# Preference Function Learning over Numeric and Multi-valued Categorical Attributes

**Lucas Marin**[1], **Antonio Moreno** and **David Isern**

**Abstract.** One of the most challenging goals of recommender systems is to infer the preferences of users through the observation of their actions. Those preferences are essential to obtain a satisfactory accuracy in the recommendations. Preference learning is especially difficult when attributes of different kinds (numeric or linguistic) intervene in the problem, and even more when they take multiple possible values. This paper presents an approach to learn user preferences over numeric and multi-valued linguistic attributes through the analysis of the user selections. The learning algorithm has been tested with real data on restaurants, showing a very good performance.

## 1 Introduction

Nowadays it is practically unconceivable to select our summer holiday destination or to choose which film to see in the cinema this weekend without consulting specialized sources of information in which, in some way or another, our preferences can be specified to aid the system to recommend us the best choices. That is because we live in an era where there are so many data easily available that it is impossible to manually filter every piece of information and evaluate it accurately. *Recommender Systems* (RS) have been designed to do this time-consuming task for us and, by feeding them with information about our interests, they are capable enough to tell us the best alternatives for us in a personalized way.

A RS stores the preferences of the user about the values of some criteria and uses this information to rate and sort a corpus of alternatives. The management of the preferences, the accuracy of the recommendations, and how these interests evolve over time are three of the most challenging tasks of these type of systems [8].

Concerning the first goal, RSs may obtain feedback from a user implicitly, explicitly or combining both approaches. This paper discusses an unsupervised way to infer the user interests, which observes the user interaction and does not require any explicit information from him [4].

The criteria used to describe the alternatives may have different natures. Some works propose the use of ontologies to represent concepts within a hierarchy [2,9]. Other researchers use fuzzy logic techniques to deal with linguistic criteria [1,10], and many approaches only consider numerical criteria [5]. This paper considers the use of linguistic and numerical data, which permit a high degree of expressivity and can be applied in a wide range of domains.

[1] Department of Computer Science and Mathematics, Universitat Rovira i Virgili, Tarragona, Catalonia (Spain). Email addresses: lucas.marin@urv.cat, antonio.moreno@urv.cat, david.isern@urv.cat.

The basic idea is to use the preferences to sort a set of alternatives, show this ordered list to the user, and observe his final selection. With this information, the preference learning algorithm is able to modify the user profile so that it captures better the user preferences and the next recommendation is more accurate.

The rest of the paper is organized as follows. Section 2 includes a brief explanation of the related work the authors conducted in the area of preference learning over linguistic and numeric attributes, explaining how the interests of users over certain attributes or criteria are managed and learned. Section 3 explains a new approach to manage categorical attributes when they can take multiple linguistic values in a single alternative. Section 4 describes how a more expressive function which defines the behaviour of the preference over numeric attributes can be automatically learned. In Section 5 the case study where our approach has been tested (restaurant recommendation) is explained, describing the data set used and the results obtained. Finally, Section 6 gives the main conclusions of the paper and identifies some lines of future research.

## 2 Preference learning over categorical and numerical attributes

When we face a decision problem in which we require the aid of a RS to help us make a choice, all of the possible alternatives to said problem are defined, in most of the cases, by the same attributes. In this work we focus only on categorical and numeric attributes. The following subsections explain how preferences over the two different kinds of attributes are expressed, how alternatives are evaluated and ranked, and how the user interests are learned and adapted from his selections.

### 2.1 Attributes and management of preferences

In a recent work ([7]) we proposed to represent the level of interest over categorical attributes by using a linguistic scale in which preference labels are defined as fuzzy sets representing values of preference such as “Very Low”, “Low”, “Medium”, “High” or “Very High” (see Figure 1).

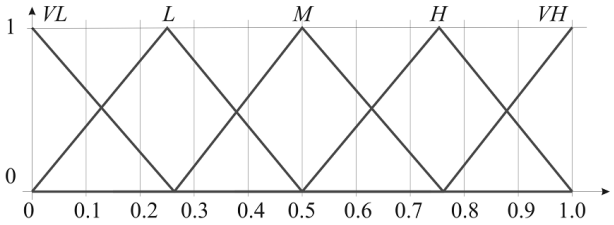


**Figure 1**. Example of a linguistic preference set

For the case of numeric attributes, we assumed that each user has a preference function for each attribute. This function has a triangular shape (see ) and is defined as

$$p_a(x) = 1 - \frac{|x - v_{pref}|}{\Delta} \quad (1)$$

where $p_a(x)$ is the preference of the value $x$ of the attribute $a$, and $\Delta$ is the width of the function, which we considered to be 10% of the attribute domain.

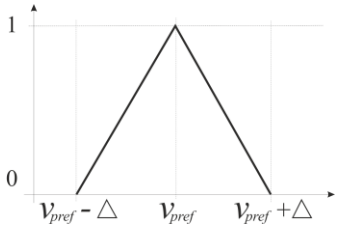


**Figure 2**. Basic numeric preference function

## 2.2 Alternatives evaluation

When evaluating an alternative, the objective is to aggregate all of the values of all of the attributes into a single value. Since we have two kinds of attributes, a conversion to the same domain is made. In our approach, we chose to translate the numerical preferences to linguistic ones. The translation is done by, first, calculating the value of preference of a certain numeric attribute value by using Eq. (1). Then that value is mapped to the fuzzy linguistic labels domain and matched with the label with a higher value in that point.

When all the attributes have been assigned a value of preference using the same fuzzy linguistic scale, all the terms are aggregated using the ULOWA aggregation operator [3]. The final result of this aggregation is the value of preference assigned to the whole alternative, used to rank the alternatives.

## 2.3 Preference learning

When the ranked alternatives are presented to the user, two things can happen: (a) the user selects the first ranked alternative or (b) the user selects any other alternative. The first case means that the recommendation process has worked accurately, since the system gave the first place to the selected alternative. However, in the second case, there were other alternatives (which we call *over ranked*) that were considered by the system as better than the one the user finally selected. Thus, that is probably indicating that the information that we have in the user profile is not accurate enough and should be modified. In a nut shell, the main intuition behind the user profile change algorithm is that we should increase the preference on the attribute values present in the selected alternative and decrease the preference on the attribute values appearing in the over ranked alternatives.

The information required to infer this reasoning is extracted from what is called "relevance feedback". In this case, it consists in the over ranked alternatives and the selected one. Numerical and categorical attributes are managed in different ways, as described in the following subsections.

### *2.3.1 Linguistic preference adaptation*

The main idea is to find attribute values repeated among the over ranked alternatives that do not appear on the selection, which will be the candidates for having his preference decreased. Similarly, the preference of the attribute values that appear on the selection and do not appear often on the over ranked alternatives is likely to be increased. The interested reader may find a more detailed explanation of the process of adaptation of linguistic preferences in [7].

The profile adaptation is conducted by two processes. The first one—called *on-line* adaptation—is executed every time the user asks the system for a recommendation, and it evaluates the information that can be extracted from the current ranked set of alternatives. The main goals of this stage are to *decrease* the preference of the attribute values that are causing non-desired alternatives to be given high scores and to *increase* the preference of the attribute values that are important for the user but are not well judged on the basis of the current user profile. For each recommendation made by the system, two sources of information are evaluated: the selected alternative, which is the choice made by the user, and the alternatives that were ranked above it. Values extracted from the over-ranked alternatives haver their level of preference *decreased* whereas the ones extracted from the user's final selection that do not appear in the set of over-ranked alternatives have their preference *increased*.

The second one—called *off-line* adaptation—is triggered after the recommender system has been used a certain number of times. It considers the information given by the history of the previous rankings of alternatives and the selections made by the user in each case, but considers that information separately. When the system faces cases in which the number of over ranked alternatives is not large enough for reliable characteristics to be extracted, it stores the small number of over ranked alternatives in a temporary buffer. After several iterations in which the number of over ranked alternatives has been insufficient for evaluation, the system will have recorded enough alternatives to start evaluating them. When there are enough saved over-ranked alternatives, the values in their attributes will be analysed and their preference *decreased*. Moreover, user selections are also stored, and after a certain number of choices have been made, they are evaluated with the objective to increase the preference of the most repeated attribute values, since their repeated selection indicates that the user is really interested in them.

### 2.3.2 *Numeric preference learning*

The numeric adaptation of the user profile presented in [6] is inspired by Coulomb's Law: "*the magnitude of the electrostatics force of interaction between two point charges is directly proportional to the scalar multiplication of the magnitudes of charges and inversely proportional to the square of the distances between them*". The main idea is to consider the value stored in the profile (current preference) as a charge with the same polarity as the values of the same criterion on the over ranked alternatives, and with opposite polarity to the value of that criterion in the selected alternative. Thus, the value of the profile is pushed away by the values in the over ranked alternatives and pulled back by the value in the selected alternative. Two stages have been considered in the adaptation algorithm. The first one, called on-line adaptation process, is performed each time the user asks for a recommendation. The other stage, called off-line process, is performed after a certain amount of interactions with the user.

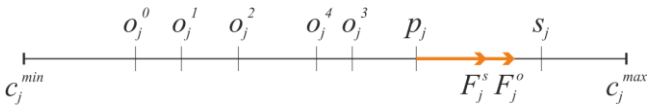


**Figure 3**. Attraction and repulsion forces

For the on-line stage, the information available in each iteration is the user selection and the set of over ranked alternatives. In order to calculate the change of the value of preference in the user profile for each criterion it is necessary to study the attraction force done by the selected alternative and the repulsion forces done by the over ranked ones in each criterion, as represented in the example in Figure 3, in which the *j*-th value of the five over ranked alternatives $o^0$, $o^1$, $o^2$, $o^3$, and $o^4$ causes a repulsion force $F^o_j$, and the value for the same criterion of the selected alternative, $s_j$, causes an attraction force $F^s_j$. Both forces are applied on the *j*-th value of the profile, $P_j$.

The attraction force $F^s$ done by the selected alternative for each attribute *j* is defined as

$$F^s(P,s,j,\alpha)=\begin{cases}\Delta_j\left(\dfrac{1}{|s_j-P_j|^{\alpha}}\cdot\dfrac{s_j-P_j}{|s_j-P_j|}\right) & if\ s_j\neq P_j\\ 0 & if\ s_j=P_j\end{cases} \quad (2)$$

In this equation, $\Delta_j$ is the range of the criterion *j*, $s_j$ is the value of the criterion *j* in the selected alternative and $P_j$ is the value of the same criterion in the stored profile *P*. The parameter α adjusts the strength of the force in order to have a balanced adaptation process. The repulsion force exerted by the over ranked alternatives for each criterion *j* is defined as a generalization of Eq.(2) as follows:

$$F^o\left(P,\{o^1,...,o^{no}\},j,\alpha\right)=\sum_{i=1}^{no}\frac{1}{|P_j-o_j^i|^{\alpha}}\cdot\frac{P_j-o_j^i}{|P_j-o_j^i|} \quad (3)$$

Finally, both forces are summed up and the resulting force is calculated.

The techniques designed for the on-line stage fail at detecting user trends over time since they only have information of a single selection. The off-line adaptation process gathers information from several user interactions. This technique allows considering changes in the profile that have a higher reliability than those proposed by the on-line adaptation process, because they are supported by a larger set of data.

The off-line adaptation process can be triggered in two ways: the first one evaluates the user choices, while the second one analyses the over ranked alternatives discarded by the user in several iterations. The possibility of running the off-line process (in any of its two possible forms) is checked after each recommendation. In the first case, the system has collected some alternatives selected by the user in several recommendation steps, and it calculates the attraction forces ($F'_s$) exerted by each of the stored selected alternatives over the values stored in the profile, using an adaptation of Eq. (2), that has as inputs the profile *P*, the past selections $\{s^1,\ldots,s^{rs}\}$, the criterion to evaluate *j*, and the strength-adjusting parameter α.:

$$F_s'\left(P,\{s^1,...,s^{rs}\},j,\alpha\right)=\sum_{i=1}^{rs}\frac{1}{|s_j^i-P_j|^{\alpha}}\cdot\frac{s_j^i-P_j}{|s_j^i-P_j|} \quad (4)$$

The second kind of off-line adaptation process evaluates the set of over ranked alternatives that have been collected through several iterations and which were not used in the on-line adaptation process (because it did not have enough over ranked alternatives in a single iteration). When the stored over ranked alternatives reach a certain number, the off-line adaptation process calculates the repulsion forces over the profile values exerted by those alternatives ($F^o$), which are calculated using Eq.(3).

## 3 Multi-valued linguistic attributes

As explained in Section 2, in our previous work we considered categorical attributes that could take only one linguistic value (For example, a city could have a single value in the "Climate" attribute). However, there are cases in which it is interesting to consider multiple values. One example of that situation could be the attribute "Types of food" in a restaurant: a restaurant can have the values {"Asian", "Seafood", "Vegetarian"} while another can have only "Italian food".

Extending our model so that it can manage lists of categorical values implies addressing two issues: how to represent and calculate the user preferences over the attribute taking into account all of the values and how to adapt dynamically those preferences.

### 3.1 Preference value on multi-valued attributes

When there is an attribute with multiple values a procedure should be defined to decide which single linguistic preference represents better the whole set of linguistic values.

Going back to the restaurant example, if a user has a "High" preference over "Asian food" restaurants and a "Low" preference over "Rice dishes", we can argue that the preference we could assign to the "Type of food" attribute in a restaurant with both values should be "Medium" (an average of the two kinds). If another restaurant only offers "Asian food" then its preference should be "High", so this restaurant would have a higher ranking than the first one. The rationale of this procedure is that it seems more adequate to reward the alternatives that are more focused in the aspects the user really likes. This example represents an "average" preference aggregation policy, however, other policies can also be considered depending on the attribute definition.

## 3.2 Preference learning on multi-valued categorical attributes

The linguistic algorithm used to adapt categorical preferences explained in Section 2 needs some improvements to be able to manage lists of values. When single-valued attributes were considered, the user selection pointed directly towards the value the user liked for that attribute. Now, however, we cannot be sure which one/s of the values listed in the attribute is/are the one/s of interest for the user. That is the reason why it has been necessary to design a "*relevance function*" which indicates how relevant is a value found among the over ranked alternatives or in the selected alternative. Relevance is measured in a [0,1] scale, with 1 meaning maximum relevance. To calculate how relevant a term *t* of the attribute *j* is among the over ranked alternatives we use this expression (the relevance value is 0 if it does not appear in the over ranked alternatives):

$$R_j^o(t) = \frac{1}{no}\sum_{i=1}^{nt}\frac{1}{nv_j^i} \tag{5}$$

Here, *no* represents the number of over ranked alternatives, *nt* the number of over ranked alternatives where *t* appears, and $nv^i_j$ the number of values that appear for the attribute *j* in the alternative *i*. In this equation we consider that every linguistic term that appears in the over ranked alternatives has a relevance which is inversely proportional to the number of other values for the same attribute that appear among the entire set of over ranked alternatives.

To calculate the relevance of a term in the selection we use:

$$R_j^s(t) = \frac{1}{2}\left(\frac{1}{nv_j} + \frac{nl}{tv}\right) \tag{6}$$

Here $nv_j$ represents the number of values that appear for the attribute *j* in the selection, *nl* the total number of linguistic attributes, and *tv* the total number of linguistic values that appear in the selection. The relevance of a term in the selection is the mean between the importance of the term among the values that appear with it in the same attribute and the importance of each linguistic term that appears in the selection compared with the number of linguistic attributes.

Finally, after calculating both partial relevancies for all the terms, the overall relevance $R_j(t)$ is calculated as:

$$R_j(t) = R_j^s(t) - R_j^o(t) \tag{7}$$

In conclusion, considering a threshold $\gamma$ to avoid making changes in the profile with low relevance, it can be deduced that:

- If $R_j(t)>\gamma$, the preference over term *t* for the attribute *j* needs to be increased (moved to the next term).
- If $R_j(t)<\gamma$, the preference over term *t* for the attribute *j* needs to be decreased (moved to the previous term).

# 4 Learning preference functions for numeric attributes

Although the numeric preference learning approach described in Section 2 provided an adequate way of learning the ideal value of preference over a numeric attribute, it was unable to learn all of the parameters that model the preference function such as the slope or the width, which were fixed. The new learning method presented in this section relies on historic data about the user selections to approximate the preference function of the numeric attributes to the most adequate one. With this approach, we have a new definition of the function of preference which now has 5 parameters (left and right slope, left and right width, and value of preference) instead of just the value of preference:

$$p_a(x)\begin{cases} 1-\dfrac{\left|x-v_{pref}\right|^{m_l}}{\Delta_l} & if\ (x<v_{pref}) \\ 1 & if\ (x=v_{pref}) \\ 1-\dfrac{\left|x-v_{pref}\right|^{m_r}}{\Delta_r} & if\ (x>v_{pref}) \end{cases} \tag{8}$$

In this expression $p_a(x)$ is the preference of the value *x* of the attribute *a*, $m_l$ and $m_r$ are the function slope values (for the left and right sides of the triangle, respectively) and $\Delta_l$ and $\Delta_r$ are the parameters which define the width of the function (also for the left and right sides of the triangle, respectively). An example of graphical representation of a preference function can be seen in Figure 4, where the left slope is a value under 1, the right slope is a value over 1, and the left width is greater than the right one.

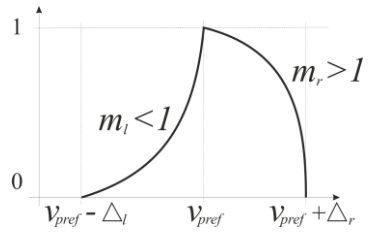


**Figure 4**. Numeric preference function with 5 parameters

The whole process of adapting the numeric preference function is depicted in Figure 5.

**function** PREF-FUNC-ADAPTATION(
  $V(v_0,\ldots,v_n)$, //historic of values of past selections
  $v_{pref}$, //value of maximum preference
  $v_{min}$, //minimum numeric value
  $v_{max}$, //maximum numeric value
  *ti*, //trust interval
  *s* //probability distribution sampling)
**begin**
  *B*=getBestValues(*V*, $v_{pref}$, *ti*);
  *PD*=calculateProbabilityDistribution(*B*, $v_{min}$, $v_{min}$, *s*);
  Δ{left,right}=calculateDelta(*PD*);
  *m*{left,right}=calculateBestSlope(*PD*, $v_{pref}$, Δ);
  *PreferenceFunction*=(Δ, *m*, $v_{pref}$);
  **return** *PreferenceFunction*;
**end**;

**Figure 5.** Preference function learning algorithm

The first step consists in obtaining the more reliable values from the historic set of selections. This is done by extracting a percentage of the values closer to the value of preference (trust interval), normally of 90%. With that we avoid considering outlier values. Then a probability distribution function, represented with a histogram, is calculated with those best values. The sample or

discretization step is a parameter, normally around 1% of the domain range. Delta values are then calculated by observing the width of the probability distribution. For example, if the first value different to 0 in the histogram is 3 and the last is 56, and the value of higher preference ($v_{pref}$) is 34, $\Delta_l$ would be 31 and $\Delta_r$ would be 22. Afterwards, the algorithm generates preference functions with different combinations of values for the slope values ($m$) (in the range from 0 to 4 in steps of 0.2), and compares the distance between each preference function and the probability distribution. The function with the lower distance shows the chosen slope. Finally, the new preference function is built with the new delta and slope values.

# 5 Case study: restaurant recommendation

In order to test our new approach to multi-valued attribute evaluation and numeric preference function learning, we have used data of the restaurants in Barcelona to implement a RS with the ability to learn the users' interests from their selections. In the first part of this section a description of the data is given. Then, a basic explanation of the whole recommender and learning algorithm is given, as well as the preferences setup. Finally, the results of the evaluation are provided.

## 5.1 Barcelona restaurants data

The data used in this problem has been collected from the *BcnRestaurantes* web page[2]. The data set contains information about 3000 restaurants of Barcelona evaluated by 5 attributes: 3 categorical ("Type of food"- 15 values, "Atmosphere"- 14 values, "Special characteristics" – 12 values) and 2 numerical ("Average price", "Distance to city center"). One example of register in the data file is "*Fonda España; National, Season cuisine, Traditional; Classic, For families; Round tables, In a hotel, With video; 45; 0.979*", being "Fonda España" the restaurant name, "National", "Season cuisine" and "Traditional" the types of food served, "Classic" and "For families" the restaurant atmosphere, "Round tables" and "In a hotel" other important restaurant characteristics, 45€ the average menu price, and 0.979 km the distance to the city centre.

## 5.2 Recommendation and adaptation

The set of 3000 restaurants has been divided in blocks of 15 alternatives that are ranked independently, which gives out a total of 200 different recommendations. An ideal profile was manually defined and three initial profiles were created randomly. The goal is to learn the ideal profile starting from these three different points. In this evaluation the preferences over the categorical attributes are represented with a linguistic label term set of 7 values, which are "Very Low", "Low", "Almost Low", "Medium", "Almost High", "High" and "Very High".

The whole process (for each of the three profiles, repeated 200 times) consists in:

1. Ranking a set of 15 alternatives according the current (initially random) profile.
2. Simulate the selection of the user by choosing the alternative that fits better the ideal profile.
3. Extract relevance feedback from the selection (over ranked alternatives and the selection itself).
4. Decide which changes need to be made to the current profile and apply them.

Some information about the whole process is stored after each iteration, including the position of the selected alternative, the distance between the ideal and current profiles, and the preferences over linguistic and numeric values.

## 5.3 Results evaluation

In order to evaluate the results of the new learning techniques, a distance function has been defined to calculate how different the profile we are learning is to an ideal profile which represents the exact preferences of the user. The first step is to calculate the distance for each attribute, taking into account if it is numeric or categorical. The distance between numeric attributes is calculated as

$$d(n,c,i) = 1 - p_n^c(v_{pref\,n}^{\,i}) \tag{9}$$

where $n$ is the numerical attribute, $c$ is the current profile (the one being learned), $i$ is the ideal profile, and $p_n^c(v_{pref\,n}^{\,i})$ is the value of preference of the $v_{pref}$ value for the attribute $n$ in $i$ using the preference function of the same attribute in the profile $c$. A distance 0 means that the $v_{pref}$ values in both profiles are equal.

The equation to calculate the distance between categorical attributes is

$$d(l,c,i) = \frac{1}{card(l)} \sum_{k=1}^{card(l)} \frac{\left| CoG(p_l^c(v_k)) - CoG(p_l^i(v_k)) \right|}{\left| CoG(s_{\min}) - CoG(s_{\max}) \right|} \tag{10}$$

where $l$ is the categorical attribute, *card(l)* is the cardinality of the attribute $l$ (i.e., the number of different linguistic values it can take), $CoG(p_l^c(v_k))$ and $CoG(p_l^i(v_k))$ are the x-coordinate of the centres of gravity of the fuzzy linguistic labels associated to the value of preference of $v_k$ in the profiles $c$ and $i$, respectively, and $CoG(s_{\min})$ and $CoG(s_{\max})$ are the centres of gravity of the minimum and maximum labels of the domain, respectively. Finally, the distance between two profiles is calculated as

$$D(c,i) = \frac{1}{na} \sum_{k=1}^{na} d(k,c,i) \tag{11}$$

where *na* is the total number of attributes.

During the three tests (one for each initial random profile) the distance between the adapting and the ideal profile has been calculated in each iteration. Figure 6 (continuous line) shows the average of the three distances. It can be seen that the initial average distance between the ideal and the adapting profiles is around 0.59. After 200 iterations it reaches a distance around 0.1. Although 200 iterations may seem a large number, it can also be observed that with only 50 iterations a very acceptable result of 0.2 is obtained.

To see to what extent the new approach to learn the numeric preference function explained in Section 4 has improved the result of our previous work (commented in Section 2), Figure 6 also compares the results with and without (dashed line) that functionality. It can be seen how the improvement has been noticeable (distance improvement of about 0.07).

[2] http://www.bcnrestaurantes.com. Last access May 30th, 2012.

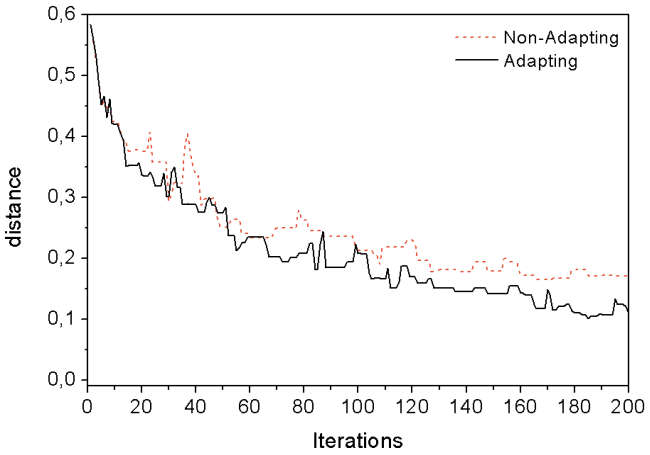


**Figure 6.** Average distance between current and ideal profile

To wrap up the results evaluation, Figure 7 shows in what position the user selection is being ranked by the RS on each of the iterations in the first test (the three give similar results). This figure shows the results in a more intuitive way. Notice that the system is accurate if the selected alternative is in the first positions of the 15-items list in each iteration. Many factors can interfere in the process and make the learning of the exact ideal profile a very hard task, but if the user selection appears in the first positions, we can consider that the learning process is working properly. As it can be observed in Fig.7, after about 50 iterations, the selected alternative is among the first three ones in 95% of the cases (and the first one in around 70% of the cases).

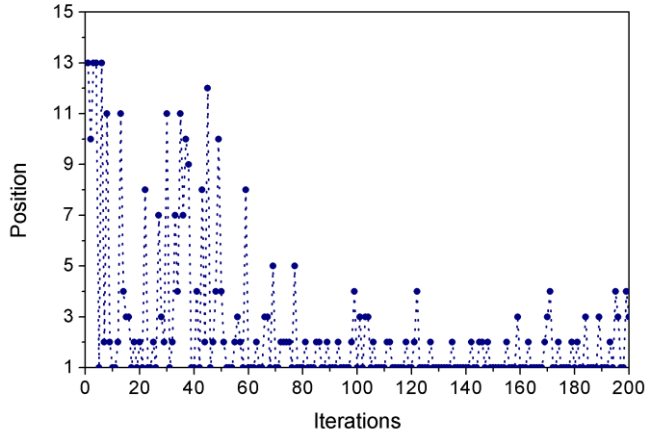


**Figure 7.** Position of the selected alternative in each iteration (test 1)

## 6 Conclusions and future work

Two main contributions with respect to our previous work have been presented in this paper. The first one consists in managing multi-valued categorical attributes in the alternatives of a RS, allowing. more expressivity in their representation. The system considers a single preference for each possible value and aggregates them to find out the preference over the whole attribute. The consideration of multi-valued attributes is mandatory when working with alternatives such as the ones presented in this paper (e.g. "Type(s) of Food" in a restaurant alternative).

The second contribution, which is learning the numeric preference function, allows shaping a more expressive and personalised representation of user preferences over each numeric attribute, defining a preference function with 5 parameters. This additional expressivity helped to improve the profile learning process by reducing the learning error around 7%.

As a future work, two interesting lines can be considered. As pointed out in Section 3, an aggregation policy can be considered in the aggregation of the preferences in a single attribute, other than the use of the common "average" policy. Research can be made in this area in order to learn the aggregation policy that fits more the user interests. Another interesting line to consider is to incorporate information about the numeric preference function in the distance measure used to evaluate the algorithm since, currently, just the value of preference is being considered.

## ACKNOWLEDGEMENTS

This work has been supported by the Universitat Rovira i Virgili (a pre-doctoral grant of L. Marin) and the Spanish Ministry of Science and Innovation (DAMASK project, Data mining algorithms with semantic knowledge, TIN2009-11005) and the Spanish Government (Plan E, Spanish Economy and Employment Stimulation Plan).

## REFERENCES

[1] G. Castellano, C. Castiello, D. Dell'Agnello, A. M. Fanelli, C. Mencar, M. A. Torsello, Learning Fuzzy User Profiles for Resource Recommendation, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 18 (4) (2010), 389-410.

[2] I. Garcia, L. Sebastia, E. Onaindia, On the design of individual and group recommender systems for tourism, Expert Systems with Applications 38 (6) (2011), 7683-7692.

[3] D. Isern, L. Marin, A. Valls, A. Moreno, The Unbalanced Linguistic Ordered Weighted Averaging Operator, in: IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 2010, IEEE Computer Society, Barcelona, Catalonia, 2010, 3063-3070.

[4] G. Jawaheer, M. Szomszor, P. Kostkova, Comparison of Implicit and Explicit Feedback from an Online Music Recommendation Service, in: 1st International Workshop on Information Heterogeneity and Fusion in Recommender Systems, ACM Press, Chicago, US, 2010, 47-51.

[5] T. Joachims, F. Radlinski, Search Engines that Learn from Implicit Feedback, Computer 40 (8) (2007), 34-40.

[6] L. Marin, D. Isern, A. Moreno, Dynamic adaptation of numerical attributes in a user profile, International Journal of Innovative Computing Information and Control (2012).

[7] L. Marin, D. Isern, A. Moreno, A. Valls, On-line dynamic adaptation of fuzzy preferences, Information Sciences doi: 10.1016/j.ins.2011.10.008 (2012).

[8] M. Montaner, B. López, J. L. de La Rosa, A taxonomy of recommender agents on the internet, Artificial Intelligence Review 19 (4) (2003), 285-330.

[9] A. Moreno, A. Valls, D. Isern, L. Marin, J. Borràs, SigTur/E-Destination: Ontology-based personalized recommendation of Tourism and Leisure Activities, Engineering Applications of Artificial Intelligence doi:10.1016/j.engappai.2012.02.014 (2012).

[10] C. Porcel, A. G. López-Herrera, E. Herrera-Viedma, A recommender system for research resources based on fuzzy linguistic modeling, Expert Systems with Applications 36 (3, Part 1) (2009), 5173-5183.