How to learn fuzzy user preferences with variable objectives

Alan Eckhardt[1],[2] Peter Vojtáš[1],[2]

1.Department of Software Engineering, Charles University
Prague, Czech Republic
2.Institute of Computer Science, Czech Academy of Science
Prague, Czech Republic
Email: {eckhardt , vojtas}@ksi.mff.cuni.cz

Abstract— *This paper studies a possibility to learn a complex user preference model, based on CP-nets, from user ratings. This work is motivated by the need of user modelling in decision making support, for example in e-commerce. We extend our user model based on fuzzy logic to capture variation of preference objectives. The proposed method 2CP-regression is described and tested. 2CP-regression uses CP-nets idea behind and can be considered as learning of a simple CP-net from user ratings.*

Keywords— user preferences, data mining, ceteris paribus

1 Introduction

The problem studied in this paper is based on the idea of helping user in making decisions. The main motivation lies in e-commerce, where user might benefit a recommendation of objects that might interest her without needing to process all objects manually. This recommendation should be rather automatic and transparent, because user typically does not want to fill in any complicated forms or invest much time in search.

When a user is buying a notebook, for example, she is considering the attributes such as price, manufacturer, size of RAM, harddisk, display. In traditional e-shop environment, it is possible to make some restrictions on these attributes and lower the number of notebooks the user has to process manually this way. But these restrictions are inflexible - when the user selects price from 1000$ to 2000$, a notebook with price 999$ would not appear in the result set. This is the reason for a flexible search, based on fuzzy user preferences.

In this paper we propose a method for handling more complicated user preferences. In [1] a phenomenon called ceteris paribus was described. Preference ceteris paribus means preference "all else being equal". When a user is comparing two notebooks, we can say that the size of RAM of 2GB is preferred to 1GB ceteris paribus, meaning that we assume that the remaining attributes of these notebooks are the same and their values are not important. A user model based on this idea was later proposed in [2] - a CP-network is a graph that captures preferences ceteris paribus or, in different view, it represents attributes, on which the preferences over the attribute RAM depend. And yet, as far as we know, there are no study of learning of CP-nets from user ratings. Our paper is a first subtle contribution to this field. We work with our user model based on fuzzy sets rather than CP-nets, still the proposal of learning the relations between attributes is strongly related to CP-nets.

In Section 2 our user model is described. Related work is studied in Section 3. In Section 4 is described in depth the proposed method for learning the relation between attributes. Then this approach is tested in Section 5 and we end with conclusion and future work in Section 6.

2 Two step user model

2.1 Notation

We will work with a set of objects X. Overall rating of an object is a fuzzy subset of X, i.e. a function $r(o) : X \to [0,1]$, where 0 means least preferred and 1 means most preferred. Every object has attributes $A_1, ..., A_N$ with domains $D_{A_1}, ..., D_{A_N}$, with one special attribute that serves as identification of object (ID). Let $X \subseteq \prod_{i=1}^{N} D_{A_i}$. We will use $X(a)$ when denoting a set of objects that have the attribute value a.

User model, in our view, is a method for representing user decision when considering user's preference of an object $o \in X$. Our user model consists of two steps. At the first step, attributes of o are normalised using fuzzy sets $f_i : D_{A_i} \to [0,1]$ (and here again, 0 is least preferred and 1 is most preferred value), so that the transformed object is in $[0,1]^N$. These fuzzy sets are also called objectives or preferences over attributes. At the second step, preference over attributes are aggregated into the rating of the whole object via a fuzzy aggregation function $@ : [0,1]^N \to [0,1]$. Aggregation function is also often called utility function.

We concentrate on learning of user models. User is expected to rate a small sample $S \subseteq X$ of objects ($r : S \to \{1,2,3,4,5\}$), where $|S| \ll |X|$. The size of training set is expected to be very small. We assume that these ratings were created by the user using some fuzzy sets $f_1, ..., f_N$ and an aggregation function @. Of course the user does not compute the result of a function, but her decision corresponds to this two step process.

Our model consist of $f_1, ..., f_N$ and of @ so we expect to construct these functions from user ratings of S. The learnt functions will be denoted as $\widehat{f_1}, ..., \widehat{f_N}$ and $\widehat{@}$. Learning of aggregation function is described in Section 2.2. This paper concentrates on learning of local preferences, which is described in Section 4.

2.2 Aggregation function learning

Aggregation function serves to combine preferences over attributes into a single rating that represents the preference of the object as a whole. There are many ways to aggregate local preferences, we are currently using a weighted average with

ISBN: 978-989-95079-6-8

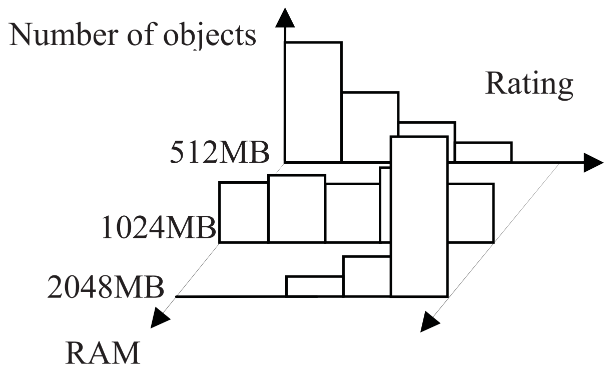


Figure 1: Uniform and expressive distributions of ratings.

weights learnt from user ratings. Method called "Statistical" was described in [3], [4]. The distribution of ratings for attribute values $a_i \in A_i$ are considered when determining the weight of attribute A_i. In Figure 1 is an example of distribution of ratings across a domain of attribute RAM for a user that prefers larger sizes of RAM. If ratings are distributed near one point as in Figure 1 for 2048GB RAM, the attribute value is considered as decisive when doing the overall rating. However, if the distribution is rather uniform such as in Figure 1 for 1024MB RAM, then attribute value a_i does not play an important role in the overall rating. The measure of importance of an attribute value is computed with formula

$$imp(a_i) = 1/\sum_{o \in X(a_i)} |r(o) - avg_{o \in X(a_i)} r(o)|/|X(a_i)|$$

when taking only those objects o that have attribute value a_i. Then the importance of attribute A_i is computed as $W(A_i) = 1/(\sum_{a_i \in A_i} imp(a_i/|A_i|)$.

Our approach is not the only one - any data mining technique can be used here. Reader can simply imagine training a multilayer perceptron with normalised attribute values.

In Figure 2 is attribute price with ratings of notebooks. When a standard linear regression (described in Section 4.1) is used, attribute price would get a small weight. However, when notebooks of only a single manufacturer are used, then the error would be much lower and importance of price would be considerably higher. This may be visible for human, after a short inspection of the graph. Another visualisation of this situation is in Figure 3 with discretised attribute price. Ratings of all attribute values are distributed uniformly, so the importance of this attribute will be considered small.

3 Related work

User preferences are a wide field so there are plenty of research areas. For our case, user preference learning is the most important one. There are two totally different approaches to recommendation : content based and collaborative. For collaborative filtering, we may cite for example [5] but there are many other interesting works. Collaborative filtering is based on the idea that similar users tend to have similar preferences of an object. However, we do not consider this approach in this paper.

We concentrate here on content based, that means that the preferences are based on attributes of objects. For example, user considers the size of harddisk and other features when

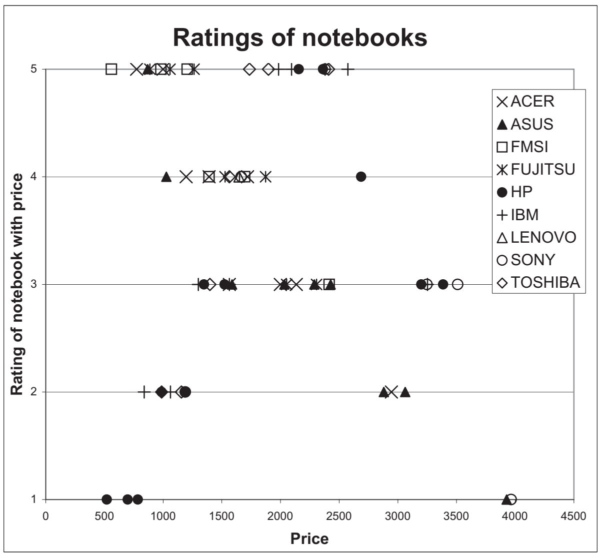


Figure 2: Distribution of ratings of notebooks with regard to price.

buying a notebook. This approach in our view corresponds more to how users decide in the real world. The difficulty here is that users are often inconsistent or take into account attributes that are not known or impossible to quantify, such as the design of notebook.

In this area, the main focus was on search of documents in the past [6, 7]. This is a specific area, because features of documents are of little structure. We are interested more in recommendation of more complicated objects like notebooks, with different types of attributes. Following works :[8, 9, 10, 11] deal with general, more structured, objects.

One of our main inspirations was CP-nets [2]. This user preference model captures complex user preferences in a graph representation, where a preference over one attribute may depend on the values of other attributes. There are many papers dealing with CP-nets, but there is none which describes a way to construct a CP-net automatically, by learning from ratings. Because preferences over attribute A_1 may depend on other attribute/s, e.g. A_2, user has to specify her preferences not only for all values of A_1 but she has to do it for every possible value of A_2. This means very much work and insight for the user, work that probably few people will undergo.

There is also another concept for representing user preferences other than ratings as in our case. These are preference relations, or specially fuzzy preference relations [12]. We do not consider this model, we rather follow approach identified by R.Fagin in [13] and consider user ratings as fuzzy sets.

4 Local preferences learning

As mentioned in Section 2, our user model is divided into two steps.

The first step is used for normalisation of every attribute to interval $[0, 1]$, so that the best object would have coordinates $[1, ..., 1]$. This can be viewed also as monotonisation of data - if for object o_1, the normalised value of its attribute value is lower in all attributes than object o_2, then o_2 is surely preferred to o_1 (if the normalisation is determined correctly).

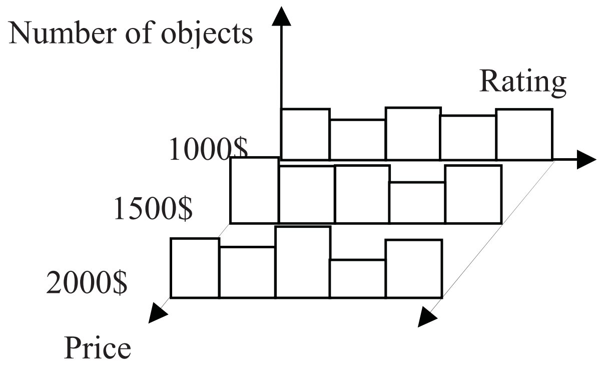
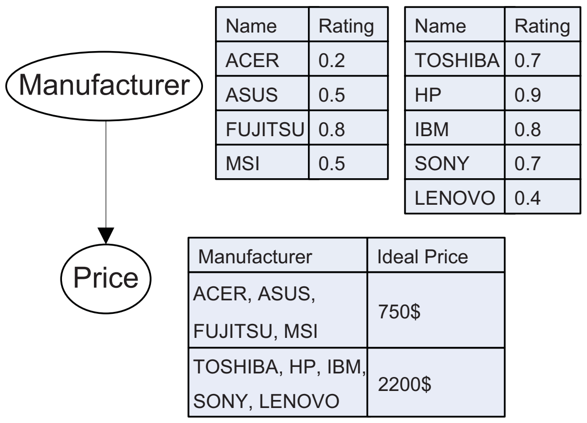


Figure 3: Uniform distributions of ratings across all domain.



Name	Rating
ACER	0.2
ASUS	0.5
FUJITSU	0.8
MSI	0.5

Name	Rating
TOSHIBA	0.7
HP	0.9
IBM	0.8
SONY	0.7
LENOVO	0.4

Manufacturer	Ideal Price
ACER, ASUS, FUJITSU, MSI	750$
TOSHIBA, HP, IBM, SONY, LENOVO	2200$

Figure 4: Example of a CP-net representing data about notebooks.

4.1 Linear regression

Linear regression is very useful method that serves to find a relation in a set of data in form of a linear function. It can be used for finding the preference of attribute values for numerical attributes, e.g. price. When given a set of notebooks' prices and ratings, linear regression will create a linear function $r(price) = \alpha * price + \beta$. Then we can normalise values of price using this function.

Traditional linear regression works on method of minimizing least squares of errors. This is not always useful, because it is dependent on the distribution of data points. For this reason, we proposed a method in [14] to accommodate better to the whole interval we study.

As a future work, we would like to implement a method for finding also more complicated functions, e.g. triangular fuzzy functions. The motivation is clear - linear functions are good for preferences of extreme values, e.g. lowest price, largest LCD etc. But if user prefers some value in the middle, like LCD of size 15", linear function would not capture this preference correctly.

4.2 2CP-regression

The main contribution of this paper is the proposal of a new method for acquisition of local preferences for numerical attributes, 2CP-regression. This method is motivated by preferences ceteris paribus [2] and CP-nets. Basic idea that represents ceteris paribus is that there are relations between attributes. For example, the manufacturer of a notebook influences the preferences of the price of that notebook. In our example, we present a user that prefers the price of 2200$ for manufacturers HP, IBM, Lenovo, Toshiba and Sony, and the price of 750$ for manufacturers Fujitsu-Siemens, Acer, Asus and MSI. A simple CP-net representing this scenario is in Figure 4, with ratings of manufacturers and ideal prices. Ideal price depends on the value of manufacturer, not on its preference. There are other examples and often the decisive attribute is a nominal one with rather small domain.

When constructing a fuzzy set for a numerical attribute A_1 (price in our case), we look at the values of other nominal attributes. For simplicity, let us consider only one, A_2. A_2 will be referred to as a "dividing attribute". The construction of the set is restricted to objects with value $a_2 \in A_2$; let us note this set $X(a_2) \subset X$. If dividing attribute A_2 really influences attribute A_1, then the values of A_1 should be "better" distributed in $X(a_2)$ than in whole X and the fuzzy set should better match the real preferences. The fuzzy set trained on $X(a_2)$ will be denoted as $\widehat{f_{a_2}^{A_1}}$. However, the size of training set may be radically decreased in this way. For that reason a general fuzzy set $\widehat{f^{A_1}}$ is constructed as in 4.1, which is used for values that did not occur during the construction of local preferences in the training set.

A method for construction of several fuzzy sets and a general one, all represented by linear functions, is described in the following pseudo code. Method `getObjectsWithA2`(a_2) returns the set $X(a_2)$, i.e. only those objects with attribute value a_2. Method `leaveOnlyA1AndRating` leaves only couples (A1, Rating) in the given set, which makes it suitable for linear regression that follows. Finally, the attribute value a_2 and the corresponding linear function $\widehat{f_{a_2}^{A_1}}$ are stored.

```
leaveOnlyA1AndRating(X);
$\widehat{f^{A_1}}$ = buildClassifier(X);
for ($\forall a_2 \in A_2$) {
    $X(a_2)$ = getObjectsWithA2($a_2$);
    if (|$X(a_2)$|< 2)
        continue;
    leaveOnlyA1AndRating($X(a_2)$);
    $\widehat{f_{a_2}^{A_1}}$ = doLinearRegressionOn($X(a_2)$);
    storeCouple($a_2, \widehat{f_{a_2}^{A_1}}$);
}
```

Let us have an object o with unknown rating, with attribute values $a_1, ..., a_n$. When we want to normalise value a_1, we use the following method:

```
if(existsFuzzySetFor($a_2$)){
    $\widehat{f_{a_2}^{A_1}}$ = getFuzzySetFor($a_2$);
    return $\widehat{f_{a_2}^{A_1}}(a_1)$ ;
}
else
    return $\widehat{f^{A_1}}(a_1)$;
```

It is possible that we did not encounter value a_2 in the training phase at all or only once. In such case, we use general function $\widehat{f^{A_1}}$ for normalisation of a_1.

4.3 A method for clustering results CP-regressions

In the previous section, we mentioned a problem that when using 2CP-regression, the number of training examples for linear regression decreases rapidly when the number of attribute values in the dividing attribute is high. In the following, we present a method for overcoming this problem.

Let us have a set of fuzzy sets $\widehat{f_{a_i}^{A_1}}$ for normalisation of attribute A_1 that depends on values of attribute A_2. Each fuzzy set is of the form $\widehat{f_{a_i}^{A_1}}(x) = \alpha * x + \beta$.

Now we start with fuzzy set $\widehat{f_{a_1}^{A_1}} = \alpha * x + \beta$ and match together those fuzzy sets, that have similar shape, ie. the values of α and β. After finding similar fuzzy sets, without the loss of generality let it be $\widehat{f_{a_2}^{A_1}}, ..., \widehat{f_{a_k}^{A_1}}$. Values $a_1, ..., a_k$ are grouped together and we construct a new linear function; we will use all objects $o \in X(a_1) \cup ... \cup X(a_k)$ for its construction. This new linear function should accommodate better to testing data, because it is constructed using a larger training set. When using this set, we treat it as $\widehat{f_{a_1,...,a_k}^{A_1}}$ so it is used for every object with attribute value in $a_1, ..., a_k$.

This method of using clusters for CP-regression was not fully implemented yet, but plan to implement it and test it in near future. We expect better performance especially for smaller training sets. In this case, we can do a bottom-up clustering that stops at a specified threshold of minimal training set size.

5 Experiments

5.1 Experiment settings

Our experiment was done on a set of 200 notebooks crawled from a web shop. There are five attributes: harddisk, display, price, producer and ram. We have created an artificial user U with preferences on the scale $\{1,2,3,4,5\}$ for every notebook. Preferences were generated using a set of fuzzy sets f_i and an aggregation function @.

The user preferences on price were dependent on the actual value of the producer of the notebook. As it was mentioned in Section 4.2, for producers HP, IBM, Lenovo, Toshiba and Sony the best price was set to 2200$ and for Fujitsu-Siemens, Acer, Asus and MSI the best price was set to 750$. Hence the same price would produce different degree of preference for different manufacturers. Distribution of ratings of notebooks with regard to their price is in Figure 2. We can clearly see that the ratings are distributed more or less evenly across all the domain. However, a peak at about 2500$ appears when only the triangles are taken into account and another peak at about 1000$ is for crosses. Peaks do not correspond exactly to the artificial preferences; this is because the overall rating is influenced by other attributes, too. The overall trend is clearly visible for human - with increasing price crosses tend to go down and on the opposite triangles go up.

Testing was performed with a traditional cross-validation method. Because we are dealing with user ratings, it can not be expected that the user will rate many objects. That is why we limited the size of the training set (TSS in following text) to 75 ratings at maximum. Methods were tested on the rest of the set. When the user model is constructed from the first 10 notebooks, then it is tested on the remaining 190 notebooks. Then the next 10 notebooks are taken as training set and again the model is tested on the remaining 190. For TSS = 75 the model is tested on remaining 125 notebooks. Resulting errors are averaged for every TSS so the results should be reliable.

There are five methods tested: Mean, Statistical with Linear regression, Statistical with 2CP-regression, Support Vector Machine and Multilayer perceptron. Mean is a baseline. This method always returns the average rating from the training set. Multilayer perceptron and Support Vector Machine are traditional data mining methods. Implementation from Weka [15] was used.

Statistical method as aggregation function was used with two different normalisations. The first one is traditional linear regression; the second is our proposed method 2CP-regression.

No tuning of any method was performed - these methods are supposed to work on arbitrary data automatically without the need of adjusting any parameters.

5.2 Experiment results

We studied three types of error measures. First, most commonly used, is RMSE - root mean squared error. This captures the average error across the whole testing set. Second, Weighted RMSE, adds weight to each contribution to error. The weight associated is the same as the real rating, because good objects are of more concern than objects that are not preferred. In other words, it is worse if a good notebook does not appear on the first page of results than if a bad notebook does. Finally, we used tau coefficient to study the methods. Tau coefficient is a measure of correspondence between two ordered lists. Objects from testing set ordered by real user preferences are in the first list. The second list consists of the same objects, ordered by preferences proposed by the method. In this way, we can estimate the similarity of the ordering of objects by the method and by the real user. Tau coefficient ranges from -1 to 1; 1 means absolute correlation, e.g. the same lists, and -1 means the opposite correlation, e.g. reversed lists. Tau coefficient of 0 means that there is no relation between the two lists.

In Figure 5 are represented counts of unpredicted rating for each method. At TSS = 2 there were 40 notebooks that every method failed to predict and SVM failed even for 90 notebooks. At TSS = 5, there were 5 notebooks for all methods and 25 notebooks for SVM without prediction. So SVM performed worse than the rest of methods for small TSS.

In Figure 6 are results for RMSE. 2CP regression is an overall winner, only at TSS = 40 it was outperformed by Multilayer perceptron and at TSS = 2 by SVM, but SVM failed to predict far more notebooks.

The results are about the same for Weighted RMSE, as we can see in Figure 7.

Finally, results for Tau coefficient are in Figure 8. 2CP-regression was outperformed by Mean for the three smallest TSS, otherwise it was again the best method.

In all cases, 2CP-regression was better than simple Linear regression and Statistical with 2CP-regression outperformed even SVM and Multilayer perceptron.

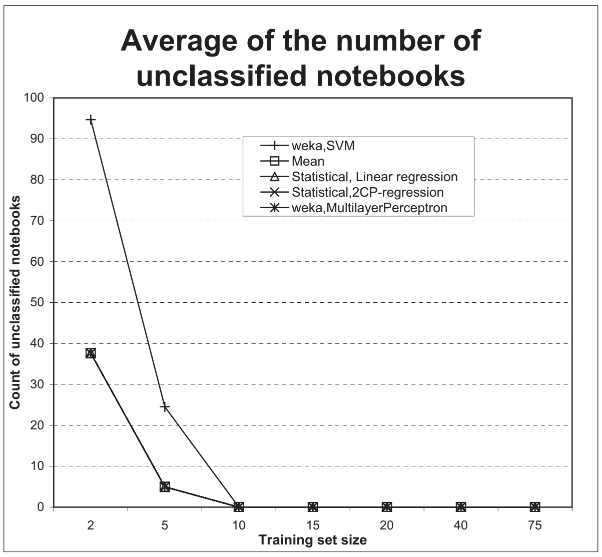


Figure 5: Count of unpredicted ratings of notebooks.

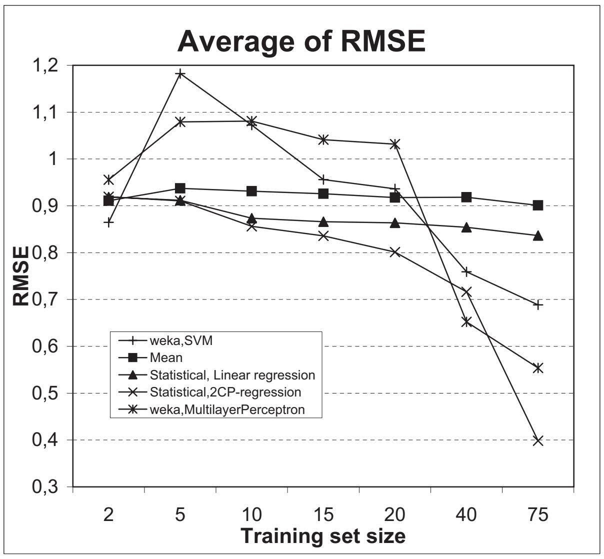


Figure 6: RMSE.

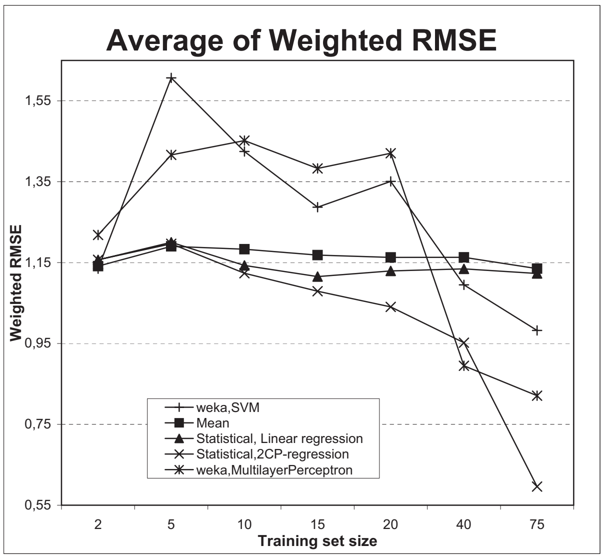


Figure 7: Weighted RMSE.

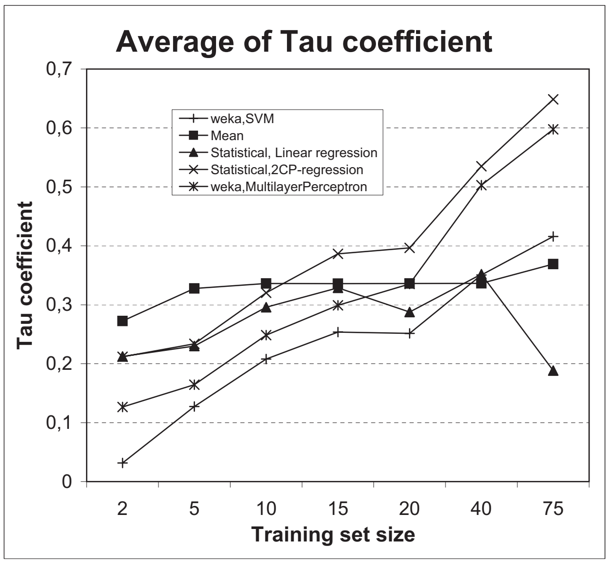


Figure 8: Tau coefficient.

6 Conclusions

We have proposed a method for finding preference dependence between two attributes, which was demonstrated on the example of a user for whom the ideal price depends on the manufacturer of the notebook. As far as we know this is the first contribution for learning ceteris paribus-network like structure from user ratings. We are aware that our model does not correspond to CP-nets completely, but the proposed part of our fuzzy model can be considered as a simple CP-net.

In our experiments, 2CP-regression performed very well, outperforming traditional data-mining techniques and also the normalisation with linear regression. Three different error measures were taken into account and in every one our proposed method was the best.

For future work, we would like to extend this approach for finding relations between more than two attributes, making a more general nCP-regression. We would also like to test it on real user preferences data, for example on NetFlix data [16].

Acknowledgment

The work on this paper was supported by Czech projects MSM 0021620838, 1ET 100300517 and GACR 201/09/H057.

References

[1] G. H. Wright. The logic of preference reconsidered. In *Theory and Decision*, volume 3, pages 140–169, 1972.

[2] Craig Boutilier, Ronen I. Brafman, Holger H. Hoos, and David Poole. Cp-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *Journal of Artificial Intelligence Research*, 21:2004, 2004.

[3] Alan Eckhardt. Inductive models of user preferences for semantic web. In Jaroslav Pokorný, Václav Snášel, and Karel Richta, editors, *DATESO 2007*, volume 235 of *CEUR Workshop Proceedings*, pages 108–119. Matfyz Press, Praha, 2007.

[4] Alan Eckhardt, T. Horváth, D. Maruščák, R. Novotný, and Peter Vojtáš. Uncertainty issues in automating process connecting web and user. In P. C. G. da Costa, editor, *URSW '07 Uncertainty Reasoning for the Semantic Web - Volume 3*, pages 97–108. The 6th International Semantic Web Conference, 2007.

[5] Al Mamunur Rashid, Istvan Albert, Dan Cosley, Shyong K. Lam, Sean M. McNee, Joseph A. Konstan, and John Riedl. Getting to know you: learning new user preferences in recommender systems. In *IUI '02: Proceedings of the 7th international conference on Intelligent user interfaces*, pages 127–134, New York, NY, USA, 2002. ACM.

[6] M. Ackerman, D. Billsus, S. Gaffney, S. Hettich, G. Khoo, D. Kim, R. Klefstad, A. Ludeman, J. Muramatsu, K. Omori, M. Pazzani, D. Semler, B. Starr, and P. Yap. Learning probabilistic user profiles: Applications to finding interesting web sites, notifying users of relevant changes to web pages, and locating grant opportunities. *AI Magazine*, 18:47–56, 1997.

[7] Thorsten Joachims. Optimizing search engines using clickthrough data. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142, New York, NY, USA, 2002. ACM Press.

[8] Kim Cao-Van. *Supervised Ranking, from semantics to algorithms*. Ph.D. dissertation, Ghent University, 2003.

[9] Bruno Apolloni, Giacomo Zamponi, and Anna Maria Zanaboni. Learning fuzzy decision trees. *Neural Networks*, 11(5):885–895, 1998.

[10] Stefan Holland, Martin Ester, and Werner Kiessling. Preference mining: A novel approach on mining user preferences for personalized applications. In *Knowledge Discovery in Databases: PKDD 2003*, pages 204–216. Springer Berlin / Heidelberg, 2003.

[11] Sung Young Jung, Jeong-Hee Hong, and Taek-Soo Kim. A statistical model for user preference. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):834– 843, June 2005.

[12] Bernard de Baets and Jnos C. Fodor. Twenty years of fuzzy preference structures (19781997). *Decisions in Economics and Finance*, 20:45–66, 1997.

[13] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. In *In proceedings of Twentieth ACM Symposium on Principles of Database Systems, 2001 (PODS 2001)*, pages 102–113. ACM, 2001.

[14] Alan Eckhardt, T. Horváth, and Peter Vojtáš. Learning different user profile annotated rules for fuzzy preference top-k querying. In H. Prade and V. Subrahmanian, editors, *International Conference on Scalable Uncertainty Management*, volume 4772 of *Lecture Notes In Computer Science*, pages 116–130, Washington DC, USA, 2007. Springer Berlin / Heidelberg.

[15] Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques, 2nd Edition*. Morgan Kaufmann, San Francisco, 2005.

[16] Netflix dataset. *http://www.netflixprize.com.*