Using Inductive Logic Programming to globally approximate Neural Networks for preference learning: challenges and preliminary results

Daniele Fossemò[1,*], Filippo Mignosi[1,2], Luca Raggioli[3], Matteo Spezialetti[1] and Fabio Aurelio D'Asaro[4]

[1]Department of Information Engineering, Computer Science and Mathematics, University of L'Aquila, Italy
[3]Department of Computer Science, University of Manchester, United Kingdom
[2]ICAR-CNR, Palermo, Italy
[4]Ethos Group, Department of Human Sciences, University of Verona, Italy

Abstract

In this paper we explore the use of Answer Set Programming (ASP), and in particular the state-of-the-art Inductive Logic Programming (ILP) system ILASP, as a method to explain black-box models, e.g. Neural Networks (NN), when they are used to learn user preferences. To this aim, we created a dataset of users preferences over a set of recipes, trained a set of NNs on these data, and performed preliminary experiments that investigate how ILASP can globally approximate these NNs. Since computational time required for training ILASP on high dimensional feature spaces is very high, we focused on the problem of making global approximation more scalable. In particular we experimented with the use of Principal Component Analysis (PCA) to reduce the dimensionality of the dataset while trying to keep our explanations transparent.

Keywords

Explainable AI, Preference Learning, Answer Set Programming, Inductive Logic Programming, ILASP, PCA

1. Introduction

The application of machine learning algorithms in very diverse fields has exponentially increased in the last decade, covering more operational areas, like robotic and drone navigation, as well as the processing of sensitive data, such as in legal reasoning. The use of this category of techniques reached remarkable performance in several complex tasks, in some cases achieving superhuman capabilities (e.g., DeepMind's AlphaZero/Go winning against the Go and chess world champions). A very relevant issue with many *Machine Learning* (ML) *models*, however, is that the internal structure and the processes are exceedingly hard to understand and interpret, even for the developers themselves – for this reasons they are sometimes referred to as *black-box*

BEWARE: Joint BRIO, MEE and AWARE Workshop @ AIxIA 2022, November 28 - December 2, 2022, University of Udine, Udine, Italy
*Corresponding author.
daniele.fossemo@student.univaq.it (D. Fossemò); filippo.mignosi@univaq.it (F. Mignosi); luca.raggioli@manchester.ac.uk (L. Raggioli); matteo.spezialetti@univaq.it (M. Spezialetti); fabioaurelio.dasaro@univr.it (F. A. D'Asaro)

(BB) *models*. This issue is even more pressing and calls for attention in cases where sensible data are involved, and when the model's reasoning and outputs may have a tangible impact in the real world. In 2016, the General Data Protection Regulation (GDPR)[1] was introduced, stressing the need for automated systems' decisions to be complemented with meaningful explanations of the logic behind its' decisions, especially when these may potentially affect individuals. This has spotlighted the need for *transparent systems* that are able to provide the users with human-understandable rationale for their decisions, as well as *post-hoc methods*, which instead attempt to formulate an explanation from black-boxes outputs (see [1] 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 such models. Transparent systems, on the other hand, often impose a trade-off between accuracy and explainability.

In this work, we propose a post-hoc method to explain black-box models for preference learning using *Answer Set Programming* (ASP) [2] and Inductive Logic Programming framework *ILASP* [3, 4, 5], extending upon the work presented in [6]. 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 Machine Learning models) for the presence of biases and artifacts resulting from training on large collections of data (see e.g. [7, 8]).

Following [6], we further 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 such black-boxes were trained on is not available. This method may help identify systematic biases such as, e.g., decisions influenced by prejudices or unfair principles.

In this work we are particularly concerned with *preference learning*, so it is worth mentioning here that ASP suits well our task as its standard syntax has constructs known as *weak constraints* [2] that can be used to naturally express preference relations. In turn, these preferences can be learned by ILASP. Thanks to its ability to generalize from examples, it can process the output of black-box methods to find an appropriate ASP program that mimics its behaviour. However, ILASP is computationally very expensive on highly dimensional data, with long execution times on average. For this reason, we used *Principal Component Analysis* (PCA) to identify the most relevant features in the data and decrease the size of the feature space, aiming to achieve a good trade-off between transparency and adherence of the output theory to the target black-box. Our target domain is that of gastronomic preferences, which we formalized as a comparison between recipes. More specifically, given a user and a pair of recipes, the model aims to classify them in one of the following three ways: "the user prefers the first recipe of the pair over the second one", "the user prefers the second recipe of the pair over the first one" and "the user does not have a strong preference of a recipe over the other one". We created a *dataset of recipes* retrieved from an Italian cooking website, GialloZafferano [2]. We then surveyed participants about their preferences about recipes in this dataset, and created a *dataset of user preferences*. We trained fully dense NNs on these user preferences (one for each user), and then trained ILASP on the output of such NNs. Finally, we studied to which extent the output theory of ILASP was able to accurately predict the output of the corresponding NN. The developed

[1] https://eur-lex.europa.eu/eli/reg/2016/679/oj
[2] https://www.giallozafferano.it/

code during the described work can be found in a public github repository [3].

As it will fully explained at the end of the article, in terms of accuracy, precision and recall, we haven't obtained very good results in approximating black-box models. That said, the proposed methods involving PCA to approach the problem of the execution time required by ILASP during training phase looks promising.

2. Related Work

This paper is a continuation of [6]. In that work, ILASP was applied to local and global approximation of *Support Vector Machines* (SVMs), a set of simple classical ML models that have often been used to benchmark explainable AI technologies (see e.g. [9]). 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.

Our work is not the only one using ILP to approximate black-boxes. For instance, [10, 11] and [12] use the Prolog-based system Aleph [13]. We use ILASP instead, which was shown to outperform Aleph accuracy-wise in some cases respect to the predictions (see e.g. [5]). Furthermore, our focus here is on preference learning tasks, which is a typical application area for ASP (as opposed to Prolog).

Obviously, ILP is not the only possible approach to explaining black-box systems – for instance, Exceptional Preferences Mining [14] and LIME [15] are other techniques whose relationship to ILP and ILASP may be the subject of future work. Furthermore, 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 we relax this constraint and perform model-specific operations throughout the training phase, as in [16] 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

In this section we give a brief overview of the basics of Answer Set Programming and Inductive Logic Programming, and how one can represent and learn preferences within them.

Preference learning [17, 18, 19] aims to develop agents that can profile users, and know how to classify what could be of interest to them. In general, a preference system can be formalized as a (total or partial) ordering among items. More formally, given an instance space $\boldsymbol{X}$ and a set of labels $L = \{\lambda_1, \lambda_2, ..., \lambda_k\}$, an ordering for an instance $\boldsymbol{x}$ can be described as ordering of the labels by means of a relation $\succ_{\boldsymbol{x}}$ such that $\lambda_i \succ_{\boldsymbol{x}} \lambda_j$ indicates that λ_i is preferred over λ_j by instance $\boldsymbol{x}$. So the ordering induced by $\succ_{\boldsymbol{x}}$ can be identified by a permutation $\pi_{\boldsymbol{x}}$ such that $\pi_{\boldsymbol{x}}(i) = \pi_{\boldsymbol{x}}(\lambda_i)$ indicates the position of the label λ_i in the ordering. So, a complete ordering of the lables in L can be described as following:

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

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

where $\pi_{\boldsymbol{x}}^{-1}(j)$ indicates the index of the label which occupates the j-th position in the ordering [19]. In order to find out an accurate $\succ_{\boldsymbol{x}}$ for each instance $\boldsymbol{x}$ and learn to ordering new elements in the future for that instance, the artificial intelligence is trained on a set of labels in which we already know the relationships. 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 element can be structured as a succesion of consecutive question "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 in the order of preferences, we can define all preference relationships in this form by using the following function:

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

the number of couples (λ_i, λ_j), if k indicates the number of elements in the collection, is equal to $k(k-1)/2$. This decomposition allow us to decompose the original problem into a set of presumably simpler sub-problems, which is useful from a machine learning point of view [17]. In our work we treat "*uncertain*" between elements. In fact, in a set of elements is possible that for an instance $\boldsymbol{x}$ is uncertain the preference relationship among some elements. We will treat this concept thanks to the algorithm 1 which we explain during the description of the creation of the dataset. Conceptually we will label couples of element for which we cannot find the preference relationship as uncertain couples, and so we pass from a problem of binary classification to one of ternary classification.

Answer Set Programming [2] is a declarative logic programming language based on the stable model semantics, whose output is a model called of the input theory usually called *stable model* or *answer set*. We are particularly concerned with a special construct that allows one to represent preferences in ASP, namely the *weak constraints*, which induces a (partial) ordering over the stable models of a theory. Given n, $m \geq 0$, a weak constraint has the form:

```
:~ b1, ..., bn [w@l, t1, ..., tm]
```

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[4]. Unlike hard constraints, weak constraints do not affect the answer sets of a theory. Instead, they induce a preference relation among them. Intuitively, each answer set satisfying the body of a weak constraint gets a penalty that is proportional to that weak constraint's weight. To describe the preference relation induced by these weak constraints, we first 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. For instance 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]
```

[4]We are not going to discuss this syntax here, but the interested reader can find the full syntax and semantics in [2].

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 $\{p(a),\ p(b),\ p(c),\ q(a),\ q(b),\ q(c)\}$. The answer set $\{p(a), p(b), p(c), q(a), q(b), q(c)\}$ satisfies all the bodies of weak constraints, thus its cost at priority level 2 is 0 (which results from adding the weights of the first and third constraint above), whereas its cost at priority level 1 is 3. 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.

ILASP (inductive language of answer set programs)[3, 4, 5] is an Inductive Logic Programming language mainly used for Explainable AI, which can learn ASP programs. Furthermore, ILASP can learn the preferences represented as weak constraint in ASP. ILASP, given some additional background knowledge (the so-called *language bias*) required to define the hypothesis space, will find a suitable theory (including weak constraints) covering the examples in the input. Instead, theories that do not cover some of the examples are penalized by ILASP, which then tries to find the theory in the hypothesis space that is minimally penalized.

4. Dataset

The domain of choice for this work is user-dependent gastronomic preferences, formalized through pairwise comparison between recipes. To this aim, we built two datasets, to gather recipes' features and users' preferences over recipes, respectively. Both are freely available online [5]. More specifically, the two dataset are:

- **Recipes dataset**: a dataset of 100 recipes crawled and processed from Giallo Zafferano website.
- **Users Preferences:** a dataset containing 54 different user answers about preferences on different subsets of recipes and preferences over the features of the recipes dataset.

To realize the recipes dataset a crawler was used to send a set of requests to Giallo Zafferano website to get the pages with raw data, allowing us to obtain about 4000 recipes. We chose the first 100 most voted recipes by the website visitors (sweets and pastries excluded).

The unprocessed and non-standardized data obtained with the crawler included 277 ingredients. In order to improve training efficiency while still maintaining a reasonable level of granularity in the data, we defined three levels of categorization:

- All the ingredients are considered as in the raw data. This is the densest level of granularity.
- The ingredients are divided into 36 classes. This is the intermediate level of granularity.
- The 36 classes are grouped into 12 meta-classes. This is the least dense level of granularity.

For example, the first ingredient "*spaghetti*" belongs to the class "*dry pasta*" which belongs, as well as "*fresh pasta*", to the meta-class "*pasta*". In the recipes dataset we considered the intermediate level of granularity, because it allows to have a reasonable level of specificity for the

[5] https://doi.org/10.5281/zenodo.7265253

Feature	Description
ID	A progressive integer number assigned to the recipe
Name	The Italian name of the recipe
Category	Integer number from 1 to 5, where: 1 → starter 2 → Complete Meal 3 → First Course 4 → Second Course 5 → Savory Cake
Cost	Integer number from 1 to 5, where: 1 → very low 2 → low 3 → medium 4 → high 5 → very high
Difficulty	Integer number from 1 to 4, where: 1 → very easy 2 → easy 3 → medium 4 → difficult
Preparation time	integer positive number, expressed in minutes
Ingredients	Vector of 36 elements in which if the element i-th has a value: $v \geq 1$ → the ingredient class represented by position i is present in the recipe and has v as importance in the composition of the recipe $v = 0$ → the ingredient class represented by position i is not present in the recipe
Preparations	Vector of 9 elements with same codification of ingredients vector
Link	string representing the link of the recipe in Giallo Zafferano website

Table 1
Recipe Dataset features

ingredients, in light of the fact that these will be used to structure the surveys to administer to the users: in the surveys we ask the users to rank all the ingredients in the dataset, and ranking 277 ingredients is, obviously, un-practical. Since ILASP struggles with high-dimensional feature spaces, we performed ILASP experiments with the 12 meta-classes as this makes the training phase computationally feasible in our application. The recipes in the dataset are described by the features in Table 1. As example, the first entry of the dataset has the following representation: **1, Spaghetti alla Carbonara, 3, 2, 2, 25, [0 0 0 0 0 0 2 0 0 0 5 3 0 0 0 0 0 0 0 0 0 0 0 3 0 0 0 0 0 0 1 0 0 0 0 0 0], [4 2 0 0 1 0 0 0], https://ricette.giallozafferano.it/Spaghetti-alla-Carbonara**. Before any further experimental step, the dataset was preprocessed as follows:

- **Normalization:** for each recipe, the ingredient vector has been divided by its sum, as well as the preparation vector;
- **Standardization:** each numerical feature has been standardized.

For the realization of the survey we have considered several criteria, such as the choice of the

questions, their structure and, most of all, the choice of how many and which recipes include in the survey. This latter was crucial as we wanted to collect as many user preferences as possible, while avoiding to present the participants with an excessive number of questions (e.g., rank 100 recipes in decreasing order of preference). To solve this problem we created 10 different surveys aiming to be as representative as possible of the distribution of the full set of 100 recipes. We furthermore tried to ensure that each recipe appeared in at least some of the surveys. To do so we used the PCA for dimensionality reduction following the *kaiser rule* (taking into account components with eigenvalue ≥ 1), reducing the 47 features to 17 Principal Components. Subsequently, we used the 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 together 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, in which we collect the following data[6]:

- **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 have to sort the recipes presented according to her/his personal preferences.
- **Ingredient classes evaluation**: for each ingredient class presented, we ask to rate a generic recipe in which that ingredient class is present.
- **Ingredient meta classes evaluation**: for each ingredient meta class presented, we ask to rate a generic recipe in which that ingredient meta class is present.
- **Preparations evaluation**: for each food preparation, we ask to rate a generic recipe in which that preparation is done.
- **Generic preferences**: generic preferences about cost, difficulty and preparation time.
- **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 doesn't like fried fish). Each combination generated by the user also needs to be rated (as in Recipes evaluation). We limited the each combination to a maximum of 3 items between ingredients and preparations, and we collected for each user a maximum of 4 combinations.

It is important to note that the section *Sorting recipes* is divided in 3 subsections, because in

[6]All data are collected in anonymous form, but in case the users were interested in viewing statistics regarding session, there is the possibility of providing their e-mail (which is not included in the dataset for privacy reasons) after the survey is concluded.

Microsoft Form[7], the service used to make the surveys, it is not possible to ask to the user to sort more than 10 elements. For this reason the user is asked to sort the 21 recipes presented in the survey, by effectively sorting 10 elements at a time. The subsections are organized in the following structure:

- $O_1 = [I_{1(1,2)}, I_{2(1,2)}, I_{3(1,2)}, I_{4(1,3)}, I_{5(1,3)}, I_{6(1,3)}, I_{7(1)}, I_{8(1)}, I_{9(1)}, I_{10(1)}]$
- $O_2 = [I_{1(1,2)}, I_{2(1,2)}, I_{3(1,2)}, I_{11(2,3)}, I_{12(2,3)}, I_{13(2,3)}, I_{14(2)}, I_{15(2)}, I_{16(2)}, I_{17(2)}]$
- $O_3 = [I_{4(1,3)}, I_{5(1,3)}, I_{6(1,3)}, I_{11(2,3)}, I_{12(2,3)}, I_{13(2,3)}, I_{18(3)}, I_{19(3)}, I_{20(3)}, I_{21(3)}]$

where O_i represent the i^{th} ordering subsection and $I_{j(L)}$ refers to the j^{th} item that is present in the list of orderings L. Note that the subsections are organized in three ordering because this allow us 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. Moreover we have created the representative subset exactly to avoid to ask too many questions to the users, and considering also the whole survey by itself, we considered that three orderings should be the best choice (indeed is the minimum number of orderings needed to cover all the 21 recipes).

Elements present in more than one subsection allow us to combine the three orderings ranked by the users, and create a list of pairwise comparisons between the 21 recipes, with pairs of recipes that are directly or indirectly in a ordering relation (e.g. $a > b$ and $b > c \rightarrow a > c$) being included in the list. More specifically, this process is structured as follows (and the pseudo-code is shown in Algorithm 1):

- First, the algorithm searches for direct preferences in each ordering (i.e. if item I_i precede I_j) and finds all the uncertainty between the partial orderings resulting from the 3 subsections (i.e. if item I_i precede I_j in an ordering but succeeds it in another). In this way an initial set S of preference pairs (I_i, I_j) are obtained, meaning that I_i is preferred to I_j.
- Then, the algorithm iteratively applies a transitive rule to the set of preferences: if $(I_i, I_j), (I_j, I_k) \in S$ then (I_i, I_k) is added to S. This rule is applied until no more changes are made to S.
- The pairs for which no preference relationships were found in the previous steps, are marked with the uncertain relationship.

It's worth noting that the application of these steps, without further constraints, could cause to two possible issues:

- Inconsistency: if more than one transitive rules can be applied in the same iteration to the same pair of items, but with an opposite verse, they could change S, by adding both (I_i, I_k) and (I_k, I_i);
- Overwriting of existing preferences: to avoid inconsistencies, we should allow the transitive rule to remove (I_k, I_i), if present, if (I_i, I_k) is discovered. Unfortunately, this would lead to the overwriting of pairs discovered in previous iterations, but also, in case of transitive rules of the same iteration acting in an opposite way on the same pair, to an endless loop.

[7]Note that we used Microsoft Form after considering different other solution and considering different criteria. Some of these criteria are ease of use, the possibility of exporting the answers via .csv files, the possibility of asking as a question the request to order a set of recipes through a ranking system, and the possibility of link the GialloZafferano recipe web pages directly on questions.

We addressed both aspects by introducing a priority level decreasing from the first step through each iteration of the algorithm and by making it "lock", with that level, pairs and uncertainties discovered at each iteration, so that their cannot be further changed. The rationale behind this choice is that we "trust" more in the first step and in earlier iterations, because they represent a shorter chain of transitive rules.

Algorithm 1 Conversion from orderings to pairwise comparison

Require: R the list of n element in the orderings; O the list of recipes orderings

$M \leftarrow 0_{n,n}$; ▷ $n \times n$ matrix of zeros used to save preferences
$L \leftarrow 0_{n,n}$; ▷ $n \times n$ matrix of zeros used to keep trace of locked preferences
for $(r_i, r_j) \in R \times R$ **do** ▷ i and j are the indices of r_i and r_j in R
    for $o \in O$ **do**
        if $r_i \in o$ and $r_j \in o$ and position(r_1, o) < position(r_2, o) **then**
            if $L[i, j] == 0$ **then**
                $M[i, j] \leftarrow 1$; $M[j, i] \leftarrow -1$; $L[i, j] \leftarrow 1$; $L[j, i] \leftarrow 1$;
            else if $M[i, j] == -1$ **then**
                $M[i, j] \leftarrow 0$; $M[j, i] \leftarrow 0$; $L[i, j] \leftarrow 1$; $L[j, i] \leftarrow 1$;
            end if
        end if
    end for
end for

$c \leftarrow 1$; $p \leftarrow 2$;
while $c == 1$ **do**
    $c \leftarrow 0$;
    for $(r_i, r_j, r_k) \in R \times R \times R$ **do**
        if $M[i, j] == 1$ and $M[j, k] == 1$ **then**
            if $L[i, k] == 0$ **then**
                $M[i, k] \leftarrow 1$; $M[k, i] \leftarrow -1$; $L[i, k] \leftarrow p$; $L[k, i] \leftarrow p$;
            else if $L[i, k] == p$ and $M[i, k] == -1$ **then**
                $M[i, k] \leftarrow 0$; $M[k, i] \leftarrow 0$; $L[i, k] \leftarrow p$; $L[k, i] \leftarrow p$;
            end if
        end if
    end for
end while
return M

Note that in the case in which we have only one list, then the effect of this algorithm will be simply the decomposition from a ranking problem to a pairwise comparison problem in which is never present the uncertain case. We used a script to redirect to one of the 10 surveys randomly, starting from a unique link that we have successively distributed. The surveys were filled out 48 times. From the third section of the surveys, then, we created the user preferences dataset. For each entry (and so, for each user) there are 210 pairs, in which the element are the ID of the recipes (pairs with elements with same ID were discarded). Each pair has label 1 if the first

	accuracy	precision	recall
Support Vector Machines	76.54%	77.29%	76.54%
K-nearest Neighbors	73.63%	67.49%	73.63%
Neural Network	82.72%	83.26%	82.43%

Table 2
Results of the considered black-box model

	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	

Table 3
Neural network considered during experiments

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.

5. ILASP and PCA experiments

The next step, after the realization of dataset, was to build a black box model to classify user preferences expressed in the pairwise form. Because we are in a user-dependent domain we aims to develop a model for each users, and so the statistics presented from now on are the mean of all the models results. To do so, we considered several methods, such as Support Vector Machines, K-nearest Neighbors and Neural Networks, tuning the hyper-parameters for each of them. The black box model exhibiting the best performance was a fully connected Neural Network, structured as in Table 3. Therefore we used this network architecture in the experiments with ILASP. Given that we are modeling user preferences, we trained a distinct model for each user. For instance, given the model approximating the preferences of user u, and the input (i_1, i_2), the NN outputs a class which should correctly represent the user preference. In order to evaluate the performance of the Neural Network, we averaged the results over the validation of each user's model. The results are reported with the results obtained with Support Vector Machines and K-nearest Neighbors in table

The aim of this work is to approximate the underlying theory of the Neural Network model using ILASP. To do so, there are two main possible approaches, that are using of ILASP as global or local approximator. In this work, we focus on global approximation, in order to extrapolate the general logic that the Neural Network use to make decision. Specifically, we samples N

pairs (i_1, i_2) at random from the feature space and we label these pairs with the predictions $B(i_1, i_2)$ made by the Neural Network; then we train ILASP on these pairs using an appropriate language bias L. For a deeper description of approximation approaches, please refer to D'Asaro et al. 2020 [6], which is the starting point of this work, and that we are trying to extent.

During the experimental phase, we focused very thoroughly on preserving the trade-off between the execution time and the complexity of the theory returned by ILASP, which is used in order to approximate the Neural Network. Such a theory is a set of weak constraint labeled with a weight, consisting of different literals, and ordered by priority. Given, for instance, the following theory:

```
:~ value(steaming,V1).[-1@3, V1]
:~ value(meat,V1).[V1@2, V1]
:~ value(prepTime,V1).[-V1@1, V1]
:~ value(oven,V1).[-1@4, V1]
:~ category(3), value(boiling,V1).[-V1@5, V1]
```

this means that according to the explanation produced by ILASP over the set of constraints obtained from the NN, the user likes (in order of priority) first course dishes that are boiled, recipes prepared in the oven, recipes with the "steamed" preparation, does not like meat and likes long preparation times. There are different parameters that influence the theory produced by ILASP, as well as the accuracy of approximation of the Neural Network, such as the number of weak constraints and the number of literals allowed for each of them. Other important parameters are the number of sampled pairs and the dimensionality of the space of features, which have a relevant impact on the performance of ILASP. Unfortunately, the time complexity tends to grows exponentially with the increase of these parameters. In Figure 1 we show how the performances and the training time of ILASP are affected by the increase of the number of sampled pairs, the maximum number of weak constraints (*maxp*) in the theory and the maximum number of literals for each weak constraint (*maxv*).

To handle this problem, we introduced the Principal Component Analysis in the workflow of approximation. The aim is to reduce the dimensionality of the dataset (as well as the execution time of ILASP), with a negligible loss of information.We explored two different approaches to use the PCA, which we refer to as *indirect* and *direct*.

In the *indirect* PCA we determine which are the most important features among the first n principal components, 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:

$$w_{ij} \geq \mu_j + 2\sigma_j \qquad j = 1, ..., n \tag{3}$$

This means that, the features whose weights are greater then the mean of the features' weights plus two times the standard deviation, among the first n PCs, are those that we keep in the dataset.

In the *direct* approach, we have passed to ILASP the recipes dataset directly after the PCA (so passing the PC as features). The idea behind this approach is to significantly reduce the feature

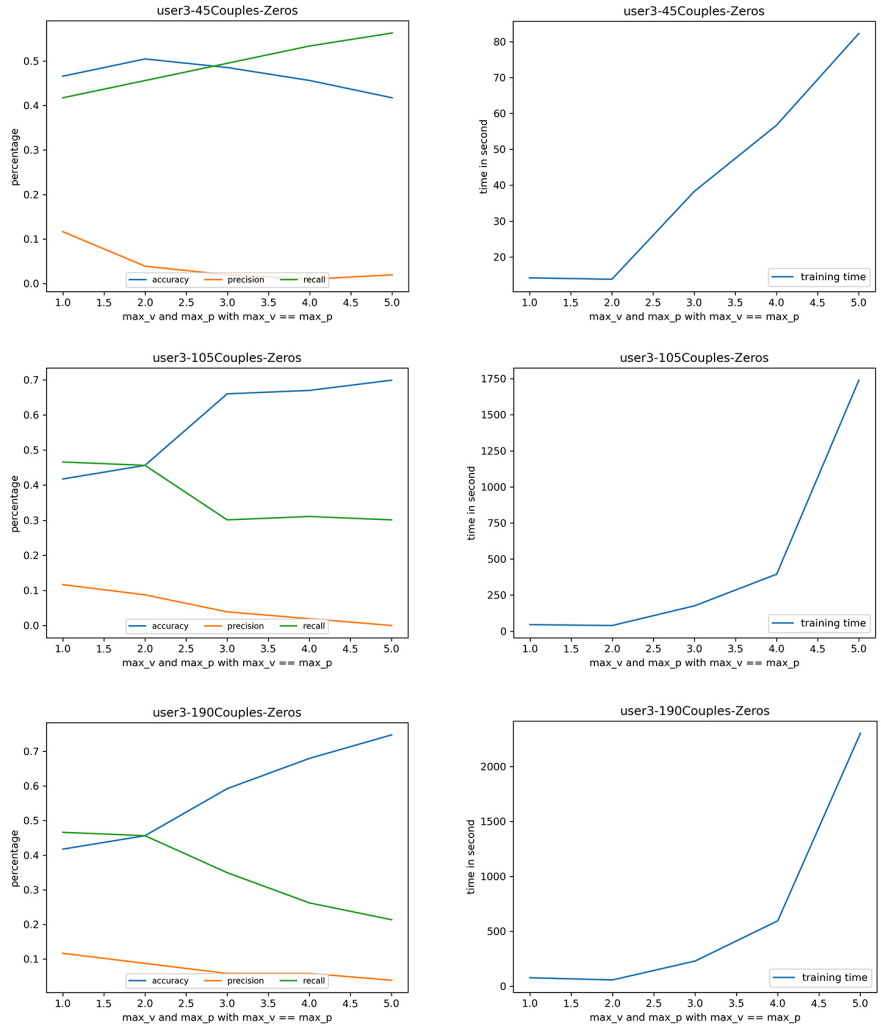

Figure 1: The graphs are divided by row respect to the size of the training set (45, 105 and 210 couples); and by column respect to estimators (accuracy, precision and recall) and execution time. These graphs are about ILASP results on user 3 when it is used as global approximator of the neural network. Note that on the abscissa we have the value of the parameters maxv and maxp, which increase equally. As cab be seen the execution time grows exponentially both with the size of training set and the increas of maxp and maxv

space, avoiding the loss of information as much as possible by exploiting the base concept of PCA, that is to make the first PCs as the ones which explain the higher variance of the dataset.

Given the theory returned with this technique, which will have a shape like the following:

```
:~ value(pc0,V1).[V1@1, V1]
```

we can understand the feature involved by retro-projecting, from the PC of interest, the features which has higher weight. Of course, the effectiveness of this technique is affected by the choice

	accuracy	precision	recall	execution time (seconds)
Indirect PCA	62.67%	39.52%	39.48%	7122.82
Direct PCA (5PC)	67.10%	37.75%	39.08%	4.86
Direct PCA (10PC)	72.09%	44.84%	49.02%	8.19
Direct PCA (15PC)	66.90%	36.63%	37.21%	30.47
Direct PCA (20PC)	65.52%	35.11%	35.58%	43.57

Table 4
Results in global approximation with *indrect* and *direct* PCA approaches over 10 user with training set of 190 couples. Accuracy, precision and recall are about performances reached on the test set, of 105 couples, while execution time refer to the time required from ILASP during the training.

of how many feature are retro-projected from the PC. For instance we can extrapolating these features using the equation 3, considering only the PC given by the theory; or considering the feature which has higher value for each PC in the theory, and so on.

6. Preliminary Results

In this section we present some preliminary results concerning a subset of 10 users. This is due to the high execution time faced during the ILASP experiments. These users are the ones with the top 10 performances obtained with the neural networks, considering the ones with the most balanced classes distribution. Unfortunately, we have a very unbalanced dataset with respect to uncertain relationship. In fact for each user, in average, the amount of pairs labeled with the uncertain relationship is 17.44% of the total. The results proposed concern the use of PCA with *indirect* and *direct* approach for ternary classification, which includes the uncertain relationship class. In the *indirect* approach, we set n to 8 (due to the execution time limitations) and preserve 15 features from the starting 48. Moreover, as another method to decrease dimensionality, we considered meta-class ingredients. In the *direct* approach, we tested a variable amount of PCs, namely *5*, *10*, *15* and *20*. Finally, the training and test sets are generated by randomly sampling from the recipes in the space of the feature (avoiding same recipes in train and test set) and using them to build the pairs. Training set has 190 pairs, while the test set has 105 pairs, with the labels being generated from the predictions of the Neural Network. The results are proposed in the table 4.

The first thing that can be seen is that there is a huge execution time difference between the *indirect* PCA and other cases. Furthermore, the best performance are obtained with the *direct* PCA method using the first 10 PCs. To better comprehend this difference, let us consider the results for the third user in the dataset using the *indirect* PCA method (in Figure 1) and the *direct* PCA method (in Figure 2). We can clearly see how in the first case (*indirect* PCA) the execution time grows exponentially as the parameters *maxv* and *maxp* grow, while in the second it grows linearly. Note that, in both cases, the dimensionality of the feature space is set to 15. We believe the main reason behind this behavior is that the PCA, from the starting feature space, creates a new orthonormal basis. Thus, when ILASP is trained in the *direct* PCA case, it has a simpler feature space compared to the starting one, which is used (reduced to 15 feature) in *indirect* PCA case.

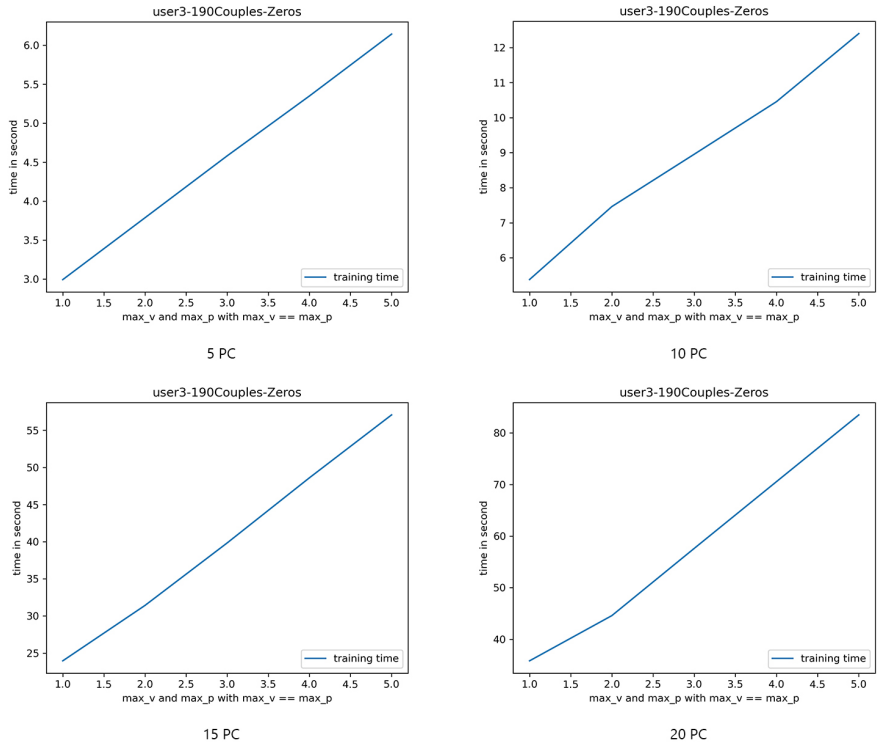

Figure 2: Execution time of ILASP when used as global approximator for user 3 with the use of the direct PCA method on a training set of 190 pairs, considering the number of PC = 5, 10, 15 and 20

	accuracy	precision	recall	execution time (seconds)
Indirect PCA	73.51%	49.19%	47.60%	10855.21
Direct PCA (5PC)	70.44%	41.16%	36.78%	4.03
Direct PCA (10PC)	72.45%	44.12%	39.78%	16.90
Direct PCA (15PC)	74.36%	44.82%	40.54%	26.72
Direct PCA (20PC)	74.87%	46.74%	41.76%	38.90

Table 5
results of ILASP when used directly on the preference dataset with *indrect* and *direct* PCA approaches

Another point to consider is that the *direct* PCA method reaches the best performance with the use of the first 10 PC, decreasing with the 15 and 20 PC cases. This could be a sort of early overfitting caused by the fact that both the feature space and the ground-truth are over-simplified by the combined use of *direct* PCA method and the use of the predictions of the neural network as labels. This is further supported by the results in Table 5, obtained using ILASP directly on the preferences contained in the dataset, labeled with the answers of the users, rather than with the predictions of the neural networks.

The last thing that has to be noted, is the difference that accuracy results has over precision and recall results. Beyond the already mentioned unbalanced dataset, we have observed that ILASP is not good to predict the uncertain relationship cases with the proposed approach. Using

the cost concept, ILASP classifies a pair (i_1, i_2) as i_1 is preferred to i_2 if, at higher penalty level in which the cost of the two element differs, the cost of i_1 is strictly less than the cost of i_2, regardless of how much big is this difference. The same applies for the case in which i_2 is preferred to i_1. This, however, doesn't hold in the case of uncertain, in which the costs of i_1 and i_2 have to be the same in every penalty level, which is statistically less likely to be observed as compared to the other two cases. Thus, starting from a unbalanced dataset, it is harder for ILASP to produce a theory for the uncertain cases compared to the other two preference relationships.

7. Conclusion and future developments

In this work, we present the complete workflow we followed to experiment with the use of ILP to explain black-box preference learning system. In this work doesn't yet test the accessibility of the output given by ILASP for the non-expert user, but this is one point in which we already give some effort and that we will go in deep in future works.

First, similarly to [9], we created a food preference dataset based on a survey where participants ranked Italian recipes. However, our dataset has a much higher dimensionality, and therefore it is a bigger challenge for logic-based learning algorithms.

We asked each surveyed user to provide orderings for three (partially overlapping) sets of recipes. The choice of asking users to provide *three* such orderings was partly due to technical limitations, since our designated tool for surveys – Microsoft Forms – did not allow users to order more than 10 elements for each survey, and partly because we wanted to study the more complicated case of a partial (rather than a total) ordering with inconsistencies and pairwise indifferent items. Then, to disambiguate, we propose an algorithm to get a full (partial) ordering from the three different orderings. In future work, it may be worth investigating how the number and structure of the surveys may affect the proposed approach. We did not just collect user preferences concerning recipes: in order to validate our approach we collected participants' opinions on some of the features, asking them questions such as "how do you like parmesan cheese?". In future work we will be able to use this information as a ground truth to compare our learned theories to.

We tested the ILASP framework as global approximator of a fully dense NN model, which is one of the most widely used deep learning methods, trained to classify users' preferences. Because of high execution times, we integrated PCA in the workflow, both with a *indirect* and *direct* approach. Our results were encouraging: we were able to achieve higher accuracy (72.09%) and significantly decreased computation time. In fact, we went from an average execution time of about 2 hours to less than 1 minute.

In future work, and following [6], we will use ILASP as local approximation of fully dense NNs, and perform similar experiments as those of the global approximation case. Another point we left behind in this work, but need to be investigated, is how to make the *direct* PCA approach transparent by showing how it is possible to translate those opaque theories into human understandable ones. We also found out that ILASP does not perform well when it comes to uncertain relationships, so we are considering repeating the experiments as a binary classification task, thus discarding uncertain relationships over the dataset.

Acknowledgments

The author Fabio Aurelio D'Asaro acknowledges the funding and support of PON "Ricerca e Innovazione" 2014-2020 (PON R&I FSE-REACT EU), Azione IV.6 "Contratti di ricerca su tematiche Green" in the context of the project titled "Il miglioramento dell'algorithmic fairness in ambito assicurativo: Un'analisi concettuale a partire da uno studio di caso".

References

[1] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, D. Pedreschi, A survey of methods for explaining black box models, ACM computing surveys (CSUR) 51 (2018) 1–42.

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

[3] M. Law, A. Russo, K. Broda, Inductive learning of answer set programs, in: European Workshop on Logics in Artificial Intelligence, Springer, 2014, pp. 311–325.

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

[5] M. Law, A. Russo, K. Broda, Inductive learning of answer set programs from noisy examples, arXiv preprint arXiv:1808.08441 (2018).

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

[7] A. Caliskan, J. J. Bryson, A. Narayanan, Semantics derived automatically from language corpora contain human-like biases, Science 356 (2017) 183–186. URL: https://science.sciencemag.org/content/356/6334/183. doi:10.1126/science.aal4230. arXiv:https://science.sciencemag.org/content/356/6334/183.full.pdf.

[8] P. Schramowski, C. Turan, S. Jentzsch, C. Rothkopf, K. Kersting, The moral choice machine, Frontiers in Artificial Intelligence 3 (2020) 36. URL: https://www.frontiersin.org/article/10.3389/frai.2020.00036. doi:10.3389/frai.2020.00036.

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

[10] J. Rabold, M. Siebers, U. Schmid, 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, 2018, pp. 105–117.

[11] J. Rabold, H. Deininger, M. Siebers, U. Schmid, Enriching visual with verbal explanations for relational concepts – combining lime with aleph, in: P. Cellier, K. Driessens (Eds.), Machine Learning and Knowledge Discovery in Databases, Springer, 2020, pp. 180–192.

[12] F. Shakerin, G. Gupta, Induction of non-monotonic logic programs to explain boosted tree models using lime, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, 2019, pp. 3052–3059.

[13] A. Srinivasan, The Aleph manual, 2004. URL: http://www.cs.ox.ac.uk/activities/machinelearning/Aleph/.
[14] C. Rebelo de Sá, W. Duivesteijn, C. Soares, A. Knobbe, Exceptional preferences mining, in: T. Calders, M. Ceci, D. Malerba (Eds.), Discovery Science, Springer International Publishing, Cham, 2016, pp. 3–18.
[15] M. T. Ribeiro, S. Singh, C. Guestrin, "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, 2016, p. 1135–1144. URL: https://doi.org/10.1145/2939672.2939778. doi:10.1145/2939672.2939778.
[16] J. Ferreira, M. de Sousa Ribeiro, R. Gonçalves, J. Leite, Looking inside the black-box: Logic-based explanations for neural networks, in: Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning, volume 19, 2022, pp. 432–442.
[17] J. Fürnkranz, Preference learning, Springer-Verlag Berlin Heidelberg, 2010.
[18] M. Gurrieri, X. Siebert, P. Fortemps, S. Greco, R. Słowiński, Advances on Computational Intelligence, 2012, pp. 613–623.
[19] Y. Zhou, Y. Liu, J. Yang, X. He, L. Liu, A taxonomy of label ranking algorithms, Journal of Computers 9 (2014) 557–565.