Learning Binary Codes for Collaborative Filtering

Ke Zhou
College of Computing
Georgia Institute of Technology
Atlanta, GA 30032
kzhou@gatech.edu

Hongyuan Zha
College of Computing
Georgia Institute of Technology
Atlanta, GA 30032
zha@cc.gatech.edu

ABSTRACT

This paper tackles the efficiency problem of making recommendations in the context of large user and item spaces. In particular, we address the problem of learning binary codes for collaborative filtering, which enables us to efficiently make recommendations with time complexity that is independent of the total number of items. We propose to construct binary codes for users and items such that the preference of users over items can be accurately preserved by the Hamming distance between their respective binary codes. By using two loss functions measuring the degree of divergence between the training and predicted ratings, we formulate the problem of learning binary codes as a discrete optimization problem. Although this optimization problem is intractable in general, we develop effective relaxations that can be efficiently solved by existing methods. Moreover, we investigate two methods to obtain the binary codes from the relaxed solutions. Evaluations are conducted on three public-domain data sets and the results suggest that our proposed method outperforms several baseline alternatives.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information filtering; I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms, Performance, Experimentation

Keywords

Recommender systems, Collaborative filtering, Learning binary codes, Discrete optimization, Relaxed solutions

1. INTRODUCTION

With the rapid growth of E-commerce, hundreds of thousands of products, ranging from books, mp3s to automobiles, are sold through online marketplaces nowadays. In addition, millions of customers with diverse backgrounds and preferences make purchases online, generating great opportunities as well as challenges for E-commerce companies — How to match products to their potential buyers not only accurately but also efficiently. Since collaborative filtering is an essential component for many existing recommendation systems, it has been actively investigated by a wide range of previous studies to improve its accuracy [1, 19]. On the other hand, due to the nature of their applications, collaborative filtering systems are usually required to learn and predict the preferences between a large number of users and items. Therefore, for a given user, it is important to retrieve products that satisfy her preferences efficiently, leading to fast response time and better user experience. Naturally, the problem can be viewed as a similarity search problem where we seek "similar" items for a given user. Recent studies show that binary coding is a promising approach for fast similarity search [9, 13, 14, 17, 21]. The basic idea is to represent data points by binary codes that preserve the original similarities between them. One significant advantage of this approach is that the retrieval of similar data points can be conducted by searching for data points within a small Hamming distance, which can be performed in time that is independent of the total number of data [17]. However, no prior studies have been focused on constructing binary codes for both users and items in the context of collaborative filtering — to the best of our knowledge — a gap we propose to fill in this paper.

One key obstacle that hinders direct exploitation of the existing approaches to learning binary codes to the collaborative filtering context is that most of them assume the similarities between any pairs of data points are given explicitly, e.g., in the form of kernel functions or similarity graphs [13, 21, 24]. However, in collaborative filtering, the similarities between users and items are not known explicitly. In fact, the main goal of collaborative filtering algorithms is to estimate and predict unobserved similarities between users and items from the training data in order to make recommendations. In this paper, we address the problem of learning binary codes for collaborative filtering. Specifically, we propose to learn compact yet effective binary codes for both users and items from the training rating data. Unlike previous works on learning binary codes, we do not assume the similarity between users and items are known explicitly. Therefore, the binary codes we construct not only accurately preserve the observed preferences of users, but they also can be used to *predict* the unobserved preferences, making the proposed method conceptually unique compared with the existing methods.

Our approach is based on the idea that the binary codes

KDD'12, August 12–16, 2012, Beijing, China.

assigned to users and items should preserve the preferences of users over items. Two loss functions are applied to measure the divergence between the training data and the estimates based on the binary codes. Unfortunately, thus formulated, the resulting discrete optimization problem is difficult to solve in general. Through relaxing the binary constraints, it turns out the relaxed optimization problem can be solved effectively by existing solvers. Moreover, we propose two effective methods for rounding the relaxed solutions to obtain binary codes. One key property of the binary codes obtained by the proposed method is that the degree of preferences of a user to items can be measured by the number of common bits between their corresponding binary codes. Hence, the major advantage of representing users and items by binary codes is to enable fast search: In order to provide recommendations for a user, we need only to search items with binary codes within a small Hamming distance to the binary codes of the given user. We evaluated the proposed method over three data sets and compare it with several baseline alternatives. The results show that the binary codes obtained by the proposed method can preserve and predict the preferences of users more accurately than the baselines.

The contributions of this paper are essentially threefold: 1) We propose to learn binary codes for collaborative filtering that accurately preserve the preference of users, which generalizes the existing works of learning binary codes to the context of collaborative filtering. 2) We propose a framework to learning binary codes based on training rating data, which leads to relaxed optimization problems that can be solved effectively by the state-of-the-art optimization techniques. 3) Our experimental evaluations show that the binary codes obtained by the proposed method can preserve the preference of users better than several baseline alternatives.

The rest of the paper is organized as follows: In Section 2, we briefly review existing studies for collaborative filtering and learning binary codes. In Section 3, we first formulate the problem of learning binary codes for collaborative filtering as a discrete optimization problem and introduce the two loss functions used in this work. Then, the learning algorithm proposed with detailed derivations based on transforming and relaxing the discrete optimization problem so that it can be optimized efficiently. Moreover, we discuss two different methods for rounding real-valued solutions to obtain binary codes. The evaluations are described and analyzed in Section 4. We conclude our work and present several future research directions in Section 5.

2. RELATED WORK

2.1 Collaborative Filtering

Many studies on recommender systems have been focused on collaborative filtering approaches. These methods can be categorized into memory-based and model-based. The reader is referred to the survey papers [1, 19] for a comprehensive summary of collaborative filtering algorithms.

Recently, matrix factorization has become a popular direction for collaborative filtering [2, 11, 15, 16]. These methods are shown to be effective in many applications. Specifically, matrix factorization methods seek to associate both users and items with latent profiles represented by vectors in a low dimensional Euclidean space that can capture their characteristics. Specifically, the preference of a user on an item is usually measured by some similarity, such as the dot-product, between their low dimensional profiles. These studies are related to our work in the sense that both of them aim to find certain representations that preserve the preference between users and items. However, one key difference is that our work aims to find binary codes in Hamming space for representing users and items, which has the nice property that the retrieval of interesting items for a user can be performed in time that is independent of the total number of items [17].

Another direction of collaborative filtering investigate the problem of using binary codes to create fingerprints for users [3–5]. The idea is create binary fingerprints for each user using randomized algorithms so that the similarity between users can be approximated according to the fingerprints. However, these studies do not address the problem of simultaneously representing users and items by binary fingerprints. Thus, the preference of users on items can not be estimated directly from these fingerprints. On the other hand, the binary codes learnt by our proposed method can be directly used to measure the preference of users over items.

2.2 Learning Binary Codes

The problem of learning binary codes for fast similarity search has been investigated in several studies [12, 13, 17, 21] Locality sensitive hashing tries to construct binary codes that preserve certain distance (e.g., L_p distance) between different points with high probability, which is usually archived by random projection [6, 8]. In [18], the problem of learning effective binary codes is solved by utilizing the idea of boosting. The work of [17] proposes to learn binary codes making use of Restricted Boltzmann Machine (RBM) for fast similarity search of documents. Recent work focuses on constructing binary codes based on a given similarity function [14, 20, 21]. The basic idea is to apply spectral analysis techniques to the data and embedding data points into a low dimension space. For example, the work [21] investigate the requirements for compact and effective binary codes. Their solution relies on spectral graph partition, which can be solved by eigenvalue decomposition of the Laplacian matrix for the graph. It has been shown that these methods archive significant performance improvements in terms of preserving the similarity between data points. Although several extension of this method has been studied [7, 24], these methods only consider the problem of obtaining binary codes for one type of entities. However, in collaborative filtering, two types of entities, users and items, are naturally involved and thus should be consider simultaneously, which makes it difficult to applies these method directly.

The work [23] proposes to learn binary codes for both documents and terms by viewing the term-document matrix as a bipartite graph and apply the method proposed in [21] to obtain the binary codes. However, this method can not deal with the problem of unobserved/missing ratings in collaborative filtering. As shown in our experiments, the binary codes obtained by this method quickly overfit the training data and lead to poor prediction accuracy.

3. LEARNING BINARY CODES

In this section, we describe the proposed method for learning binary codes for collaborative filtering. We first describe the general formulation for this problem through optimiza-

tion using squared and pairwise loss functions, respectively. Then, the learning method based on solving the relaxed problem is derived in detail. Finally, we discuss two methods to obtain binary codes from the real-valued solutions of the relaxed problems.

3.1 Problem Formulation

The goal of collaborative filtering is to recommend interesting items to users according to their past ratings on the items. Formally, we assume that r_{ui} represents the rating of user $u \in \mathcal{U}$ for item $i \in \mathcal{I}$, where $\mathcal{U}$ and $\mathcal{I}$ are the user and item space, respectively. Without loss of generality, we further assume that r_{ui} is a real number in the interval $[0, 1]$. Moreover, we assign binary codes $f_u \in \{-1, 1\}^B$ for each user u and $h_i \in \{-1, 1\}^B$ for each item i, where B is the length of the binary codes. Our goal is to construct binary codes for users and items that preserve the preferences between them — the degree of preference of user u over item i can be estimated by the similarity between their binary codes f_u and h_i. A natural way to define the similarity between user u and item i is the fraction of common bits in their binary codes f_u and h_i, leading to the similarity function,

$$\mathrm{sim}(f_u, h_i) = \frac{1}{B}\sum_{k=1}^{B} \mathbb{I}(f_u^{(k)} = h_i^{(k)}),$$

where $f_u^{(k)}$ and $h_i^{(k)}$ represent the k-th bit of the binary codes f_u and h_i, respectively. $\mathbb{I}(\cdot)$ denotes the indicator function that returns 1 if the statement in its parameter is true and zero otherwise.

It is easy to check that the following holds for the similarity function $\mathrm{sim}(\cdot, \cdot)$ defined above:

$$\mathrm{sim}(f_u, h_i) = 1 - \frac{1}{B}\mathrm{dist}_{\mathrm{H}}(f_u, h_i),$$

where $\mathrm{dist}_{\mathrm{H}}(f_u, h_i)$ is the Hamming distance between two binary codes f_u and h_i. The above fact suggests that the smaller the Hamming distance is, the more similar their binary codes become. Therefore, in order to find items with similar binary codes to a user represented by f_u, it is sufficient to search items i within a small Hamming distance $\mathrm{dist}_{\mathrm{H}}(f_u, h_i)$. This allows us to find similar items in time that is independent to the total number of items [17].

In order to make accurate recommendations to users, we need to find binary codes f_u and h_i for users and items such that the preferences between them are preserved by the similarities between their respective binary codes. In addition, in collaborative filtering, we only observe a subset of all the possible ratings $\{r_{ui}|(u, i) \in \mathcal{O}\}$ where $\mathcal{O} \subset \mathcal{U} \times \mathcal{I}$ and we need to recommend items to users according to their preferences over items whose ratings are unobserved. Therefore, the key for accurate recommendations is to construct binary codes that can not only preserve the observed ratings but also accurately predict the preferences of users on unobserved items.

Our approach to learn binary codes is to estimate them from observed ratings. Specifically, we propose to construct binary codes that minimize the degree of divergence between the observed ratings and the ratings estimated from the binary codes. To this end, we apply two objective functions to measure the degree of divergence between the observed ratings and the model estimates:

- **Squared Loss.** Using this loss function, we seek to minimize the squared error of the observed ratings and the similarity estimations from the binary codes, which is a commonly used loss function for collaborative filtering:

$$\min_{f_u, h_i \in \{\pm 1\}^B} \mathcal{L}_{sq} = \sum_{(u,i)\in\mathcal{O}} (r_{ui} - \mathrm{sim}(f_u, h_i))^2. \quad (1)$$

- **Pairwise Loss.** Since we are more interested in preserving the relative orders between items rather than their absolute values, it is natural to consider the pairwise loss function described as follows:

$$\min_{f_u, h_i \in \{\pm 1\}^B} \mathcal{L}_{pair} = \sum_{u} \sum_{i,j\in\mathcal{O}_u} \Big((r_{ui} - r_{uj}) - (\mathrm{sim}(f_u, h_i) - \mathrm{sim}(f_u, h_j))\Big)^2, \quad (2)$$

where $\mathcal{O}_u = \{(u, i) \mid (u, i) \in \mathcal{O}\}$. Minimizing the above loss function requires the binary codes to preserve the relative difference between each pair of items rated by the user.

Additionally, we also require the binary codes to be *balanced* — we would like each bit of the binary codes to have equal chance to be 1 or -1. The balance constraint is equivalent to maximizing the entropy of each bit of the binary codes, which indicates that each bit carries as much information as possible. Specifically, we will enforce the following constraints to the binary codes:

$$\sum_u f_u = 0, \quad \text{and} \quad \sum_i h_i = 0.$$

The above constraints motivate the following regularized objective function for learning binary codes. For example, for squared loss, we have the following objective function:

$$\min_{f_u, h_i \in \{\pm 1\}^B} \sum_{(u,i)\in\mathcal{O}} (r_{ui} - \mathrm{sim}(f_u, h_i))^2 + \lambda(\|\sum_u f_u\|^2 + \|\sum_i h_i\|^2), \quad (3)$$

where the first term is the loss over observed ratings and the second term represents that we prefer balanced binary codes. The parameter λ controls the trade-off between minimizing the empirical errors and the enforcement of the constraints. $\|\cdot\|$ indicates Euclidean norm of a vector.

Similarly, we have the following regularized objective function for the pairwise loss function:

$$\min_{f_u, h_i \in \{\pm 1\}^B} \sum_{u} \sum_{i,j\in\mathcal{O}_u} \Big((r_{ui} - r_{uj}) - (\mathrm{sim}(f_u, h_i) - \mathrm{sim}(f_u, h_j))\Big)^2 + \lambda(\|\sum_u f_u\|^2 + \|\sum_i h_i\|^2). \quad (4)$$

3.2 Learning

The objective functions defined in Equation (3) and Equation (4) are defined over the discrete space $\{\pm 1\}^B$, which makes them difficult to optimize in general. Therefore, we propose to solve it approximately by transforming the objective functions and then relaxing the space of solutions to be $[-1, 1]^B$. For the sake of concreteness, we describe our

method for solving the squared loss in Equation (3) in detail. The pairwise loss in Equation (4) can be optimized in a similar approach.

First, we notice that for binary codes $f, h \in \{\pm 1\}^B$, the following property holds:

$$\begin{aligned}\text{sim}(f,h) &= \frac{1}{B}\sum_{k=1}^{B}\mathbb{I}(f^{(k)} = h^{(k)}) \\ &= \frac{1}{2B}\left(\sum_{k=1}^{B}\mathbb{I}(f^{(k)} = h^{(k)}) + (B - \sum_{k=1}^{B}\mathbb{I}(f^{(k)} \neq h^{(k)}))\right) \\ &= \frac{1}{2B}\left(B + \sum_{k=1}^{B} f^{(k)}h^{(k)}\right) \\ &= \frac{1}{2} + \frac{1}{2B}f^T h.\end{aligned}$$

Thus, by substituting the above equation into the regularized objective function defined in Equation (3), we can express the objective function for squared loss as follows:

$$\begin{aligned}\min_{f_u, h_i \in \{\pm 1\}^B} \mathcal{L}_{reg} = &\sum_{(u,i)\in\mathcal{O}} (r_{ui} - \frac{1}{2} - \frac{1}{2B}f_u^T h_i))^2 \\ &+ \lambda(\|\sum_u f_u\|^2 + \|\sum_i h_i\|^2). \end{aligned} \quad (5)$$

A widely used approach to obtain approximate solutions to the above discrete optimization problem is to relax the space of solution to be real values and thus enables the application of the continuous optimization techniques to solve the problem. To this end, we first relax the space of solution to be real vectors in $[-1,1]^B$ and then we will round the real-valued solutions into $\{\pm 1\}^B$. The details of rounding are discussed in Section 3.3.

It is also interesting to note that the above formulation also reveals a nice connection between learning binary codes and the matrix factorization approaches that are widely applied in collaborative filtering. In particular, the first term in (5) is the objective function that factorizes the linearly transformed matrix of observed ratings to find low-dimensional representations for users and items. The second term is different from the usual ℓ_2 regularization used in traditional matrix factorization since we would like the binary codes to balanced rather than close to zero in this case.

Given the relaxed problem, the partial derivatives of the objective function $\mathcal{L}_{reg}$ with respect to f_u and h_i can be expressed as follows:

$$\begin{aligned}\frac{\partial \mathcal{L}_{reg}}{\partial f_u} &= -\frac{1}{B}\sum_{i\in\mathcal{O}_u}(r_{ui} - \frac{1}{2} - \frac{1}{2B}f_u^T h_i)h_i + 2\lambda\sum_{u'} f_{u'}, \\ \frac{\partial \mathcal{L}_{reg}}{\partial h_i} &= -\frac{1}{B}\sum_{u\in\mathcal{O}_i}(r_{ui} - \frac{1}{2} - \frac{1}{2B}f_u^T h_i)f_u + 2\lambda\sum_{i'} h_{i'}.\end{aligned}$$

The relaxed problem can be solved by methods such as LBFGS [25] and stochastic gradient descent.

3.3 Obtaining Binary Codes

After solving the relaxed optimization problem defined in Equation (5), we obtain real-valued vectors $\tilde{f}_u$ and $\tilde{h}_i \in [-1,1]^B$ for each user u and item i. In this section, we propose two methods to obtain binary codes f_u and $h_i \in \{\pm 1\}^B$ from these real-valued vectors.

3.3.1 Rounding to Closest Binary Codes

A straightforward method is to find binary vectors f_u and $h_i \in \{\pm 1\}^B$ that are closest to $\tilde{f}_u$ and $\tilde{h}_i$. Specifically, we seek to optimize the following objective function to obtain f_u for all $u \in \mathcal{U}$:

$$\min_{f_u \in \{\pm 1\}^B} \sum_u \|f_u - \tilde{f}_u\|^2 \quad (6)$$

subject to $\sum_u f_u = 0$. Similarly, we can obtain h_i for all item $i \in \mathcal{I}$ by:

$$\min_{h_i \in \{\pm 1\}^B} \sum_i \|h_i - \tilde{h}_i\|^2 \quad (7)$$

subject to $\sum_u h_i = 0$.

It turns out that the optimization problems defined in Equation (6) and Equation (7) have the following solution:

$$f_u^{(k)} = \begin{cases} 1, & \tilde{f}_u^{(k)} > \text{median}(\tilde{f}_u^{(k)} : u \in \mathcal{U}), \\ -1, & \text{Otherwise}, \end{cases}$$

and

$$h_i^{(k)} = \begin{cases} 1, & \tilde{h}_i^{(k)} > \text{median}(\tilde{h}_i^{(k)} : i \in \mathcal{I}), \\ -1, & \text{Otherwise}, \end{cases}$$

where median$(\cdot)$ represents the median of a set of real numbers.

3.3.2 Improved Rounding by Orthogonal Transformations

Another method to obtain the binary codes from the relaxed solution $\tilde{f}_u$ and $\tilde{h}_i$ makes use of the structure of the solutions to the relaxed optimization problem. Similar ideas have been investigated for spectral clustering [22] and we extend the idea to the context of learning binary codes. First, we observe that if $\tilde{f}_u$ and $\tilde{h}_i$ are optimal solutions for (5), then $Q\tilde{f}_u$ and $Q\tilde{h}_i$ are also optimal solutions achieving the same value of the objective function for an arbitrary orthogonal matrix $Q \in \mathbb{R}^{B\times B}$, i.e., $Q^T Q = I$. This observation can be proved as follows:

$$\begin{aligned}&\mathcal{L}_{reg}(Q\tilde{f}_u, Q\tilde{h}_i) \\ &= \sum_{(u,i)\in\mathcal{O}} (r_{ui} - \frac{1}{2} - \frac{1}{2B}(Q\tilde{f}_u)^T(Q\tilde{h}_i))^2 \\ &+ \lambda(\|\sum_u (Q\tilde{f}_u)\|^2 + \|\sum_i (Q\tilde{h}_i)\|^2) \\ &= \sum_{(u,i)\in\mathcal{O}} (r_{ui} - \frac{1}{2} - \frac{1}{2B}\tilde{f}_u^T\tilde{h}_i)^2 + \lambda(\|\sum_u \tilde{f}_u\|^2 + \|\sum_i \tilde{h}_i\|^2) \\ &= \mathcal{L}_{reg}(\tilde{f}_u, \tilde{h}_i),\end{aligned}$$

where the second equation utilizes the fact the Q is an orthogonal matrix. The above observation shows that applying orthogonal transformations to an optimal solution of relaxed optimization problem does not change the value of the objective function, which motives the following method to obtain binary codes from the relaxed solution:

$$\min_{Q, f_u, h_i \in \{\pm 1\}^B} \sum_u \|f_u - Q\tilde{f}_u\|^2 + \sum_i \|h_i - Q\tilde{h}_i\|^2 \quad (8)$$

subject to:

$$\sum_u f_u = 0, \quad \sum_i h_i = 0, \quad Q^T Q = I.$$

Intuitively, instead of directly finding binary codes that are close to the relaxed solutions, we seek binary codes that are close to some orthogonal transformation of the relaxed solutions. Introducing the orthogonal transformation Q not only preserves the optimality of the relaxed solutions but provides us more flexibility to obtain better binary codes.

The optimization problem defined in Equation (8) can be solved by minimizing with respect to f_u, h_i and Q alternatively.

Optimization with respect to f_u and h_i. Specifically, we first fix the orthogonal transformation Q and optimizing with respect to f_u and h_i:

$$\min_{f_u, h_i \in \{\pm 1\}^B} \sum_u \|f_u - Q\tilde{f}_u\|^2 + \sum_i \|h_i - Q\tilde{h}_i\|^2$$

subject to $\sum_u f_u = 0, \sum_i h_i = 0$. The solution can be expressed as follows:

$$f_u^{(k)} = \begin{cases} 1, & (Q\tilde{f}_u)^{(k)} > \text{median}((Q\tilde{f}_u)^{(k)} : u \in \mathcal{U}), \\ -1, & \text{Otherwise}, \end{cases}$$

and

$$h_i^{(k)} = \begin{cases} 1, & (Q\tilde{h}_i)^{(k)} > \text{median}((Q\tilde{h}_i)^{(k)} : i \in \mathcal{I}), \\ -1, & \text{Otherwise} \end{cases}$$

where $(Q\tilde{f}_u)^{(k)}$ represents the k-th element of the transformed vector $Q\tilde{f}_u$.

Optimization with Respect to Q. In this case, we fix f_u and h_i for all $u \in \mathcal{U}$ and $i \in \mathcal{I}$. Then we solve the following optimization problem to update the orthogonal transformation Q:

$$\begin{aligned} \min_{Q \in \mathbb{R}^{B \times B}} L(Q) &= \sum_u \|f_u - Q\tilde{f}_u\|^2 + \sum_i \|h_i - Q\tilde{h}_i\|^2 \\ &= \|F - Q\tilde{F}\|_F^2 + \|H - Q\tilde{H}\|_F^2 \end{aligned} \quad (9)$$

subject to the constraint $Q^TQ = I$, where $F = [f_1, \ldots, f_{|\mathcal{U}|}]$, $\tilde{F} = [\tilde{f}_1, \ldots, \tilde{f}_{|\mathcal{U}|}]$, $H = [h_1, \ldots, h_{|\mathcal{I}|}]$ and $\tilde{H} = [\tilde{h}_1, \ldots, \tilde{h}_{|\mathcal{I}|}]$. $\|\cdot\|_F$ indicates the Frobenius norm. The following theorem enables us to solve the optimization problem efficiently by singular value decomposition:

THEOREM 1. *Let UDV^T be the singular value decomposition of the matrix $(F\tilde{F}^T + H\tilde{H}^T)$. Then, $Q = UV^T$ minimizes the objective function defined in Equation (9).*

PROOF. First notice that the objective function $L(Q) = \|F\|^2 + \|\tilde{F}\|^2 - \text{trace}(F\tilde{F}^TQ^T) + \|H\|^2 + \|\tilde{H}\|^2 - \text{trace}(H\tilde{H}^TQ^T)$. Therefore, the optimization problem is equivalent to maximizing the following function subject to the orthogonality constraint $Q^TQ = I$:

$$\text{trace}(F\tilde{F}^TQ^T) + \text{trace}(H\tilde{H}^TQ^T) = \text{trace}((F\tilde{F}^T + H\tilde{H}^T)Q^T).$$

Let us consider the Lagrange

$$L(Q, \Lambda) = \text{trace}((F\tilde{F}^T + H\tilde{H}^T)Q^T) - \frac{1}{2}\text{trace}(\Lambda(Q^TQ - I)),$$

where Λ is a symmetric matrix. By taking the gradient with respect to Q, we have

$$(F\tilde{F}^T + H\tilde{H}^T) - \Lambda Q = 0,$$

Thus, $\Lambda = (F\tilde{F}^T + H\tilde{H}^T)Q^T = UDV^TQ^T$, which implies that $\Lambda^2 = UD^2U^T$. Hence, $\Lambda = UDU^T$. Substituting it into the above equation, we have $Q = UD^{-1}U^TUDV^T = UV^T$. □

In general, we perform the above two steps alternatively until the solution converges and obtain the binary codes f_u and h_i.

4. EXPERIMENTS

In this section, we describe the experiments conducted to evaluate the proposed method for learning binary codes. For the sake of simplicity, we denote the proposed method with squared loss and pairwise loss defined in Equation (3) and Equation (4) by CFCodeReg and CFCodePair, respectively.

4.1 Evaluation Metrics

We apply two evaluation metrics to evaluate the performance of CFCodeReg and CFCodePair. Our goal is to evaluate whether the obtained binary codes can accurately preserve the preferences of users to items. The evaluation metrics are described as follows:

- **Discounted Cumulative Gain (DCG).** DCG [10] is widely used to evaluate the quality of rankings. In order to compute DCG, we sort the items according to the Hamming distance between their binary codes to the binary codes for the user. The DCG value of a ranking list is calculated by the following equation:

$$\text{DCG@n} = \sum_{i=1}^{n} \frac{2^{r_i} - 1}{\log(i+1)},$$

  where r_i is the rating assigned by a user to the i-th item in the ranking list. DCG mainly focuses on evaluating whether the obtained binary codes can accurately preserve the relative orders of the of items rated by each user. We use DCG@5 as a evaluation metric in our experiments. When computing DCG, we only consider the observed ratings in the test set.

In order to evaluate the performance of using binary codes for recommending top-K items to users, we apply the following evaluation metrics:

- **Precision:** For each user u, we retrieve the set of items S_u with binary codes within Hamming distance 3 to the binary codes of the user. The precision is defined as the fraction of relevant items in S_u. Formally,

$$\text{Prec} = \frac{|\{i : i \in S_u \text{ and item } i \text{ is relevant to user } u\}|}{|S_u|}.$$

  In our experiments, all items with ratings that are greater than or equal to 5 are regarded as relevant items of users.

4.2 Data Sets

We used three data sets to evaluate the performance of CFCodeReg and CFCodePair, MovieLens, EachMovie and Netflix with their statistics summarized in Table 1.

- The MovieLens[1] data set contains 3900 movies, 6040 users and about 1 million ratings. In this data set, about 4% of the user-movie dyads are observed. The ratings are integers ranging from 1 (bad) to 5 (good).
- The EachMovie data set, collected by HP/Compaq Research, contains about 2.8 million ratings for 1628 movies by 72,916 users. The ratings are integers ranging from 1 to 6.

[1]http://www.grouplens.org/node/73

Table 1: Statistics for Data Sets

Dataset	No. Users	No. Items	No. Ratings
MovieLens	6040	3900	1,000,209
EachMovie	72,916	1628	2,811,983
Netflix	480,189	17,770	104,706,033

- The Netflix[2] is one of the largest test bed for collaborative filtering. It contains over 100 million ratings and was split into one training and one test subsets. The training set consists of 100,480,507 ratings for 17,770 movies by 480,189 users. The test set contains about 2.8 million ratings. All the ratings are ranged from 1 to 5.

We split the three data sets into training and test sets as follows: for Movielens and EachMovie, we randomly sample 80% ratings for each user as the training set and the rest 20% is used as the test set. These two data sets are very sparse and thus a lot of ratings are not observed, which may lead to biased evaluation results for precision. Therefore, we also construct a dense data set from the netflix data as follows: We first select 5000 items with the most ratings and then sample 10000 users with at least 100 ratings to construct a relatively dense data set. For this data set, we sample 20% ratings for each users as the training set and the rest ratings are used as the test set. For all three data sets, we generate five independent splits and report the averaged performance in our evaluations. Moreover, we exclude all ratings in the training set and use only the ratings in the test set when computing the evaluation metrics.

4.3 Comparison of Rounding Methods

In Section 3.3, we describe two methods for obtaining binary codes from the approximate real-valued solutions. We now compare the performance of these methods. To this end, we denote the method that rounding to closest binary codes described in Section 3.3.1 as Closest and the method using orthogonal transformation described in Section 3.3.2 by OrthTrans. We apply CFCodeReg and CFCodePair with binary codes of length 10 to all the three data sets and compare the binary codes obtained by Closest and OrthTrans. We report the performance on three data sets measured by precision in Figure 1. From Figure 1, we can see that the binary codes obtained by OrthTrans outperform the corresponding binary codes obtained by Closest, which suggests that the OrthTrans can obtain better approximations for binary codes. Intuitively, OrthTrans is more flexible than Closest through introducing the orthogonal transformation and thus enable us to obtain better approximation. We will use OrthTrans to obtain binary codes in the rest of our evaluations.

4.4 General Performance Results

4.4.1 Baseline Alternative Methods

We compare CFCodeReg and CFCodePair to the following baselines:

- **Spectral Hashing [21] (SH):** This method has been shown to be effective to learning binary codes. Specifically, it formulates the problem of learning binary codes as an eigenvalue problem for the similarity graph. In particular, the training ratings are viewed as the similarities between users and items and a bipartite graph is constructed between nodes representing users and items [23]. Then, spectral hashing is applied to obtain binary codes for users and items based on this graph.

- **BinMF:** In this baseline, we first apply low-rank matrix factorization to fit the training ratings and obtain low dimensional real-valued latent profiles for users and items. Then, we binarize these vectors to obtain binary codes through the orthogonal transformations described in Section 3.3.2 since it archives better performance as a rounding method.

[2]http://www.netflixprize.com/

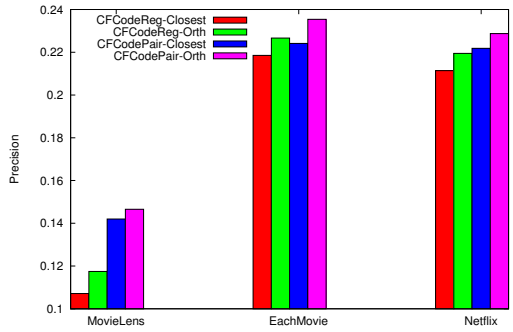


Figure 1: Performance of two rounding methods on MovieLens, EachMovie and Netflix data sets

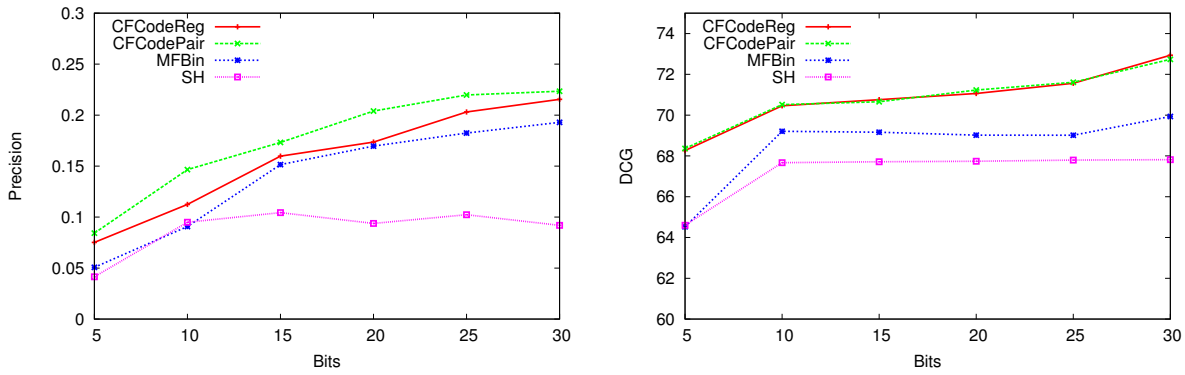
4.4.2 Performance Analysis

We apply the proposed CFCodeReg and CFCodePair to all three data sets and compare the obtained binary codes to the two baselines described in Section 4.4.1. Specifically, we plot the performance measured by DCG and precision with respect to the length of the binary codes in Figure 2, Figure 3 and Figure 4 for MoveiLens, EachMovie and Netflix data sets, respectively.

We can observe that DCG and precision of both CFCodeReg and CFCodePair increase in most cases with larger length of binary codes. Hence, the performance of CFCodeReg and CFCodePair improves when the number of bits increases. Therefore, we conclude that both methods can utilize the available bits to preserve the preference of users more accurately. We can also observe from the figures that the binary codes obtained by CFCodeReg and CFCodePair outperform other baselines in terms of DCG. Thus, the proposed method can better preserve the relative orders between items according to the preference of users. Moreover, the improvement over baselines in terms of precision indicates that the binary codes obtained by CFCodeReg and CFCodePair can be used to recommend interesting items to users accurately.

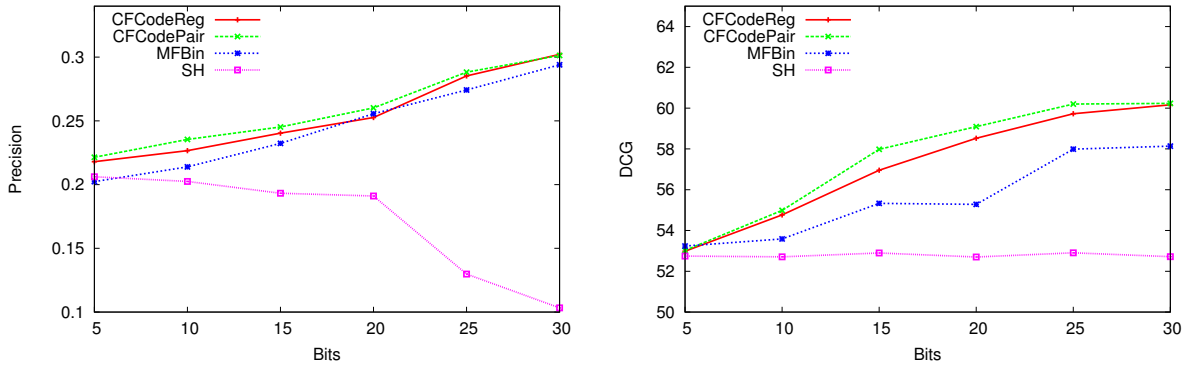
Comparing the performance of CFCodeReg and CFCodePair, we can find that CFCodePair outperforms CFCodeReg in most cases, which suggests that the pairwise loss function is more suitable for learning binary codes. This is because the pairwise loss function emphasis more on the orders between different items rather than their absolute ratings, which makes it a more reasonable loss function in the ranking scenario for collaborative filtering.

Another interesting observation is that SH does not work very well in our case. Specifically, it overfits the training data very quickly when the length of binary codes increases.



(a) Precision (b) DCG

Figure 2: Performance with respect to the length of binary codes on Movielens data set



(a) Precision (b) DCG

Figure 3: Performance with respect to the length of binary codes on EachMoive data set

By observing the results, we find that SH usually fits the training data very well. However, it frequently assign similar distances for users and items whose ratings are not in the training set. Therefore, its performance over the test set is reduced. In order to further investigate this point, we vary the length of binary codes and plot the variance of Hamming distances on unobserved ratings for binary codes generated by SH and CFCodeReg in Figure 5. We can observe that the variances produced by SH decrease when the length of binary codes grows. On the other hand, the variance generated by CFCodeReg are generally much higher than those generated by SH, which indicates that CFCodeReg generates more diverse codes when the length of binary codes increases. We think the reason is that SH usually fits the observed similarities while fails to predict the unobserved ones. This observation verifies that CFCodeReg can not only fit the observed preferences very well, but it also can predict the unobserved preference accurately.

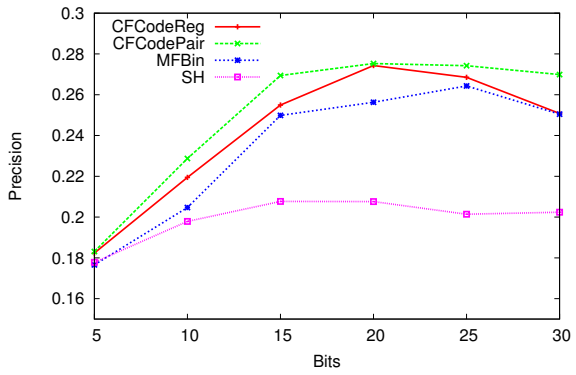
4.4.3 *Impact of Parameters*

We investigate the impact of the regularization parameter λ for the propose methods. To this end, we report the performance of CFCodeReg and CFCodePair measured by DCG with respect to different values of λ in Figure 6. We only show the results on the MovieLens data set due to the lack of space. From Figure 6, we can observe that the performance measured by DCG first increases and then decreases in most cases, which indicates that a good value of λ can enhance the learning process and thus improves the accuracy of learnt binary codes. In general, the value of λ can be determined by *cross validation*.

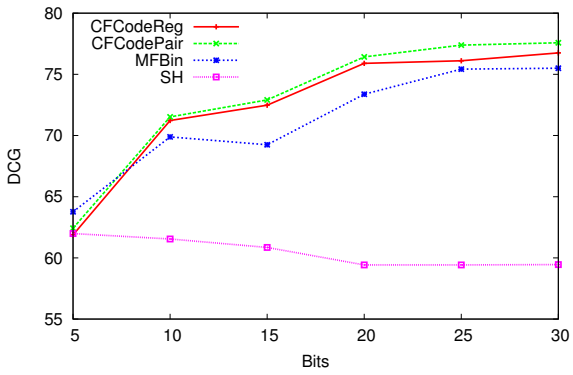
In our experiments, the relaxed optimization problem of Equation (5) is solved by LBFGS, which is an effective iterative solver for optimization problems. In Figure 7, we present the performance measured by DCG with respect to the number of iterations on MovieLens data set. We can observe from Figure 7 that the performance measured by DCG both training and test set increases when the number of iterations grows. The training process usually converges in about one hundred iterations.

4.4.4 *Compare with Low-rank Matrix Factorizations*

It is also interesting to compare CFCodeReg to the low-rank matrix factorization method that is widely exploited for collaborative filtering. Since CFCodeReg is restricted to use binary codes in order to facilitate fast search, it can be viewed as an approximation of the low-rank factorization methods. Thus, it is natural to investigate how close CFCodeReg can approximate the performance of low-rank matrix factorization. To this end, we vary to length of the binary codes from 10 to 110 and report the performance measured by DCG on MovieLens data set in Figure 8. We also report the performance of low-rank matrix factorizations when varying the rank of the factorization. We can see that the performance of CFCodeReg increases in general when the length of the binary codes grows and become very close to the performance of low-rank matrix factorizations. On the other hand, the performance of low-rank matrix factorizations is slightly reduced when the number of latent dimensions increases which is generally explained by the overfitting of the training data.



(a) Precision



(b) DCG

Figure 4: Performance with respect to the length of binary codes on Netflix data set

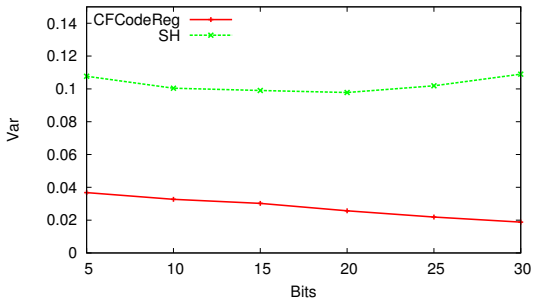


Figure 5: Variance of predicted similarity on unknown ratings with respect to the length of binary codes

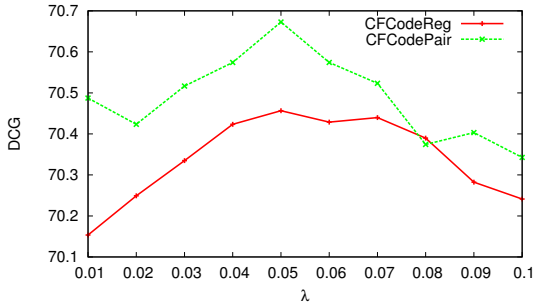


Figure 6: Performance measured by DCG with respect to the regularization parameter λ on MovieLens data set

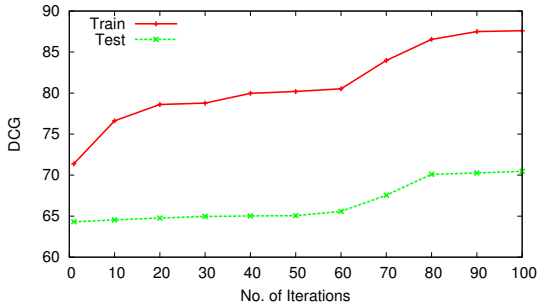


Figure 7: Performance measured by DCG with respect to the number of iterations on MovieLens data set

4.5 Recommendation Efficiency

We also compare the efficiency of obtaining top-K recommendations. In particular, for MF we compute the predicted scores for every item for a given user and then select top-K items with the highest scores. For CFCodeReg, we retrieve items with binary codes within Hamming distance 3 to the binary codes of the user. We measure efficiency by the total time required to generate recommendations for all users. To this end, we run the recommendation program for 10 times and report the average running time. The evaluation is conducted on a server with 16G main memory and use one of its eight 2.5GHz cores. On MovieLens data set, CFCodeReg takes 0.586 seconds to process all users while MF takes 64.9 seconds. The significant efficiency improvement is expected and can be explained by the fact that CFCodeReg only goes through a small fraction of items while MF computes the prediction for all items. This confirms that recommendation efficiency can be significantly improved by utilizing binary codes.

5. CONCLUDING REMARKS

In this paper, we address the problem of learning binary codes that preserves the preferences of users to items. In particular, we propose a framework that constructs binary codes such that the Hamming distances of a user and her preferred items are small. By applying two loss functions, the problem is formulated as a discrete optimization problem defined on the training ratings data set. It turns out that the resulting optimization problem can be solving approximately by transforming the objective function and relaxing the variables to real values. Moreover, we study two methods to obtain the binary codes from the real-valued approximations. Experiments on three data sets show that the proposed methods outperform several baselines and thus and can preserve the preference of users more accurately.

For future research directions, we plan to further investigate other methods for solving the discrete problem more accurately. Specifically, we can investigate how to apply semidefinite programming for relaxing the original problem. Another direction is to study how to learn binary codes incrementally. In particular, we would like to construct the binary codes bit by bit in a sequential and incremental manner. It has the advantage that new bits of binary codes can be introduced to improve the accuracy without re-training all of the bits. Finally, the problem of incorporating features for users and items, such as demographical features for users and descriptions of items, to learn the binary codes can be also very interesting.

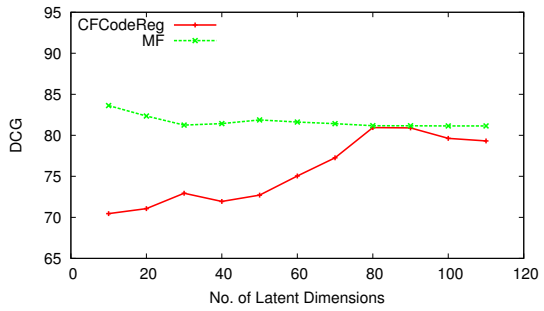


Figure 8: Comparison of low-rank matrix factorization and CFCodeReg on MovieLens data set

6. ACKNOWLEDGEMENT

Part of the work is supported by NSF IIS-1116886, NSF IIS-1049694 and NSFC 61129001/F010403.

7. REFERENCES

[1] G. Adomavicius and a. Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, June 2005.

[2] D. Agarwal, B. Chen, and P. Elango. Fast online learning through offline initialization for time-sensitive recommendation. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 703–712. ACM, 2010.

[3] Y. Bachrach and R. Herbrich. Fingerprinting Ratings For Collaborative Filtering Theoretical and Empirical Analysis. *String Processing and Information Retrieval*, pages 25–36, 2010.

[4] Y. Bachrach, E. Porat, and J. Rosenschein. Sketching techniques for collaborative filtering. In *Proceedings of the 21st international joint conference on Artifical intelligence*, pages 2016–2021. Morgan Kaufmann Publishers Inc., 2009.

[5] A. S. Das, M. Datar, A. Garg, and S. Rajaram. Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th international conference on World Wide Web - WWW '07*, page 271, New York, New York, USA, 2007. ACM Press.

[6] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry - SCG '04*, page 253, New York, New York, USA, 2004. ACM Press.

[7] J. He and W. Liu. Scalable similarity search with optimized kernel hashing. *Proceedings of the 16th ACM SIGKDD*, pages 1129–1138, 2010.

[8] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing - STOC '98*, pages 604–613, New York, New York, USA, 1998. ACM Press.

[9] H. Jégou, T. Furon, and J.-J. Fuchs. Anti-sparse coding for approximate nearest neighbor search. *Arxiv preprint arXiv:1110.3767*, (October):577–580, Oct. 2011.

[10] J. Kekäläinen. Binary and graded relevance in IR evaluations - Comparison of the effects on ranking of IR systems. *Information Processing and Management*, 41:1019–1033, 2005.

[11] Y. Koren. Factor in the neighbors: Scalable and accurate collaborative filtering. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 4(1):1–24, 2010.

[12] B. Kulis and T. Darrell. Learning to hash with binary reconstructive embeddings. *Proceedings of Advances in Neural Information Processing Systems*, 2009.

[13] W. Liu, J. Wang, S. Kumar, and S. Chang. Hashing with Graphs. In *Proceedings of the 28th International Conference on Machine Learning*, 2011.

[14] M. Norouzi and D. Fleet. Minimal Loss Hashing for Compact Binary Codes. In *Proceedings of the 28th International Conference on Machine Learning*, volume 1, 2011.

[15] A. Paterek. Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of KDD Cup and Workshop*, volume 2007, pages 5–8. Citeseer, 2007.

[16] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. Factorizing personalized Markov chains for next-basket recommendation. *Proceedings of the 19th international conference on World wide web - WWW '10*, page 811, 2010.

[17] R. Salakhutdinov and G. Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978, July 2009.

[18] G. Shakhnarovich, P. Viola, and T. Darrell. Fast pose estimation with parameter-sensitive hashing. In *Proceedings Ninth IEEE International Conference on Computer Vision*, pages 750–757 vol.2. IEEE, 2003.

[19] X. Su and T. M. Khoshgoftaar. A Survey of Collaborative Filtering Techniques. *Advances in Artificial Intelligence*, 2009(Section 3):1–20, 2009.

[20] J. Wang and S. Kumar. Sequential projection learning for hashing with compact codes. *International Conference on Machine Learning*, 2010.

[21] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *Neural Information Processing Systems*, number 1, pages 1–8, 2008.

[22] S. X. Yu and J. Shi. Multiclass spectral clustering. In *Proceedings Ninth IEEE International Conference on Computer Vision*, number 0, pages 313–319 vol.1, Washington, DC, USA, 2003. IEEE.

[23] D. Zhang, J. Wang, D. Cai, and J. Lu. Laplacian co-hashing of terms and documents. *Advances in Information Retrieval*, pages 577–580, 2010.

[24] D. Zhang, J. Wang, D. Cai, and J. Lu. Self-taught hashing for fast similarity search. In *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 18–25. ACM, 2010.

[25] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software*, 23(4):550–560, Dec. 1997.