Preference Detection in Multi-Attribute Multi-item Choice Environments

Amir Konigsberg and Ron Asherov
General Motors R&D
Email: amir.konigsberg@gm.com, rasherov@gmail.com

***Abstract*—We propose a novel method for evaluating, detecting, and inferring preferences in choice situations involving items with multiple attributes. Our method locates characteristics that reflect the relative weight that a user gives to varying attributes belonging to an item, when these attributes are combined into a unified multi-attribute utility function. Our method enables the attainment of coefficients that reflect the preferential prism of a user in relation to items with multiple attributes. Broadly, we translate the form of a u-function into an inequality with scalar variables which define half-spaces on a plane. These half-spaces intersect and form closed shapes (in a k-dimensional world). The closed shapes with the most intersections are the most likely areas in which the vector $(x_1, \dots, x_k)$ lies. Attaining the values of the various xi allow a computational system to restore the u-function. This enables the system to predict the alternative item's total utilities in yet unmet choice-making scenarios. A novel extension of methods relates to the identification of inconsistent choice-making. In addressing the latter problem, we note that hyper-planes split the n-dimensional world into parts. We relate to every one of these parts (segments or rays in 1D, shapes, bounded or unbounded, in 2D), and count the number of half-spaces that contain it; this number reflects the probability that the actual (unknown) parameters are in it. Counting the number of half-spaces containing each segment allows us to consider multiple user profiles and considerations. This paves the way to the construction of more complex frameworks for understanding user choice as a multi-criteria decision making problem.**

***Index Terms*—preference detection, multi-attribute decision making, multi-item choice, recommender systems.**

I. Introduction

Item choice is often attribute-based (see: [1]-[9]). If we take movies as an example, we often choose movies because of their attributes – the actors, the plot, the genre, etc. And when a user chooses an item, we often believe that his selection is a function of the values of the attributes that the chosen item possesses, in relation to attributes of the same type that all other (un-chosen) items possess. Generally speaking, once we know *which* attributes a user prefers, and to what extent, we can infer user preferences in relation to other items in a designated domain. And once we have an understanding of user preferences we can predict their acceptance of and satisfaction with unfamiliar items. We propose a model with this purpose in mind.

II. Model

Mathematically, every option a in a specific item domain (e.g., movie recommendations, local service suggestions) can be thought of as a vector: $a = (a_1, \dots, a_k)$, in which the *i*-th component houses the value with respect to the *i*-th attribute; what we mean by this is that the subjective value of an attribute belonging to an item is a function of the conjunction of the objective attribute of that same item and the personal preference that a user has in relation to it.

So if, for example, we consider the domain of "movies," and the first attribute of a movie is "actors," $a_1$will be a number representing the extent to which the user likes the actors of the movie. Generally speaking, in our model different items in the same domain will have the same attributes, but with different values. Therefore a single decision-making scenario can be modeled by a matrix with *n* (the number of alternative items (or options)) rows and k (the number of attributes) columns. See Table I, below:

TABLE I. Matrix

a_{11}	a_{12}	...	a_{1k}
a_{21}	a_{22}	...	a_{2k}
⋮	⋮	⋱	⋮
a_{n1}	a_{n2}	...	a_{nk}

In this case the *i*-th row represents the *i*-th item in the option space.

When a user faces a choice-selection scenario, we assume that he computes a *total utility* of an item from the set of available items. This utility represents how appealing that item appears to him in view of its attributes. We believe this to be true even if the user is not aware of having considered the attributes. And hence we believe this matrix is reflective of the item selection scenario, whether or not the user is aware of the values of the varying item's attributes. Moreover, so that this model does not appear far removed from the way people actually make choices or accept recommendations, let us note that in our proposed method it is sufficient to know the total utility of an item for the purpose of evaluating and comparing options. It is because of this that we add a "utility column" to the matrix, as in Table II:

Manuscript received December 5, 2013; revised February 10, 2014.

doi: 10.12720/joace.3.1.46-50

TABLE II. MATRIX

a_{11}	a_{12}	...	a_{1k}	$\boldsymbol{u_1}$
a_{21}	a_{22}	...	a_{2k}	$\boldsymbol{u_2}$
⋮	⋮	⋱	⋮	⋮
a_{n1}	a_{n2}	...	a_{nk}	$\boldsymbol{u_n}$

The user assesses the items and, because he is a maximizer, he chooses the item with the greatest utility. In abstraction, a user's behavior can thus be modeled in the following way:

- Find details about the available items
- With respect to the details found, find the total utility of each item
- Choose the best item

In accordance with these three steps taken by the user in relation to the items, we model a computational system's behavior:

- Fill in (populate) the matrix
- Add another column to the matrix, and enter the total utility of every row
- Pick the row with the maximal last-column value

While steps #1 and #3 are relatively simple for a computational system, step #2 appears to be the real challenge – how should the system derive the appropriate function that aggregates all the attributes of an item into a single total utility?

It is to this problem that we presently turn. Let us begin by modeling the problem (see also: (Beliakov, Calvo, and James 2011)), and then proceed to offer a solution to it. First of all, it follows from the above noted stipulation that the single, total score, of an option is a function of its attributes:

$$u_i = u_i(a_{i1}, a_{i2}, \dots, a_{ik})$$

The next, seemingly trivial step is to remove the index of the function, so that the total utilities are computed in the same way for all items in the option space.

$$u_i = u(a_{i1}, a_{i2}, \dots, a_{ik})$$

This step represents the fact that when comparing several options the user relates unvaryingly to the (same) features of the different options. Hence a user may strongly prefer a higher score on the first attribute than on the second attribute, but he cannot have such preferences for the first item without having the same preferences for the second item too. For example, someone who, when it comes to movies, systematically prefers a good plot to good actors ought to do so for all the movies (items) he considers (note that this does not mean that a movie with a poor plot may not be chosen; it may, if for instance its actors are extremely good and thus outweigh the low score of the movie in terms of its *plot*).

In the system model that we propose, understanding the user's mechanism of rating (and then ranking) items is based on understanding the nature of this utility-function (u-function). Generally speaking, this u-function can be appraised by examining a user's previous choice-behavior. For instance, if in the past a user has systematically chosen movies by their director, it can be assumed that the user's u-function "prefers" - i.e. *gives greater weight to* - the attribute associated with the director of the movie.

For the sake of simplicity, we assume that *u* takes the form of a weighted sum:

$$u(a_1, a_2, \dots, a_k) = \sum_{n=1}^{k} x_n a_n$$

We choose the form of a weighted sum because it is general enough to allow biases and counter-relations among the different attributes. By changing the x_i's we can modify the weights and hence understand what factors or attributes are more important to a specific user, and to what extent. The relations between the x_i 's represent the relations between the importance that the user assigns to the varying attributes. Another advantage of this weighted form lies in its simplicity - in using it we reduce the problem (i.e., step #2 above) from estimating a function to estimating a "direction vector" $(x_1, \dots, x_k)$that represents the importance that the user assigns to each of the attributes – i.e., what factors are more important in the general estimation of an item ($high x_i$), and what factors don't matter ($low x_i$). Since we only compare utilities (we are not interested in the absolute value of the total utility of an item, but only in the relation between the different total utilities of different items), we may assume that $x_k = 1$, and then any other x_i is measured with respect to that $x_k = 1$.We will presently continue to demonstrate the method by which our system learns what is important to a specific user when it comes to choices between items.

Consider two items, A and B. Suppose a user picks A. Using the proposed model, this entails that$u_A > u_B$. And because we know the form of the u-function, this is translated to an inequality with scalar variables:

$$\sum_{n=1}^{k} x_n a_n > \sum_{n=1}^{k} x_n b_n$$

And this inequality defines a half-space:

$$\sum_{n=1}^{k} x_n (a_n - b_n) > 0$$

These half-spaces intersect and form closed shapes (in a k-dimensional world). And the closed shapes with the most intersections are the most likely areas in which the vector $(x_1, \dots, x_k)$lies. Attaining the values of the various $x_i$will allow the system to restore the u-function from above. Once that is done, the system can, in later choice-making scenarios, predict the total utilities of alternative choice-items, and can therefore give valuable recommendations, or alert the user when an inconsistent choice is made.

Let us demonstrate this approach in an item domain that has two attributes. As noted above, we can assume that one of the weights is 1 (otherwise we will just divide all weights by 1). The weights represent the coordinates of the direction vector $(x_1, \dots, x_k)$, which reflects the

extent to which a user wishes to maximize a particular attribute of an item in a choice situation. The general form of such a u-function is thus$u(a,b) = Aa + b$, where A is a constant that represents the importance (or weight) that the user assigns to the first attribute, with respect to the second attribute. Therefore in order to find the user's u-function, we have to estimate the constant A.

Consider this set of items, composed of attributes *a* and*b*, in which the item selected by a user is designated by the gray row in Table III:

TABLE III. TABLE SHOWING ITEM AND ATTRIBUTE VALUES

Item #	A	B
I	1	6
II	3	7
III	**5**	**4**
IV	6	1
V	5.5	2.75

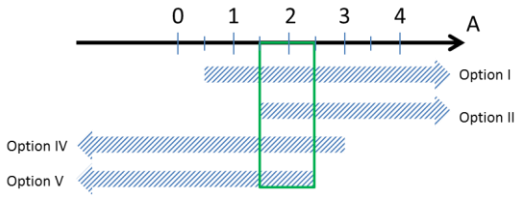
Now let us compare item I and item III (the selected, grey, item). We assume that III's utility is greater than I's utility (because III was chosen). So as a result we get the following inequality:

$$5A + 4 > A + 6$$

Which translates to $A > 0.5$. In the same way, comparing items II and III gives us $5A + 4 > 3A + 7$ and so $A > 1.5$. So far we have only acquired lower bounds on A (which is because we have only considered items with a smaller *a*-value than the selected option's *a*-value); comparing the selected item (III) to item IV gives us $5A + 4 > 6A + 1$ and so $A < 3$; comparing item III to item V results in $5A + 4 > 5.5A + 2.75$ and so $A < 2.5$.

Hence by considering all possible comparisons between the selected item and the alternative items, we get $1.5 < A < 2.5$, which represents the *range* within which all additional choices will be consistent with the choice-behavioral pattern that we have assessed. And therefore a good estimation for what A is within this domain is $A = 2$, the median between 1.5 and 2.5. Furthermore, tracking further decision scenarios can lead to tighter bounds and hence more accurate estimations. The method proposed is illustrated in the Table IV below.

TABLE IV. ILLUSTRATION OF METHOD



III. MULTI-ATTRIBUTE DOMAINS

The transition from a two attribute domain to a three or multiple attribute domain is a matter of logical progression. In a domain with three attributes the u-function takes the form: u(a,b,c) = Aa + Bb + c. And therefore understanding the user's preferences under this model is in effect akin to finding a point in the two-dimensional plane, representing the constants (A,B). Every comparison between two choices is translated mathematically to an inequality of the form: $\alpha A + \beta B > \gamma$; this defines a half-space, bounded by the equation $\alpha A + \beta B = \gamma$, which defines a line (in the (A,B) plane). Further comparisons between items result in more lines and more half-spaces; their intersection gives a two-dimensional shape, in which the actual A,B values are expected to be found.

IV. EXTENSIONS

A first, rather simple extension of this model relates to inconsistencies. While in the previous example all half-spaces (which were rays) intersected, there is no guarantee that things will always be this way. It is quite possible that as a result of a mistake made by a user, or rather a misjudgment, or just simple spontaneity, an inconsistent choice is made. Following the last example, we might get a choice that implies $A > 3$, which contradicts our previous inequality $A < 2.5$ (more complicated examples of higher dimensions can easily be thought of). In order to solve this issue, we note that these hyper-planes split the n-dimensional world into parts. We can consider every such part (segments or rays in 1D, shapes, bounded or unbounded, in 2D), and count the number of half-spaces that contain it; this number reflects the probability that the actual parameters are contained in it. Relating to the previous example we get the following (Table V):

TABLE V. NUMBER OF HALF SPACES PER SEGMENT

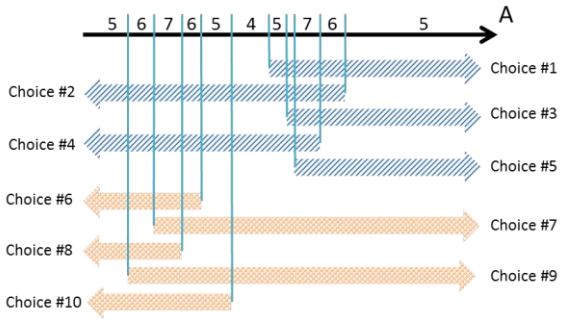
Segment	Number of half-spaces
$A < 0.5$	2
$0.5 < A < 1.5$	3
$1.5 < A < 2.5$	4
$2.5 < A < 3$	3
$A > 3$	2

According to this logic we see that the segment $1.5 < A < 2.5$ is chosen (highlighted in grey), because it is contained in the *most* half-spaces. If for some reason we infer from a user's choice-behavior that $A > 4$, despite the fact that it is inconsistent with previous choices, the segment $1.5 < A < 2.5$ will still be chosen.

Furthermore, this extension also enables us to track several kinds of preferences that a user may have in relation to the same, or similar, sets of items. Consider, for example, a user that has two "modes" – either he uses A very close to 2 (e.g., when he's in a good mood), or very close to 5 (e.g., when he is in a bad mood). Counting the number of half-spaces containing each segment allows us to consider multiple user profiles and considerations. And this paves the way for the construction of more complex frameworks for understanding user choice as a multi-criteria decision making problem. After all, users often make choices on the basis of multiple criteria, in addition to making choices on the basis of multiple attributes. And some criteria may lead users to modify their choice-making between different items and constituent attributes.

The figure below shows the relevant half-spaces for a set of ten independent choices made by a user. As can be seen, every half-space on the number axis defines a point and its edge, and so the ten choices define eleven segments. Above the number axis, directly above each of ten choices, the number of choices with which each segment is consistent is noted. Thus one can easily identify that the user whose choice-behavior is reflected in this illustration has two choice-*modes* – either his coefficient is in the intersection of choices 1-5 and 7, 9, or in the intersection of choices 2, 4 & 6-10, as in Table VI:

TABLE VI. ILLUSTRATION OF TWO CHOICE-MODES



This capacity to detect multi-mode choice behavior implies several things:

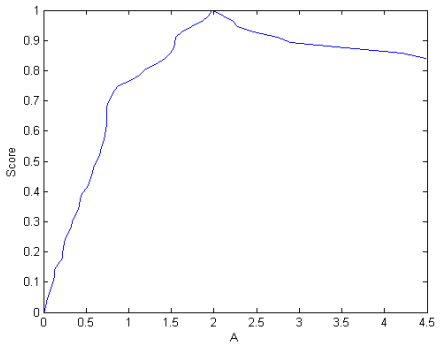
- It is extends the notion of *choice consistency*: it shows that choices can be consistent in some situations but not in others; for example, John may prefer sad movies on weekends but prefer comedies throughout the working-week.
- Learning (see also: (Rubens, Kaplan, and Sugiyama 2011)): a system can implement the method outlined above and then *learn* that a specific user has several choice "modes." The system can also learn to detect and predict each of these modes. Moreover, it can also learn to match specific items to each mode, thus increasing the predicted accuracy of item placements and suggestions.

V. EVALUATION

As a means of evaluation, we used a computer application that modeled the system presented here, with a randomly generated input. Every decision scenario contained ten options, each had two attributes (*a* and*b*) which were drawn randomly and uniformly between 0 and 1. We, as testers of the system, knew the parameters (*A*), and so could feed the system with those ten options and the selected option (after deciding a-priori the value of *A*). We then tested if the system could estimate the missing factor correctly.

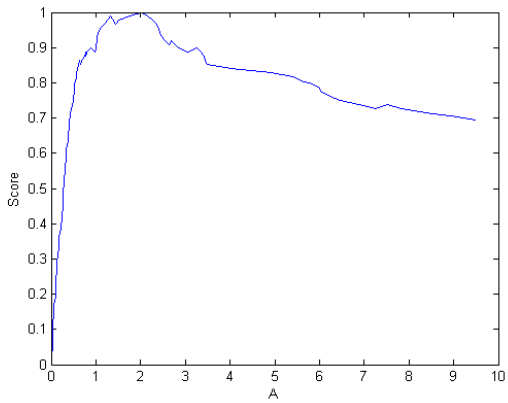
For a pre-decided value of A = 2, and for twenty randomly generated decision scenarios, the system came up with the following graph, whose x-axis represents the estimated value of A, and the y-axis represents the likelihood that this value of A was actually chosen (the y-axis is, in fact, the number of half-spaces that contain that point, normalized to lie in [0,1]). As can be seen in Table VII, the system's estimation for A is around 1.98, which is fairly close (with a relative error of about 1%) for a relatively small database.

TABLE VII. ESTIMATION OF A



We then proceeded to evaluate the system's reaction to inconsistencies (Table VIII) Assuming an error rate of 15% (in which a random option was selected), it took the system 50 scenarios to reach the same 1% relative error. It should be made clear that tolerating greater relative error (which may be true when we compare different utilities, as will be shown shortly) allows the system to use far less scenarios to learn the user.

TABLE VIII. ILLUSTRATION OF SYSTEM'S RESPONSE TO INCONSISTENCIES



As mentioned above, knowing the *exact* value of the parameter is not crucial in order to achieve high prediction rates. Repeating the last example, but having only 15 trials in the learning phase, the system estimated the value 2.138. While it is quite far from the actual value of 2, using that estimation the system guessed the user's choice correctly in 97915 out of 10000 randomly generated decision scenarios of ten choices each (almost 98% success rate, assuming the system had to guess the right one out of ten options; if we only have five options per scenario, the rate gets even higher).

For the detection of multiple profiles (or "modes"), a slight modification for the algorithm has been used.

Unfortunately, detection of multiple profiles (or "modes") are not as successful, as with randomly chosen options, it very likely that a selection made under one mode will affect the system's notion of the other modes,

or possible modes around the real mode. For instance, if one mode has A = 2, and the other A = 0.3, then the system may mistakenly consider any number between 0.3 and 2 to be a mode. This problem requires further research. One possible solution is for the system to, upon detection of a mode, ignore all decisions that are consistent with the detected mode, and run a second "mode analysis" on the remaining scenarios. Yet this can be problematic in the sense that it ignores many useful scenarios too (for example, A > 0.1 works for both modes and hence will be mistakenly deleted).

It is important to note that in the evaluation presented above, the system was fed with completely randomly generated options; we conjecture that the system may work significantly better if it specifically asks for the user to make decisions in tailored scenarios, fit for the system's needs. For example, if the system detects that the parameter A may lie between 1 and 3, it will not ask if A is greater or smaller than 4, but rather ask if it is greater or smaller than 2. An approach similar to the binary search algorithm may be used here.

VI. Conclusion

We presented an outline of a method that locates characteristics that reflect the relative weight that a user gives to attributes belonging to an item in making utility based decisions with regard to that item. The method we propose supports decision making in multi-choice scenarios. We showed how our method can detect preference changes of a single user and users with multiple preferences. The method can also serve de-biasing purposes, such as monitoring for consistent and coherent choice patterns.

References

[1] J. Basilico and T. Hofmann, "Unifying collaborative and content-based filtering," in *Proc. Twenty-First International Conference on Machine Learning*, New York, USA: ACM, 2004.

[2] G. Beliakov, T. Calvo, and S. James. "Aggregation of preferences in recommender systems," in *Recommender Systems Handbook*, January 1, 2011, pp. 705–734.

[3] M. Deshpande and G. Karypis, "Item-based top-n recommendation algorithms," *ACM Trans. Inf. Syst.*, vol. 22, no. 1, pp. 143–177, January 2004.

[4] N. Rubens, D. Kaplan, and M. Sugiyama, "Active learning in recommender systems," in *Recommender Systems Handbook*, F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, Eds., Boston, MA: Springer US, 2011, pp. 735–767.

[5] M. Salehi, I. N. Kamalabadi, and M. B. G. Ghoushchi, "Personalized recommendation of learning material using sequential pattern mining and attribute based collaborative filtering," *Education and Information Technologies*, December 29, 2012.

[6] M. Salehi, M. Pourzaferani, and S. A. Razavi, "Hybrid attribute-based recommender system for learning material using genetic algorithm and a multidimensional information model," *Egyptian Informatics Journal*, vol. 14, no. 1, pp. 67–78, March 2013.

[7] S. Sinha, K. S. Rashmi, and R. Sinha, *Beyond Algorithms: An HCI Perspective on Recommender Systems*, 2001.

[8] R. Srikant and R. Agrawal, "Mining sequential patterns: Generalizations and performance improvements," in *Research Report RJ 9994, IBM Almaden Research*, 1995.

[9] M. Stritt, K. H. L. Tso, and L. S. Thieme, "Attribute aware anonymous recommender systems," in *Advances in Data Analysis*, R. Decker and H.-J. Lenz, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, January 15, 2014, pp. 497–504.

Dr. Amir Konigsberg is a senior scientist at General Motors Research and Development Labs, where he works on advanced technologies in the field of artificial intelligence and human machine interaction. Amir gained his PhD from the Center for the Study of Rationality and Interactive Decision Theory at the Hebrew University in Jerusalem and the Psychology Department at Princeton University.

Ron Asherov is an intern research at General Motors R&D.

