HITSnDIFFS: A fast algorithm for consecutive ones with applications in item labeling

Subhodeep Mitra

CMU-RI-TR-19-30

Submitted in partial fulfillment of the requirements for the degree of Masters in Robotics Research

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213
May 2019

Thesis Committee
R. Ravi
Stephen Smith
Nihar B. Shah
Jingyan Wang

Abstract

We analyze a general problem in a crowd-sourced setting: *users* pick a *label* from a set of candidates for a set of *items*; the problem is to determine the most likely label for each item, as well as a ranking of the users based on their ability to pick correct labels for the items.

We start by defining an idealized setting for this problem where the relative performance of users is *consistent* across items, and observe that the response matrices in this ideal case obey the *Consecutive Ones Property* (C1P). While the consecutive ones problem is well understood algorithmically with various discrete algorithms, we devise a simple variant of the HITS algorithm called "HITSnDIFFS" and prove that it can recover the ideal C1P-permutation in case it exists.

Unlike fast combinatorial algorithms for finding the consecutive ones permutation (if it exists), HITSnDIFFS also returns an ordering when such a permutation does not exist, thus providing a *principled heuristic* for the problem that returns the correct answer in the ideal case.

We compare HITSnDIFFS's performance with previously proposed iterative and spectral algorithms to solve similar real-world problems. Our experiments on both real and simulated datasets show that HITSnDIFFS produces user rankings and item labelings with superior accuracy compared to the various scalable methods, and is competitive with other slower state-of-the-art methods while providing an asymptotic improvement in running time.

Acknowledgments

I would like to express deep gratitude to my advisor, Prof. R Ravi. I am greatly indebted to him for all his support, guidance and encouragement over the last few years. The stimulating discussions we had on various topics helped me learn a lot.

I would also like to thank Prof. Wolfgang Gatterbauer for his guidance and advice along the way. His close collaboration and insights were always very helpful. I also thank Tanvi Bajpai for supporting me in my work.

Finally, I would like to thank my committee - my co-advisor Prof. Stephen Smith, Prof. Nihar Shah, and Jingyan Wang for their valuable suggestions and feedback.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The problem. We study the following general problem: We have m users and n items, each of which has up to k possible labels to describe it. Each user picks one of these labels for each of the items. Derive a principled way to *determine a ranking for the abilities of the users, and a ranking for the appropriateness of each of the labels for describing the items* based solely on the user label choices.

Applications. This general problem occurs in a wide range of problems related to crowdsourcing and truth discovery [33]. The crowd-labeling approach has been refined for various tasks, such as query answering [9], annotating Twitter data [8], and various other labeling tasks [28, 30]. As a consequence, this problem has been studied across a number of variants over the years: whether the options among item labels are binary [6, 11, 12, 16] or multiple choices [7, 34, 35, 36]; whether we are only interested ranking the users [1, 4] or only finding the most likely label options [11, 16, 25] or both [6, 7, 34, 35, 36]. In typical item labeling applications, the labels may be crowdsourced from the users. Furthermore, each user may not label all images, and all images may not have the same number of labels. Therefore, we consider this general scenario, to be formalized subsequently.

Our approach. We are interested in the general scenario, where items have varying numbers of labels, and users can pick any number of labels for any item. We start by studying the following core underlying problem: there are m users and n items; every user chooses exactly one among a fixed number k of labels for each item. We first define an ideal case when the user responses are consistent across the items and characterize it using a matrix property. We then use a spectrally inspired method to give a very efficient algorithm for reconstructing such ideal orderings if they exist. Our method generalizes to the non-ideal case and allows us to compare it with other methods developed for ranking users or items or both. We then repurpose a generative model from item response theory used for modeling difficulties of multiple choice quiz questions to produce synthetic data for our problem. Using this and real-world data from image labeling applications from previous work, we demonstrate that our new method is better or competitive with existing alternates while being asymptotically faster.

Our assumption that k is fixed across items is useful only in proving that our method reconstructs ideal orderings. The method itself generalizes to the cases when (*i*) different items have different number of options, (*ii*) when each user picks labels for only a subset of the items, and (*iii*) when a users chooses more than one label for an item. Details are in section 5.2.

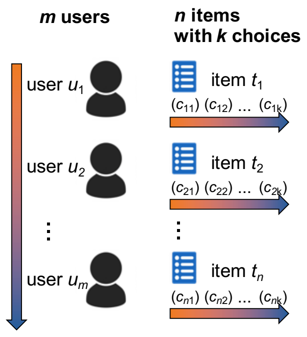
1.1 Problem Definition

Consider a setting with m users choosing an option for each of n items, where each item has exactly k options for its label.

The choices from all users for all items can be represented as an $(m \times k \times n)$-dimensional binary response tensor $\mathcal{C}$, in which each 1 in position (i, h, j) represents a choice of option h for item j (thus choice c_{jh}) by user i, and 0 otherwise (see top of Figure 1.1b; to simplify representation, we do not show 0 entries in tensors). Since each user chooses exactly 1 option for each item, any $(1 \times k \times 1)$-dimensional fiber contains exactly one single 1 entry, and the other entries are 0. This response tensor can be flattened into an isomorph $(m \times kn)$-dimensional matrix $\mathbf{C}$ (see bottom of Figure 1.1b). To summarize, we have an $m \times kn$ binary response matrix $\mathbf{C}$, where a row represents a user's choices of an option for each item and each column represents an option for some item. Also note that the number of non-zeros in the response matrix is mn and each row has sum n.

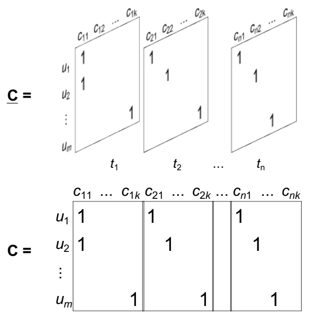
Each user u_i is assumed to have an associated *selection ability* θ_i, and each option h for item t_j has an associated quality α_{jh}. When presented with the response matrix $\mathbf{C}$, our desired algorithm should return a ranking of users that is close to the ranking according to $\boldsymbol{\theta}$, and the ranking of options for item j should be close to the ranking according to $\boldsymbol{\alpha}_{j:}$.

The ideal case with consistent responses. We argue that a response matrix is *consistent* if there is an unambiguous ordering of users according to their abilities that is reflected in their responses. This way, the ability is a unique skill that is tested across their responses to all items. In this ideal case, if a user u_1 chooses a better option for an item t_1 than user u_2, then she



m users
n items with *k* choices
user u_1
user u_2
user u_m
item t_1
(c_{11}) (c_{12}) ... (c_{1k})
item t_2
(c_{21}) (c_{22}) ... (c_{2k})
item t_n
(c_{n1}) (c_{n2}) ... (c_{nk})

(a) Our problem



C =
u_1
u_2
u_m
t_1
t_2
t_n
c_{11} ... c_{1k} c_{21} ... c_{2k} ... c_{n1} ... c_{nk}

(b) The input

Figure 1.1: (a) Our setup is very general: we have m users who choose one from k choices of labels for each of n items. Based on their responses, we like to determine the relative ranking of users' abilities, and the relative ranking of labels appropriateness for each item. (b) The input to our algorithm is the $(m \times k \times n)$ response tensor $\mathcal{C}$, or equivalently its flattened $(m \times kn)$ response matrix $\mathbf{C}$.

must also choose an equal or better option for any other item t_2 to reflect this consistency. This implies there is an implicit ordering among the choices for each item from best to worst and the better users pick better options for every item. Formally, assume that the abilities θ_i are all distinct, and also that for every item t_j, the qualities α_{jh} over all the options are distinct so that there is a unique linear ordering of the users, and of the options for each item.

Definition 1 (Consistent Responses)**.** *A response matrix* $\mathbf{C}$ *is* consistent *if there exists an assignment of selection abilities* $\boldsymbol{\theta}$ *and item qualities* $\boldsymbol{\alpha}$*, s.t. for any pair of users* u_1 *and* u_2 *with* $\theta_1 > \theta_2$*, and for any item* t_j *where* u_1 *chooses option* h_1 *and* u_2 *chooses option* h_2*, we have* $\alpha_{jh_1} \geq \alpha_{jh_2}$*.*

Consecutive ones property (C1P). We observe that consistent response matrices, when row-sorted according to user abilities, satisfy a widely studied ordering property in seriation, called the *consecutive ones property* (C1P). We will follow the notation from seriation theory, and call this a *P-matrix.*

Definition 2 (C1P, P-matrix & pre-P-matrix [1])**.** *A binary matrix satisfies* C1P *if in each column, all the* 1*'s are consecutive. A binary matrix that satisfies* C1P *is also called a* P-matrix*. If the rows of a matrix can be permuted so it becomes a P-matrix, we call it a* pre-P-matrix*.*

$$\begin{array}{c|ccc|ccc|}
 & c_{11} & c_{12} & c_{13} & c_{21} & c_{22} & c_{23} \\
\hline
u_3 & 1 & & & 1 & & \\
u_4 & 1 & & & & 1 & \\
u_1 & & 1 & & & & 1 \\
u_2 & & 1 & & 1 & & \\
u_5 & & & 1 & & 1 & \\
\end{array}$$

(a) **C**

$$\begin{array}{c|ccc|ccc|}
 & c_{11} & c_{12} & c_{13} & c_{21} & c_{22} & c_{23} \\
\hline
u_1 & & 1 & & & & 1 \\
u_2 & & 1 & & 1 & & \\
u_3 & 1 & & & 1 & & \\
u_4 & 1 & & & & 1 & \\
u_5 & & & 1 & & 1 & \\
\end{array}$$

(b) $\mathbf{C}'$

Figure 1.2: Example 1: pre-P-matrix **C** and P-matrix $\mathbf{C}'$ (vertical separators between columns c_{13} and c_{21} are just for illustration).

In other words, no 0's appear between any two 1's in a column in a P-matrix. To see that consistent responses with users sorted by abilities θ give a P-matrix, suppose for a contradiction that a column corresponding to an option for an item has two or more blocks of ones. Then the users corresponding to the zeros in between these blocks will have chosen another option for which the quality is strictly higher or lower than that of this option since the choice qualities are assumed to be distinct. But this violates consistency since the rows are ordered by user ability.

Observation 1 (Consistent Responses give Consecutive Ones Matrices)**.** *A response matrix* **C** *is consistent iff it is is a pre-P-matrix.*

Consequently, ranking the users in the case of consistent responses corresponds to the problem of determining a permutation of the rows of **C** so that the result obeys C1P.

Example 1 (pre-P-matrix)**.** *Matrix* **C** *in Figure 1.2a is not a P-matrix because it has two* 0*'s between* 1*'s in the column* c_{21}*. However, it is a pre-P-matrix because after swapping rows* (u_1, u_2) *with* (u_3, u_4)*, it does satisfy* C1P *(see Figure 1.2b). Also notice that any P-matrix automatically determines a ranking on the options consistent with the ranking of the users. For Figure 1.2b, these are* $(c_{12} \to c_{11} \to c_{13})$ *and* $(c_{23} \to c_{21} \to c_{22})$*.*

The general case. Our goal is to develop a fast and principled algorithm for solving the problem in the more general practical case when the response matrix is *not pre-P*, yet returns a P-matrix in the special case of pre-P matrix inputs. To that end, we review prior work on reconstructing P-matrices and for the general case.

Method	C1P	non pre-P	$k \geq 2$	scalable	complexity
BL [4]	✔	✘	✔	✔	$\mathcal{O}(mn)$
ABH [1]	✔	✔	✔	✔	$\mathcal{O}(mn)$
EM [7]	✘	✔	✔	✘	–
DDKR1 [6]	✘	✔	✘	✔	$\mathcal{O}(mn)$
DDKR2 [6]	✘	✔	✘	✘	$\mathcal{O}(m^2n)$
GKM [12]	✘	✔	✘	✔	$\mathcal{O}(mn)$
HITSnDIFFS	✔	✔	✔	✔	$\mathcal{O}(mn)$

Table 1.1: A comparison of the various desirable properties achieved by prior work and our new method HITSnDIFFS: "C1P" refers to reconstruction of the correct ordering when the input is pre-P; "non pre-P" refers to whether the algorithm can be applied to non pre-P instances; "$k \geq 2$" refers to whether the algorithm can handle non-binary choices; and "scalable" refers to linear time complexity per iteration (which is reported in the last column).

1.2 State-of-the-art

Some of the state-of-the-art methods for item labeling and algorithms for reconstructing the C1P property are discussed below.

BL. Booth and Leuker [4] proposed the fastest known algorithm to find all possible permutations of the rows that reconstruct the C1P ordering in time linear in the number of nonzero entries in the matrix, thus taking time $O(mn)$ in our setting. Their method constructs a "PQ-tree", which represents all possible permutations of the rows that reconstruct the C1P ordering if at least one C1P ordering exists, and otherwise informs that there is no such ordering.

Since their method fails to produce an ordering of the rows when the matrix is not a pre-P-matrix, it therefore cannot be used as a general heuristic for simulated or real-world datasets that are not ideal.

ABH. Atkins et al. [1] proposed an elegant spectral sort method to determine whether a matrix obeys C1P, thus giving a rare continuous method to solve a seemingly combinatorial ordering problem. The method returns a PQ tree (all possible permutations of the rows that reconstruct the C1P ordering) like BL, but in addition, it is *general* and also adapts as a real-world heuristic when the input matrix does not obey C1P. However even though it can be used as a heuristic on datasets that do not satisfy C1P, the accuracy of this method on many simulated and real datasets is poor.

EM. Dawid and Skene [7] proposed an approach for label aggregation, where they assume that each user has a latent confusion matrix for labeling. The off-diagonal elements represent the probabilities that a user mislabels an arbitrary item from one class to another while the diagonal elements correspond to her accuracy in each class. The confusion matrices and true labels are jointly estimated by maximizing the likelihood of observed labels. This method is very popularly used in item labeling, but does not always return a C1P ordering in the ideal case.

DDKR1, DDKR2. Dalvi et al. [6] proposed two methods DDKR1 and DDKR2 that simultaneously output a measure of user ability as well as the label for each item, along with error bounds for arbitrary user-label graphs, The error bound for both the algorithms they propose is governed by the expansion of the graph. By constructing an *assigment matrix* $\mathbf{A}$ and a *rating matrix* $\mathbf{B}$, both the algorithms rely on finding eigenvectors of the symmetric matrices $\mathbf{A}^\intercal\mathbf{A}$ and $\mathbf{B}^\intercal\mathbf{B}$. The difference between the algorithms is that DDKR1 uses the ratio of the eigenvectors for computing the user ability ranking and the labels, while DDKR2 uses the eigenvectors of the ratio (i.e. the Hadamard or entrywise division) of the matrices. In the scenario where every user labels every item, both DDKR1 and DDKR2 produce the same user ability output; the performance of the two methods starts deviating as the number of responses decreases. Among the two, DDKR2 requires quadratic time while DDKR1 can be run in linear time using the power method.

GKM. Ghosh et al. [12] proposed a method **GKM** that only outputs the item labels. The robustness of this algorithm to manipulation by adversarial or strategic raters is also analyzed with error bounds. Like DDKR1 and DDKR2, this method also involves calculating the first eigenvector of a symmetric matrix, and so can be run in linear time using the power method.

Thus, with the exception of ABH and BL, none of the methods $\mathrm{DDKR1}, \mathrm{DDKR2}, \mathrm{GKM}, \mathrm{EM}$ are guaranteed to return a consecutive ones ordering if the input matrix is pre-P. However, BL cannot be used in general crowdsourced settings since it cannot generalize to datasets that do not satisfy C1P, and while ABH can be used as a heuristic for real datasets, its performance is generally poorer than the other methods (as shown in Chapter 5), as the method is mostly aimed at solving the C1P problem.

Apart from the method BL designed exclusively to satisfy C1P, we compare our own spectral method introduced in this paper to the spectral methods DDKR1, DDKR2, GKM and ABH, as well as the classic expectation-maximization algorithm EM. There are also several other newer crowdsourcing item-labeling algorithms, many of which are based on expectation-maximization, along with bounds on rates of convergence and error rates. Gao and Zhou [11] proved convergence rates of a projected EM algorithm for the Dawid-Skene estimator. The revealed exponent in the rate of convergence was shown to be optimal via a lower bound argument. Gao et al. [10] established matching upper and lower bounds under the classic Dawid-Skene model - the minimax rate of misclassification for estimating the truth from crowdsourced labels were shown to have upper and lower bounds with exact exponents that match each other. Khetan and Oh [16] introduced a novel adaptive scheme to assign labeling tasks to users, which was shown to be optimal given a fixed budget on the number of responses collected on the crowdsourcing system. Karger et al. [13, 14] tackled the problem of minimizing the total number of labeling task assignments to achieve a target overall reliability, by introducing a new algorithm for deciding which tasks to assign to which workers, and for inferring correct answers from the workers' answers. They showed that the method was order-optimal through comparison to an oracle that knows the reliability of every worker. Liu et al. [19] transformed the problem of aggregating crowdsourced labels into a standard inference problem in graphical models, and applied approximate variational methods, including belief propagation (BP) and mean field (MF). They showed that by choosing a good distribution on the workers' reliability, their methods are competitive with state-of-the-art algorithms based on more complicated modeling assumptions. Zhang et al. [32] proposed a two-stage efficient algorithm for multi-class crowd labeling problems - the first stage uses the spectral method to obtain an initial estimate of parameters, and the second stage refines the estimation by optimizing the objective function of the Dawid-Skene estimator via the EM algorithm. This algorithm was shown to achieve the optimal convergence rate up to a logarithmic factor, and is comparable to the most accurate empirical approach. Shah et al. [24] proposed a permutation-based model for crowd labeled data that is a significant generalization of the common Dawid-Skene model, which offers significant robustness in estimation, while incurring only a small additional statistical penalty as compared to the Dawid-Skene model. They also proposed a computationally-efficient method that is uniformly optimal over a class intermediate between the permutation-based and the Dawid-Skene models, and is uniformly consistent over the entire permutation-based model class.

1.3 Our method "HITSnDIFFS"

We present a new method for item labeling called HITSnDIFFS that *preserves the ideality properties of* BL *and* ABH*, while retaining the speed of the iterative methods that do not return the correct solution in the ideal case*

($\mathrm{DDKR1}, \mathrm{DDKR2}, \mathrm{GKM}, \mathrm{EM}$*), and also performs as well in various measures of accuracy in the general case on both simulated and real-life data.* Our method, is a variant of the popular HITS algorithm [18], applied to the the bipartite graph corresponding to response matrix with two crucial differences: 1) HITS relies on taking the sums of adjacent components of the graph in each step; we use averages instead of sums. 2) we prove that if the input response matrix obeys C1P, then the ordering of the users corresponding to the *second largest eigenvector* of the update matrix of the average version of HITS reconstructs this ordering. To construct the second largest eigenvector of the average version of HITS, one could proceed by using deflation methods [31] that subtract out the subspace corresponding to the first eigenvector from the original matrix and applying the power method to find the largest eigenvector in the resulting matrix. However, such methods are known to suffer from numerical issues [22], especially as the first two eigenvalues of a matrix become more strongly graded.

Instead, inspired by the proof methodology of ABH, we propose a new intermediate step taking differences of user scores in the updates of the averaging version of HITS, and prove that when the response matrix obeys C1P, our new algorithm that we term "HITS and DIFFS" (or HITSnDIFFS) reconstructs the correct ordering of the rows. It is possible that despite the deflation method suffers from numerical issues, the relative ordering of elements in the eigenvector is still preserved, however it is unproven, and thus we rely on our new intermediate step of user score differences. As the number of users m scales, computing the update matrix explicitly takes time $\Theta(m^2 n)$. However each iteration of the power method on this matrix using the average HITS and DIFFS idea (or alternatively by the deflation method) takes time only $O(mn)$ by using factored matrix-vector products instead of matrix-matrix products. Furthermore, on synthetic and real-world data, our fast method extends to compute a heuristic ordering that is competitive with other spectral methods.

1.4 Contributions of this thesis

1. We define a notion of *consistent responses* in crowd-sourced label choice experiments by using matrices with C1P from seriation theory that we argue should be retained by any principled approach to solving the problem.

2. We propose and analyze a simple variant of the HITS algorithm of Kleinberg [18] inspired by the proof technique of the spectral method

by Atkins et al. [1] that we call HITSnDIFFS for ranking users and label choices. Our method is both scalable and competitive in quality with other slower methods.

3. We show that HITSnDIFFS recovers the consecutive ones ranking of users in the ideal case when there is a unique order obeying C1P.

4. Unlike fast combinatorial algorithms for finding the C1P ordering only if one exists, HITSnDIFFS can deal with the general case when such orders do not exist. We can thus compare it with other methods for item labeling $(\mathrm{DDKR1}, \mathrm{DDKR2}, \mathrm{GKM}, \mathrm{EM})$ on synthetic and real data sets. even though these methods cannot guarantee to reconstruct the ideal ordering in the consistent case. We also show superior performance compared to ABH on these synthetic and real data sets.

5. We use ideas from Item Response Theory (IRT) to define the BSTS model as a new synthetic data generator for the crowd-sourced label choice problem. We then show that HITSnDIFFS works surprisingly well over synthetic data from this model, as well as in many real data sets involving binary and multiple choice options for label selection for items, to give accurate item labels as well as user orderings.

Outline. In the rest of the paper, we provide some more background and related work (Chapter 2), describe our new method HITSnDIFFS and prove its surprising properties (Chapter 3), describe our synthetic and real data sets (chapter 4), show experimental results of our method's relative performance (Chapter 5), and discuss future work (chapter 6) and conclusions (chapter 7).

Chapter 2

Preliminaries and Related work

2.1 The method of Atkins et al. (ABH)

Given a pre-P matrix $\mathbf{C}$, the spectral method of Atkins et al. [1] (which we refer to as "ABH") indirectly uses the product of the response matrix $\mathbf{C}$ and its transpose to form $\mathbf{C} \cdot \mathbf{C}^\intercal$ to find all possible orderings of users that reconstruct the C1P property for $\mathbf{C}$. Their method has two main parts: showing that if $\mathbf{C}$ is a P-matrix, (1) then $\mathbf{C} \cdot \mathbf{C}^\intercal$ has the special property of being an R-matrix (which we will define below), and (2) the eigenvector corresponding to the *second smallest* eigenvalue of the related Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{C} \cdot \mathbf{C}^\intercal$ is monotonic (i.e. its elements are in either increasing or decreasing order). Here $\mathbf{D}$ is a diagonal matrix filled with the row sums of $\mathbf{C}\mathbf{C}^\intercal$. Given these two facts, they prove that if $\mathbf{C}$ is a pre-P matrix, then the eigenvector corresponding to the *second smallest eigenvalue* of $\mathbf{L}$ can be found by a method called SpectralSort [1] to find all row permutations that reconstruct the C1P property for $\mathbf{C}$.

Definition 3 (R-matrix [1])**.** *A matrix* $\mathbf{A}$ *is called an* R-matrix *if it is symmetric and*

$$\begin{aligned} a_{ij} &\le a_{ik} \text{ , for } \quad j < k < i \\ a_{ij} &\ge a_{ik} \text{ , for } \quad i < j < k \end{aligned}$$

If the rows and columns of $\mathbf{A}$ *can be permuted symmetrically to become an R-matrix, then we say that* $\mathbf{A}$ *is pre-R.*

Intuitively, the definition describes a matrix where, the values fall off as we move away from the diagonal along any row. Since for $\mathbf{C} \cdot \mathbf{C}^\intercal$, each matrix entry represents the dot product of the responses of two users, when the response matrix is sorted by user ability, the entries represent the number of common responses for a pair of users. These values are highest for a user with herself and fall off as we move away along the correct linear ordering of the abilities represented in the correctly sorted response matrix. The R-matrix property is preserved if we add a constant to all off-diagonal entries, so we can assume WLOG that all off-diagonal values are non-negative.

Our method. We will prove that a similar theoretical guarantee holds for the variant of HITS that we will propose. In particular, given a pre-P matrix $\mathbf{C}$, we will define an update matrix $\mathbf{U}$ (which is a function of $\mathbf{C}$) to find all possible orderings of users that reconstruct the C1P property for the matrix $\mathbf{C}$. To do this, we will prove the following two statements: in lemma 6, we show that if $\mathbf{C}$ is a P-matrix, then $\mathbf{U}$ is an R-matrix, and in lemma 7, we show that the eigenvector corresponding to the second *largest* eigenvalue of $\mathbf{U}$ is monotonic. These two statements imply that if $\mathbf{C}$ is a pre-P matrix, then the eigenvector corresponding to the second largest eigenvalue of $\mathbf{U}$ can be used to find all orderings that reconstruct the C1P property for the matrix $\mathbf{C}$. However, we will do this without explicitly computing the update matrix $\mathbf{U}$ - by adding a new user score difference layer to the bipartite graph representing users and item labels and extending the power method to this construct, we perform updates directly on the graph corresponding to the response matrix.

2.2 Hubs and Authorities (HITS)

The HITS algorithm by Kleinberg [18] implements the power method to converge on a set of hub and authority scores in a bipartite graph where the hubs pick and point to a subset of the authorities. If the link structure of the p hubs to q authorities is captured in a $p \times q$ binary matrix $\mathbf{M}$, then the hub scores converge to the largest eigenvector of the matrix $\mathbf{M}\mathbf{M}^\intercal$. These scores have the nice property that the hub scores are proportional to the sum of the authority scores of the nodes they point to and the authority scores are proportional to the sum of the scores of the hubs pointing to them, thus reflecting a mutually consistent set of scores.

Our method. We build upon this fundamental idea of updating scores in a bipartite graph by iterative summation of scores from one side, and updating the other side accordingly. However, we need to modify the algorithm in two ways: 1) our precursor to HITSnDIFFS is an averaging version of HITS where instead of summing the scores of the authorities that a hub points to, we take their *average* value as the new hub score. 2) We further modify this average version to use differences among the hub score vectors to compute the ordering of the *second largest eigenvector* of the update matrix in our HITSnDIFFS method. We implement this method by performing updates in a *tripartite* instead of a bipartite graph.

Chapter 3

The HITSnDIFFS Algorithm

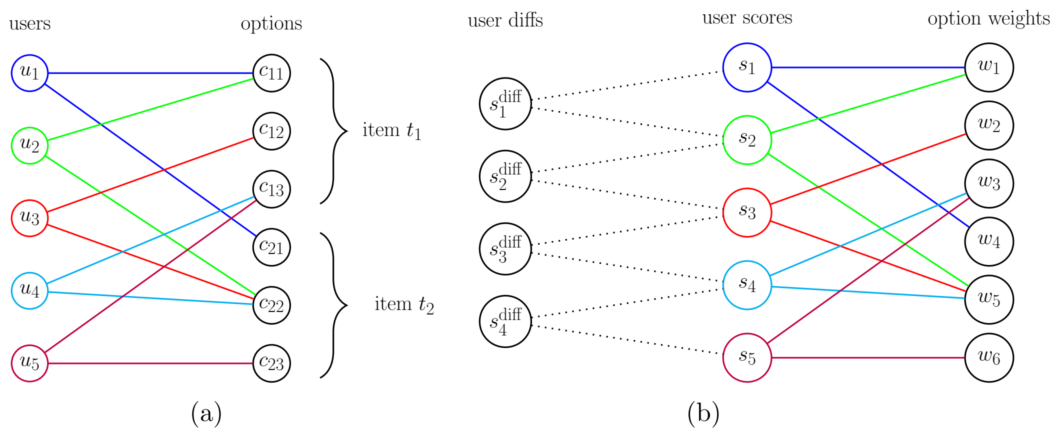
We describe a natural averaging version of the HITS algorithm that we call "avgHITS." Our key observation is that the second eigenvector of the update matrix corresponding to avgHITS reconstructs the row ordering with C1P if one exists and is unique. We then describe an efficient variant that we call "HITSnDIFFS" to find this second eigenvector: it uses an additional vector of differences between adjacent scores and updates it in the standard loop of the avgHITS algorithm to compute the ordering we require. We then compare its time complexity with other methods and prove that it returns the correct ordering when the input responses are consistent.

3.1 "avgHITS": a precursor to "HITSnDIFFS"

As shown in Figure 3.1a, we can construct a bipartite graph $G = (L \cup R, E)$ equivalent to the response matrix $\mathbf{C}$:

- Partition L contains a vertex for each user: $L = \{u_1, ..., u_m\}$.
- Partition R is a collection of n vertex sets $R = \{I_1, ..., I_n\}$, representing items. Each set I_j contains k vertices: $I_j = \{c_{j1}, ..., c_{jk}\}$; here, c_{jh} represents option h of item j.
- We add an edge between a user u_i and an option c_{jh} if user i picks option h for item j.

Define $\mathbf{s}$ as a $(m \times 1)$ user score vector with one score for each user. Also define $\mathbf{w}$ as an $(kn \times 1)$ option weight vector denoting weights for each of the kn options, according to their order in the response matrix $\mathbf{C}$.



$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

(c)

Figure 3.1: (a): Bipartite graph of users and item options they choose. (b): *Tripartite graph* of user scores, item weights, and user diffs that are used by HITSnDIFFS. (c): Response matrix $\mathbf{C}$ corresponding to the bipartite graph from (a).

avgHITS is the modification of the popular HITS update rule that uses *averages* instead of *sums* to iteratively update the user scores and option weights: an option weight c_{jh} is updated to be the *average* of the scores of all users who picked it, and the user score s_i of the i^{th} user is updated to be the *average* of the weights of all the options that the user i picked.

An equivalent matrix formulation of the avgHITS rule is as follows. Denoting columns of $\mathbf{C}$ by $\mathbf{C}_{:j}$, define column-normalized response matrix $\mathbf{C}^{\text{col}}$ of size $m \times kn$ with columns $\mathbf{C}^{\text{col}}_{:j}$ by:

$$\mathbf{C}^{\text{col}}_{:j} = \frac{\mathbf{C}_{:j}}{\sum_{i=1}^{m} C_{ij}}$$

Similarly, denoting rows of $\mathbf{C}$ by $\mathbf{C}_{i:}$, define row-normalized response matrix

$\mathbf{C}^{\text{row}}$ of size $m \times kn$ with rows $\mathbf{C}^{\text{row}}_{i:}$ by:

$$\mathbf{C}^{\text{row}}_{i:} = \frac{\mathbf{C}_{i:}}{\sum_{j=1}^{nk} C_{ij}}$$

At each iteration, we update the option weight vector $\mathbf{w}$ and user score vector $\mathbf{s}$ as follows until convergence:

$$\begin{aligned}\mathbf{w}^{(t)} &\leftarrow (\mathbf{C}^{\text{col}})^{\intercal}\mathbf{s}^{(t-1)} \\ \mathbf{s}^{(t)} &\leftarrow \mathbf{C}^{\text{row}}\mathbf{w}^{(t)}\end{aligned}$$

By concatenating above two update equations, we can update user scores between iterations directly by multiplying the two normalized response matrices into one update matrix $\mathbf{U}$ as in:

$$\mathbf{s}^{(t)} \leftarrow \underbrace{\mathbf{C}^{\text{row}}(\mathbf{C}^{\text{col}})^{\intercal}}_{\mathbf{U}}\mathbf{s}^{(t-1)} \tag{3.1}$$

These iterations are not yet very helpful. Indeed, we observe (see lemma 2) that the eigenvector corresponding to the largest eigenvalue of $\mathbf{U}$ is the all-ones vector $\mathbf{e}$, and this is the vector of user scores that AVGHITS converges to. However, it turns out that it is the eigenvector corresponding to the *second largest eigenvalue* of $\mathbf{U}$ that we seek. We will prove in section 3.4 the following theorem:

Theorem 1 (2nd eigenvector of AVGHITS recovers C1P)**.** *If* $\mathbf{C}$ *is a pre-P-matrix with a unique consecutive ones ordering of its rows, then the unique consecutive ones ordering of the rows of* $\mathbf{C}$ *is given by the ranking of the rows sorted by values in the eigenvector corresponding to the second largest eigenvalue of* $\mathbf{U}$*.*

We now describe our modification of the AVGHITS iterative loop to find the exact ordering of this second eigenvector very efficiently in linear time per iteration. Note that all nomenclature details used are listed together in appendix A.

3.2 Our algorithm HITSnDIFFS

Rather than updating the user scores iteratively, our variant HITSNDIFFS updates the *differences* between adjacent user scores by using a suitably modified update matrix, and results in the scores converging to the ordering according to the second eigenvector of $\mathbf{U}$. Furthermore, this modification only adds a linear overhead of computing the user score difference vectors and normalizing it in every iteration. Yet, as we will show later in section 3.4, when there is a unique solution to the consecutive ones ordering of the rows of the response matrix, then HITSNDIFFS will discover it correctly.

Consider response matrix $\mathbf{C}$ from Figure 3.1c corresponding to the bipartite graph from Figure 3.1a. We add a third partition of nodes to the left of the user nodes, which represents *differences* between user scores (Figure 3.1b): To go from the user scores (middle) to the user difference scores (left), we find differences between consecutive user scores.

Defining the left partition of nodes as $\mathbf{s}^{\text{diff}}$, we can rewrite as

$$\begin{aligned}\mathbf{s}_i^{\text{diff}} &= \mathbf{s}_{i+1} - \mathbf{s}_i \qquad i \in [m-1] \\ &= \mathbf{S}\mathbf{s}\end{aligned}$$

where $\mathbf{S} \in \mathbb{R}^{(m-1)\times m}$ is defined as:

$$\mathbf{S} = \begin{bmatrix} -1 & 1 & 0 & \cdots & 0 \\ 0 & -1 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -1 & 1 \end{bmatrix}$$

In the reverse direction, there are infinitely many vectors $\mathbf{s}$ that can be generated from a given $\mathbf{s}^{\text{diff}}$, all shifted by a constant. Since we only want a final ordering of users, we can WLOG set the first element of the vector $\mathbf{s}$ to be 0. The transformation then is

$$\mathbf{s} = \mathbf{T}\mathbf{s}^{\text{diff}}$$

where $\mathbf{T} \in \mathbb{R}^{m\times(m-1)}$ is the lower triangular matrix

$$\mathbf{T} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 1 & 0 & \cdots & 0 \\ 1 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 1 & 1 & \cdots & 1 \end{bmatrix}$$

We can now get a user difference score update rule of the form $\mathbf{s}^{\text{diff}} \leftarrow$

$\mathbf{U}^{\text{diff}}\mathbf{s}^{\text{diff}}$ for iteration (t) as follows:

$$\begin{aligned}\mathbf{s}^{\text{diff}(t)} &= \mathbf{S}\mathbf{s}^{(t)} \\ &= \mathbf{S}\mathbf{C}^{\text{row}}(\mathbf{C}^{\text{col}})^{\intercal}\mathbf{s}^{(t-1)} \\ &= \mathbf{S}\mathbf{C}^{\text{row}}(\mathbf{C}^{\text{col}})^{\intercal}\mathbf{T}\mathbf{s}^{\text{diff}(t-1)} \\ &= \mathbf{S}\mathbf{U}\mathbf{T}\mathbf{s}^{\text{diff}(t-1)} \qquad (3.2)\end{aligned}$$

In other words, $\mathbf{U}^{\text{diff}} := \mathbf{SUT}$ is a 'difference update' matrix that is used to update $\mathbf{s}^{\text{diff}}$ from one iteration to the next, and the update equations for HITSnDIFFS take on the form of:

$$\mathbf{s}^{\text{diff}(t)} \leftarrow \underbrace{\mathbf{S}\mathbf{C}^{\text{row}}(\mathbf{C}^{\text{col}})^{\intercal}\mathbf{T}}_{\mathbf{U}^{\text{diff}}}\,\mathbf{s}^{\text{diff}(t-1)} \qquad (3.3)$$

It follows that by running the mutual updates of $\mathbf{w}$, $\mathbf{s}$ and $\mathbf{s}^{\text{diff}}$ as described in algorithm 1, we will converge to the eigenvalue corresponding to the largest eigenvector of $\mathbf{U}^{\text{diff}}$. Our algorithm HITSnDIFFS that implements this is described in algorithm 1. We prove the following crucial property in section 3.4 (see eq. (3.6)):

Lemma 1 (Eigenvector correspondence)**.** $\mathbf{x}$ *is the eigenvector corresponding to the second largest eigenvalue of* $\mathbf{U}$ *iff* $\mathbf{y} = \mathbf{S}\mathbf{x}$ *is the eigenvector corresponding to the largest eigenvalue of* $\mathbf{U}^{\text{diff}}$*.*

In other words, converting the converged $\mathbf{y}$ back into a user score, we regain the ordering of the rows according to values in the second largest eigenvector of $\mathbf{U}$. This, along with Theorem 1, gives our main result in theorem 2 that HITSnDIFFS detailed in algorithm 1 reconstructs ideal consistent orderings. More details follow in section 3.4.

Note that algorithm 1 can be performed by using matrix-vector products only; we do not actually have to multiply any matrices directly, thus avoiding a possible quadratic time complexity. In particular, in the case when each of m users picks exactly one option out of k for each of n items, then each of the matrix-vector products in updating $\mathbf{s}^{\text{diff}(t)}$ from the values of $\mathbf{s}^{\text{diff}(t-1)}$ involves $O(mn)$ time. Also note the output of HITSnDIFFS provides us with additional information on the item options: we obtain both an ordering of all users (the $\mathbf{s}$ vector obtained at convergence), as well as an ordering of each option of each question (the $\mathbf{w}$ vector obtained at convergence).

Theorem 2. *If* $\mathbf{C}$ *is a pre-P-matrix with a unique consecutive ones ordering of its rows, then* HITSnDIFFS *reconstructs the consistent ordering of the users taking only time* $O(mn)$ *(linear in the number of nonzeros in* $\mathbf{U}$*) per iteration.*

Algorithm 1: HITSnDIFFS: A ranking algorithm that uses a user-item response matrix $\mathbf{C}$

Input: Response matrix $\mathbf{C}$, randomly initialized student scores $\mathbf{s}_0$
Output: Student scores $\mathbf{s}$, item option weights $\mathbf{w}$

```
1 $\mathbf{s} \leftarrow \mathbf{s}_0$            // initialize user scores
2 repeat
3     $\mathbf{w} \leftarrow (\mathbf{C}^{\text{col}})^\intercal \mathbf{s}$    // update item option weights
4     $\mathbf{s} \leftarrow \mathbf{C}^{\text{row}} \mathbf{w}$         // update user scores
5     $\mathbf{s}^{\text{diff}} \leftarrow \mathbf{S}\mathbf{s}$         // update user score differences
6     Normalize $\mathbf{s}^{\text{diff}}$ to be a unit vector
7     $\mathbf{s} \leftarrow \mathbf{T}\mathbf{s}^{\text{diff}}$    // update user scores from differences
8 until convergence or iteration limit
```

3.3 Complexity Comparison

We now outline the asymptotic complexity (in terms of the number of users) of the different methods described earlier - BL [4], ABH [1], DDKR1, DDKR2 [6], and GKM [12].

ABH. Given m users with their pairwise similarities recorded in a symmetric matrix, finding all orderings that reconstruct C1P has complexity $\mathcal{O}(m(T(m) + m \log m))$, where $T(s)$ is the time for an eigen-calculation on a matrix with s entries. Using Lanczos algorithm (as stated in [1]) to compute the second smallest eigenvector results in $T(m)$ being nearly linear, but having a dependence on the difference between the first two distinct eigenvalues. When $\mathbf{C}$ is not a pre-P matrix, ABH can return the ordering given by the second smallest eigenvector as the final user ordering as a heuristic. However, the matrix whose second smallest eigenvector is calculated is the Laplacian of the matrix $\mathbf{C}\mathbf{C}^\intercal$. Just like with HITSnDIFFS, using matrix-vector products instead of matrix-matrix products, the overall running time for returning a ranking of users as a heuristic from ABH has complexity $\mathcal{O}(mn)$.

GKM and DDKR1. The binary classification approaches GKM [12] and DDKR1 [6] make use of the first eigenvectors of different matrices. Both of these can be implemented to run in $\mathcal{O}(mn)$ time per iteration like our method using the factorization of the update matrix. However, this method is not proven to return C1P in the ideal case.

DDKR2. Given an assigment matrix $\mathbf{A}$ and a rating matrix $\mathbf{B}$, this method involves finding the first eigenvector of the Hadamard (entrywise) division $\mathbf{A}^\intercal\mathbf{A} \oslash \mathbf{B}^\intercal\mathbf{B}$. This requires explicit computation of the matrices

$\mathbf{A}^\intercal\mathbf{A}$ and $\mathbf{B}^\intercal\mathbf{B}$, and thus takes time $\mathcal{O}(m^2n)$.

EM. The complexity of EM is tricky to analyze - the open-source version used [5] is shown experimentally to be slower than other methods, but an optimized version of EM may have much faster performance.

BL. The original paper by Booth and Leuker (BL) [4] for reconstructing the C1P property however can work directly on the initial response matrix and runs in linear time. But this method is purely discrete and does not extend to given any meaningful heuristic in the non-ideal case.

3.4 HITSnDIFFS recognizes C1P

We show that if $\mathbf{C}$ is a pre-P matrix with a unique consecutive ones ordering, then the eigenvector corresponding to the second largest eigenvalue of $\mathbf{U}$ can be used to find this ordering. To show this, we use the two theorems we alluded to in section 3.2.

Note that if there are multiple connected components in the user-option bipartite graph, there is no way to get a total ordering on the users or questions, since we cannot compare between the different connected components; we can only get an ordering for each connected component *separately.* Thus, in the sequence, we assume a single connected component, and thus, multiplicity 1 of the largest eigenvalue of the update matrix $\mathbf{U}$.

We first state two Lemmas that will be useful in the proofs.

Lemma 2 (Constant row sums)**.** *If all rows of a non-negative square matrix $\boldsymbol{A}$ sum to a scalar b, then the largest eigenvalue of $\boldsymbol{A}$ is b with the corresponding eigenvector in the direction of* $\mathbf{e} = \mathbf{1}_n$.

We next restate the Perron-Frobenius Theorem [21]:

Lemma 3. *There exists a user who picked options for all questions that no other user picked (i.e. there exist 2 users not connected in the bipartite user-options graph), if and only if the largest eigenvalue of the matrix* $\mathbf{U}$ *has multiplicity > 1.*

Now we proceed to prove lemmas that will enable us to prove theorem 1 and theorem 2.

Lemma 4 ($\mathbf{U}$ is row-stochastic)**.** *Each row of* $\mathbf{U}$ *has sum* 1.

Proof. Define v_i as the sum of elements of the i^{th} row of $\mathbf{U}$: $v_i = \sum_{j=1}^{m} U_{ij}$. We want to show $v_i = 1 \quad \forall i \in \{1, 2, ..., m\}$.

Recall that $\mathbf{U} = \mathbf{C}^{\text{row}}(\mathbf{C}^{\text{col}})^{\intercal}$, so $\forall i$, the i^{th} row of $\mathbf{U}$, $\mathbf{U}_{i:}$ is generated from the corresponding row $\mathbf{C}^{\text{row}}_{i:}$ of $\mathbf{C}^{\text{row}}$ and all the rows $\mathbf{C}^{\text{col}}_{j:}$ of $\mathbf{C}^{\text{col}}$. Then,

$$\begin{aligned} v_i &= \sum_{j=1}^{m} \mathbf{C}^{\text{row}}_{i:} \cdot \mathbf{C}^{\text{col}}_{j:} \\ &= \mathbf{C}^{\text{row}}_{i:} \cdot \sum_{j=1}^{m} \mathbf{C}^{\text{col}}_{j:} \end{aligned}$$

By construction, $\sum_{j=1}^{m} \mathbf{C}^{\text{col}}_{j:}$ is the ones vector $\mathbf{e}$. So,

$$\begin{aligned} v_i &= \mathbf{C}^{\text{row}}_{i:} \cdot \sum_{j=1}^{m} \mathbf{C}^{\text{col}}_{j:} \\ &= \mathbf{C}^{\text{row}}_{i:} \cdot \mathbf{e} \\ &= 1. \end{aligned}$$

The last step follows by definition of row-normalized response matrix $\mathbf{C}^{\text{row}}$. □

Lemma 5 (1st eigenvector of avgHITS)**.** *If the largest eigenvalue of* $\mathbf{U}$ *has multiplicity 1, the fixed point of the* avgHITS *update rule is the ones vector* $\mathbf{e} = \mathbf{1}_m$

Proof. The avgHITS update rules above imply for any iteration $l > 0$,

$$\begin{aligned} \mathbf{s}^{(l)} &= \mathbf{U}\mathbf{s}^{(l-1)} \\ \implies \mathbf{s}^{(l)} &= \mathbf{U}^{l}\mathbf{s}^{(0)} \end{aligned}$$

Let the largest eigenvalue of $\mathbf{U}$ be λ_1 with corresponding eigenvector z_1. By the power rule, if λ_1 has multiplicity 1, $\mathbf{s}^{(l)}$ converges in the direction of z_1 as l goes to infinity (barring the very unlikely random initialization that is orthogonal to z_1).

Note that $\mathbf{U}$ is square and non-negative since $\mathbf{C}^{\text{row}}$ and $\mathbf{C}^{\text{col}}$ are both non-negative. By lemma 4, each row of $\mathbf{U}$ has sum 1, and we can then use lemma 2. This gives us the result, since the fixed point of avgHITS is in the direction of the eigenvector corresponding to the largest eigenvalue λ_1 of $\mathbf{U}$ (provided λ_1 has multiplicity 1) and lemma 2 implies $\mathbf{U}$ has largest eigenvalue 1 with corresponding eigenvector $\mathbf{e}$. □

We can now prove our main result in theorem 1 using the following two lemmas.

Lemma 6. *If the response matrix* $\mathbf{C}$ *is a P-matrix and each user picks the same number of options, then the update matrix* $\mathbf{U}$ *is an R-matrix.*

Proof. If $\mathbf{C}$ is a P-matrix, $\mathbf{C}\mathbf{C}^{\intercal}$ is an R-matrix [15]. By a similar argument, we show if $\mathbf{C}$ is a P-matrix, $\mathbf{U} = \mathbf{C}^{\text{row}}(\mathbf{C}^{\text{col}})^{\intercal}$ is an R-matrix. We need to show neither of

$$u_{ij} > u_{ik} \qquad \forall j < k < i \tag{3.4}$$

$$u_{ij} < u_{ik} \qquad \forall i < j < k \tag{3.5}$$

is true. Equation (3.4) implies that, out of the columns of $\mathbf{C}$ which have a 1 in the i^{th} row (and therefore nonzero entries in the i^{th} row of $\mathbf{C}^{\text{row}}$), strictly more have 1's in the j^{th} row than the k^{th} row. But since $j < k$, the column which has a 1 in the j^{th} row but not the k^{th} row leads to a violation of the consecutive ones property. Equation (3.5) leads to a similar contradiction.

Next, we also need to prove $\mathbf{U}$ is symmetric; for this, we need the property that each user picks the same number of options - say each user chooses n^{ans} options. Consider any $i, j \in \{1, 2, ..., m\}, i \neq j$. We need to show $u_{ij} = u_{ji}$.

$$u_{ij} = \sum_{h=1}^{nk} c_{ih}^{\text{row}} c_{jh}^{\text{col}}$$
$$u_{ji} = \sum_{l=1}^{nk} c_{jl}^{\text{row}} c_{il}^{\text{col}}$$

To contribute to the sum, the respective elements of $\mathbf{C}^{\text{row}}$ and $\mathbf{C}^{\text{col}}$ have to both be nonzero. When c_{ih}^{row} and c_{jh}^{col} are both nonzero, it implies c_{ih} and c_{jh} are 1. But by definition of $\mathbf{C}^{\text{row}}$, all nonzero entries are equal: $c_{ih}^{\text{row}} = \frac{1}{n^{\text{ans}}}$ because each user chooses n^{ans} options. And by definition of $\mathbf{C}^{\text{col}}$, $c_{ih}^{\text{col}} = c_{jh}^{\text{col}}$ for all nonzero c_{ih}^{col}, c_{jh}^{col}. So, $u_{ij} = u_{ji}$ for all $i \neq j$, and thus $\mathbf{U}$ is symmetric. □

The following lemma shows the desired property of the second eigenvector of $\mathbf{U}$ in the ideal case.

Lemma 7. *If $\mathbf{C}$ is a P-matrix and each user chooses equal number of options, the eigenvector corresponding to the second largest eigenvalue of $\mathbf{U}$ is monotonic.*

Proof. We will prove this in a method similar to [1]. Define the matrices $\mathbf{S} \in \mathbb{R}^{(m-1)\times m}$ and $\mathbf{T} \in \mathbb{R}^{m\times(m-1)}$ as in section 3.2.

Note that $\mathbf{TS} = (\mathbf{I}_m - \mathbf{e}\mathbf{e}_1^\intercal)$. Note that for any vector $\mathbf{v}$, $\mathbf{Sv} = (v_2 - v_1, v_3 - v_2, ..., v_r - v_{r-1})^\intercal$. Similarly, the i^{th} row of $\mathbf{SU}$ is just the difference between the $(i+1)^{\text{th}}$ and i^{th} rows of $\mathbf{U}$. However, as shown in lemma 4, each row of $\mathbf{U}$ sums to 1. This implies each row of $\mathbf{SU}$ sums to 0.

Let $\mathbf{x}$ be an eigenvector of $\mathbf{U}$ that is not in the direction of the all ones vector $\mathbf{e} = \mathbf{1}_m$, i.e. $\mathbf{x} \neq \alpha\mathbf{e}$. Then,

$$\begin{aligned}
\mathbf{U}\mathbf{x} &= \lambda\mathbf{x} \\
\mathbf{S}\mathbf{U}\mathbf{x} &= \lambda\mathbf{S}\mathbf{x} \\
\mathbf{S}\mathbf{U}(\mathbf{I}_m - \mathbf{e}\mathbf{e}_1^\intercal)\mathbf{x} &= \lambda\mathbf{S}\mathbf{x} \\
\mathbf{S}\mathbf{U}\mathbf{T}\mathbf{S}\mathbf{x} &= \lambda\mathbf{S}\mathbf{x} \\
\mathbf{U}^{\text{diff}}\mathbf{y} &= \lambda\mathbf{y}, \quad \text{where} \quad \mathbf{y} = \mathbf{S}\mathbf{x} \qquad (3.6)
\end{aligned}$$

The equivalence between the second and third lines above is because each row of $\mathbf{SU}$ sums to 0 as shown above, so $\mathbf{SUe} = \mathbf{0}$. So, any eigenvalue λ of $\mathbf{U}$ is also an eigenvalue of $\mathbf{U}^{\text{diff}}$, except for the eigenvalue of $\mathbf{U}$ corresponding to the $\mathbf{e}$ eigenvector. So, $\mathbf{U}^{\text{diff}}$ has exactly the same eigenvalues as $\mathbf{U}$, except the largest eigenvalue 1, and the eigenvectors of $\mathbf{U}^{\text{diff}}$ are the differences between the elements of the corresponding eigenvector of $\mathbf{U}$.

Using an argument identical to Theorem 3.2 in [1], we know the off-diagonal elements in $\mathbf{U}^{\text{diff}}$ are non-negative because $\mathbf{U}$ is an R-matrix.

Since off-diagonal elements of $\mathbf{U}^{\text{diff}}$ are non-negative, the matrix $\mathbf{U}^{\text{diff}} + t\mathbf{I}_{m-1}$ is non-negative for some $t \in \mathbb{R}$. We can now apply Perron-Frobenius Theorem - there exists a non-negative eigenvector of $\mathbf{U}^{\text{diff}}$ corresponding to the largest eigenvalue of $\mathbf{U}^{\text{diff}}$. But we know $\mathbf{U}^{\text{diff}}$ has exactly the same eigenvalues as $\mathbf{U}$, except the largest eigenvalue 1, and the eigenvectors of $\mathbf{U}^{\text{diff}}$ are the differences between the elements of the corresponding eigenvector of $\mathbf{U}$. Since the differences between the elements of the eigenvector corresponding to the second largest eigenvalue of $\mathbf{U}$ (largest eigenvalue of $\mathbf{U}^{\text{diff}}$) is non-negative, that eigenvector of $\mathbf{U}$ is monotonic. $\square$

The above two lemmas imply that if we start with a pre-P matrix $\mathbf{C}$, sorting the rows according to the second eigenvector ordering of the corresponding update matrix $\mathbf{U}$ gives a P-matrix, proving theorem 1. This in turn gives our main theorem 2.

Chapter 4

Synthetic and Real-World Data

We next describe the BSTS generative model for synthetic data that we repurpose from *Item Response Theory* (IR) for our problem and explain why it is particularly suited for our problem. We then provide details of real-world data sets we use. The BSTS model generates exactly one option for each item for each user and hence a response matrix with equal row sums. However, some of its variants and the real-world data we work with involve response matrices with unequal row sums. We describe how our method handles this generalization to unequal row sums in section 5.2.

The BSTS Data Generator from IRT. To test our method on the label choice problem, we propose a new synthetic data generator that combines models from the statistical field of Item Response Theory (IRT). IRT [2] is widely used to assess students, e.g. in the Scholastic Aptitude Test (SAT) [20] and Graduate Record Examinations (GRE) [17]. It models the probability of a student providing a correct response as a function of latent traits describing student ability and item factors characterizing the question.

We use an extension of a multiple choice response model due to Thissen and Steinberg [29] to derive a generative model of how users select item label options, which we describe next. In the widely studied two parameter logistic (2PL) model from IRT for binary responses, the *ability* of student i is captured by a single latent trait variable θ_i, and question j has a *difficulty* and a *discrimination factor*. The model specifies item characteristic curves that plot the probability of a student answering the item correctly as a function of her ability. The basic 2PL model has been extended to handle multiple ("polytomous") responses [3, 23].

We use a combination of these extensions proposed by Thissen and Steinberg (see the chapter by Thissen and Steinberg in [29]) for generating our simulated data. We call this the BSTS model combining the last names of all four contributors (Bock-Samejima-Thissen-Steinberg). The model postulates a latent "*don't-know*" choice numbered zero along with the given k options. Each option including the latent one has two parameters α_h and κ_h. The probability that a user of ability θ_i chooses the option h between 1 and k is given by the following response function:

$$\mathbb{P}(C_{ih} = 1|\theta_i, \alpha_h, \kappa_h) = \frac{e^{\alpha_h\theta_i+\kappa_h} + \frac{1}{k}e^{\alpha_0\theta_i+\kappa_0}}{\sum_{h=0}^{k} e^{\alpha_h\theta_i+\kappa_h}}.$$

Note that for an item labeling problem with k options per item, this model requires $2(k+1)$ parameters, including a pair for each option and a pair for the don't-know option. This is an appropriate generalization of the binary choice model for choosing among multiple options especially when there is an ordering among the distractors to the best label as reflected in Figure 4.1. Note that the BSTS model generalizes the widely studied Multinomial Logit (MNL) model in discrete choice theory by setting both α_0 and κ_0 to negative infinity thus ruling out the don't-know option. The MNL itself generalizes the Logistic model for binary choice which itself is widely used in data mining and neural network models.

We reiterate why the BSTS model is particularly appropriate for the crowd-sourced item labeling problem. In this model, note that as the ability of the chooser deteriorates, the choices become *randomized among all the options* rather than converge on one focal bad option. This is particularly representative of user choices among item labels when there is a clear appropriate label chosen by the high ability users and the other choices are somewhat ordered but eventually *non-skilled choosers pick among all choices randomly*.

Three real-world data sets. We use the NLP-RTE and NLP-TEMP datasets [27] for binary item labeling and the GAL dataset [26] for multiple choice item labeling. NLP-RTE consists of 164 users labeling 800 binary-choice items. Each item is a pair of sentences $\{A, B\}$ and users choose whether or not sentence A implies sentence B. Note that there are 8000 responses, and not all users label all items. NLP-TEMP is another binary labeling dataset with 76 users labeling 800 items, consisting of 4620 responses. Each item consists of a text paragraph and two events; users choose which event occurred first based on the text paragraph. The GAL (Get Another Label) dataset is a multiple choice dataset from Amazon Mechanical Turk

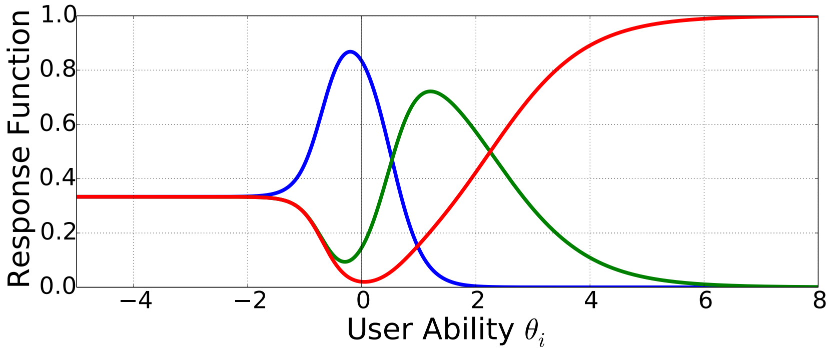


Figure 4.1: Item-characteristic curves under the BSTS model: X-axis represents the user ability, Y-axis the probability of choosing an option. The increasing order of appropriateness of the items is blue , green and red. Note that for users of low ability, the chance of choosing all of them converge to the same random guessing value $\frac{1}{3}$.

with 613 users labeling 1594 items which has 13021 unique responses. Each item is a website; users choose which of 5 categories each website belongs in. We see that these three datasets provide real-world examples of users labeling items into different categories based on their ability.

Chapter 5

Experiments and Results

Our experiments compare *accuracy* and *scalability* of the various methods. In particular, we will show that: 1) HITSnDIFFS produces user rankings and item labelings with *superior accuracy* compared to the various linear spectral methods; and 2) HITSnDIFFS is *competitive* with other state-of-the-art methods that are asymptotically slower.

5.1 Methods and their implementation

In addition to our method HITSnDIFFS, we implement DDKR1 and DDKR2 for binary item labeling [6], GKM for binary item labeling [12], EM for multi-choice item labeling [7], and the ABH heuristic for multi-choice item labeling [1].

DDKR1 and DDKR2 can simultaneously output a measure of user ability as well as the label for each item. These methods show theoretical error bounds for arbitrary user-item graphs. In the scenario where every user labels every item, both DDKR1 and DDKR2 produce the identical output; the performance of the two methods starts deviating as the number of responses decreases. The error bound for both the algorithms they propose is governed by the expansion of the graph. Recall that by constructing an assigment matrix $\mathbf{A}$ and a rating matrix $\mathbf{B}$, both the algorithms rely on finding eigenvectors of the symmetric matrices $\mathbf{A}^\intercal\mathbf{A}$ and $\mathbf{B}^\intercal\mathbf{B}$. The difference between the algorithms is that DDKR1 uses the ratio of the eigenvectors for computing the user ability ranking and the labels, while DDKR2 uses the eigenvectors of the ratio (i.e. the Hadamard or entrywise division) of the matrices.

Notice that GKM [12] can only output item labels. Like DDKR1, it can be run in linear time using the power method. Like DDKR1 and DDKR2, this method also involves calculating the first eigenvector of a symmetric matrix, and so can be run in linear time using the power method. It also shows certain theoretical guarantees against manipulation by adversarial or strategic raters.

Recall that the EM algorithm assumes that each user has a latent confusion matrix for labeling. The off-diagonal elements represent the probabilities that a user mislabels an arbitrary item from one class to another while the diagonal elements correspond to her accuracy in each class. The confusion matrices and true labels are jointly estimated by maximizing the likelihood of observed labels. Again, this method does not guarantee reconstruction of the C1P property in the ideal case.

ABH is the only prior method that can reconstruct the C1P condition (if present) as well as work on general datasets. It also provides a SpectralSort algorithm to find all possible orderings that reconstrcts C1P. Note that ABH originally returns only a student ability measure.

Similar to [6], we choose the weight of an item's option to be the sum of the user ability estimates of the users who chose that option, and the correct option chosen for an item is the one with the highest weight.

We also use this approach for generating the item labels for the HITSnDIFFS based on the user scores it outputs.

5.2 Implementation Issues

Unequal row sums. While proving these theoretical C1P results for our methods, we often make the assumption that each row of the matrix **C** has the same row sum, i.e. the same number of 1s in each row. This is WLOG, because given any arbitrary matrix **C**, we can add additional columns to the matrix **C**, where each additional column has exactly one 1 and $m-1$ zeros, till each row has the same number of 1s. Note that adding such columns doesn't affect the C1P property, since each additional column we add has only a single 1. In practice, the ordering of users might be affected by padding the matrix with columns in this way, so we only make this assumption of equal row sums for proving results related to the C1P property, and not while using this method as a heuristic on datasets in this section.

Entropy-based symmetry breaking. All methods for solving C1P suffer from a *symmetry breaking problem* of deciding between the order arrived at by the algorithm *or its inverse* (reversing the order of a P-matrix still leaves it as a P-matrix). This would not be an issue if we were only interested is the adjacencies in the ordering, but not the actual ranking. However in our applications, we require to know if a user performs best or worst, and similarly for the item options.

Our solution to this symmetry-breaking problem is motivated by the following observation: when one choice is the most appropriate label for an item, the higher quality users tend to converge on it as a majority answer while at the other end of the ordering the alternate options are chosen more randomly by the lower quality users. Thus, intuitively, the higher quality end in a user ordering can be distinguished by the lower entropy of choices picked by that end compared to the opposite end.

We encode this idea in a new heuristic that is very effective in practice: Given a final ranking of the users, we compute for every item, for the top and the bottom decile, the entropy of the fractions of item options that appear in the deciles, for each item. We pick the end that has lower entropy of options picked as the higher quality end. Since the other methods such as ABH, DDKR1, DDKR2, and GKM also are faced with this problem, in our implementation of these methods, we also added the "*decile entropy method*" described above in outputting the final order of user abilities and item option orderings. The idea of using the end with lower entropy is also supported by the BSTS generative model: the set of choices picked by users of low ability are uniformly random among the distractors, hence having high entropy compared to the single correct choice picked by the user with the highest ability.

5.3 Experiments on synthetic data sets

We compare the various methods by following metrics: 1) *user accuracy* (i.e. correlation of user ranking output by the methods with abilities $\boldsymbol{\theta}$); 2) *item labeling accuracy* (i.e. fraction of correctly labeled items); 3) item labeling accuracy *when users label only a fraction of items*; and 4) *execution times.*

We use the BSTS model from chapter 4 to generate either binary ($k = 2$) or multiple choice data ($k = 5$). We uniformly sample α_h and κ_h for $h \in \{0, \ldots, k\}$ in the range $[-0.2, 0.2]$ for each question, and θ in the range $[-100, 100]$ for each student, which ensures that incorrect item options are not easily discarded by users. We vary the number of users m and keep the number of items fixed at $n = 50$ and the probability that each student answers a question at $p = 0.7$. For each parameter setting, we average over 500 repetitions. Figures 5.1a to 5.1c and 5.3a show the results on binary choice data and Figures 5.2a to 5.2c on multiple-choice data.

1. User accuracy. Figures 5.1a and 5.2a compare the user accuracy over varying sizes of users. We calculate accuracy as the correlation between the user ability estimates returned by the methods with $\boldsymbol{\theta}$. Notice that GKM can only find the item labels and EM returns a confusion matrix for each user (but not a ranking of the users) so we do not report user accuracy for GKM and EM. Also recall that DDKR1 and DDKR2 only work for binary user data. We see that HITSnDIFFS shows better accuracy than the other methods.

2. Item labeling accuracy. Figures 5.1b and 5.2b compare the item accuracy over varying sizes of users. Here we calculate the fraction of correctly labeled items for each method. EM performs the best, but HITSnDIFFS outperforms other methods.

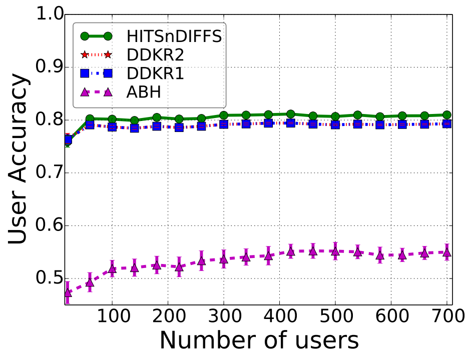
3. Varying fractions of labeled items. Figures 5.1c and 5.2c compare the item accuracy when the users label a varying fraction of items. Here, we fixed $m = 50$ users. HITSnDIFFS outperforms other methods as long as the probability of users labeling items is not very low. Notably, HITSnDIFFS is far more robust than ABH, whose performance falls drastically as the probability of users labeling items decreases.

4. Execution time. Figure 5.3a shows the execution times of the various methods as the number of users m grows. HITSnDIFFS, DDKR1, ABH and GKM show linear growth in m.

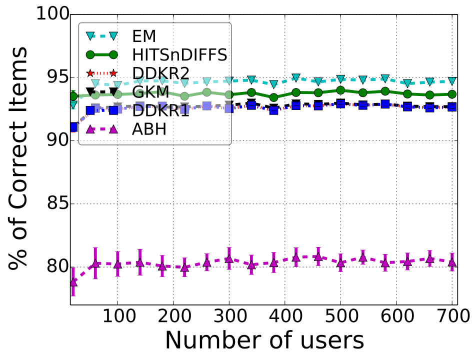
Take-aways. In summary, HITSnDIFFS outperforms the other method that can handle multiple choice data ABH and reconstruct C1P in terms of accuracy, as well as is far more robust as the probability of users labeling items decreases. It also outperforms the other linear methods in accuracy of user correlation for binary data (DDKR1 and GKM), and is a highly scalable method for multiple-choice questions.

5.4 Results on real data sets

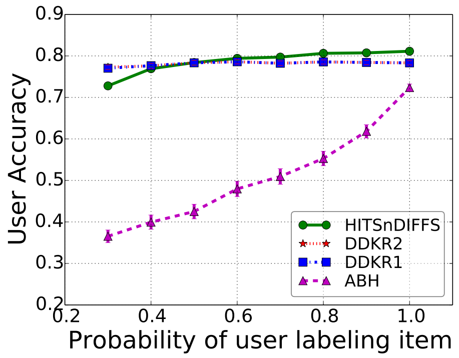
Figure 5.4 compares the item labeling accuracies on the 3 real datasets described in chapter 4. For the binary datasets NLP-RTE and NLP-TEMP, HITSnDIFFS outperforms both GKM, DDKR1 and ABH, while being competitive with DDKR2 and EM. For the multiple choice dataset GAL, we see identical performance from HITSnDIFFS and ABH. This is to be expected for GAL (and other sparse datasets) which are close to being pre-P matrices (with pre-P inputs, we know the user rankings from ABH and HITSnDIFFS are identical; with slight perturbations, the user rankings



(a) Correlation user score with θ vs. number of users (binary)



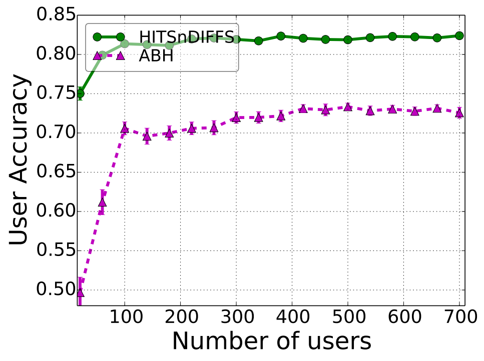
(b) Percentage of correct items vs. number of users (binary)



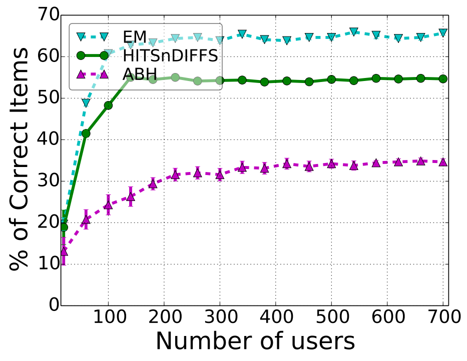
(c) Percentage of correct items vs. probability for labeling (binary)

Figure 5.1: Experiments on binary synthetic data

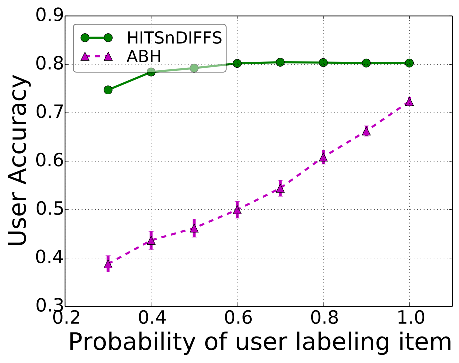
might differ but still result in the same label outputs for each item).



(a) Correlation of user score with θ vs. number of users (multi-choice)

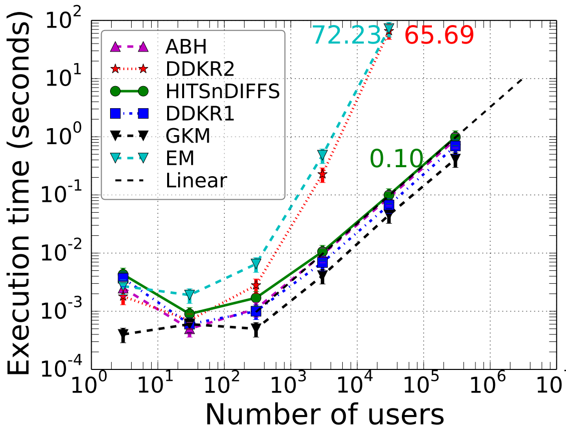


(b) Percentage of correct items vs. number of users (multi-choice)

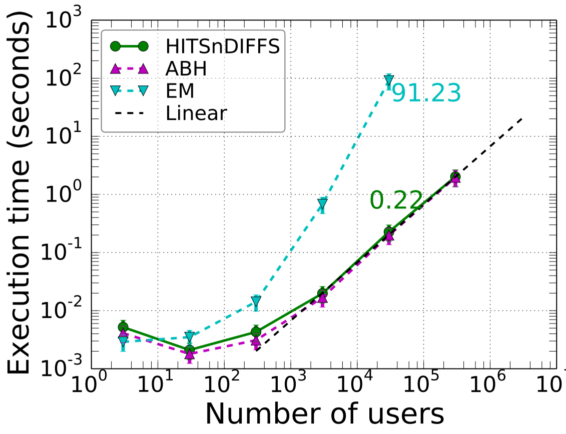


(c) Percentage of correct items vs. probability of labeling (multi-choice)

Figure 5.2: Experiments on multi-choice synthetic data

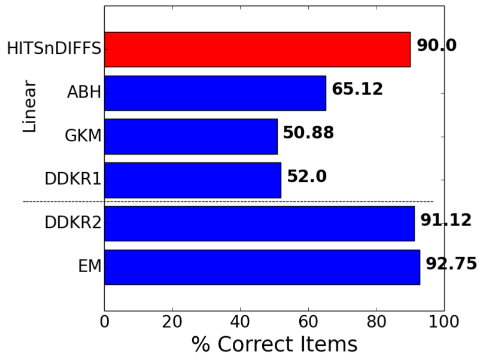


(a) Execution time vs. number of users (binary)

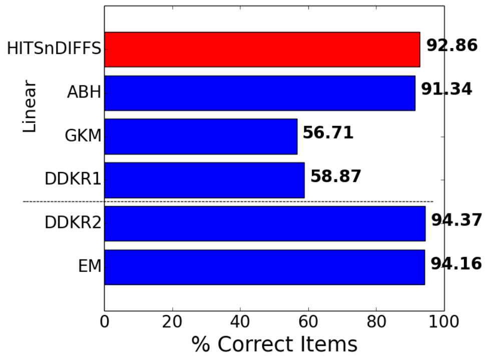


(b) Execution time vs. number of users (multi-choice)

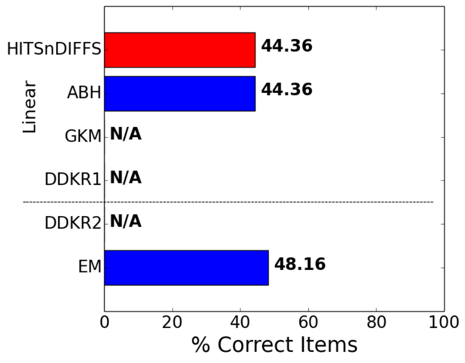
Figure 5.3: Time complexity comparison on simulated data. The dashed line showing linear growth confirms the linear performance of HITSnDIFFS



(a) NLP-RTE dataset



(b) NLP-TEMP dataset



(c) GAL dataset

Figure 5.4: Percentage of correctly labeled items on real datasets.

Chapter 6

Future work

The HITSnDIFFS algorithm was shown to handle the problem of user ranking and item labeling, and is competitive in accuracy with asymptotically slower algorithms on synthetic and real-life data. In addition to the methods compared against (ABH, DDKR1, DDKR2, GKM, EM) in this work, the newer work mentioned in Section 1.2, as well as the maximum likelihood estimator for the BSTS generative model can be compared against. Research is also ongoing to better explain the superior performance of HITSnDIFFS to ABH, since these are the two spectral methods that are both used for item-labeling, as well as satisfy C1P. Also, the current experiments in Chapter 5 on real data only compare the item-labeling accuracy. The performance of the various methods on real datasets on user ranking accuracy is also a future direction, since our HITSnDIFFS method is intrinsically designed for finding a user ranking.

Chapter 7

Conclusions

We have proposed a variation of the widely studied HITS algorithm with surprising theoretical and practical properties for this choice problem. On the theoretical side, we showed that 1) C1P of the response matrix models consistent solutions for the problem; 2) our method reconstructs the correct user rankings in the consistent case; 3) does so in linear time; 4) can handle more general cases (in contrast to other linear discrete algorithms); 5) on the practical side, can handle the problem of user ranking and item labeling,s and is competitive in accuracy with asymptotically slower algorithms on synthetic and real-life data.

We believe this method thus contributes a principled and practical solution to our problem that can reconstruct the ideal order if it is unique, scales well, and provides highly accurate data sets on real data sets that are not consistent.

Bibliography

[1] J. E. Atkins, E. G. Boman, and B. Hendrickson. A spectral algorithm for seriation and the consecutive ones problem. *SIAM Journal on Computing*, 28(1):297–310, 1998.

[2] F. K. Baker and S.-H. Kim. *Item Response Theory: Parameter Estimation techniques (2nd Ed).* Marcel Dekker Inc., 2004.

[3] R. D. Bock. Estimating item parameters and latent ability when responses are scored in two or more nominal categories. *Psychometrika*, 37(1):29–51, 1972.

[4] K. S. Booth and G. S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using pq-tree algorithms. *Journal of Computer and System Sciences*, 13(3):335–379, 1976.

[5] Dallascard. Implementation of the estimator for combining noisy observations from dawid and skene (1979). *Github.*

[6] N. Dalvi, A. Dasgupta, R. Kumar, and V. Rastogi. Aggregating crowdsourced binary ratings. In *WWW*, pages 285–294. ACM, 2013.

[7] A. P. Dawid and A. M. Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *Applied statistics*, pages 20–28, 1979.

[8] T. Finin, W. Murnane, A. Karandikar, N. Keller, J. Martineau, and M. Dredze. Annotating named entities in twitter data with crowdsourcing. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 80–88. ACL, 2010.

[9] M. J. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin. Crowddb: answering queries with crowdsourcing. In *SIGMOD*, pages 61–72. ACM, 2011.

[10] C. Gao, Y. Lu, and D. Zhou. Exact exponent in optimal rates for crowdsourcing. *International Conference on Machine Learning (ICML)*, 2016.

[11] C. Gao and D. Zhou. Minimax optimal convergence rates for estimating ground truth from crowdsourced labels. *arXiv preprint arXiv:1310.5764*, 2013.

[12] A. Ghosh, S. Kale, and P. McAfee. Who moderates the moderators?: crowdsourcing abuse detection in user-generated content. In *EC*, pages 167–176. ACM, 2011.

[13] D. Karger, S. Oh, and D. Shah. Budget-optimal crowdsourcing using low-rank matrix approximations. *Annual Allerton Conference on Communication, Control, and Computing*, 2011.

[14] D. Karger, S. Oh, and D. Shah. Iterative learning for reliable crowdsourcing systems. *Advances in neural information processing systems*, 2011.

[15] D. Kendall. Incidence matrices, interval graphs and seriation in archeology. *Pacific Journal of mathematics*, 28(3):565–570, 1969.

[16] A. Khetan and S. Oh. Reliable crowdsourcing under the generalized dawid-skene model. *arXiv preprint arXiv:1602.03481*, 2016.

[17] N. M. Kingston and N. J. Dorans. The feasibility of using item response theory as a psychometric model for the gre aptitude test. *ETS Research Report Series*, 1982(1), 1982.

[18] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *JACM*, 46(5):604–632, 1999.

[19] Q. Liu, J. Peng, and A. T. Ihler. Variational inference for crowdsourcing. *Advances in Neural Information Processing Systems, pages 692–700*, 2012.

[20] F. M. Lord, M. R. Novick, and A. Birnbaum. Statistical theories of mental test scores. 1968.

[21] C. D. Meyer, editor. *Matrix Analysis and Applied Linear Algebra.* Society for Industrial and Applied Mathematics, 2000.

[22] Y. Saad. Numerical methods for large eigenvalue problems. In *Numerical Methods for Large Eigenvalue Problems*, pages 91–94, 2003.

[23] F. Samejima. Evaluation of mathematical models for ordered polychotomous responses. *Behaviormetrika*, 23(1):17–35, 1996.

[24] N. B. Shah, S. Balakrishnan, and M. J. Wainwright. A permutation-based model for crowd labeling: Optimal estimation and robustness. *arXiv preprint arXiv:1606.09632*, 2016.

[25] N. B. Shah and D. Zhou. Double or nothing: Multiplicative incentive mechanisms for crowdsourcing. In *NIPS*, pages 1–9. 2015.

[26] V. S. Sheng, F. Provost, and P. G. Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *KDD*, pages 614–622. ACM, 2008.

[27] R. Snow, B. O'Connor, D. Jurafsky, and A. Y. Ng. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *EMNLP*, pages 254–263. ACL, 2008.

[28] A. Tarasov, S. J. Delany, and C. Cullen. Using crowdsourcing for labelling emotional speech assets. *W3C EmotionML Workshop*, 2010.

[29] W. van der Linden and R. Hambleton. *Item response theory: Brief history, common models, and extensions*, pages 1–28. Springer, 1997.

[30] P. Welinder and P. Perona. Online crowdsourcing: rating annotators and obtaining cost-effective labels. In *CVPRW*, pages 25–32. IEEE, 2010.

[31] J. H. Wilkinson. The algebraic eigenvalue problem. *Oxford Science, Oxford, England*, 1965.

[32] Y. Zhang, X. Chen, D. Zhou, and M. I. Jordan. Spectral methods meet em: A provably optimal algorithm for crowdsourcing. *Journal of Machine Learning Research*, 2014.

[33] Y. Zheng, G. Li, Y. Li, C. Shan, and R. Cheng. Truth inference in crowdsourcing: Is the problem solved? *PVLDB*, 10(5):541–552, 2017.

[34] D. Zhou, S. Basu, Y. Mao, and J. C. Platt. Learning from the wisdom of crowds by minimax entropy. In *NIPS*, pages 2195–2203, 2012.

[35] D. Zhou, Q. Liu, J. Platt, and C. Meek. Aggregating ordinal labels from crowds by minimax conditional entropy. In *ICML*, pages 262–270, 2014.

[36] D. Zhou, Q. Liu, J. C. Platt, C. Meek, and N. B. Shah. Regularized minimax conditional entropy for crowdsourcing. *arXiv preprint arXiv:1503.07240*, 2015.

Appendix A

Nomenclature

u_i	user i
t_j	item j
c_{jh}	choice h for item j
m	number of users
n	number of items
k	number of choices (or labels) for each item
θ_i	ability of user u_i
α_{jh}	quality of choice h of item t_j
$\mathbf{s}$	user score vector $(m \times 1)$ where s_i denotes score of user i
$\mathbf{s}^{\text{diff}}$	user difference vector $((m-1) \times 1)$ with $\mathbf{s}_i^{\text{diff}} = \mathbf{s}_{i+1} - \mathbf{s}_i, i \in [m-1]$
$\mathbf{C}$	response matrix $(m \times kn)$ obtained by flattening the response tensor
$\mathbf{C}^{\text{col}}$	column-normalized response matrix $(m \times kn)$
$\mathbf{C}^{\text{row}}$	row-normalized response matrix $(m \times kn)$
$\mathbf{w}$	option weight vector $(kn \times 1)$
$\mathbf{U}$	Update matrix $(m \times m)$ used by AVGHITS to update user scores
$\mathbf{U}^{\text{diff}}$	Difference Update matrix $((m-1) \times (m-1))$ used by HITSNDIFFS to update user score differences $\mathbf{s}^{\text{diff}}$