direct PCA. Indirect PCA preserves the exponential growth observed with No PCA, albeit at a reduced rate; Direct PCA instead exhibits an approximately linear scaling, with the slope decreasing as fewer principal components (PCs) are retained.

the higher the PC considered, the better the degree of generalization of the obtained theories. The better results in ground-truth scores suggest that by reducing the dataset's complexity through *indirect* PCA, ILASP can concentrate on the neural network's most salient predictive information. In fact, PCA is specifically designed to preserve maximal variance, thereby filtering out noise and redundant features. This ensure that the subsequent rule-learning phase operates on a distilled representation of the model's behavior. As a result, ILASP generates rules that capture the network's core decision criteria without being distracted by spurious correlations or minor fluctuations in less significant feature. Thus, the outcome is a more accurate explanatory theory that aligns closely with the true user preferences reflected in the ground-truth data. All this evidence allows us to conclude that PCA-based approaches not only enable us to address **Q3**, but also contribute to providing more robust results for **Q2**.

ILASP as classifier and PCA hypothesis. The results in Table 6 show the outcomes achieved by using ILASP directly on the user-labeled preferences in the dataset rather than using neural network predictions. Note that in this case, the training set has 157 pairs, while the test set only 53 (since we have 210 couples for each user in the preferences' dataset). As can be seen, even with *indirect* PCA approach with $n = 8$ we obtain an execution time that is significantly greater of the case in which we do not use PCA at all when using ILASP as global approximator on dataset with 190 pairs. On the other hand, the *direct* approach has an execution time comparable to that of the global approximation case, and generally provides better estimators. Specifically, when using PCA *directly* the increase of results do not stop at 10 PC as before, but until 20. These results support the two hypothesis regarding the use of *direct* PCA approach: in this case, there is not the simplification induced by the approximation made by neural network, thus the feature space given to ILASP is more complex than before. This reflects in the absence of overfitting in *direct* approach. Furthermore, even if the *indirect* PCA execution time increase with respect to the global case, the *direct* PCA approach is not, strengthening the idea that its execution time reduction is given by the orthonormal basis created by PCA.

https://doi.org/10.1017/S1471068426100441 Published online by Cambridge University Press

https://doi.org/10.1017/S1471068426100441 Published online by Cambridge University Press

Table 6. *ILASP as classifier on preferences' training set of 157 pairs*

	Accuracy	Precision	Recall	Time(s)	#WC	accuracy$_{GT}$	precision$_{GT}$	recall$_{GT}$
Indirect (8 PC)	**77.52%**	**48.99%**	**50.86%**	155306.17	**5.0**	60.13%	**42.67%**	**42.32%**
Direct (5 PC)	71.44% (−6.08%)	42.05% (−6.94%)	36.96% (−13.90%)	**5.51 (−155300.66)**	1.7 (−3.3)	54.34% (−5.79%)	26.54% (−16.13%)	26.94% (−15.38%)
Direct (10 PC)	71.80% (−5.72%)	42.24% (−6.75%)	38.03% (−12.83%)	24.69 (−155281.48)	1.9 (−3.1)	**62.64% (+2.51%)**	29.30% (−13.37%)	34.19% (−8.13%)
Direct (15 PC)	72.09% (−5.43%)	42.24% (−6.75%)	38.17% (−12.69%)	40.45 (−155265.72)	1.9 (−3.1)	59.87% (−0.26%)	29.89% (−12.78%)	32.99% (−9.33%)
Direct (20 PC)	72.41% (−5.11%)	42.81% (−6.18%)	37.79% (−11.07%)	58.77 (−155247.40)	2.0 (−3.0)	58.87% (−1.26%)	25.77% (−16.90%)	34.35% (−7.97%)

https://doi.org/10.1017/S1471068426100441 Published online by Cambridge University Press

Table 7. *ILASP as local approximator on training set of 45 pairs*

std	Fidelity	Precision$_{BB}$	Recall$_{BB}$	Time(s)	#WC	accuracy$_{GT}$	precision$_{GT}$	recall$_{GT}$
1	64.59%	41.31%	36.43%	75.61	**2.36**	62.50%	49.28%	41.49%
0.1	83.54% (+18.95%)	63.40% (+22.09%)	60.81% (+24.38%)	46.30 (−29.31)	1.33 (−1.03)	**63.20% (+0.70%)**	**52.87% (+3.59%)**	**45.83% (+4.34%)**
0.01	96.97% (+32.38%)	**71.61% (+30.30%)**	63.40% (+26.97%)	4.61 (−71.00)	1.02 (−1.34)	59.64% (−2.86%)	52.40% (+3.12%)	33.49% (−8.00%)
0.001	**97.97% (+33.38%)**	66.98% (+25.67%)	**68.22% (+31.79%)**	**4.26 (−71.35)**	1.02 (−1.34)	58.83% (−3.67%)	41.72% (−7.56%)	31.84% (−9.65%)

https://doi.org/10.1017/S1471068426100441 Published online by Cambridge University Press

Table 8. *ILASP as local approximator on training set of 105 pairs*

std	Fidelity	Precision_{BB}	Recall_{BB}	Time(s)	#WC	accuracy_{GT}	precision_{GT}	recall_{GT}
1	65.82%	42.66%	36.60%	960.43	**2.57**	62.04%	45.30%	39.46%
0.1	86.09% (+20.27%)	66.01% (+23.35%)	67.02% (+30.42%)	192.16 (−768.27)	1.43 (−1.14)	**64.05% (+2.01%)**	**56.56% (+11.26%)**	**46.09% (+6.63%)**
0.01	96.77% (+30.95%)	66.84% (+24.18%)	66.05% (+29.45%)	4.47 (−955.96)	1.06 (−1.51)	58.81% (−3.23%)	51.17% (+5.87%)	33.17% (−6.29%)
0.001	**98.06% (+32.24%)**	**68.32% (+25.66%)**	**68.43% (+31.83%)**	**3.89 (−956.54)**	1.00 (−1.57)	58.09% (−3.95%)	37.81% (−7.49%)	29.49% (−9.97%)

Table 9. *ILASP as local approximator on training set of 190 pairs*

std	Fidelity	Precision$_{BB}$	Recall$_{BB}$	Time(s)	#WC	accuracy$_{GT}$	precision$_{GT}$	recall$_{GT}$
1	–	–	–	–	–	–	–	–
0.1	84.40%	62.10%	60.40%	401.62	**1.48**	**61.23%**	52.10%	**43.11%**
0.01	96.53% (+12.13%)	**68.88% (+6.78%)**	62.10% (+1.70%)	**7.81 (−393.81)**	1.07 (−0.41)	59.53% (−1.70%)	**54.53% (+2.43%)**	34.95% (−8.16%)
0.001	**98.03% (+13.63%)**	65.26% (+3.16%)	**65.02% (+4.62%)**	1101.55 (+699.93)	1.02 (−0.46)	59.65% (−1.58%)	46.11% (−5.99%)	31.83% (−1.28%)

6.3 Local approximation

In our experiment using ILASP for local approximation, we tested different standard deviation values for the Gaussian noise applied to the queries. Such experiment concerns *indirect* PCA approach with $n = 8$, this because it is the case in which we obtain the best trade-off between estimators, execution time and complex theories in the global approximation. In fact, as described in Section 6.2, *direct* PCA approach generally has better estimators and execution time results, but worse ground-truth results and simpler theories. On the other hand, *indirect PCA* halves the execution time, and still obtain an overall increase in all the estimators, which is more marked when $n = 8$. Before to start to show and comment results, must be reported that at average of 5–10% couples of test set have empty theory as explanation returned by ILASP. Such cases occurred when the training set mainly consisted of couples labeled with uncertainty (about 90–95% of the couples in the training set). ILASP seeks to minimize the length of a theory plus the cost of not covering examples, but due to the limitation in the local range, it struggles to define a theory that properly covers couples with uncertainty, leading to empty set as a result. Note that, as fully described later, it is possible to obtain a training set with over 90% of couple with the same label when there is a high degree of locality (e.g., std = 0.001).

Locality degree and general results. The results are reported in Tables 7, 8 and 9. Any instances where the returned theory is empty have not been included in the table. This decision was made because it is unclear how to test an empty theory, and these cases only account for 5–10% of the total. By definition of Gaussian noise, the choice of standard deviation affects the grade of similarity of generated data respect to the original query, as well as the labels predicted by the Neural Network, which are then fed to ILASP to validate the learned theory as an approximation. A lower standard deviation increases the likelihood that elements of the training set are labeled with the same value, making ILASP's task simpler. Conversely, a higher standard deviation presents ILASP with a more challenging task. This trend is reflected in the results: as the standard deviation decreases, there is an increase in fidelity, precision $_{BB}$, and recall $_{BB}$, and a decrease in execution time. However, this pattern does not hold when the training set comprises 190 pairs, as the execution time explodes with standard deviation = 0.001. As this is an isolated case with no apparent justification for the explosion, it is treated as an outlier. On the other hands, lesser is the standard deviation, lesser are the differences between

https://doi.org/10.1017/S1471068426100441 Published online by Cambridge University Press

generated data respect to the original. This means that are also lesser the weak constraints that ILASP can extrapolate from a training set, leading to less complex theory (and so with lesser #WC), as visible in the results. Finally, note that results for std = 1 are omitted from Table 9 because using ILASP on all pairs of the full training set became prohibitively time-consuming.

Lastly, ground-truth score yields better results when std is 0.1, and worse at lowers std. This could suggest that considering a locality degree too small can lead to overfitting. For this reason, and since we consider it the best trade-off between estimators, execution time, and number of weak constraints, we choose a standard deviation of 0.1 for the next experiments using local approximator.

Once more, the results presented in the tables, along with the supporting evidence, confirm positive answers to **Q1** and **Q2**, as will be further clarified later in this section.

Global vs local execution time. When standard deviation is 0.1, training set has 190 pairs and *indirect* PCA is used with $n = 8$, our method lead to an average execution time of only 401.62 s for a single query. This is significantly lower than the corresponding time of 6802.32 s for global approximation. But when using ILASP as a global approximator, the returned theory can be used to predict the number of pairs needed, whereas for local approximation, the theory can only be used for the specific query ILASP was used on. In the given example, local approximator need 401.62 s for each pair, while the test set used for global approximator has 105 pairs. This means that the local approximator would take 42170.1 s to approximate all the pairs in the test set, compared to the 6802.32 s needed by the global approximator. However, the theories returned by the local approximator are more specific to individual pairs, resulting in significantly better fidelity, precision $_{BB}$, and recall $_{BB}$ compared to the global approach. In conclusion, also considering **Q3**, the global approximator appears to be more efficient than the local one, while the local approach is more effective than the global one.

Local approximation performances. The result discussed in previous paragraph is also visible in Table 10, where are reported the result of ILASP as local approximator both when *indirect* PCA approach is applied (with $n = 8$, 17) and when is not, with standard deviation equal to 0.1. We did not consider *direct* approach because of the already too little complex theory, which would become further simplified by the use of this technique, as seen in global approximation study. In terms of fidelity, precision $_{BB}$ and recall $_{BB}$, these results behave similarly to what seen in the global case, but now the increase in significantly greater, answering to **Q2** also from the local perspective. In fact, while in the global case the greater difference with the *indirect* PCA was of 2.03%, here we get a difference of about 6.20% with $n = 8$ and 14.29% with $n = 17$. In our opinion, this significant difference is justified because local approximators generally seek less complex theories, requiring fewer features (e.g., *browning* preparation) to discriminate preferences among recipes. Thus, *indirect* PCA improves results by effectively helping ILASP select necessary features and removing those with minor roles. Conversely, in the global case, ILASP seeks more complex theories where dimensionality reduction only marginally enhances performance, as even minor role features remain important for general decision-making. Put simply, in the local case we get less complex but more accurate theories, which are better enhanced with respect to the global case (where we necessarily need more complex theories) by the dimensionality reduction.

https://doi.org/10.1017/S1471068426100441 Published online by Cambridge University Press

Table 10. *ILASP as local approximator on training set of 105 pairs (std = 0.1)*

	Fidelity	Precision_{BB}	Recall_{BB}	Time(s)	#WC	accuracy_{GT}	precision_{GT}	recall_{GT}
No PCA	79.89%	62.62%	60.52%	216.17	1.60	64.60%	51.96%	46.80%
Indirect (8 PC)	86.09% (+6.20%)	**66.01% (+3.39%)**	67.02% (+6.50%)	**192.16 (−24.01)**	1.43 (−0.17)	**64.05% (−0.55%)**	**56.56% (+4.60%)**	46.09% (−0.71%)
Indirect (17 PC)	**94.18% (+14.29%)**	65.84% (+3.22%)	**69.18% (+8.66%)**	318.04 (+101.87)	**1.49 (−0.11)**	63.76% (−0.84%)	51.17% (−0.79%)	**46.72% (+0.92%)**

7 Conclusion and future developments

In recent years, XAI has emerged as a critical discipline to make machine learning models transparent and interpretable, delivering tangible benefits in domains such as healthcare, finance and recommendation systems. In the specific context of preference learning, the ability to explain a black-box model's rankings not only help end users to understand its decision, but also enables bias detection and performance improvement through an iterative feedback cycle. In this context, a promising approach in this field is that of Inductive Logic Programming, a subfield of artificial intelligence which aims to generate intuitive and readable explanation of the decision-making of black box, which can be fed even to lay users. In this work we propose several approaches for using of ILP to explain black-box preference learning system while dealing with the issue of time complexity. In fact, among the challenges with XAI methods (Saeed and Omlin 2023), Scalability is one of the most pressing ones, caused by increasingly dimensionality in data or cases for which huge number of explanation are needed. Specifically, we propose ILASP as global and local approximator of a Neural Network involving PCA in the process to reduce the dimensionality of the dataset and experimenting by sampling dataset of increasing size from dataset feature space. This not only led to a significant reduction in execution time, but also helps to improve results, thanks to the ability of PCA to compress great part of the informative content of the dataset in the firsts Principal Components.

Within this work, we also created a food preference dataset, which is at the base of the experiments. This dataset is based on a survey where participants ranked Italian recipes. The results of the survey are then processed in order to obtain the data used to encode the ranking problem as pairwise comparison, including the uncertainty scenario. Another challenge of XAI in which many efforts are invested, is to obtain methods with better fairness. In fact, a given explanation that entail a prediction might not be the correct one anyway, as the user could have another reason. One way to turn-around these problems is the use of ground-truth to validate the results. To this end, we defined ground-truth scores to evaluate theories generated by ILASP. In this way, we can assess ILASP performances as approximator using standard metric in XAI like fidelity (Guidotti *et al.* 2018), and as faithful representations of user's actual preferences with aforementioned ground-truth scores.

As mentioned in our paper, a key strength of ILASP lies in its ability to produce theories that can be readily rendered into natural language. In future work, we plan

https://doi.org/10.1017/S1471068426100441 Published online by Cambridge University Press

to enhance this capability leveraging either ad-hoc parser or prompt-engineered large language models to translate ILASP's output. This will enable lay end users to directly assess both the accuracy and the interpretability of the generated theories.

The resulting dataset is unbalanced on uncertainty class, and this reflected is visible in the final results. In future works, we aim to address this issue. An example to deal with it is by considering sampling techniques during training and test sets definition both for global and local experiments (e.g., by taking into account only sets of couple which labels given by the Neural Network are balanced in the set). Furthermore, as discussed in Section 6.1, sampled dataset with 190 pairs used in global approximation is actually less similar to the dataset used to train the black-box model than the sampled dataset with 105 pairs. This lead ILASP to obtain worst results in the sampled dataset with 190 pairs, despite its size. In future work, we plan to incorporate similarity metrics among dataset, for example Maximum Mean Discrepancy (Gretton *et al.* 2012), to ensure that our sampled datasets closely resemble the black-box training distribution. This should enable ILASP to achieve better results when used on larger samples.

As mentioned, we involved PCA during experiments to deal with the time execution problem that a system like ILASP brings out when used on high-dimensional dataset. Specifically, we proposed two approaches, which are *indirect* and *direct* PCA. While the first obtain more correct (with respect to the ground-truth) and complex theories, the second obtains the better results in terms of estimators. Given that, in both cases, we obtain important reduction of execution time, passing from execution time of about 6 hours, to less than 1 minute. A possible development that is worth to be further investigated could be the composition of these two approaches. In fact, *direct* approach could be used as a sort of pre-processing for *indirect* one. Instead of considering the first j PCs in (8) for the definition of features to maintain, we could use the PCs contained in the theory returned by the *direct* approach. In turn, we can use ILASP on the same pairs using the resulting features and the corresponding theory as a starting point. Such composition should exploit the advantages of both *direct* and *indirect* approaches, leading to better performance without oversimplified theories.

As fully described in the previous chapter, ILASP appears to be more efficient when used as a global approximator, while it is more effective as a local approximator. Although local approximation yields good results compared to the global case, it also has a longer execution time because it can only be used for a pair, whereas the global approximator can be used for a set of pairs. We anticipate that transfer learning could lead to improvements in this regard. For instance, one could use the theory generated by the global approximator as a basis for the local approximator to reduce execution time. Besides that, other approaches could be investigated in the future in order to obtain better results, such as neurosymbolic methods. Furthermore, we note that this study is restricted to structured, symbolic inputs: extending the approach to raw, non-symbolic modalities (e.g., images, audio or raw text) would require additional concept-extraction or representation-learning steps (following a neurosymbolic approach) and is therefore left for future exploration. Finally, integrating diverse data such as gender, age, and region of origin into the existing ILASP framework may lead to better results, especially in a scenario like preference learning.

https://doi.org/10.1017/S1471068426100441 Published online by Cambridge University Press

References

Abdi, H. and Williams, L. J. 2010. Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics* 2, 433–459.

Adadi, A. and Berrada, M. 2018. Peeking inside the black-box: A survey on explainable artificial intelligence (XAI). *IEEE Access* 6, 52138–52160. https://doi.org/10.1609/aimag.v40i2.2850.

Angwin, J., Larson, J., Mattu, S. and Kirchner, L. (2016). *Machine Bias: There's Software Used Acroos the Country to Predict Future Criminals. And it's Biased Against Black.* ProPublica.

Arrieta, A. B., Díaz-Rodríguez, N., Ser, J. D., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R. and Herrera, F. 2020. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion* 58, 82–115. https://doi.org/10.1016/j.inffus.2019.12.012.

Bostrom, N. and Yudkowsky, E. 2018. The ethics of artificial intelligence. In *Artificial intelligence safety and security.* Chapman and Hall/CRC, 57–69.

Brunot, L., Canovas, N., Chanson, A., Labroche, N. and Verdeaux, W. 2022. Preference-based and local post-hoc explanations for recommender systems. *Information Systems* 108, 102021. https://doi.org/10.1016/j.is.2022.102021.

Calimeri, F., Faber, W., Gebser, M., Ianni, G., Kaminski, R., Krennwallner, T., Leone, N., Maratea, M., Ricca, F. and Schaub, T. 2020. ASP-Core-2 input language format. *Theory and Practice of Logic Programming* 20, 294–309.

Caliskan, A., Bryson, J. J. and Narayanan, A. 2017. Semantics derived automatically from language corpora contain human-like biases. *Science* 356, 183–186. https://doi.org/10.1126/science.aal4230.

Cath, C., Wachter, S., Mittelstadt, B., Taddeo, M. and Floridi, L. 2018. Artificial intelligence and the 'Good Society': The US, EU and UK approach. *Science and Engineering Ethics* 24, 505–528.

Collevati, M., Eiter, T. and Higuera, N. 2024. Leveraging Neurosymbolic AI for slice Discovery. In *International Conference on Neural-Symbolic Learning and Reasoning*. Springer Nature Switzerland, Cham, 403–418.

D'Asaro, F. A., Spezialetti, M., Raggioli, L. and Rossi, S. 2020. Towards an inductive logic programming approach for explaining black-box preference learning systems. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, vol. 17, 855–859. https://doi.org/10.24963/kr.2020/88.

De Ville, B. 2013. Decision trees. *Wiley Interdisciplinary Reviews: Computational Statistics* 5, 6, 448–445. https://doi.org/10.1002/wics.1278.

Fan, F. L., Xiong, J., Li, M. and Wang, G. 2021. On interpretability of artificial neural networks: A survey. *IEEE Transactions on Radiation and Plasma Medical Sciences* 5, 6, 741–760.

Ferreira, J., de Sousa Ribeiro, M., Gonçalves, R. and Leite, J. 2022. Looking inside the black-box: Logicbased explanations for neural networks. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, vol. 19, 432–442.

Fossemò, D., Mignosi, F., Raggioli, L., Spezialetti, M. and D'Asaro, F. 2022. Using inductive logic programming to globally approximate neural networks for preference learning: challenges and preliminar results. In *BEWARE 2022: BIAS, Ethical AI, Explainability and the Role of Logic and Logic Programming.* CEUR-WS.org, 67–83. https://ceur-ws.org/Vol-3319/paper7.pdf.

Fürnkranz, J. 2010. *Preference Learning.* Springer-Verlag, Berlin Heidelberg.

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B. and Smola, A. 2012. A kernel two-sample. *Journal of Machine Learning Research* 13, 723–773.

https://doi.org/10.1017/S1471068426100441 Published online by Cambridge University Press

Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F. and Pedreschi, D. 2018. A survey of methods for explaining black-box models. *ACM Computing Surveys (CSUR)* 51, 1–42.

Gunning, D. and Aha, D. 2019. DARPA's explainable artificial intelligence (XAI) program. *AI Magazine* 40, 2, 44–58. https://doi.org/10.1609/aimag.v40i2.2850.

Gurrieri, M., Fortemps, P. and Siebert, X. 2014. *Alternative Decomposition Techniques for Label Ranking*. In A. Laurent, O. Strauss, B. Bouchon-Meunier and R. R. Yager (eds.), *Information Processing and Management of Uncertainty in Knowledge-Based Systems: 15th International Conference, IPMU 2014, Montpellier, France, July 15–19, 2014, Proceedings, Part II*. Communications in Computer and Information Science, Vol. 443. Springer, pp. 464–474. https://doi.org/10.1007/978-3-319-08855-6_47.

Harper, F. M. and Konstan, J. A. 2015. The Movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems* 5, 4, 1–19. https://doi.org/10.1145/2827872.

Hillerström, F. and Burghouts, G. 2024. Towards probabilistic inductive logic programming with neurosymbolic inference and relaxation. *Theory and Practice of Logic Programming* 24, 4, 628–643.

Hong, W., Zheng, N., Xiong, Z. and Hu, Z. 2020. A parallel deep neural network using reviews and item metadata for cross-domain recommendation. *IEEE Access* 8, 41774–41783.

Hu, R., Chau, S. L., Ferrando, H. L. and Sejdinovic, D. 2022. Explaining preferences with shapley values. *Advances in Neural Information Processing Systems* 35, 27664–27677.

Kamishima, T. 2003. Nantonac collaborative filtering: recommendation based on order responses. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 583–588.

Law, M. 2018. Inductive Learning of Answer Set Programs. Ph.D. thesis, Imperial College London. https://ethos.bl.uk/OrderDetails.do?uin=uk.bl.ethos.762179.

Law, M., Russo, A. and Broda, K. 2014. Inductive learning of answer set programs. In European Workshop on Logics in Artificial Intelligence, Springer, 311–325.

Law, M., Russo, A. and Broda, K. 2015. Learning weak constraints in answer set programming. *Theory and Practice of Logic Programming* 15, 511–525.

Law, M., Russo, A. and Broda, K. 2018. Inductive learning of answer set programs from noisy examples. *CoRR* abs/1808.08441. arXiv:1808.08441.

Law, M., Russo, A., Bertino, E., Broda, K. and Lobo, J. 2020. FastLAS: Scalable inductive logic programming incorporating domain-specific optimisation criteria. *Proceedings of the AAAI Conference on Artificial Intelligence* 34, 03, 2877–2885. https://doi.org/10.1609/aaai.v34i03.5678.

Li, B., Fan, Y., Papka, M. E. and Lan, Z. 2022. Encoding for reinforcement learning driven scheduling. In *Workshop on Job Scheduling Strategies for Parallel Processing*. Springer Nature Switzerland, Charm, 68–87.

Liu, H., Liu, Q., Li, P., Zhao, P. and Wu, X. 2022. A deep selective learning network for cross-domain recommendation. *Applied Soft Computing* 125, 109160.

Rabold, J., Siebers, M. and Schmid, U. 2018. Explaining black-box classifiers with ILP - empowering LIME with Aleph to approximate non-linear decisions with relational rules. In *International Conference on Inductive Logic Programming*, Springer, 105–117.

Rabold, J., Deininger, H., Siebers, M. and Schmid, U. 2020. Enriching visual with verbal explanations for relational concepts - combining lime with aleph. In *Machine Learning and Knowledge Discovery in Databases*, P. Cellier and K. Driessens, Eds. Springer, 180–192.

Rabelo de Sá, C., Duivesteijn, W., Soares, C. and Knobbe, A. 2016. Exceptional preferences mining. In *Discovery Science*, T. Calders, M. Ceci and D. Malerba, Eds. Springer International Publishing, Cham, 3–18.

https://doi.org/10.1017/S1471068426100441 Published online by Cambridge University Press

Rajpurkar, P. and Lungren, M. P. 2023. The current and future state of AI interpretation of medical images. *New England Journal of Medicine* 388, 1981–1990.

Ribeiro, M. T., Singh, S. and Guestrin, C. 2016. Why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16*, Association for Computing Machinery, New York, NY, USA, 1135–1144. https://doi.org/10.1145/2939672.2939778

Saeed, W. and Omlin, C. 2023. Explainable AI (XAI): A systematic meta-survey of current challenges and future opportunities. *Knowledge-Based Systems* 263, 110273.

Schramowski, P., Turan, C., Jentzsch, S., Rothkopf, C. and Kersting, K. 2020. The moral choice machine. *Frontiers in Artificial Intelligence* 3, 36. https://doi.org/10.3389/frai.2020.00036.

Severyn, A. and Moschitti, A. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 373–382.

Shakerin, F. and Gupta, G. 2019. Induction of non-monotonic logic programs to explain boosted tree models using lime. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 3052–3059.

Soori, M., Arezoo, B. and Dastres, R. 2023. Artificial intelligence, machine learning and deep learning in advanced robotics, A review. *Cognitive Robotics* 3, 54–70. https://doi.org/10.1016/j.cogr.2023.04.001.

Srinivasan, A. 2019. The Aleph manual. URL: http://www.cs.ox.ac.uk/activities/machinelearning/Aleph/

Weber, L., Lapuschkin, S., Binder, A. and Samek, W. 2023. Beyond explaining: Opportunities and challenges of XAI-based model improvement. *Information Fusion* 92, 154–176.

Zhang, C. and Meng, Y. 2024. Bayesian deep learning: An enhanced AI framework for legal reasoning alignment. *Computer Law & Security Review* 55, 106073. https://doi.org/10.1016/j.clsr.2024.106073.

Zhang, Q., Liao, W., Yuan, B., Zhang, G., Yuan, B. and Lu, J. 2021. A deep dual adversarial network for cross-domain recommendation. *IEEE Transactions of Knowledge and Data Engineering* 35, 4, 3266–3278.

Zhou, Y., Liu, Y., Yang, J., He, X. and Liu, L. 2014. A taxonomy of label ranking algorithms. *Journal of Computers* 9, 557–565.

Zhu, F., Wang, Y., Chen, C., Liu, G., Orgun, M. A. and Wu, J. 2018. A deep framework for cross-domain and cross-system recommendations. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 3711–3717. https://doi.org/10.24963/ijcai.2018/516.

Appendix A. ILASP technical details

In Section 3.3, we discussed the main components of ILASP, focusing primarily on their theoretical aspects rather than on technical details. This appendix provides the complete syntactic specification and additional technical details omitted from the main text, allowing the reader to gain a full understanding of the methodologies employed in this work.

Mode declaration and placeholder. Placeholder are terms of the form $var(t)$ or $const(t)$ for some constant term t expressing that such terms can be replaced by any variable or constant of type t. Each constant c which is allowed to replace a *const* term of type t is specified as $\#constant(t, c)$. So, an atom a is compatible with a mode declaration M if each of the placeholder constants and placeholder variables in M has been replaced

https://doi.org/10.1017/S1471068426100441 Published online by Cambridge University Press

by a constant or variable of the correct type. For instance $\#modeo(\cdot)$ declarations (also known as *optimization body declaration*) forces ILASP to include weak constraints of the provided form in the search space. There exist five types of mode declaration, which are:

- $\#modeh(p(\cdot))$ to declare what can be in the head ($\#modeha(p(\cdot))$ if we want to allow aggregates).
- $\#modeb(p(\cdot))$ to declare what can be in the body.
- $\#modec(\cdot)$ to declare conditions, where the argument can be, for instance $var(t1) > var(t2)$.
- $\#modeo(\cdot)$ declare what atoms can appear in a weak constraint.

For each mode declaration can be defined the *recall*, which is a positive integer value specifying the maximum number of times that the mode declaration can be used in each rule. For example, $\#modeb(2, p(var(t)))$ indicate that the atom $p(|\cdot)$, which take as argument a variable of type t, can be used at most two times.

Finally, consider the listing 1 of Example 2:

```
:~ value(p,V1).[1@2, V1]
:~ value(p,V1).[-1@2, V1]
:~ value(p,V1).[V1@2, V1]
:~ value(p,V1).[-V1@2, V1]
:~ value(p,V1).[V1@1, V1]
:~ value(p,V1).[-1@1, V1]
:~ value(p,V1).[1@1, V1]
:~ value(p,V1).[-V1@1, V1]
```

Listing 1: constraints generated by the first line in Example 2

Notably, the `constant` of type `val` is defined as `p` using $\#\texttt{constant}(\texttt{val}, \texttt{p})$ below, and the possible *weights* for the constraints are 1 and -1 times the value of the variable in the constraint, as specified through the $\#\texttt{weight}$ declaration.

Positive and negative examples. Syntactically, a positive example is defined as follows:

$$\#\texttt{pos}(\texttt{id}, \texttt{i1}, \ldots, \texttt{im}, \texttt{e1}, \ldots, \texttt{en}). \tag{A1}$$

where `id` identifies the example, $E^{inc} =$ `i1`, ...`in` are atoms called *inclusions* while $E^{exc} =$ `e1`, ..., `en` atoms called *exclusions*, and serve to define what atoms should be included and excluded by an answer set which "cover" that positive example. Let define as $AS(P)$ the set of all answer sets of a ASP program P, then a positive example is covered by an answer set if there exist an answer set $A \in AS(P)$ such that $(E^{inc} \subseteq A) \wedge (E^{exc} \cap A = \emptyset)$ (in this case we say that A extends $E = \langle E^{inc}, E^{exc} \rangle$). Negative examples can be defined similarly with the only difference that a negative example is covered if there is no answer set of the learned program that extends it.

Ordering examples. There exist two kind of ordering example, namely *cautious ordering* and *brave ordering*. Syntactically we can define them as follow:

$$\begin{gathered}\#\texttt{cautious_ordering}(\texttt{e1}, \texttt{e2}, \texttt{s}) \\ \#\texttt{brave_ordering}(\texttt{e1}, \texttt{e2}, \texttt{s})\end{gathered} \tag{A1}$$

https://doi.org/10.1017/S1471068426100441 Published online by Cambridge University Press

Where `e1` and `e2` are two examples and `s` denoted one of the rational operator $=, \geq, \leq, >, <$. If `s` is equal to $<$, *cautious ordering* states that every answer set which extends `e1` must be preferred to any answer set which extends `e2` (if `s` is equal to $>$, it means the opposite, while if it assumes $=$ value, means that they are equally preferred). *Brave orderings*, considering `s` is equal to $<$, state that at least one answer set which extends `e1` must be preferred to any answer set which extends `e2` (if `s` is equal to $>$, it means the opposite, while if it assumes $=$ value, means that they are equally preferred).

Appendix B. Neural Network training and architecture

In this section, we describe the experimented NN, describing its training and architecture.

The experiments follow an inter-users approach, and so for each user is trained a neural network with the same structure but with the user's preferences as input. As described in Section 4.2 the problem that these models have to solve is a ternary classification problem. The samples are pairs of recipes, while the classes are the preference relationships that occur among the recipes with respect to the user. Specifically, a sample has the form $\langle ID1, ID2\rangle = [a_1, a_2, \ldots, a_k, b_1, b_2, \ldots, b_k]$, where a_i and b_i are the feature associated to recipes with ID1 and ID2, respectively. On the other hand, classes are labeled with 1 ("the first recipe is preferred over the second"), -1 ("the second recipe is preferred over the first") and 0 ("it is uncertain which recipe is preferred between the two"). Before training, PCA is applied to the data for dimensionality reduction following the "Kaiser rule," that is, taking into account the PCs with corresponding eigenvalue greater or equal to 1. Thanks to the PCA we reduced each recipe to have 17 feature, so each sample $\langle ID1, ID2\rangle$ has 34 feature. Note that differently from ILASP, data are feed considering ingredient classes instead meta-classes. Finally, for each user we have 210 pairs, which are divided in 157 pairs for the training set and 53 for the testing set.

To define the architecture of the NN to use for the experiment we perform a fine-tuning of hyperparameters. Tuned hyperparameters are activation function, optimization function, number of nodes for hidden layers (32, 64, and 128), the learning rate (0.01, 0.001, 0.0005, 0.0001 and 0.0005) and the use or not of dropout (if used we consider 0.1, 0.2 and 0.5 as dropout value) and batch normalization. Because of the huge number of combination with such hyperparameters to tune, we make a two-stages tuning. In the first stage we tuned activation functions (setting a constraint to avoid using linear activation function for more than one hidden layer) and the optimization function to use, fixing the learning rate to 0.0005, the number of nodes to 64 and the use of dropout (with 0.1 dropout value) and batch normalization, while in the second stage we tuned the remaining fixing activations and optimization function to those found in the first stage.

The architecture which return the best results, that are 82.72% of accuracy, 83.26% of precision and 82.43% of recall is reported in Table B1. Finally, in (Fossemò *et al.* 2022) we conducted comparative study on this model and tuned Support Vector Machines and K-nearest Neighbors, obtaining best results with Neural Network.

https://doi.org/10.1017/S1471068426100441 Published online by Cambridge University Press

Table B1. *Deep neural network obtained after the tuning of hyperparameters*

	Activation function	Nodes
Input layer	tanh	64
Dropout	Rate al 10%	
Hidden layer	ReLU	64
Dropout	Rate al 10%	
	batch normalization	
Hidden layer	linear	64
Dropout	Rate al 10%	
Output layer	softmax	64
Optimization function	SGD	
learning rate	0,0005	

Appendix C. Ingredient classes vs meta-classes

In Table C1 are reported the ingredient meta-classes and corresponding classes.

Table C1. *Recipes dataset features*

Ingredient meta-class	Ingredient classes
Cereals	Cereals
Dairies	Butter; Cheese; Milk and cream; Soft cheese; Yogurt
Eggs	Eggs
Flouries	Bread; Flour
Fruit	Berries; Citrus fruits; Nuts; Salty fruits; Sweet fruits
Herb, Spices, Seasonings	Broth; Condiments; Herbs; Spices; Wines and Spirits
Meat	Beef; Chicken meat; Pork meat; Rabbit meat; Sheep meat; Turkey Meat
Mushrooms and Truffles	Mushrooms
Pasta	Dry pasta; Fresh pasta
Seafood	Lake fish; Mollusks; Sea fish; Shellfish
Sweeteners	Sweeteners
Vegetables	Greens; Legumes; Vegetables

Appendix D. ILASP code structure to approximate the Neural Network

In this section we discuss the code structure of the ILASP program to obtain a theory approximating the neural network in our context.

https://doi.org/10.1017/S1471068426100441 Published online by Cambridge University Press

```
#pos(item-i, {}, {}, {category(v1). value(cost, v2). value(difficulty,
v3). value(prepTime, v4). value(cereals, v5). ... value(vegetables,
v16). value(boiling, v17). ... value(stewing, v24).}).

#maxv(1).
#maxp(h).

#modeo(1, value(const(val), var(val))).
#modeo(1, category(const(mg), (positive)).
#weight(val).
#weight(1).
#weight(-1).
#constant(val, category).
#constant(val, cost).
#constant(val, difficulty).
#constant(val, prepTime).
#constant(mg, 1).
...
#constant(mg, 5).
#constant(val, cereals).
...
#constant(val, vegetables).
#constant(val, boiling).
...
#constant(val, stewing).
#brave_ordering(o_n@p_n, item-o_j1, item-o_j2, s_n).
```

Listing 2: Code structure to apply ILASP to obtain a theory approximating the Neural Network where $v_1, \ldots, v_{24} \in \mathbb{N}$; $h \in \{1, \ldots, 5\}$; `item-i` denote the i_{th} recipe of the dataset, with $1 \leq i \leq |D|$ (where $|D|$ denote the dataset size); `o_n` denote the n_{th} brave ordering in the program and `p_n` the corresponding associated penalty, with $1 \leq n \leq N$. With $\texttt{item} - \texttt{o_j1}$ ($\texttt{item} - \texttt{o_j2}$) we identify the first (second) item present in the n_{th} ordering example. Finally, with `s_n` we denote the symbol of the corresponding brave ordering. Clearly $|D| = 100$ (since our recipes dataset has 100 recipes), while N varies depending on the experiments. Furthermore, since we are in the global approximation case, $p_n = 1 \ \forall \ n = 1, \ldots, N$. Note that, since we want to obtain a theory composed of weak constraints, we use `brave_ordering` and `#modeo( · )` as mode declaration. With `brave_ordering` we are inducing a preference relation between answer sets, depending on the value of `s_n`. Here the pairs ($\texttt{item} - \texttt{o_j1}$, $\texttt{item} - \texttt{o_j2}$) correspond to the N sampled pair (j_1, j_2) (we refer to the $n_t h$ sampled pair with the notation $(j_1, j_2)_n$), while `s_j` is defined as follow:

$$\texttt{s_n} = \begin{cases} < \ if \ B((j_1, j_2)_n) = 1 \\ = \ if \ B((j_1, j_2)_n) = 0 \\ > \ if \ B((j_1, j_2)_n) = -1 \end{cases} \tag{D4}$$

https://doi.org/10.1017/S1471068426100441 Published online by Cambridge University Press

note that here "$<$" should not be confused with "`item` $-$ `o_j2` is preferred over `item` $-$ `o_j1`," since in ILASP it means the opposite, that is "`item` $-$ `o_j1` is preferred over `item` $-$ `o_j2`." This difference comes from the cost concept introduced by ASP, explained in Section 3.2. Finally, #`modeo`($\cdot$) mode declaration serves to give shape to our weak constraint, which will presents as the one reported below as an example

Example 3.

```
:~value(vegetables, V1).[-V1@1, V1]
:~value(meat, V1).[-V1@2, V1]
:~value(difficulty, V1).[-V1@3, V1]
:~value(stewing, V1).[V1@4, V1]
:~value(dairies, V1), category(3).[V1@5, V1]
```

note that this is a theory obtained with $h = 5$ and that the presence of `V1` as weight denotes that the penalty cost paid at the corresponding level of priority (if weak constraint is activated) is equal to the value of `V1` in the atoms in which it appears.

https://doi.org/10.1017/S1471068426100441 Published online by Cambridge University Press