Recommendations and Predictions With ϵ_t-greedy Active Learning

Sindhu Kutty (skutty) Fu Yu (yufu)

December 17, 2009

1 Introduction

Almost everyone has turned to the internet to vet a product before purchase. Some online retailers – like amazon.com – even provide you with recommendations for products you might like. In this age of online retailers and multiple anonymous raters, it is a worthwhile goal to use the pool of collective knowledge and opinions to guide product recommendations. The objective of recommender systems is to automate this process.

Recommender systems aim to extract, the often implicit, information about a user or an item to make product recommendations. This benefits both the retailer and the end-user. One of the goals of machine learning is to make predictions on previously unseen data by learning from past experience. Therefore, recommender systems provide a natural context for applying machine learning techniques. We can think of a recommender system as learning about the preferences of a user to model the user's preferences and to use this model to provide recommendations tailored to that user.

Active Learning is a sub-class of supervised learning problems where the labels on the data points are not available a priori. The learning algorithm can actively query the user to obtain labels on a subset of data points in order to improve its model of the user's preferences. However, there is a cost associated with obtaining each label from the user. The challenge in active learning is to minimize the number of active queries while maximizing the accuracy of predictions.

The ultimate goal of a recommender system is to present items to the user that the user will like (i.e. rate highly). However, in order to build an accurate model of the user's preferences we also need to query the user to learn about his/her preferences. Similar to the multi-armed bandit problem in learning, this presents an *exploration vs. exploitation* dilemma. In the exploration phase, the goal is to obtain labels (aka *ratings*) so as to improve the overall quality of predictions on the unlabeled data points (aka *items*). In the exploitation phase, the goal is to present the item with the highest predicted rating with the information obtained so far. In this work we employ active learning techniques to address this problem.

We present a modification to existing techniques in recommender systems that show an improvement in prediction accuracy relative to published results. We also present a recommender system that makes an adaptive exploration/exploitation tradeoff by shifting the emphasis from exploration to exploitation as we learn more about a user's preferences. Offline experiments on the MovieLens dataset [mle] are used to study the performance of our contributions. We also include some work spectral clustering and a probabilistic framework used for clustering.

2 Preliminaries and Related Work

In a supervised learning-based classification problem we are given a set of data points with labels. As discussed earlier, active learning pertains to sub-class of these problems where the learning algorithm can actively request labels on the unlabeled data. Consider the recommender problem with a finite set $\mathcal{N} = \{1, 2, \ldots, n\}$ of users and a finite set $\mathcal{M} = \{1, 2, \ldots, m\}$ of items such that a label $\mathcal{L} = \{1, 2, 3, 4, 5\}$ on an item is a function $f_i : \mathcal{M} \to \mathcal{L}$ where $i \in \mathcal{N}$. Our goal is to approximate f_c for a given user $c \in \mathcal{N}$; c is the *active user* who serves as the *target* for our recommender system. Suppose the target has previously labeled some items, $\mathcal{M}' \subset \mathcal{M}$. If we also have access to other users who have labeled the same items, we can use this information to predict the labels on items in $\mathcal{M} \setminus \mathcal{M}'$. This is the key idea in *collaborative filtering*. In order to aggregate this information, we define *similarity* between customers in terms of Pearson's correlation coefficient defined below.

The data is organized into a $\mathcal{N} \times \mathcal{M}$-dimensional matrix known as a *ratings matrix*. Then, for a fixed order of items, we can consider the vector (of ratings) for each $c \in \mathcal{N}$. Note that this is a partial realization of the function f_c. Pearson's Correlation Coefficient allows us to quantify how similar these vectors are in terms of their linear dependence on each other.

Let $c, c' \in \mathcal{N}$ and X be the set of items that have been rated by both. Then Pearson's Correlation Coefficient is defined as:

$$sim(c, c') = \frac{\sum_{x \in X}(y_{c,x} - \bar{y}_c)(y_{c',x} - \bar{y}_{c'})}{\sqrt{\sum_{x \in X}(y_{c,x} - \bar{y}_c)^2 \sum_{x \in X}(y_{c',x} - \bar{y}_{c'})^2}} \tag{1}$$

where $\bar{y}_c$ is the average of user c's ratings and $y_{c,x}$ is the user's rating for item x. $\bar{y}_{c'}$ and $y_{c',x}$ are similarly defined. Note that $sim(c, c') \in [-1, 1]$: $sim(c, c') = 1$ indicates that c and c' agree on the rating of all items that have been rated by both, while $sim(c, c') = -1$ corresponds to the case when a high rating by one corresponds to a low rating by the other, and $sim(c, c') = 0$ means that the two users are are uncorrelated.

We make predictions on the unlabeled items based on the ratings of other users whose preferences are similar to those of the target c, For a particular item, this is done by taking a weighted average over all users who have rated that item. The weights are assigned based on the similarity of the users to the target. In this setting the preferences of a perfectly negatively correlated user c' (i.e., with $sim(c, c') = -1$) is just as valuable as those of a perfectly positively correlated user c'' (i.e., with $sim(c, c'') = -1$).

For classification with classes with equally-spaced integral labels centered around 0, the prediction, which on an item x for user c is a continuous function given by [AT05]:

$$y^c_{pred}(x) = k \sum_{c' \in C^x} sim(c, c') y^{c'}(x) \tag{2}$$

where C^x is the set of users who have rated x, $c' \in C^x$ rated x as $y^{c'}(x)$ and k is the normalization constant given by $k = 1/\sum_{c \in C^x} |sim(c, c')|$. The *absolute* value of the similarity is used to assign equal importance to raters with positive and negative correlations with the target. In our application, we will modify the denition slightly to relax the assumption on the class labels. In [NA98] the authors use a weighted majority prediction metric based on learning a weight on each user. We use the original similarity measure for its demonstrated applicability to this dataset and domain.

So far, we have been considering how to make predictions based on a static ratings matrix. In order to populate this matrix further, we will need to query the target. Of course, more ratings from the target will allow us to compute similarity more accurately, which will in turn lead to better predictions. But as far as possible we want to query the target on items that (a) are most informative (i.e. they tell us the most about the user's preferences over all of the remaining unrated items) and, (b) the user will like. These goals are not necessarily aligned. In fact, this is the crux of the explorative vs. exploitative dilemma. The second goal of exploitation implies that we should always present the item with the highest prediction to the target, since it is the likeliest to be rated highly. Let us now consider the first goal.

If we consider the ratings by other users on the items as yet unrated by the target, it may be that some items have a wider spread of ratings than others. This would translate into greater uncertainty in their prediction. In other words, the other raters have widely differing ratings for these items which makes it difficult to predict a rating for these items by consolidating the preferences of the other raters. Getting the user's input on such an item via an active query reduces the prediction uncertainty and thus, improves our model of the user's preferences. One way to select such items is to compute the variance on the ratings of each item and pick the one with maximal variance[CGJ96]. That is, pick $x \in \mathcal{N}$, such that

$$\arg\max_x \frac{\sum_{c \in C^x} (y_x^c - \bar{y}_x)^2}{|C^x|} \tag{3}$$

where C^x is the set of users who have rated x, $y^c(x)$ is c's rating on x and $\bar{y}_x$ is the average rating on x by users in C^x.

In [RS07], the authors argue convincingly that this measure of uncertainty reduction is local. That is, picking an item with maximal variance does not guarantee that the system's overall uncertainty is reduced. In addition to reducing local uncertainty, it is also crucial to consider the global impact. Thus, it is also important to consider how much knowing a rating on this item will effect predictions on other items. They suggest using a measure they call *influence*.

Intuitively, influence of an item is a measure of the effect of obtaining a rating on that item. It is computed by summing over the change in predictions of the unrated items for the target. Influence of an unrated item x, for a target c, is defined as

$$I(x) = \sum_{x' \in \mathcal{M}'', x \neq x'} \sum_{\delta} |y_{pred}^{Y_c}(x') - y_{pred}^{Y_{c,\delta}}(x')| \tag{4}$$

where $\mathcal{M}''$ is the set of all items not rated by c, Y_c is the set of all ratings by c and $Y_{c,\delta}$ is the set of all ratings by c including a hypothetical rating on x. This hypothetical rating is computed by making a prediction on x as usual, and shifting it by δ. They leave the range of δ unspecified and determine it experimentally. We handle this differently as described in the next section. $y_{pred}^{Y_c}(x')$ is the prediction on x' based on the set of ratings in Y_c. $y_{pred}^{Y_{c,\delta}}(x')$ is similarly defined. The item to be queried is picked as that which maximizes the product of variance and influence.

3 Proposed Methodology

3.1 Restricted Variance

In computing the variance of an unrated item, the existing technique is to consider the distribution of ratings over all users. Consider an item that has low variance over all users' ratings but has high variance amongst the users most similar to the target. Such an item will not be picked by the existing maximum variance method. However, if the similarity measure is meaningful, then it is intuitively clear that the uncertainty in the prediction on this item is still high. In order to correct for this, we modify the existing technique to restrict the computation of variance to those users who are most similar to the target. This will give a better sense of the local uncertainty inherent in the item.

3.2 Probabilistic Influence

In order to estimate the global impact of obtaining a rating on an unrated item, the influence measures the change in prediction induced over all other items as yet unrated by the user. This is done by including a hypothetical rating and summing over the change in prediction over each unrated item. In the case of five possible ratings, this would mean that the measure considers each of these five possibilities as equally likely. It is clear that this approach can get computationally expensive as the size of $\mathcal{L}$ grows. We propose overcoming this problem by using the information that we already know about the user to limit the number of possible labels that we need to consider. We assign probabilities to each of these events based on the current prediction for that item and the variance of its ratings. The probabilistic influence of an unrated item x is defined as

$$I(x) = \sum_{x' \in \mathcal{M}'', x \neq x'} \sum_{\delta \in \{1,2,3,4,5\}} p_{y^c_{pred}(x), modvar^c(x)}(\delta) |y^{Y_c}_{pred}(x') - y^{Y_{c,\delta}}_{pred}(x')| \tag{5}$$

where $y^c_{pred}(x)$ is the prediction of the target's rating on item x, $modvar^c(x)$ is the restricted variance of x with respect to the target and the other variables are as defined before. Using the information known about similar customers, their preferences and the spread of possible ratings allows us to reduce computation time by considering only those ratings δ, whose probability of occurrence lies above a predened threshold.

3.3 ϵ_t-greedy Active Queries

The intuition behind using active learning in the context of recommender systems is to pick the items to query that most improve the prediction over the rest of the unrated items. This involves presenting the item thus chosen to the target to rate. We call these active queries. We need to balance this with the fact that the ultimate goal of the recommender system is to, on the whole, present items that are tuned to the customers tastes. In other words, it is desirable to present an unrated item with the highest predicted rating as often as possible, while not sacrificing accuracy too much. We call these exploitative queries.

We now consider a solution to this exploration-exploitation tradeoff. As the algorithm learns more about the target, and thus builds a more accurate model of the target's tastes, we reduce the proportion of active queries to exploitative queries. Thus, we consider an ϵ_t-greedy algorithm that presents exploitative queries with increasing probability over time.

This idea in a multi-armed bandit setting was presented in [ACBF02]. We appropriate the idea of decaying ϵ with time in the recommender system context.

4 Experimental Design and Implementation

We use the Movielens dataset to run an offline experiment to test the methodology laid out in the previous section. Specifically, we compare:

- Standard variance vs. restricted variance for choosing items to actively query that reduce the uncertainty in the system
- Standard variance weighted by influence vs. restricted variance weighted by probabilistic influence for choosing an active query
- ϵ-greedy approach vs. a pure exploration (only active queries) and pure exploitation (only exploitative queries) approach.

The dataset consists of 100,000 ratings for 1682 movies by 943 users in no particular order. The ratings are in the set $\{1, \ldots, 5\}$. We construct a ratings matrix based on this dataset. A default rating of 0 indicates an unrated item. We will now present a brief sketch of the structure of the algorithm.

We fix four targets chosen at random from the dataset. These have each rated at least 100 movies. We fix these targets for ease of comparison across techniques. For each technique, we run the corresponding algorithm on each target separately.

For a given target we partition the rated items into training (the first half) and test (the latter half) data. As this is an offline experiment we withold the test data until it is actively queried by the algorithm. The items ratings that remain allow us to compute similarity of the target with each other user based on equation(1). We run each algorithm for 20 iterations, corresponding to 20 queries. Each learning algorithm picks an item to query on each iteration based on different criteria. To ensure a response on this query, we restrict the algorithms to choose from the set of rated but withheld items.

We then recompute the similarity vector based on this additional rating, where the i^{th} component of the similarity vector corresponds to similarity of the target to user i. Note that this set of known ratings can be (and is) very different for the different algorithms because of the distinct queries issued. The new similarity vector thus computed gives us a prediction for each of the still withheld items. When Pearson's correlation coefficient is undefined (for instance, in the case where the denominator is 0 because a rater has rated all items exactly the same), we define the similarity to be 0. This corresponds to a user uncorrelated with the target.

We compute the absolute error for target c over each iteration as $\sum_{x \in U^c} |y_{pred}(x) - y_{actual}(x)|$ where U^c are the items rated by c but not as yet queried and thus unseen by the algorithm. y_{pred} is the prediction based on the recomputed similarity vector. The prediction is computed by a translated version of equation(2) to correspond to equally spaced ratings centered around 0. In the anomalous case where no one other than the target has rated an item, we predict 0. Let T be the set of targets. We compute the Mean Absolute Error per rating over these targets as

$$\frac{1}{|T|} \sum_{c \in T} \sum_{x \in U^c} \frac{|y_{pred}(x) - y_{actual}(x)|}{|U^c|} \tag{6}$$

In order to compute modified variance, we define 'most similar customers' as those whose similarity with the target had an absolute value of at least one-half. Note that the continuous nature of the prediction function means that a prediction could lie in the range $[0, 5]$. For probabilistic influence of an item, we assume a normal distribution over the ratings with mean at the predicted value for that item and variance as computed by the restricted variance measure. This captures the fact that we have partial model of the target and would like to incorporate that into the weights of the hypothetical ratings in the influence computation. We then normalize the probabilities at each rating so that they sum to 1 and thus obtain a valid discrete probability distribution. In the restricted variance-probabilitic influence method, we compute variance and influence and pick the withheld item that maximizes the product.

In the ϵ_t-greedy algorithm, we simulate a biased coin flip with probability of heads $= \epsilon_t = 1/t$, where t is the query number. On a heads outcome we execute an active query based on restricted variance-probabilitic influence. Thus the probability of an active query decays with time which is consistent with the intuition that as we learn more about a user's preferences, the fewer additional questions we need to ask of him/her. In the exploitative phase we query the item with the maximum prediction. This is similar to the Softmax Action Selection algorithm [SB98], even in the exploration phase we utilize some knowledge of the user in using restricted variance.

5 Discussion of Results

In figure 1 we compare the standard variance weighted by influence measure against the restricted variance weighted by probabilistic influence measure. The restricted variance-probabilistic influence has a sharper gradient indicating that as we learn more about the target this measure outperforms the standard variance measure. We also include for comparison the outcome of doing random queries on the withheld items. As expected the change in accuracy is not immediately evident in that case and both measures outperform a blind query.

Figure 2 shows the advantage of including influence when picking an item to query actively. The reduction in error indicates that including a global measure of uncertainty helps ascertain the best items to query for improving prediction accuracy.

In figure 3 we compare the ϵ_t-greedy approach with pure exploration or only active queries and pure exploitation or only exploitative queries. In choosing active queries we used the restricted influence-probabilistic variance measure. As expected, the error obtained from using an ϵ_t-greedy lies between that of pure exploration versus pure exploitation. When we pick only exploitative queries, it is possible that the items that are picked have low informativeness and hence add little to the prediction of the as yet withheld items. This explains the arbitrary nature of the curve. With pure exploration, we are likely to pick informative items at the cost of presenting low rated items to the user. This is why the error drops in this case.

Note that the adversarial multi-armed bandit model is not directly applicable to this case since the regret (or prediction loss) is bounded with respect to the best expert (users other than the target) over all targets. This is not a convincing bound, as it is possible no one user would provide an accurate model of all targets. We also considered a weighted (rather than restricted) variance. However, this approach appears not to fit the dataset as well and this line of investigation was abandoned as the preliminary results obtained

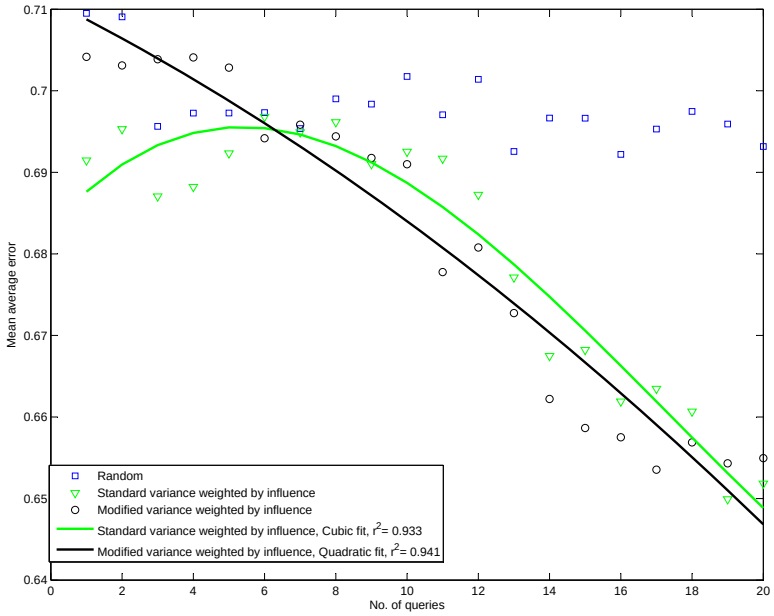

Figure 1: Active queries that reduce uncertainty outperform random active queries. Our approach is better than the existing approach.

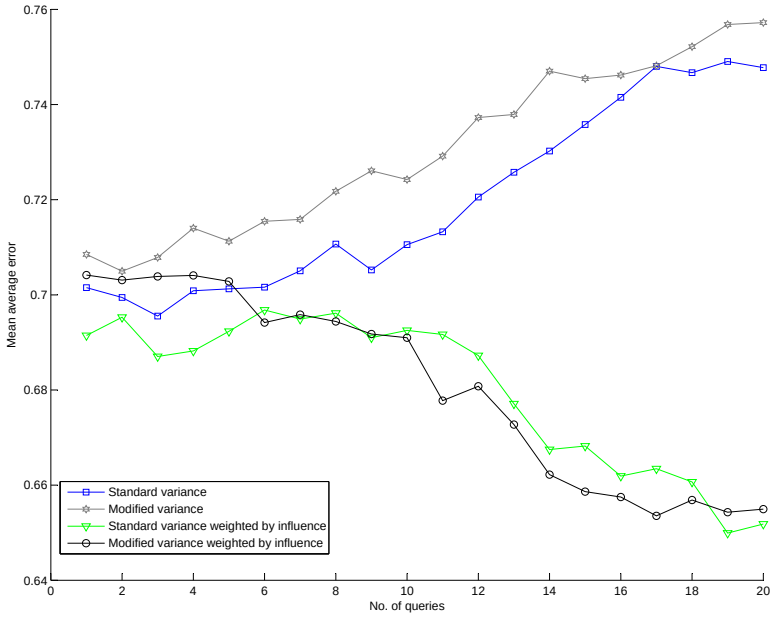

Figure 2: Variance weighted by influence outperforms only variance in choosing active queries.

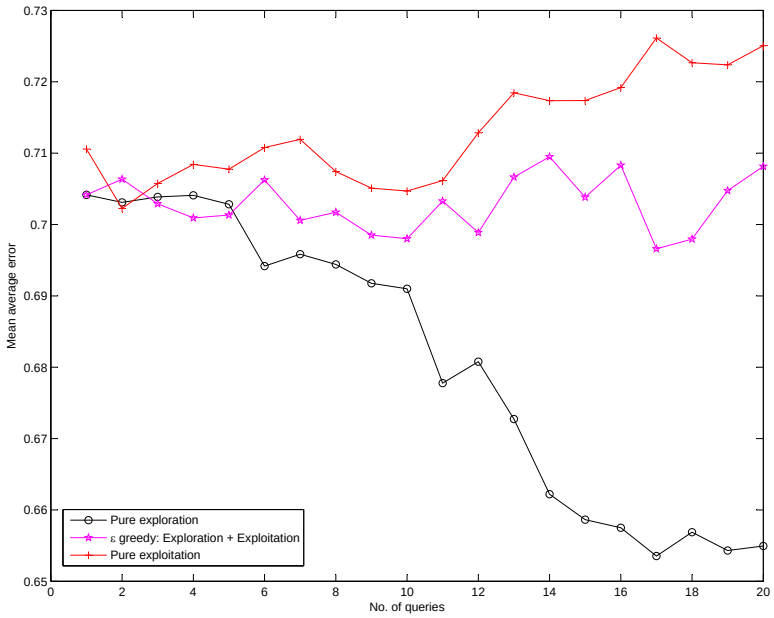

Figure 3: ϵ_t-greedy outperforms pure exploration, but is outperformed by pure exploration as expected.

were uninspiring.

6 The probability framework

Beyond the collaborative filtering, we also examine a probability framework described in [PHLG00] to incorporate the inferenced errors.

The true rating for a person i may be described as

$$R_i^{true} = \langle R_{i1}^{true}, R_{i2}^{true}, \ldots, R_{im}^{true} \rangle$$

Where $1, 2, \ldots, m$ stands for the index of the rated articles. We regard this as the internal perference for the articles and the actual rate would depends on this based on certain model. That is, if we assume the model is Gaussian, we can write

$$P(R_{ij} = x | R_{ij}^{true} = y) \propto e^{-(x-y)^2/2\sigma^2}$$

where σ is the parameter. In our work, we treat it as our confidence of our inference.

For the prior probability of the rating distribution, the simplest assumption is the uniform distribution. If we think R_a^{true} is a random variable taht can takes one of the n possible values, $R_1, R_2, \ldots, R_n$, we can write

$$Pr(R_a^{true} = R_i) = \frac{1}{n}$$

.

However, I think from the statistics of all the ratings or the personal psychological reaction, we can get a better prior distribution describing the possibability.

From the above two equations, by applying Bayes' rule, we can compute the probability that the active user is of the same personality type as another user:

$$\begin{aligned} Pr(R_a^{true} = R_i | R_{a1} = x_1, \ldots, R_{am} = x_m) &\propto Pr(R_{a1} = x_1, \ldots, R_{am} = x_m, R_a^{true} = R_i) \\ &\propto Pr(R_{a1} = x_1 | R_{a1}^{true} = R_{i1}) \ldots \\ &Pr(R_{am} = x_m | R_{am}^{true} = R_{im}) Pr(R_a^{true} = R_i) \end{aligned}$$

Here, we have a reasonal assumption. When we know the true perference, then all the ratings are independent, since the ratings only depends on the person's internal preference.

Using the above posterior probability, we can inference user j's rating based on the probability that other users is the same to he or she.

$$\begin{aligned} Pr(R_{aj} = x_j | R_{a1} = x_1, ..., R_{am} = x_m) = &\sum_{i=1}^{n} Pr(R_{aj} = x_j | R_a^{true} = R_i) \\ &\bullet Pr(R^{true} = R_i | R_{a1} = x_1, \ldots, R_{am} = x_m) \end{aligned}$$

We implement this probability framework for our further experiments.

6.1 Clustering

A problem arises naturally when we want to do active learning. In active learning, one part is to category users and do the exploration query to reduce the uncertainty of certain articles. That is, it would be very helpful if we can cluster the users accurately.

After calculating the similarity between each pair of users using Pearson's Correlation Coefficient, we can build the similarity matrix for all the active users. Next step, we try to use this similarity as a measure of the distance of two users and use spectral clustering to get the user clusters.

However, the result of spectral clustering is not satisfatory and it doesn't show a reasonable clustering result.

7 Conclusion

We presented an algorithm that uses a novel technique for solving the exploration-exploitation problem in recommender systems. We use a new measure of variance and probabilistic influence that exploits domain specific knowledge to learn a model for a target user in order to best predict his/her ratings. We use these projected ratings to make recommendations. We also use an ϵ_t-greedy technique to balance making the best recommendation with making the best predictions. With more time and computational resources, it would be worthwhile to run the algorithm on the larger dataset provided by Movielens on more targets. We could also try varying the ϵ_t value to determine the best fit for this particular domain.

7.1 Individual Contribution

Sindhu defined the problem and scope of this project. She also wrote the proposal, the progress report and this document except for Section 6. She defined the methodology

described in the rest of the report and defined the algorithms, wrote and ran the code to test it. Fisher did the write-up for Section 6. He also tested spectral clustering based on Pearson's correlation coefficient.

References

[ACBF02] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.*, 47(2-3):235–256, 2002.

[AT05] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowl. and Data Eng.*, 17(6):734–749, 2005.

[CGJ96] David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. Active learning with statistical models. *J. Artif. Int. Res.*, 4(1):129–145, 1996.

[mle] Movielens dataset: http://www.grouplens.org/node/73.

[NA98] Atsuyoshi Nakamura and Naoki Abe. Collaborative filtering using weighted majority prediction algorithms. In *ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning*, pages 395–403, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[PHLG00] David Pennock, Eric Horvitz, Steve Lawrence, and C. Giles. Collaborative filtering by personality diagnosis: A hybrid memory- and model-based approach. In *Proceedings of the Proceedings of the Sixteenth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-00)*, pages 473–4, San Francisco, CA, 2000. Morgan Kaufmann.

[RS07] Neil Rubens and Masashi Sugiyama. Influence-based collaborative active learning. In *RecSys '07: Proceedings of the 2007 ACM conference on Recommender systems*, pages 145–148, New York, NY, USA, 2007. ACM.

[SB98] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.