Application of ART2 Networks and Self-Organizing Maps to Collaborative Filtering

Guntram Graef and Christian Schaefer

Telecooperation Office (TecO), University of Karlsruhe, Vincenz-Priessnitz Str. 1, 76131 Karlsruhe, Germany, Tel.: +49 (721) 6902-89, Fax: -16,
E-Mail: {graef}@teco.edu

Abstract. Since the World Wide Web has become widespread, more and more applications exist that are suitable for the application of social information filtering techniques. In collaborative filtering, preferences of a user are estimated through mining data available about the whole user population, implicitly exploiting analogies between users that show similar characteristics. These preferences are then normally used to filter content or functionality of an application. Two important factors for the quality of the filtering process are the number of users and the amount of information (such as observed behaviors) available about each user. Another factor is the number of objects in the pool of the application that can be considered during the filtering process. Today in most cases memory based approaches to collaborative filtering are used. Unfortunately with O(#users * #items) those do not scale well. Therefore we implemented a model based approach using two different types of neural networks and benchmarked them against a widely used memory based approach. Especially with ART2 networks we obtained some encouraging results.

1 Introduction

The World Wide Web has been established as a major platform for information and application delivery. The amount of content and functionality available often exceeds the cognitive capacity of users. This problem has also been characterized as information overload [15].
Various approaches exist that address this issue, such as search engines [7], web catalogs or filtering techniques based on user profiles, such as collaborative filtering [21].
In collaborative filtering, user profiles are generated that describe user preferences in relation to items within a specific domain. Depending on the application, items can e.g. be Web resources, components [14], services [12], or products [1]. Initial knowledge about user preferences can be obtained either explicitly such as from ratings by users [21] or implicitly through behaviour analysis [19] [13]. For each user a vector is generated with one entry for each known item. The profile vectors are then used as input for either a memory based or a model based method to compute item recommendations by exploiting information stored in profiles that show similarities to a given profile. An often used memory based method is the *Mean Squared*

Differences Algorithm [21]. As illustrated in figure 1, a given profile D is compared to profiles of other users to find the n nearest neighbours i.e. the n most similar profiles that are not equal to D. For this purpose a vector distance metric is employed. In the example, n=2 and thus profile A and C are selected. A target vector is then computed as the average of the neighbour vectors. Depending on the type of the application, the target vector is used to recommend items or to find estimates for blank parts of the original user profile.

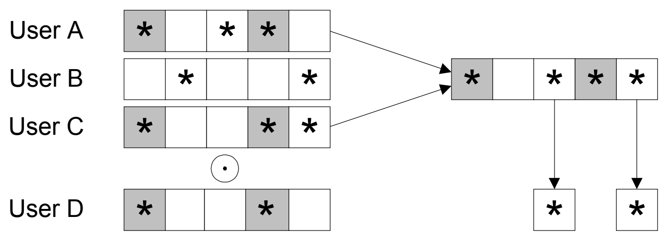


Fig. 1. Memory based collaborative filtering

In model based approaches such as [10] [6], all available profile vectors are first learned by a model. Later single profile vectors can be applied to the model to either obtain a target vector or directly receive recommendations. Figure 2 illustrates this.

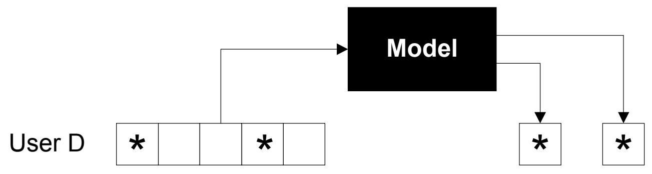


Fig. 2. Model based collaborative filtering

A model usually is smaller in size than the whole set of profile data and no any-to-any matching of profiles is necessary which often leads to performance problems in memory based approaches. On the other hand, an abstraction is performed that usually leads to an information loss and the adaptability of the model to changes to a profile is an issue.

Some applications of collaborative filtering can be found in [21], [17], [22] and [14].

In the first part of this contribution we describe two memory based approaches to collaborative filtering using self-organizing maps (SOMs) [16], and Adaptive Resonance Theory (ART) networks [9]. In the second part of the paper we perform an evaluation with two sets of test data from real world applications and compare the two approaches with a widely used memory based approach.

2 Using Neural Networks for Collaborative Filtering

A large variety of neural networks have been described [20]. Neurons are modeled after nerve-cells in animals. Quite popular is the McCulloch-Pitts Neuron [8] shown in figure 3 (left). The activation of a neuron *j* is computed by comparing a threshold value θ_j to the weighed input $w_{ij}s_i$. In effect, the neuron performes pattern recognition with the angle between input and weight vector being a measure of conformity. To obtain an adaptive filter this angle is also used during learning, instead of the output value that is used in supervised learning. The weight vector of the neuron is then adjusted towards the input vector as shown in figure 3 (right). This is called unsupervised learning. An important parameter is the learning rate δ that determines how quickly the weight vector is adjusted. Usually a high learning rate is used at the beginning which is then decreased when more input vectors are learned.

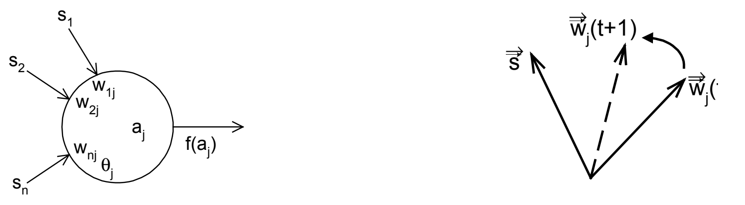


Fig. 3. McCulloch-Pitts Neuron (left), Adjustment of a weight vector to an input pattern (right)

Neurons are connected to form networks where the output signal of one neuron is used as input signal for other neurons. Competitive learning can be realized by organizing McCulloch-Pitts Neurons in a layer and presenting an input pattern to each neuron and then only allowing the neuron with the highest activation to produce an output signal. Thus the neuron layer responds to each input pattern with a specific neuron. It independently classifies all patterns into clusters.
Two types of competitive learning are described with the Adaptive-Resonance-Theorie [9] and Self-Organizing Maps [16].

2.2 Self-Organizing Maps

In Self-Organizing Maps (SOMs) [16] not only weights are important but also the location of neurons within the layer. Similar neurons that classify similar patterns are closely located. During the learning process a spacial distribution is created so that neighbour neurons are activated by similar signals. Following the biological example, neurons are arranged in spaces of low dimensionality. This is why the term "map" is used. A topology preserving mapping is attained that maps highly dimensional input patterns to few spacial dimensions. Thus, a strong compression of dimensionality is performed.
Usually, two dimensional maps are used for classification tasks since they allow for greater flexibilty in neighbourship relations than one dimensional maps while allowing for easier computations than maps with more than two dimensions. As

shown in figure 4 (left) for an example of three neurons, each neuron of the map is fully connected to the input layer. The weight vectors are thus of the same dimensionality as the input vectors. The neurons at the border of the map have less neighbours than those in the center. This leads to an undesired preferential treatment of some neurons. To avoid this, the map is projected to the surface of a sphere [3]. In figure 4 (left) the neighbourhood of a neuron is shown in the case of a spherical mapping.
During the learning phase competitive learning is used to determine a winner neuron for each pattern in the training set. But now not merely the winner neuron is adjusted according to the training pattern but also all neighbour neurons. The learning process of a neuron is not independent from other neurons which is why a global learning method must be used [5].

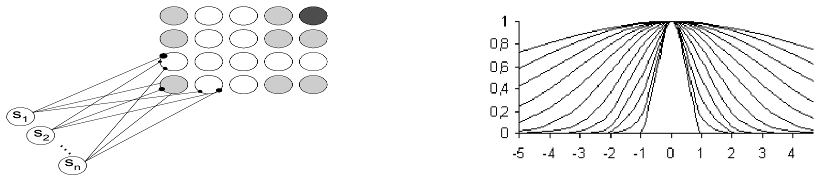


Fig. 4. 2-dimensional SOM with input signal (s{n}), weighted connections showing neighbourhood relations (grey) of a neuron (black)

In each learning step the neuron with the strongest activation by the external signal is determined. If for the McCulloch-Pitts neurons the additional conditions are met that the sum of all weights is constant and the input signal are normed, then the winner neuron k can be determined by applying the euklidian norm.
The weight adjustments of the winner neuron are performed according to the formula that describes the learning rule for competitive learning: $w_j(t+1) = w_j(t) + \delta (t)*(s-w_j(t))$. The adjustment of the neighbouring neurons j depends on their distance d_{jk} from the winner neuron k. The strength of the adjustment is described as a function of the distance. With this function $h_{jk}(t)$ and learning rate $\delta(t)$ the following learning rule is true: $w_{ij}(t+1) = w_{ij}(t)+ \delta(t) * h_{jk}(t) * (s_i - w_{ij}(t))$.
In [16] and in [5] the calculation of the adaptation strength is performed with the Gauss bell: $h_{jk}(t) = -d_{jk} / e^{2\sigma(t)* 2\sigma(t)}$. At the beginning of the learning phase the general structure of the map must develop. After some time smaller details should manifest themselves on the map. Therefore at the beginning even neurons that are relatively distant are adjusted, while later only local adjustments are performed. Figure 4 (right) shows how through the dependance of the Gauss function on the learning duration the neighbourhood is affected less and less by the learning process.

Stability and Plasticity
The application of a SOM to a classification problem is only interesting if there are more patterns than neurons. A neuron must be able to represent more than one pattern of a cluster. If the patterns greatly differ, after each learning step the weight vector is adjusted towards the applied pattern but away from previously learned ones. Thus the neuron "jumps" within the cluster. To avoid this problem and to obtain a stable net,

the learning rate is reduced in order that later changes only slightly modify the weight vector. This way a convergence is forced.
To create a clustering for a set, representative samples must be presented to the map several times during the learning phase. If the net shall be able to learn new patterns even after the learning phase, the learning rate can't be reduced too much and the influence on the neighbourhood during the learning process must stay rather strong. Thus, the ability of the net to be shaped after the learning phase can only be maintained if stable weights are abandoned. A stable net can't be adjusted. This problem is called stability-plasticity problem and has been addressed in [9] with the Adaptive-Resonance-Theorie.

2.3 Adaptive Resonance Theorie

On the SOM the number of neurons is fixed and can't be changed. If during the learning phase a pattern is applied to which no neuron strongly responds, a neuron is selected pretty much by random and adjusted to that pattern. In the worst cases that neuron had already been adjusted optimally to a class of patterns from the training set. By adjusting the weights to the new pattern the weight vector representing that class is changed which results in the classification being unlearned. In that case weights won't stabilize.
The Adaptive-Resoncance-Theorie solves this problem by making the neuron layer adaptive. This means new neurons can be added step by step if a pattern does not match any existing neuron closely enough. Furthermore, a winner neuron can reject the pattern if the similarity is too low. The winner neuron therefore sends its weight vector back to the input layer and only learns the new pattern if it lies within a cone around the weight vector [20]. The size of the cone is determined by the vigilance parameter ρ. Figure 5 (right) shows the attentiveness cone.

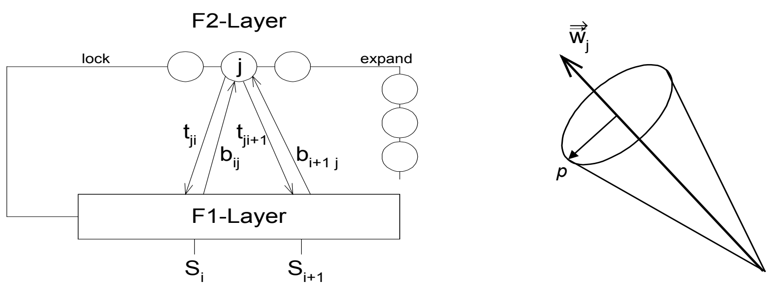


Fig. 5. Set-up of an ART network (left) and attentiveness cone of a weight vector (right)

The net contains two layers F1 and F2. F2 is the competitive layer and consists of a set of ordered neurons that do not consitute any neighbourhood relations. For each input pattern a winner neuron is determined in that layer. The F1 layer controls the classification. If a pattern is not within the attentiveness cone the F1 layer blocks the winner neuron for that pattern. Between the layers weighted connections exist in both directions. Figure 5 (left) shows the set-up.

During the learning process a pattern is applied to the F1 layer. Through the weights that are directed upwards the neuron with the strongest activation is determined by computing the scalar. By applying the weight vector that is directed downwards, the F1 layer checks if the pattern lies within the attentiveness cone. If that is the case, both weight vectors are adjusted to the pattern. Otherwise the F1 layer blocks the winner neuron during the further processing of the pattern and tries to locate another winner neuron. If none of the existing neurons fits, then a new neuron that matches the pattern is added. A detailed description of the F1 layer and the learning process can be found in [11].
The descriped approach makes sure that the net always converges to a stable state and still maintains plasticity. A potential disadvantage is that under certain circumstances the optimal number of classes can be greatly surpassed [5]. Thus the extension of the competitive layer F2 is limited to a maximum number of neurons. After the limit has been reached, no new neurons will be added and rejected patterns must be discarded.
The Adaptive Resonance Theorie exists in two versions. One version is restricted to binary input patterns while the ART2 networks that we selected for our research have been designed to process analog input patterns.

3 Applied Model

In the previous sections two neural networks have been introduced that are able to cluster a training set without supervision and to classify patterns applied after a learning phase. The nets can therefore be used to assign a profile to a class of similar profiles. Information is stored in the weight vectors of the neurons. Besides classification this allows for another application of the nets, as described in the following section.
The weight vectors of the neurons in the competitive layer are also called reference vectors. During each step of the learning phase, the weight vectors are adjusted towards an applied pattern. In the case of competitive learning a weight adjustment is only performed if a similar enough pattern is used. Then the weight vector converges towards the average of all learned patterns. It more or less describes each learned pattern. The weight vector is thus a *"codebook"* for the represented class.
In an ART2 network, the weight vectors that are directed downwards are the reference vectors. For the learning process in an ART2 net two different learning methods can be applied. But only the slower one of the two methods produces a reference vector. In contrast to a SOM the weight vectors are not normed. Still, they converge i.e. the vector norm converges towards a constant value $1/(1-d)$ whereas d is the constant output of a winner neuron. To reach that limit a large number of learning steps is required which is why usually convergence is forgone.
In the SOM the normed weight vector also describes the "learning history" of a neuron. However, in that case during the learning phase the reference vector is also influenced via neighbourhood relations. But since the neighbourhood contains similar neurons this does not necessarily have a negative effect. It is rather assumed that this modification of the reference vector is desirable since it prevents neurons to specialize too strongly on a small number of similar patterns.

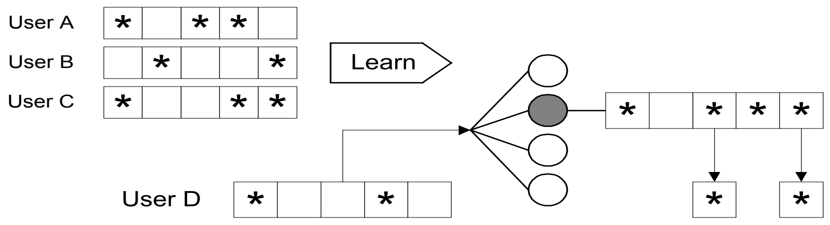


Fig. 6. A neural network as a model for generating predictions. Asterixes mark user interest.

The ability to classify and the formation of reference vectors form the basis for using neural networks as a model in collaborative filtering. Figure 6 shows the set-up.
The first step is to determine the network parameters. The learning rate and the number of learning steps are empirical values and can be chosen similar for every application. In contrast the network size must be estimated individually for different data sets. If the network is too large, neurons specialize too much on a few patterns. Is the network too small the neurons generalize too much and the reference vector becomes useless. As an indicator for estimating the network sizes the number of profiles is used. It is assumed that with a large number of profiles more different behavioral patterns i.e. more independent profiles exist than with a smaller number.
In an ART2 net the similarity parameter must be specified. It can be set automatically during the learning process. A maximum value is used for initialization. If during the learning process a pattern of no neuron can be accepted the parameter is decreased until the pattern can be assigned to a neuron.
After the determination of the profile and the initialization of the weights, the network learns the existing profiles. After the learning process has been completed the network is able to classify profiles and to assign them to a reference vector that can be used as a basis for compiling recommendations.

Learning

From the database a number of representative and expressive profiles are compiled to a training set that is then learned by a neural network. Less expressive profiles are those that are only sparsely populated.
The higher the dimensionality of the input vectors the more difficult it becomes for a neural network to a achieve a useful clustering because the number of independent patterns that do not show similarities increases. For each of those patterns an individual neuron must be learned.
If in addition to this, the training set contains sparsely populated profile vectors, the network must use a large part of its resources or neurons for the classification of vectors with low expressive value. The result is a network that does badly represents the training set. Thus only profiles that contain a minimum number of entries should be selected for training.
Both SOMs and ART networks possess the ability to learn after completion of the initial learning phase. If the profile of a user changes it can be learned again. However, this leads to an advantage for profiles that change frequently over profiles

that seldom change because it is learned more often. In the worst case, other profiles might be unlearned as the weight vector shifts towards the re-learned profile.
The re-learning of changed profiles can be handled best by the SOM because in that case the global structure of the map needn't be modified. A changed profile usually is still similar enough to the original profile to be represented by the same neuron or at least one of its neighbours. Thus it is unlikely that existing patterns are un-learned. Changes can be performed localy. On the other hand the ART2 network has problems if the changed profile lays no longer within the similarity cone of the neuron that it had been assigned to before. In that case the profile will be assigned to a new neuron and over time the optimal clustering of the training set will be destroyed.
If a new profile is added that has no similarities to existing profiles, it can easily be added to the ART2 network while the SOM would have to adjust its global structure, which would have to be done by re-learning the whole training set. Thus a SOM tends to forget patterns when new patterns are learned.
It is also possible to add new items after a network has been trained. For this purpose the number of input neurons must be increased because a new dimension is added to the profile vectors. In both cases the extension is easily performed by adding a new initial weight to each neuron. To avoid violating the requirement of equal weight sums it only needs to be ensured that the initial weight values stay within a small interval.

Requests
After the training phase the network is ready to classify profiles. To generate recommendations for a user his or her user profile is applied to the network. Each neuron of the competitive layer computes its activation and a winner neuron is determined. This neuron represents the class of the profile. Instead of generating a target profile from all profiles of this class, the reference vector of the neuron is used i.e. the target profile has already been generated when computing the reference vector during the learning phase. As in the case of the *Mean Squared Differences Algorithm*, the target vector is compared to the applied profile and recommendations are made for items that appear to have been underused by the user.
When determining the winner neurons, a scalar product must be computed for each neuron of the competitive layer. The complexity of a request for recommendations thus depends on the number of items and the number of neurons: O(#neurons * #items). The number of neurons in the network does not depend on the number of users but on the number of relevant user clusters. This factor tends to be smaller by orders of magnitude and can be treated as a quality of service parameter.

Limitations
A winner neuron is determined by choosing the neuron with the highest activation. This means, only the similarity between input vector and weight vector is recognized. But if two vectors are opposite to each other this can also carry potentially useful information.

4 Evaluation

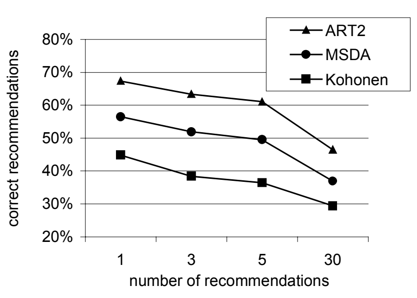
Three requirements for neural networks can be identified: Requests for recommendations must be processed quickly, the recommendations should be of high quality and the network should be able to adapt to changes to profile data during run-time. The first experiments presented here have been performed to test if those requirements were met when using two test data sets. The performance of the neural networks has been compared to memory-based Collaborative Filtering based on the *Mean Squared Differences Algorithm*. After presenting our first experiments, the learning duration, selection of parameters and the dependance of the quality of recommendations on the training set will be discussed. Based on those criteria SOMs and ART2 networks will be compared.

4.1 Test Data

We used two test data sets. Most results presented here have been obtained by using the publicly available EachMovie dataset [18]. It contains 2.811.983 ratings on a scale from 1 to 5 for 1628 movies by 72.916 users. As in [4] we randomly selected a subset of 2000 users that had a minimum of 80 entries in their profiles. In each profile 30 entries are then randomly selected as control set while the others are used as input for the filtering algorithms. To test the performance of the different methods with sparse profile data from a different domain, we used data that we obtained in a large intranet E-Commerce application, Eurovictor II [14]. The data describes the intensity with which employees have accessed different services of the application.

4.2 Response Time and Quality of Predictions

For our experiments we used a 10x10 SOM and an ART2 network for a maximum of 50 neurons. We also used *Mean Squared Differences Algorithm* with a neighbourhood size of 20. We used the EachMovie-dataset. To simulate a growing database, we did three experiments using 20%, 60% and 100% of available profile entries, with 30 control set entries in each case that we used to evaluate the computed recommendations.
The three different subsets have been used as training sets for the neural networks and as input for the memory based method. Afterwards one, three, five and 30 ommendations have been computed and compared to the control set of 30 hidden profile entries. The results of those experiments can be found in figure 7 (left). As can be seen, the ART2 network performed significantly better than the memory based method while the SOM produced results of the lowest quality.
Each time recommendations were computed the response time has been measured. As expected and shown in figure 7 (right), the performance of the neural network based methods mostly scales with the number of items, while the number of users has a much less significant influence. The three test sets contain profile entries for 400, 1200 and 2000 of users. While the memory based approach has to calculate scalars for all users, the neural networks only need to compute 50 and 100 scalars respectively



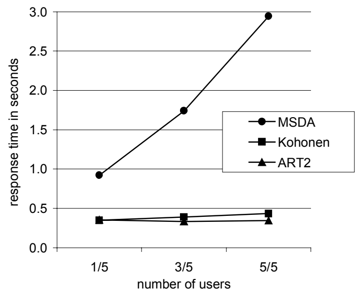
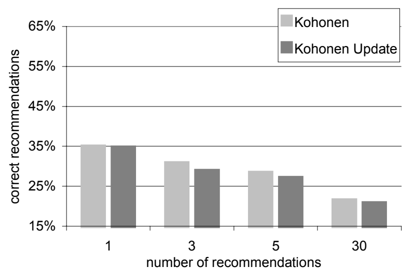


Fig. 7. Quality of results (left) and response time with growing number of users (right)

4.3 Adaptivity

A third important requirement besides quality of recommendations and performance is the ability of the neural networks to adapt during run-time. The networks can't relearn all profiles whenever a single profile changes since the complete learning process takes a considerable amount of time, in the case of our test sets several minutes. Therefore new or changed profiles must be (re-)learned by the network without repeating the initial learning phase.
We performed an experiment to test the ability of our networks to adapt to changes or additions of profiles. The same test data as in the previous experiments was used. This time we started with 400 profiles that we randomly selected from the complete set of 2000 user profiles and used them as training data on the networks. Then we randomly selected 800 profiles that were learned by the networks, then 1200, 1600 and finally all 2000. Please note that those data sets were not disjunct and some profiles were learned more often than others. Then, recommendations were generated and compared to the control set of 30 profile entries that were not used for training.
Figure 8 (left) shows the results of the experiment using a SOM. To un-learn as few patterns as possible when updating the network, the map's learning rate is reduced from 0.8 to 0.5. There is only a slight difference in the quality of the obtained results. The SOM shows a high degree of adaptivity.



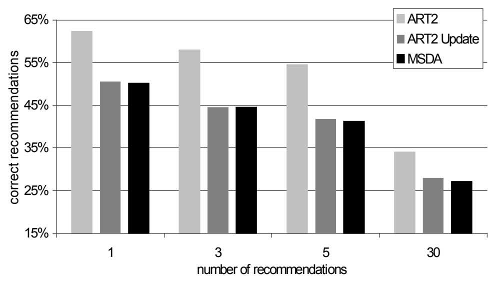


Fig. 8. Quality of results when gradually updating the network compared to relearning all data

Figure 8 (right) shows the corresponding results for an ART2 network. To preserve the knowledge of previously learned patterns the number of learning steps has been reduced from 10 to 5. The quality of predictions decreased in our experiments and after (re-)learning all profiles from the 5 training subsets the quality of the recommendations fell to the level of what we obtained using our *Mean Squared*

Differences Algorithm implementation. The neurons have not been adapted to profile changes in a step by step manner. Modified profiles have been rejected as dissimilar to existing neurons and have been assigned to new neurons. This prevented an optimal clustering of the input data.

4.4 Learning Speed

The decrease of prediction quality that occure after updates of an ART2 network can be countered by occasionally relearning the whole data set. Figure 9 shows that compared to a SOM an ART2 network can be trained fairly quickly. The same test data and networks as in the previous experiments were used. The comparatively slow learning speed of the SOM results from the complex computation of neighbourhood relations using the Gauss bell. The learning speed can be increased by using a slightly less exact linear function. The duration for (re-)learning a single profile from our training set was on average 0.06 seconds with an ART2 network and between 0.5 and 1 second for a SOM.

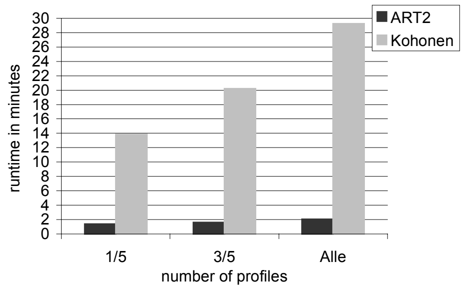


Fig. 9. Learning duration for 400, 1200 and 2000 profiles

4.5 Choice of Parameters

For most parameters of the neural networks standard values can be used. Two important parameters that have to be set are the number of learning steps (*epoch*) and the size of the network In the case of our ART2 networks ten learning steps have been enough, often the network stabilized even faster.

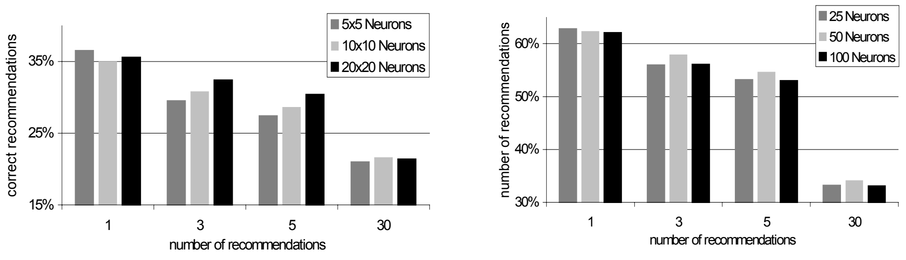


Fig. 10. Quality of results in relation to network size

For the SOM we also used ten learning steps. The map needs a global learning process and can then make more and more fine grained local weight adjustments

while decreasing the learning rate. This fine grained learning phase needs much longer and can't be completed after ten learning steps. We performed experiments with up to 100 steps without being able to observe any measurable improvements. The most important parameter is the network size which means the number of neurons in the competitive layer. In order to decrease run-time during profile evaluation the network size should be constant and as small as possible. But the quality of recommendations also depends on the number of neurons. Thus, depending on the number of relevant clusters in the profile set more or less neurons are necessary.
When using SOMs, too few neurons lead to an instability while too many neurons prevent a useful classification. The ART2 network only uses new neurons when a pattern does not lie within a cone of attentiveness. Thus the number of neurons in an ART2 network can never be too high. The number of neurons in an ART2 network is only limited for performance reasons. If the set maximum number of neurons has been reached and a new pattern does not lie within the attentiveness cone of a neuron then the cone radius is increased automatically until the pattern can be assigned to a neuron. Figure 10 (left) shows that in the case of a SOM in the range of 25 to 400 neurons only small differences in quality can be observed with our test data. Figure 10 (right) shows that there is also no significant difference when increasing the number of neurons in an ART2 network from 25 to 100.

4.6 Influence of Training Set

To test the influence of the population of the training set on the quality of results obtained with the three collaborative filtering methods, we performed two more experiments using the Eurovictor II data set. Since that data set is smaller and very sparsely populated we chose 16 neurons as the network size for both the ART2 network and the SOM and only computed 5 recommendations per profile. The Eurovictor II data set contains 770 profiles, only 91 with 5 or more entries.

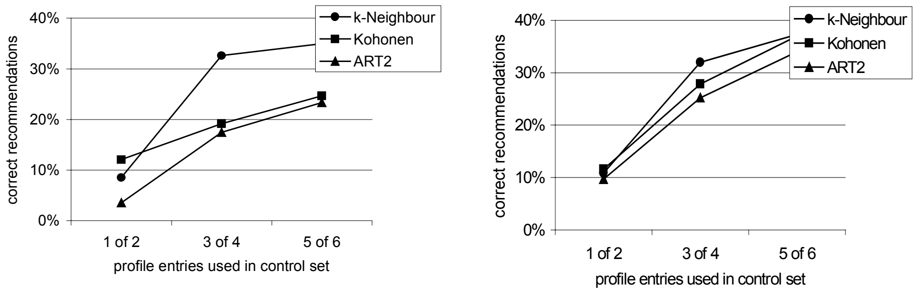


Fig. 11. Results with E-Victor data set: all profiles (left), only profiles with 5+ entries (right)

Figure 11 (left) shows results using all profiles as training set and figure 11 (right) visualizes the results that were obtained only using profiles with 5 or more entries. In both cases three different query sets have been created: These are sets that contain profiles with two or more, four or more and six or more entries respectively and use one, three or five of those entries for the control set. The result shows that the neural networks' performance is affected negatively by the sparsity of the training data. With very sparse data our memory based implementation produced better quality results.

6 Conclusion

We have described two model based approaches to collaborative filtering based on Self-Adaptive Maps and ART2 networks. When evaluating them with test data from two real life applications we found out that ART2 networks produce even better results than a widely used memory based approach if the profile vectors in the training data are not sparsely populated. The model based algorithms needed by several orders of magnitude less memory and less computations to produce recommendations. The ART2 network proofed to be adaptive but the quality of predictions still degraded slowly after more and more changed profile vectors were (re-)learned suggesting that it is necessary to relearn the complete data set once in a while.
In the near future we want to include other memory based algorithms into our comparison, such as suggested in [10] or [6]. We also plan to explore wether it is possible to improve results further by combining several methods, including the neural network models explored in this contribution.

References

[1] AMAZON.COM, Amazon Homepage: http://www.amazon.com (accessed: May 2000)

[2] R. ARMSTRONG, D. FREITAG, T. JOACHIMS, T. MITCHELL, WebWatcher: a learning apprentice for the World Wide Web, in: AAAI Spring Symposium, Stanford, U.S., pp. 6-12.

[3] W. BENN, O. GOERLITZ, Semantic Navigation Maps for Information Agents, in: Proceedings of the 2nd Intl. Workshop on Cooperative Agents, Lecture Notes in Artificial Intelligence 1435, Springer-Verlag, Heidelberg, 1998,

[4] D. BILLSUS, M. J. PAZZANI, Learning Collaborative Information Filters, in: Proceedings of the 15th Intl. Conference on Machine Learning, Morgan Kaufmann Publishers, 1998

[5] H.BRAUN, J.FEULNER, R.MALAKA, Praktikum Neuronale Netze, Springer Verlag, 1996

[6] J. S. BREESE, D. HECKERMAN, C. KADIE, Empirical Analysis of Predictive Algorithms for Collaborative Filtering, in: Proceedings of the 14th Annual Conference on Uncertainty in Artificial Intelligence, pp. 43-52, 1998

[7] S. BRIN, L. PAGE, The Anatomy of a Large-Scale Hypertextual Web Search Engine, in: Proceedings of the 7th World Wide Web Conference, 1998

[8] G. A. CARPENTER, Neural Network Models for Pattern Recognition and Associative Memory, 1989, in: G. A. Carpenter and S. Grossberg, Pattern Recognition by Self-Organizing Neural Networks, MIT Press, 1991

[9] G. A. CARPENTER, S. GROSSBERG, ART2: Self-Organization of Stable Category Recognition Codes for Analog Input Patterns, 1987, in: G. A. Carpenter and S. Grossberg, Pattern Recognition by Self-Organizing Neural Networks, MIT Press, 1991

[10] M. K. CONDLIFF ET AL., Bayesian Mixed-Effects Models for Recommender Systems, in: Proceedings of the ACM SIGIR Workshop on Recommender Systems: Algorithms and Evaluation, 22nd Intl. Conf. on Research and Development in Information Retrieval, 1999

[11] L. FAUSETT, Fundamentals of Neural Networks, Prentice Hall International Inc., 1994

[12] M. GAEDKE, C. SEGOR, H.-W. GELLERSEN, WCML: Paving the Way for Reuse in Object-Oriented Web Engineering, ACM Symposium on Applied Computing, Italy, 2000

[13] G. GRAEF, Adaptation of Web-Applications Based on Automated User Behaviour Analysis, 2nd Annual Conference on World Wide Web Applications, Johannesburg, 2000

[14] G. GRAEF, M. GAEDKE, Construction of Adaptive Web-Applications from Reusable Components, in: Lecture Notes on Computer Science, 1st Conference on Electronic Commerce and Web Technology, Springer-Verlag Heidelberg, 2000

[15] D. K. HAWES, Information literacy in the business schools, in: Sep/Oct Journal of Education in Business, 70, pp 54-62, 1994

[16] T. KOHONEN, Self-Organizing Maps, Springer-Verlag, Heidelberg, 1997

[17] J. A. KONSTAN ET AL., Applying Collaborative Filtering to Usenet News, in: Communications of the ACM, Vol. 40, No. 3, pp. 77-87, 1997

[18] P. MCJONES, EachMovie collaborative filtering data set, DEC Systems Research Center, http://www.research.compaq.com/SRC/eachmovie/.

[19] D. M. NICHOLS, Implicit Rating and Filtering, in: Proceedings of the Fifth DELOS Workshop on Filtering and Collaborative Filtering, Budapest, 1997

[20] R. ROJAS, Theorie der Neuronalen Netze, Springer Verlag, Heidelberg, 1993

[21] U. SHARDANAND, P. MAES, Social information filtering: algorithms for automating "word of mouth", in: Human factors in computing systems (CHI'95), Denver, USA, 210-217

[22] A. M. A. WASFI, Collecting User Access Patterns for Building User Profiles and Collaborative Filtering,in: Proceedings of the International Conference on Intelligent User Interfaces, pp. 57-64, 1999.