Support Vector Machines For Collaborative Filtering

Zhonghang Xia
Department of Computer Science
Western Kentucky University
1906 Collee Heights Blvd.
Bowling Green, KY 42101, U.S.A.
Zhonghang.xia@wku.edu

Yulin Dong
Department of Applied Mathematics
Dalian University of Technology
Dalian, Liaoning, 116023, China
dyl@student.dlut.edu.cn

Guangming Xing
Department of Computer Science
Western Kentucky University
1906 Collee Heights Blvd.
Bowling Green, KY 42101, U.S.A.
Guangming.xing@wku.edu

ABSTRACT

Support Vector Machines (SVMs) have successfully shown efficiencies in many areas such as text categorization. Although recommendation systems share many similarities with text categorization, the performance of SVMs in recommendation systems is not acceptable due to the sparsity of the user-item matrix. In this paper, we propose a heuristic method to improve the predictive accuracy of SVMs by repeatedly correcting the missing values in the user-item matrix. The performance comparison to other algorithms has been conducted. The experimental studies show that the accurate rates of our heuristic method are the highest.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Algorithms, Measurement, Experimentation

Keywords

Recommendation systems, Support Vector Machines, Machine Learning, Collaborative Filtering

1. INTRODUCTION

Rapid advances of the Internet and the World Wide Web have greatly facilitated the growth of online applications, such as distance learning, digital library, news on demand, e-commerce, etc. As the availability of these applications continues to increase, users face the tremendous work of retrieving interesting information that matches their preferences. Consequently, users are spending more and more time to search their desired targets and the searching has also drastically increased the consumption of system resources. According to workload analysis of HPLabs Media server [7], where 79% of video files belong to a long video group (longer than 30 minutes), 77%-79% of media sessions last less than 10 minute long. This implies that many users are not interested in watching the selected videos completely. They may spend a lot of time to search interesting videos. The demand for efficient and effective tools to help users find their desired targets is required.

Recommender systems provide automated methods for users to search for interesting items with respect to users' preferences. The underlying techniques used in current recommender systems can be classified into collaborative filtering (CF) and content-based filtering (CBF). CF algorithms exploit similarities among users or items based on users' feedbacks. CBF systems, on the other hand, recommend items of interest to the active user by exploiting content information of the items already rated. Typically, a profile is formed for a user individually by analyzing information regarding the content of items, such as desired actors/actresses, title, and description, etc. Additional items can be inferred from this profile. In general, the content is difficult to analyze, thus CF algorithms are more successful in a broad range of areas, including recommending movies [20], news [3] and research papers [11].

According to [5], CF algorithms can be categorized into memory-based and model-based algorithms. Memory-based [15] CF algorithms find neighbors for a new user (active user) and use neighbors' preferences to predict the unknown preferences of the active user. In contrast, model-based [5] CF algorithms first develop a model based on the historical data and then use the model to predict new preferences for users.

Current CF recommender algorithms suffer from a fundamental problem, called sparsity problem. Since the set of all possible available items in a system is very large, most users may have rated very few items, and, hence, it is difficult to find the active user's neighborhood with high similarity. As a result the accuracy of the recommendations may be poor.

A lot of researches have been launched to improve the quality of recommendation systems. Machine learning is a standard paradigm of predicting ratings and preferences for users' interests by casting the prediction problem as a classification problem. Compared to other machine learning methods, support vector machines (SVMs) can be analyzed theoretically using concepts from computational learning theory, and at the same time they have been successfully applied to many applications, such as text classification [8], image classification [19], and face recognition [14]. Some characteristics of recommender systems are shared by text categorization. For example, a text document is represented by a vector of word occurrences in the document, and, similarly, a user (corresponding to a document) can be represented by a vector using the user's ratings as its elements. The success of SVMs in text categorization naturally leads to its possible extension to recommender systems. However, the standard SVM classifier is not very successful [22] when it is applied in recommender systems due to the sparsity problem. A simple solution to the sparsity problem is default voting [5] which inserts default rating values for unrated items to increase the density of the user-item matrix. Normally, default voting uses neutral or negative preference value for those unrated items. This method, however, can mislead the classifiers in most cases.

In this paper, we address the sparsity problem by repeatedly estimating the missing ratings for the items which users have not rated. We first initialize these missing values with default values to provide enough training examples for learning machines, and, then, build classifiers based on these training examples. After the classifiers are obtained, those missing values are re-estimated. This procedure is repeated until the termination criterion is met. Our heuristic method is based on the smoothing SVM (SSVM) method [9]. We compare our heuristic method with item-based [22] and user-based [5] CF algorithms. The experimental studies show that our solution outperforms these algorithms.

The rest of the paper is organized as follows. Section 2 discusses user and item based algorithms. In Section 3, we describe some existing linear SVM classifiers. Our heuristic method is presented in Section 4. Experimental studies are presented in Section 5. Section 6 states the conclusion of the paper.

2. BASIC COLLABORATIVE FILTERING METHODS

Consider a recommendation system consisting of M users $U = \{u_1,...,u_M\}$ and N items $I = \{1,...,N\}$. There is a particular user u_a, called active user. The task of collaborative filtering is to predict the preference of the active user based on the opinions of a set of similar users. Each user u_j has given opinions on a set of items I_j and its opinion on item n is given as numeric rating x_{jn}. Note that I_j can be empty. To predict the preference of the active user, we need to estimate its rating on item $n \notin I_a$. Let A be a user-item matrix, where the value of *i-th* row and *j-th* column is x_{ij}. That is,

$$A = \begin{pmatrix} x_{11} & ? & ... & x_{1n} \\ ? & x_{22} & ... & x_{2n} \\ ... & & ... & ... \\ x_{m1} & ? & ... & x_{mn} \end{pmatrix}$$

Note that some elements in A are missing because the users have not rated the corresponding items. The predication task can now be treated as filling in those missing values [4].

2.1 Memory-Based Collaborative Filtering

In memory-based collaborative filtering, this estimation depends on the active user's mean rate $\bar{x}_a$ and ratings of its similar users (we also refer to them as u_a's neighbors). Based on the set of ratings by u_j, we can define its mean rating as $\bar{x}_a = \frac{1}{|I_a|}\sum_{n \in I_a} x_{an}$.

Usually a closer neighbor u_j to u_a, should contribute a larger weight to the estimation. The weight can be measured by the similarity between two users. A widespread measure is the Pearson correlation coefficient which was first introduced in [15]. The weight that u_j contributes to u_a is defined as

$$w(a,j) = \frac{\sum_{n \in I_a \cap I_j} (x_{an} - \bar{x}_a)(x_{jn} - \bar{x}_j)}{\sqrt{\sum_{n \in I_a \cap I_j} (x_{an} - \bar{x}_a)^2 \sum_{n \in I_a \cap I_j} (x_{jn} - \bar{x}_j)^2}}$$

Without loss of generality, we define $w(a,j) = 0$ if $I_a \cap I_j = \varnothing$. Once we have determined these weights, u_a's rating on item n, denoted by p_{an}, can be predicted by

$$p_{an} = \bar{x}_a + \frac{\sum_{j=1}^{M} w(a,j)(x_{jn} - \bar{x}_j)}{\sum_{j=1}^{M} w(a,j)}.$$

After the prediction on each item $n \notin I_a$, the system can recommend to the active user a list of items which are not in I_a and have top ratings.

2.2 Model-Based Collaborative Filtering

Since memory-based algorithms seriously suffer from the sparsity problem, model-based approaches have been studied to overcome this problem by learning a model for predicting ratings of unobserved items. These approaches include item-based [16], clustering [20], and classification [2], etc.

The item-based method assumes that users like to purchase items similar to those items they have selected in the history. To measure the similarity between two items, it first searches a set of users who have rated both of the two items, and, then, compute the similarity with some techniques. Let U_{in}={users who have rated both item i and n}.

The similarity of item i and item n is computed by

$$S_{in} = \frac{\sum_{u \in U_{in}} (x_{ui} - \bar{x}_u)(x_{un} - \bar{x}_u)}{\sqrt{\sum_{u \in U_{in}} (x_{ui} - \bar{x}_u)^2} \sqrt{\sum_{u \in U} (x_{un} - \bar{x}_u)^2}}$$

where $\bar{x}_u$ is the average of the u-th user's ratings, that is

$$\bar{x}_u = \frac{\sum_{n \in I_u} x_{un}}{|I_u|}$$

After the similarity computation, we can predict the preference of u_a on item n. It is given as follows:

$$P_{an} = \frac{\sum_{i \in S} (S_{ni} * x_{ai})}{\sum_{i \in S} |S_{ni}|},$$

where S is the set of items similar to item n.

3. LINEAR SVM CLASSIFIERS

In this paper, we recast collaborative filtering as a classification problem. Based on its numeric rating, an item or a user can be classified into a corresponding class. There are two ways to cast the problem [1]. One way is to treat every item as a separate classification problem. Given an item n, one can build a classifier to predict which class the active user belongs to. Every user u_j is represented as a vector in the feature space by using u_j's ratings on items other than n. A more common way to cast the classification problem is to treat every user as a separate problem [4]. One can build a classifier for the active user u_a by using items as training instances. To be specific, training instance n is represented as a feature vector x_n in which elements are ratings provided by other users. Without loss of generality, we consider the first user u_1 as the active user and u_1 has rated the first l items, that is, $I_1 = \{1,...,l\}$. Then, the feature vector of item $n, 1 \le n \le l$, is $x_n = (x_{2n}, x_{3n}, ..., x_{Mn})^T$ and its class label y_n is rating x_{1n}. We need to predict labels for all other feature vectors $x_n, l+1 \le n \le N$. For simplicity, we classify all items into two classes, for example, *like* and *dislike*. The class labels are denoted by +1 and -1, respectively. For multi-class problems, we can use the on-against-rest scheme.

A general notion of the above classification problem can be described as follows [6][13]. Given a set of training data $(x_1,y_1),(x_2,y_2),...,(x_l,y_l) \in R^n \times \{-1,1\}$, all drawn i.i.d. from an unknown distribution $P(x,y)$. The goal is to estimate a function $f: R^n \to \{-1,1\}, x \mapsto f(x)$ such that it will correctly classify unseen data (x,y). Errors in prediction will be penalized according to a loss function, i.e.

$$c(y,f(x)) = \begin{cases} 1 & f(x) \ne y \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

Hence, the best prediction function is the one minimizing the expected error

$$R(f) = \int c(y,f(x))dP(x,y).$$

However, distribution $P(x,y)$ is unknown. In this case, empirical error is defined (based on the observations):

$$R_{emp}(f) = \frac{1}{l}\sum_{i=1}^{l} c(y_i, f(x_i)) \quad (2)$$

One might think a function f minimizing $R_{emp}(f)$ would also minimizes $R(f)$. Unfortunately, this is not correct. For example, one can define a function $f(x_n) = y_n$, for all $1 \le n \le l$. Then, one has $R_{emp}(f) = 0$, but the prediction errors for unseen data can be very large, and, so, $R(f)$ is still large. This is the so-called overfitting problem. To overcome this problem, one can restrict the complexity of f [18]. This can be done by imposing a penalty on function f. For example, in [17], authors use a convex and continuous penalty $Q(f)$ and minimize $R(f) \triangleq R_{emp}(f) + \lambda Q(f)$, where λ is a regularization parameter $\lambda \ge 0$. Vapnik-Chervonenkis (VC) theory [21] can be used to characterize the upper bound of $\lambda Q(f)$. Let h denote VC dimension of a family of functions where f is chosen from. For any $0 \le \eta \le 1$, the following inequality

$$R(f) \le R_{emp}(f) + \sqrt{\frac{h(\ln\frac{2l}{h}+1) - \ln(\frac{\eta}{4})}{l}} \quad (3)$$

holds with probability of $1-\eta$. Note that the second term of (3) is a increasing function of h, denoted by $B(h)$.

According to (3), a simple way to minimize $R(f)$ is to keep $R_{emp}(f) = 0$ and minimize $B(h)$. Let us start with a simple case that training data can separated by a hyperplane. That is, there exists a linear function $g(x) = w^T x + b$ such that $f(x) = \text{sgn}(g(x))$, where $y_n g(x_n) \ge \rho$, for all $1 \le n \le l$, and $\rho > 0$ is defined as the margin between the two classes. Intuitively, it is desirable to seek classifiers that have large margins since one expect the classifiers can also predict well on unseen data. Actually, Vapnik [21] shows

$$h \le \min\{R^2/\rho^2, n\} + 1,$$

where R is the radius of the smallest ball enclosing the data [13]. Thus, a large margin ρ results in a tight upper bound for $R(f)$.

SVMs determine the hyperplane by maximizing the margin. One can solve the following optimization problem

$$\begin{aligned} &\max\ \rho \\ &s.t.\ \|w\| = 1, \\ &\quad y_n((w^T x_n) + b) \ge \rho, 1 \le n \le l \end{aligned} \quad (4)$$

After a standard transformation, (4) can be converted to the standard SVMs formulation

$$\begin{aligned} &\min\ \frac{1}{2}\|w\|^2 \\ &s.t.\ y_n((w^T x_n) + b) \ge 1, 1 \le n \le l \end{aligned} \quad (5)$$

However, in practice, a given data set may not be linearly separable. Suppose that two classes overlap due to some noise. A standard SVMs classifier obtained by solving (3) may perform poorly.

To deal with this problem, one can introduce the slack variables $\xi_n \ge 0, 1 \le n \le l$, in the constraints to allow some classification errors. Also, consider penalty for each misclassification, one can redefine the (1) as

$$c(y,f(x)) = \begin{cases} 1 - y_n(w^T x_n + b), & \text{if } y_n(w^T x_n + b) \ge 1 \\ 0 & \text{otherwise} \end{cases}$$

For x_n which satisfies $y_n(w^T x_n + b) \ge 1$, one has $\xi_n = 0$. Generally, R_{emp} can be written as $\frac{1}{l}\sum_{n=1}^{l}\xi_n$. A standard optimization problem is given as follows.

$$\begin{aligned} &\min\ \frac{1}{2}\|w\|^2 + \frac{1}{l}\sum_{n=1}^{l}\xi_n \\ &s.t.\ y_n((w^T x_n) + b) \ge 1 - \xi_n, 1 \le n \le l \\ &\quad \xi_n \ge 0, 1 \le n \le l \end{aligned} \quad (6)$$

Problem (6) is a quadratic programming, thus one can use a standard Lagrange multiplier technique to solve it [6]. However, the non-smoothness of cost function (1) makes the optimization problem more difficult.

To overcome this problem some smoothing methods can be used. In [22], Zhang replaced the cost function (1) with cost function as follows,

$$c(y, f(x)) = \begin{cases} (1 - y_n(w^T x_n + b))^2, & \text{if } y_n(w^T x_n + b) \geq 1 \\ 0 & \text{otherwise} \end{cases}$$

Then, the problem (6) is converted to

$$\begin{aligned} &\min \ \frac{1}{2} \| w \|^2 + \frac{1}{l} \sum_{n=1}^{l} \xi_n^2 \\ &s.t. \ y_n((w^T x_n) + b) \geq 1 - \xi_n, 1 \leq n \leq l \qquad (7) \\ &\quad \xi_n \geq 0, 1 \leq n \leq l \end{aligned}$$

Another smoothing method, called Smoothing SVM (SSVM), is introduced in [9]. With this method, one can extend problem (7) to (N+1) dimensional space. In other words, (w, b) is treated as a variable in R^{N+1}. Then, the standard format of a SVM problem can be converted into an equivalent unconstrained optimization problem as follows:

$$\min_{w,b} \frac{v}{2} \| \gamma - (A_1 w - \gamma b) \|^2 + \frac{1}{2}(w^T w + b^2) \ , \quad (8)$$

where $v > 0$ is a constant, γ denotes a column vector of ones, and $A_1 \in R^{(M-1) \times l}$ is a matrix of feature vectors.

By introducing a smoothing functions

$$p(x, \alpha) = x + \frac{1}{\alpha} \ln(1 + e^{-\alpha x}), \alpha > 0 \ ,$$

One obtains a smooth SVM (SSVM):

$$\min_{(w,\gamma) \in R^{n+1}} \frac{v}{2} \| p(\gamma - (A_1 w - \gamma b), \alpha) \|^2 + \frac{1}{2}(w^T w + b^2) \ . \quad (9)$$

4. A HEURISTIC METHOD

Due to the sparsity problem, most elements of feature vectors are empty. Learning machines will not do well based on these incomplete training instances. A straightforward method is to fill those empty elements in the user-item matrix with some default values, e.g., zeros or average ratings of the users. However, this method may mislead the learning machine because a user has not rated an item, it can be either the case that the user is not interested in it, or the case that the user is interested in it, but has not purchased the item yet. Furthermore, training a good learning machine needs a large number of training examples. In practice, only a few labeled data are available.

In this section, we present a SSVM-based heuristic (SSVMBH) method to overcome these problems by iteratively estimating missing elements in the user-item matrix *A*. For each element $a_{mn} \in A$, we have

$$a_{mn} = \begin{cases} x_{mn}, & \text{if } n \in I_m \\ p_{mn}, & \text{otherwise} \end{cases}$$

Initially, We randomly assign 0 or 1 to p_{mn} . Then, for each user u_m and item *n* where $n \notin I_m$, a linear classifier f_{mn} is trained by a SVM algorithm according to feature vector $a_k = (a_{1k}, a_{2k}, ..., a_{Mk})$, $1 \leq k \leq N$, $k \neq n$. Based on the experimental studies in the Section 5, the predictive accuracy of the SSVM is higher than that of the MLLS. Hence, we use the SSVM in our heuristic method. According to f_{mn} , a new p_{mn} is given. After each p_{mn} is re-computed, we test the current classifiers with the test data, denoted by *T*. Let $| T_c |$ be the total number of correct labels computed with current classifiers. The accurate rate is defined as $| T_c | / | T |$. If the difference of accurate rates between two consecutive steps is less than a predefined value ε , the algorithm stops. Otherwise, this procedure is repeated. The details of the algorithm are given as follow.

```
Algorithm
Initialize $k = 1, \varepsilon > 0, T_c^0$=0, and $T_c^1$=2$\varepsilon$
Initialize user-item matrix $A$ by randomly filling 0 or 1
for empty entries
while $(T_c^k - T_c^{k-1}) \leq \varepsilon$
  for $m = 1,...,M$
      for $n = 1,...,N$
        if $n \notin I_m$, then
           build a classifier $f_{mn}$ with SSVM
           predict the value of $a_{mn}$ with $f_{mn}$
           re-assign a label to $a_n$ based on the prediction
        end
      end
  $k = k + 1$
  Compute $T_c^k$ for test data
end
```

5. EXPERIMENTAL STUDIES

Our goal is to evaluate the performance of the above heuristic method in CF systems. The experimental studies include two parts. We first compare the performances of the SSVM and MLLS, and, then, compare our heuristic method with the user-based and item-based method.

5.1 Data Set

To compare the two SVM algorithms, we use five datasets obtained from the UCI repository: WPBC, Ionosphere, Cleveland Heart, Pima Indians, and BUPA Liver. The first dataset WPBC provides diagnostic information of breast cancer patients. This dataset contains 569 instances, 2 classes (malignant and benign), and 30 numeric attributes. The second dataset Ionosphere contains 351 complete instances of radar returns from the ionosphere. There are 2 classes and 34 numeric attributes. The third dataset Cleveland Heart contains 297 instances. 13 of 75 attributes are chosen. The fourth dataset Pima Indians Diabetes contains 768 instances of tested data for diabetes. There are 2 classes and 8 attributes. The last dataset BUPA Liver contains 345 instances of patients with liver disorders. The data has 7 attributes and belongs to 2 classes.

For the second part of experimental studies, we use a dataset from MovieLens[12]. In this database, there are about 43000 users who have given ratings on 3500 different movies. Before the training process, some data, e.g., some users who just rated on very few movies and some movies which were rated by very few users, have to be cleaned out. The remaining data were randomly divided into training set and test set according to 80/20 ratio. We follow the experimental procedures introduced in [16][22]. Two training set dataset A and dataset B were created.

5.2 Results

We use the average accurate rate to evaluate the predictive accuracy. Each data set is run 10 times with different random choices for the training data and test data. Mean values are computed over 10 experiments.

Table 1. Comparison of the accurate rate between the SSVM and MLLS methods

	SSVM (%)	MLLS (%)
WPBC (24 months)	81.29	82.79
WPBC (60 months)	75.05	72.53
Ionosphere	87.17	86.89
Cleveland Heart	83.82	81.81
Pima Indians	77.34	70.17
BUPA Liver	68.10	65.50

All accurate rates of the two algorithms are shown in Table 1. As we can see, the accurate rates of the SSVM are usually higher than those of the MLLS except the dataset WPBC (24 months). We note that it takes the SSVM a little longer training time than the MLLS. The reason is that the complexity of the SSVM is higher than that of the MLLS. It uses Newton method and Armijo linear search to solve problem (9). It is well known that Newton method is quadratic. Also, we find that parameter ν is critical to the accurate rates. According to our experimental studies, we fix it as $\nu = 0.001$. In recommendation systems, the time of training a classifier is not critical because this process can be done off-line. Hence, we select the SSVM as the classifier of our heuristic method.

In the second part of our experimental studies, we will show the efficiency of the SSVMBH method by comparing it with the user-based, item-based, and SSVM algorithms aforementioned (all missing values are initialized with zeros). We still use the average accurate rate as a metric to evaluate the performance of algorithms. ε is selected as 0.005 in the entire experiment. Since we only consider binary classification problem in this paper, the accurate rates achieved by SSVMBH cannot be directly compared with the user-based and item-based algorithms. We re-assigned label +1 to the data with rating 4 or 5, and label 0 to the data with rating 1,2, or 3.

The results achieved by four algorithms for two datasets are given in Table 2. As we can see, the accurate rates of the SVM methods are higher than the user-based and item-based approaches. The SSVMBH method outperforms all other approaches. The accurate rates of the SSVMBH method have been increased by about 3% compared with the SSVM method. Note that since we transformed all ratings from five ratings to 0 or 1, the accurate rates of the user-based and item-based are reasonable worse than the results reported in [16].

Table 2. Comparisons of the SSVMBH with other approaches

	Dataset A(%)	Dataset B(%)
SSVMBH	71.255	69.760
SSVM	68.580	66.834
Item-based	61.326	60.132
User-based	61.271	60.321

6. CONCLUSIONS

Using machine learning algorithms for recommendations is an interesting topic in collaborative filtering system. In this paper, we have proposed a SVM-based heuristic method to overcome the problem caused by the sparsity of user-item matrix. The feature vectors based on a sparse user-item matrix can mislead a learning machine and cause a bad result. In our heuristic method, we repeatedly estimated the missing values in the user-item matrix according to the results obtained from the previous iteration. We compared the SVMBH method with standard SSVM, user-based, and item-based algorithms. The performance data has shown that the accurate rates of the SVMBH method are higher than those of other algorithms.

7. REFERENCES

[1] Justin Basilico, Thomas Hofmann: Unifying collaborative and content-based filtering. ICML 2004

[2] Basu, C., Hirsh, H., and Cohen, W. (1998). Recommendation as Classification: Using Social and Content-based Information in Recommendation. In *Recommender System Workshop '98*. pp. 11-15.

[3] Bharat, K., T. Kamba, and M. Albers. Personalized, Interactive News on the Web. *Multimedia Systems,* 6(5), 1998, pp. 249-358.

[4] Billsus, D. and Pazzani, M. J. 1998. Learning collaborative information filters. In Proceedings of the 15th International Conference on Machine Learning. Morgan Kaufmann, San Francisco, CA, 46–54.

[5] Breese, J. S., Heckerman, D., and Kadie, C. (1998). Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence,* pp. 43-52.

[6] C. Burges. A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery, 2:121–167, 1998.

[7] M. Chesire, A. Wolman, G. M. Voelker, and H. M. Levy, “Measurement and analysis of a streaming media workload,” *Proc. 3rd USENIX Symposium on Internet*

Technologies and Systems, San Francisco, CA, March 26-28, 2001.

[8] T. Joachims. Text categorization with support vector machines. In European Conference on Machine Learning (ECML), 1998.

[9] Yuh-Jye Lee and O.L. Mangasarian, SSVM: Smooth Support Vector Machine. Philadelphia INFORMS, 1999

[10] Melville, P., R.J. Mooney, and R. Nagarajan. Content-Boosted Collaborative Filtering. *ACM SIGIR 2001 Workshop on Recommender Systems*, New Orleans, LA, 2001.

[11] McNee, S., I. Albert, D. Cosley, P. Gopalkrishnan, S.K. Lam, A.M. Rashid, J.A. Konstan, and J. Riedl. On the Recommending of Citations for Research Papers. In Proceedings of the ACM 2002 Conference on Computer Supported Cooperative Work (CSCW 2002), New Orleans, LA, 2002, pp. 116-125. 17.

[12] MovieLens, http://www.cs.umn.edu/Research/GroupLens/

[13] Müller, K.-R., S. Mika, Rätsch, G., K. Tsuda and B. Schölkopf: An Introduction to Kernel-Based Learning Algorithms. IEEE Transactions on Neural Networks **12*(2)***, 181-201 (2001)

[14] P. Jonathon Phillips: Support Vector Machines Applied to Face Recognition. NIPS 1998: 803-809.

[15] Resnick, P., N. Iacovou, M. Sushak, P. Bergstrom, and J. Riedl. GroupLens: An open architecture for collaborative filtering of netnews. In Proceedings of the ACM 1994 Conference on Computer Supported Collaborative Work (CSCW '94), Chapel Hill, NC, 1994, pp. 175-186.

[16] Sarwar, B. M., Karypis, G., Konstan, J. A., and Riedl, J. 2001. Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th International World Wide Web Conference (WWW10). Hong Kong.

[17] A.J. Smola, A. Elisseeff, B. Schölkopf, and R.C. Williamson. Entropy numbers for convex combinations and MLPs. In A.J. Smola, P.L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 369-388, Cambridge, MA, 2000. MIT Press.

[18] A. J. Smola. *Learning with Kernels*. PhD thesis, Technische Universität Berlin, 1998. GMD Research Series No. 25

[19] Olivier Teytaud, David Sarrut: Kernel Based Image Classification. ICANN 2001: 369-375

[20] Ungar, L. and Foster, D. 1998. Clustering methods for collaborative filtering. In Proceedings of the Workshop on Recommendation Systems. AAAI Press, Menlo Park California.

[21] Vapnik, V. (1998). *Statistical learning theory*. New York: JohnWiley.

[22] Tong Zhang and Vijay S. Iyengar, Recommender Systems Using Linear Classifiers, Journal of Machine Learning Research, 2:313-334, 2002.