A Recommender System using Principal Component Analysis

Article · July 2008

Source: OAI

4 authors, including:

Manolis G. Vozalis
University of Macedonia
30 PUBLICATIONS **442** CITATIONS

SEE PROFILE

Konstantinos G. Margaritis
University of Macedonia
411 PUBLICATIONS **1,615** CITATIONS

SEE PROFILE

A Recommender System using Principal Component Analysis

Manolis G. Vozalis and Konstantinos G. Margaritis

Parallel Distributed Processing Laboratory, Department of Applied Informatics,
Unversity of Macedonia, Egnatia 156, 54006, Thessaloniki, Greece,
{mans,kmarg@uom.gr}

Abstract

In this paper we examine the use of a mathematical procedure, called Principal Component Analysis, in Recommender Systems. The resulting filtering algorithm applies PCA on user ratings and demographic data, aiming to improve various aspects of the recommendation process. After a brief introduction to PCA, we provide a discussion of the proposed PCA-Demog algorithm, along with possible ways of combining it with different sources of filtering data. The experimental part of this work tests distinct parameterizations for PCA-Demog, identifying those with the best performance. Finally, the paper compares their results with those achieved by other filtering approaches, and draws interesting conclusions.

Keywords: Recommender Systems, Collaborative Filtering, Personalization, Principal Component Analysis (PCA)

1. Introduction

Recommender Systems were introduced as a computer-based intelligent technique, designed to assist people with the problem of information and product overload. Their purpose is to generate efficient personalized solutions in e-business domains, benefiting both the customer and the merchant.

There are two basic entities featured in Recommender Systems: the *user* and the *item*. Users who utilize the Recommender System, are required to state their opinion about past items. The purpose of the Recommender System is to generate suggestions about new items for these particular users. It achieves that by applying the selected filtering algorithm on the input provided, which isusually expressed in the from of user ratings on items.

Recommender Systems suffer from a number of fundamental problems that reduce the quality of generated predictions. Such problems are: *sparsity*, *scalability*, and *synonymy*. A number of solutions have been introduced, intending to solve these problems [Melville et al. (2001), Claypool et al. (1999), Sarwar et al (2000)]. We are specifically interested in mathematical procedures that successfully tackle the

aforementioned problems by locating effective ways to reduce the dimensionality of the initial data. Singular Value Decomposition (SVD) is such a technique. Therefore, a number of researchers [Billsus et al. (1998), Sarwar (2001), Vozalis et al (2006a)] suggested its utilization in co-operation with standard filtering methods, based on SVD's ability to provide the best low-rank approximation of the original data set.

In this paper we will focus on an alternative approach, called *Principal Component Analysis* (PCA). Similarly to SVD, PCA can facilitate dimensionality reduction and lead to faster computation of recommendations. It is a multivariate mathematical procedure, which transforms a set of possibly correlated variables into a new set of uncorrelated variables. Its members are called principal components and correspond to linear combinations of the original variables. They are normally ordered in reduced variability, meaning that the first principal component is a combination of the original variables with the greatest amount of variation, and each succeeding principal component represents the next largest amount of variation [Jolliffe (2002)].

PCA-Demog, the algorithm to be presented in the following paragraphs, constructs a novel representation of the original data, which has the form of data tuples that incorporate both user ratings on items and user/item demographic data, and applies PCA on them. The resulting data tuples, which can be viewed as projections in a reduced space, are then utilized for prediction generation.

The subsequent sections are structured as follows: Section 2 details past research work that involves the use of Principal Component Analysis. In Section 3 we provide an analysis of our experimental methodology, regarding the data set and evaluation metrics. Section 4 gives a step-by-step description of the proposed filtering method. The experimental work, which follows in Section 5, closes by comparing our technique with past filtering algorithms. We conclude with a summary and possible directions for future work.

2. Related Work

Eigentaste [Goldberg et al. (2001)] is a collaborative filtering algorithm that combines universal queries, clustering and the application of PCA, in a recommendation procedure that is broken in offline and online stages.

Shared Wisdom through the Amalgamation of Many Interpretations (SWAMI) [Fisher et al. (2000)] is a Java-based framework that supports the building and studying of collaborative filtering systems. The authors were able to reach useful conclusions regarding their data set, by representing it with the help of a 6 dimensional axis set - as defined by the first 6 eigenvectors which were generated by the application of PCA.

Lam (2004) presents an algorithm called CLAM, which can be described as a system that performs Collaborative Filtering with the help of Linear Associative Memory.

The author utilizes Principal Component Analysis in order to reduce the dimensions of his model. After the application of PCA, only the *k* most significant eigenvectors are retained. The newly generated model should then be used for prediction generation.

In his Ph.D. dissertation [Yu (2004)], Kai Yu describes Generalized Probabilistic PCA (GPPCA), a probabilistic latent-variable model for mixed types of data, which allows a unified modelling of continuous, binary and categorical observations. The proposed model can be utilized not only for dimensionality reduction, but also for feature fusion, detection of latent relevant information, and support of fast and accurate retrieval/filtering.

Schein et al. (2003) utilize the MovieLens collaborative filtering data set in combination with Principal component Analysis, not intending to generate predictions, but merely to evaluate their proposed algorithm, which is a variation of logistic PCA. Their presented experimental results show that logistic PCA performs better than linear PCA in the reconstruction of binary data.

3. Experimental Methodology

3.1 MovieLens: A GroupLens Data Set

For the execution of our subsequent experiments we utilized the data publicly available from the GroupLens movie recommender system. The MovieLens data set, used by several researchers [Schein et al. (2002), Ujjin et al. (2003), Herlocker et al. (2004)], consists of 100.000 ratings, which were assigned by 943 users on 1682 movies. Starting from the initial data set, five distinct splits of training and test data were generated, where 80% of the original set was included in the training and 20% of it was included in the test data. The test sets in all cases were disjoint. An actual sample from the GroupLens data set can be found in Vozalis et al. (2006b). Except for ratings awarded by users on items, the MovieLens data set includes information regarding specifically the users and the items. This information falls under the "demographic data" category. It will be used extensively, as part of the algorithmic approaches, which will be proposed in the following sections. Actual samples from the user and item-related information can be found in Vozalis et al. (2006b).

3.2 The Evaluation Metric

Several techniques have been used to evaluate Recommender Systems. An analysis of such metrics can be found in Herlocker et al. (2004). The choice among them should be based on the selected user tasks and the nature of the data sets.

Bearing in mind the aforementioned criteria, we proceeded in the selection of the initial evaluation metric for our experiments. That metric was *Mean Absolute Error (MAE)* [Shardanand et al. (1995)].

$$MAE = \frac{\sum_{i=1}^{k} pr_i - r_i}{k} \quad \textbf{(1)}$$

It is a statistical accuracy metric which measures the deviation of predictions, pr_i, generated by the Recommender System, from the true rating values, r_i, as they were specified by the user, for k {*user*, *item*} pairs. MAE is measured only for these items, for which a user has expressed his opinion.

4. PCA-Demog: Applying PCA for Demographically Enhanced Prediction Generation

In the following paragraphs we will present a first approach describing how Principal Component Analysis can be combined with user ratings on items and existing user/item demographics, for the construction of a novel Recommender System.

1. Data Formation

For Principal Component Analysis to be applied, the data needs to be structured accordingly. We can imagine $m \times n$ tuples, conforming to one of three forms. All possible tuple forms are included in Table 1.

Table 1*. Possible tuple forms for the data representation of PCA-Demog*

tuple #	**tuple structure**
a	{<rating r_{ij}> <user u_i demogs>}
b	{<rating r_{ij}> <item i_j demogs>}
c	{<rating r_{ij}> <user u_i demogs> <item i_j demogs>}

In each of these cases, <rating r_{ij}> would be the rating assigned by user u_i on item i_j, according to the selected rating scale, <user u_i demogs> correspond to a set of binary values, representing the demographic features for user u_i, and <item i_j demogs> correspond to a set of binary values, representing the demographic features for item i_j. Still, before any ratings should be included in these tuples, we need to normalize them. That is necessary especially in the case of filtering systems, where the data is very sparse.

In the MovieLens data set, which was utilized in our case, most users have rated only a small subset of the total possible items. This leads to a representation equivalent to an $m \times n = 943 \times 1682$ user-item matrix, with numerous empty slots, denoting missing ratings. The normalization procedure first fills each empty slot with the

corresponding column average, which essentially is equal to the mean rating of the item in mind. Then, the row average, essentially the mean rating of the corresponding user, is subtracted from all the slots. The resulting normalized values can be used as part of the PCA tuples.

Regarding the user and item demographic data, which are extracted from the MovieLens data set, we had to represent them appropriately in order to incorporate them in the same data tuples. For that purpose, we adopted the use of a 27-feature user demographic vector and a 19-feature item demographic vector, structured according to what was discussed in Vozalis et al. (2006b). The feature count of the resulting PCA tuples is analyzed in Table 2. The indices (*tuple #*) that describe each possible PCA tuple in this table, are taken from Table 1.

Table 2. *The feature count of PCA data tuples*

tuple #	rating	u demogs	i demogs	feature count
a	√	√		28
b	√		√	20
c	√	√	√	47

† 1 feature slot for rating, 27 feature slots for user demographics
19 feature slots for item demographics

As mentioned previously, and regardless of the selected tuples structure, the PCA representation of the MovieLens data set requires the construction of 1.586.126 distinct tuples, one for each possible pair of the 943 users and 1682 items. Consequently, the dimensions of the resulting data representation, *DataNorm*, will vary in the fashion described in Table 3, depending on the size of the selected tuple structure - as defined by the corresponding feature count (Table 2).

Table 3. *The dimensions of DataNorm, depending on tuple structure*

tuple #	DataNorm size
a	1.586.126 × 28
b	1.586.126 ×20
c	1.586.126 ×47

2. Covariance Matrix Calculation

Calculate the Covariance Matrix of the data. It is important that the data is structured according to what was discussed in the previous step. Depending on the selected data representation (Table 1) and the corresponding tuple size (Table 2), and assuming the use of the MovieLens data set, the dimensions of the covariance matrix will vary according to what is included in Table 4.

Table 4*. The dimensions of the Covariance Matrix depending on tuple structure*

tuple #	**covariance matrix size**
a	28×28
b	20×20
c	47×47

3. Calculation of Eigenvectors/Eigenvalues and Formation of a Feature Vector

The Covariance Matrix, computed in the previous step, is by default square. Thus, we can use it in order to calculate the corresponding eigenvectors and eigenvalues. The dimensions of the Covariance Matrix define the number of the resulting eigenvectors and eigenvalues. Specifically, a $c \times c$ Covariance Matrix will lead to c eigenvalues and c eigenvectors, with the latter representing lines in the original axis system, which appear to be able to characterize the data.

Once all eigenvectors and eigenvalues are derived from the Covariance Matrix, we can sort the eigenvectors in decreasing order, according to their eigenvalues. This arrangement will lead to a list of components of the original data set, ordered by their significance. Obviously, the eigenvector with the highest eigenvalue, which corresponds to the principle component of the data set, will capture the most important relationship between the data dimensions.

We can now construct the feature vector, which essentially will assume the form of a matrix of eigenvectors. Specifically, out of the c original eigenvectors, we decide to keep only r eigenvectors ($eigvect_1$, $eigvect_2$, ..., $eigvect_r$, where normally $r < c$) that correspond to the highest eigenvalues. The resulting *Feature Vector* will be of the following form:

$$FeatureVector = (eigvect_1\ eigvect_2 \ldots\ eigvect_r) \quad \textbf{(2)}$$

Since the size of the eigenvectors is defined by the dimensions of the Covariance Matrix (Step 2), which in turn is determined by the selected tuple representation (Step 1), the size of the tuple representation, <*tuple size*>, which was adopted in the Data Formation step of the procedure, will define the dimensions of the resulting feature vector. Specifically, the feature vector will be of size {<*tuple size*> × r}, with <*tuple size*> taking one of three possible values {28,20,47} for reasons discussed in Step 1.

4. Construction of the new Data Set

After selecting the most significant components - having the form of the eigenvectors with the highest eigenvalues - for the construction of the Feature Vector, we can now derive our new data set, *NewData*, in terms of these components:

$$NewDataT = FeatureVector^T \times DataNorm^T \quad \textbf{(3)}$$

The dimensions of *FeatureVector* are {<*tuple size*> × *r*} and thus, its transpose, $FeatureVector^T$, would be {*r* ×<*tuple size*>}. The dimensions of *DataNorm*, as discussed in Step 1, are {(*m* × *n*) × <*tuple size*>}, for *m* users and *n* items. As a result, $DataNorm^T$ would be {<*tuple size*> × (*m* × *n*)}. Consequently, the dimensions of the new data set, *NewDataT*, would be {*r* × (*m* × *n*)}. Its transpose of size {(*m* × *n*) × *r*} can be considered as including *m* × *n* tuples, one for each <user, item> pair, in the reduced *r*-dimensional space, which was defined by the *r* most significant components retained in Step 3 of the procedure.

5. Return to the Old Data

Bearing in mind that by keeping only *r* out of the total *c* eigenvectors from the original data set, part of the initially available information will be lost, our course of action in order to get back to the old data involves transforming Equation 2 as follows:

$$DataNorm^T = (FeatureVector^T)^T \times NewDataT = FeatureVector \times NewDataT \quad \textbf{(4)}$$

The final step in getting back to the "reduced" original data, requires adding the mean of the original ratings, which were subtracted during the normalization process.

$$DataReducedOriginal^T = FeatureVector \times NewDataT + OriginalRatingsMean \quad \textbf{(5)}$$

5. Experimental Evaluation of the PCA-Demog Filtering Algorithm

In this section we report the results from our initial experiments regarding the application of the PCA-Demog filtering algorithm on the MovieLens data set. First, we present the results from applying PCA on data tuples that combine user ratings on items and user demographics. Then, we present the results from applying PCA on data tuples that combine user ratings on items and item demographics. Finally, we present the results from applying PCA on data tuples that combine user ratings on items, user demographics and item demographics. In each case, we experimented with different values for *r*, corresponding to varying numbers of eigenvectors retained out of the *c* eigenvectors of the initial data set. Our intention was to evaluate how the performance of the recommender system would be affected, when ignoring components of lesser importance, based on their eigenvalues.

5.1 Applying PCA-Demog on tuples including only user demographics

In this set of experiments we utilized data tuples of the following form:

{<rating r_{ij}> <user u_i demographics>}

Obviously, the lone varying parameter was the value of r, representing the number of eigenvectors kept, out of those originally available after the application of PCA on the initial data set.

Figure 1 displays the Mean Absolute Errors (MAEs) which were obtained by applying the PCA-Demog algorithm on the MovieLens data set in combination with user demographics. For the experimental runs, 4 distinct values of r, *r={2,5,10,20}* were utilized. The depicted results were averaged over all five MovieLens data splits. The behavior displayed by line *pca-u* was not unexpected. It shows that as the value of *r* was increasing from 2 to 20, and more components were taken into account, the accuracy of the system was improving.

5.2 Applying PCA-Demog on tuples including only item demographics

For this set of experiments we used only data tuples of the following form:

{<rating r_{ij}> <item i_j demographics>}

Based on these tuples, it is clear that the single parameter, which varied among different runs was the value of *r*, representing the number of eigenvectors kept, out of those originally available after the application of PCA on the initial data set.

Line *pca-i* from Figure 1 displays the Mean Absolute Errors (MAEs), which were obtained by applying the PCA-Demog algorithm on the MovieLens data set in combination with item demographics. For the experimental runs, 4 distinct values of *r*, *r={2,5,10,20}*, were utilized. The depicted results were averaged over all five MovieLens data splits. An interesting observation regarding these experimental runs is that better accuracy values where obtained when keeping less eigenvectors. Specifically, the lowest MAE value was observed for *r*=2, meaning that only 2 components, out of the 20 initially available, were able to lead to an axis system of reduced dimensions that captured the relations between the original data with the least loss of information. This result comes in contrast with our observations from the previous section.

5.3 Applying PCA-Demog on tuples including both user & item demographics

For the last set of experiments we used only data tuples of the following forms:

{<rating r_{ij}> <user u_i demographics> <item i_j demographics>}

We varied a single parameter between our experimental runs. That was the value of *r*, corresponding to the number of eigenvectors that we retained, after the application of PCA on the aforementioned tuples.

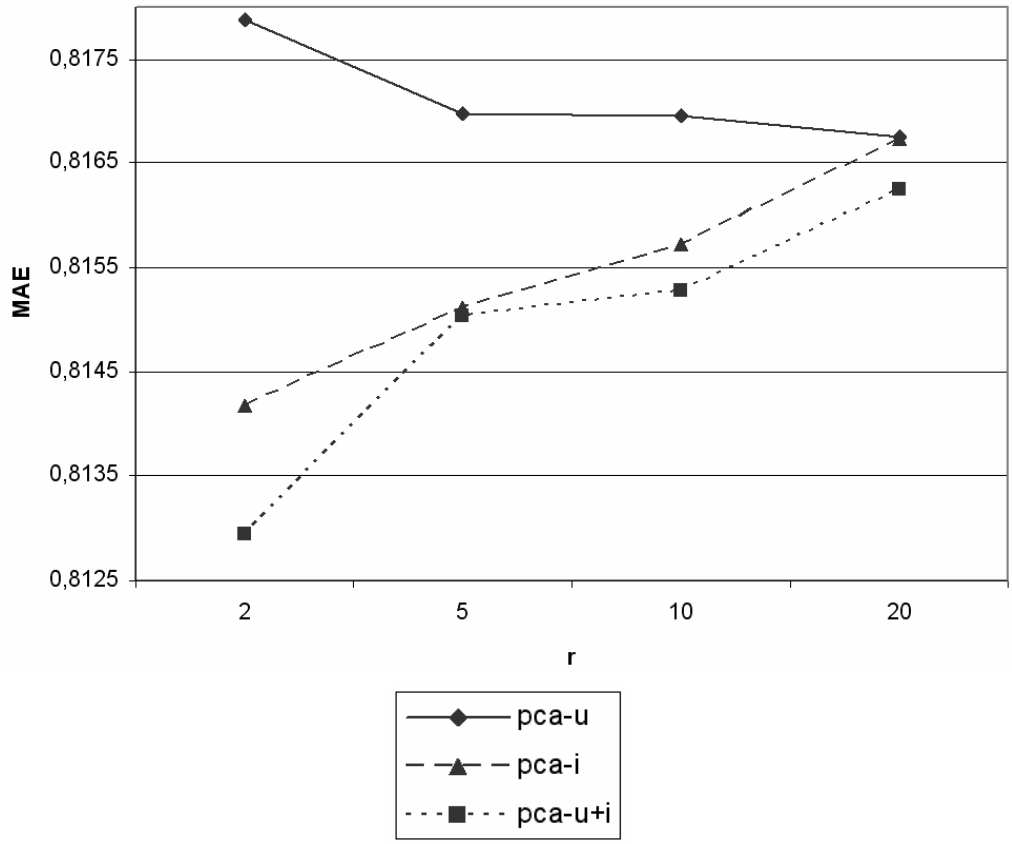

Figure 1*. Three implementations of PCA-Demog utilizing different demographic data*

Line *pca-u+i* from Figure 1 depicts the Mean Absolute Errors (MAEs), which were generated by applying the PCA-Demog algorithm on tuples constructed from the MovieLens data set, when combined with *both* user & item demographics. 4 distinct values were utilized for *r*, *r={2,5,10,20}*. The results, as displayed in the figure, were averaged over the five MovieLens data splits. Based on the line *pca-u+i*, we can claim that its behavior is resembling the results generated when applying PCA-Demog only on item demographics (Section 5.2), rather than those observed when applying PCA-Demog on user demographics (Section 5.1): it attains its best overall accuracy when *r=2*, and gets continuously worse as more eigenvectors are added, towards the construction of the new axis system of reduced dimensions.

5.4 Comparing PCA-Demog with past Collaborative Filtering approaches

After experimenting with different variations of PCA-Demog (*PCA-Demog with user demographics*, *PCA-Demog with item demographics* and *PCA-Demog with user+item demographics*), which we tested for a series of possible values for r, we have settled to their optimal settings in each of these cases.

At this point, it was in our intentions to contrast the best results achieved by each of our distinct PCA-Demog implementations against each other. Furthermore, we decided to compare PCA-Demog with two past filtering approaches, which were User-based Collaborative Filtering (UbCF) [Herlocker et al. (1999)] and Item-based Collaborative Filtering (IbCF) [Sarwar et al. (2001)].

Table 5 averages the best MAEs achieved by each of the 5 filtering approaches that participated in our comparison, for each of the five available data splits, and holds the resulting values. The filtering approaches which participated in the overall comparison were: PCA-Demog with user demographics (*pca-u*), PCA-Demog with item demographics (*pca-i*), PCA-Demog with user & item demographics (*pca-u+i*), User-based CF (*u-b*), and Item-based CF (*i-b*). Regarding the last two methods, the included results were taken from Vozalis et al. (2003).

Table 5. *Best MAE values of PCA-Demog against past CF methods*

	pca-u	**pca-i**	**pca-u+i**	**u-b**	**i-b**
MAE	0,8167	0,8141	0,8127	0,7809	0,8284

Our final observations, based on both the results of Table 5 and the benefits of selecting PCA for dimensionality reduction, are the following:

- All PCA-Demog implementations gave accuracy values that lie between the accuracy values of UbCF and IbCF. UbCF appeared to generate the most accurate predictions, while IbCF the least accurate ones.
- Among PCA-Demog implementations, the one involving richer demographic information, i.e. both user & item demographics, generated the lowest error values. Between the remaining implementations, *pca-i* proved to be more accurate than *pca-u*. The latter result was not totally expected, since past experiments have showed that methods basing their predictions on user ratings or user demographics outperform methods basing their predictions on item ratings or item demographics [Vozalis et al. (2006b)].
- The application of PCA, and the dimensionality reduction achieved by it, may map similar users or items to the same eigenvector of the reduced axis system, thus alleviating the problem of synonymy.

- Before PCA can be executed, the required preprocessing phase fills all the empty entries of the user-item matrix. By execution's end, all $m \times n$ user-item couples are represented in the reduced r-dimensional space, the difference being that in the reduced space *all* users have rated *all* items. As a result, the sparsity problem is solved, and the achieved coverage of PCA-Demog is always equal to 100%.

6. Conclusions and Future Work

Past filtering algorithms have employed Singular Value Decomposition as a means of tackling several issues related to the use of recommender systems. Principal Component Analysis is a mathematical procedure that shares common properties with SVD. In his paper we presented the initial work, which was conducted as part of our wider effort to study how PCA can be employed in order to assist the recommendation process.

PCA-Demog can be described as an algorithm closer to LSI [Deerwester et al. (1990)] than to collaborative filtering approaches. It adopts a data representation that incorporates both user ratings on items and user/item demographics, and employs PCA to reduce its dimensions. Based on the presented experimental results, we can state that our initial proposal regarding the incorporation of PCA into the recommendation process did lead to satisfactory predictions. Its achieved accuracy improved on the predictions generated by IbCF, but was worse than the accuracy of UbCF. Still, we have to see the whole picture, and, rather than focus solely on the accuracy results, take also into account the additional benefits brought by the use of PCA, such as the alleviation of the sparsity and the synonymy problems.

Future work may involve the utilization of PCA in combination with existing collaborative filtering algorithms. Furthermore, it is in our intentions to experiment with Random Projection. We view it as another method, which can possibly enhance the filtering procedure, by assisting in dimensionality reduction.

References

Billsus, D., Pazzani, M.J. (1998), *Learning collaborative information filters*, in Proc. of the15th International Conference on Machine Learning, Madison, WI.

Claypool, M., Gokhale, A., Miranda, T., Murnikov, P., Netes, D., Sartin, M. (1999), *Combining content-based and collaborative filters in an online newspaper*, in ACM SIGIR Workshop on Recommender Systems-Implementation and Evaluation, Berkeley, CA.

Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R. (1990), *Indexing by latent semantic analysis*. Journal of the American Society for Information Science, vol. 41, pp. 391-407.

Fisher, D., Hildrum, K., Hong, J., Newman, M., Thomas, M., Vuduc, R. (2000), *Swami: a framework for collaborative filtering algorithm development and evaluation*, in Proc of the 23rd Annual ACM Conference on Research and Development in Information Retrieval, Athens, Greece.

Goldberg, K., Roeder, T., Gupta, D., Perkins, C. (2001), *Eigentaste: A constant time collaborative filtering algorithm*. Information Retrieval Journal, vol. 4, pp. 133-151.

Herlocker, J., Konstan, J.A., Borchers, A., Riedl, J.T. (1999), *An algorithmic framework for performing collaborative filtering*, in the 1999 Conference on Research and Development in Information Retrieval.

Herlocker, J.L., Konstan, J.A., Terveen, L.G., Riedl, J.T. (2004), *Evaluating collaborative filtering recommender systems*. ACM Transactions on Information Systems, vol. 22, pp. 5-53.

Jolliffe T.I. (2002). *Principal Component Analysis*, Springer, ISBN 0-3879-5442-2.

Lam, C. (2004), *Collaborative filtering using associative neural memory*, in Proc. AAAI Workshop on Semantic Web Personalization, San Jose, CA.

Melville, P., Mooney, R.J., Nagarajan, R. (2001), *Content-boosted collaborative filtering*, in Proc. of the Eighteenth National Conference on Artificial Intelligence (AAAI-2002), Edmonton, Canada.

Sarwar, B.M., Karypis, G., Konstan, J.A., Riedl, J.T. (2000), *Analysis of recommendation algorithms for e-commerce*, in Proc. of the 2nd ACM Conference on Electronic Commerce.

Sarwar, B.M., Karypis, G., Konstan, J.A., Riedl, J.T. (2001), *Item-based collaborative filtering recommendation algorithms*, in the 10th International World Wide Web Conference (WWW10), Hong Kong.

Sarwar, B.M. (2001), *Sparsity, Scalability, and Distribution in Recommender Systems*, Ph.D. thesis, University of Minnesota.

Shardanand, U., Maes, P. (1995), *Social information filtering: Algorithms for automating 'word of mouth'*, in Proc. of Computer Human Interaction.

Schein, A.I., Popescul, A., Ungar, L.H., Pennock, D.M. (2002), *Methods and metrics for cold-start recommendations*, in Proc. ACM SIGIR-2002, Tampere, Finland.

Schein, A.I., Saul, L.K., Ungar, L.H. (2003), *A generalized linear model for principal component analysis of binary data*, in Proc. of the Ninth International Workshop on Artificial Intelligence and Statistics, Kew West, FL.

Ujjin, S., Bentley, P.J. (2003), *Particle swarm optimization recommender system*, in Proc. of the IEEE Swarm Intelligence Sympoisum, Indianapolis.

Vozalis, E.G., Margaritis, K.G. (2003), *Recommender systems: An experimental comparison of two filtering algorithms*, in Proc. of the 9th Panhellenic Conference in Informatics - PCI 2003.

Vozalis, M.G., Margaritis, K.G. (2006a), *Applying SVD on generalized item-based filtering*. Special Issue on Web-based Recommender Systems of the International Journal of Computer Science and Applications, vol. 3, pp. 27-51.

Vozalis, M., Margaritis, K.G. (2006b), *On the enhancement of collaborative filtering by demographic data*. Web Intelligence and Agent Systems, An International Journal, vol. 4, pp. 117-138.

Yu, K. (2004), *Statistical Learning Approaches to Information Filtering*, Ph.D. thesis, Ludwig-Maximilians-Universitaet Muenchen.