A clustering algorithm to find groups with homogeneous preferences[1]

J. Díez, J.J. del Coz, O. Luaces and A. Bahamonde

Centro de Inteligencia Artificial. Universidad de Oviedo at Gijón, Campus de Viesques, E-33271 Gijón (Asturias), Spain
{jdiez, juanjo, oluaces, antonio}@aic.uniovi.es

Introduction

When we are learning people's preferences the training material can be expressed as in regression problems: the description of each *object* is then followed by a number that assesses the degree of satisfaction. Alternatively, training examples can be represented by *preference judgments*: pairs of vectors (**v**, **u**) where someone expresses that he or she prefers **v** to **u**. Usually, obtaining preference information may be easier and more natural than obtaining the labels needed for a classification or regression approach. Moreover, this type of information is more accurate, since people tend to rate their preferences in a relative way, comparing objects with the other partners in the same batch.

Starting form a set of preference judgments we have to learn a real *ranking function* **f** in such a way that **f(v) > f(u)** whenever **v** is preferable to **u**. If we accept linear ranking functions, **f** is represented by a hyperplane able to separate increasing or positive differences (like **v – u** when **v > u**) from decreasing or negative differences (like **u – v**); see [2, 3, 6]. We will employ an SVM classifier: SVM^{light} [4].

Clustering of preference criteria

In this paper we present a new algorithm for clustering preference criteria. This is a useful task. For instance, in [6], Joachims presents an information retrieval system equipped with a ranking function learned from click-through data collected from user interaction with a www search engine. To improve his proposal, the author acknowledges the need to obtain feasible training data. This requires developing clustering algorithms to find homogeneous groups of users.

An Adaptive Route Advisor is described in [3]; the system is able to recommend a route to lead users through a digitalized road map taking into account their preferences. An interesting extension discussed in the paper is to modify route recommendations depending on the time of the day or the purpose of the trip. The approach suggested includes an algorithm that clusters user preferences into contexts.

[1] The research reported in this paper has been supported in part under the Spanish MCyT and Feder grant TIC2001-3579.

Another important field of application is the analysis of sensory data used to test the quality (or the acceptability) of market products that are principally appreciated through sensory impressions. Here there are panels of experts and panels of untrained consumers. Both to select experts to the first kind of panel, and to discover groups of preferences in consumers, clustering algorithms are necessary.

The clustering algorithm

The straightforward approaches to find groups of people with similar preferences require datasets with users' ratings instead of preference judgments; this has the difficulties alluded to above. These approaches include Pearson's correlation or the cosine of rating vectors; they were devised for prediction purposes in collaborative filtering (see for instance [1]) and they are not easily extendable to clustering.

The algorithm proposed in this paper has as input a family of preference judgment sets of a number of people. The key insight involved in the clustering algorithm is that ranking functions, learned from each preference judgment set, codify the rationale for these preferences. Therefore, we will try to merge data sets with *similar* ranking functions. So, given the preference judgments of two people labeled by 1 and 2, PJ_1, and PJ_2, we compute the similarity of the director vectors $\mathbf{w}_1$ and $\mathbf{w}_2$ of their ranking functions by means of their cosine:

$$\text{similarity}\,(\mathbf{w}_1, \mathbf{w}_2) = \frac{\mathbf{w}_1 \times \mathbf{w}_2}{\|\mathbf{w}_1\| \times \|\mathbf{w}_2\|}. \qquad (\mathbf{1})$$

However, this measure is solely a heuristic to suggest pairs with similar preference criteria. The reason for this is that it is not necessary for $\mathbf{w}_1$ and $\mathbf{w}_2$ to be similar vectors in order to guarantee that two people have similar *preference criteria*. On the other hand, due to possible overfitting of the learning algorithm used, and the typical sparsity of data, sometimes rather different director vectors may codify similar preference criteria.

Taking all this into account, given two groups of people with similar $\mathbf{w}_1$ and $\mathbf{w}_2$, we learn the ranking function from the merged data set, say $\mathbf{w}$. Then we aggregate the groups if the estimated quality of $\mathbf{w}$ is higher than that of the individual groups. The algorithm stops when no more merges can be achieved. See Figure 1 for a detailed description of the algorithm.

Experimental results

To evaluate the algorithm we used a collection of preference judgments taken from EachMovie, a publicly available collaborative filtering database for movie ratings. The ratings were used only to build preference judgments pairs, so no kind of regression was used in our work. In fact we used EachMovie just in order to *simulate* reasonable situations that can be easily found in real word data involved in the study of preferences of groups of people.

```
A set of clusters ClusterPreferencesCriteria
   (a list of preference judgments ($PJ_i$: i = 1,…, N)) {
      Clusters = ∅;
      for each i = 1 to N {
         $\mathbf{w}_i$ = Learn a ranking hyperplane from ($PJ_i$);
         Clusters = Clusters ∪ {($PJ_i$, $\mathbf{w}_i$)};
      }
      repeat {
         let ($PJ_1$, $\mathbf{w}_1$) and ($PJ_2$, $\mathbf{w}_2$) be the clusters
              with most similar $\mathbf{w}_1$ and $\mathbf{w}_2$;
         $\mathbf{w}$ = Learn a ranking hyperplane from ($PJ_1$ ∪ $PJ_2$);
         if (the estimated quality of $\mathbf{w}$ >=
               (the estimated quality of $\mathbf{w}_1$ +
                the estimated quality of $\mathbf{w}_2$))
         then replace the clusters ($PJ_1$, $\mathbf{w}_1$) and ($PJ_2$, $\mathbf{w}_2$)
                      by ($PJ_1$ ∪ $PJ_2$, $\mathbf{w}$) in Clusters;
      } until (no new merges can be tested);
      return Clusters;
}
```

Fig. 1. Clustering Algorithm. Starting from a list of N sets of preference judgments, the algorithm outputs a set of clusters endowed with their ranking functions

We considered the set of the 100 spectators with more ratings in EachMovie in order to look for groups of them with similar preferences. The movies are described in our experiments by vectors of 808 components: the ratings provided by the rest of the spectators who submitted at least 200 ratings for the movies with at least 1,000 ratings. The resulting 504 movies were then randomly separated into three subsets: training 60%, verification 20%, and test 20%. We will call these data sets the *808-collection*. The scores achieved with this collection are reported in Table 1. To deal with smaller sets, we also considered the *89-collection*, where each movie is described by the ratings of the 89 spectators, from the set of 808, with more than 275 ratings; see Table 2.

In these experiments, to estimate the quality of the ranking functions, we compute the confidence interval of the probability of error when we apply each ranking function to the corresponding verification set; we use the confident level $\alpha = 0.05$. Let [L, R] be such an interval. Then the quality is the estimated proportion of successful generalization errors in the pessimistic case; that is, 1-R. However, according to the availability of data, it is possible that we can not afford to have a separate verification dataset. In those cases, we should use any available tool to estimate the generalization error using only the training set. In some cases, the Xi-alpha estimator [5] is a good candidate for this job.

Table 1. 808-Collection. Generalization errors of ranking functions of the biggest clusters computed with test examples and the average of individuals separately. The clusters are ordered by the number of spectators included. The number of individual spectators considered is 100

Number of	By Clusters		Individually		Error
spectators	Test ex.	Errors	Av. test ex.	Errors	Difference
35	22,590	20.06%	645	18.32%	-1.74%
13	7,770	21.47%	605	20.75%	-0.72%
5	3,420	17.02%	684	14.82%	-2.19%
4	2,280	37.50%	570	33.29%	-4.21%
4	2,650	28.19%	662	23.55%	-4.64%

Table 2. The scores of Table 1, for the 89-Collection

Number of	By Clusters		Individually		Error
spectators	Test ex.	Errors	Av. test ex.	Errors	Difference
54	34,990	21.16%	648	21.20%	0.03%
18	12,240	27.60%	680	26.08%	-1.52%
9	5,210	27.85%	579	29.02%	1.17%
4	2,040	35.64%	510	32.40%	-3.24%
4	2,590	33.78%	647	33.59%	-0.19%

References

1. Breese, J.S., Heckerman, D., Kadie, C.: Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In: Cooper, G. F. and Moral, S. (eds.): Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence. Morgan Kaufmann, San Francisco (1998) 43-52
2. Díez, J., del Coz, J.J., Luaces, O., Goyache, F., Peña, A. M., Bahamonde, A.: Learning to Assess from Pair-wise Comparisons. LNCS Vol. 2527. Springer-Verlag, (2002) 481-490
3. Fiechter, C.N., Rogers, S.: Learning Subjective Functions with Large Margins. Proceedings of the Seventeenth ICML. Morgan Kaufmann, (2000) 287-294
4. Joachims, T.: Making Large-Scale SVM Learning Practical. In: Schölkopf, B., Burges, C., Smola, C. (eds.): Advances in Kernel Methods - Support Vector Learning, chapter 11. MIT Press, (1998)
5. Joachims, T.: Estimating the generalization performance of a SVM efficiently. In *Proceedings of the International Conference of Machine Learning*, San Francisco. Morgan Kaufman, (2000)
6. Joachims, T.: Optimizing Search Engines Using Clickthrough Data. Proceedings of the ACM Conference on KDD. (2002)