Signals in the Silence: Models of Implicit Feedback in a Recommendation System for Crowdsourcing

Christopher H. Lin*
Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195
chrislin@cs.washington.edu

Ece Kamar and **Eric Horvitz**
Microsoft Research
Redmond, WA 98052
{eckamar,horvitz}@microsoft.com

Abstract

We exploit the absence of signals as informative observations in the context of providing task recommendations in crowdsourcing. Workers on crowdsourcing platforms do not provide explicit ratings about tasks. We present methods that enable a system to leverage implicit signals about task preferences. These signals include types of tasks that have been available and have been displayed, and the number of tasks workers select and complete. In contrast to previous work, we present a general model that can represent both positive and negative implicit signals. We introduce algorithms that can learn these models without exceeding the computational complexity of existing approaches. Finally, using data from a high-throughput crowdsourcing platform, we show that reasoning about both positive and negative implicit feedback can improve the quality of task recommendations.

Introduction

Although the growth of crowdsourcing has facilitated an unprecedented explosion of open-call work, such growth has also brought to the fore new inefficiencies and opportunities for optimization. For example, the large number of tasks presented in an unorganized way in marketplaces like Amazon's Mechanical Turk, where over 1,300 types of tasks can be available at a given time [1], can make it challenging for crowdworkers to identify tasks that they enjoy or that they have special competency with solving. Such problems with task identification can reduce overall throughput, accuracy, and engagement in a crowdsourcing system.

We present methods that help crowdworkers discover tasks that they have not yet worked on, but would be interested in performing. A key challenge in crowdsourcing is that users do not typically provide explicit feedback about their preferences by asserting which tasks they like and dislike through ratings. To help workers find desirable tasks, we explore methods for leveraging the absence of explicit signals in task recommendation for crowdsourcing. We take inaction in the face of displayed information on available tasks as evidence of preferences. Rather than take a content-based approach, we build a collaborative filter (Goldberg et al. 1992). A collaborative filtering methodology can handle the inherent diversity of crowdsourcing tasks; the creativity encouraged by crowdsourcing makes it difficult to create explicit profiles using a fixed set of features for tasks and users.

Observing the behaviors of users in the crowdsourcing marketplace provides evidence about users' preferences in the form of *implicit feedback*. Previous work in collaborative filtering uses positive implicit feedback in the context of a TV recommendation system, by representing the degree to which a viewer likes a certain TV show with the number of times that viewer watches the show (Hu, Koren, and Volinsky 2008). While we demonstrate that this approach can be used to generate crowdsourcing recommendations, we show that it has a significant shortcoming: the prior approach does not address *negative* implicit feedback. For example, it does not model when viewers may dislike certain shows, leading to a system that only learns the shows that users like and not the shows they dislike.

With crowdsourcing, worker behavior in the marketplace provides both positive and negative implicit feedback that can be used to learn more comprehensive models of workers' preferences. Workers who complete large quantities of a task provide a signal that they have a preference for that task, whereas workers who do not work on tasks that have high availability on the system provide a signal that they may not desire those tasks. We present a general-purpose model that can represent both positive and negative implicit feedback. Our model is expensive to learn with existing approaches. To address this computational challenge, we propose a modified coordinate-descent algorithm for tractable learning. We show that the complexity of this approach does not exceed the complexity of learning simpler models that consider only positive implicit feedback. In addition, we propose an alternative sampling-based approach that can generate recommendations efficiently by combining multiple models learned independently via sampling subsets of the data.

We make the following contributions:

- We develop two new methods for incorporating negative implicit feedback into a predictive modeling system in a computationally efficient way.
- We present what we believe to be the first application of recommendation systems to crowdsourcing. We demonstrate how implicit signals can be extracted from the logs

*Research was performed while the author was an intern at Microsoft Research.

[1] retrieved February 2014

of a crowdsourcing platform.

- We evaluate the proposed approaches on a dataset collected from a live crowdsourcing platform and show that the addition of negative implicit feedback is beneficial.

Background on Collaborative Filtering

Two common types of methods used in collaborative filtering systems are neighborhood-based approaches and matrix factorization approaches. Neighborhood-based approaches (Herlocker et al. 1999) estimate ratings based on either the ratings of like-minded users or the ratings for similar items (Linden, Smith, and York 2003; Sarwar et al. 2001). Recent work has shown that matrix factorization approaches have superior performance compared to neighborhood-based models (Koren, Bell, and Volinsky 2009), so we focus on matrix factorization models but we present a comparison of these models in the experimental section.

We first review matrix factorization approaches for explicit feedback domains. Then, we move on to approaches for harnessing implicit feedback. Let m be the number of users and let n be the number of items. Let R be an $m \times n$ matrix where most entries are missing and each non-missing $r_{ij} \in R$ is an explicit rating given by user i to item j. The goal is to learn an $m \times k$ matrix U and a $k \times n$ matrix V that minimize the loss function

$$L(U,V) = \sum_{i,j}(r_{ij} - u_i v_j)^2 + \lambda(||U||_2^2 + ||V||_2^2)$$

where u_i denotes row i of U, v_j denotes column j of V, and λ denotes a regularization parameter. Then, $\hat{R} = UV$ contains the desired predictions. k can be picked by cross validation, and U and V can be estimated by a variety of methods, including stochastic gradient descent (Funk 2006) and alternating least squares (Bell and Koren 2007; Zhou et al. 2008). Learning U and V exactly is a non-convex minimization problem (Srebro and Jaakkola 2003).

When explicit feedback is not available, implicit feedback can be used to make predictions about users. We generalize the implicit feedback matrix factorization model (IFMF) proposed by Hu, Koren, and Volinsky to use any function of some positive implicit feedback signal as confidences in matrix factorization. Given a set of positive implicit feedback signals $c_{ij} \geq 0$, we fit and predict an $m \times n$ binary matrix P, where the entry $p_{ij} = 1$ if $c_{ij} > 0$ and $p_{ij} = 0$ otherwise. Let $w_{ij} = f(c_{ij})$ be weights for each entry determined by some weighting function f. Then, we find factors U and V that minimize the loss function:

$$L(U,V) = \sum_{i,j} w_{ij}(p_{ij} - u_i v_j)^2 + \lambda(||U||_2^2 + ||V||_2^2)$$

Intuitively, p_{ij} denotes whether or not user i likes item j and each of these entries is weighted by some function of the implicit signal we receive, representing the confidence we have in the signal. We can perform an iteration of alternating least squares in time $O(k^2\omega + k^3(n+m))$ (Hu, Koren, and Volinsky 2008) , where ω is the number of non-zero entries in P, which is usually much smaller than the total number of entries in P.

Harnessing Implicit Positive Feedback for Task Recommendation in Crowdsourcing

We use the terms workers and users interchangeably. Users may have different backgrounds and capabilities and may have preferences about which types of tasks to work on. A user works on an abstract *task* by completing at least one of a collection of instances of that task posted by a single requester.

We define the problem of task recommendation in crowdsourcing as follows: Given a set of users and a set of tasks, we would like to compile, for each user, a personalized recommendation list with tasks that the user has not worked on yet but that we predict the user would like to work on in the future. Since users cannot express explicitly which tasks they like and which tasks they dislike, we use implicit signals about their behaviors as evidence about their interests. Specifically, we take the number of times a worker completes a given task as a positive signal, such that more completions signal a stronger preference for the task.

IFMF can be applied to the problem of recommending crowdsourcing tasks as follows: Let c_{ij}, the implicit positive signal, be the number of times user i worked on task j. Let the weighting function f be $f(c_{ij}) = 1 + c_{ij}$. Intuitively, for every task j that user i has worked on, p_{ij} is set to 1, denoting that user i likes task j; and for any task j that the user has not worked on, p_{ij} is 0, denoting that user i dislikes task j. By weighting the positive entries of P with the number of tasks a user has completed, we are able to express our confidence in each $p_{ij} = 1$, and when we learn the factors U and V, we try harder to fit the entries we have higher confidence in. For instance, if a user has worked on a given task 10,000 times, then we are confident that the user likes that task ($p = 1$), and we would very much like to predict $\hat{p} = 1$.

Incorporating Implicit Negative Feedback

IFMF treats all tasks that a user has not worked on the same by assigning the minimum weight to the corresponding entries. Since this model cannot represent negative implicit feedback, the loss function has little emphasis on learning about the tasks that users dislike. We introduce a novel model, IFMF2, to address this shortcoming of the IFMF approach. IFMF2 uses IFMF as its building block and extends it in the following way: Let $p_{ij} = 1$ if $c_{ij} > 0$ and $p_{ij} = 0$ otherwise. Then, let $d_{ij} \geq 0$ be a set of negative implicit negative signals. Then, we redefine the weights to be

$$w_{ij} = \begin{cases} f(c_{ij}) & if \ \ p_{ij} = 1 \\ g(d_{ij}) & if \ \ p_{ij} = 0 \end{cases}$$

where f and g are normalization functions. Because c_{ij} and d_{ij} can have very different ranges, f and g can be used to transform confidences appropriately.

We apply IFMF2 to the crowdsourcing domain by using the availability of a task as a negative signal since ignoring a task with high availability may indicate a strong dislike towards the task. Formally, we let d_{ij} denote the maximum total number of task instances available for task j. We let f and g be logistic functions: $f(x) = \frac{1}{1+e^{\alpha_1(\log(x)+\beta_1)}}$ and $g(x) = \frac{1}{1+e^{\alpha_2(\log(x)+\beta_2)}}$ We choose logistic functions

for normalization because they capture our intuition about users. Consider the pairs of workers and tasks such that $p_{ij} = 0$. When there are very few tasks available for a given task type, we cannot be sure that the user does not like the task simply because the user has not worked on the task. The user may have just not seen it yet, because tasks with low availability have much lower visibility in crowdsourcing UIs. However, as the number of tasks available for a given task type increases, thereby increasing the visibility of that task type, we become more confident that the user is purposely avoiding that task. When the task is overwhelmingly available to the user, we become confident that the user does not like the task, even though the user has given no such explicit signal. Beyond a certain availability however, we do not substantially increase our confidence hence agreeing with the behavior of the sigmoid function. The reasoning is similar for the weights for the tasks that users select (when $p_{ij} = 1$).

Fast Optimization

We apply an alternating least squares approach to learn the U and V factors that minimize the squared loss function. By fixing one factor, say U, solving for the other factor, V, becomes a simple least-squares optimization and can be solved in a closed form. Then, by iterating through the factors, we effectively employ a coordinate descent to reach a local optimum. This approach is efficient if the matrix is sparse or can be made efficient when many entries have the same weight (Hu, Koren, and Volinsky 2008). When negative feedback is included, we increase the diversity of the weights and must develop new methods for maintaining tractability. We now present an algorithm that can learn with positive and negative feedback as efficiently as IFMF under certain conditions.

Theorem 1. *If one of the following conditions is true:*

1. *For all j, we have $d_{i_1 j} = d_{i_2 j}$ for all i_1, i_2*
2. *For all i, we have $d_{i j_1} = d_{i j_2}$ for all j_1, j_2*

then each iteration of coordinate descent takes at most time $O(k^2\omega + k^3(n+m))$ where ω is the total number of nonzero entries.

Algorithm (Proof): Our techniques are similar to those in (Hu, Koren, and Volinsky 2008). We assume without loss of generality condition (1) of the theorem. To solve for U when V is fixed, we begin by writing down the least-squares solution. Let W^i be the diagonal $m \times m$ matrix such that $W^i_{ll} = w_{il}$. This matrix contains all the weights for user i on the diagonal. Let p_i be the row in P corresponding to user i. Then,

$$u_i \quad = \quad (VW^iV^T + \lambda I)^{-1} VW^i p_i^T$$

The computation is dominated by VW^iV^T. We can speed up the computation of VW^iV^T. Let A be an $n \times n$ diagonal matrix such that $A_{ll} = \sum_i d_{il}$. This matrix contains the availability for each task. Then, we note that $VW^iV^T = VAV^T + V(W^i - A)V^T$. Conveniently, VAV^T can be precomputed once, and $(W^i - A)$ has only ω_i nonzero elements, where ω_i is the number of tasks that user i completed. ω_i is typically much smaller than n. We can thus perform computation of each row u_i in time $O(k^2\omega_i + k^3)$ and computation of the factor U in time $O(k^2\omega + k^3 m)$.

To solve for V when U is fixed, redefine W^j to be the diagonal $n \times n$ matrix such that $W^j_{ll} = w_{lj}$. This matrix contains all of the weights for task j on the diagonal. Note that these weights are *not* all the same. Only the weights for users who did not do any of task j are the same. Let p_j be the column in P corresponding to task j. Then,

$$v_j \quad = \quad (U^TW^jU + \lambda I)^{-1} U^TW^j p_j$$

We now see that $U^TW^jU = U^T(A_{jj}I)U + U^T(W^j - A_{jj}I)U$, which is equal to $(A_{jj}I)U^TU + U^T(W^j - A_{jj}I)U$. While $(A_{jj}I)U^TU$ must be computed for each task j, U^TU can be precomputed once, and multiplying every entry by A_{jj} only takes time $O(k^2)$. Abusing notation, let ω_j be the number of users who have worked on task j. Because $W^j - A_{jj}I$ only has ω_j nonzero entries, we can perform computation of each column v_j in time $O(k^2\omega_j + k^3)$ and computation of the factor V in time $O(k^2\omega + k^3 n)$.

Then, with fixed k, we achieve the stated overall running time for one iteration of coordinate descent for our model. We note this running time is linear in the number of nonzero entries in P, so this computation is highly scalable as more users do more tasks. □

The conditions of the theorem reflect the characteristic of negative implicit feedback seen in crowdsourcing. In our model, the confidence on negative signals (i.e., task availability) is independent of individual users, which satisfies condition 1. Note that the achieved running time means that including negative implicit feedback makes the model no harder to learn than a model with only positive implicit feedback.

Sampling-Based Learning

We now provide a sampling-based solution method for IFMF2, which we denote IFMF2S. This method has similarities to the approaches proposed in (Pan et al. 2008) and (DeCoste 2006); it aggregates the predictions of multiple independently learned models to improve the stability and accuracy of predictions under noisy data. In addition, this approach addresses a weakness in many matrix factorization approaches in that no uncertainty is associated with the predictions. In approaches that do output distributions (e.g., (Koren and Sill 2011; Lawrence and Urtasun 2009; Salakhutdinov and Mnih 2007)), the distributions are picked a priori as a modeling decision instead of emerging intrinsically from the data. IFMF2S's predictions are naturally in the form of a distribution, where the prediction of each learned model becomes a data point in this distribution.

We would like distributions on the entries of our predicted matrix as a point prediction frequently does not contain enough information. For example, suppose IFMF2 predicts that user i "rates" task j 0.5, which is right in the middle of the possible "ratings." At least two explanations exist for such a rating. One, user i neither likes nor dislikes the task and feels apathetic, or two, she either really enjoys or really hates the task, but our model does not have enough information to discriminate among these preferences.

In IFMF2S, we first constrain the normalization functions f and g to be between 0 and 1. Now, instead of generating weights for each entry in P, they denote the probability that that entry will be selected in the following sampling procedure. We sample Γ matrices $P^1, \ldots, P^\Gamma$ from P according to the probabilities of the entries. More precisely, if $p_{ij} = 1$, then for each matrix P^γ, we sample p^γ_{ij} as

$$p^\gamma_{ij} = \begin{cases} 1 & \text{with probability } f(c_{ij}) \\ \text{Missing} & \text{with probability } 1 - f(c_{ij}) \end{cases}$$

And if $p_{ij} = 0$, then

$$p^\gamma_{ij} = \begin{cases} 0 & \text{with probability } g(d_{ij}) \\ \text{Missing} & \text{with probability } 1 - g(d_{ij}) \end{cases}$$

The resulting P^γ matrix is sparse with equal weights on all entries and can be learned efficiently with traditional approaches. We can also apply this sampling-based learning method to IFMF by sampling both ones and zeroes with just f. We call this approach IFMFS.

An advantage of the sampling approach is that is it highly parallelizable. Factors for each of the sampled matrices $P^1, \ldots, P^\Gamma$ are learned independently to get a series of predictions $\hat{R}^1, \ldots, \hat{R}^\Gamma$. The point prediction for an entry can be obtained by taking the mean of $\{\hat{r}^1_{ij}, \ldots, \hat{r}^\Gamma_{ij}\}$. In addition to the point prediction, the collection of predictions from each sample forms a distribution of predictions for each entry in $\hat{R}$.

Experiments

Our experiments aim at answering the following questions regarding the generation of task recommendations in crowdsourcing: 1) How do the performances of matrix factorization approaches compare to simpler baselines? 2) By introducing implicit negative feedback, do our models IFMF2 or IFMF2S produce better recommendations than IFMF and IFMFS? and 3) Is our matrix factorization approach able to capture human-understandable trends in crowdsourcing marketplaces in addition to providing improvements in harder-to-understand quantitative metrics?

Data

We use data collected from Microsoft's internal Universal Human Relevance System (UHRS). Similar to other crowdsourcing marketplaces like Mechanical Turk, UHRS connects a large number workers from around the globe with human intelligence tasks. The training data was collected between February 1st 2013 and May 1st 2013. The testing data was collected between May 1st 2013 and May 14th 2013. The dataset overall includes over 17,000 users and over 2,100 tasks. Summary statistics for $c_{ij} > 0$ are: min: 1, max: 66630, mean: 919.67, sd: 3090.279. We logged the number of task instances done for each task and user pair in training and testing periods. The availability of tasks in the marketplace was not available to us. We use the number of instances completed by all users for a task as a proxy for the availability of that task. No further information about users or tasks (e.g., user demographics, task characteristics) is provided to the approaches for generating their recommendations.

Metrics and Methodology

We first learn $\hat{R} = UV$. Since explicit feedback is unavailable, our evaluation will be based on implicit signals: the amount of work done for each task by each user in the testing set. To generate task recommendations for a user, we multiply each predicted rating with the number of tasks available in the test set to produce expected throughputs, and produce a sorted list of tasks accordingly.

The first metric we use in our evaluations is the Mean Percentile Rank (MPR) (Hu, Koren, and Volinsky 2008): $MPR = \frac{\sum_{ij} c_{ij}\rho_{ij}}{c_{ij}}$ where ρ_{ij} is the percentile ranking of task j within a ranked list of recommended tasks for user i, and c_{ij} still refers to the number of tasks completed by user i for task j, but now in the test set. A lower MPR is better, and a random recommendation results in an expected MPR of 50%. The MPR is minimized by a recommendation that exactly matches the sorted list of tasks for each user with respect to their throughput in the testing set. The MPR metric is not perfect. It tends to give very high weight to tasks that users work on a great deal and heavily penalizes mistakes in predicting the exact ordering of tasks with high throughput.

We are less interested in whether or not we can perfectly predict the correct order of recommendations, and focused more on whether or not our recommendations can generally predict the tasks that the user will do. To this end, we also use a second metric that takes the form of a precision-recall (PR) curve. To produce a PR curve, we pick a set of percentile thresholds. For each threshold $t\%$, we take the top $t\%$ predictions and compute a precision and recall on retrieving the top $t\%$ of tasks that each user has completed. This metric allows us to evaluate the general correctness of our predictions rather than how effective our models are at exactly sorting the tasks users complete.

In all experiments, we compute metrics using only the tasks that a user did not complete in the training set, since predicting new tasks the user will do is much harder and interesting than predicting tasks that we already know users like. For all matrix factorizations, we perform 10 iterations of coordinate descent. All results we present use the best regularization settings we find for each model. We set $k = 20$. For normalization functions f and g, we find that setting $\alpha_1 = -1.4, \beta_1 = -5, \alpha_2 = -1.0, \beta_2 = -16.0$ work the best for their respective models. We use a 2.66 Ghz Intel(R) Xeon(R) processor.

Baselines

We use three models as baselines. The first, which we call *Global*, simply sorts the tasks based on their popularity. Each user receives the same recommendation list, and the top recommendation is the task that has been completed the most number of times.

We also try a neighborhood approach (Hu, Koren, and Volinsky 2008). First, for all pairs of tasks j_1, j_2, we compute their similarity $s_{j_1 j_2} = \frac{r_{j_1}^T r_{j_2}}{||r_{j_1}||||r_{j_2}||}$ where r_j is column j of R. Then we compute our prediction for user i and task j as $\hat{r}_{ij} = \sum_l s_{jl} r_{il}$. Intuitively, two tasks are similar if users show the same patterns of use for those two tasks. Training this approach takes ≈1734 seconds, which we will see

	10000	20000	All
Global	45.693	27.711	39.055
Neighborhood	16.402	12.915	4.605
IFMF	11.482	8.790	3.351
IFMF2	10.528	8.803	3.365
IFMFS	9.928	6.267	2.189
IFMF2S	7.736	6.180	1.902
LB	0.0325	0.0216	0.0256

Table 1: Average testing MPR of various models on different training sizes.

is about the same amount of time required for training the matrix factorization approaches.

Finally, we use a model that we call *LowerBound* (LB), which is not a baseline, but represents the best any model can perform on the MPR metric. We simply assume that we exactly know the data in the test set and construct a recommendation to minimize the MPR.

Baselines vs. Matrix Factorization

Table 1 compares the performances of baselines and the matrix factorization models when varying sizes of training data are available: a small 10,000 entry subset, a medium-sized 20,000 entry subset, and the full training dataset, containing over 70,000 entries. To account for randomness in initializations, we run each model 200 times and average the MPRs. A single training of IFMF on the largest set takes $\approx$1739 seconds. The testing set always remains the same. The figure shows that there is a large benefit in personalizing the recommendations; the global baseline has significantly higher MPR values than all other recommendation approaches. The results also show that matrix factorization approaches perform consistently better than the neighborhood-based approach, agreeing with results obtained in other domains such as movie recommendations. Another trend we see is that the performance of all personalized recommendation models improves with increases in the size of the training sets.

IFMF vs. IFMF2

Next, we evaluate how reasoning about negative feedback in making recommendations affects performance. A single training of IFMF2 takes $\approx$ 1741 seconds. As displayed in Table 1, IFMF2 achieves the best MPR using the smallest sized training sets, but performs slightly worse than IFMF when trained on the other sized datasets.

This result is not surprising, because IFMF is very good at learning the tasks that users like and perform large quantities of. As we increase the amount of training data, IFMF becomes very good at "beating" the MPR metric, which heavily weights tasks that users select often. The benefit of the IFMF2 model is learning about tasks that users dislike and the MPR metric does not highlight this characteristic. Therefore, we look at the PR curves in Figures 1a, 1b, and 1c. We see that despite achieving a slightly worse MPR on the larger training sets, IFMF2 achieves large precision and recall gains for all training sets. This result means that while

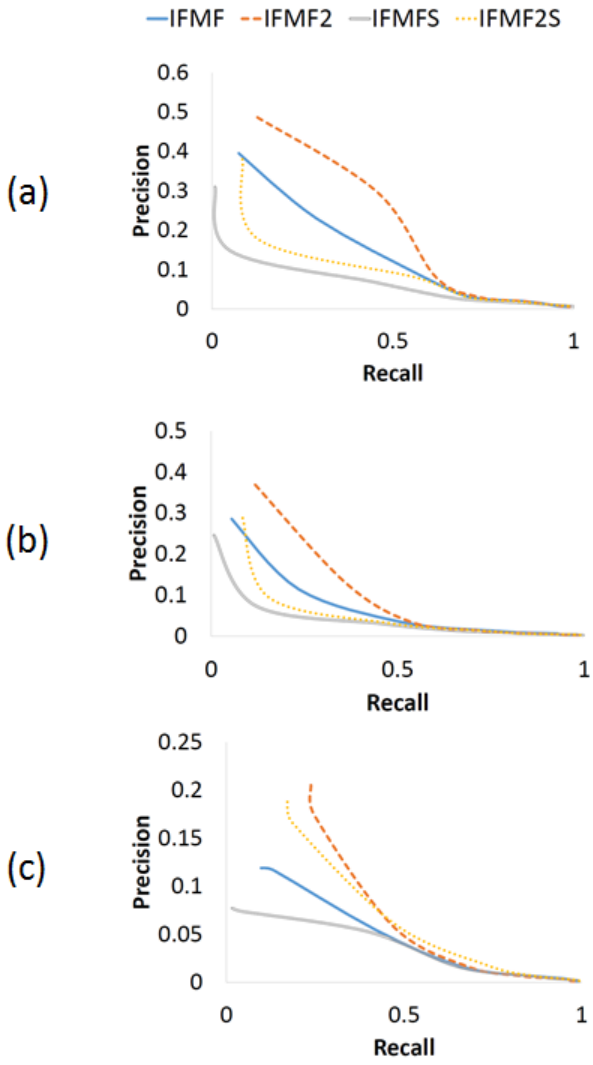

Figure 1: PR curves for IFMF, IFMF2, IFMFS and IFMF2S models trained on the small (a), medium (b), and large (c) training sets.

our model may not be able to exactly rank the tasks that users do by how many they have done, it is able to recall the tasks that users like with higher precision at all levels of user interest. In other words, while IFMF2 may not be able to order the top 20 tasks for each user exactly, it is much better at predicting the top 20 tasks, top 30 tasks, and so on.

Sampling-Based Learning

We also compare IFMF and IFMF2 when they are learned using the sampling-based approach (IFMFS and IFMF2S). We set the number of sampled matrices to be $\Gamma = 200$. The settings of f and g result in sampled matrices that are very sparse; each sampled matrix is 99.5% empty. We see in Table 1 and Figure 1 that adding negative implicit feedback to the model lowers MPR and increases precision and recall.

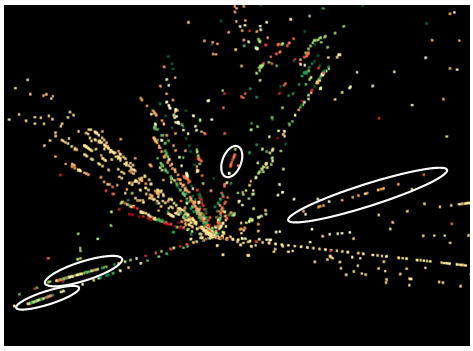

Figure 2: Starting from the bottom-left and moving clockwise, the circled clusters represent users who select tasks relating to videos, who select tasks relating to images, Australian users, and Irish users.

Table 1 shows that the models learned using sampling achieve better MPRs for all training set sizes. However, we also see that at least for the two smaller training sets, sampling causes the models to produce worse PR curves than their non-sampling counterparts. These results are not surprising given the characteristics of the sampling approaches. We can see the bagging effect on MPR values; by averaging over many different samples, we improve the models' abilities to learn exactly what the user will do and exactly what the user will not do by minimizing the effect of low-confidence entries as well as outliers. On the other hand, each sample includes only a subset of the training data, mostly missing entries with lower confidences. Thus, the models are less able to learn about the tasks that do not elicit extreme preferences. This explanation is further substantiated by observing the precipitous drops in precision as we increase recall. The models are very good at learning what the user likes, but are not so good at getting the middle ranges correct. On the full training set however, we see that IFMF2S almost catches up to IFMF2 in precision and recall, suggesting that the sampled data is now good enough to learn more sophisticated models, and that IFMF2S can be used as an alternative to IFMF2.

Qualitative Evidence and Explainability

We investigate whether the factors learned by our models are meaningful and can be explained qualitatively. Intuitively, the learned user factor U can be thought of as a vector of users, where each user is a vector of features. Although we cannot easily determine what these features mean, we can still cluster the users using their feature vectors. Therefore, we cluster the users using the U learned by the IFMF2 approach by k-means into 100 clusters. We next create a visualization of the users by performing a principal component analysis on the users' feature vectors to reduce the vectors to 2 dimensions, and plot each user in a 2D plane using SandDance[2] (Figure 2). So that the picture is a little more interesting, we color users corresponding to where they reside. Notably, the "spokes" in this galaxy of users correspond to clusters.

Inspection of these clusters reveals meaningful relationships. Some clusters correspond to the type of task that users select. For instance, one cluster is composed of users who select tasks with videos and another is composed of users who select tasks with news articles. There are clusters of users from the same countries, including clusters of users from Brazil, India, Great Britain, Canada, France, Australia, and many more. This visualization provides evidence that the matrix factorization approach learns about important trends in crowdsourcing marketplaces, such as users choosing tasks based on the countries they reside in, the languages they speak, and the interests and abilities that they have.

[2]http://research.microsoft.com/en-us/projects/sanddance

Related Work

Cosley et al. (2006) show that using smart but simple algorithms to select tasks for users can improve throughput. SuggestBot (Cosley et al. 2007) suggests Wikipedia articles for users to edit by analyzing article graphs and similarities. Several works address the problem of dynamic task routing. Pick-a-Crowd (Difallah, Demartini, and Cudre-Mauroux 2013) is a content-based approach to recommending tasks to users that focuses on dynamically pushing tasks to users to maximize accuracy. Ho et al. (2013; 2012) formalize the task routing problem for labeling tasks, and provide an algorithm that is provably near-optimal in minimizing a budget. Tran-Tranh et al. (2012) model task routing using bounded multi-armed bandits with the goal of maximizing the overall utility achieved. These models all assume that users complete the tasks that they are assigned, whereas we focus on producing a list of recommended tasks from which users can freely pick, to improve discovery and throughput.

Steck (2010) points out that for conventional explicit feedback domains, missing data is usually not missing at random. Therefore, he inserts all missing data into the matrix with some low rating, and shows that a weighted matrix factorization where the inserted data has very small weight can outperform conventional unweighted matrix factorization methods on matrices with missing data.

Several studies (Gantner et al. 2012; Pan et al. 2008) consider how to weight negative examples, but forego weighting positive examples in a meaningful manner. Additionally, their models are not built for implicit feedback. In contrast, our work presents a general model for introducing negative implicit feedback into existing implicit feedback models, without sacrificing existing elements.

Some efforts (Paquet and Koenigstein 2013; Rendle et al. 2009) consider more complex models or different optimization schemes for recommendation. Our goal is not to compare and contrast various existing implicit or explicit positive feedback models, but to show that incorporating negative implicit feedback can generate better recommendations.

Conclusion and Future Work

We introduced a methodology for harnessing implicit feedback in a recommendation system that integrates negative feedback. We presented two algorithms for learning our model in a computationally feasible way. We demonstrated

and evaluated the work on the application domain of crowdsourcing and showed that the quality of task recommendations is improved with our models. Finally, we examined qualitatively the user factor that is produced via learning our models, and showed that we can cluster users into meaningful groups that represent their backgrounds and interests.

We presented the best results that we had obtained for each model via a limited search in the parameter space. We believe that more efficient and principled methods for parameter selection would be beneficial. We considered offline methods for learning. Faster online methods, like those proposed in (Rendle and Schmidt-Thieme 2008), can be beneficial for practical applications. In addition, the distributions generated by the sampling-based method can be used by decision-theoretic systems. For example, such a system might recommend tasks with high entropy in order to learn more about users so as to maximize the utility gained from future recommendations.

Our work can also be extended by developing more sophisticated implicit feedback models for crowdsourcing and investigating the generalizability of our models to other domains where negative implicit feedback can be helpful in making recommendations. We foresee models for the crowdsourcing domain that bring many factors together, including task reward and task difficulty, in order to estimate the probability that a user is aware of a task and is purposely not selecting it. Further, the nature of UHRS is such that many tasks are available for a long period of time, allowing us to learn about them and recommend them before they are removed. For platforms with more ephemeral tasks, one can use clustering techniques to group tasks into task types and then apply our models. Finally, while our experiments showed that our approaches can make good recommendations, the ultimate evaluation requires deploying these recommendations in a live crowdsourcing system to understand the effects on throughout, accuracy, and engagement.

Acknowledgments

We thank Rajesh Patel, Steven Shelford, and Hai Wu for providing access to data and for discussions and feedback. We thank Steven Drucker for assistance with using the SandDance visualization tool.

References

Bell, R. M., and Koren, Y. 2007. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In *ICDM*.

Cosley, D.; Frankowski, D.; Terveen, L.; and Riedl, J. 2006. Using intelligent task routing and contribution review to help communities build artifacts of lasting value. In *CHI*.

Cosley, D.; Frankowski, D.; Terveen, L.; and Riedl, J. 2007. Suggestbot: Using intelligent task routing to help people find work in wikipedia. In *IUI*.

DeCoste, D. 2006. Collaborative prediction using ensembles of maximum margin matrix factorizations. In *ICML*.

Difallah, D. E.; Demartini, G.; and Cudre-Mauroux, P. 2013. Pick-a-crowd: Tell me what you like, and i'll tell you what to do. In *WWW*.

Funk, S. 2006. Netflix update: Try this at home. http://sifter.org/ simon/journal/20061211.html.

Gantner, Z.; Drumond, L.; Freudenthaler, C.; and Schmidt-Thieme, L. 2012. Personalized ranking for non-uniformly sampled items. *Journal of Machine Learning Research* 18:231–247.

Goldberg, D.; Nichols, D.; Oki, B. M.; and Terry, D. 1992. Using collaborative filtering to weave an information tapestry. *Communications of the ACM* 35(12):61–70.

Herlocker, J. L.; Konstan, J. A.; Borchers, A.; and Riedl, J. 1999. An algorithmic framework for performing collaborative filtering. In *SIGIR*.

Ho, C.-J., and Vaughan, J. W. 2012. Online task assignment in crowdsourcing markets. In *AAAI*.

Ho, C.-J.; Jabbari, S.; and Vaughan, J. W. 2013. Adaptive task assignment for crowdsourced classification. In *ICML*.

Hu, Y.; Koren, Y.; and Volinsky, C. 2008. Collaborative filtering for implicit feedback datasets. In *ICDM*.

Koren, Y., and Sill, J. 2011. Ordrec: An ordinal model for predicting personalized item rating distributions. In *Recsys*.

Koren, Y.; Bell, R.; and Volinsky, C. 2009. Matrix factorization techniques for recommender systems. *Computer* 42(8):30–37.

Lawrence, N. D., and Urtasun, R. 2009. Non-linear matrix factorization with gaussian processes. In *ICML*.

Linden, G.; Smith, B.; and York, J. 2003. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing* 7(1):76–80.

Pan, R.; Zhou, Y.; Cao, B.; Liu, N. N.; and Lukose, R. 2008. One-class collaborative filtering. In *ICDM*.

Paquet, U., and Koenigstein, N. 2013. One-class collaborative filtering with random graphs. In *WWW*.

Rendle, S., and Schmidt-Thieme, L. 2008. Online-updating regularized kernel matrix factorization models for large-scale recommender systems. In *RecSys*.

Rendle, S.; Freudenthaler, C.; Gantner, Z.; and Schmidt-Thieme, L. 2009. Bpr: Bayesian personalized ranking from implicit feedback. In *UAI*.

Salakhutdinov, R., and Mnih, A. 2007. Probabilistic matrix factorization. In *NIPS*.

Sarwar, B.; Karypis, G.; Konstan, J.; and Riedl, J. 2001. Item-based collaborative filtering recommendation algorithms. In *WWW*.

Srebro, N., and Jaakkola, T. 2003. Weighted low-rank approximations. In *ICML*.

Steck, H. 2010. Training and testing of recommender systems on data missing not at random. In *KDD*.

Tran-Tranh, L.; Stein, S.; Rogers, A.; and Jennings, N. R. 2012. Efficient crowdsourcing of unknown experts using multi-armed bandits. In *ECAI*.

Zhou, Y.; Wilkinson, D.; Schreiber, R.; and Pan, R. 2008. Large-scale parallel collaborative filtering for the netflix prize. In *Algorithmic Aspects in Information and Management*.