Learning Binary Preference Relations

A Comparison of Logic-based and Statistical Approaches

Nunung Nurul Qomariyah
Artificial Intelligence Group
Computer Science, University of York
York, United Kingdom
nq516@york.ac.uk

Dimitar Kazakov
Artificial Intelligence Group
Computer Science, University of York
York, United Kingdom
dimitar.kazakov@york.ac.uk

ABSTRACT

It is a truth universally acknowledged that e-commerce platform users in search of an item that best suits their preferences may be offered a lot of choices. An item may be characterised by many attributes, which can complicate the process. Here the classic approach in decision support systems - to put weights on the importance of each attribute - is not always helpful as users may find it hard to formulate their priorities explicitly. Pairwise comparisons provide an easy way to elicit the user's preferences in the form of the simplest possible qualitative preferences, which can then be combined to rank the available alternatives. We focus on this type of preference elicitation and learn the individual preference by applying one statistical approach based on Support Vector Machines (SVM), and two logic-based approaches: Inductive Logic Programming (ILP) and Decision Trees. All approaches are compared on two datasets of car preferences and sushi preferences collected from human participants. While in general, the statistical approach has proven its practical advantages, our experiment shows that the logic-based approaches offer a number of benefits over the one based on statistics.

CCS CONCEPTS

•Information systems →Recommender systems;

KEYWORDS

recommender systems; e-commerce; machine learning; inductive reasoning; inductive logic programming; pairwise and mixed-type data comparisons

ACM Reference format:
Nunung Nurul Qomariyah and Dimitar Kazakov. 2016. Learning Binary Preference Relations. In *Proceedings of Joint Workshop on Interfaces and Human Decision Making for Recommender Systems, Como, Italy, August 27, 2017 (IntRS 2017),* 5 pages.
DOI:

1 INTRODUCTION

Preference learning (PL) is a subtopic in machine learning that works with an ordinal dataset, either in partial or full order. Nowadays, PL plays an important role in machine learning research and practice because the ordinal data itself is used frequently in many areas, such as behavioural, medical, educational, psychological and social science [5]. For these domains, people can express their unique "value of preference" that may differ from others. For example, some buyers may give a rating on a Likert scale to show whether they like a certain product or not; some paper submissions may be weakly accepted, rejected or accepted depending on the review results. PL is the important step in the beginning stage of the recommender system process. The preferences that have been learned will be very useful to produce the recommendation list for the user.

1.1 Pairwise Preferences

There exist two main ways of modelling preferences: quantitative and qualitative preferences. The first modelling is associated with a number (or a quantity) representing the values of preferences (e.g., "my preference for car type is a sedan"), while the second type of modelling relates each item via pairwise comparisons (e.g., "I prefer car 1 over car 2"). The first model is not quite easy for everyone since humans are not always comfortable to express their preferences directly in terms of a value. It is normally much easier and arguably more natural to provide information about preferences in separate pieces, preferably in a qualitative way [4]. In practice, this is achieved through queries consisting of pairs of items along with their descriptions, where the user only needs to select the better of the two items. The use of pairwise comparisons in preference learning is still limited, although there are exceptions [2, 7, 13, 15]. This is not only because the approach is yet to be adopted by the major e-commerce companies, but also as choosing the most useful pairs and building a hypothesis about the user preferences are still challenging issues. This paper will focus on the second modelling preferences approach as illustrated in Figure 1.

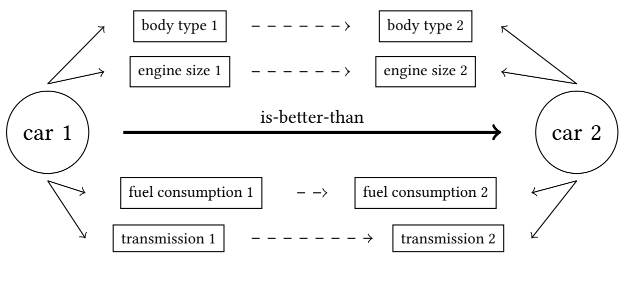

Figure 1: User annotation

Figure 1 shows how we derive conclusions about preferences regarding combinations of individual attributes of the form "car 1 is-better-than car 2". The bold arrow represents the annotation from the user and the dotted arrows show possible implications about individual attributes that the learning algorithm will consider.

1.2 Problem Statement

Supervised learning is a type of machine learning algorithm in which the learner receives a set of labelled examples as training data and makes predictions for all unseen data points [9]. This matches the description of our problem to make a prediction of the user's preferences. One common type of the supervised learning problem is binary classification, which we learn from two classes. The nature of our pairwise dataset lends itself to two big learning tasks, namely:

(1) Learning to order pairs of items (the relative order of preferences).
    In this task, we learn about how the items relate to the other items through the "better-than" relationship.
(2) Learning the top preference class (the best of all).
    While in this task, we learn the characteristics of the group of items that cannot be shown to be inferior to any other item.

We provide the evaluation of a number of existing supervised machine learning algorithms, explained in Section 3, to predict the users' individual preferences. The experiments in this paper are focused only on the first learning task, i.e. learning the relative order of preferences.

2 MODELLING PARADIGMS

AI research has tended to fall into two largely separate approaches: logical and statistical, in which the former tends to emphasize handling complexity, while the latter focuses on the uncertainty [3]. The first approach represents knowledge symbolically and the system attempts to reason using the symbolic knowledge. Systems that fall into this category include Inductive Logic Programming (ILP), classical planning, symbolic parsing, rule induction, etc. The second approach uses mathematical function to build the model. Systems that fall into this category include Naive Bayes, SVM, k-nearest neighbour, neural networks, etc. The mapping from representation on to the choice of machine learning algorithms and their implementation used here is shown in Table 1.

Table 1: Representation → algorithm → implementation

Representation	Algorithm	Implementation
Propositional Logic	Decision Tree	`fitctree` on Matlab
First-Order Logics	ILP	Aleph on yap
Statistical	SVM	`fitcsvm` on Matlab

2.1 Logic-based Approaches

In this paper, we show the results of systems that come from two families of logic: zero-order (propositional) and first-order.

2.2 Propositional Logic

Propositional Logic is concerned with propositions and their interrelationships (logical connectives). The notion of a proposition here will not be defined precisely – it suffices to say that a proposition is a possible condition of the world that is either true or false, e.g., the possibility that it is raining, the possibility that it is cloudy, and so forth [6]. Learning in this logic also follows the restrictions given by the representation, i.e. both concepts and facts are expressed through a set of propositions and logical connectives. Either examples and the hypothesis produced are enumerated in all possible values. One example of a learning algorithm which uses this logic is Decision Tree learning. The main reason to include the Decision Tree algorithm in our experiment is because it can produce readable rules for further analysis, which, for instance, can be used to produce the recommendation list.

2.3 First-Order Logics

First-Order Logics are more expressive than propositional logic, as they make use of variables and quantifiers (both universal and existential) to describe a concept. Inductive Logic Programming (ILP) is one of the learning algorithms which uses this logic to represent both examples and concepts (models) learnt [12]. More specifically, ILP uses the Horn clauses subset of First-Order Logic. ILP-based learners include FOIL [14], Golem [11], Aleph [16] and Progol [10]. We use Aleph in our experiment to learn a binary classifier from both positive and negative examples.

2.4 Statistical Approaches

We compare the results of logic-based learning with a statistical machine learning approach, namely, Support Vector Machines (SVM). In a binary classification problem, the SVM searches for the optimal linear separator of all data points in an n-dimensional space then use it to make predictions for new data. The method has previously been used by Qian et. al. [13] in a setup similar to ours, with good results.

3 EXPERIMENTS

3.1 Dataset

We use two publicly available datasets [1] [8]. Both the sushi and the car datasets have 10 items to rank which leads to 45 preference pairs per user. We take 60 users from each dataset and perform 10-fold cross validation for each user's individual preferences.

3.1.1 Car preferences dataset. In the car preferences dataset [1], there are 10 items with 4 features used in their experiment. The users were asked to choose the better of two cars as described by their attributes. The data was collected from 60 different users from the United States through Amazon's Mechanical Turk. The following ten car profiles in Table 2 were presented to the participants.

We treat all of these attributes as categorical in order to make the experiment comparable, so the last column (engine size) was discretized as follows:

- Small: for engine size under 3.5L
- Medium: for engine size from 3.5L and less than 4.5L
- Large: for engine size of 5.5L and over.

The other attributes are coded as:

- Body type: sedan (1), suv (2)
- Transmission: manual (1), automatic (2)
- Fuel consumption: hybrid (1), non-hybrid (2)

Table 2: Car profiles in the dataset

ID	Body Type	Transmission	Fuel Cons.	Engine Size
1	suv	manual	non-hybrid	2.5L
2	sedan	automatic	hybrid	5.5L
3	sedan	manual	non-hybrid	4.5L
4	sedan	manual	non-hybrid	6.2L
5	suv	manual	non-hybrid	3.5L
6	suv	automatic	hybrid	3.5L
7	sedan	automatic	hybrid	3.5L
8	suv	automatic	hybrid	2.5L
9	sedan	automatic	non-hybrid	3.5L
10	suv	automatic	non-hybrid	4.5L

3.1.2 Sushi preferences dataset. The sushi preferences dataset [8] have 7 attributes: style, major, minor, heaviness, how frequently consumed by a user, price and how frequently sold. This dataset contains 5000 users providing their preferences in full order. We convert each user's full order of preferences into a set of pairwise preferences. In our experiment, we take only 60 users from this dataset to make the size equal with the car dataset.

3.2 Experiment settings

We run the SVM and Decision Tree CART algorithm on Matlab R2016a running on Mac OS X version 10.11.2. For the ILP algorithm, we run Aleph 5.0 on a Prolog compiler, yap 6.3.2. We use 10-fold cross validation method for all experiments. In this section, we explain all the examples and results using car dataset description. The experiment for sushi dataset follows the same settings as for the car dataset.

We feed the learner 2 sets, containing positive, resp. negative examples. The positive examples are a set of correctly ordered pairs for each user. Then we build a set of negative examples from the opposite order of the user's preferences. For each user, we have a complete set of 90 observations, consisting of 45 positive examples and 45 negative examples.

The numeric input for SVM and Decision Tree is shown as below:

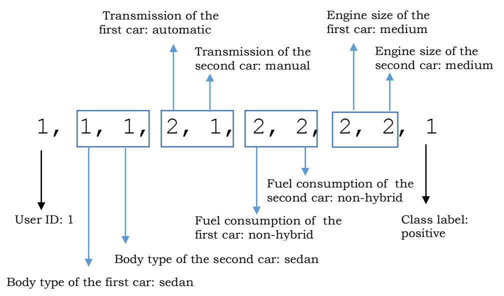
Transmission of the first car: automatic
Engine size of the first car: medium
Transmission of the second car: manual
Engine size of the second car: medium
1, 1, 1, 2, 1, 2, 2, 2, 2, 1
Fuel consumption of the second car: non-hybrid
User ID: 1
Fuel consumption of the first car: non-hybrid
Class label: positive
Body type of the second car: sedan
Body type of the first car: sedan

Figure 2: Input for SVM and Decision Tree Algorithms

Each row represents a user's preference on a pair of cars. The first column represents the user ID followed by the next 8 numeric attributes, and the last column is the class label (1=positive; -1=negative).

```
bt(car8,car2).
bt(car2,car9).
bt(car3,car10).
bt(car2,car3).
```

(a) File in '.f' extension containing positive examples

```
bt(car2,car8).
bt(car9,car2).
bt(car10,car3).
bt(car3,car2).
```

(b) File in '.n' extension containing negative examples

Figure 3: Input for Aleph system

The input for Aleph is shown in Figure 3. Aleph uses separate files to differentiate between positive and negative examples. One line in the positive examples file (Figure 3a) states: `bt(car8,car2)`, this means that we specify "car 8 is better than car 2" as a positive example. As we learn about order of preferences, in the negative examples file we put the arguments (`car8` and `car2`) in the opposite ways: `bt(car2,car8)`), which says that "car 2 is better than car 8" is a negative example. We run the experiment for each user and using the same settings every time.

```
:- modeh(1,bt(+car,+car)).
:- modeb(1,carfuel(+car,#fuelconsumption,
                   +car,#fuelconsumption)).
carfuel(A,X,B,Y):- hasfuelcons(A,X), car(A), car(B),
                   hasfuelcons(B,Y), X\=Y .

fuelconsumption(hybrid).
fuelconsumption(nonhybrid).
car(car1).
car(car2).
hasfuelcons(car1,nonhybrid).
hasfuelcons(car2,hybrid).
```

Figure 4: Aleph background knowledge

Aleph has a different way to represent the data and hypothesis. It uses Horn clause as shown in Figure 4 which means that we want to allow Aleph to consider the hypotheses produced by the template *'in a pair, the first car is better than the second car, if the first car has fuel consumption of (hybrid or non-hybrid) and the second car has fuel consumption of (hybrid or non-hybrid), in which the two cars do not have the same type of fuel consumption'.*

From the head mode (`modeh`), we can see that we want to allow Aleph to build hypothesis by looking for any examples that match this pattern `bt(+car,+car)` (see Figure 3); `bt()` means a predicate 'better than', while `+car` means it is an input of type 'car'. We set the body modes (`modeb`) as a function `carfuel(+car, #fuelconsumption, +car, #fuelconsumption)` which has two types of input: 'car' and 'fuel consumption'. How this function works is specified in the following line:
`carfuel(A,X,B,Y) :- hasfuelcons(A,X), car(A), car(B), hasfuelcons(B,Y), X\=Y`, which means that we want to create hypothesis for the pair of car that has different fuel consumption.

3.3 Results

The accuracy of the three algorithms is shown in Table 3. Our experiment shows that in terms of accuracy, SVM significantly outperformed Aleph and Decision Tree in car dataset, but Decision Tree shows the highest accuracy amongst the other algorithms on sushi dataset. According to ANOVA test, there is a significant difference amongst the accuracy of the algorithms as shown in Table 4. ANOVA is conceptually similar to multiple two-sample t-tests but is more conservative (results in less type I error). We also perform several experiments with the algorithms by varying the proportion of training examples and test it on 10% of examples. For a more robust result, we validate each cycle with 10-fold cross validation. We take the average accuracy of both datasets and the result of this experiment is shown in Figure 5.

Table 3: Mean and standard deviation of 10-fold cross validation test

	SVM	DT	Aleph
car dataset	0.8317±0.12	0.7470±0.10	0.7292± 0.08
sushi dataset	0.7604±0.09	0.8094±0.06	0.7789±0.06

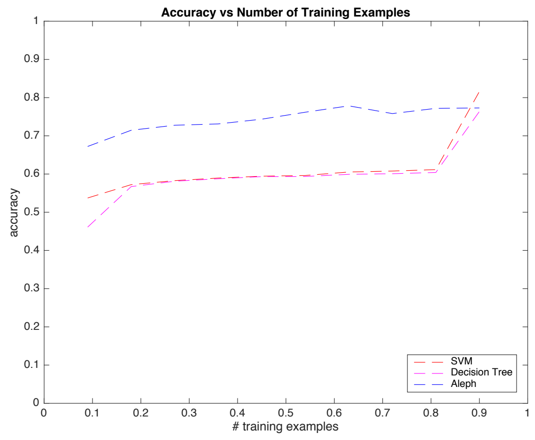

Figure 5: Accuracy by varying number of training examples

Table 4: ANOVA ($\alpha = 0.05$) statistical results

Dataset	F	*p-value*	F-crit
Car preferences	17.3163	1.35×10^{-7}	3.0470
Sushi preferences	6.7198	1.54×10^{-3}	3.0470

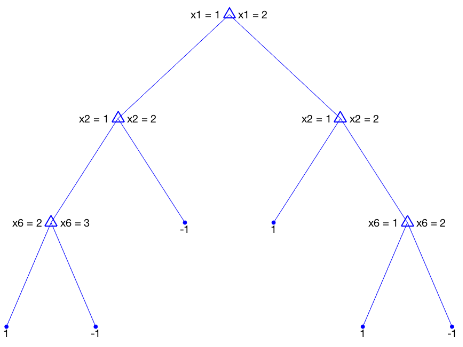

Figure 6: Decision Tree rules

Some of the rules produced by Decision Tree are shown in Figure 6. We interpret `x1` as body type (see Figure 2) of the first car, `x2` as body type of the second car, `x3` as transmission of the first car, `x4` as transmission of the second car, `x5` as fuel consumption of the first car, `x6` as fuel consumption of the second car, `x7` as engine size of the first car and `x8` as engine size of the second car. The Decision Tree uses the rules to predict the new unseen data. Some of the rules in Figure 6 can be read as below:

- If the body type of the first car is sedan and the body type of second car is sedan and the fuel consumption of second car is non-hybrid, then it is a positive class.
- If the body type of the first car is suv and body type of the second car is sedan, then it is a positive class.

```
bt(A,B) :-
    carbodytype(B,sedan,A,suv).
bt(A,B) :-
    cartransmission(B,manual,A,automatic).
bt(A,B) :-
    carfuel(B,nonhybrid,A,hybrid).
```

Figure 7: Aleph's consistent hypotheses

Some of the consistent hypotheses produced by Aleph are shown in Figure 7. They can be read as below:

- any car A is better than any car B if car A has body type: suv and car B has body type: sedan.
- any car A is better than any car B if car A has transmission: automatic and car B has transmission: manual.
- any car A is better than any car B if car A has fuel consumption: hybrid and car B has fuel consumption: non-hybrid.

4 DISCUSSION AND ANALYSIS

The results of the three different approaches are quite interesting. The statistical approach, SVM, works very well when we codify all the data into numerical and it can be very practical. On the other hand, Aleph and Decision Tree also show the good results with some advantages of a more readable model (a set of rules) and they can work well for both numerical or categorical data. In contrast to Decision Tree, Aleph, as the first order logic learner algorithm, has a special feature to be more flexible in defining the rules. For example, while the Decision Tree only expands the tree for each individual feature, we can let Aleph build the hypothesis based on the comparison of the same attributes in pair (e.g. `bt(A,B):-carbodytype(B,sedan,A,suv)`) means that the rule says 'car A is better than car B if car A is suv and car B is sedan'). The set of rules that has been produced by Aleph can be used for further analysis, i.e. to recommend the best characteristics of items to the users.

5 CONCLUSIONS AND FUTURE WORK

In this paper, we reported an experiment with the logic-based and statistical learning algorithms to predict user's individual preferences. Although the statistical approach shows a very good result, the logical approaches can be more promising in such ways:

- The logical approach can be used to learn from a smaller number of examples, as we cannot guarantee that we will have enough responses from the users.
- We can use the flexibility of adding a background knowledge to the logical learner to limit the rules and built more meaningful result (e.g. using Aleph implementation).
- The results of logical approach are more readable and easier to be understood for the further use (i.e. provide a recommendation).

The main benefit of the paper is methodological, establishing a comparison between statistical and logic-based methods, and that the full benefits of logic-based methods are to be expected for richer representations, where a range of background concepts (either provided by the software designers or gradually inferred from the data) can be used to model users with complex preferences.

All of this experiment can only be performed in batch mode. The dataset we used in this experiment only have a limited number of choices, which limits the number of possible combinations of features between the pairs. We plan to implement an algorithm which employs active learning to help with incremental learning. It will be tested in another forthcoming experiment, which will also offer the learner the choice of explicitly asking the users to confirm or reject parts of the model and its implications regarding the best choice of car.

ACKNOWLEDGMENTS

This work is supported by Indonesia Endowment Fund for Education (LPDP) , Ministry of Finances, The Republic of Indonesia [grant no. 20130622010016].

REFERENCES

[1] E. Abbasnejad, S. Sanner, E. V. Bonilla, and P. Poupart. 2013. Learning Community-based Preferences via Dirichlet Process Mixtures of Gaussian Processes. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI)*.

[2] S. Balakrishnan and S. Chopra. 2010. Two of a Kind or the Ratings Game? Adaptive Pairwise Preferences and Latent Factor Models. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*. 725–730. https://doi.org/10.1109/ICDM.2010.149

[3] Pedro Domingos, Daniel Lowd, Stanley Kok, Aniruddh Nath, Hoifung Poon, Matthew Richardson, and Parag Singla. 2016. Unifying logical and statistical AI. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*. ACM, 1–11.

[4] Carmel Domshlak, Eyke Hüllermeier, Souhila Kaci, and Henri Prade. 2011. Preferences in AI: An overview. *Artificial Intelligence* 175, 7-8 (2011), 1037–1052.

[5] Johannes Fürnkranz and Eyke Hüllermeier. 2010. *Preference learning: An introduction*. Springer.

[6] Michael Genesereth and Eric Kao. 2013. *Introduction to Logic*. Morgan and Claypool. 163 pages. https://doi.org/10.2200/S00518ED2V01Y201306CSL006

[7] B.S. Jensen, J. Saez Gallego, and J. Larsen. 2012. A predictive model of music preference using pairwise comparisons. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*. 1977–1980. https://doi.org/10.1109/ICASSP.2012.6288294

[8] Toshihiro Kamishima. 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.

[9] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. 2012. *Foundations of machine learning*. MIT press.

[10] Stephen Muggleton. 1995. Inverse entailment and Progol. *New generation computing* 13, 3-4 (1995), 245–286.

[11] Stephen Muggleton, Cao Feng, et al. 1990. Efficient induction of logic programs. In *Proceedings of the First Conference on Algorithmic Learning Theory*, S. Arikawa, S. Goto, S. Ohsuga, and T. Yokomori (Eds.). Japanese Society for Artificial Intelligence, 368–381.

[12] Shan-Hwei Nienhuys-Cheng and Ronald De Wolf. 1997. *Foundations of inductive logic programming*. Vol. 1228. Springer Science & Business Media.

[13] Li Qian, Jinyang Gao, and HV Jagadish. 2015. Learning user preferences by adaptive pairwise comparison. *Proceedings of the VLDB Endowment* 8, 11 (2015), 1322–1333.

[14] J. Ross Quinlan. 1990. Learning logical definitions from relations. *Machine Learning* 5, 3 (1990), 239–266.

[15] Lior Rokach and Slava Kisilevich. 2012. Initial profile generation in recommender systems using pairwise comparison. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on* 42, 6 (2012), 1854–1859.

[16] A Srinivasan. 2003. The Aleph manual version 4. (2003).