2010 IEEE International Conference on Data Mining

A Log-Linear Model with Latent Features for Dyadic Prediction

Aditya Krishna Menon
Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093-0404
akmenon@cs.ucsd.edu

Charles Elkan
Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093-0404
elkan@cs.ucsd.edu

***Abstract*—In dyadic prediction, labels must be predicted for pairs (dyads) whose members possess unique identifiers and, sometimes, additional features called side-information. Special cases of this problem include collaborative filtering and link prediction. We present a new log-linear model for dyadic prediction that is the first to satisfy several important desiderata: (i) labels may be ordinal or nominal, (ii) side-information can be easily exploited if present, (iii) with or without side-information, latent features are inferred for dyad members, (iv) the model is resistant to sample-selection bias, (v) it can learn well-calibrated probabilities, and (vi) it can scale to large datasets. To our knowledge, no existing method satisfies all the above criteria. In particular, many methods assume that the labels are binary or numerical, and cannot use side-information. Experimental results show that the new method is competitive with previous specialized methods for collaborative filtering and link prediction. Other experimental results demonstrate that the new method succeeds for dyadic prediction tasks where previous methods cannot be used. In particular, the new method predicts nominal labels accurately, and by using side-information it solves the cold-start problem in collaborative filtering.**

***Keywords*-Dyadic prediction; collaborative filtering; link prediction; log-linear model.**

I. Introduction

In dyadic prediction, the training set consists of pairs of objects $\{(r_i, c_i)\}_{i=1}^n$, called *dyads*, with associated labels $\{y_i\}_{i=1}^n$. The task is to predict labels for unobserved dyads, that is for pairs (r', c') that do not appear in the training set. In a well-studied special case of dyadic prediction, the dyads are (user, item) pairs, and the labels are a user's numeric rating of an item, often on a scale from 1 to 5. The task of predicting the ratings of unobserved (user, item) pairs is *collaborative filtering*, where the ultimate goal is to recommend to users new items they might like.

Existing methods for dyadic prediction are limited in several ways. One common issue is that they assume the observed labels have a *numerical* scale. This assumption is reflected in the loss function used for training, often mean absolute error or mean square error. But in many applications of dyadic prediction, the outcomes are *nominal*, that is discrete and unordered. For example, an online store may have information about customers' interactions with products, with possible outcomes including {viewed, purchased, returned}. On such datasets, imposing a numerical or ordinal structure is inappropriate. A dyadic prediction model should be flexible enough to handle nominal labels.

Another issue is that many models cannot exploit additional attributes of dyad members, called *side-information* or covariates. A model that only uses members' unique identifiers is limited in the *cold-start* setting, where the test set contains a dyad (r', c') where at least one of r' or c' is not present in any training set dyad. An example of this setting is predicting how a user will rate a movie that is yet to be released, and thus has no existing ratings.

In this paper, we propose a new model that addresses the above issues, and more. An appealing property of the new model is its conceptual and technical simplicity: it is a log-linear model where the log-odds are approximated by a low-rank matrix. Based on this, the new model is named LFL for *latent feature log-linear*.

II. Existing methods and their limitations

In this section, we relate the dyadic prediction task to matrix completion. We discuss six important desirable properties for a dyadic prediction model, and explain how no existing method possesses all these properties.

A. The dyadic prediction problem

In dyadic prediction, the training set is $\{((r_i, c_i), y_i)\}_{i=1}^n$, where the pairs (r_i, c_i) are called dyads. Usually, but not necessarily, the only information available about each element in each pair is a unique identifier. the goal is to predict the label y for an unobserved dyad (r', c'). We can interpret this task as a form of *matrix completion* by associating the training set with a matrix $M \in \mathcal{Y}^{|\mathcal{R}| \times |\mathcal{C}|}$, where $r_i \in \mathcal{R}, c_i \in \mathcal{C}, y_i \in \mathcal{Y}$. Each row of M is associated with some $r \in \mathcal{R}$ and each column with some $c \in \mathcal{C}$, so the training data is a subset of observed entries in M. The task is to fill in the missing entries of M. Based on this interpretation, we refer to r_i as a *row object* and c_j as a *column object*.

The dyadic prediction framework encompasses many real-world problems where the input is naturally modeled as interactions between entities. Important examples include recommending items to users, predicting links in a social

1550-4786/10 $26.00 © 2010 IEEE
DOI 10.1109/ICDM.2010.148

network, predicting whether users will click on ads, and predicting the contamination in food production plants [1]. In the absence of side-information, a powerful approach to the problem is based on learning *latent features*. Ignoring missing entries for the moment, and supposing that $\mathcal{Y} = \mathbb{R}$, the classical way to learn latent features from M is the singular value decomposition (SVD) [2]. This expresses M as the product of matrices U and V, which are representations of the row and column objects in a latent feature space. In the case of collaborative filtering, the latent space can be interpreted as scores for user and item characteristics (e.g. whether the item is a status symbol, whether the user likes foreign films). In matrix completion we do not know the entire matrix M, which means the SVD cannot be applied as-is. In this setting we can restate the task as finding a low-rank approximation that agrees with M only on its observed entries; the hope is that with suitable priors on U, V, this approximation will generalize to the missing entries as well. Such latent feature approaches have been very successful in real-world applications of dyadic prediction, in particular collaborative filtering [3].

B. Desirable properties of a dyadic prediction model

As mentioned in the introduction, there are several desiderata for a dyadic prediction model:

Agnostic to nature of labels. Labels in a dyadic prediction task may be ordinal, for example star ratings for a (user, movie) pair, or nominal, for example one of {viewed, purchased, returned} for a (user, product) pair. Ideally, a model should handle both types of labels.

Exploiting side-information. If a model cannot use side-information, when it is provided, then the model is severely limited in the aforementioned cold-start setting.

Learning latent features. It is desirable for a model to induce latent features for dyad elements just from the identities of row/column objects; otherwise, we are severely limited when we have no side-information. Aside from endowing the model with stronger predictive power, it also helps one understand the data better. (See the experimental results in Section V-D, for example, where we use the latent features to cluster the data.)

Resisting sample selection bias. Sample selection bias is the situation where the training and test sets follow different distributions. This is manifested in movie rating prediction for example, where users are generally more likely to provide ratings for movies that they like. Sample selection bias poses a challenge for generative models, because they need to model the joint distribution of examples and labels. A discriminative model, in contrast, only focuses on conditional probabilities of labels, and so is inherently robust against sample selection bias [4].

Calibration of probabilities. Often, a classifier is only a sub-model in a larger framework. In this setting, having the classifier output well-calibrated probabilities [5] rather than just ranking scores helps one make decision-theoretically optimal choices. For example, in online advertising, publishers would like to display ads that bring about the maximum expected revenue. To estimate this revenue, one needs accurate estimates of the probability that a user will click on an ad [6].

Fast training. Large datasets are common for practical applications of dyadic prediction such as collaborative filtering, so scalable training algorithms are essential.

To the best of our knowledge, the log-linear model proposed in this paper, which we call LFL, is the first method to meet all the criteria above. We now briefly discuss existing methods for dyadic prediction, pointing out which issues they fail to address.

C. Existing dyadic prediction methods

We focus here on models for two well-studied special cases of dyadic prediction, namely collaborative filtering and link prediction. Note that the focus of this paper is on models for the general dyadic prediction task; as such, we are not interested in tuning a model to the specifics of a collaborative filtering problem, which is the focus of many published results in the literature. Nonetheless, it is conceptually simple to incorporate many of the tricks for collaborative filtering into the LFL model. This is discussed further in Section IV-A.

Matrix factorization. The idea of learning latent features through matrix factorization has been successful in collaborative filtering. A popular example of this is maximum margin matrix factorization (MMMF) [7], where the idea is to approximate the input matrix M (containing missing entries) by a matrix X whose complexity is controlled by a convex approximation to the rank:

$$\min_X \sum_{i,j \in I} \ell(X_{ij}, M_{ij}) + \lambda ||X||_\Sigma$$

where I is the set of observed entries in M, ℓ is a modified hinge-loss, and $||\cdot||_\Sigma$ is the trace norm of a matrix (the sum of its singular values, also known as the nuclear norm). Penalizing the trace norm favors matrices X that are explained by a few latent factors i.e. if $X = U^T V$ for rank k matrices U, V, then $||U||_F^2 + ||V||_F^2$ is small. The above formulation requires solving an SDP (semidefinite program) and so is not scalable. A faster alternative is to learn directly U, V with a gradient-based method [15], although this sacrifices convexity. Probabilistic extensions of MMMF such as [8], [16] give a Bayesian treatment of the problem, and obtain higher accuracy. These were further extended in [10], which interprets the problem in a Gaussian process framework to learn a *nonlinear* matrix factorization. A related result is [9], where the focus is on a nonparametric model with the number of latent factors k determined automatically.

As noted earlier, a significant amount of research has focused on improving the performance of matrix factorization

Table I
SUMMARY OF VARIOUS METHODS IN THE LITERATURE IN TERMS OF MEETING THE SIX DESIDERATA FROM SECTION II-B.

Method	**Reference**	Nominal labels?	Side-info?	Latent features?	Resists SS bias?	Calibrated probs?	Fast training?
MMMF	[7]	No	No	Yes	Yes	No	No
PMF	[8]	No	No	Yes	No	No	No
NPCA	[9]	No	No	Yes	No	No	No
GPMF	[10]	No	Yes	Yes	No	No	Yes
BRISMF	[11]	No	No	Yes	Yes	No	Yes
FactNgbr	[12]	No	No	Yes	Yes	No	Yes
RBM	[13]	No	No	Yes	No	No	No
IBP	[14]	No	Yes	Yes	Yes	Yes	No
LFL	This paper	Yes	Yes	Yes	Yes	Yes	Yes

methods for the collaborative filtering setting. One such line of work has studied how to combine *neighborhood models* with standard matrix factorization. A neighborhood model is based on the idea that similar users tend to rate movies similarly. These models are good at capturing local effects in collaborative filtering data, and have been shown to improve performance of standard matrix factorization models. Recent models based on this hybrid approach are [11] and [12]. The latter also gives a way to incorporate *implicit feedback* in collaborative filtering, where we exploit information about *which* movies a user rated, even if we do not know the actual rating. However, it does not address the issue of incorporating more general forms of side-information. We note that it is conceptually simple to use neighborhood information in the LFL model using techniques similar to these papers; we emphasize once more that our goal is to address a broader set of concerns than just improving accuracy of collaborative filtering methods.

All matrix factorization methods assume that the input labels are numerical. Most of them do not incorporate side-information, although recent exceptions are [10], [17]; to our knowledge the latter has not been tested extensively on a number of datasets.

Boltzmann machines. Restricted Boltzmann machines (RBMs) have enjoyed some success for collaborative filtering [13]. The probability model is *generative*: $p(x, y, h; w) \propto \exp(\Psi(x, y, h))$, where x, y are the inputs and labels, h a vector of binary-valued hidden units, and Ψ some linear function of its inputs. Since the RBM is a generative model, exact training is intractable; also, as mentioned earlier, this makes it susceptible to sample-selection bias. A discriminative form of the RBM was proposed in [18], but it has not been applied to collaborative filtering. Further, to our knowledge, RBM-based models have not been extended to incorporate side-information, and have not been applied to datasets with nominal labels.

Models that are mathematically similar to the discriminative RBM have been successfully applied to structured prediction problems [19]. Unlike our model, these methods are not based on matrix factorization, and are not obviously applicable to dyadic prediction tasks. A closer study of this point would be valuable future research.

Link prediction models. In link prediction, the input is the adjacency matrix M of a graph with some missing entries, which we want to fill in. This is a dyadic prediction problem where both objects in the dyad belong to the same space, e.g. people in a social network. Additional (optional) constraints are that the graph is undirected and unweighted i.e. M is binary and symmetric. Two link prediction models that are relevant for our work are [20] and [14]. [20] handles the case of binary M using logistic regression, where the log-odds are modeled by a low-rank matrix approximation. This is similar to the model we propose, but our training procedure is considerably simpler than the MCMC scheme used in the paper, and also our model addresses the general dyadic prediction task, with binary link prediction as a special case. [14] also uses logistic regression, but with two important distinctions from [20]: (i) the matrix decomposition involves binary matrices, indicating the presence of a particular latent feature, and (ii) the decomposition is nonparametric, so the number of latent factors need not be specified *a priori*. Training in this model is involved, and it is not clear that it scales to large datasets.

Summary. We close this section with Table I, which summarizes various existing methods in the literature in terms of whether they possess each of the six desiderata we listed in Section II-B. We see that the LFL model proposed in this paper is the first method that meets all six desiderata.

III. THE LATENT FEATURE LOG-LINEAR (LFL) MODEL

In this section, we describe a new log-linear model for dyadic prediction. We then extend the model by adding latent features, yielding the latent feature log-linear (LFL) model. We explain how to make predictions with and train the LFL model for both nominal and ordinal labels.

A. A simple log-linear model

Given an observation $x \in \mathcal{X}$ and label $y \in \mathcal{Y}$, a log-linear model represents the conditional distribution $p(y|x)$ via

$$p(y|x; w) \propto \exp\left[\sum_i w_i f_i(x, y)\right].$$

Here, w is a vector of real-valued weights to be learned. The functions $f_i : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ are called *feature functions*, and measure interactions between the inputs and labels.

Recall that each example in a dyadic prediction task is $((r, c), y)$, where (r, c) is the dyad and y the label. We define $x = (r, c)$, and use $r(x)$ and $c(x)$ to denote the respective elements in x. Suppose there is no side-information; then, r, c are just indices into sets $\mathcal{R}, \mathcal{C}$ denoting the space of row and column objects respectively. We apply a log-linear model for $p(y|x; w)$ with three sets of feature functions:

- $(\forall r \in \mathcal{R},\, y' \in \mathcal{Y})\; f^{(1)}_{ry'}(x, y) = \mathbf{1}[r(x) = r, y = y']$
- $(\forall c \in \mathcal{C}, y' \in \mathcal{Y})\; f^{(2)}_{cy'}(x, y) = \mathbf{1}[c(x) = c, y = y']$
- $(\forall y' \in \mathcal{Y})\; f^{(3)}_{y'}(x, y) = \mathbf{1}[y = y']$

where $\mathbf{1}[\cdot]$ denotes an indicator function. We can think of each feature function as inducing a weight for each object-label pair. Specifically, split the weight vector into three components: $\alpha \in \mathbb{R}^{|\mathcal{R}|\times|\mathcal{Y}|}$, $\beta \in \mathbb{R}^{|\mathcal{C}|\times|\mathcal{Y}|}$ and $\gamma \in \mathbb{R}^{|\mathcal{Y}|}$. Then, the corresponding log-linear model is

$$p(y|x; w) \propto \exp\left[\alpha_{r(x)y} + \beta_{c(x)y} + \gamma_y\right] \tag{1}$$

This model is conceptually simple, but limited in expressiveness: specifically, it only learns *propensities* of the row and column objects towards a particular outcome. To see this limitation, fix a row object r_1 and consider dyads of the form (r_1, c). From Equation 1, for some fixed outcome y, the ranking of all c's in decreasing order of $p(y|(r_1, c); w)$ depends *only* on β_{cy}. That means we get the exact same ranking of c's for a different row object r_2.

B. A richer latent feature model

To capture interactions between row and column objects, we propose the following model:

$$p(y|x; w) \propto \exp\left(\sum_{k'=1}^{k} \alpha^y_{r(x)k'} \beta^y_{c(x)k'}\right) \tag{2}$$

where k is a constant that is the number of *latent factors* learned from the training data. (The γ_y term from the previous model is dealt with in Section IV-A.) For each outcome y, α^y is a matrix whose entries represent weights for each row object and one of k latent features, and similarly for β^y. So, we are learning tensors $\alpha \in \mathbb{R}^{|\mathcal{Y}|\times|\mathcal{R}|\times k}$ and $\beta \in \mathbb{R}^{|\mathcal{Y}|\times|\mathcal{C}|\times k}$. It is easy to check that this model is not subject to the aforementioned ranking limitation.

Before discussing the intuition for these weights, we take care of a technical point to make further exposition clearer. In multinomial logistic regression, it is standard to fix one category to be the base, by fixing the weights for that category to be 0. This defines a scale for the other categories' weights, and thus improves the identifiability of the model. Here, we fix the first outcome to be the base category: if $\mathcal{Y} = \{y_0, y_1, \ldots\}$, then we make $\alpha^{y_0}, \beta^{y_0} \equiv 0$.

Now we see why we call α, β latent weights. To do this, focus on the case $|\mathcal{Y}| = 2$ i.e. a logistic regression model. Let the outcomes be $y = 1$ and $y = 0$ without loss of generality. Following the above discussion, let $y = 0$ be the base class. In a slight abuse of notation, let α, β denote the matrices α^1, β^1. Then, in the LFL model the log-odds are

$$\log \frac{p(y = 1|x; w)}{p(y = 0|x; w)} = \alpha^T_{r(x)} \beta_{c(x)}$$

where $\alpha_{r(x)}$ denotes row $r(x)$ of matrix α, and similarly for $\beta_{c(x)}$. Defining a matrix P with entries $P_{rc} = p(y = 1|(r, c); w)$ gives the decomposition $P = \sigma(\alpha^T \beta)$, where $\sigma(\cdot)$ is the sigmoid function applied elementwise. That is, we model the log-odds by a rank k matrix approximation, where each dimension for α, β conceptually corresponds to a latent feature. As a result, we call this model the *latent feature log-linear model* or *LFL*. When $|\mathcal{Y}| > 2$, for the base class $y = y_0$, we model the pairwise log-odds by a low-rank approximation:

$$\log \frac{p(y|x; w)}{p(y_0|x; w)} = (\alpha^y_{r(x)})^T \beta^y_{c(x)}.$$

Now suppose that we have side-information in the form of a vector $s(x)$ for a dyad x. Then, we augment the low-rank approximation with a linear discriminator, yielding

$$p(y|x; w) \propto \exp\left((\alpha^y_{r(x)})^T \beta^y_{c(x)} + (w^y_s)^T s(x)\right).$$

Here, w_s represents the weights used for side-information. The model can use either *or* both of latent features and side-information to make predictions. The idea of extending multinomial logistic regression to incorporate information beyond the linear discriminator $w^T x$ has been studied in *random effects* models in statistics, e.g. [21]. Ours is the first contribution that applies a low-rank approximation of the log-odds for dyadic prediction, with the explicit aim of targeting both nominal and ordinal prediction tasks.

To complete the specification of the model, we now address two important issues: how to make predictions using the model, and how to train the model. The answers to both these issues depend on the nature of the labels.

C. Making predictions

Let $F(x; w)$ denote the model's prediction for the dyad x. A sensible choice for $F(x; w)$ depends on the whether the outcomes in $\mathcal{Y}$ are nominal or ordinal. The mode $\operatorname{argmax}_y p(y|x; w)$ is the most reasonable choice when the outcomes are nominal. The median $F(x; w) = \text{median}(p(y|x; w))$ is sensible when the labels are ordinal. If we further assume that the labels have a numeric structure, then the mean $\mathbb{E}[y] = \sum_y y p(y|x; w)$ is a sensible alternative to the median.

D. Objective function for training

When the y values are nominal, a sensible objective function is the conditional log-likelihood (CLL) of the data.

Since we are learning a large number of weights, it is important to regularize the objective function with an ℓ_2 penalty. Thus, the objective function we minimize is regularized negative CLL, which for a training set $\{(x_i, y_i)\}_{i=1}^n$ is

$$f_{\text{nom}}(w) = \frac{\lambda}{2}||w||^2 - \sum_{i=1}^{n} \log p(y_i|x_i; w).$$

Since this objective function is differentiable, we can apply stochastic gradient descent (SGD) for training, which makes the model scalable to large datasets.

For numeric labels, one can train the model using f_{nom}, but this ignores the structure in the labels and reduces the model's predictive power. Instead, if we assume the labels have a numeric structure, it is more sensible to minimize the discrepancy between the true label and the model's prediction $F(x; w)$. A simple measure of discrepancy often used in collaborative filtering is the mean absolute error (MAE), where the error for predicting the true label y as $\hat{y}$ is $\ell(y, \hat{y}) = |y - \hat{y}|$. For this choice, the regularized objective function is

$$f_{\text{ord}}(w) = \frac{\lambda}{2}||w||^2 + \sum_{i=1}^{n} |y_i - F(x_i; w)|.$$

If we use MAE as the training objective, ideally we would like to use $F(x; w) = \text{median}(p(y|x; w))$ since the median is the solution to $\text{argmin}_t \sum_i |x_i - t|$. However, this makes the objective non-differentiable, which means we cannot train using SGD. The simplest recourse here is to instead use $F(x; w) = \mathbb{E}[y]$, which is differentiable.

Note that MAE can be replaced with other loss functions, such as MSE (mean squared error, defined as $\ell(y, \hat{y}) = (y - \hat{y})^2$), or even the modified hinge loss of MMMF [7]; this does not change the underlying model, only the objective function. The choice of loss function is determined by two factors, which can be at odds with each other. One point is that it makes sense to use the same loss for training as for testing: if test error is measured by MAE, then it is sensible to optimize MAE during training. The second point is that we would like to return well-calibrated probabilities. If we train using a *proper loss* function, then it is guaranteed that we obtain well-calibrated probability estimates [22]. Examples of proper losses include the previously discussed CLL and MSE. MAE is not a proper loss function, so if we optimize it on the training set, we need a post-processing step to return calibrated probability estimates, such as isotonic regression [23]. Therefore, the choice of loss for ordinal labels potentially requires making a trade-off between directly optimizing the test set objective and automatically generating calibrated probabilities.

There are other options when modeling ordinal data with a log-linear or logistic regression model, but these require modifying the underlying probability distribution [24], [25]. the focus here is on using the same model as the nominal case, but modifying the training procedure to exploit the ordinal (or numeric) structure of labels.

E. Strengths and weaknesses of the LFL model

The LFL model meets all the goals for dyadic prediction listed in Section II-B. First, it can handle both nominal and ordinal $\mathcal{Y}$, by changing the training objective. Second, it is easy for the model to handle side-information when present, which helps address the cold-start problem. Crucially, the model does not assume that *only* side-information or only unique row and column object identifiers are relevant: it can make use of just one, or both of these pieces of information. Third, by virtue of being a discriminative probabilistic model, it can produce well-calibrated probabilities and is resistant to sample-selection bias. Finally, the objective function is differentiable, so one can use SGD for training, which scales to large datasets.

However, unlike for the basic log-linear model, the LFL objective function is not convex; it is only convex in α with β fixed, and vice versa. (The same is true of all existing matrix factorization methods for collaborative filtering.) Moreover, the experimental results below demonstrate that it is easy to find good local optima for the LFL model.

Another observation is that a separate set of weights is learned for each $y \in \mathcal{Y}$. If $|\mathcal{Y}|$ is large, then we will be learning many parameters, which increases the risk of overfitting. In nominal settings where there is truly no order to the various outcomes, then it is sensible to have separate weights for each outcome. This is plausible even in the ordinal setting: e.g., the characteristics that make a user rate a movie 1 star may be quite different from those that make the user rate it 5 stars. However, it is still desirable to exploit the ordinal structure to reduce the number of parameters. We address this issue in Section IV-C.

IV. Extensions to the LFL model

We now explain some important variations of the new latent feature model. Specifically, we discuss adding bias weights, regularization, and reducing the number of parameters in the ordinal setting. We then show how one can apply the model for the special case of link prediction.

A. Adding bias weights

In Equation 2, there is no explicit bias term γ_y. We include biases by forcing one component of each of α_r^y and β_c^y to have the constant value 1, for each y, r, c. This is equivalent to having a separate bias for each row and column object, and can be thought of as capturing baseline effects, e.g. whether a user tends to give high ratings, and whether a movie is popular. Note that in each vector α_r^y, two components are related to bias terms: one is the constant 1 and the other is the row bias. Therefore, henceforth, when we speak of a rank k latent feature model, we mean one

trained with k general parameters plus these additional two parameters.

The LFL model can incorporate other suggestions that have been used in standard matrix factorization models. For example, in the context of collaborative filtering, [8] suggests imposing a prior on a user's latent weight vector that takes into account the identity of the movies she has rated.

B. Regularization

Earlier, we proposed regularization that penalizes all components of w equally. But it is plausible that the row and column weights have different penalties, meaning separate regularization parameters $\lambda_\alpha, \lambda_\beta$. It is especially plausible that the weights for side-information be penalized differently. Also, as suggested in [26], it can be beneficial to apply the regularization inversely proportional to the (square root of the) number of observed entries for a particular row or column object, i.e. the number of times that row object r or column object c appears as part of a dyad in the training set. This ensures that the penalty for objects that appear only infrequently is larger than that for those that appear often, which is sensible because we expect to overfit more for objects for which we have only limited data. We applied all these extensions when training the LFL model.

C. Reducing parameters in the ordinal setting

The LFL model keeps a separate set of weights for each $y \in \mathcal{Y}$, but it is plausible that these weights share some structure in the ordinal setting. One way to reduce the number of weights is to have a low-rank approximation of the latent weights themselves. This is similar to the stereotype model for multinomial logistic regression [24], but we apply it to the latent weights rather than the discriminator w. Essentially, we assume that for every label y the weights can be decomposed as a linear combination of some set of base weights that are independent of y. Specifically, for any $y \in \mathcal{Y}$, we assume there are ρ scalars ϕ^i such that

$$(\alpha^y)^T \beta^y = \phi_y^1 (\tilde{\alpha}^1)^T \tilde{\beta}^1 + \ldots + \phi_y^\rho (\tilde{\alpha}^\rho)^T \tilde{\beta}^\rho.$$

Here, the weights $(\tilde{\alpha}^i, \tilde{\beta}^i)$ are shared among all outcomes y, and only the ϕ^i values vary. If $\rho \ll |\mathcal{Y}|$, this dramatically reduces the number of parameters to be learnt.

D. Application to link prediction

In standard link prediction the row and column objects belong to the same space. Consider the setting where the input graph is unweighted and undirected, so that the adjacency matrix is symmetric and binary. To apply the LFL model, we need to enforce symmetry: we need $p(y = 1|(r, c); w) = p(y = 1|(c, r); w)$. One way to achieve this is to maintain just one set of weights α, for both the row and column objects. Setting $y = 0$ to be the base class, we get the model

$$p(y = 1|x; w) \propto \exp(\alpha_{r(x)}^T \alpha_{c(x)}).$$

For a directed graph, it is no longer appropriate to have a symmetric decomposition inside the exponent. There are at least two options here. One is to keep three sets of weights: one for the incoming edges, α, one for the outgoing edges, β, and another for the global node information (i.e. weights that depend only on the identity of the node), γ. Then, we use the decomposition $\alpha^T \beta + \gamma^T \gamma$. The other approach is to use the decomposition $\alpha^T \Lambda \alpha$, where Λ is a full non-symmetric matrix [27].

E. Multi-relational data

In an important extension of the basic dyadic prediction task, there are a series of outcomes. For example, in link prediction, there may be different types of links: "is-family-member-of", "is-colleague-of", *etc*. This is not the same as having nominal links, because every dyad can have many links of different types, rather than just one link with many possible outcomes. It is akin to multi-label prediction rather than multi-class prediction. In this setting, we can extend the LFL model to capture the structure underlying the various relations. Suppose we have R binary relations, with outcomes $y^1, \ldots, y^R$. Then, for any $t \in \{1, \ldots, R\}$, we can model

$$p(y^t = 1|(r, c); w) = \sigma(\alpha_r^T \Lambda^t \beta_c).$$

That is, we share the row and column object weights among all relations, but we give a different scaling factor depending on the relation.

V. Experimental results

The experiments here demonstrate the flexibility of the LFL model, and its competitiveness with state-of-the-art methods on a range of different tasks. We present results for four important regimes: nominal data, ordinal collaborative filtering with and without side-information, and link prediction. In the process, we show that the LFL model scales to large datasets. We also present results on a matrix completion task involving handwritten digits. We emphasize that no existing dyadic prediction method is applicable for all these problems.

In all experiments, we pick the regularization parameters using three-fold cross-validation on the training set. We use different regularization for side-information and latent weights. We also report results for a varying number of latent features, k. In general increasing k increases accuracy, but there is a greater risk of overfitting. Accuracy measures are chosen to be sensible for each of the regimes listed above. In particular, we report MAE for ordinal prediction tasks, and AUC for link prediction tasks. We report the mean and standard deviation of test set performance from five runs of the training algorithm, where each run uses a different initialization of the weight vector. Different initializations lead to local optima of differing quality, since the optimization problem is nonconvex.

For the medium-sized datasets, we train using LBFGS [28]. Unlike first-order gradient methods, this does not require tuning of a learning rate, but at the price of higher computational cost. To demonstrate that the LFL model is scalable, results on large datasets are with stochastic gradient descent (SGD) as the training optimizer.

For the collaborative filtering tasks, we compare the LFL method to MMMF, which is representative of most matrix factorization methods. We use the MATLAB code for this provided by the author at http://people.csail.mit.edu/jrennie/matlab/. Note that this code does not include the suggestions in [26]. The results reported for all other methods are the ones that appear in previously published experiments. All experiments were run in MATLAB 2008b on a 2.67GHz Core i7 machine with 8 GB of RAM.

A. Results on synthetic nominal data

We run experiments on a synthetic dataset to check that we can learn from nominal data, and that it is possible to find good local optima of the objective function despite its non-convexity. Another question is how well a method for ordinal labels like MMMF performs on this dataset. Intuitively, because such a method imposes an artificial structure on the outcomes, it will be difficult to learn a good model.

We constructed a matrix $M \in \mathbb{R}^{n \times n}$ with entries in $\{1, 2, 3\}$. These can be thought of as indices into a nominal set such as {viewed, purchased, returned}; the numeric encoding is just for convenience. We picked the entries of M by sampling $M_{rc} \sim p(y|r, c) \propto \exp((\alpha_r^y)^T \beta_c^y)$, where there are $k = 5$ latent factors for the α and β matrices. The entries of α, β were drawn uniformly at random from the interval $[-3, 3]$. We set some fraction of entries to be unobserved, which were used for testing, and let the remaining entries form the training set. The goal of training is to choose α, β that maximize the log-likelihood. Two parameters that influence the quality of the learned model are the size of the matrix, n, and the retention rate, that is the fraction of entries kept for training.

The simplest measure of quality of a model is 0-1 accuracy. But this is meaningful only given a base accuracy to measure against. Since we know the underlying probability distribution $p(y|x; w)$, we can compute the Bayes error for a single matrix entry as $1 - \max_y p(y|r, c)$. A good model should approach the mean Bayes error.

Results are presented in Table II. For varying choices of n and the retention rate, the LFL model has high 0-1 accuracy. The error rate is closest to the Bayes error when the training set is large: this is intuitive, because we expect the learning task to be simpler with more samples. Another promising result is that the accuracy of the LFL model increases with the size of the training data, *despite* the increase in missing data. As expected, MMMF does significantly worse than the LFL model, demonstrating that an ordinal encoding is not sensible for this task.

Table II
0-1 ERROR OF RANK 5 MODELS ON SYNTHETIC NOMINAL DATASET.

n	Bayes	Retention	LFL	MMMF
500	4.8%	80%	7.7%	52.2%
		50%	8.2%	53.0%
		25%	12.0%	55.3%
1000	4.8%	80%	5.9%	48.9%
		50%	6.3%	53.2%
		25%	8.1%	54.4%
1500	4.8%	80%	5.5%	47.2%
		50%	5.9%	47.3%
		25%	7.0%	48.0%

B. Results on ordinal datasets

An important question is how the LFL model compares with existing methods for the ordinal setting. We focus on the canonical case of this problem, collaborative filtering. We emphasize that the goal is not to be the single best method for a collaborative filtering task, but rather to be competitive with existing methods while being more general than them.

Results on `bookcrossing`. The `bookcrossing` dataset-consists of $1,149,780$ ratings given by $278,858$ users for $271,379$ books [29]. Following [30], we pre-process the dataset to remove all ratings with the value 0, users with fewer than 3 ratings, and books with fewer than 6 ratings. This leaves $342,464$ ratings over $35,689$ users and $138,660$ books. Unlike [30], we did not use just the books with Amazon reviews; we also did not use any side-information, and only learned latent features.

Given the size of the dataset, we trained using SGD. We could not run the MMMF code, as it required too much memory. We performed three-fold cross-validation where each fold contained $1/3$ of each user's ratings. The rank 5 LFL model attains an MAE of 1.0580 ± 0.0028, which is only slightly worse than the reported MAE for the fLDA method (1.0317) proposed in [30]. fLDA only uses side-information for making predictions: this information is hard to duplicate as it involves mining Amazon customer reviews. It is possible that the LFL model would achieve higher accuracy with this extra side-information. The training time of fLDA is not reported in [30]; the LFL method processes the large dataset efficiently, and trains in 10 minutes.

Results on $1M$ `movielens`. The 1M `movielens` dataset consists of $1,000,209$ ratings for 6040 users and 3900 movies. Following [7], we randomly selected 5000 users and for each user, picked a random rating and put it in the test set. The other ratings are the training set. We ended up with a training set of $836,865$ examples, each a user-movie rating dyad, and 5000 test examples. We trained the LFL model using SGD on this dataset. Table III presents the MAE and training time for the LFL model and MMMF for various choices of latent factors k. We see that the LFL model trains much faster than MMMF, while achieving competitive accuracy. In particular, LFL results for ranks 1 and 5 are statistically indistinguishable from MMMF results.

Table III
RESULTS ON THE $1M$ MOVIELENS DATASET.

Method	k	Test set MAE	Train time
MMMF	1	0.6557 ± 0.0000	35 mins
LFL	1	$\mathbf{0.6541 \pm 0.0041}$	7 mins
MMMF	2	$\mathbf{0.6397 \pm 0.0009}$	40 mins
LFL	2	0.6446 ± 0.0035	7 mins
MMMF	5	$\mathbf{0.6203 \pm 0.0001}$	72 mins
LFL	5	0.6215 ± 0.0059	14 mins

C. Results in the cold-start setting

Here, we present results showing that the LFL model is able to learn to use side-information, and make useful predictions in the cold-start setting. We took the 100K `movielens` dataset, and randomly discarded 50 users from the training set. These people are cold-start users when they appear in the test. In the experiments, we considered the following scenarios: (i) the standard setting, where there are no cold-start users or movies, (ii) there are cold-start users, but movies are known, (iii) full cold-start, where both users and movies are unobserved. For (ii), we took the test set for (i) and then discarded all movies that occur in it from the training set. The side-information we used was the user's age, gender and occupation, and the movie's genre.

We compare to a baseline method of predicting the mean from a latent feature model trained without side-information. That is, for the dyad (u, m) where u was not present during training, the prediction is $\frac{1}{n_m} \sum_{(u',m)\in\mathcal{T}} F(u', m)$, where n_m is the number of users in the training set $\mathcal{T}$ who have rated m, and $F(u', m)$ is the standard latent feature based prediction for dyad (u', m). When both u, m are not present in training, we use the mean predicted rating for the entire training set.

When training with side-information, the following heuristic helps in avoiding bad local optima: first train the model without any side-information to learn latent weights. Then, use this as initialization to the model with side-information weights included, that is initialize the latent weights for the second model to those learned by the first one. For the second optimization, we obtained better results when the latent weights were frozen. This is a form of block coordinate descent, where we optimize over two groups of variables by optimizing over each one in turn.

All models use rank $k = 5$ and are optimized with LBFGS. Table IV summarizes the results. Learning with side-information significantly improves accuracy in the cold-start setting. Also, with side-information we do almost as well in the cold-start and standard settings; by contrast, the standard latent feature model does much worse in the cold-start setting. Comfortingly, side-information gives better MAE than the basic model when tested in the standard setting, i.e. no cold-start users or movies. This means that, as expected, taking side-information into account can give slightly better predictions than just learning latent features.

Table IV
COLD-START RESULTS, RANK 5 MODEL.

Setting	Baseline MAE	Side-info MAE
(i) Standard	0.7162 ± 0.0059	$\mathbf{0.7063 \pm 0.0000}$
(ii) Cold-start	0.8039 ± 0.0000	$\mathbf{0.7118 \pm 0.0208}$
(iii) Full cold-start	0.9608 ± 0.0000	$\mathbf{0.7451 \pm 0.0196}$

D. Results on link prediction tasks

We use two datasets for the link prediction experiments. The `coauthor` dataset [31] is a binary matrix indicating whether two authors have written a paper for NIPS together. The `alyawarra` dataset [32] contains kinship relations between people of the Alyawarra tribe in Australia. For both datasets, we use a random 80-20 train-test set split: this means that for training, we assume that 20% of entries in the matrix are missing, and we predict these at test time. We report area under the ROC curve (AUC) as the performance measure, since this is commonly used in the link prediction literature. Because both datasets are symmetric, we use the $\alpha^T\alpha$ decomposition discussed in Section IV-D.

Results on `coauthor`. Following [14], we focus on the 234×234 submatrix of authors who collaborated with the most number of people. We compare our results to those given in [14] for an Indian Buffet Process (IBP) model, the infinite relational model (IRM), and the mixed membership stochastic block model (MMSB). We also compare the LFL method to MMMF, which is not explicitly designed for link prediction–it does not exploit the fact that the row and column spaces are the same–but helps gauge the improvement that methods designed for link prediction provide over collaborative filtering methods.

Results are shown in Table V. The MMMF method is outperformed by all other methods, which suggests that exploiting the structure in link prediction tasks is essential to achieve good performance. (MMMF gives slightly worse results, not shown, for ranks over 5, perhaps due to overfitting.) For all ranks, the LFL model is more accurate than the MMSB and IRM models, but the IBP method is slightly better overall. It is important to note that the training/test split often has a significant impact on the accuracy of a learned model. We do not know the dataset splits used in [14] for the MMSB, IRM, or IBP methods, so this issue needs to be kept in mind when interpreting the results.

Table V
AUC OF ALTERNATIVE METHODS ON COAUTHOR DATASET.

Method	k	Test set AUC
MMSB	Unknown	0.8705
IRM	Unknown	0.8906
IBP	20	**0.9509**
MMMF	5	0.8193 ± 0.0132
LFL	5	0.9235 ± 0.0049
LFL	10	0.9290 ± 0.0165
LFL	20	0.9424 ± 0.0093

The learned weights for each author are useful for clustering. We applied k-means clustering on the user weights with 7 clusters, and sorted the authors according to the learned clusters. Figure 1 shows the resulting coauthor graph. Clearly the user weights have identified significant cliques in the coauthor graph.

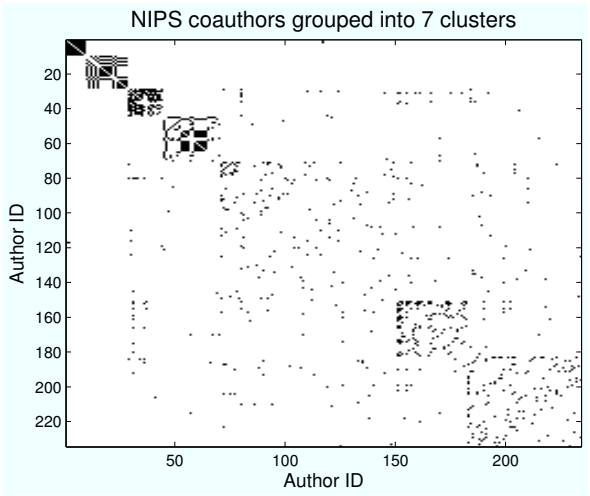

Figure 1. `coauthor` after clustering model output. Black/white entries indicate the presence/absence of a link.

Results on `alyawarra`. It is not sensible to run MMMF on this dataset, because the task is not binary, and it is not sensible to impose an ordinal scale on the matrix entries. We can view this dataset as *multi-relational*, so that each possible kinship relation defines a separate matrix, or as multi-category, where the number of classes is the number of kinship relations. We chose the latter because there is only one kinship relation observed between any two people. The reported results on this dataset [14] use the multi-relational viewpoint. With this choice, each relation can have separate weights (a "single" model), or shared weights (a "global" model). We ran our method for a number of ranks k, with log-likelihood as the objective function. The results in Table VI show that the rank 20 LFL model gives the best results of all methods.

Table VI
AUC OF ALTERNATIVE METHODS ON `ALYAWARRA` DATASET.

Method	k	Test set AUC
MMSB global	Unknown	0.8943
IRM global	Unknown	0.9143
IBP global	9	0.9183
MMSB single	Unknown	0.9005
IRM single	Unknown	0.9310
IBP single	9	0.9443
LFL	5	0.9390 ± 0.0006
LFL	10	0.9469 ± 0.0013
LFL	20	$\mathbf{0.9475 \pm 0.0005}$

E. Results on matrix completion task

We ran an experiment on the `usps` dataset of handwritten digits, following [33]. We pick 100 random images each of digits $1, 2$ or 3. The images are binarized so that the pixel values are ± 1. For 16 of these images, we occlude their bottom half by setting the pixel values to be "missing." The goal is to fill in the missing entries, or equivalently, to reconstruct the bottom half of the 16 partial images. This is a dyadic prediction problem where each digit is a row object, and each pixel position is a column object. We applied the LFL model on this dataset with rank $k = 3$, optimizing MAE using LBFGS. Results are shown in Figure 2. The top row shows the original versions of the 16 occluded images, the middle row shows the data as presented to the training algorithm, and the last row shows the predictions made by the LFL model. (For this row, even the top half is the model prediction.) We see that the trained LFL model accurately reconstructs most of the images. One exception is an image whose true digit is 3, but which the LFL model reconstructs as 1; this behavior is understandable, because there is not enough information in the top half for even a human to see the right answer.

VI. CONCLUSION

Existing methods for dyadic prediction are limited in one or more ways, most commonly in that they assume the labels lie on a binary or numerical scale, and cannot exploit both dyad members' unique identifiers and side-information. An ideal model should overcome these limitations. Other desirable properties are resistance to sample-selection bias and the generation of well-calibrated probabilities. Last but not least, given the size of datasets in real-world dyadic prediction tasks like collaborative filtering, it is essential for a model to be scalable. In this paper, we have presented a latent feature log-linear model (LFL) that addresses all these issues. The new model learns latent features based on dyad identifiers, but can easily use side-information also when it is available. We can apply it to large datasets using stochastic gradient descent. It is a discriminative probabilistic method, and so has resistance to sample-selection bias and produces well-calibrated probabilities. Experiments show success in learning from both nominal and ordinal labels, and that the model can use dyad identifiers and side-information to achieve competitive accuracy to existing methods for collaborative filtering and link prediction.

REFERENCES

[1] A. D. Purnamrita Sarkar, Lujie Chen, "Dynamic network model for predicting occurrences of salmonella at food facilities," in *BioSecure 2008*, vol. LNCS 5354, 2008, pp. 56–63.
[2] A. K. Menon and C. Elkan, "Fast algorithms for approximating the singular value decomposition," To appear in ACM Transactions of Knowledge and Data Discovery, 2010.

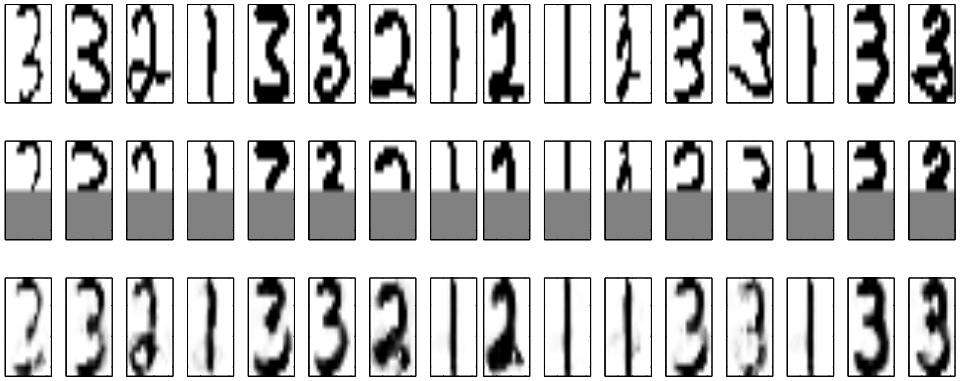

Figure 2. **Top**: Original images; **Middle**: Occluded images in training set; **Bottom**: Reconstruction using rank 3 model.

[3] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[4] A. T. Smith and C. Elkan, "Making generative classifiers robust to selection bias," in *KDD*, 2007.

[5] B. Zadrozny and C. Elkan, "Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers," in *ICML '01*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, pp. 609–616.

[6] M. Richardson, E. Dominowska, and R. Ragno, "Predicting clicks: Estimating the click-through rate for new ads," in *WWW '07*, 2007, pp. 521–530.

[7] N. Srebro, J. D. M. Rennie, and T. Jaakkola, "Maximum-margin matrix factorization," in *NIPS*, 2004.

[8] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," in *NIPS '07*, 2007.

[9] K. Yu, S. Zhu, J. Lafferty, and Y. Gong, "Fast nonparametric matrix factorization for large-scale collaborative filtering," in *SIGIR '09*. New York, NY, USA: ACM, 2009, pp. 211–218.

[10] N. D. Lawrence and R. Urtasun, "Non-linear matrix factorization with gaussian processes," in *Proceedings of the 26th Annual International Conference on Machine Learning*. New York, NY, USA: ACM, 2009, pp. 601–608.

[11] G. Takács, I. Pilászy, B. Németh, and D. Tikk, "Scalable collaborative filtering approaches for large recommender systems," *JMLR*, vol. 10, pp. 623–656, 2009.

[12] Y. Koren, "Factorization meets the neighborhood: A multifaceted collaborative filtering model," in *KDD '08*. New York, NY, USA: ACM, 2008, pp. 426–434.

[13] R. Salakhutdinov, A. Mnih, and G. Hinton, "Restricted Boltzmann machines for collaborative filtering," in *ICML '07*. New York, NY, USA: ACM, 2007, pp. 791–798.

[14] K. Miller, T. Griffiths, and M. Jordan, "Nonparametric latent feature models for link prediction," in *Advances in Neural Information Processing Systems 22*, 2009, pp. 1276–1284.

[15] J. D. M. Rennie and N. Srebro, "Fast maximum margin matrix factorization for collaborative prediction," in *ICML '05*.

[16] R. Salakhutdinov and A. Mnih, "Bayesian probabilistic matrix factorization using Markov chain Monte Carlo," in *ICML '08*. New York, NY, USA: ACM, 2008, pp. 880–887.

[17] J. Abernethy, F. Bach, T. Evgeniou, and J.-P. Vert, "A new approach to collaborative filtering: Operator estimation with spectral regularization," *JMLR*, vol. 10, pp. 803–826, 2009.

[18] H. Larochelle and Y. Bengio, "Classification using discriminative restricted Boltzmann machines," in *ICML '08*, 2008, pp. 536–543.

[19] L.-P. Morency, A. Quattoni, and T. Darrell, "Latent-dynamic discriminative models for continuous gesture recognition," in *CVPR*, 2007.

[20] P. D. Hoff, "Multiplicative latent factor models for description and prediction of social networks," Center for Statistics and the Social Sciences, University of Washington-Seattle, Working Paper no. 54, 2006.

[21] J. Hartzel, A. Agresti, and B. Caffo, "Multinomial logit random effects models," *Statistical Modelling*, vol. 1, pp. 81–102, 2001.

[22] M. DeGroot and S. Fienberg, "The comparison and evaluation of forecasters," *Statistician*, vol. 32, pp. 12–22, 1982.

[23] B. Zadrozny and C. Elkan, "Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers," in *ICML '01*, 2001, pp. 609–616.

[24] J. Anderson, "Regression and ordered categorical variables," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 46, pp. 1–30, 1984.

[25] J. D. M. Rennie, "Loss functions for preference levels: Regression with discrete ordered labels," in *Proceedings of the IJCAI Multidisciplinary Workshop on Advances in Preference Handling*, 2005, pp. 180–186.

[26] M. Weimer, A. Karatzoglou, and A. J. Smola, "Improving maximum margin matrix factorization," in *ECML/PKDD (1)*, 2008, p. 14.

[27] S. Zhu, K. Yu, Y. Chi, and Y. Gong, "Combining content and link for classification using matrix factorization," in *SIGIR '07*. New York, NY, USA: ACM, 2007, pp. 487–494.

[28] D. C. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization," *Math. Program.*, vol. 45, no. 3, pp. 503–528, 1989.

[29] C.-N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen, "Improving recommendation lists through topic diversification," in *WWW '05*, 2005.

[30] D. Agarwal and B.-C. Chen, "fLDA: Matrix factorization through latent Dirichlet allocation," in *WSDM '10*. New York, NY, USA: ACM, 2010, pp. 91–100.

[31] S. Roweis, "NIPS `coauthor` dataset," http://www.cs.nyu.edu/~roweis/data.html, 2002.

[32] W. W. Denham, "Alyawarra dataset," http://www1.aiatsis.gov.au/exhibitions/AlyaWeb/public/about.html, 1971.

[33] E. Meeds, Z. Ghahramani, R. Neal, and S. Roweis, "Modeling dyadic data with binary latent factors," in *NIPS*, 2006, pp. 977–984.