Link Prediction Under Imperfect Detection: Collaborative Filtering for Ecological Networks

Xiao Fu, Eugene Seo, Justin Clarke, and Rebecca A. Hutchinson

Abstract— *Matrix completion* based *collaborative filtering* is considered scalable and effective for online service link prediction (e.g., movie recommendation) but does not meet the challenges of link prediction in ecological networks. A unique challenge of ecological networks is that the observed data are subject to systematic *imperfect detection*, due to the difficulty of accurate field sampling. In this work, we propose a new framework customized for ecological bipartite network link prediction. Our approach starts with incorporating the Poisson N-mixture model, a widely used framework in statistical ecology for modeling imperfect detection of a single species in field sampling. Despite its extensive use for single species analysis, this model has never been considered for link prediction between different species, perhaps because of the complex nature of both link prediction and N-mixture model inference. By judiciously combining the Poisson N-mixture model with a probabilistic nonnegative matrix factorization (NMF) model in latent space, we propose an intuitive statistical model for the problem of interest. We also offer a scalable and convergence-guaranteed optimization algorithm to handle the associated maximum likelihood identification problem. Experimental results on synthetic data and two real-world ecological networks data are employed to validate our proposed approach.

1 Introduction

Link prediction [1] aims at inferring unseen connections between entities in a complex network, which lies at the heart of a large variety of data mining problems. For example, social network analytics involves many link prediction problems (e.g., community detection) [2]. In online service recommender systems (e.g., streaming services and online shopping) [3]–[5], link prediction plays an essential role in learning user preferences.

Among many link prediction techniques, low-rank matrix completion (MC)-based collaborative filtering (CF) [6] is one of the most popular approaches—possibly because of its simplicity and effectiveness, as well as its elegance in mathematics [7], [8]. Taking the movie recommendation problem as an example, the idea of MC-based CF is as follows: If many users have similar preferences and a lot of movies have similar traits, then the complete (but partially observed) user-movie matrix (whose entries are users' ratings for the movies) should exhibit correlations across rows and columns—and thus be an approximately low-rank matrix. Therefore, imputing those unobserved ratings should not be arbitrary—it should be done under the constraint that the imputed movie rating matrix is low-rank.

Existing MC-based CF approaches have been quite successful in the aforementioned online service-related domains, but note that there is a clear distinction in these applications between missing and non-missing values. For example, an online movie streaming system knows perfectly which ratings are recorded and which are missing. Our work is motivated by a domain in which this distinction is *not* clear: species interaction networks. Data on these networks are often collected through field observations, which may fail to record some interactions due to limited time for sampling or poor observation conditions. For example, hundreds of species of pollinators and plants exist in the Cascade mountains, Oregon, United States [9]. Ecologists are eager to know the interaction patterns between pollinators and plants, in order to predict the effects of species extinctions or invasions and to protect ecosystem services (e.g., pollination) provided by pertinent species [10], [11] (see Fig. 1 for illustration).

At first glance, it appears that pollinator-plant link prediction is quite similar to user-movie link prediction, and that approaches proposed for the latter should also be applicable for the former. Indeed, such approaches have been attempted for these data with mixed results; CF techniques sometimes failed to outperform a baseline of simply predicting the most popular plants (i.e., items) [12]. The shortcomings of traditional CF methods are due to key differences among the application domains. One particular challenge is that the data recorded for pollination networks are very different from those recorded in online services. For example, the interaction counts between pollinators and plants in the Cascade mountains were recorded by student researchers at montane meadows in the HJ Andrews Experimental Forest in Blue River, Oregon, United States; see details in [9]. Even for meticulous observers (or even if the student observers were replaced by cameras), there is a significant chance that many interactions were not recorded for a number of reasons—since factors such as the season, location, and weather at the time of observation all play roles in the recorded number of observed interactions. This means that every entry in the pollinator-plant interaction matrix is subject to imperfect detection. For example, "2 interactions recorded" may be the result of "8 interactions really happened" plus the effect of missed detections. Thus, the non-zero recorded counts in the interaction matrix are likely biased low. This effect applies to the recorded zeros as well; the zeros are a mixture of true zeros (no interaction) along with unseen interactions that actually occurred (missed detections). Algorithms proposed for user-movie link prediction assume that while the data may be sparse (as are pollination data), the observed ratings are essen-

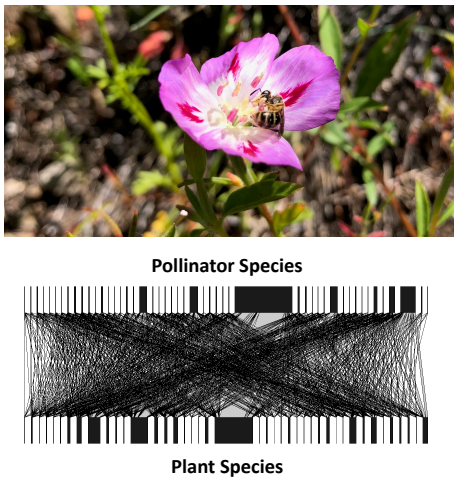


Fig. 1. Dataset that motivates this work [9]. Top: *Halictus* bee species visiting Clarkia amoena flower. Bottom: Subset of the plant-pollinator network recorded by student observers, which is subject to systematic under-counting.

Figure 1: Top: *Halictus* bee species visiting Clarkia amoena flower. Bottom: Subset of the plant-pollinator network observed by REU students, which is subject to systematic under-counting.

Unlike noise-free observations, in ecological networks, pervasive missed detections violate this assumption and demand new link prediction approaches.

The issue of imperfect detection has been recognized as a universal challenge in ecology [13], since cryptic and secretive species are frequently hard to observe. A family of statistical models have been proposed to handle this problem, e.g., occupancy models for binary observations [14] and the N-mixture model for counts [15]. The idea is to impose a probabilistic model that links the true species occurrence or abundance (ground-truth) and the observations (e.g., by modeling the observation as a binomial selection of the ground truth). Combined with a latent probabilistic model for the events of interest (e.g., a Poisson model), this leads to an intuitively pleasing two-layer graphical model for generating the observed data. This idea makes a lot of sense, and has been widely used in applications such as animal abundance estimation [15]. Nevertheless, because of the intrinsic computational difficulties, it has been unclear whether this family of models could be extended to link prediction in ecological networks.

Contributions In this work, we offer a computational framework for link prediction under imperfect detection. Our work is motivated by the pollination network analysis problem mentioned before, but other ecological networks share similar properties and challenges (e.g., host-parasite networks, food webs). Inspired by the efficacy of statistical ecology tools for modeling imperfect detection, we propose a collaborative filtering framework that integrates the Poisson N-mixture model and low-rank nonnegative matrix factorization (NMF)—so that the challenging ecological network link prediction problem can be effectively modeled and approached. Our detailed contributions include:

1) **Statistical Model for Ecological Link Prediction** We propose a graphical model for the generative process of observing interactions between two species in an ecosystem. Our idea is motivated by the Poisson N-mixture model [15] that has been used for single-species observations. We impose a latent Poisson-nonnegative matrix factorization (NMF) model for generating the real counts of observations, and a Binomial-linear regression model to model the imperfect detection effect. This way, both interaction preferences (modeled by latent NMF) and factors that affect the observation (modeled by features of the regression model) can be considered under a unified framework.
2) **Effective Optimization Algorithm** We propose an effective algorithm to handle the inference problem associated with our model. The maximum likelihood (ML) estimator under the generative model poses a very challenging optimization problem, due to the Poisson distribution in the latent space, parameterized by NMF—which is NP-hard to compute. Nevertheless, we propose a *block coordinate descent* (BCD)-based algorithm [16] to handle the ML estimator, and show that the algorithm is convergent. In addition, we judiciously design the updates so that the algorithm only consists of algebraically simple operations.
3) **Evaluations on Ecological Network Data** We test the proposed approach on synthetic data and two real-world ecological networks (plant-pollinator interactions and host-parasite interactions) prediction problem. The pollination and host-parasite data are publicly available; see [9], [17]. Open-sourced code for our method and the baselines are also made available through GitHub at https://github.com/Hutchinson-Lab/Poisson-N-mixture.

Notation Throughout the paper, we will use boldface capital letters such as $\boldsymbol{X}$ to denote matrices. We use the Matlab notation $\boldsymbol{X}(i,j)$ to denote the entry of $\boldsymbol{X}$ at the ith row and jth column. $\boldsymbol{X}(i,:)$ and $\boldsymbol{X}(:,j)$ denote the ith row and the jth column of $\boldsymbol{X}$, respectively. $\boldsymbol{X}^\top$, $\boldsymbol{X}^{-1}$, and $\boldsymbol{X}^\dagger$ denote the transpose, inverse, and pseudo-inverse of $\boldsymbol{X}$, respectively. Boldface lowercase letters such as $\boldsymbol{x}$ are used to denote vectors. The inequalities $\boldsymbol{X} \geq \boldsymbol{0}$ and $\boldsymbol{x} \geq \boldsymbol{0}$ mean that the entries of the matrix and vector are all nonnegative.

2 Motivation and Background

As we have mentioned briefly in the introduction, our problem is motivated by studying pollination networks in the Cascade mountains of Oregon, United States [9]. In this study, student researchers recorded all interactions they observed between plant and pollinator species. The goal of this study, from an ecological point of view, is to infer the structure and dynamics of the pollination network, in order to understand how the network will respond to perturbations like species extinctions and invasions. To achieve this goal, one must first infer the true pattern of connections between different species of pollinators and plants, despite the incomplete (and highly biased) data provided by field sampling. To this end, we propose an approach that combines ideas from both collaborative filtering and statistical ecology. Below, we review key concepts from these fields.

2.1 Matrix Completion-Based Collaborative Filtering

One of the key techniques that enables movie recommendation is the so-called collaborative filtering (CF) [18]. In

CF, we are given a movie rating matrix $\boldsymbol{X}$ whose rows and columns represent users and movies, respectively—$\boldsymbol{X}(i,j)$ is the rating of movie j by user i. Only a small portion of $\boldsymbol{X}(i,j)$'s are observed, indexed by $(i,j) \in \boldsymbol{\Omega}$. The task of CF is to impute or predict the missing entries, which is essentially link prediction for bipartite graphs. One of the popular formulations is

$$\underset{\boldsymbol{U},\boldsymbol{V}}{\text{minimize}} \sum_{(i,j)\in\boldsymbol{\Omega}} \Big(\boldsymbol{X}(i,j) - \boldsymbol{u}_i^\top \boldsymbol{v}_j\Big)^2 \tag{1}$$

where $\boldsymbol{u}_i$ and $\boldsymbol{v}_j$ stand for the ith row and jth column of $\boldsymbol{U} \in \mathbb{R}^{I\times F}$ and $\boldsymbol{V} \in \mathbb{R}^{J\times F}$, respectively, and $\boldsymbol{U}$ and $\boldsymbol{V}$ collect all the latent representations (embeddings) of the users and movies, respectively. The rationale behind the above formulation is to model the rating of a movie by a user using the 'correlation' between the user embedding and movie embedding. In addition, one can view the formulation in (1) from a low-rank matrix factorization viewpoint (with many missing values)—since if many users have similar tastes and many movies have similar traits, $\boldsymbol{X}$ is expected to be a low-rank matrix. Hence, the collaborative filtering problem boils down to a *low-rank matrix completion* (MC) problem.

The formulation in Eq. (1) is more suitable for continuous data. When $\boldsymbol{X}(i,j)$ denotes a counted number, Poisson priors are commonly used, which assumes the following observation model [19], [20]:

$$\boldsymbol{X}(i,j) \sim \mathsf{Poisson}(\boldsymbol{u}_i^\top \boldsymbol{v}_j),\ \boldsymbol{u}_i \geq \boldsymbol{0},\ \boldsymbol{v}_j \geq \boldsymbol{0},\ \forall i,j \tag{2}$$

Note that the nonnegativity constraints on $\boldsymbol{u}_i$ and $\boldsymbol{v}_j$ are needed to ensure that the Poisson parameter $\lambda_{ij} = \boldsymbol{u}_i^\top \boldsymbol{v}_j$ characterizing the expected number of interactions within a certain time interval is nonnegative. The corresponding maximum likelihood (ML) estimator becomes

$$\underset{\boldsymbol{U},\boldsymbol{V}}{\text{minimize}} \sum_{i,j} \Big[\boldsymbol{u}_i^\top \boldsymbol{v}_j - \boldsymbol{X}(i,j)\log(\boldsymbol{u}_i^\top \boldsymbol{v}_j)\Big] \tag{3a}$$

$$\text{subject to } \boldsymbol{U} \geq \boldsymbol{0},\ \boldsymbol{V} \geq \boldsymbol{0}. \tag{3b}$$

The problem above is essentially the same as the Kullback-Leibler (KL)-divergence based NMF, which is well-studied in the literature; see [19], [20].

Note that the Poisson modeling in (3) has several appealing features. Most notably, it naturally models missing elements in $\boldsymbol{X}$, since $\boldsymbol{X}(i,j) = 0$ is allowed to happen with positive probability under a Poisson distribution. This way, the model interprets two major reasons for having missing links (namely, interactions do not exist and interactions were not observed) in a ecological network using a simple and unified way. The Poisson NMF model is preferred also because many ecological data are recorded as counts of interactions or encounters—which are both natural numbers rather than continuous values. This makes the formulation in (3) more appealing for link prediction in ecosystems. Nevertheless, the Poisson NMF model does not consider the imperfect detection effect, and thus naively applying (3) does not account for systematically under-counted data. In particular, the mixture of both true and false zeros leads to over-dispersion (zero-inflation) of the Poisson distribution if imperfect detection is not incorporated.

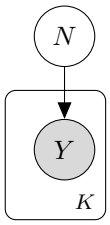


Fig. 2. Graphical model of a Poisson N-mixture model when K observations are made for a single spices.

2.2 Poisson N-Mixture Model

In statistical ecology, the Poisson N-mixture model is applied widely to handle the imperfect detection effect in field sampling—especially when observing animals. The model is simple and intuitive: the true abundance of an animal is modeled as

$$N \sim \mathsf{Poisson}(\lambda).$$

The observed count is modeled as a binomial selection of N, i.e.,

$$Y \sim \mathsf{Binominal}(N,p)$$

where p is the detection probability; see the illustration in Fig. 2. This model admits inference on both the true and observed counts by explicitly accounting for the observation process. Both λ and p can be linked to features that affect true abundance (e.g., habitat characteristics) and detection probability (e.g., weather conditions), respectively. However, most applications of this model have been concerned with estimating p and λ for *single species* [15], [21], and this model has not been applied to species interaction networks. The standard N-mixture model is already computationally hard [21], so incorporating it into link prediction problems is quite challenging.

3 Proposed Approach

Our approach combines the strengths of the Poisson N-mixture model for estimating species abundance under imperfect detection and low-rank CF models for link prediction. We propose the following generative model:

$$\lambda_{ij} = \boldsymbol{u}_i^\top \boldsymbol{v}_j,\ \boldsymbol{u}_i \geq \boldsymbol{0},\ \boldsymbol{v}_j \geq \boldsymbol{0}, \tag{4a}$$

$$N_{ij} \sim \mathsf{Poisson}(\lambda_{ij}) \tag{4b}$$

$$p_{ij} = \boldsymbol{\alpha}^\top \boldsymbol{z}^{(ij)}, \quad 0 \leq p_{ij} \leq 1 \tag{4c}$$

$$y_{ij} \sim \mathsf{Binomial}(N_{ij}, p_{ij}) \tag{4d}$$

We model λ_{ij}, i.e., the Poisson parameter for the (i,j)th interaction. A bigger λ_{ij} means that the interaction occurs more frequently on average. We assume that the interaction between pollinator i and plant j is more frequent if their embeddings are more 'correlated'. Note that we constrain $\boldsymbol{U}$ and $\boldsymbol{V}$ to be elementwise nonnegative, because λ_{ij} has to be nonnegative.

In the observation model, we let the detection probability of the (i,j)th interaction be $p_{ij} = \boldsymbol{\alpha}^\top \boldsymbol{z}^{(ij)}$, where $\boldsymbol{z}^{(ij)} \in \mathbb{R}^R$ is a interaction-specific feature vector. This vector is defined over a feature space that affects the detection probability for a particular pair of (i,j)—e.g., abundance of plant j, size of pollinator i, and the time of observation can all be features collected by $\boldsymbol{z}^{(ij)}$. This way, side information

that we may have in the data generating process can be seamlessly incorporated into the model. Note that p_{ij} is a detection probability, and thus naturally lies in between 0 and 1.

4 Optimization Algorithm

Following the generative model in (4), we are interested in learning $\boldsymbol{U}$, $\boldsymbol{V}$, and $\boldsymbol{\alpha}$. Let us denote the probability mass function of Y_{ij} by

$$\Pr(Y_{ij} = y_{ij}; \lambda_{ij}, p_{ij}).$$

By the assumed generative model and the law of total probability, the log-likelihood function of the parameters of interest is as follows:

$$\log\left(\prod_{ij}\sum_{n=y_{ij}}^{\infty}\mathsf{Pr}(N_{ij}=n;\lambda_{ij})\mathsf{Pr}(y_{ij}|N_{ij}=n;p_{ij})\right)$$
$$=\sum_{ij}\log\left(\sum_{n=y_{ij}}^{\infty}\left(\frac{\lambda_{ij}^n e^{-\lambda_{ij}}}{n!}\frac{n!}{y_{ij}!(n-y_{ij})!}p_{ij}^{y_{ij}}(1-p_{ij})^{n-y_{ij}}\right)\right)$$

The above formulation has an infinite sum, which appears problematic at first glance: If one directly applies maximum likelihood estimation using the above, then an infinite number of terms are involved in the estimator, which might be very hard to handle.

To proceed, we invoke the following lemma [21]:

Lemma 1 *The following equality holds:*

$$\sum_{n=y_{ij}}^{\infty}\frac{\lambda_{ij}^n e^{-\lambda_{ij}}}{n!}\frac{n!}{y_{ij}!(n-y_{ij})!}p^{y_{ij}}(1-p)^{n-y_{ij}}=\frac{(p\lambda_{ij})^{y_{ij}}e^{-\lambda_{ij}p}}{y_{ij}!}.$$

Lemma 1 is a classic result that is employed in many fields [21]—to be self-containing, we present a short proof in Appendix A. Nonetheless, the result is very useful in our context, since it elegantly converts a quite complicated ML estimator to one that is much easier to maneuverer for subsequent steps.

Applying Lemma 1, we have the following simplified log-likelihood:

$$\begin{aligned}&\sum_{ij}\log\left(\frac{(p_{ij}\lambda_{ij})^{y_{ij}}\,e^{-\lambda_{ij}p_{ij}}}{y_{ij}!}\right)\\&=\sum_{ij}\left(y_{ij}\log p_{ij}+y_{ij}\log\lambda_{ij}-\lambda_{ij}p_{ij}-\log y_{ij}!\right).\end{aligned}\tag{5}$$

Hence, under our model, we aim to optimize the following:

$$\begin{aligned}\underset{\boldsymbol{U},\boldsymbol{V},\boldsymbol{\alpha}}{\text{minimize}}\ &-\sum_{ij}\Big[y_{ij}\log\left(\boldsymbol{\alpha}^\top\boldsymbol{z}^{(ij)}\right)+y_{ij}\log\boldsymbol{u}_i^\top\boldsymbol{v}_j\\&\qquad-(\boldsymbol{u}_i^\top\boldsymbol{v}_j)\left(\boldsymbol{\alpha}^\top\boldsymbol{z}^{(ij)}\right)\Big]\\\text{subject to}:\ &\boldsymbol{u}_i\geq\boldsymbol{0},\ \boldsymbol{v}_j\geq\boldsymbol{0},\ \forall i,j &\text{(6a)}\\&0\leq\boldsymbol{\alpha}^\top\boldsymbol{z}^{(ij)}\leq 1,\ \forall i,j. &\text{(6b)}\end{aligned}$$

The problem is still very challenging because of the coupled nature of $\boldsymbol{U},\boldsymbol{V}$, and $\boldsymbol{\alpha}$. In addition, the constraints on the optimization variables are not straightforward to handle. To tackle the formulated maximum likelihood estimation problem, we propose a *block coordinate descent* (BCD) [16] based algorithm. The basic idea is to cyclically solve 'partial' optimization problems w.r.t. a single block variable among $\boldsymbol{U}$, $\boldsymbol{V}$, and $\boldsymbol{\alpha}$, respectively, while fixing the other two. This way, the subproblems can be handled efficiently—leveraging computational tools for probabilistic NMF. The high-level algorithmic structure is summarized in Fig. 3, where we use $f(\boldsymbol{U},\boldsymbol{V},\boldsymbol{\alpha})$ and t to denote the objective function in (6) and the index of iteration, respectively. The updates are carried out in each iteration until a certain convergence criterion is met.

repeat the following until convergence:

$$\begin{aligned}\boldsymbol{\alpha}^{t+1}&\leftarrow\arg\min_{0\leq\boldsymbol{\alpha}^\top\boldsymbol{z}_{ij}\leq 1,\ \forall i,j} f(\boldsymbol{U}^t,\boldsymbol{V}^t,\boldsymbol{\alpha})\\\boldsymbol{U}^{t+1}&\leftarrow\arg\min_{\boldsymbol{U}\geq\boldsymbol{0}} f(\boldsymbol{U},\boldsymbol{V}^t,\boldsymbol{\alpha}^{t+1})\\\boldsymbol{V}^{t+1}&\leftarrow\arg\min_{\boldsymbol{V}\geq\boldsymbol{0}} f(\boldsymbol{U}^{t+1},\boldsymbol{V},\boldsymbol{\alpha}^{t+1})\\t&\leftarrow t+1\end{aligned}$$

Fig. 3. The algorithmic framework for the ML estimator in Eq. (6).

4.1 The α-update

First, consider the $\boldsymbol{\alpha}$-update. Since $\boldsymbol{U},\boldsymbol{V}$ are fixed, the relevant part of the objective function (6) is:

$$\begin{aligned}\underset{\boldsymbol{\alpha}}{\text{minimize}}\ &\sum_{(i,j)\in\Omega}-\left[y_{ij}\log\left(\boldsymbol{\alpha}^\top\boldsymbol{z}^{(ij)}\right)-\left(\boldsymbol{\alpha}^\top\boldsymbol{z}^{(ij)}\right)\lambda_{ij}\right]\\&\text{s.t. }0\leq\boldsymbol{\alpha}^\top\boldsymbol{z}^{(ij)}\leq 1,\end{aligned}\tag{7}$$

where we have omitted the superscript 't' for notational simplicity. We use $\lambda_{ij}=\boldsymbol{u}_i^\top\boldsymbol{v}_j$ to denote the current estimate of λ_{ij}. This is a convex optimization problem. Nevertheless, there is a serious scalability issue if standard optimization toolboxes (e.g., CVX and the interior point method (IPM) [22], [23]) are employed. The scalability challenge lies in the large number of constraints, i.e., IJ constraints, which makes IPM-like algorithms very slow (i.e., $O(R^2IJ)$ flops are needed and both I and J can be large).

To circumvent the scalability issue, we propose an *alternating direction method of multipliers* (ADMM) [24] based algorithm. As will be seen, with judicious reformulation, the updates of the ADMM algorithm are all very lightweight. Specifically, we introduce an auxiliary variable $\boldsymbol{P}\in\mathbb{R}^{I\times J}$ such that $\boldsymbol{P}(i,j)=p_{ij}=\boldsymbol{\alpha}^\top\boldsymbol{z}^{(ij)}$ and we denote

$$\boldsymbol{p}=\mathrm{vec}(\boldsymbol{P}).$$

Consequently, Problem (7) can be rewritten as

$$\begin{aligned}\min_{\boldsymbol{\alpha},\boldsymbol{p}}\ &\sum_{ij}-\left[y_{ij}\log p_{ij}-p_{ij}\lambda_{ij}\right]\\&\text{s.t. }0\leq p_{ij}\leq 1,\ \boldsymbol{\alpha}^\top\boldsymbol{z}^{(ij)}=p_{ij}.\end{aligned}\tag{8}$$

The augmented Lagrangian is

$$\mathcal{L}=\sum_{ij}-\left[y_{ij}\log p_{ij}-p_{ij}\lambda_{ij}\right]+\frac{\rho}{2}\|\boldsymbol{p}-\boldsymbol{Z}\boldsymbol{\alpha}+\boldsymbol{\omega}\|_2^2$$

where $\boldsymbol{p}$ is the vectorized version for $\boldsymbol{P}$, $\rho>0$ is a pre-specified regularization parameter (see details in [24, Chapter 3] for how to select this parameter), $\boldsymbol{\omega}=\mathrm{vec}(\boldsymbol{W})$,

$\boldsymbol{W}(i,j)$ is the dual variable associated with $\boldsymbol{\alpha}^\top \boldsymbol{z}^{(ij)} = p_{ij}$, and

$$\boldsymbol{Z} \in \mathbb{R}^{IJ \times R},\ \boldsymbol{Z}((j-1)I+i,:) = (\boldsymbol{z}^{(ij)})^\top,$$

which is a matrix that collects all $(\boldsymbol{z}^{(ij)})^\top$'s as its rows. The updates of the ADMM algorithm are as below:

$$\boldsymbol{p} \leftarrow \arg\min_{\boldsymbol{0}\le \boldsymbol{p}\le \boldsymbol{1}} \sum_{ij} -[y_{ij}\log p_{ij} - p_{ij}\lambda_{ij}] + \frac{\rho}{2}\|\boldsymbol{p} - \boldsymbol{Z}\boldsymbol{\alpha} + \boldsymbol{\omega}\|_2^2 \quad (9a)$$

$$\boldsymbol{\alpha} \leftarrow \arg\min_{\boldsymbol{\alpha}} \|\boldsymbol{p} - \boldsymbol{Z}\boldsymbol{\alpha} + \boldsymbol{\omega}\|_2^2 \quad (9b)$$

$$\boldsymbol{\omega} \leftarrow \boldsymbol{p} - \boldsymbol{Z}\boldsymbol{\alpha} + \boldsymbol{\omega} \quad (9c)$$

In (9), the dual update (i.e., $\boldsymbol{\omega}$) is trivial and naturally lightweight. The second line can be updated via the following closed form solution, since it is simply a least squares problem:

$$\boldsymbol{\alpha} \leftarrow \boldsymbol{Z}^\dagger(\boldsymbol{p}+\boldsymbol{\omega}), \quad (10)$$

and the pseudo-inverse $\boldsymbol{Z}^\dagger$ can be computed and stored in advance (which only needs to be computed once before the whole BCD algorithm starts)—meaning that the only operation needed here is matrix-vector multiplication.

It seems that the most difficult subproblem is Problem (9a). Nonetheless, this subproblem can be shown to admit a closed-form solution:

Lemma 2 *The solution to Problem* (9a) *can be obtained by the following algebraic form:*

$$\boldsymbol{p} \leftarrow \left[\frac{(\rho\bar{\boldsymbol{p}} - \boldsymbol{\lambda}) + \sqrt{(\rho\bar{\boldsymbol{p}} - \boldsymbol{\lambda})^2 + 4\rho\boldsymbol{y}}}{2\rho}\right]_{[0,1]}, \quad (11)$$

where all the power, square, and division operators are taken elementwise, $[\boldsymbol{Q}]_{[0,1]} = \min[\max[\boldsymbol{Q}, \boldsymbol{0}], \boldsymbol{1}]$, *and*

$$\boldsymbol{y} = \mathrm{vec}(\tilde{\boldsymbol{Y}}).$$

Proof: Notice that the problem is separable w.r.t. each (i,j), and thus it suffice to show (11) holds for each p_{ij}. In other words, Problem (9a) can be solved via solving the following scalar problem:

$$\underset{0\le p\le 1}{\text{minimize}} \ -y\log p + p\lambda + \frac{\rho}{2}(p-\bar{p})^2$$

where we omitted the subscripts and let

$$\bar{\boldsymbol{p}} = \boldsymbol{Z}\boldsymbol{\alpha} - \boldsymbol{\omega}.$$

First consider the case where $y > 0$. Then, the optimal solution must satisfy $p > 0$, since $p = 0$ results in an infinitely large objective value. This univariate function in the objective is *convex*, and thus can be easily solved via taking the first-order derivative w.r.t. p and setting it to zero, which leads to

$$\frac{-y}{p} + \lambda + \rho(p - \bar{p}) = 0.$$

The above amounts to a root-finding problem for a second-order polynomial. Hence, the optimal solution is either at the boundary of $p \in [0,1]$ (in this case, only 1) or

$$p = \left[\frac{(\rho\bar{p} - \lambda) + \sqrt{(\rho\bar{p} - \lambda)^2 + 4\rho y}}{2\rho}\right], \quad (12)$$

repeat the following until convergence:

update $\boldsymbol{p} \leftarrow$ Eq (11);
update $\boldsymbol{\alpha} \leftarrow \boldsymbol{Z}^\dagger(\boldsymbol{\omega} + \boldsymbol{p})$;
update $\boldsymbol{\omega} \leftarrow \boldsymbol{p} - \boldsymbol{Z}\boldsymbol{\alpha} + \boldsymbol{\omega}$;

Fig. 4. The ADMM algorithm for solving (7).

if the above is smaller than 1.

Also consider the case where $y = 0$. This is even simpler since the problem becomes

$$\underset{0\le p\le 1}{\text{minimize}} \ +p\lambda + \frac{\rho}{2}(p-\bar{p})^2,$$

which is a univariate quadratic problem. One can show that

$$p = \frac{(\rho\bar{p} - \lambda) + \sqrt{(\rho\bar{p} - \lambda)^2}}{2\rho}$$

or on the boundary of $[0,1]$ by the same reasoning. Therefore, the solution for $\boldsymbol{p}$ can be summarized as that in (11). □

The ADMM algorithm is summarized in Fig. 4. One can see that all the updates are very lightweight and thus the algorithm is easily scalable. Since the $\boldsymbol{\alpha}$-subproblem is convex, the ADMM algorithm guarantees to solve it to optimality [24].

4.2 The U-update

The subproblem w.r.t. $\boldsymbol{u}_i$ and $\boldsymbol{v}_j$ is

$$\min_{\boldsymbol{u}_i\ge\boldsymbol{0}, \boldsymbol{v}_j\ge\boldsymbol{0}} \sum_{i,j}\left[p_{ij}\boldsymbol{u}_i^\top\boldsymbol{v}_j - y_{ij}\log(\boldsymbol{u}_i^\top\boldsymbol{v}_j)\right], \quad (13)$$

where $p_{ij} = \boldsymbol{\alpha}^\top\boldsymbol{z}^{(ij)}$. In essence, the above problem can be understood as a weighted version of KL-divergence based nonnegative matrix factorization— the only difference between (13) and KL-divergence NMF is that the terms $\boldsymbol{u}_i^\top\boldsymbol{v}_j$ are scaled by p_{ij}. Techniques such as multiplicative updates (MU) [25] that are used for KL-divergence NMF can still be applied here, with careful modifications. To be specific, consider the objective function of the $\boldsymbol{U}$-subproblem, which can be decoupled to I subproblems w.r.t. $\boldsymbol{u}_i$ as follows:

$$f(\boldsymbol{u}_i) = \sum_j \left[p_{ij}\boldsymbol{u}_i^\top\boldsymbol{v}_j - y_{ij}\log(\boldsymbol{u}_i^\top\boldsymbol{v}_j)\right]$$

and its tight upper bound

$$g(\boldsymbol{u}_i, \bar{\boldsymbol{u}}_i) = \underbrace{\boldsymbol{u}_i^\top\left(\sum_j p_{ij}\boldsymbol{v}_j\right)}_{\bar{\boldsymbol{u}}_i} - \sum_j y_{ij}\sum_{r=1}^R \beta_r^{(ij)}\log\left(\frac{u_{i,r}v_{j,r}}{\beta_r^{(ij)}}\right), \quad (14)$$

where $\beta_r^{(ij)} = \frac{\bar{u}_{i,r}v_{j,r}}{\bar{\boldsymbol{u}}_i^\top\boldsymbol{v}_j}$. Note that the upper bound is derived from the Jensen's inequality; see details in [19], [20]. The upper bound is tight in the following sense:

$$f(\boldsymbol{u}_i) \le g(\boldsymbol{u}_i, \bar{\boldsymbol{u}}_i),\ \forall \boldsymbol{u}_i \ge \boldsymbol{0}, \quad (15a)$$

$$f(\bar{\boldsymbol{u}}_i) = g(\bar{\boldsymbol{u}}_i, \bar{\boldsymbol{u}}_i) \quad (15b)$$

$$\nabla_{\boldsymbol{u}_i} f(\bar{\boldsymbol{u}}_i) = \nabla_{\boldsymbol{u}_i} g(\bar{\boldsymbol{u}}_i, \bar{\boldsymbol{u}}_i); \quad (15c)$$

i.e., the two functions are 'tangent' at $\boldsymbol{u}_i = \bar{\boldsymbol{u}}_i$.

The idea of MU-like algorithms is to update $\boldsymbol{u}_i$ via solving

$$\boldsymbol{u}_i \leftarrow \arg\min_{\boldsymbol{u}_i \geq \boldsymbol{0}} \; g(\boldsymbol{u}_i, \bar{\boldsymbol{u}}_i), \tag{16}$$

instead of directly solving $f(\boldsymbol{u}_i)$ that is hard to tackle. This is widely known to be *majorization minimization* (MM) in the optimization literature [26], which ensures decreasing the cost value of $f(\boldsymbol{u}_i)$ in each iteration.

To solve the surrogate optimization problem (16), let us take derivative w.r.t. $u_{i,r}$, we have

$$\begin{aligned}\nabla_{u_{i,r}} g(\boldsymbol{u}_i, \bar{\boldsymbol{u}}_i) &= \tilde{u}_{i,r} - \sum_j y_{i,j} \beta_r^{(ij)} \left(\frac{\beta_r^{(ij)}}{u_{i,r} v_{j,r}}\right) \left(\frac{v_{j,r}}{\beta_r^{(ij)}}\right) \\ &= \tilde{u}_{i,r} - \left(\sum_j y_{i,j} \beta_r^{(ij)}\right) \frac{1}{u_{i,r}}.\end{aligned}$$

Setting the above equal to zero leads to

$$u_{i,r} = \frac{\left(\sum_j y_{i,j} \beta_r^{(ij)}\right)}{\tilde{u}_{i,r}}, \tag{17}$$

which is surely nonnegative, being reminiscent of the multiplicative update (MU)—under the condition that $\tilde{u}_{i,r}$ is nonzero, which hardly happens in practice if $\boldsymbol{U}$, $\boldsymbol{V}$ and $\boldsymbol{P}$ are initialized with nonzero matrices.

By role symmetry, we immediately have the following:

$$v_{j,r} = \frac{\left(\sum_i y_{i,j} \beta_r^{(ij)}\right)}{\tilde{v}_{j,r}}, \quad \tilde{\boldsymbol{v}}_j = \sum_i p_{i,j} \boldsymbol{u}_i \tag{18}$$

The update of $\boldsymbol{U}$ and $\boldsymbol{V}$ can be represented in a more compact form, i.e.,

$$\boldsymbol{U} \leftarrow (\boldsymbol{U} \circ \boldsymbol{\Phi}) \,/ \tilde{\boldsymbol{U}}, \; \boldsymbol{\Phi} = \left(\boldsymbol{Y} / \boldsymbol{U}\boldsymbol{V}^\top\right) \boldsymbol{V} \tag{19}$$

$$\boldsymbol{V} \leftarrow (\boldsymbol{V} \circ \boldsymbol{\Psi}) \,/ \tilde{\boldsymbol{V}}, \; \boldsymbol{\Psi} = \left(\boldsymbol{Y}^\top / \boldsymbol{V}\boldsymbol{U}^\top\right) \boldsymbol{U}, \tag{20}$$

where "$\circ$" denotes the Hadamard product and "$/$" denotes the elementwise division (i.e., "./" in Matlab). One can see that the above updates indeed exhibit a flavor of MU, with the newly defined $\tilde{\boldsymbol{U}}$ and $\tilde{\boldsymbol{V}}$ caused by incorporating the weighting matrix $\boldsymbol{P}$; see Eqs (14) and (18) for the definitions of the ith row of $\tilde{\boldsymbol{U}}$ and jth row of $\tilde{\boldsymbol{V}}$, respectively.

Remark 1 *Our formulation and algorithm naturally work for networks with many zeros without knowing if they are "undetected links" (observation made but no interaction between the associated pairs detected) or "true misses" (no observation made for the associated pairs). One remark is that in many cases, domain knowledge can be used to distinguish "undetected links" and "true misses", at least for part of the network. In incorporating such domain knowledge is often useful, especially when dealing with noisy real-world data. Minor modifications to the algorithm suffice to take "true misses" into consideration. To be specific, we can keep the same $\boldsymbol{\alpha}$-update as before, but only applied to the entries where observations were made (no matter if interactions were recorded). For the $\boldsymbol{U}, \boldsymbol{V}$-updates, we first impute the missing entries (where no observations were made) $\widehat{y}_{i,j} = \widehat{\boldsymbol{u}}_i^\top \widehat{\boldsymbol{v}}_j$, where $\widehat{\boldsymbol{u}}_i$, $\widehat{\boldsymbol{v}}_j$ denote the current estimations for $\boldsymbol{u}_i$ and $\boldsymbol{v}_j$, respectively, and then apply the proposed algorithm. This simple heuristic admits an expectation-maximization (EM) interpretation in the NMF literature [27].*

4.3 Convergence Properties

Our algorithm involves an exactly solved block subproblem (i.e., the $\boldsymbol{\alpha}$) and two inexactly solved blocks (i.e., the $\boldsymbol{U}$ and $\boldsymbol{V}$ blocks). One natural question is: does the algorithm converge? The answer is affirmative, under certain conditions. This can be shown by invoking recent results regarding *inexact* block coordinate descent [26], [28]. Specifically, we have the following proposition:

Proposition 1 *Assume that the $\boldsymbol{\alpha}$-subproblem is solved by the ADMM algorithm in each iteration, and $\boldsymbol{U}$ and $\boldsymbol{V}$ are updated by MU in each iteration. Also assume that there is no zero element appearing in $\boldsymbol{U}$ and $\boldsymbol{V}$ throughout the iterations. Then, every limit point of the solution sequence produced by the proposed algorithm is a stationary point of Problem* (6).

Proposition 1 asserts that the algorithm is convergent under certain conditions—and every limit point satisfies the necessary conditions for optimality. The detailed proof is relegated to Appendix B. Since we use a MU-like algorithm for updating $\boldsymbol{U}$ and $\boldsymbol{V}$, the caveat in theory is that convergence is only guaranteed if there is no zero elements appeared in $\boldsymbol{U}$ and $\boldsymbol{V}$ in each iteration—which is not easy to check or ensure. Even worse, if zero appears in $\boldsymbol{U}$ or $\boldsymbol{V}$, the update rules such as Eq. (18) are ill-defined. Nevertheless, there are many pragmatic ways to 'robustify' the algorithm, e.g., by adding a very small $\epsilon > 0$ to $\tilde{u}_{i,r}$ and $\tilde{v}_{j,r}$ in each iteration [20] [cf. Eq. (18)]. This usually improves the performance of MU-type algorithms [20], [25] (also see Appendix B).

5 Numerical Experiments

In this section, we offer a series of numerical results to showcase the effectiveness of the proposed algorithm. We start with synthetic data that obeys the considered generative model, as a sanity check for convergence, complexity, and estimation accuracy. Then, we use the pollination network data collected in the Cascades in Oregon [9] and the host-parasite network data collected in a New Mexican desert ecosystem [29] to evaluate the performance of the proposed approach on real ecological network link prediction problems.

5.1 Synthetic Data

We generate the synthetic data following the model described in (4). Specifically, we first generate $\boldsymbol{U}$ and $\boldsymbol{V}$ whose elements are drawn uniformly at random between zero and $\gamma > 0$—note that we change γ under different problem settings such that $\boldsymbol{u}_i^\top \boldsymbol{v}_j$ is not too small or too large, leading to pathological cases. We also enforce the *separability condition* on $\boldsymbol{U}$ and $\boldsymbol{V}$ so that the underlying NMF model is identifiable [30]–[32]. Then, we generate the feature vectors $\boldsymbol{z}^{(ij)}$ and the linear regression coefficients $\boldsymbol{\alpha}$ following the same uniform distribution.

For synthetic data, we use two baselines as benchmarks. The first one is the Poisson NMF algorithm [20], [25] that handles Problem (3), without considering Binomial selection. The second algorithm is MC-based CF that handles a similar problem as in (1), with regularizations for enhancing performance [33].

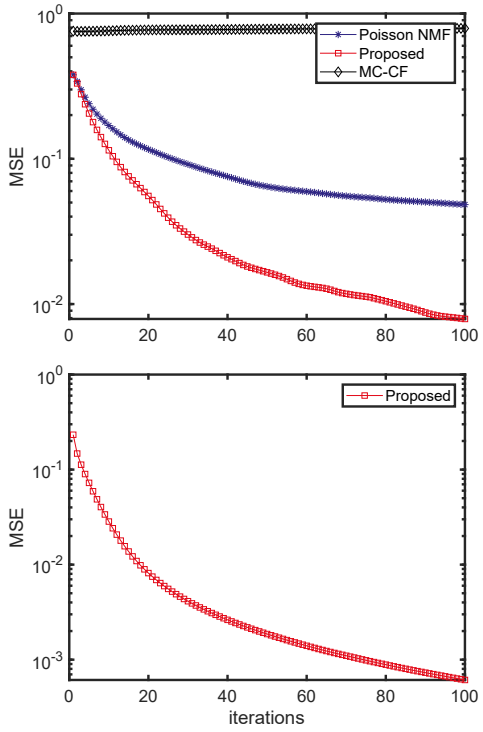


Fig. 5. The average MSEs of the estimated $\boldsymbol{U}, \boldsymbol{V}$ (left) and $\boldsymbol{\alpha}$ (right) produced by the algorithms under test.

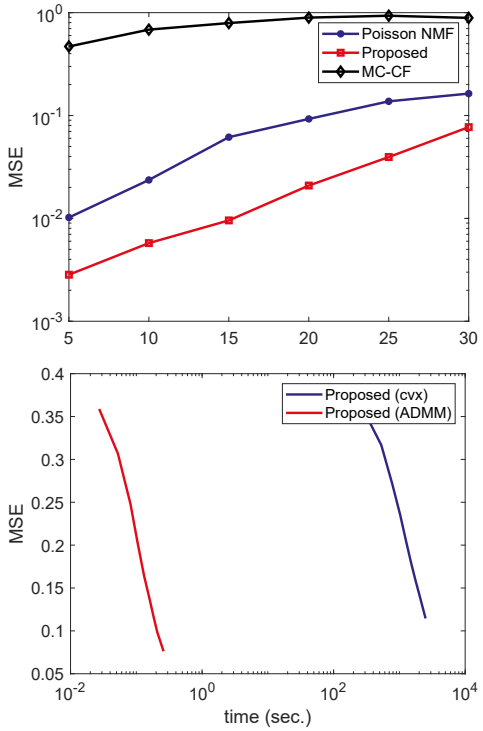


Fig. 6. Left: the average MSEs of the estimated $\boldsymbol{\alpha}$ under different ranks. Right: Runtime performance of the proposed algorithm using ADMM and CVX, respectively.

We adopt a straightforward evaluation strategy for simulated data. Since we know the ground-truth model parameters $\boldsymbol{U}$, $\boldsymbol{V}$, and $\boldsymbol{\alpha}$, we use mean squared error (MSE) to evaluate performance; e.g., the MSE of the estimated $\boldsymbol{U}$ is defined as following:

$$\mathsf{MSE} = \min_{\pi(f)\in\{1,\ldots,F\}} \frac{1}{F}\sum_{f=1}^{F}\left\| \frac{\boldsymbol{U}(:,\pi(f))}{\|\boldsymbol{U}(:,\pi(f))\|_2} - \frac{\widehat{\boldsymbol{U}}(:,f)}{\|\widehat{\boldsymbol{U}}(:,f)\|_2}\right\|_2^2$$

where $\widehat{\boldsymbol{U}}$ denotes the estimate of $\boldsymbol{U}$ and $\pi(f)$'s are under the constraint $\{\pi(1),\ldots,\pi(F)\} = \{1,\ldots,F\}$—i.e., $[\boldsymbol{U}(:,\pi(1)),\ldots,\boldsymbol{U}(:,\pi(F))]$ is a column-permuted version of $\boldsymbol{U}$. We use the above because NMF has intrinsic scaling and permutation ambiguities, which need to be fixed before evaluation [30], [34]. For $\boldsymbol{V}$ we use the same evaluation. For $\boldsymbol{\alpha}$, we measure the MSE by $\mathsf{MSE} = \frac{1}{R}\|\widehat{\boldsymbol{\alpha}} - \boldsymbol{\alpha}\|_2^2$, where $\widehat{\boldsymbol{\alpha}}$ denotes the algorithm-estimated $\boldsymbol{\alpha}$.

All the algorithms under test in this subsection are coded in Matlab, and are all initialized by the same random $\boldsymbol{U}$ and $\boldsymbol{V}$. For the proposed method, $\boldsymbol{\alpha}$ is also randomly initialized. All the entries of the initializations of $\boldsymbol{U}, \boldsymbol{V}$, and $\boldsymbol{\alpha}$ are drawn uniformly at random between zero and one.

Fig. 5 (left) shows the averaged MSEs of the estimated $\boldsymbol{U}$ and $\boldsymbol{V}$ of a case where $R = 8$, $I = J = 50$ and $F = 15$. In this simulation we set $\gamma = 15$. The results are averaged over 50 random trials. One can see that MC-CF does not work for this data model, since it is not specialized for integer data and does not take into consideration the imperfect detection effect. The Poisson NMF algorithm works to a certain extent—since it is a natural choice for count data as in our model. Nevertheless, the estimation accuracy is not ideal. The proposed algorithm exhibits the highest estimation accuracy because it explicitly accounts for imperfect detection. In particular, when the number of iteration reaches 100, the MSE output by the proposed method is all most one order of magnitude lower relative to Poisson NMF. This shows that considering the imperfect detection effect explicitly is quite helpful for model identification. This result also shows that our optimization strategy is quite effective in handling the hard ML optimization problem.

Fig. 5 (right) shows the average MSE of $\boldsymbol{\alpha}$ for the same experiment. One can see that the accuracy increases along with the iterations—and merely using 100 iterations achieves quite a satisfactory estimation accuracy for this case. This is encouraging, and indicates that identifying the underlying NMF model together with the selection model is possible, despite of the complex nature of this generative model.

Fig. 6 (left) shows the estimation accuracy of latent factors $\boldsymbol{U}, \boldsymbol{V}$ when the rank changes under $I = J = 50$. The performance of all the algorithms under test deteriorate gracefully when R increases – which is understandable since for fixed data size, increasing the rank of $\boldsymbol{U}\boldsymbol{V}^\top$ means increasing the number of unknown parameters, and thus increasing the problem difficulty. Nevertheless, the proposed algorithm outperforms the benchmarks under all ranks under test.

Fig. 6 (right) shows the efficiency of the proposed ADMM-based α-update. The setting is the same as that use in Fig. 5. One can see that the ADMM algorithm outperforms a generic convex optimization solver by a very large margin, because the updates in the ADMM algorithm are all simple operations.

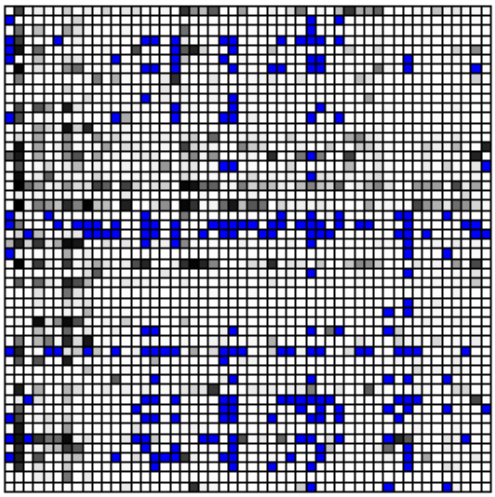


Fig. 7. Visualization of the real pollination data subset we analyzed. Plants are rows, pollinators are columns, and blue entries means interaction between the corresponding pairs never co-occurred in space/time. Gray level of the grids are used to represent the number of observed counts.

5.2 Ecological Network Data

Plant-Pollinator Network Dataset. We focus here on data collected by undergraduate researchers at the H.J. Andrews (HJA) Long-term Ecological Research site in the Oregon Cascade mountains, which are publicly available [9]. These data are comprised of observations at about 12 meadows over seven years. Students record every visit they observe from a pollinator to a flower—688 pollinator species and 148 plant species so far. The data are sparse; only about 3.5% of all possible links have ever been observed. The visitation counts are also highly skewed, with many links observed only once and a few observed over 1,000 times. The recorded zeros in the data are a mix of true zeros and links that likely occurred but were unobserved during the students' watches.

These network data are accompanied by *side information* of varying types. First, before recording the visits, the students exhaustively survey the observation plots and record the number of stalks per flowering plant species present. In addition, there are data describing traits of the species themselves, like flower color and pollinator tongue length.

In this analysis, we focused on the most common species observed in the visitation data for which we also had information about species traits. Based on commonness and trait information, we constructed a 50×50 plant-pollinator visitation matrix for analysis by aggregating over all observations from 2011 through 2017 (Fig. 7). For these common species, 42% of the entries were non-zero. Among the zero entries, we used the flower survey data to distinguish between two cases of zeros: 1) the plant and pollinator species occurred in the same time and place at least once but did not interact (49% of the matrix; white cells in Fig. 7), vs. 2) the plant and pollinator species never co-occurred (9% of the matrix; blue cells in Fig. 7). We treated the second case as truly missing values. For the first kind of zeros, we leave it to our probabilistic model to determine which are true zeros and which may have been undercounted. We used six categorical features in $\boldsymbol{z}^{(ij)}$ based on species traits: pollinator energy requirements (7 values), pollinator tongue length (8 values), plant soil community (8 values), plant life form (2 values), plant flower form (2 values), and plant exclusion platform type (whether it excluded pollinators based on its shape; 3 values).

Host-Parasite Dataset We also use another ecological dataset, v.i.z., the host-parasite networks that is a part of the Sevilleta Long-Term Ecological Research program. The networks are also publicly available [17]. These data are collected over 5 years at six sites, resulting in 22 host species and 87 parasite species except for host species with less than five observations during the sampling period. These data also form a sparse and highly skewed visit count graph; about 13.5% of all possible links have been observed.

These host-parasite network data also contain host and parasite traits as side information. Host traits include life history traits (e.g., host diet breadth, body mass, home range size, maximum age, and species abundance) and some phylogenetic information. Parasite traits consist of life history, transmission modes, genus, type, and location.

Similar to the plant-pollinator dataset, we focus on the most common 19 host and 49 parasite species observed for analysis. For these species, 22% of the entries are non-zero. Unlike the pollination data, the host-parasite dataset does not have additional information to distinguish "zeros" or "truly missing entries".

Semi-Real Data Evaluation. While the primary scientific objective for pollination networks is inference of the latent counts $\hat{\boldsymbol{N}}$ from the observations $\boldsymbol{Y}$, we do not have ground truth for $\boldsymbol{N}$ against which to compare our predictions. Instead, we employ two other evaluation strategies that are possible with the data at hand. First, we "observe" a real pollination data matrix according to our detection model with known parameters. This essentially replicates the simulated data evaluation above with real count networks acting as $\boldsymbol{N}$. The point here is that real counts $\boldsymbol{N}$ are not likely to strictly follow the Poisson distribution, so this analysis asks whether or not our Poisson modeling is useful in practice. The rationale is as follows: If our two-layer model holds, the observed counts are Poisson distributed with parameter $p_{ij}\lambda_{ij}$ (also see Lemma 2). Our experiments here essentially treat the real observed counts as $\boldsymbol{N}$ in our model and impose another layer of Binomial detection. If we can accurately identify the imposed the detection model (i.e., accurately estimating $\boldsymbol{\alpha}$), it implies that the Poisson modeling for the counts is quite promising.

We test the algorithms under two settings. First, we use a constant detection probability for all entries. Second, we generate $p_{ij} = \boldsymbol{\alpha}^\top \boldsymbol{z}^{(i,j)}$ using real traits $\boldsymbol{z}^{(i,j)}$ and a random vector $\boldsymbol{\alpha}$; the generating process is carefully controlled such that $p_{ij} \in [0, 1]$. When the detection model consists of a constant detection probability of 0.9, the model learns this parameter with generally low MSEs [see Fig. 8, 9 (upper)], if the rank is appropriately selected.For the second case, one can see that the estimation of $\boldsymbol{\alpha}$ is even more robust to the change of ranks [see Fig. 8, 9 (lower)], perhaps because the second case is more consistent with the assumed generative model. Note that the MSE of the estimated $\boldsymbol{\alpha}$ is not monotonic along the iterations, which is understand-

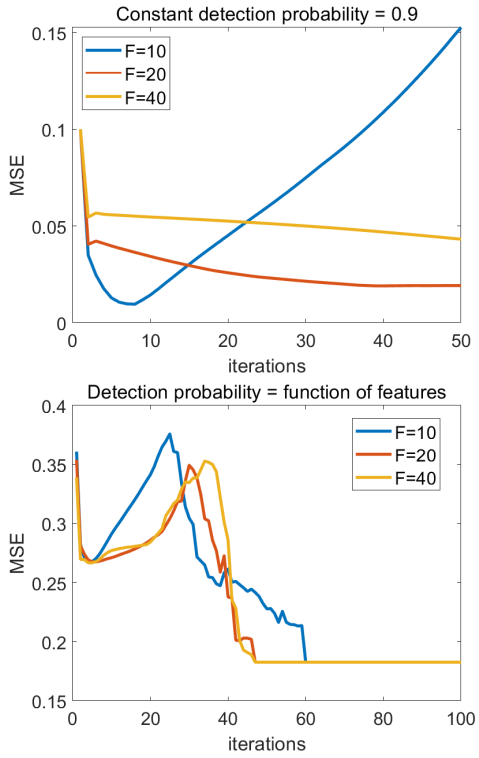


Fig. 8. The average MSEs of the estimated $\boldsymbol{\alpha}$ produced by the algorithms in the plant-pollinator network. Left: constant detection probability. Right: feature-dependent detection probability.

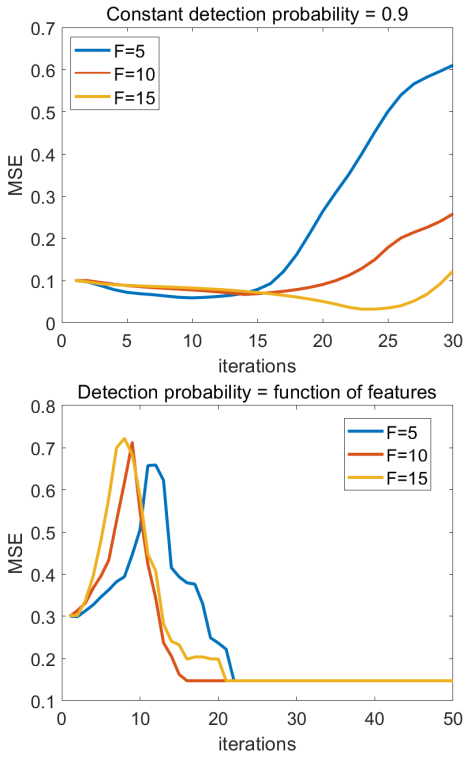


Fig. 9. The average MSEs of the estimated $\boldsymbol{\alpha}$ produced by the algorithms in the host-parasite network. Left: constant detection probability. Right: feature-dependent detection probability.

able since BCD algorithms only ensure monotonic decrease of the objective function. These results indicate that the Poisson modeling in the proposed framework is plausible. In addition, the results here also suggest that selecting an appropriate rank F and using a good stopping criterion are important.

Real Data Evaluation. The second evaluation strategy is a more typical approach in collaborative filtering: we split the matrix elements into ten folds and compare the true $\boldsymbol{Y}$ with $\hat{\boldsymbol{Y}}$ on held-out elements. We compute relative root mean squared error ($\mathrm{rRMSE} = \mathrm{RMSE}/\overline{Y}$, where $\overline{Y}$ is the mean value of $\boldsymbol{Y}$), area under the receiver operating characteristic curve (AUROC), and area under the precision-recall curve (AUPRC) on predictions for the non-missing elements aggregated across the ten test folds. Since the pollination network has some truely missing entries, we use the EM variation mentioned in Remark 1 for this dataset. We compare against four other matrix factorization methods: Poisson NMF, implicit feedback matrix factorization (IFMF) [35], MC-CF, truncated singular value decomposition (SVD). We used performance on a validation fold as a stopping criterion for the iterative methods and ran each method with five different ranks ($F \in \{2, 5, 10, 20, 40\}$). The proposed method uses the species traits as detection features in the binomial model. We also use a competitive baseline, namely, the recently developed *neural network-based collaborative filtering* (NCF) [36] as a benchmark. We also compare against Poisson regression [37], boosted regression trees (BRT; with Poisson link) [38] and random forests (RF) [39] which predict the same ten folds without using the matrix structure. For NCF model, we tested the batch size of [256, 512], the layer-wise neuron setup of [8, 16, 32, 64] and [16, 32, 64, 128] and the learning rate of [0.0001, 0.0005, 0.001, 0.005] to tune hyper-parameters. The tree-based methods were tuned among $\{1000, 2500, 5000, 10000\}$ trees. These baselines speak to the utility of the trait data.

TABLE 1
Performance of eight methods predicting held-out interactions in the plant-pollinator network data. Rank and no.of trees were tuned on a validation fold to minimize rRMSE; reporting majority vote of rank and no. trees and average iterations over 10 folds. Methods above the double line employ some form of matrix factorization; methods below the double line predict interactions solely based on trait features.

Method	Rank	rRMSE or Trees	AUROC	AUPRC
Pois. N-mix.	5	**3.873**	**0.750**	**0.733**
Pois. NMF	5	9.038	0.735	0.709
MC-CF	2	5.939	0.622	0.644
IFMF	2	7.444	0.631	0.626
Trunc. SVD	2	15.349	0.526	0.535
NCF	n/a	n/a	0.731	0.695
Pois. Regr.	n/a	10.241	0.588	0.524
BRT	n/a	10.187	0.603	0.537
RF	n/a	10.308	0.661	0.574

The results show that the proposed approach outperforms all competitors on all three metrics (Table 1). Among the baselines, Poisson NMF have the most similar performance as that of the proposed method. This is not entirely surprising—as we have seen in Lemma 1, if $\boldsymbol{N}$ is Poisson distributed, then $\boldsymbol{Y}$ is also Poisson distributed, given that the observation is truly a Binomial selection process. Nevertheless, the proposed method still performs better compared to a simple Poisson NMF formulation, even for

TABLE 2
Performance of eight methods predicting held-out interactions in the host-parasite network data.

Method	Rank	rRMSE or Trees	AUROC	AUPRC
Pois. N-mix.	5	**3.538**	**0.759**	**0.692**
Pois. NMF	10	3.781	0.707	0.635
MC-CF	5	3.644	0.665	0.627
IFMF	2	3.872	0.639	0.587
Trunc. SVD	2	5.335	0.462	0.451
NCF	n/a	n/a	0.689	0.393
Pois. Regr.	n/a	5.877	0.676	0.370
BRT	n/a	5.905	0.697	0.383
RF	n/a	5.920	0.663	0.343

the task of predicting $\boldsymbol{Y}$. The reason might be that the proposed approach can effectively incorporate trait features, which should be helpful in practice. The proposed model also outperforms NCF which is known to be effective for collaborative filtering by replacing the inner product with a neural architecture. It shows that capturing hidden interaction mechanisms from the sampled counts in the field, which could have many missing data points unlike user feedback, can be a challenging problem to user-oriented recommendation models. In this regard, the proposed approach can predict additional information in the form of latent network counts, i.e., $\boldsymbol{N}$ (to be discussed shortly), which is more important in ecology.

As a third evaluation strategy, we examine predictions made from our model for the true missing entries in the pollination network (blue in Fig. 7). We sort the 226 missing entries from largest to smallest predicted interaction counts. We showed the top ten and bottom ten interactions to an expert entomologist and asked which set were more likely to interact if they did co-occur (without revealing our predictions). The expert chose the set of the top ten interactions as more likely than the bottom ten, in part because the bottom ten included some plants with exclusionary characteristics that would forbid some insects from visiting them. While it is challenging to get quantitative validation, this qualitative feedback is encouraging and also very intriguing.

Finally, we examine differences in the $\hat{\boldsymbol{N}}$ estimated from pollination data with the observed $\boldsymbol{Y}$. As mentioned above, only 42% of the possible links in the pollination network were observed. In contrast, 80.1% of the links were estimated to have a latent count greater than zero. This implies that the data maybe only recorded roughly half of the actual links in the network. This effect of imperfect detection extends to other network statistics of ecological importance as well. One potential application of these estimates $\hat{\boldsymbol{N}}$ is that they can be offered to ecologists as references for designing and taking field observations.

6 Conclusion and Discussion

In this work, we proposed a two-layer statistical model for network analysis in ecology. Our model captures a key aspect of many ecological networks (e.g., pollination networks)—the interaction counts between species are usually systematically undercounted, which makes existing collaborative filtering approaches inapplicable for such networks. We proposed a generative model that is a judicious integration of Poisson low-rank latent matrix factorization and Binomial selection, and we devised an effective optimization algorithm to handle the associated challenging maximum likelihood model identification problem. We evaluated the proposed method on both synthetic and real data. Excitingly, evaluation on the real pollination network shows that the proposed model and algorithm are promising.

As future work, one possible extension is to introduce a richer model for the Binomial selection stage. Our current model uses a linear regression model, which strikes a good balance between simplicity and effectiveness. However, a nonlinear regression model using kernels or neural nets may capture the reality even better, thereby further boosting performance. In addition, we plan to develop a version of this modeling framework that allows both latent *and* observed features to inform the network model. This would allow the trait data to influence the interaction counts directly rather than through the effects of imperfect detection, leaving the latent features to capture unmeasured aspects of the propensity for interactions to occur.

References

[1] L. Lü and T. Zhou, "Link prediction in complex networks: A survey," *Physica A: statistical mechanics and its applications*, vol. 390, no. 6, pp. 1150–1170, 2011.

[2] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," *Journal of the American society for information science and technology*, vol. 58, no. 7, pp. 1019–1031, 2007.

[3] F. Ricci, L. Rokach, and B. Shapira, "Recommender systems: introduction and challenges," in *Recommender systems handbook*. Springer, 2015, pp. 1–34.

[4] L. Xiong, X. Chen, T.-K. Huang, J. Schneider, and J. G. Carbonell, "Temporal collaborative filtering with bayesian probabilistic tensor factorization," in *Proceedings of the 2010 SIAM International Conference on Data Mining*. SIAM, 2010, pp. 211–222.

[5] Q. Gu, J. Zhou, and C. Ding, "Collaborative filtering: Weighted nonnegative matrix factorization incorporating user and item graphs," in *Proceedings of the 2010 SIAM international conference on data mining*. SIAM, 2010, pp. 199–210.

[6] K. Yu, S. Zhu, J. Lafferty, and Y. Gong, "Fast nonparametric matrix factorization for large-scale collaborative filtering," in *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2009, pp. 211–218.

[7] R. H. Keshavan, A. Montanari, and S. Oh, "Matrix completion from a few entries," *IEEE transactions on information theory*, vol. 56, no. 6, pp. 2980–2998, 2010.

[8] E. J. Candes and Y. Plan, "Matrix completion with noise," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 925–936, 2010.

[9] J. A. Jones and V. Pfeiffer, "Plant pollinator data at HJ Andrews Experimental Forest, 2011 to present."

[10] J. M. Tylianakis, E. Laliberté, A. Nielsen, and J. Bascompte, "Conservation of species interaction networks," *Biological Conservation*, vol. 143, no. 10, pp. 2270–2279, 2010. [Online]. Available: http://dx.doi.org/10.1016/j.biocon.2009.12.004

[11] J. Memmott, N. M. Waser, and M. V. Price, "Tolerance of pollination networks to species extinctions." *Proceedings of the Royal Society B: Biological Sciences*, vol. 271, no. 1557, pp. 2605–11, Dec 2004. [Online]. Available: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=1691904{\&}tool=pmcentrez{\&}rendertype=abstract

[12] E. Seo and R. A. Hutchinson, "Predicting Links in Plant-Pollinator Interaction Networks using Latent Factor Models with Implicit Feedback," in *AAAI'18*, 2018.

[13] D. I. MacKenzie, J. D. Nichols, J. A. Royle, K. H. Pollock, L. L. Bailey, and J. E. Hines, *Occupancy estimation and modeling: inferring patterns and dynamics of species occurrence*. Elsevier, San Diego, USA, 2006.

[14] D. I. MacKenzie, J. D. Nichols, G. B. Lachman, S. Droege, J. A. Royle, and C. A. Langtimm, "Estimating Site Occupancy Rates When Detection Probabilities Are Less Than One," *Ecology*, vol. 83, no. 8, pp. 2248–2255, Aug 2002. [Online]. Available: http://www.esajournals.org/doi/abs/10.1890/0012-9658(2002)083[2248:ESORWD]2.0.CO;2

[15] J. A. Royle, "N-Mixture Models for Estimating Population Size from Spatially Replicated Counts," *Biometrics*, vol. 60, no. March, pp. 108–115, 2004.

[16] D. P. Bertsekas, *Nonlinear programming*. Athena Scientific, 1999.

[17] T. Dallas, J. M. Drake, and A. W. Park, "Data and code to reproduce Dallas, Park, and Drake "Predicting cryptic links in host-parasite networks"," 5 2017. [Online]. Available: https://figshare.com/articles/Data_and_code_to_reproduce_Dallas_Park_and_Drake_Predicting_cryptic_links_in_host-parasite_networks_/4965038

[18] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan, "Large-scale parallel collaborative filtering for the netflix prize," in *International Conference on Algorithmic Applications in Management*. Springer, 2008, pp. 337–348.

[19] C. Févotte, N. Bertin, and J.-L. Durrieu, "Nonnegative matrix factorization with the itakura-saito divergence: With application to music analysis," *Neural computation*, vol. 21, no. 3, pp. 793–830, 2009.

[20] E. C. Chi and T. G. Kolda, "On tensors, sparsity, and nonnegative factorizations," *SIAM Journal on Matrix Analysis and Applications*, vol. 33, no. 4, pp. 1272–1299, 2012.

[21] E. B. Dennis, B. J. Morgan, and M. S. Ridout, "Computational aspects of n-mixture models," *Biometrics*, vol. 71, no. 1, pp. 237–246, 2015.

[22] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambriadge Press, 2004.

[23] M. Grant, S. Boyd, and Y. Ye, "Cvx: Matlab software for disciplined convex programming," 2008.

[24] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, pp. 1–122, 2011.

[25] D. Seung and L. Lee, "Algorithms for non-negative matrix factorization," *Advances in neural information processing systems*, vol. 13, pp. 556–562, 2001.

[26] M. Razaviyayn, M. Hong, and Z.-Q. Luo, "A unified convergence analysis of block successive minimization methods for nonsmooth optimization," *SIAM Journal on Optimization*, vol. 23, no. 2, pp. 1126–1153, 2013.

[27] Y.-D. Kim and S. Choi, "Weighted nonnegative matrix factorization," in *Proc. IEEE ICASSP 2019*, 2009, pp. 1541–1544.

[28] M. Hong, M. Razaviyayn, Z.-Q. Luo, and J.-S. Pang, "A unified algorithmic framework for block-structured optimization involving big data: With applications in machine learning and signal processing," *IEEE Signal Process. Mag.*, vol. 33, no. 1, pp. 57–77, 2016.

[29] T. Dallas, A. W. Park, and J. M. Drake, "Predicting cryptic links in host-parasite networks," *PLoS computational biology*, vol. 13, no. 5, p. e1005557, 2017.

[30] X. Fu, K. Huang, N. D. Sidiropoulos, and W.-K. Ma, "Nonnegative matrix factorization for signal and data analytics: Identifiability, algorithms, and applications," *arXiv preprint arXiv:1803.01257*, 2018.

[31] D. Donoho and V. Stodden, "When does non-negative matrix factorization give a correct decomposition into parts?" in *NIPS*, vol. 16, 2003.

[32] K. Huang, N. D. Sidiropoulos, and A. Swami, "Non-negative matrix factorization revisited: Uniqueness and algorithm for symmetric decomposition," *IEEE Transactions on Signal Processing*, vol. 62, no. 1, pp. 211–224, 2014.

[33] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, no. 8, pp. 30–37, 2009.

[34] N. Gillis, "The why and how of nonnegative matrix factorization," *Regularization, Optimization, Kernels, and Support Vector Machines*, vol. 12, p. 257, 2014.

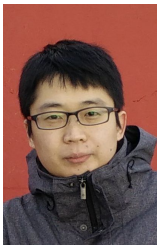
[35] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative Filtering for Implicit Feedback Datasets," *Eighth IEEE International Conference on Data Mining (ICDM'08)*, 2008. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs{_}all.jsp?arnumber=4781121

[36] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th International Conference on World Wide Web*, ser. WWW '17. Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2017, pp. 173–182. [Online]. Available: https://doi.org/10.1145/3038912.3052569

[37] I. Marschner, *glm2: Fitting Generalized Linear Models*, 2018, r package version 1.2.1. [Online]. Available: https://CRAN.R-project.org/package=glm2

[38] B. Greenwell, B. Boehmke, and J. Cunningham, *gbm: Generalized Boosted Regression Models*, 2019, r package version 2.1.5. [Online]. Available: https://CRAN.R-project.org/package=gbm

[39] A. Liaw, M. Wiener, L. Breiman, and A. Cutler, *randomForest: Breiman and Cutler's Random Forests for Classification and Regression*, 2018, r package version 4.6-14. [Online]. Available: https://CRAN.R-project.org/package=randomForest



Xiao Fu (S'12-M'15) is an Assistant Professor in the School of Electrical Engineering and Computer Science, Oregon State University, Corvallis, Oregon, United States. He received his Ph.D. degree in Electronic Engineering from The Chinese University of Hong Kong (CUHK), Hong Kong, 2014. He was a Postdoctoral Associate in the Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN, United States, from 2014-2017. His research interests include the broad area of signal processing and machine learning. He received a Best Student Paper Award at ICASSP 2014, and co-authored a Best Student Paper Award at IEEE CAMSAP 2015.

Eugene Seo received her B.Eng. degree in Computer Science and Electronic Engineering from Handong Global University, South Korea, in 2007, and M.S. degree in Computer Science from Korea Advanced Institute of Science and Technology, South Korea, in 2011. She is currently a Ph.D. candidate student at the School of Electrical Engineering and Computer Science, Oregon State University, Corvallis, Oregon, United States. Her research interests include artificial intelligence, machine learning, data mining, recommender systems, and computational sustainability.

Justin Clarke is a senior at Oregon State University majoring in Computer Science. He is currently working as an intern at the NASA Ames Research Center in the Robust Software Engineering group. This fall he will join the Ph.D. program at the University of Massachusetts, Amherst. His research interests include machine learning and causal modeling.

Rebecca A. Hutchinson received her B.S. degree in Computer Science and Engineering from Bucknell University, Lewisburg, PA, United States, in 2002. She received her Ph.D. degree in Computer Science from Carnegie Mellon University, Pittsburgh, PA, United States, in 2009. From 2009 to 2015 she has been a Post Doctoral Scholar and Fellow with the School of Electrical Engineering and Computer Science, Oregon State University, Corvallis, Oregon, United States. She is currently a Assistant Professor in the School of Electrical Engineering and Computer Science and the Department of Fisheries and Wildlife, Oregon State University, Corvallis, Oregon, United States. Her research is at the intersection of machine learning and ecology, including computational sustainability, species distribution modeling, hierarchical latent variable models, and robust parameter estimation methods.

APPENDIX A
PROOF OF LEMMA 1

The derivation is a classic result in statistics [21]. To obtain the simple expression in (5), let us begin with the following:

$$\sum_{n=y_{ij}}^{\infty} \frac{\lambda_{ij}^n e^{-\lambda_{ij}}}{n!} \frac{n!}{y_{ij}!(n-y_{ij})!} p^{y_{ij}}(1-p)^{n-y_{ij}} \tag{21}$$

$$= \frac{p^{y_{ij}} e^{-\lambda_{ij}}}{y_{ij}!} \sum_{n=y_{ij}}^{\infty} \frac{\lambda_{ij}^n (1-p)^{n-y_{ij}}}{(n-y_{ij})!}. \tag{22}$$

Now, consider the following change of variables: $m_{ij} = n - y_{ij}$. Then, what we have is

$$\begin{aligned}
&\sum_{n=y_{ij}}^{\infty} \frac{\lambda_{ij}^n e^{-\lambda_{ij}}}{n!} \frac{n!}{y_{ij}!(n-y_{ij})!} p^{y_{ij}}(1-p)^{n-y_{ij}} \\
&= \frac{p^{y_{ij}} e^{-\lambda_{ij}}}{y_{ij}!} \sum_{n=y_{ij}}^{\infty} \frac{\lambda_{ij}^n (1-p)^{n-y_{ij}}}{(n-y_{ij})!} \\
&= \frac{p^{y_{ij}} e^{-\lambda_{ij}}}{y_{ij}!} \sum_{m_{ij}=0}^{\infty} \frac{\lambda_{ij}^{m_{ij}+y_{ij}} (1-p)^{m_{ij}}}{m_{ij}!} \\
&= \frac{(p\lambda_{ij})^{y_{ij}} e^{-\lambda_{ij}}}{y_{ij}!} \sum_{m_{ij}=0}^{\infty} \frac{\lambda_{ij}^{m_{ij}} (1-p)^{m_{ij}}}{m_{ij}!} \\
&= \frac{(p\lambda_{ij})^{y_{ij}} e^{-\lambda_{ij} p}}{y_{ij}!}
\end{aligned}$$

where the last equality is by the Taylor expansion.

APPENDIX B
PROOF OF PROPOSITION 1

In this section, we present the convergence proof of the proposed algorithm. Let us first introduce the *block successive upper bound minimization* (BSUM) [26], [28], which will be applied to our case.

Consider an optimization problem

$$\underset{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_n}{\text{minimize}}\ f(\boldsymbol{x}_1,\ldots,\boldsymbol{x}_n) \tag{23a}$$

$$\text{suject to } \boldsymbol{x}_1 \in X_1,\ldots,x_n \in X_n, \tag{23b}$$

where $\boldsymbol{x}_i$ denotes the ith block of the optimization variables and X_i denotes a convex closed set. The BSUM framework advocates the following algorithm to update $\boldsymbol{x}_i$ for $i = 1,\ldots,n$ cyclically:

$$\boldsymbol{x}_i^{t+1} \leftarrow \arg\min_{\boldsymbol{x}_i \in X_i}\ g_i(\boldsymbol{x}_i; \boldsymbol{x}_{-i}^t) \tag{24}$$

where

$$\boldsymbol{x}_{-i}^t = \left(\boldsymbol{x}_1^{t+1},\ldots,\boldsymbol{x}_{i-1}^{t+1},\boldsymbol{x}_{i+1}^t,\ldots,\boldsymbol{x}_n^t\right).$$

The BSUM framework bears a lot of resemblances to the Gauss-Seidel type *block coordinate descent* (BCD) scheme [16]. The key difference lies in the employment of (24): Assuming both f and g_i are continuously differentiable, the g_i function is an *optimization surrogate* that satisfies

$$g_i(\boldsymbol{x}_i; \boldsymbol{x}_{-i}^t) \geq f(\boldsymbol{x}_i; \boldsymbol{x}_{-i}^t),\ \forall \boldsymbol{x}_i \in X_i \tag{25}$$

$$g_i(\boldsymbol{x}_i^t; \boldsymbol{x}_{-i}^t) = f(\boldsymbol{x}_i^t; \boldsymbol{x}_{-i}^t), \tag{26}$$

$$\nabla_{\boldsymbol{x}_i} g_i(\boldsymbol{x}_i^t; \boldsymbol{x}_{-i}^t) = \nabla_{\boldsymbol{x}_i} f(\boldsymbol{x}_i^t; \boldsymbol{x}_{-i}^t). \tag{27}$$

Eq. (25) means that g_i is a blockwise tight upper bound of f. It was shown in [26] that the following holds:

Theorem 1 *Assume that $f(\cdot)$ is differentiable with respect to all $\boldsymbol{x}_i$ for $i = 1,\ldots,n$, and that the block optimization surrogate $g_i(\cdot)$ for all i satisfies* (25)*. Then, every limit point of the solution sequence $\{x^t\}$ produced by the BSUM algorithm is a stationary point of Problem* (23)*.*

The proposed algorithm admits three blocks, i.e., $\boldsymbol{U}$, $\boldsymbol{V}$, and $\boldsymbol{\alpha}$. let us denote $\boldsymbol{x}_1 = \mathrm{vec}(\boldsymbol{U})$, $\boldsymbol{x}_2 = \mathrm{vec}(\boldsymbol{V})$ and $\boldsymbol{x}_3 = \boldsymbol{\alpha}$. It is clear that $g_3(\cdot)$ satisfies (25), since

$$g_3(\boldsymbol{x}_3; \boldsymbol{x}_{-3}^t) = f(\boldsymbol{x}_3; \boldsymbol{x}_{-3}^t)$$

in our case. In addition, for $\boldsymbol{x}_1$, we notice that Eq. (25) also holds by the construction in Eq. (15a). A subtle point is the upper bound construction holds in Eq. (15a) only when there is no zero element in $\boldsymbol{U}$ and $\boldsymbol{V}$ — which is why Proposition 1 has this condition assumed. In practice, one normally has no control of intermediate iterates $\boldsymbol{U}^t$ and $\boldsymbol{V}^t$, and thus assuming this is considered relatively strong. One pragmatic remedy is to modify the update in (19) to

$$\boldsymbol{U} \leftarrow (\boldsymbol{U} \circ \boldsymbol{\Phi}) \,/\, \left(\tilde{\boldsymbol{U}} + \epsilon \mathbf{1}\mathbf{1}^\top\right),\ \boldsymbol{\Phi} = \left(\boldsymbol{Y} / \boldsymbol{U}\boldsymbol{V}^\top\right)\boldsymbol{V}$$

$$\boldsymbol{V} \leftarrow (\boldsymbol{V} \circ \boldsymbol{\Psi}) \,/\, \left(\tilde{\boldsymbol{V}} + \epsilon \mathbf{1}\mathbf{1}^\top\right),\ \boldsymbol{\Psi} = \left(\boldsymbol{Y}^\top / \boldsymbol{V}\boldsymbol{U}^\top\right)\boldsymbol{U},$$

where $\mathbf{1}\mathbf{1}^\top$ is an all-one matrix with proper size, and $\epsilon > 0$ is small number. With this modification, $\boldsymbol{U}^t$ and $\boldsymbol{V}^t$ are always positive. This trick has been found effective for stabilizing such multiplicative updates.