Preprocessing Algorithm for Handling Non-Monotone Attributes in the UTA method

Alan Eckhardt[1] and **Tomáš Kliegr**[2]

Abstract. UTA is a method for learning user preferences originally developed for multi-criteria decision making. UTA expects that the input attributes are monotone with respect to preferences, which limits the applicability of the method and requires manual input for each attribute. In this paper, we propose a heuristic attribute preprocessing algorithm that transforms arbitrary input attributes into a space approximately monotone with respect to user preferences, thus making it suitable for UTA. In an experimental evaluation on several real-world datasets, preprocessing the input data with the proposed algorithm consistently improved the results in comparison with the UTA-ADJ variation of the UTA STAR algorithm.

1 Introduction

The UTA (UTilités Additives) method (Jacquet-Lagreze [9]) learns an additive piece-wise linear utility model. Although a relatively old approach, it is used as a basis for many recent utility-based preference learning algorithms, e.g. [8]. UTA takes a set of alternatives ordered according to user's preferences, and learns utility functions for each attribute. Using these functions, the utility for individual attribute values are combined into the overall utility for a given object.

UTA expects that the input attributes are monotone with respect to preferences, which not only requires manual input for each attribute, but also limits the applicability of the method. In this paper, we propose a heuristic attribute preprocessing algorithm that transforms arbitrary input attributes into a space approximately monotone with respect to user preferences, thus making it suitable for UTA.

This paper is divided into four sections. Section 2 describes the UTA method with focus on its monotonicity constraints. The proposed algorithm for learning local preferences is introduced in Section 3. Experimental evaluation of the algorithm is presented in Section 4, and the concluding remarks are in Section 5.

2 Related Work

UTA method [9] draws inspiration from the way humans handle decision making tasks at a level of abstraction provided by the microeconomic utility theory. The principal inductive bias used in the UTA method is the monotonicity of utility functions. The UTA method also incorporates additional assumptions, particularly piece-wise linearity and additivity of utility functions.

UTA method aims at inferring one or more additive value functions from a given ranking (weak ordering) on the reference set X of objects[3]. Each object is described by N criteria $G = \{g_1, \dots, g_N\}$. The evaluation scales of each criterion is given by the real-valued function $g_i : X \to [g_{i*}, g_i^*]$. The value g_{i*} is considered the worst level of the worth of the object from the decision maker's point of view with respect to criterion g_i and g_i^* the best level, $g_i(\mathbf{x})$ denotes the evaluation of object $\mathbf{x} \in X$ in criterion g_i. The method uses linear programming to find such N partial value functions u_i that best explain given preferences. The overall preference rating for an object $\mathbf{x}$ is computed as a sum of utility values across all criteria:

$$U(\mathbf{x}) = \sum_{i=1}^{N} u_i(\mathbf{x})$$

where $\mathbf{x}$ is an object and u_i are nondecreasing marginal value functions, which we call *partial utility functions*. It should be noted that the partial utility functions are piece-wise linear. For each criterion, the interval $[g_{i*}, g_i^*]$ is cut into $\alpha_i - 1$ equal intervals $[g_{i*}, g_i^2], \dots, [g_i^{\alpha_i - 1}, g_i^*]$. This discretization is not significant for nominal criteria; for these, the number of breakpoints α_i (including the end points g_{i*}, g_i^*) corresponds to the number of values of the criteria. However, for cardinal criteria, the marginal value of an object $g_i(\mathbf{x}) \in [g_i^j, g_i^{j+1}]$ is approximated using linear interpolation between the two nearest breakpoints g_i^j, g_i^{j+1}.

The *condition of the monotonicity* requires that if for $\mathbf{x}, \mathbf{x}' \in X$, object $\mathbf{x}$ is weakly preferred to object $\mathbf{x}'$, then for another object $\mathbf{x}'' \in X$, such that $g_h(\mathbf{x}'') \geq g_h(\mathbf{x})$, for all $h = 1, \dots, N$, object $\mathbf{x}''$ should be also weakly preferred to object $\mathbf{x}'$ [2].

In the area of Multi-Criteria Decision Making (MCDM), it is the role of the expert to derive the set of criteria from the properties of the object. The expert not only checks that the criteria meet the monotonicity requirement, but also orders the domain so that value g_{i*} is considered the worst and value g_i^* the best for criterion i.

If UTA is to be applied in wider machine learning context, rather than in the narrow area of decision support systems, a manual approach to transforming attributes to criteria is not feasible.

Kliegr in [10] proposed a non-monotonic extension of the UTA Star algorithm, called UTA-NM, which allows u_i to change direction from ascending to descending, thus allowing to use criteria[4] that do not meet the monotonicity requirement. Every change of the direction within one partial utility function is penalized to ensure that the resulting model is not overly complex and overfitted to the data.

In this paper, we investigate another option for dealing with non-monotonicity, which is based on an automatic transformation of the original, potentially non-monotonic attributes of the input objects

[1] Department of Software Engineering, Charles University in Prague, Czech Republic, email: eckhardt@ksi.mff.cuni.cz
[2] Department of Information and Knowledge Engineering, University of Economics in Prague, Czech Republic, and Multimedia and Vision Group, Queen Mary University, London, UK. email: tomas.kliegr@vse.cz

[3] Usually referred to as "actions" or "alternatives" in the UTA literature.

[4] Since the notion of criterion is closely tied with the monotonicity requirement, it would be more precise to speak of "attributes".

into criteria. The algorithm is heuristic, i.e. the resulting criteria are not guaranteed to meet the condition of monotonicity. The performance of this preprocessing step is evaluated against a baseline obtained with a non-monotonic UTA implementation coming out of UTA-NM.

3 Heuristic Algorithm for Transformation of Attributes to Criteria

Modern preference learning research deals directly with properties of input objects, an object **x** is typically described by a vector of attribute values $(x_1, \ldots, x_N)$, no special requirements on the attribute values are typically made [7]. In contrast, the original UTA model, and MCDA in general, abstracts from the original attributes of the object. It is already assumed that the *expert* has used these attributes to construct the criteria set G; each object is represented by criteria values $(g_1(\mathbf{x}), \ldots, g_N(\mathbf{x}))$. The individual criteria are assumed to meet several properties, including the condition of monotonicity introduced in the previous section.[5]

In this section, we present a heuristic algorithm that transforms the original values of the attributes, so that the result better meets the key requirement put on *criteria* addressed in this paper – the condition of monotonicity. Since the input for the algorithm is the preference rating of one user[6], we call it *local preference* transformation.

Using D_{A_i} to denote the domain of the original attribute i, the local preferences transformation can be viewed as function $f_i : D_{A_i} \to [g_{i*}, g_i^*]$, which is a "concretization" of the criterion function $g_i : X \to [g_{i*}, g_i^*]$ in the formal UTA model.

Each object **x** originally described by attribute values $x_1, \ldots, x_N$ is now described by values $f_1(x_1), \ldots, f_N(x_N)$. The intuition is that $f_i(x_i)$ is an "average" of ratings of the particular user across objects that have value x_i in attribute i, we denote this average as $r(\mathbf{x})$. Since the preference information which is consumed by the UTA method is in the form of weak order of alternatives rather than ratings, for the purpose of the local transformation algorithm this weak order needs to be transformed to ratings. A straightforward approach is to assign values in the $[0, 1]$ range corresponding to the position of the object in the weak preference order, with the best ranked object retrieving the highest rating 1 and the worst ranked object the lowest ranking 0, with the remaining ranks equidistantly distributed in the $[0, 1]$ range.

The definition of the transformation function f_i depends on the type of the input attribute, which can be either nominal or cardinal. In our present work ordinal attributes are not addressed, however, they can be cast to integer and handled as a cardinal data type. In the remainder of this section, we discuss definition of the local transformation function f_i first for nominal and then for cardinal attributes.

3.1 Nominal Attributes

"Representants" is a method for finding preferences over nominal attributes, which we proposed in [4]. It is based on the analysis of the distribution of ratings. The local preference function for a nominal attribute A_i has the following form:

$$f_i(a) = \frac{\sum_{\{\mathbf{x}|x_i=a\}} r(\mathbf{x})}{|\{\mathbf{x}|x_i = a\}|},$$

where a is an attribute value.

[5] In this paper, we make the assumption that the original N input attributes are converted to the same number of criteria.

[6] Decision maker in the MCDA terminology.

Figure 1 illustrates the computation on the notebook choice problem. Ratings of three hypothetical users with respect to the nominal *notebook colour* attribute are presented. For each user and colour, a frequency distribution of ratings is shown. Let us focus on one value only, e.g. black. Black-coloured notebooks are highly preferred for users 1 and 2. The vertical line in the histogram represents the average rating of each object with colour black - for both users, the average is close to 1. The average will be used as the *representative* rating of the value black for the given user, or in other words, preference of black colour.

Formally, the set of black objects is denoted as

$$\{\mathbf{x}|x_{colour} = black\}$$

and the sum of ratings of black objects is then

$$\sum_{\{\mathbf{x}|x_{colour}=black\}} r(\mathbf{x})$$

.

The resulting representative ratings do not necessarily maintain the monotonicity of ratings. Consider five objects (given in the order of increasing stated preference by the user) $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5$ with two attributes *colour* and *price*. The description of these objects with respect to these attributes is given by Table 1. Once the stated preference, originally given in terms of stars, is converted to a numerical rating, the original attributes can be transformed with the local preferences function obtaining $f_{colour}(black) = 0.375$ and $f_{colour}(red) = 0.58$, $f_{cpu}(intel) = 0.25$ and $f_{cpu}(amd) = 0.5$, $f_{cpu}(motorola) = 1.0$.

This example can be also used to illustrate the fact that the result of the local preferences transformation is not guaranteed to not violate the condition of monotonicity: $\mathbf{x}_4$ is preferred to $\mathbf{x}_3$, for $\mathbf{x}_2$ it holds that $g_h(\mathbf{x}_2) \geq g_h(\mathbf{x}_4)$, for all $h = 1, \ldots, N$, therefore $\mathbf{x}_2$ should also be weakly preferred to $\mathbf{x}_3$. Since this is not satisfied, the condition of monotonicity is violated.

Table 1. Notebook choice example – nominal attributes.

	object				
	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$
stated order	*	**	***	****	*****
rating	0	0.25	0.5	0.75	1
	original attribute values				
colour	black	red	red	black	red
cpu	intel	amd	intel	amd	motorola
	transformed values (criteria)				
colour	0.375	0.58	0.58	0.375	0.58
cpu	0.25	0.5	0.25	0.5	1

3.2 Cardinal Attributes

This section describes the proposed approach for transformation of cardinal (numerical) attributes.

Linear regression is a very useful method that finds a relation in a data set in the form of a linear function. In the local preferences transformation, *univariate* linear regression is used to find a relationship between each cardinal input attribute and the rating as the dependent variable. Expanding our notebook example, consider additional *notebook price* attribute. Applying the local transformation for cardinal attributes, notebook price is used as a regressor and the

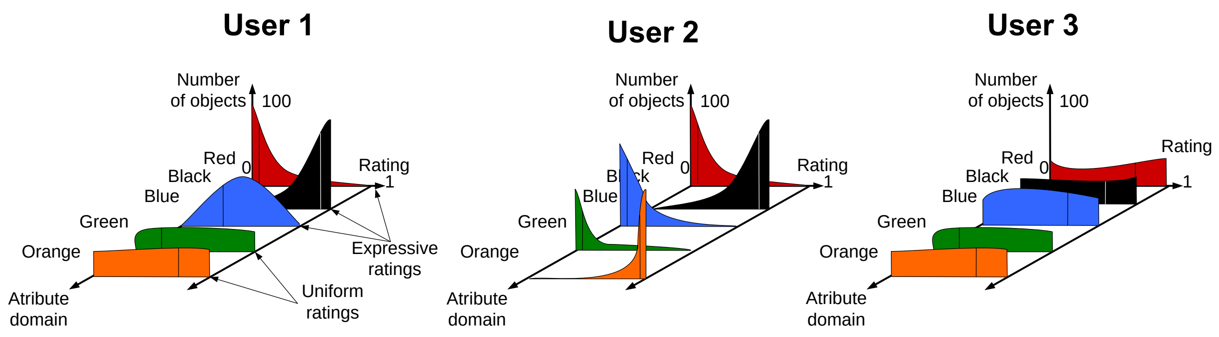

Figure 1. Illustrative ratings of three users for the notebook colour attribute.

user *rating* as the dependent variable. The result of the regression is a linear function of the form:

$$f_{price}(price) = \alpha * price + \beta$$

.

If the underlying attribute is of so called *cost* type, i.e. user preference decreases with increasing attribute value, a direct use of the attribute value x_i in place of the criterion value $g_i(\mathbf{x})$ would violate the condition that the value g_{i*} is considered the worst level of the worth of the object from the decision maker's point of view with respect to criterion g_i and g_i^* the best level (Figure 2b). Linear regression will generally solve this problem, rearranging the values appropriately.

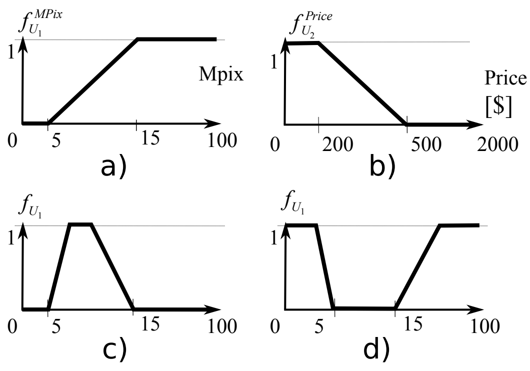

Figure 2. Four basic types of preference over cardinal domains

However, the weakness of linear regression is that it finds only linear orderings, therefore it does not generally fix a violation of the condition monotonicity. Figure 2 shows four typical types of preferences over cardinal attributes. Only two of them are linear. Linear regression is not able to find “hill” and “valley” types. These types can be learned using our method “Peak” proposed in [5].

We have considered also other possibilities for finding preferences over cardinal attributes. In our preliminary experiments on small training sets, using quadratic regression consistently worsened the results.

4 Experiments

4.1 Performance measures

We will use the following notation: $r(o)$ is the user's rating of object o, $\widehat{r}(o)$ is the rating predicted by the method.

Tau coefficient expresses the similarity of two ordered lists L_1, L_2. In our case, the first list L_1 is ordered according to the user's rating and the second list L_2 according to the rating estimated by the method. The lists are sorted in decreasing order, so that the most preferred objects are on the top of the lists. In the simplest case, the lists consist only of ids of objects. The tau coefficient is then computed according to the number of concordant pairs. A pair of objects o,p is concordant, if either o is before p in both lists, or p is before o in both lists. A pair that is not concordant, is discordant. Then, tau coefficient can be computed as follows:

$$\tau(L_1, L_2) = \frac{n_c - n_d}{1/2 * n * (n-1)},$$

where n_c is the number of concordant pairs and n_d is the number of discordant pairs. δ in following formula stands for Kronecker delta - $\delta(condition) = 1$ if *condition* is true, 0 otherwise.

$$n_c = \sum_{o,p \in \mathcal{X}} \delta(sgn(L_1(o) - L_1(p)) = sgn(L_2(o) - L_2(p)))$$

Another measure expressing the similarity of two lists is Pearson correlation. This measure is often used in machine learning for studying the dependency of two variables. If the correlation is high, the two lists are similar, if the correlation is low, the lists are different. The value of the correlation coefficient ranges from -1 to 1. Value -1 means lists with exactly inverse ordering, 1 corresponds to the same ordering. Correlation 0 means there is no connection between the two orderings.

$$corr(r, \widehat{r}) = \frac{\sum_{o \in \mathcal{X}} (r(o) - \bar{r})(\widehat{r}(o) - \bar{\widehat{r}})}{(n-1) s_r s_{\widehat{r}}},$$

where $\bar{r}$ is the average rating, $\bar{\widehat{r}}$ is the average predicted rating, s_r is the standard deviation of the original ratings and $s_{\widehat{r}}$ is the standard deviation of the predicted ratings.

UTA method is focused on preserving the order of objects rather than estimating their ratings, so it does not make sense to use the Root Mean Square Error (RMSE) metric, which is commonly used for other methods.

Two measures of computational performance will be studied. The first measure is the time required to train the method on a set of ranked objects. The second one is the time required to evaluate a testing object.

Table 2. Datasets with monotone class attribute from the UCI repository.

Dataset name
Car Evaluation
Contraceptive Method Choice
Nursery
Poker Hand
Post-Operative Patient
Teaching Assistant Evaluation
Wine
Wine Quality[1]

4.2 Datasets

UCI [6] contains a number of datasets with different properties. We were most concerned in classification tasks, where there is an ordering of classes. For evaluation of the UTA method, only datasets suitable for the object ranking task were relevant. E.g. the famous Iris dataset were excluded, because we are not able to order the three classes - Iris Setosa, Iris Versicolour and Iris Virginica - in a meaningful way. Every classification dataset of UCI was studied for the presence of monotonicity in the class attribute. The chosen datasets from UCI repository are in Table 2.

We acknowledge that using UCI for validation of user preference learning methods may not give representative results, since these are not real-world preference datasets.

4.3 Experimental Implementation

For the experimental evaluation, we used our implementation of non-monotonic UTA called UTA-ADJ. It is a based on similar ideas as the UTA-NM algorithm, introduced in our earlier work [10]. UTA-NM removes the monotonicity constraints imposed by the UTA Star algorithm. A penalization element is added to prevent overfitting by excessive number of changes in shape of partial utility functions, however, the penalization entails an excessive computational cost.

To remedy the computational issue, UTA-ADJ takes a different route to allow non-monotonicity in partial utility functions. UTA-ADJ runs the standard UTA Star algorithm multiple times, gradually testing the influence of placing a change of shape at individual breakpoints across all partial utility functions. The breakpoint in which the change of shape yields the largest increase in the objective function in comparison with the baseline, is retained.

This procedure is repeated until the preset threshold of maximum number of changes in shape is retained or the improvement in the objective function is lower than a preset minimum increase. In the first iteration, the baseline is the objective value attained by a fully monotonic run of the UTA Star algorithm, in subsequent iterations it is the best objective value from the previous iteration.

The computational costs of multiple runs of the UTA Star algorithm is in our experience much lower than the cost of a single UTA-NM run with the penalization constraints. Nevertheless, it should be noted that the computational advantage of UTA-ADJ over UTA-NM is inversely related to the maximum number of changes in shape threshold.

UTA-ADJ algorithm closely resembles the method proposed by Despotis and Zopounidis [3], with the difference that breakpoints, where the change of shape occurs, are not externally set parameters, but are determined automatically. Also, more changes of shape per partial utility functions are allowed.

The UTA-ADJ implementation is available as an interactive online application at `http://nlp.vse.cz/uta`.

4.4 Experimental Results

This section describes our experiments on the UCI datasets. The results are averaged across all the datasets listed in Table 2. For the given training set size, 20 different (randomized) training sets were used for each dataset. We compare UTA-ADJ with the (possibly) improved UTA+Local, which is also an UTA-ADJ run, but involving the local preferences transformation. The maximum number of changes in shape threshold was set to 2 for all experiments.

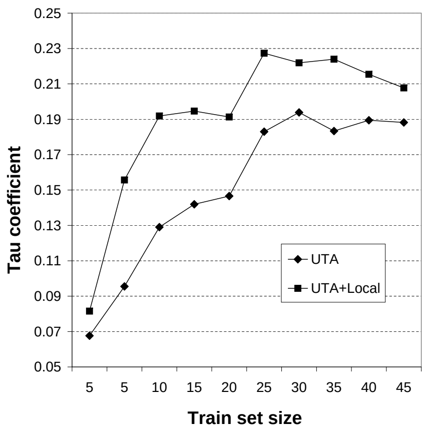

Figure 3. Tau rank coefficient for all datasets.

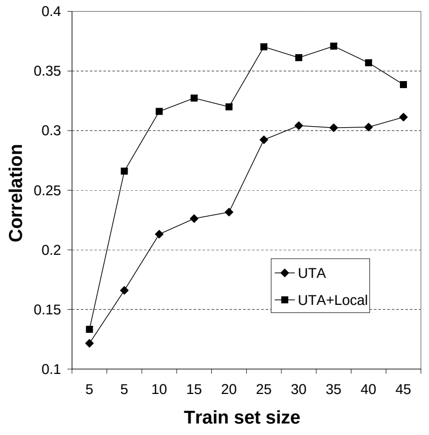

Figure 4. Correlation for all datasets.

Figure 3 presents a comparison of performance on individual datasets in terms of the tau coefficient. Here, the advantage of using local preferences is clear, the improvement of the value of tau coefficient for moderate train size is around 0.05. A comparable result for Pearson correlation can be observed in Figure 4. The relative increase in correlation and tau coefficient is about 28%.

The results are convincing - using local preferences with the UTA-ADJ variant of the UTA method significantly improved its performance in all performance measures compared to UTA-ADJ only baseline. Moreover, the time to train the classifier has also decreased. All results were significant at the level $p < 0.05$.

5 Conclusion

We proposed a preprocessing algorithm called *local preferences transformation*, which allows to use the UTA method with non-monotone attributes.

The experimental results confirm the benefits of the proposed approach as a preprocessing step for UTA-ADJ, a variant of the UTA method, which already has some adjustments to handle non-monotone attributes. We assume the effectiveness of our local preferences preprocessing algorithm to hold also for the common UTA Star algorithm. Nevertheless, an experimental evaluation of this hypothesis is a priority for future work.

A promising direction for extending our research is using more elaborate methods than linear regression for preprocessing cardinal attributes. Regarding nominal attributes, further work should be directed at investigation of circumstances, in which the proposed algorithm is and is not effective in maintaining the condition of monotonicity.

It would also be interesting to find out the relation between the degree of monotonicity of the data and the performance of UTA with/without applying the proposed local preferences transformation in the preprocessing phase.

The UTA method implementation used in the experiments is available in a form of an interactive web application at `http://nlp.vse.cz/uta`. We plan to make the local transformation algorithm available to the community by integrating it with this software.

Acknowledgment

We would like to thank the reviewers for their comments, which helped to improve the quality of this paper considerably. This research was supported by projects SVV-2010-261312, MŠM 0021620838 and grant GAČR P202/10/0761.

REFERENCES

[1] Paulo Cortez, António Cerdeira, Fernando Almeida, Telmo Matos, and José Reis, 'Modeling wine preferences by data mining from physicochemical properties', *Decision Support Systems*, (June 2009).

[2] Krzysztof Dembczyński, Wojciech Kotłowski, Roman Słowiski, and Marcin Szelag, 'Learning of rule ensembles for multiple attribute ranking problems', in *Preference Learning*, eds., Johannes Fürnkranz and Eyke Hüllermeier, 217–247, Springer-Verlag, (2010).

[3] D.K. Despotis and Constantin Zopounidis, 'Building additive utilities in the presence of non-monotonic preference', in *Advances in Multicriteria Analysis*, eds., P.M. Pardalos, Y. Siskos, and C. Zopounidis, 101–114, Kluwer Academic Publisher, Dordrecht, (1993).

[4] Alan Eckhardt and Peter Vojtáš, 'Considering data-mining techniques in user preference learning', in *2008 International Workshop on Web Information Retrieval Support Systems*, pp. 33–36, (2008).

[5] Alan Eckhardt and Peter Vojtáš, 'Learning user preferences for 2cp-regression for a recommender system', in *SOFSEM '10: Proceedings of the 36th Conference on Current Trends in Theory and Practice of Computer Science*, pp. 346–357, Berlin, Heidelberg, (2010). Springer-Verlag.

[6] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[7] Johannes Fürnkranz and Eyke Hüllermeier, *Preference Learning*, Springer, 2010.

[8] Salvatore Greco, Vincent Mousseau, and Roman Slowinski, 'Ordinal regression revisited: multiple criteria ranking using a set of additive value functions', *European Journal of Operational Research*, **191**(2), 415–435, (December 2008).

[9] E. Jacquet-Lagreze and Yannis Siskos, 'Assessing a set of additive utility functions for multicriteria decision-making, the uta method', *European Journal of Operational Research*, **10**(2), 151–164, (June 1982).

[10] Tomáš Kliegr, 'UTA - NM: Explaining stated preferences with additive non-monotonic utility functions', in *Proceedings of Workshop Preference Learning in ECML/PKDD'09*, (September 2009).