Symmetric Collaborative Filtering using the Noisy Sensor Model

by

Rita Sharma

M.Tech., University of Roorkee, India

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE STUDIES

(Department of Computer Science)

We accept this thesis as conforming
to the required standard

The University of British Columbia

February 2001

In presenting this thesis/essay in partial fulfillment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying for this thesis for scholarly purposes may be granted by the Head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Feb 6, 2001
Date

Department of Computer Science
The University of British Columbia
2366 Main mall
Vancouver, BC
Canada V6T 1Z4

Abstract

Collaborative filtering is the process of making recommendations regarding the potential preference of a user, for example shopping on the Internet, based on the preference ratings of the user and a number of other users for various items. This thesis considers the problem of symmetric collaborative filtering based on explicit ratings. To evaluate the algorithms, we consider only *pure* collaborative filtering, using given ratings and excluding other information about the people or items.

Our approach is to predict an active user's preferences regarding a particular item by using other people's ratings of that item and other items rated by the active user as noisy sensors. The noisy sensor model uses Bayes' theorem to compute the probability distribution for the active user's rating of a new item. We give two variants for learning the noisy sensor model: one, for explicit binary rating data; and the second, for explicit multi-valued rating data. The model for binary rating data is based on Bayesian learning. Its performance motivate us to further explore the use of noisy sensor model for multi-valued rating data.

We give two variant models for multi-valued rating data: in one, we learn a linear model of how users rate items; in another, we assume different users rate items identically, but that the accuracy of the sensors must be learned. We compare the two models of multi-valued rating data with state-of-the-art techniques and show how they are significantly better whether a user has rated only two items or many. We report empirical results using the EachMovie database of movie ratings. We also show that by considering the items similarity along with the users similarity the accuracy of the prediction increases.

Contents

List of Tables

List of Figures

Acknowledgements

I would like to thank my supervisor, David Poole, for his ideas, comments, guidance and financial support. I would like to thank Holger Hoos, as my second reader, for his valuable advice on my thesis draft, discussions, and providing Perl scripts to make EachMovie subsets.

I would like to thank David M. Pennock for providing the EachMovie dataset subsets. I would like to thank Kirsty Barclay for taking the time to proof read my thesis draft.

Finally, I would like to thank my husband, Sudhir, and my son, Ashutosh, for their unconditional encouragement and support. Without them I might not be here! I would also like to thank my parents for their encouragement and support.

RITA SHARMA

The University of British Columbia
February 2001

Chapter 1

Introduction

1.1 Information Filtering

In today's world, filtering vast amount of information has become an important part of daily life for an increasing number of people. The amount of information available through books, movies, news, advertisements, and in particular, on-line sources such as email, Usenet News, and Web sites, has become enormous and is considerably more than any one person can absorb.

The World Wide Web has emerged as a relevant electronic marketplace. It is no longer feasible for a person to filter through this sea of information unaided, and to quickly and conveniently find the most interesting and relevant information to them. The situation is worsening. In response to the challenge of information overload, information filtering techniques are sought to help people quickly identify what will likely be of most interest to them.

There are two major approaches to information filtering: content- based filtering and collaborative filtering.

1.1.1 Content-based Filtering

Content-based filtering techniques filter the information based on their content. For example, in text documents it compares the contents of text documents with a user's profile (preferences) and selects documents whose contents match the user's profile. Examples of these systems include *kill files* (used to filter out advertising), e-mail filtering software (which sorts e-mail into categories based on sender information or based on keyword matches in the mail fields). The techniques used in content-based filtering can vary greatly in complexity. Keyword-based search is one of the simplest techniques that involves matching different combinations of keywords (sometimes in boolean form). Content-based filtering does not depend on having other users in the system. However, there are some limitations to content-based filtering [SM95].

Its primary limitation is its lack of inherent methods for finding related documents not specified by the user. Since in general all information about a user's prefernces is obtained from material that the user has seen, the user cannot specifically call for items that he is not aware of already. In practice, additional hacks must often be added to content-based filtering.

Another problem is that the items must be in some machine parsable form, or attributes must have been assigned to the items by hand. With current technology, media such as sound, photographs, art, video or physical items cannot be analyzed automatically for relevant attribute information. Often it is not practical or possible to assign attributes by hand due to the limitation of resources.

Finally, current content-based filtering makes recommendations based on analytical criteria, such as the frequency of a keyword used in the documents, but it knows nothing about the quality of the documents. For example,

it cannot distinguish between a well written and a badly written document if both documents happen to use the same keywords.

1.1.2 Collaborative Filtering

Collaborative filtering techniques filter items for a user based on human judgements, not on analytical criteria, as in content-based filtering. The user for whom we are predicting a rating is called the active user. It uses the knowledge of many users to make predictions about individual user preferences. It automates the process of *word-of-mouth* by which people recommend products or services to one another [SM95]. Before the existence of automated information filtering systems, we may have sought the recommendations of friends for filtering the information. Collaborative filtering assists and augments this process by relying on potentially thousands of other people, and considering potentially thousands of different items.

Collaborative filtering systems build a database of user ratings of available items. Users of collaborative filtering systems usually rate items they have experience with in order to establish a profile of their interests. The collaborative filtering system then matches an active user to people with similar interests to make a recommendation for the active user. People's ratings can be explicit or implicit. *Explicit ratings* require that the users rate the items with a score, rank, or other classification of the item. *Implicit ratings* are collected by electronic agents, which monitor the user's behavior. This can be accomplished for example, by timing visits to sites using cookie files and history files, or by watching the user's browsing activities, such as save and print operations. Collaborative filtering enables the user to find new items that she may be interested in by drawing from the preferences of like-minded

people.

In collaborative filtering, the main premise is that the active user will prefer items which like-minded people prefer, or even that dissimilar people don't prefer. It relies on the fact that people's preferences are not randomly distributed; there are patterns within the preferences of a person and among similar groups of people, creating correlation.

This is best illustrated by example. Imagine three users: Mike, David and Brian have been asked to give their three favorite musicians.
Mike likes:

- Elvis
- Buddy Holly
- Little Richard

David likes:

- Jimmy Hendrix
- James Brown
- Aretha Franklin

Brian likes:

- Little Richard
- Elvis
- Jerry Lee Lewis

Collaborative filtering compares their preferences and finds that Mike and Brian are similar, it then swaps each man's prefernces: *Mike, you may like Jerry Lee Lewis*; *Brian, you may like Buddy Holly.* If people's preferences are not correlated, then such predictions can not be made.

Collaborative filtering can be formalized as: given a set of ratings for various user-item pairs, predict a rating for a new user-item pair. It is interesting that the abstract problem is symmetric between users and items.

Collaborative filtering has been a very lively research area in recent years. Several collaborative filtering algorithms have been suggested, ranging from binary to non-binary rating, implicit and explicit rating. Most comparisons to date are empirical in nature [BP98, HKBR99, RIS+94, BHK98, SM95, SKR99]. Initial collaborative filtering algorithms were based on statistical methods using correlation between user preferences. Collaborative filtering problem is not a standard machine learning problem because the data available is very sparse and also there is no other information about the users and items available except their ratings. Recently some researchers have used machine learning methods [BP98, UF98] for collaborative filtering algorithms. These methods essentially discover the hidden attributes for users and items, which explain the similarity between users and items.

Collaborative filtering is a key technology used to build Recommender Systems on the Internet. It has been used by recommender systems such as Amazon.com — a book store on the web, CDNow.com — a CD store on the web, and MovieFinder.com — a movie site on the internet [SKR99]. Amazon.com suggests books to customers based on other books the customers have told Amazon they like. CDNow.com helps customers choose CDs to purchase as gifts, based on other CDs the recipient has liked in the past.

1.2 Contributions

Most of the approaches suggested for collaborative filtering are asymmetric in nature, meaning they either attempt to find similar users based on their preferences of items (e.g., the GroupLens system [RIS+94]), *or* they attempt to find similar items based on user preferences (e.g. the Bayesian network model described in [BHK98]). However, the abstract problem is symmetric between users and items. There is no convincing reason to use only user similarity or item similarity. In this thesis we show that a symmetric collaborative filtering technique using both user and item similarity offers significant advantages over asymmetric techniques.

In this thesis we focus on symmetric collaborative filtering. We propose and evaluate a probabilistic approach based on a noisy sensor model. We describe how a collaborative filtering problem could be solved using the noisy sensor model. Our approach is based on the idea that to predict a user rating for a particular item, we can use all those people who rated that item and other items rated by that user as the noisy sensors. We view the noisy sensor model as a belief network. The conditional probability table associated with each sensor node reflects the noise in the sensor. After learning the noisy sensor model, or the conditional probability table, associated with each sensor node, we use Bayes' theorem to compute the probability distribution for the active user's rating of a new item.

There are many ways of learning the conditional probability tables associated with each sensor node. Here we give two variants of the general idea for learning the noisy sensor model: one, for explicit binary rating data; and the second, for explicit multi-valued rating data. Algorithm *AlgoBayes*, based on Bayesian statistics (Bayesian Learning), is for binary rating data. *AlgoBayes*

is based on the assumption that two users are similar if they rate an item the same. We use the *maximum a posteriori* hypothesis for computing the similarity between two users, given their preferences over co-rated items. We show that the performance of *AlgoBays* improves when we use both items and users as noisy sensors, which motivates us to further explore the noisy sensor model for multi-valued rating data.

The algorithms (*AlgoEqual*, *AlgoLinear*) for multi-valued rating data are based on assuming a linear relationship with Gaussian error between users and items. In *AlgoLinear* we learn a classical linear regression model of how users rate items, and in *AlgoEqual* we assume that the different users rate items the same, but the accuracy of the sensor needs to be learn.

We compare the two models of multi-valued rating data with state-of-the-art techniques and show how they are significantly better in all cases ranging from cases where a user has only rated two items, to cases where the user has rated many items. We report empirical results on the EachMovie database of movie ratings. We show that symmetric collaborative filtering, which employs both users and item similarity, offers better accuracy than asymmetric collaborative filtering.

1.3 Outline of the Document

The rest of the thesis is organized as follows. Chapter Two presents related work. Previous research in this area is discussed. Chapter Three presents a noisy sensor model and its application to collaborative filtering. The algorithm *AlgoBays* for binary rating data and its results on the EachMovie database are discussed. Chapter Four presents the use of a noisy sensor model for multi-valued rating data. Algorithms *AlgoLinear* and *AlgoEqual* are discussed. The

empirical analysis of results and comparison between algorithms are discussed. Chapter Five presents conclusions and possibilities for future work.

Chapter 2

Related Work

Collaborative filtering involves making predictions about a person's preferences for a particular item, using pre-existing information about this user in conjunction with other users' information. Collaborative filtering incorporates the assumption that by finding similar users and examining their behavior or preferences, we can make useful recommendations for an active user.

The concept of collaborative filtering originates from the work of Goldberg et al. [GNOT92] in the area of information filtering. Tapestry is one of the first collaborative filtering systems used for recommending electronic documents, such as e-mail and Netnews. Tapestry was developed at Xerox Palo Alto Research Center [GNOT92]. In this system users could attach annotations as they read documents. Annotations became accessible as virtual fields of the documents, and the filters that searched the annotations for interesting articles were constructed by the end user, using a special query language designed for this task. The query could involve many different criteria, including keywords, subject, author and annotations given to the document by other people. The collaborative filtering provided by Tapestry was not automated but the recommendations of other people were taken into account. In this sys-

tem the user had to specify other users with whom she would share interests inorder to obtain the recommendations [GNOT92].

GroupLens [RIS+94], a neighborhood-based algorithm, is one of the best known early automated collaborative filtering systems for recommending the articles of Usenet news. GroupLens used the Pearson correlation coefficient to weight user similarity, used all those users who rated the item j, and computed a weighted average of deviations from the user's mean:

$$S_{ai} = \overline{S}_a + \frac{\sum_{u=1}^{n} \left(S_{ui} - \overline{S}_u\right) * r_{au}}{\sum_{u=1}^{n} |r_{au}|}$$

S_{ui} represents the prediction for the user u for item i, n is the number of users, $\overline{S}_u$ represents the average rating for user u and r_{au} is the similarity between the active user and user u as defined by the Pearson correlation coefficient.

$$r_{au} = \frac{\sum_{i=1}^{m} \left(S_{ai} - \overline{S}_a\right) * \left(S_{ui} - \overline{S}_u\right)}{\sqrt{\sum_{i=1}^{m} \left(S_{ai} - \overline{S}_a\right)^2 * \sum_{i=1}^{m} \left(S_{ui} - \overline{S}_u\right)^2}}$$

where the summations over m are over the co-rated items, items rated by both users a and u.

The Ringo music recommender [SM95] expanded upon the GroupLens algorithm. Shardanand and Maes [SM95] compared four different recommendation algorithms based on the mean errors in prediction using each algorithm. They claimed better performance by computing correlations using a constrained Pearson correlation coefficient:

$$r_{au} = \frac{\sum_{i=1}^{m} (S_{ai} - 4) * (S_{ui} - 4)}{\sqrt{\sum_{i=1}^{m} (S_{ai} - 4)^2 * \sum_{i=1}^{m} (S_{ui} - 4)^2}}$$

where 4 was chosen because it was the midpoint of their seven-point rating scale. Ringo considers only those neighbors whose correlation was greater

than a fixed threshold. To generate predictions, Ringo computed a weighted average of ratings from all users in the neighborhood.

Breese et al. [BHK98] proposed the use of vector similarity, based on vector cosine to weight user similarity. This method is often employed in the field of information retrieval for measuring the similarity between two documents. A weighted average of ratings from all users in the neighborhood is used to generate the prediction. The authors also performed an empirical analysis of neighborhood-based collaborative filtering algorithms. The authors found that vector similarity does not perform as well as Pearson correlation [BHK98].

Neighborhood or Correlation based algorithms predict the active user rating as a similarity-weighted sum of the other users ratings. These algorithms are simple, work reasonably well in practice, and new data can be added easily and incrementally. They are also referred to as memory based algorithms [BHK98]. These algorithms become computationally expensive as the size of the database grows.

All the above mentioned neighborhood-based algorithms [RIS+94, SM95, BHK98] are asymmetric in nature. They implement only users' similarity in making predictions.

These algorithms do not account for what amount of trust can be placed in a similarity measure with a neighbor. It is not uncommon in a collaborative filtering system for the active user to have highly similar neighbors assessed on a very small number of co-rated items (often three to five co-rated items). The more co-rated items available for comparing the opinions of two users, the more we can trust that the computed similarity is representative of the true similarity between them. While trying to fit linear relationship, one can often get a good linear fit with small amount of data, even though the sensor is not

a reliable sensor; one may have a better sensor with more data, but not such a good linear relationship. Our prosposed noisy sensor models for collaborative filtering address this problem.

Breese et al. [BHK98] proposed several model based collaborative filtering algorithms. In model based collaborative filtering, first the user database is used to learn a model of users, items, and/or ratings; predictions are then made using the model. The authors described and evaluated two probabilistic models for collaborative filtering: *cluster models* and *Bayesian networks.* In the cluster model, users with similar preferences are clustered together. This model structure is a naive Bayesian network, wherein a user's preferences regarding various items are independent given his class membership. The model's parameters, the number of clusters, and the conditional probability of ratings given class are estimated from the training data. In the Bayesian network, nodes correspond to items in the database. The state of each node corresponds to the possible ratings for that item. The structure of the network and the conditional probabilities are learned from the training data. Breese et al. showed that a Bayesian network approach outperformed correlation-based and cluster-based methods on an implicit rating database.

Ungar and Foster [UF98] suggest a clustering methods for collaborative filtering. Both users and items are classifieds into clusters. For each cluster of users, the probability that they like each cluster of items is estimated. They use K-means clustering and Gibbs sampling algorithms for clustering and model estimation. To compare the algorithms, they use both synthetic and real data.

Poole et al. [PHB00] proposed symmetric collaborative filtering for binary ratings data. They proposed two alternative algorithms for clustering the users and items based on their similarity: Clustering with Decision Trees and Clustering with Tables.

Memory requirements for model based algorithms are generally less than for storing a full database. Predictions can be calculated quickly once the model is generated, though the time needed to learn the model may be prohibitive, and addition of new data may require a full recompilation.

Pennock et al. [PHLG00] propose a collaborative filtering algorithm called Personality Diagnosis (PD). This algorithm is based on a probabilistic model of how people rate items. In this approach it is assumed that each user reports ratings with Gaussian error. Given the active user's known ratings for the items, the probability that the user has the same personality type as every other user is computed. After computing the personality type for an active user, the probability distribution of the active user's rating for a new item is computed, and the most probable rating returned as the predicted value. Authors present the empirical evaluation of PD with other neighborhood-based and model based algorithms on the Eachmovie and CiteSeer databases. They show that PD makes better predictions than four other algorithms (*Correlation, Vector Similarity, Bayesian Network, and Bayesian Clustering*).

Personality Diagnosis makes a strict assumption that the variance σ^2 of the normal distribution of a user's rating for an item is the same for all users: this parameter was tuned to maximize accuracy.

Chapter 3

Symmetric Collaborative Filtering Using The Noisy Sensor Model

3.1 Filtering Problem and Mathematical Notation

Let N be the number of users and M be the total number of items in the database. S is a $N \times M$ matrix of all user's ratings for all items, S_{ui} is the rating given by the user u to item i. In collaborative filtering, S, the user-item matrix, is generally very sparse since each user will only have rated a small percentage of the total number of items. With this formulation, the collaborative filtering problem becomes predicting those S_{ui} which are not defined in S, the user-item matrix. The active user is denoted by a such that $a \in \{1, 2, \ldots, N\}$.

Let the ratings be on a cardinal scale with m values that we denote

$v_1, v_2, \ldots, v_m$. Then each rating S_{ui} has a domain of possible values $(v_1, v_2, \ldots, v_m)$. In our equations, it is always assumed that we perform operations on those values of S_{ui} that are exist.

3.2 Noisy Sensor Model

We propose a simple probabilistic approach for symmetric collaborative filtering using the noisy sensor model [RN95] for calculating the rating by the active user a of an unseen item j, based on the ratings of those users who rated the item j, and the observed ratings of the active user on other items. We designate all those users who rated the item j and all other items rated by active user a to be the noisy sensors.

The noisy sensor model is depicted as a naive Bayesian network in Figure 3.1. The direction of the arrow shows that the prediction of active user a for item j "causes" the sensor u to take on a particular prediction for item j and similarly, the prediction of active user a for item j "causes" sensor a to take on a particular prediction for item k. The sensor model is the conditional probability table associated with each sensor node. The noise in the sensor is reflected by the probability of incorrect prediction; that is, by the conditional probability table associated with it. To make the model simple we use the independence assumption that the prediction of any sensor for item j is independent of others given the prediction of active user a for item j.

We need the following probabilities for Figure 3.1:

$Pr\left(S_{uj}|S_{aj}\right)$: the probability of user u's prediction for item j, given the prediction of active user a for item j.

$Pr\left(S_{ak}|S_{aj}\right)$: the probability of active user a's prediction for item k, given the prediction of active user a for item j.

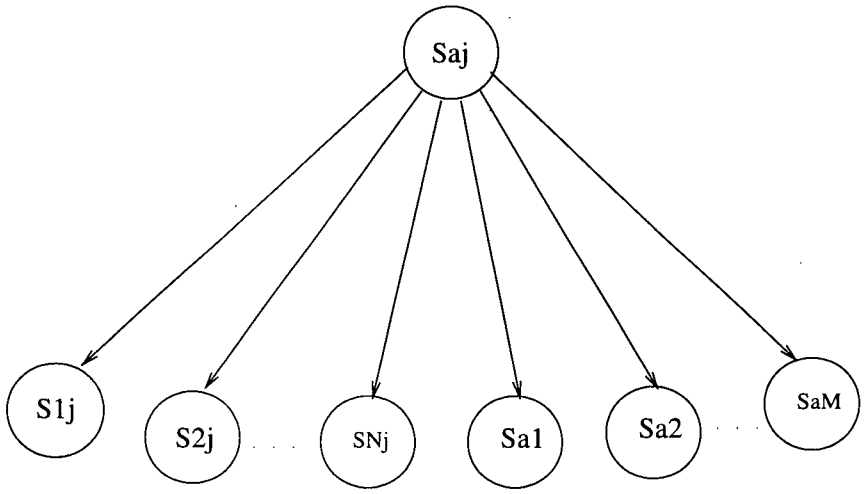

Figure 3.1: Naive Bayesian Network Semantics for The Noisy Sensor Model

$Pr(S_{aj})$: the prior probability of active user a's prediction for item j.

We compute the prior probability distribution $Pr(S_{aj} = v_i)$ of the active user's rating for item j by the fraction of rating v_i in the training data set, where $v_i \in (v_1, v_2, \ldots, v_m)$.

Given the conditional probabilities table for all noisy sensors, we can compute the probability distribution for the active user's rating of an unseen item j using the noisy sensor model, described in Figure 3.1. By applying Bayes' rule we can have the following:

$$Pr(S_{aj} | (S_{1j}, \ldots, S_{Nj}) \wedge (S_{a1}, \ldots, S_{aM}))$$
$$\propto Pr(S_{aj}) \cdot \prod_{u=1}^{N} Pr(S_{uj}|S_{aj}) \cdot \prod_{k=1}^{M} Pr(S_{ak}|S_{aj})$$

We have showed how a collaborative filtering problem could be solved using the noisy sensor model, now we need the probability table for probabilities $Pr(S_{uj}|S_{aj})$ and $Pr(S_{ak}|S_{aj})$ to make the resultant recommendation.

Consider first the problem of estimating $Pr(S_{uj}|S_{aj})$, which is the problem of estimating user u's rating for item j given active user a's rating of it. Generally, in collaborative filtering the data available is very sparse. There-

fore, to compute the probability table from sparse data we need to make some prior assumptions about the relationship. In this thesis we propose three different algorithms for learning the noisy sensor model. The first algorithm, *AlgoBayes*, the simplest algorithm for explicit binary rating data, is based on Bayesian learning. The second algorithm, *AlgoLinear*, for multi-valued rating data, is based on a *linear relationship with Gaussian error* between the preferences of users and similarly between the preferences received by items. The third algorithm *AlgoEqual*, a variant of *AlgoLinear*, is based on the idea that different users rate items the same, but the accuracy of the sensors need to be learned.

3.3 Algorithm *AlgoBayes*

The algorithm *AlgoBayes* is based on Bayesian learning. Suppose that active user a and user u co-rated n items, and that their ratings over n co-rated items are denoted by n pairs of observations $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$. Then x_l denotes the rating given by user a and y_l denotes the rating given by user u for the l^{th} observation. We assume the binary ratings as 1 and 0. To compute the probability table $Pr(S_{uj}|S_{aj})$ we need to learn two parameters.

Generally, in collaborative filtering the data available is very sparse. Therefore, to reduce the number of parameters we assume the following:
$\Pr(S_{uk} = 1|S_{ak} = 1) = Pr(S_{uk} = 0|S_{ak} = 0) = \theta_{au}$, for random k
where M is the total number of items in the database.

Now, we have the following:

$\Pr(S_{uk} = S_{ak}) = \theta_{au}$, for random k

$\Pr(S_{uk} \neq S_{ak}) = 1 - \theta_{au}$, for random k

We want to find the *maximum a posteriori* value of the parameter θ_{au}

given the ratings of user a and u over all co-rated items. $Pr\left((S_{ak} = S_{uk})\,|D\right)$ is the posterior probability of users a and u for rating the items identical. It reflects our confidence that users a and u will rate the items same after we have seen the data D (ratings given by users a and u over n co-rated items). Using Bayes' rule we can write the posterior probability of similarity thus:

$$Pr\left((S_{ak} = S_{uk})\,|D\right) = \frac{Pr\left(D|\,(S_{ak} = S_{uk})\right) * Pr\left((S_{uj} = S_{aj})\right)}{Pr\left(D\right)}$$

$Pr\left(D\right)$ is a normalizing constant and $Pr\left(S_{uj} = S_{aj}\right)$ is the prior probability of users a and u for rating the items same. We need a method to find the prior distribution of probability $Pr\left(S_{ak} = S_{uk}\right)$. We assume the *beta* distribution for the prior probability as it is convenient and most commonly used distribution [Hec95].

$Pr\left(S_{ak} = S_{uk}\right) = \theta_{au}^{K} * (1 - \theta_{au})^{K}$

where $K \geq 0$ is the parameter of beta distribution.

Now,

$Pr\left(D|\,(S_{ak} = S_{uk})\right) = \prod_{l=1}^{n} Pr\left(D_l|\,(S_{ak} = S_{uk})\right)$

The datacase $D_l = (x_l, y_l)$ denotes the rating given by users a and u to the l^{th} co-rated item.

$Pr\left(D_l|\,(S_{ak} = S_{uk})\right) = Pr\left(x_l, y_l, |\,(S_{ak} = S_{uk})\right)$

The ratings of users a and u are not independent given that they rate the items same. Hence, the above equation can be written thus:

$Pr\left(D_l|\,(S_{ak} = S_{uk})\right) = Pr\left(x_l|y_l \wedge (S_{ak} = S_{uk})\right) * Pr\left(y_l|\,(S_{ak} = S_{uk})\right)$

The probability $Pr\left(y_l|\,(S_{ak} = S_{uk})\right)$, probability of user u's rating for item l given that users a and u rate the items same, is a constant because user a's rating is not known. It can be computed as the frequency of the rating y_l in the training set. The probability $Pr\left(x_l|y_l \wedge (S_{ak} = S_{uk})\right)$ is θ_{au} or $1 - \theta_{au}$

depending upon whether the ratings x_l and y_l are the same or not.

$$Pr\left(D_l|\left(S_{ak}=S_{uk}\right)\right)=\begin{cases}\theta_{au}*\text{constant} & \text{if } x_l=y_l\\ \left(1-\theta_{au}\right)*\text{constant} & \text{if } x_l\neq y_l\end{cases}$$

If users a and u give the same ratings on l items and different ratings for m items, the posterior probability denoted by M

$$Pr\left(\left(S_{ak}=S_{uk}\right)|D\right)=M$$

$$=\frac{Pr\left(D/\left(S_{ak}=S_{uk}\right)\right)*Pr\left(S_{uj}=S_{aj}\right)}{Pr\left(D\right)}=constant*\theta_{au}^{l+K}*\left(1-\theta_{au}\right)^{m+K}$$

To find the *maximum a posteriori* value of the parameter θ_{au} we differentiate M w.r.t θ_{au} and set equal to zero. The *maximum a posteriori* value of θ_{au} is given as:

$$\theta_{au}=\frac{l+K}{l+m+2K}$$

The value of θ_{au} shows how similar the users a and u are. The Bayesian learning approach for computing θ_{au}, the *maximum a posteriori hypothesis*, takes into account the prior probability which helps us in two ways. First, if users a and u give the same ratings over all co-rated items then the probability table of the sensor u will be deterministic for $K=0$ (but $K>0$ avoids this problem). We do not want a deterministic sensor in our model as it discounts the effect of other sensors and the deterministic prediction of the active user a for unseen item j will be based on only one sensor. Second, maximum a posteriori hypothesis approach does not enable the best sensor based on fewer co-rated items. For example, if users a and u rated 4 items together and $l=3, m=1$. If users a and v rated 12 items together and $l=9, m=3$ then $\theta_{av}=.71$ and $\theta_{au}=.66$ for $K=1$ and $\theta_{au}=\theta_{av}$ for $K=0$. For $K=1$, $\theta_{av}>\theta_{au}$, which means user a is more similar to user v than to user

u. Therefore, clearly the best or most reliable sensors will be based on more co-rated items. On the other hand, if we use a maximum likelihood technique for computing the value of θ_{au} then the above mentioned problem will arise, as it does not consider the prior probability.

We compute θ_{au} between the active user and all other users who have rated the item j and also between the item j and all other items rated by the active user a. To find the reliability of the noisy sensor we use the value of θ_{au}; the greater the value of θ_{au} the more reliable the noisy sensor. Note, for reliability we consider only similarity, not dissimilarity. We use the best U user noisy sensors and best I item noisy sensors for computing the active user a's prediction for unseen item j. The parameters setting for U and I is explained in the next section.

To predict a rating (for example, to compare it with other algorithms that predict ratings), we predict the expected value of the rating. The expected value of the rating is defined as follows:

$$E(S_{aj}) = Pr(S_{aj} = 1 | (S_{1j}, \ldots, S_{Nj}) \wedge (S_{a1}, \ldots, S_{aM})) * 1 + Pr(S_{aj} = 0 | (S_{1j}, \ldots, S_{Nj}) \wedge (S_{a1}, \ldots, S_{aM})) * 0$$

3.4 Results for *AlgoBayes*

There are no other explicit binary rating collaborative filtering algorithms to compare with, nor any explicit binary rating data to measure the performance of *AlgoBayes*. To test *AlgoBayes* we use the EachMovie database, available from the Digital Equipment Research Center. The EachMovie database contains ratings from 72,916 users for 1,628 movies, elicited on a integer scale from zero to five. We binarised the original rating data using a split point of 2.5. The ratings ≥ 2.5 are mapped to 1 and ratings < 2.5 are mapped to

0. We extract the subset of data for a restricted number n_u of users and n_m of movies. The resulting data sets are partitioned into the training and test sets by randomly selecting a fraction of ratings and moving them into the test set, keeping all remaning ratings as the training set. We created five subsets from the data by selecting one thousand users and one thousand movies. We select one thousand movies and five sets of thousand users. We keep the same one thousand movies in each subset but we select a different one thousand users. We also dreated two small subsets, one containing one hundred users and one hundred movies, and the other containing two hundred users and two hundred movies. We compare the performance of *AlgoBayes* with *Correlation* [RIS+94] and *Decision Tree* [PHB00] using the average absolute deviation metrics explained in Section 4.3 of the next chapter.

To see the effect of K on the performance of *AlgoBayes* we did experiments with different values of K. Figure 3.2 shows the variation of average absolute deviation with user noisy sensors for different values of K. This experiment was performed using best ten item noisy sensors: with $K = 0$, the performance of the algorithm is very poor. When we do not take the prior probability into consideration, *AlgoBayes* does not find good sensors. From Figure 3.2 we can not see clearly the performance of the algorithm for $K > 0$. Therefore, we show the performance of *AlgoBayes* for $K = 1$ to 3 in Figure 3.3. The average absolute deviation error with zero user noisy sensor, as shown in Figures 3.2 and 3.3, is based on the best ten item noisy sensors. The Figure 3.3 shows that for the small number of user noisy sensors (for example 15) the performance of the algorithm is better with $K = 3$ and 4, but the average absolute deviation error fluctuates a lot with small number of users and when it stabilizes then the performance of the algorithm is best, with $K = 1$. However, the improvement in accuracy is on the third place of decimal. We

set the value of K as 1 for all further experiments reported in this section.

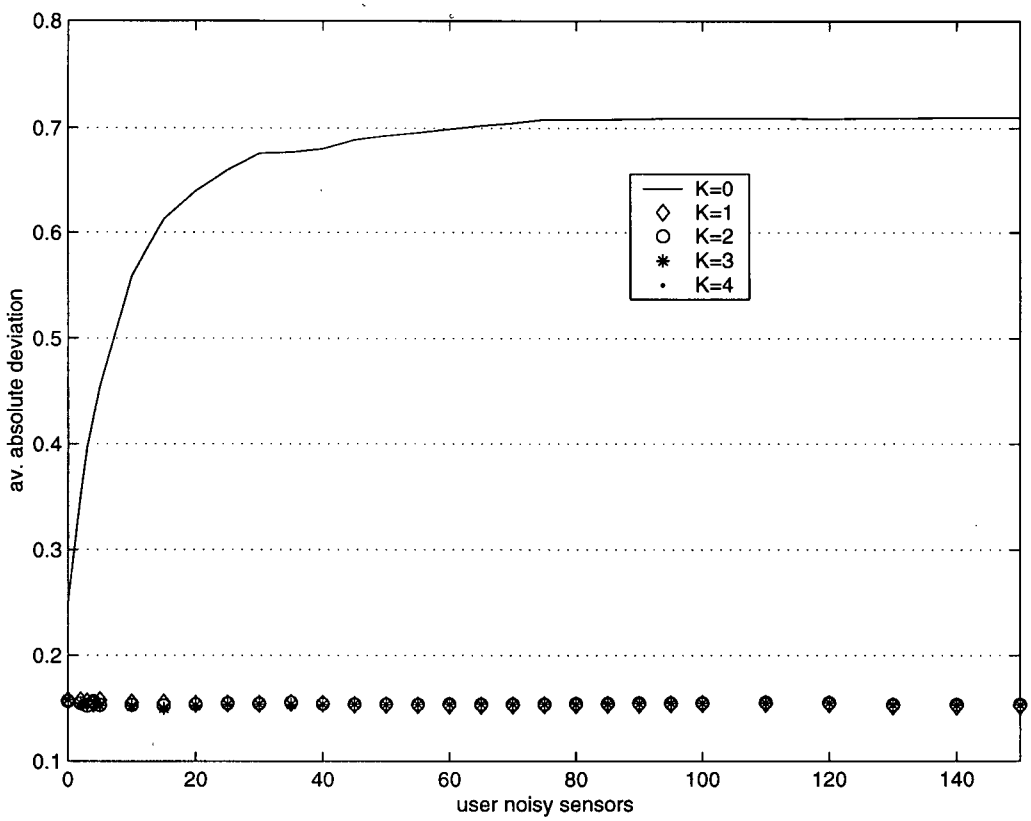

Figure 3.2: Effect of K on algorithm *AlgoBayes*. This experiment was performed using $I = 10$.

Figure 3.4 shows the variation of average absolute deviation with best user noisy sensors for different numbers of best item noisy sensors for *AlgoBayes*. It shows that the accuracy of *AlgoBayes* increases as we include items as noisy sensors along with the user sensors. It also shows that the average absolute deviation error first decreases with the increase in number of user noisy sensors and then increases as more user noisy sensors are user for prediction. This is because the large number of user sensors results in too much noise for those user sensors that have good reliability. It shows that *AlgoBayes* gives

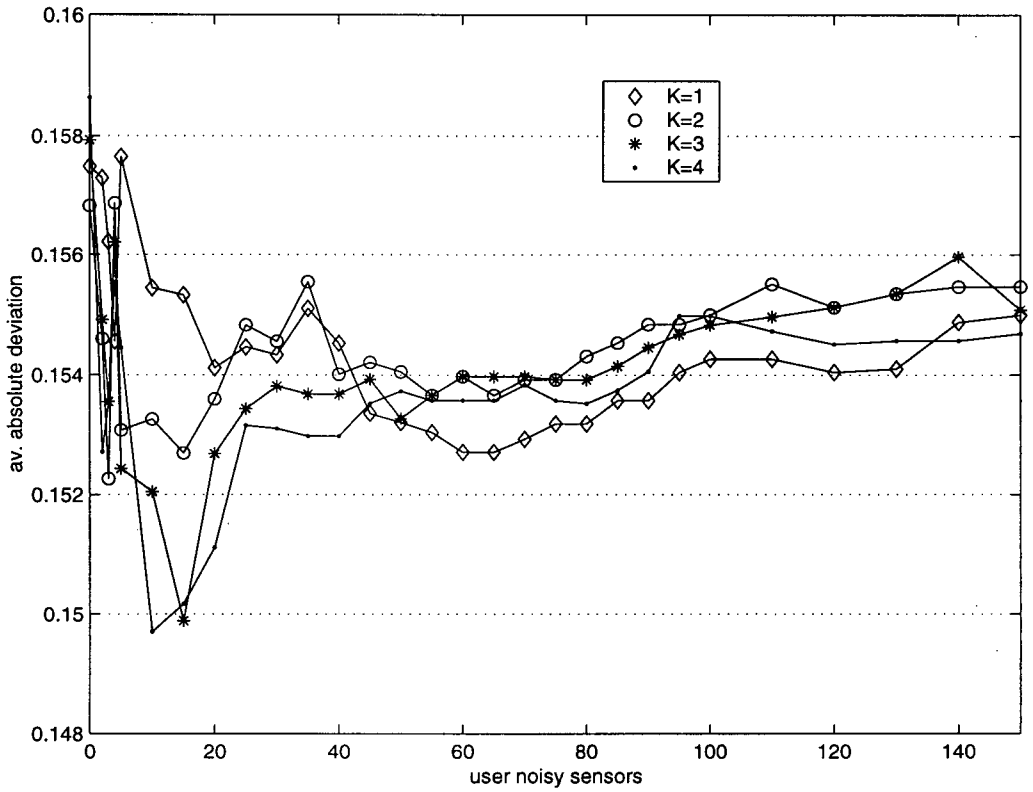

Figure 3.3: Effect of K on algorithm *AlgoBayes*. This experiment was performed using $I = 10$.

best accuracy with 10 items noisy sensors. With did experiment with all subsets and found that *AlgoBayes* gives better accuracy with ten-to-twenty items and forty-to-seventy users as noisy sensors. We set the parameter U as sixty (sixty users noisy sensors) and I as ten (ten items noisy sensors) for *AlgoBayes* for the experiments following in this section. The parameters U and I depend on the database in case of EachMovie database the number of users are more than the movies and each user has rated only few movies because of this the number of best user noisy sensors are more than the number of best item noisy sensors.

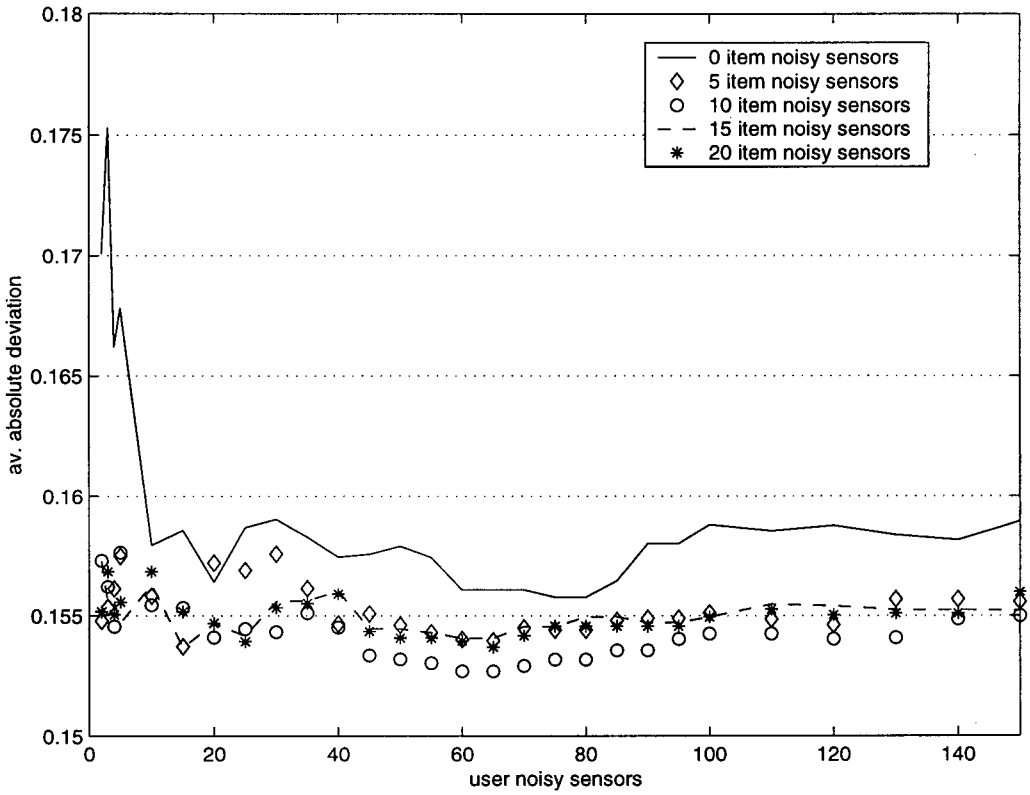

Figure 3.4: The average absolute deviation as a function of the number of best user noisy sensors for different numbers of best item noisy sensors.

Table 3.1 shows the results for *AlgoBayes*, *Correlation* and *Decision Tree* on EachMovie database. Table 3.1 shows that *AlgoBayes* works better than *Correlation* over all instances and better than *Decision Tree* over all 1000×1000 instances. The results of *AlgoBayes* for symmetric collaborative filtering using a noisy sensor model are very good and motivate us to further explore the use of a noisy sensor model for multi-valued rating data. The application of a noisy sensor model for multi-valued rating data is presented in the next chapter.

Instances	*Algorithm*		
	AlgoBayes	*Decision Tree*	*Correlation*
$em100 \times 100$	.242	**.216**	.321
$em200 \times 200$	.169	**.152**	.308
$em1000 \times 1000 - a$	**.133**	.145	.232
$em1000 \times 1000 - b$	**.154**	.161	.262
$em1000 \times 1000 - c$	**.198**	.228	.324
$em1000 \times 1000 - d$	**.236**	.245	.337
$em1000 \times 1000 - e$	**.225**	.240	.336

Table 3.1: Average absolute deviation scores on the EachMovie data for *AlgoBayes*, *Decision Tree* and *Correlation* (lower scores are better).

Chapter 4

Noisy Sensor Model for Multi-Valued Rating Data

We have described the Noisy Sensor model and its application in a collaborative filtering algorithm for explicit binary rating data in the previous chapter. However, very often one wants to make recommendations based on multi-valued ratings data. To compute $Pr\left(S_{uj}|S_{aj}\right)$, $m \times m$ probability table, where m is the possible number of ratings in multi-valued rating data, we need to learn m^2 parameters. However, the data available is very sparse and we need to make some prior assumptions about the relationship. To make the model simple or to reduce the number of learning parameters. Here, we assume that there is a linear relationship with Gaussian error between the preferences of users and similarly between the ratings received by the items. Because this is the simplest model as supported by **Occam's razor** [Mit97]. In this model we need to learn only three parameters, it reduces our problem of learning m^2 parameters to three parmeters.

Here we give two variants of the general idea for learning the noisy sensor model for explicit multi-valued rating data: one, where we learn a

linear relationship with Gaussian error between the preferences of users and between the ratings received by items (*AlgoLinear*); and another, where we assume that the different users rate items the same (the preferences of active user a and another user u are the same) and learn the variance of uesr u's prediction (*AlgoEqual*).

4.1 *AlgoLinear*

In AlgoLinear we learn the linear relationship with Gaussian error between the preferences of users and between the ratings received by items. Suppose the rating of the active user a (the independent variable) is denoted by x and the rating of the user u (the dependent variable) is denoted by y. Suppose that the active user a and user u co-rated n items and their ratings over n co-rated items are denoted by n pairs of observations $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$. We want to find a straight line which best fits these points.

We use *classical normal linear regression model* to find the relationship between the preferences of the active user and other user. We assume that the mean of y can be expressed as a linear function of independent variable x. Since a model based on the independent variable cannot predict exactly the observed values of y, it is necessary to introduce an error term e_i. For the i^{th} observation, we have the following:

$$y_i = \alpha + \beta x_i + e_i$$

We assume that unobserved errors (e_i) are independent and normally distributed with a mean of zero and the variance σ^2. Since y_i is a linear function of e_i, which is normally distributed, y_i itself will also be normally distributed. We assume that the variance σ^2 is the same for all observations. The mean and variance of y_i are given thus:

$E(y_i) = \mu = \alpha + \beta x_i$

$var(y_i) = \sigma^2$

For the i^{th} observation, the probability distribution function may be written thus:

$$P(y = y_i | x = x_i) = \frac{1}{\sqrt{2\pi\sigma^2}} exp\left[\frac{-1}{2\sigma^2}(y_i - \mu_i)^2\right]$$

where $\mu_i = \alpha + \beta x_i$

The joint probability distribution function or the likelihood of all the observations is the product of the individual $P(y_i/x_i)$ over the entire observations. We denote the likelihood function by $LF(\alpha, \beta, \sigma^2)$.

$$\begin{aligned} LF\left(\alpha, \beta, \sigma^2\right) &= \prod_{i=1}^{n} P(y_i|x_i) \\ &= \frac{1}{\left(\sqrt{2\pi\sigma^2}\right)^n} exp\left[\frac{-1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \mu_i)^2\right] \end{aligned}$$

We apply the maximum likelihood method [Guj95] to estimate the unknown parameters $(\alpha, \beta, \sigma^2)$. Maximizing the likelihood of LF is equivalent to maximizing the log likelihood of LF. The log likelihood of LF can be expressed thus:

$$\begin{aligned} \log LF &= \log \prod_{i=1}^{n} P(y_i|x_i) \\ &= \sum_{i=1}^{n} \log P(y_i|x_i) \\ &= -\frac{n}{2}\log(2\pi) - \frac{n}{2}\log\left(\sigma^2\right) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - (\alpha + \beta x_i))^2 \end{aligned}$$

Differentiating the above equation partially with respect to α, β and σ^2 and setting equal to zero, we get the following values of the parameters:

$$\begin{aligned}
\beta &= \frac{n\sum y_i - \sum y_i x_i}{n\sum x_i - \sum x_i^2} \\
\alpha &= 1/n\left(\sum y_i - \beta \sum x_i\right) \\
\sigma^2 &= \frac{1}{n}\sum (y_i - \alpha - \beta x_i)^2 \\
&= \frac{1}{n}\sum e_i^2
\end{aligned}$$

After calculating the parameters α, β and σ^2 we can write the probability distribution of the user u's preference for item j given the active user a's preference of item j as follows:

$$\begin{aligned}
Pr\left(S_{uj} = x_{uj} | S_{aj} = x_{aj}\right) &= P\left(y = x_{uj} | x = x_{aj}\right) \\
&= \frac{1}{\sqrt{2\pi\sigma^2}} exp\left[\frac{-1}{2\sigma^2}\left(x_{uj} - (\alpha + \beta x_{aj})\right)^2\right]
\end{aligned}$$

After computing this probability distribution for all users who rated the item j, and for all the items rated by the active user a, we can compute the probability distribution of the active user's rating for item j using the noisy sensor model as described in the previous chapter.

When the linear relationship exceeds the maximum value of the rating scale, we use the maximum value and when it is lower than the minimum value of the rating scale, we use the minimum value.

To predict a rating (for example, to compare it with other algorithms that predict ratings), we predict the expected value of the rating. The expected value of the rating is defined as follows:

$$E\left(S_{aj}\right) = \sum_{i=1}^{m} Pr\left(S_{aj} = v_i | \left(S_{1j}, \ldots, S_{Nj}\right) \wedge \left(S_{a1}, \ldots, S_{aM}\right)\right) * v_i$$

4.1.1 K dummy observations

While trying to fit lines with sparse data, we often find a perfect linear relationship, even though the sensor isn't perfect. If there is a perfect fit between users a and u then the variance σ^2 will be zero according to the above calculations. Therefore, the sensor u's prediction for item j will be perfect or deterministic; that is, the conditional probability table associated with sensor u will be purely deterministic. We do not want this for our noisy sensor model because the deterministic sensor will discount the effect of other sensors.

We hypothesize that the above mentioned problem can be overcome if we add K dummy observations along with n observations (co-rated item). We assume that active user a and user u give ratings over K dummy items $(K > 0)$ in such a way that their ratings for K dummy items are distributed over all possible rating pairs. For m scale rating data there are m^2 possible combination of the rating pairs. We compute the prior distribution of each rating pair by the frequency of that rating pair in the training data. That is, in the training data for all user pairs we count the occurrence of their rating pairs over their co-rated items and then we normalize this to compute the prior distribution each rating pair. We use the prior distribution of rating pairs for distributing the effect of K dummy points over all rating pairs like a *hierarchical prior*, which reduces our ability to guarantee that the ratings for K items are distributed overall possible rating pairs.

For example, if users a and u co-rated 2 movies in EachMovie database. Suppose for the first movie the rating of user a is 1 and the rating of user u is 2, and for the second movie the rating of user a is 3 and the rating of user u is 5. So, the observed rating pairs are: (1,2) and (3,5). In EachMovie database the ratings are on a scale of 0 to 5. The possible number of rating pairs are 36

($(0,0), (0,1), \ldots, (5,5)$). If the prior of each rating pair is same (i.e. prior = 1/36) then effect of $K = 1$ (1 dummy point) on each pair is 1/36. Now, there is already one observed point at (1,2) and (3,5), therefore, the effect of (1,2) and (3,5) rating pair is (1+1/36) and the effect of the rest of the rating pairs is 1/36.

For calculating the coefficients α, β and σ^2 we need the values of $\sum y_i$, $\sum x_i$, $\sum x_i y_i$, $\sum x_i^2$ and $\sum e_i^2$. Suppose h_i denotes the prior distribution of the i^{th} rating pair and X_i, Y_i denotes the x and y values of the i^{th} rating pair. The expressions for $\sum y_i, \sum x_i, \sum x_i y_i, \sum x_i^2$ and $\sum e_i^2$ can then be written as follows:

$$\sum y_i = \sum_{i=1}^{n} y_i + \sum_{i=1}^{m^2} (K * h_i) * Y_i$$

$$\sum x_i = \sum_{i=1}^{n} x_i + \sum_{i=1}^{m^2} (K * h_i) * X_i$$

$$\sum x_i y_i = \sum_{i=1}^{n} x_i y_i + \sum_{i=1}^{m^2} (K * h_i) * X_i Y_i$$

$$\sum x_i^2 = \sum_{i=1}^{n} x_i^2 + \sum_{i=1}^{m^2} (K * h_i) * {X_i}^2$$

$$\sum e_i^2 = \sum_{i=1}^{n} (y_i - \alpha - \beta x_i)^2 + \sum_{i=1}^{m^2} K * h_i * (Y_i - \alpha - \beta X_i)^2$$

We did experiments with different values of K and its effect on the performance, explained in the next section.

4.1.2 Selecting Noisy Sensors

To determine the prediction for an active user we could use all items and users, but we do not need to. Instead, we investigate using only the most reliable item and user sensors. For determining the reliability of the noisy sensors, we consider the **goodness of fit** of the fitted regression line to a set of observations. we use the coefficient of determination r^2 [Guj95], is a measure that tells how well the regression line is fitted to the observations. It measures the proportion or percentage of the total variation in the dependent

variable explained by the regression model. r^2 is calculated as follows [Guj95]:

$$r^2 = 1 - \frac{\sum e_i^2}{\sum (y_i - \overline{y})^2}$$

where $\overline{y}$ is the mean of the ratings of user u.

The value of r^2 lies between 0 and 1 ($0 \leq r^2 \leq 1$). If $r^2 = 1$, there exists a perfect linear relationship between the preferences of active user a and user u; that is, $e_i = 0$ for each observation (co-rated item). On the other hand, if $r^2 = 0$, it means there is no linear relationship between users a and u and the best fit line is a horizontal line going through the mean $\overline{y}$. We order the user and item noisy sensors according to r^2. We use the best U user noisy sensors and best I item noisy sensors for making the prediction. The parameters setting for U and I is explained in the next section.

4.1.3 Variant *AlgoEqual* of *AlgoLinear*

The problem with *AlgoLinear* is that we are often fitting linear relationship with very little data (co-rated items). It may be better to assume *a priori* the linear model and then just learn the noise. The algorithm *AlgoEqual* is based on the idea that different users rate the items same. We assume that the preferences of active user a and another user u are the same; that is, the expected value of user u's preference of any item is equal to active user a's preference for that item.

$$E(y_i | x = x_i) = \mu = x_i$$

We learn the variance of user u's prediction. The algorithm *AlgoEqual* can be derived from algorithm *AlgoLinear* by making the regression coefficients

$\alpha = 0$ and $\beta = 1$. We also add the K dummy observations because the same problem can arise in *AlgoLinear*.

We are not fitting the relationship between active user a and user u, but we assume an equal relationship. In this case it doesn't make sense to use the coefficient of determination r^2 for finding the reliability of the noisy sensors. To find the reliability of the noisy sensor, therefore, we use the variance; the less variance the more reliable the noisy sensor. We use the best U user noisy sensors and best I item noisy sensors for making the prediction. The parameters setting for U and I is explained in the next section. The addition of K dummy point also helps in finding the good reliable sensors based on more co-rated items. For example, if two users have few co-rated items, the effect of K dummy points becomes significant and the variance will be more. For large n or more co-rated items, the effect of K dummy points is insignificant.

We evaluate empirically both *AlgoLinear* and *AlgoEqual* with state-of-the-art techniques on EachMovie database. The evaluation criteria, the various protocols, the dataset used in the analysis, and the experiments are all presented in the next section.

4.2 Evaluation Framework

The most common approach taken to evaluating the accuracy of collaborative filtering algorithms is the *training and test set* approach. In this approach, the dataset of users (and their ratings) are separated into two: a *training set*, which is used as the collaborative filtering dataset; and a separate *test set*, which is used to evaluate the accuracy of the collaborative filtering algorithm. We treat each user from the test set as the *active* user. To carry out testing, we divide the ratings for each test user into a set of ratings that we treat as

observed, I_a, and a set that we will attempt to predict, P_a. We attempt to predict the ratings in P_a using a CF algorithm, observed ratings, and training data.

4.2.1 Metrics

To evaluate the accuracy of collaborative filtering algorithm we use the following metrics:

Coverage metrics evaluate the number of items for which the collaborative filtering algorithm could provide the prediction. The coverage is computed as the percentage of items for all users of which a prediction is requested and for which the system is able to produce a prediction. Maximal coverage may be less than 100% coverage because there may be no ratings in the data for certain items, or because very few people rated an item, or because all user sensors had zero co-rated items with the active user. All experimental results shown in this thesis have maximal coverage by design.

Statistical accuracy metrics evaluate the accuracy of the collaborative filtering algorithm by comparing the predicted rating against the user observed rating for the items that have both predicted and observed ratings. Breese et al. [BHK98] and Pennock et al. [PHLG00] both have used average absolute deviation to measure the performance of the collaborative filtering algorithm. Other metrics that have been used are root mean squared error [RIS+94]. Both the above metrics generally provide the same conclusions. In this thesis we only report average absolute deviation as it is the more commonly used metric.

We calculate the average absolute deviation of the predicted rating against the actual rating of items by users in the test set. That is, if the

number of predicted ratings in the test set for the active user is n_a, then the average absolute deviation for a user is given as follows:

$S_a = \frac{1}{n_a} \sum_{j \in P_a} |p_{a,j} - r_{a,j}|$,

where P_a is the test set for active user a, p_{aj} is active user a's observed rating for item j and r_{aj} is active user a's predicted rating for item j.

These absolute deviations are then averaged over all users in the test set of users. This metric measures how accurate the collaborative filtering algorithm is and gives an estimate of the average error a user of the collaborative filtering algorithm can expect. The lower the average absolute deviation, the more accurate the collaborative filtering algorithm is.

Statistical significance is assessed for the difference in average absolute deviation between two algorithms using the randomization paired sample test of differences of mean [Coh95]. The sampling distribution of the mean difference between two algorithms is achieved by repeatedly shuffling and recalculating the mean difference in 10,000 different permutations. The shuffling involves reversing the sign of the difference score for each sample with probability of .5. The probability of achieving a difference less than or equal to the mean difference is reported as a result. That is, the probability of incorrectly rejecting the null hypothesis that the deviation scores of both algorithms arises from the same distribution [Coh95].

4.2.2 Data and Protocols

We compared the explicit multi-valued versions of our noisy sensor model to *PD* (Personality Diagnosis) which its authors recently compared to a number of other memory and model based algorithms [PHLG00]. To compare the performance we used the same subset of the EachMovie database as Breese

et al. [BHK98] and Pennock et al. [PHLG00] had used, consisting of 1,623 movies, 381,862 ratings, 5,000 users in the training set, and 4,119 users in the test set. On average, each user rated about 46.3 movies. We also tested the algorithms on other subsets to verify that our findings are not overly biased by the peculiarities of the subset.

To understand the effect of the amount of data on the prediction accuracy of the collaborative filtering algorithm, we ran experiments to examine how well the algorithm predicts given the different amounts of information about a user's preferences. As in [BHK98] for the *AllBut1* protocol, we withheld a single randomly selected rating for each user in the test set, and tried to predict its value given all other ratings by the user. As in [BHK98], for each user in the *test set*, for each *GivenX* protocol, we selected X items at random from these the user has rated. Using these selected ratings as the observed ratings, we then attempted to predict ratings for the remaining items. We did the experiments for $X = 2, 5$ and 10. With fewer items rated we expect a decrease in accuracy. As in [BHK98], users who have not rated enough items to satisfy a protocol are eliminated from that protocol, so the exact number of predictions in each *GivenX* protocol varies. For reporting the results we use the same observed ratings and test ratings for each test user of the test set as Pennock et al. [PHLG00]. We also tested the algorithms by randomly selecting the observed and test ratings for each test user of the test set.

4.2.3 Selecting K for *AlgoLinear*

The performance of *AlgoLinear* depends on the value of K. To select its value we did experiments with different values of K for all protocols. The value of K can not be zero because it can make a sensor deterministic (for details

see section 4.1.1). Figure 4.1 shows the effect of K for *AllBut1* and *Given10* protocols. Figure 4.2 shows the effect of K for *Given5* and *Given2* protocols.

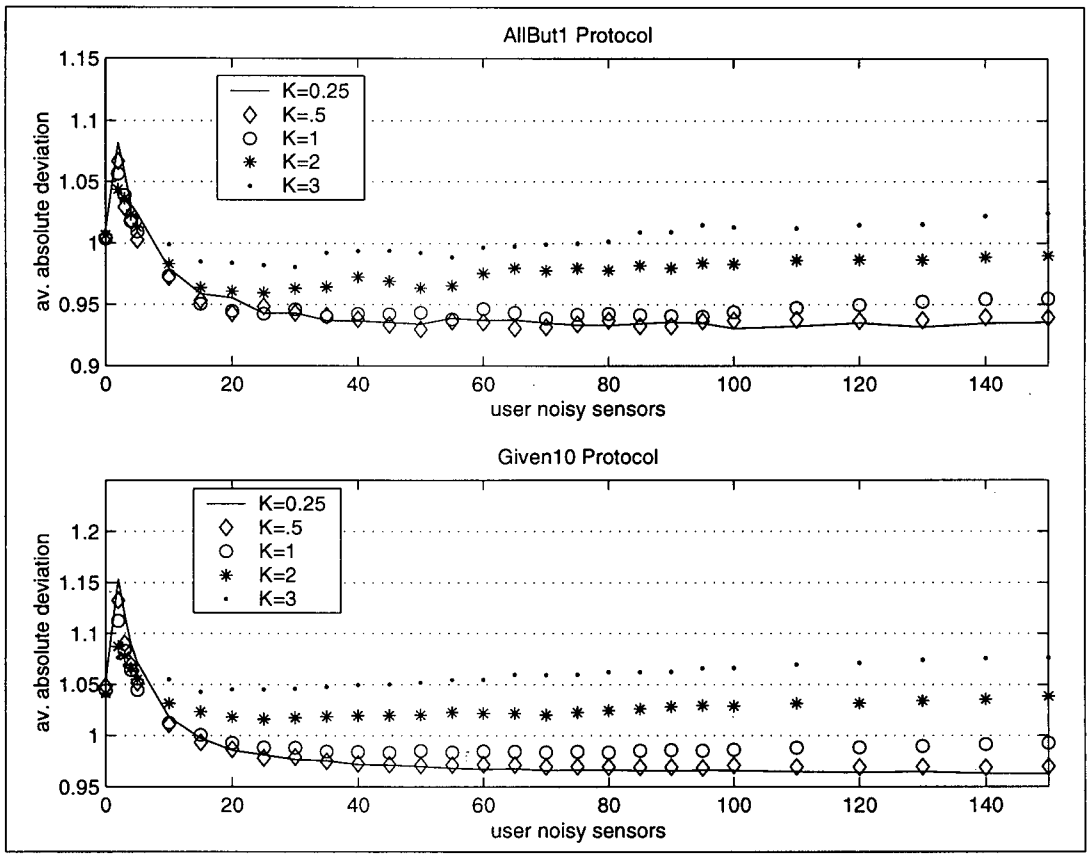

Figure 4.1: Effect of K for *AllBut1* and *Given10* protocols on *AlgoLinear*

The average absolute deviation error is more when $K > 1$ for all protocols. This is because the effect of the error introduced by K dummy points for large K is significant and hence, *AlgoLinear* is not finding good user sensors for the active user a. *AlgoLinear* works better for *AllBut1* protocol with $K = .5$ but for *Given10* and *Given5* protocols the performance is better with $K = .25$ but the difference is not significant. This is the expected behaviour because the effect of K dummy points for large K is more than the effect of the amount of information available in *GivenX* protocol, which decreases the

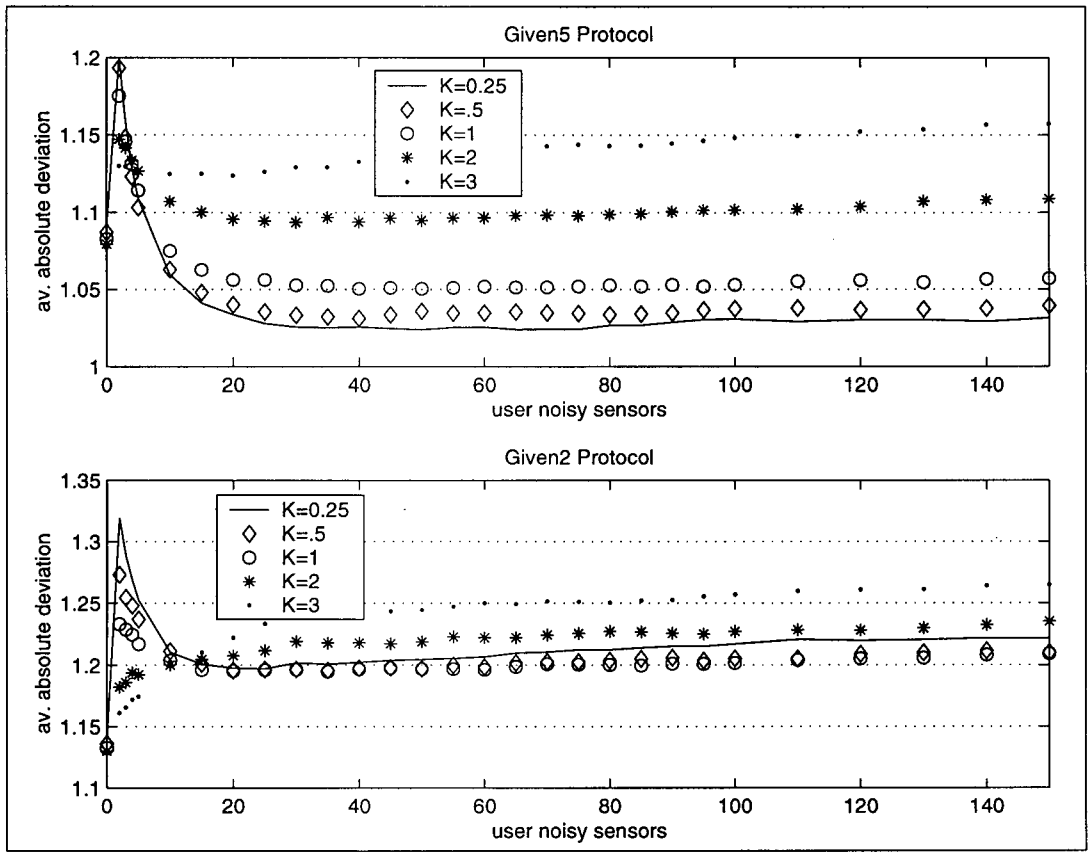

Figure 4.2: Effect of K for *Given5* and *Given2* protocols on *AlgoLinear*

accuracy of the algorithm.

The effect of K on *Given2* protocol is different than on other protocols because each user has rated only two items and co-rated items between users are less than or equal to 2. With so much less information *AlgoLinear* does not fit the good relationships between users and hence it is not finding good user sensors. We can also see from Figure 4.2 that for *Given2* protocol the *AlgoLinear* performs better when we consider only items as a noisy sensor, while the average absolute deviation error increases with the increase in number of user sensors. This is because the items are rated by many users and *AlgoLinear* fits them better.

Figure 4.2 shows that the algorithm gives better accuracy with K = .5 and K = 1 for *Given2* protocol when using user sensors 30 - 70. Based on all these experiments we found that overall *AlgoLinear* gives best performance with K = .5 for all protocols. For subsequent experiments we, therefore, chose K = .5 for *AlgoLinear*.

4.2.4 Selecting K for *AlgoEqual*

The performance of *AlgoEqual* also depends on K. The effect of K on the performance of *AlgoEqual* is different from the effect on *AlgoLinear* because in *AlgoEqual* we are not fitting the relationship between users. Instead, we are using K for learning the variance. To select the value of K we did experiments with different values for all protocols. Figure 4.3 shows the effect of K for *AllBut1* and *Given10* protocols. Figure 4.4 shows the effect of K for *Given5* and *Given2* protocols.

AlgoEqual performs best for the *AllBut1, Given10* and *Given5* protocols with $K = 1$ and $K = 2$. The error is more when $K < 1$ or $K > 2$. Figures 4.3 and 4.4 show that with a decrease in the amount of available information (for *GivenX* protocol) the performance of *AlgoEqual* improves for lower values of K. Figures 4.3 and 4.4 also show that there is no significant difference in the accuracy with $K = 1$ and $K = 2$ for *AllBut1, Given10* and *Given5* protocols. Figure 4.4 shows that *AlgoEqual* performs best for *Given2* protocol (for user sensor range thirty-to-seventy) with $K = .25$ to $K = 1$. Based on all these experiments we found that overall *AlgoEqual* gives best accuracy with $K = 1$ for all protocols. For subsequent experiments we, therefore, chose K = 1 for *AlgoEqual*.

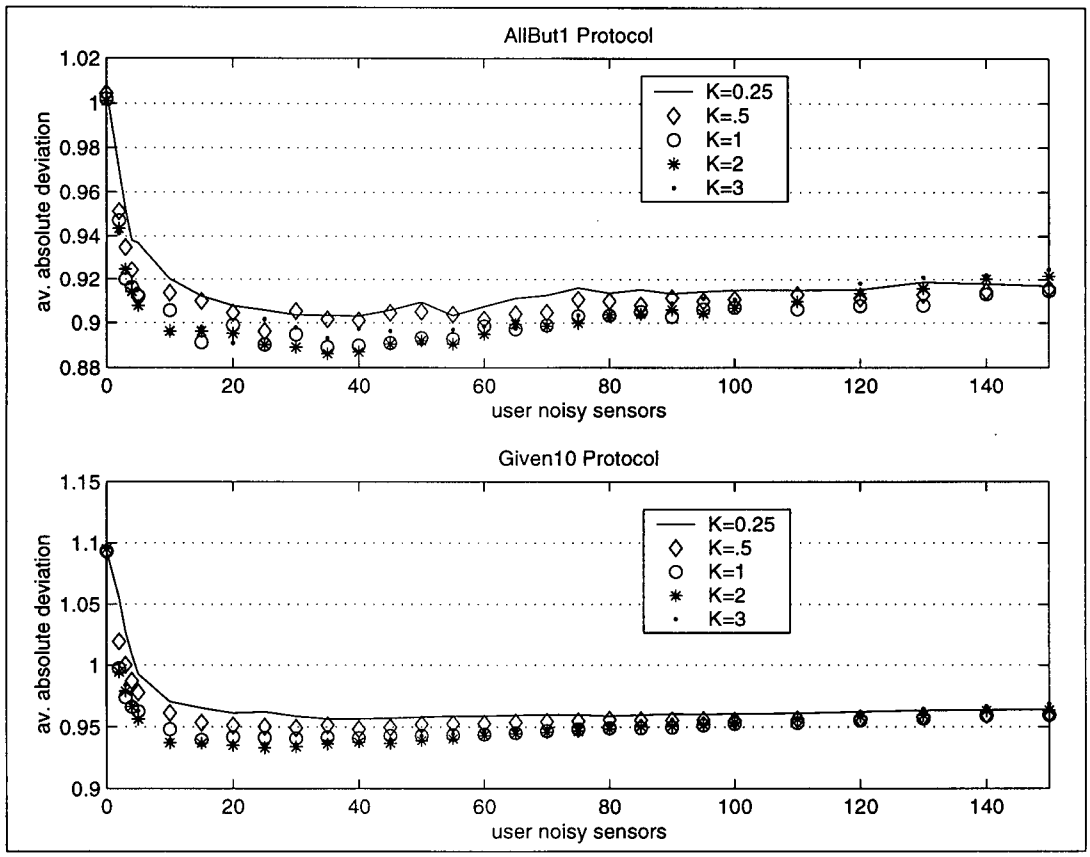

Figure 4.3: Effect of K on *AlgoEqual* for *AllBut1* and *Given10* protocols

4.2.5 Selecting Noisy Sensors

After learning the noisy sensor model we determine which noisy sensors should be used in making the prediction for the active user.

Figure 4.5 shows the variation of average absolute deviation with user noisy sensors for different numbers of items noisy sensors for *AlgoLinear*. Figure 4.6 shows the same for *AlgoEqual*. This experiment was performed on *AllBut1* protocol with $K = 0.5$. We used the same training and test set as described above but the test rating and the observed ratings for each test user of the test set were selected randomly.

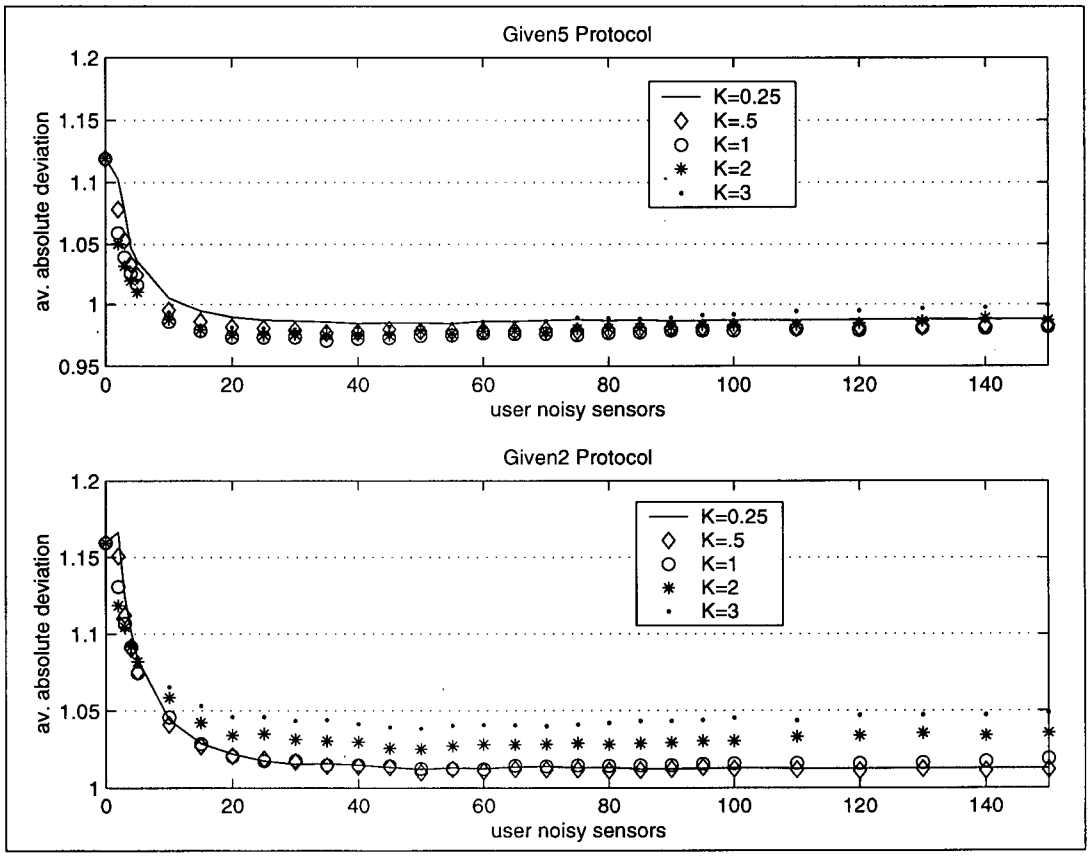

Figure 4.4: Effect of K on *AlgoEqual* for *Given5* and *Given2* protocols

Figures 4.5 and 4.6 show that the average absolute deviation of the prediction decreases with the increase in number of item sensors. There is no significant improvement in accuracy when the number of item sensors is more than twenty. It also shows that the average absolute deviation first decreases with the increase in number of user sensor and then increases as more user sensors are used for prediction. This is because the large number of user sensors results in too much noise for those user sensors that have good reliability.

From the experiments, we found that both algorithms give better performances with ten-to-twenty five item noisy sensors. We also found that algorithm *AlgoLinear* gives less average absolute deviation error with forty-

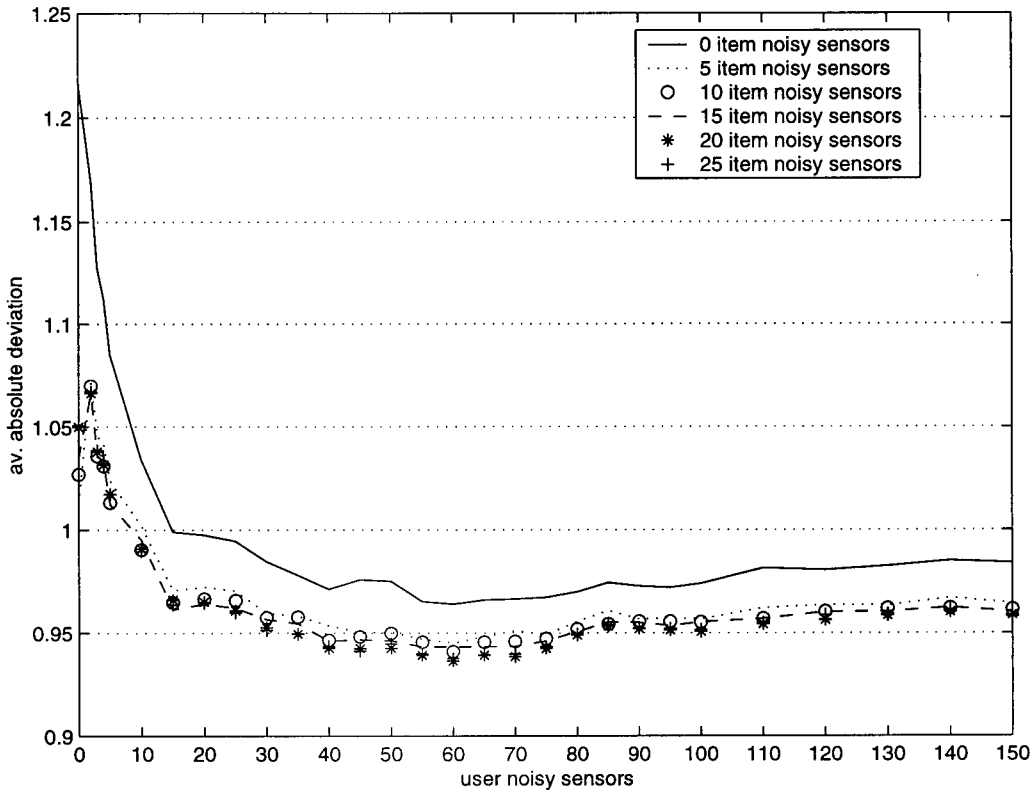

Figure 4.5: The average absolute deviation as a function of the number of best user noisy sensors for different numbers of best item noisy sensors. This experiment was performed using *AlgoLinear* and $K = 0.5$ for *AllBut1* protocol.

to-seventy user noisy sensors and *AlgoEqual* gives better performance with twenty-to-fifty user noisy sensors. For the experiments reported in the following section, we use the best fifty user noisy sensors and best twenty item noisy sensors (i.e. $U = 50$ and $I = 20$) for *AlgoLinear*. We use thirty user noisy sensorsand the best twenty item noisy sensors (i.e. $U = 30$ and $I = 20$) for *AlgoEqual*. The parameters U and I depend on the database in case of EachMovie database the number of users are more than the movies and each user has rated only few movies because of this more best user noisy sensors are selected than the best item noisy sensors.

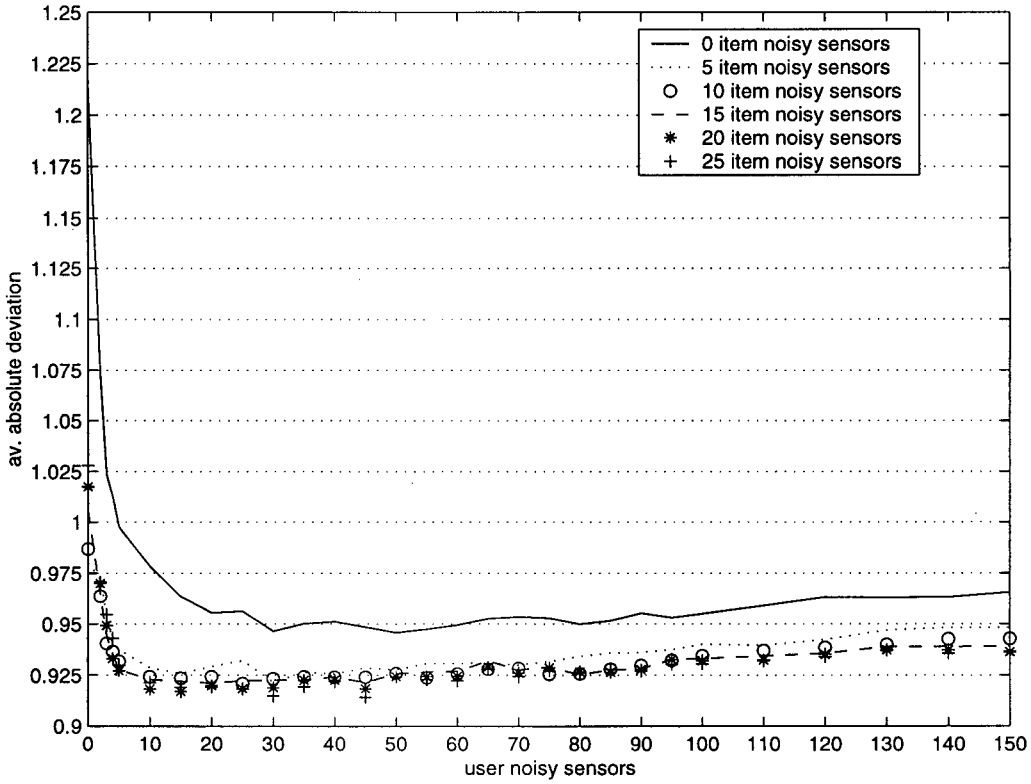

Figure 4.6: The average absolute deviation as a function of the number of best user noisy sensors for different numbers of best item noisy sensors. This experiment was performed using *AlgoEqual* and K =1 for *AllBut1* protocol.

From Figure 4.5 we also see that the minimum average absolute deviation is .936 when we use both user and item noisy sensors (with sixty user and twenty item noisy sensors). It is .964 when we use only the user noisy sensors, shown by the zero item noisy sensors case, and 1.027 when we use only the item noisy sensors, shown on the y-axis for ten item noisy sensors. This shows that by including the item noisy sensors along with the user noisy sensors the quality of the prediction improves considerably. It also shows that if we use only the item noisy sensors for prediction then the average absolute deviation becomes greater than when we use only user noisy sensors. There-

fore, symmetric collaborative filtering offers better accuracy than asymmetric collaborative filtering.

4.2.6 Comparison with Other Methods

We compared the models *AlgoLinear*, *AlgoEqual*, *Correlation* and *PD* using the same training and test set as Pennock et al. [PHLG00]. For each test user we tested the same item as Pennock et al.

The results of comparing *AlgoLinear*, *AlgoEqual*, *Correlation* and *PD* are shown in Table 4.1. We re-implemented *PD* and *Correlation.* Our results for *PD* match exactly with those reported in Pennock et al. [PHLG00], though not the case for *Correlation.* The coverages of *AlgoLinear, AlgoEqual* and *PD* are the same, but the coverage of *Correlation* is less. We believe that Pennock et al. [PHLG00] might have made some changes in *Correlation* to provide the same coverage as *PD.* Since we can not compare two collaborative filtering algorithms if their coverages are not same. We report the results for *Correlation* from Pennock et al. [PHLG00].

AlgoLinear performed better than *PD* and *Correlation* for *AllBut1* and *Given10* protocols. For *Given5* and *Given2* protocols *AlgoLinear* performance is better than *Correlation* but not better than *PD.* We believe that *AlgoLinear*'s poor performance can be explained by the fact the lines that are fitted to very small data sets are often a poor fit to the actual relationship. The algorithm *AlgoEqual,* based on an equal relationship between users, doesn't suffer from the same problem, outperformed better than each of the other three algorithms under all protocols.

Shardanand and Maes [SM95] and Pennock et al. [PHLG00] proposed that the accuracy of a collaborative filtering algorithm is most crucial when

Algorithm	*Protocol*			
	AllBut1	*Given10*	*Given5*	*Given2*
AlgoEqual	**.893**	**.943**	**.974**	**1.012**
AlgoLinear	.943	.983	1.021	1.196
PD	.964	.986	1.016	1.039
Correlation	.999	1.069	1.145	1.296

Table 4.1: Average absolute deviation scores on the subset of EachMovie database as Pennock et al. [PHLG00] for *AlgoLinear*, *AlgoEqual*, *PD* and *Correlation*, lower scores are better.

predicting extreme ratings (very high or very low ratings) for items. The idea is that most of the time people are interested in suggestions about items they might like or dislike, but not about items they are unsure of. Pennock et al. [PHLG00] define *extreme ratings*, as ratings which are 0.5 above the average rating or 0.5 below the average rating of the subset.

Algorithm	*Protocol*			
	AllBut1	*Given10*	*Given5*	*Given2*
AlgoEqual	1.001	**1.057**	**1.087**	**1.124**
AlgoLinear	**.997**	1.063	1.125	1.249
PD	1.0296	1.0874	1.128	1.163
Correlation	1.108	1.127	1.167	1.189

Table 4.2: Average absolute deviation scores on the EachMovie database for *AlgoLinear*, *AlgoEqual*, *PD* and *Correlation*, for extreme ratings 0.5 above or 0.5 below the overall average rating.

To compare the performance of algorithms with extreme ratings we computed the predicted ratings for those test cases (from the test sets of all protocols) whose observed rating is less than $\overline{R} - 0.5$ or greater than $\overline{R} + 0.5$ where $\overline{R}$ is the overall average rating in the subset. The results for the extreme ratings are shown in Table 4.2.

The results for extreme ratings show that *AlgoLinear* performs better than *AlgoEqual* for *AllBut1* protocol. It also performs better than *PD* and

Correlation over *Given10* and *Given5* protocols. *AlgoEqual* performs better than the other three algorithms over *Given10, Given5* and *Given2* protocols.

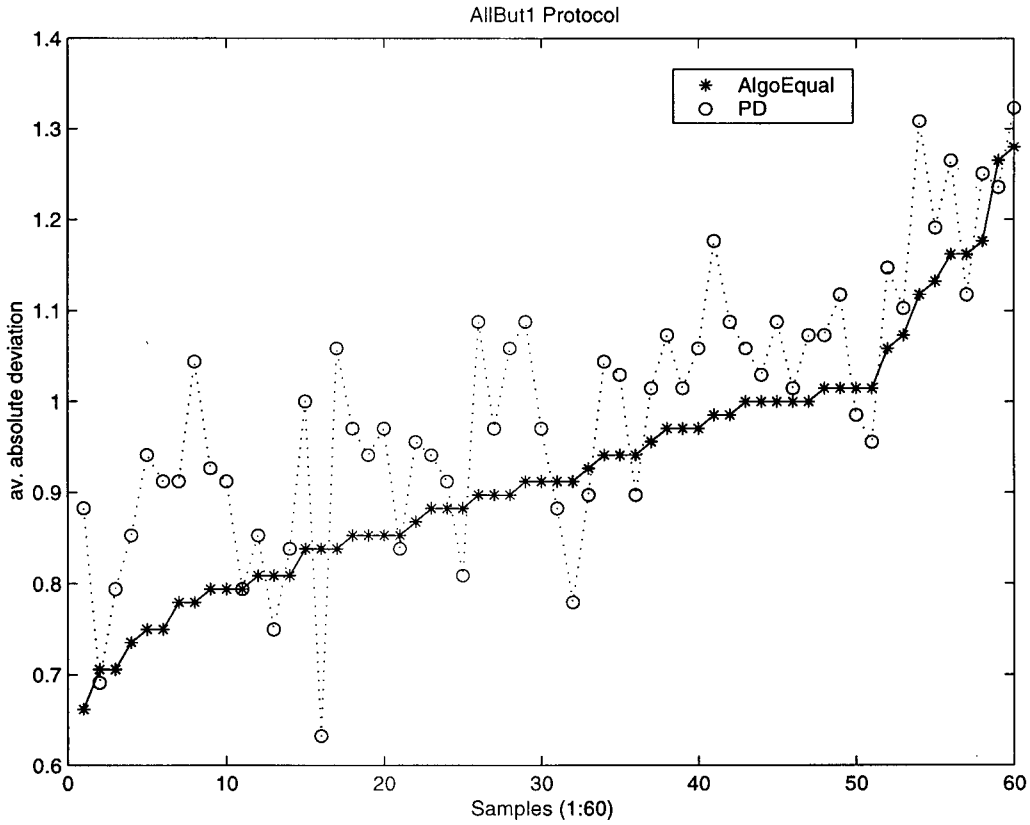

Figure 4.7: Average absolute deviation error of *AlgoEqual* and *PD* in 60 samples for *AllBut1* protocol with $K = .5$, $U = 30$, and $I = 20$.

To determine the statistical significance of these results, we computed the *significance levels* for the differences in average absolute deviation between *AlgoLinear* and *PD*, *PD* and *AlgoLinear*, *AlgoEqual* and *PD*, and *PD* and *AlgoEqual*, for all protocols. For doing this, we divided the test set for all protocols into 60 samples of equal size and use randomization paired sample test of differences of mean [Coh95] (for details please see Section 4.3.1). Figure 4.7 shows the average absolute deviation error of *AlgoEqual* and *PD* in 60

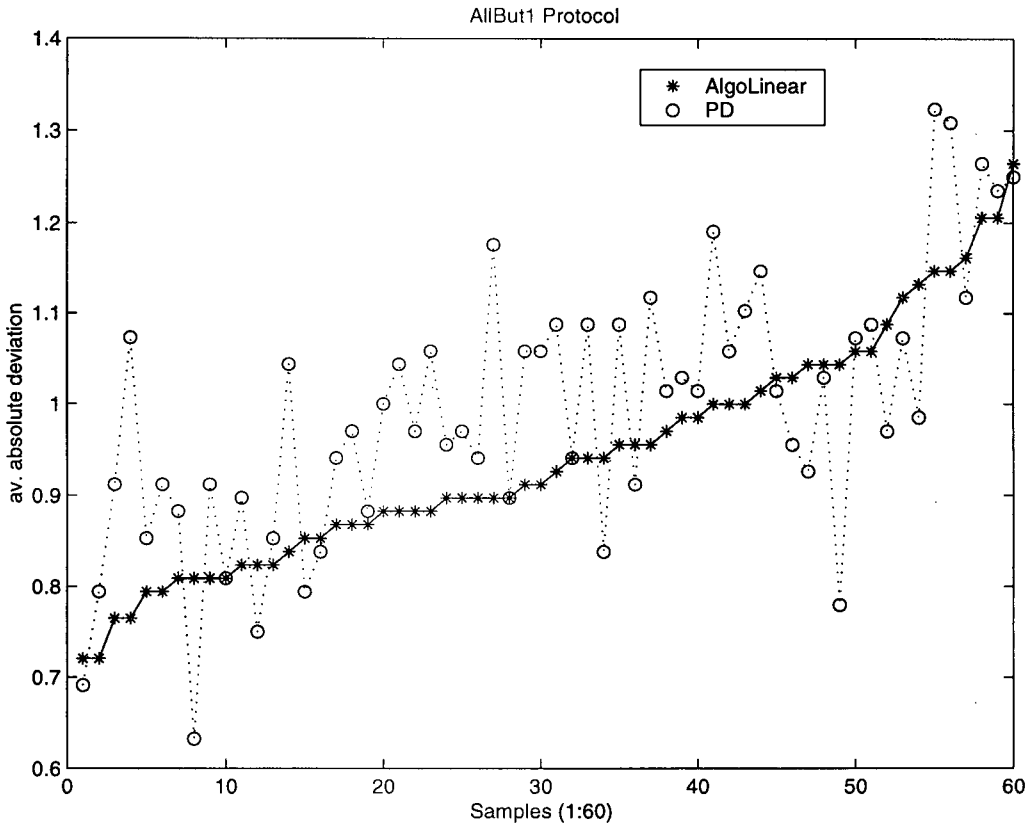

Figure 4.8: Average absolute deviation error of *AlgoLinear* and *PD* in 60 samples for *AllBut1* protocol with $K = 1$, $U = 50$, and $I = 20$.

samples for *AllBut1* protocol. Figure 4.8 shows the average absolute deviation error of *AlgoEqual* and *PD* in 60 samples for *AllBut1* protocol. We found similar results for the other protocols.

The statistical significance of the EachMovie data results is given in Table 4.3; it shows the probability of achieving a difference less than or equal to the mean difference. That is, it shows the probability of incorrectly rejecting the null hypothesis that both algorithms' deviation scores arise from the same distribution.

Recently, Pennock et al. [PHLG00] showed that *PD* is the most accurate

Protocol	*AlgoLinear* vs. *PD*	*PD* vs. *AlgoLinear*	*AlgoEqual* vs. *PD*	*PD* vs. *AlgoEqual*
AllBut1	**.0006**	.9994	**.0001**	.9999
AllBut1(extreme)	**.0003**	.9997	**.0127**	.9873
Given10	**.1377**	.8623	**.0001**	.9999
Given10(extreme)	**.0211**	.9789	**.0009**	.9991
Given5	.9064	**.0936**	**.0043**	.9957
Given5(extreme)	**.2897**	.7103	**.0001**	.9999
Given2	.9999	**.0001**	**.0019**	.9981
Given2(extreme)	.9999	**.0001**	**.0001**	.9999

Table 4.3: Significance levels of the differences in average absolute deviation between *AlgoLinear* and *PD*, and between *AlgoEqual* and *PD*, on EachMovie data computed using the randomization test. Low significance levels indicate that the difference in results are unlikely to be coincidental.

explicit multi-valued rating collaborative filtering algorithm. The statistical significance results from Table 4.3 show that the performance of *AlgoEqual* is better than *PD* over all protocols and the performance of *AlgoLinear* is better than *PD* for the *AllBut1* and *Given10* protocols.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

In this work, we have concerned ourselves with symmetric collaborative filtering based on explicit ratings, a technique that can be used for making recommendations for a user based on ratings of various items by a number of people.

We proposed a probabilistic approach using a *noisy sensor model* for *symmetric collaborative filtering*, which, to make predictions allows to consider both the user and the item similarity. We presented an explicit binary rating version of the noisy sensor model *AlgoBayes*, based on Bayesian Learning. We showed the performance results of *AlgoBayes* applied to several subsets of the EachMovie database, whch indicate that the *AlgoBayes* performs better than the *Correlation* and *Decision Tree* algorithms.

To predict recommendations for a user based on multi-valued rating data, we devised a mechanism for learning the noisy sensor model, assuming a linear relationship between the users' preferences. We presented two variants of the multi-valued rating versions of the noisy sensor model, *AlgoLinear* and

AlgoEqual. We compared the two algorithms with other collaborative filtering state-of-the-art techniques. We reported empirical results for the EachMovie database of movie ratings. According to the absolute deviation metrics, *Algo-Linear* makes better predictions than *PD* over *Allbut1* and *Given10* protocols, and better than *Correlation* over all protocols. *AlgoEqual* makes better prediction than *PD* and *Correlation* over all protocols. We analysed and confirmed the statistical significance of these results. Our results from symmetric collaborative filtering using noisy sensor model show that by including the items' similarity along with users' similarity the accuracy of prediction typically increases.

5.2 Future Work

This thesis has created many new opportunities for future work. This section suggests some specific avenues for future research may take.

The noisy sensor model has been shown to produce an average error of about 1, but the performance of this model degrades with the number of users and items. The noisy sensor model requires computation time that expands with both the number of users and the number of items. For large data sets it may become impractical to find all the reliable noisy sensors for an active user, except in an infrequent off-line manner. New technologies are needed to quickly produce high quality recommendations, especially for large data sets. Techniques of dimensionality reduction are currently being investigated in an attempt to reduce the amount of online computation. Example techniques are sampling of users, and singular value decomposition [BP98]. One could explore the use of dimensionality reduction techniques to improve the performance of noisy sensor model over a large database.

Collaborative filtering provides inaccurate predictions regarding users that are not similar to many other users. Also, using collaborative filtering system alone, the first users to rate an item cannot get a recommendation regarding it. Recently some initial work has been done to integrate content-based techniques with collaborative filtering and thereby alleviate the above problem [BS97]. One could explore the hybrid system of noisy sensor model with other existing content-based techniques.

The proposed noisy sensor model for collaborative filtering works very well with explicit rating data. We did not explore the use of noisy sensor model for implicit rating. An example of implicit rating would be a web page recommendation system in which we can log what pages a user sees but cannot collect explicit ratings [BHK98]. In this case, a simple way to get explicit rating would be to use a binary scale with the ratings *viewed* or *did not view*. While we believe that the noisy sensor model could accommodate this situation through an appropriate transformation of ratings, this is another interesting area for future research.

One limitation of the present model is the use of a linear relationship between the preferences of users, for learning the noisy sensor model. If a pair of users have highly similar preferences but their relation is nonlinear, this may not be recognized as a reliable or good sensor and may hence be rejected by the noisy sensor model when it is considering only the best noisy sensors for making the recommendations. One could explore the use of polynomial regression for finding the relationship between users' preferences. The polynomial regression may not be feasible if the data available is very sparse. However, polynomial regression may not be feasible if the data available is very sparse.

One could also extend the noisy sensor model by incorporating user and item properties beyond ratings — for example, user age group or movie genres

can be explored with the use of multiple linear regression for learning the noisy sensor model.

Finally, the proposed algorithms should be applied to a diverse set of prediction domains, to determine what results hold true generally across prediction domains. Unfortunately, sufficiently large test sets containing multi-valued ratings are not as yet, readily available. One could also test the proposed algorithms on the synthetic data. The synthetic data can be generated from the EachMovie database using the certain properties of the data, such as the same name of users, same number of items, and same average rating.

Bibliography

[BHK98] John S. Breese, David Heckerman, and Carl Kadie. Empirical ananlysis of predictive algorithms for collaborative filtering. In *proceedings of the 14th conference on Uncertainity in artificial Intelligence*, pages 43–52, July 1998.

[BP98] Daniel Billsus and Michael J. Pazzani. Learning collaborative information filters. In *Proceedings of the 15th International Conference on Machine Learning*, pages 46–54, July 1998.

[BS97] Marko Balabanovic and Yoav Shoham. Content-based, collaborative recommendations. In *Communications of the ACM, 40(3)*, pages 66–72, March 1997.

[Coh95] Paul R. Cohen. *Empirical Methods for Artificial Methods.* MIT Press, 1995.

[GNOT92] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. In *Communications of the ACM*, pages 61–70, December 1992.

[Guj95] Damodar N. Gujarati. *Basic Econometrics.* McGraw Hill, Inc., third edition, 1995.

[Hec95] David Heckerman. *A Tutorial on Learning with Bayesian Networks*, March 1995. Technical Report, MSR-TR-95-06.

[HKBR99] J. Herlocker, J. Konstan, A. Brochers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 1999 Conference on Research and Development in Information Retrieval*, Berkley, CA, August 1999.

[Mit97] Tom M. Mitchell. *Machine Learning.* McGraw Hill, Inc., 1997.

[PHB00] David Poole, Holger H. Hoos, and Craig Boutilier. *Model-based Symmetric Collaborative Filtering*, February 2000. Technical Report.

[PHLG00] David M. Pennock, Eric Horvitz, Steve Lawrence, and C. Lee Giles. Collaborative filtering by personality diagnosis: A hybrid memory- and model-based approach. In *Proceedings of the 16th Conference on Uncertainty in Artificial intelligence*, pages 473–480, June 2000.

[RIS+94] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of ACM CSCW'94 Conference on computer supported cooperative work*, pages 175–186, 1994.

[RN95] Stuart Russell and Peter Norvig. *Artificial Intelligence A Modern Approach.* Prentice Hall, 1995.

[SKR99] J. Ben Schafer, Joseph Konstan, and John Riedl. Recommender system in e-commerce. In *Proceedings of the ACM Conference on Electronic Commerce (EC-99).*, pages 158–166, November 1999.

[SM95] Upendra Shardanand and Patti Maes. Social information filtering: Algorithms for automating "word of mouth". In *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*, pages 210–217, 1995.

[UF98] Lyle H. Ungar and Dean P. Foster. Clustering methods for collaborative filtering. In *Workshop on Recommendation Systems at the 15th National Conference on Artificial Intelligence*, July 1998.