JMLR: Workshop and Conference Proceedings 25:1–16, 2012 Asian Conference on Machine Learning

Learning From Ordered Sets and Applications in Collaborative Ranking

Truyen Tran[†‡] TRUYEN.TRAN@DEAKIN.EDU.AU
Dinh Phung[†] DINH.PHUNG@DEAKIN.EDU.AU
Svetha Venkatesh[†] SVETHA.VENKATESH@DEAKIN.EDU.AU
[†]*Pattern Recognition and Data Analytics, Deakin University, Waurn Ponds, Vic 3216, Australia.*
[‡]*Department of Computing, Curtin University, Bentley, WA 6102, Australia*

Editor: Steven C.H. Hoi and Wray Buntine

Abstract

Ranking over sets arise when users choose between groups of items. For example, a group may be of those movies deemed 5 stars to them, or a customized tour package. It turns out, to model this data type properly, we need to investigate the general combinatorics problem of partitioning a set and ordering the subsets. Here we construct a probabilistic log-linear model over a set of ordered subsets. Inference in this combinatorial space is highly challenging: The space size approaches $(N!/2)6.93145^{N+1}$ as N approaches infinity. We propose a `split-and-merge` Metropolis-Hastings procedure that can explore the state-space efficiently. For discovering hidden aspects in the data, we enrich the model with latent binary variables so that the posteriors can be efficiently evaluated. Finally, we evaluate the proposed model on large-scale collaborative filtering tasks and demonstrate that it is competitive against state-of-the-art methods.

Keywords: Ordered sets, ranking with ties, split-merge, MCMC, latent models, Boltzmann machines, collaborative filtering

1. Introduction

Rank data has recently generated a considerable interest within the machine learning community, as evidenced in ranking labels (Dekel et al., 2003; Vembu and Gärtner, 2010) and ranking data instances (Cohen et al., 1999; Weimer et al., 2008). The problem is often cast as generating a list of objects (e.g., labels, documents) which are arranged in decreasing order of relevance with respect to some query (e.g., input features, keywords). The treatment effectively ignores the grouping property of compatible objects (Wagstaff et al., 2010). This phenomenon occurs when some objects are likely to be grouped with some others in certain ways. For example, a grocery basket is likely to contain a variety of goods which are complementary for household needs and at the same time, satisfy weekly budget constraints. Likewise, a set of movies are likely to given the same quality rating according to a particular user. In these situations, it is better to consider ranking groups instead of individual objects. It is beneficial not only when we need to recommend a subset (as in the case of grocery shopping), but also when we just want to produce a ranked list (as in the case of watching movies) because we would better exploit the compatibility among grouped items.

This poses a question of how to group individual objects into subsets given a list of all possible objects. Unlike the situation when the subsets are pre-defined and fixed (e.g., sport teams in a particular season), here we need to explore the space of set partitioning and ordering simultaneously. In the grocery example we need to partition the stocks in the store into baskets and then rank them with respect to their utilities; and in the movie rating example we group movies in the same quality-package and then rank these groups according to their given ratings. The situation is somewhat related to multilabel learning, where our goal is to produce a subset of labels out of many for a given input, but it is inherently more complicated: not only we need to produce all subsets, but also to rank them.

This paper introduces a probabilistic model for this type of situations, i.e., we want to learn the statistical patterns from which a set of objects is partitioned and ordered, and to compute the probability of any scheme of partitioning and ordering. In particular, the model imposes a log-linear distribution over the joint events of partitioning and ordering. It turns out, however, that the state-space is prohibitively large: If the space of complete ranking has the complexity of $N!$ for N objects, then the space of partitioning a set and ordering approaches $(N!/2)6.93145^{N+1}$ in size as N approaches infinity (Mureşan, 2008, pp. 396–397). Clearly, the latter grows much faster than the former by an exponential factor of 6.93145^{N+1}. To manage the exploration of this space, we design a `split-and-merge` Metropolis-Hastings procedure which iteratively visits all possible ways of partitioning and ordering. The procedure randomly alternates between the `split` move, where a subset is split into two consecutive parts, and the `merge` move, where two consecutive subsets are merged. The proposed model is termed Ordered Sets Model (OSM).

To discover hidden aspects in ordered sets (e.g., latent aspects that capture the taste of a user in his or her movie genre), we further introduce binary latent variables in a fashion similar to that of restricted Boltzmann machines (RBMs) (Smolensky, 1986). The posteriors of hidden units given the visible rank data can be used as a vectorial representation of the data - this can be handy in tasks such as computing distance measures or visualisation. This results in a new model called Latent OSM.

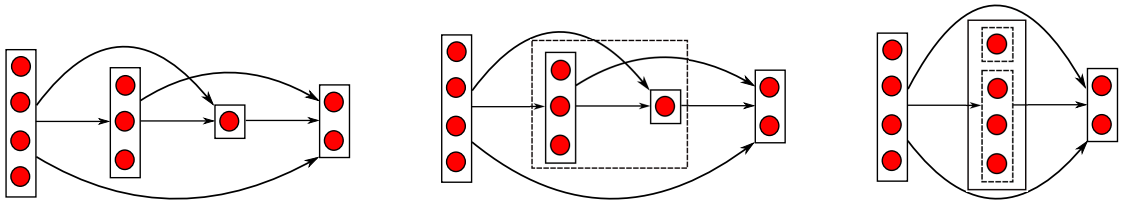
Finally, we show how the proposed Latent OSM can be applied for collaborative filtering, e.g., when we need to take seen grouped item ratings as input and produce a ranked list of unseen item for each user. We then demonstrate and evaluate our model on large-scale public datasets. The experiments show that our approach is competitive against several state-of-the-art methods.

The rest of the paper is organised as follows. Section 2 presents the log-linear model over ordered sets (OSM) together with our main contribution – the `split-and-merge` procedure. Section 3 introduces Latent OSM, which extends the OSM to incorporate latent variables in the form of a set of binary factors. An application of the proposed Latent OSM for collaborative filtering is described in Section 4. Related work is reviewed in the next section, followed by the conclusions.

2. Ordered Set Log-linear Models

2.1. General Description

We first present an intuitive description of the problem and our solutions in modelling, learning and inference. Fig. 1(a) depicts the problem of grouping items into subsets (repre-



(a) OSM (b) (c)

Figure 1: (a) Ordered Set Model; (b) the `split` operator; and (c) the `merge` operator. The figure in (c) represents the result of a merge of the middle two subsets in (b). Conversely, the (b) figure can be considered as result of a splitting the middle subset of the (c) figure. Arrows represent the preference orders, not the causality or conditioning.

sented by a box of circles) and ordering these subsets (represented by arrows which indicate the ordering directions). This looks like a high-order Markov chain in a standard setting, and thus it is tempting to impose a chain-based distribution. However, the difficulty is that the partitioning of set into subsets is also random, and thus a simple treatment is not applicable. Recently, Truyen et al (2011) describe a model in this direction with a careful treatment of the partitioning effect. However, their model does not allow fast inference since we need to take care of the high-order properties.

Our solution is as follows. To capture the grouping and relative ordering, we impose on each group a subset potential function capturing the relations among compatible elements, and on each pair of subsets a ordering potential function. The distribution over the space of grouping and ordering is defined using a log-linear model, where the product of all potentials accounts for the unnormalised probability. This log-linear parameterization allows flexible inference in the combinatorial space of all possible groupings and orderings.

In this paper inference is carried out in a MCMC manner. At each step, we randomly choose a `split` or a `merge` operator. The `split` operator takes a subset at random (e.g., Fig. 1(c)) and uniformly splits it into two smaller subsets. The order between these two smaller subset is also random, but their relative positions with respect to other subsets remain unchanged (e.g., Fig. 1(b)). The `merge` operator is the reverse (e.g., converting Fig. 1(b) into Fig. 1(c)). With an appropriate acceptance probability, this procedure is guaranteed to explore the entire combinatorial space.

Armed with this sampling procedure, learning can be carried out using stochastic gradient techniques (Younes, 1989).

2.2. Problem Description

Given two objects x_i and x_j, we use the notation $x_i \succ x_j$ to denote the expression of x_i is ranked higher than x_j, and $x_i \sim x_j$ to denote the between the two belongs to the same group. Furthermore, we use the notation of $X = \{x_1, x_2, \ldots, x_N\}$ as a collection of N objects. Assume that X is partitioned into T subsets $\{X_t\}_{t=1}^{T}$. However, unlike usual notion of partitioning of a set, we further posit an *order* among these subsets in which members of each subset presumably share the same rank. Therefore, our partitioning process is *order-*

sensitive instead of being exchangeable at the partition level. Specifically, we use the indices $1, 2, ..., T$ to denote the *decreasing* order in the rank of subsets. These notations allow us to write the collection of objects $X = \{x_1, \ldots, x_N\}$ as a union of T ordered subsets:[1]

$$X = X_1 \bigcup X_2 \ldots \bigcup X_T \tag{1}$$

where $\{X_t\}_{t=1}^{T}$ are non-empty subsets of objects so that $x_i \sim x_j, \forall x_i, x_j \in X_t \; i \neq j, \forall t$.

As a special case when $T = N$, we obtain an exhaustive ordering among objects wherein each subset has exactly one element and there is no grouping among objects. This special case is equivalent with a complete ranking scenario. To illustrate the complexity of the problem, let us characterise the state-space, or more precisely, the number of all possible ways of partitioning and ordering governed by the above definition. Recall that there are $s(N, T)$ ways to divide a set of N objects into T partitions, where $s(N, T)$ denotes the *Stirling numbers of second kind* (van Lint and Wilson, 1992, p. 105). Therefore, for each pair (N, T), there are $s(N, T)\, T!$ ways to perform the partitioning with ordering. Considering all the possible values of T give us the size of our model state-space:

$$\sum_{T=1}^{N} s(N, T)\, T! = \text{Fubini}(N) = \sum_{k=1}^{\infty} \frac{k^N}{2^{k+1}} \tag{2}$$

which is also known in combinatorics as the Fubini's number (Mureşan, 2008, pp. 396–397). This number grows super-exponentially and it is known that it approaches $N!/(2\,(\log 2)^{N+1})$ as $N \to \infty$ (Mureşan, 2008, pp. 396–397). Taking the logarithm, we get $\log N! - (N + 1) \log\log 2 - \log 2$. As $\log\log 2 < 0$, this clearly grows faster than $\log N!$, which is the log of the size of the standard complete permutations.

2.3. Model Specification

Denote by $\Phi(X_t) \in \mathbb{R}^+$ a positive potential function over a single subset[2] X_t and by $\Psi(X_t \succ X_{t'}) \in \mathbb{R}^+$a potential function over a ordered pair of subsets $(X_t, X_{t'})$ where $t < t'$. Our intention is to use $\Phi(X_t)$ to encode the compatibility among all member of X_t, and $\Psi(X_t \succ X_{t'})$ to encode the ordering properties between X_t and $X_{t'}$. We then impose a distribution over the collection of objects as:

$$P(X) \;=\; \frac{1}{Z}\Omega(X), \qquad \text{where } \Omega(X) = \prod_{t} \Phi(X_t) \prod_{t'>t} \Psi(X_t \succ X_{t'}) \tag{3}$$

and $Z = \sum_X \Omega(X)$ is the partition function. We further posit the following factorisation for the potential functions:

$$\Phi(X_t) \;=\; \prod_{i,j \in X_t | j>i} \varphi(x_i \sim x_j); \qquad \Psi(X_t \succ X_{t'}) = \prod_{i \in X_t} \prod_{j \in X_{t'}} \psi(x_i \succ x_j) \tag{4}$$

1. Alternatively, we could have proceeded from the permutation perspective to indicate the ordering of the subsets, but we simplify the notation here for clarity.
2. In this paper, we do not consider the case of empty sets, but it can be assumed that $\phi(\emptyset) = 1$.

where $\varphi(x_i \sim x_j) \in \mathbb{R}^+$ captures the effect of *grouping*, and $\psi(x_i \succ x_j) \in \mathbb{R}^+$ captures the relative ordering between objects x_i and x_j. Hereafter, we shall refer to this proposed model as the *Ordered Set Model* (OSM).

2.4. `Split-and-Merge` MCMC Inference

In order to evaluate $P(X)$ we need to sum over all possible configurations of X which is in the complexity of the Fubini(N) over the set of N objects (cf. Section 2.2, Eq. 2). We develop a Metropolis-Hastings (MH) procedure for sampling $P(X)$. Recall that the MH sampling involves a proposal distribution Q that allows drawing a new sample X' from the current state X with probability $Q(X'|X)$. The move is then accepted with probability

$$P_{\mathsf{accept}} = \min\{1, l \times p\}, \qquad \text{where } l = \frac{P(X')}{P(X)} = \frac{\Omega(X')}{\Omega(X)} \text{ and } p = \frac{Q(X|X')}{Q(X'|X)} \tag{5}$$

To evaluate the likelihood ratio l we use the model specification defined in Eq (3). We then need to compute the proposal probability ratio p. The key intuition is to design a random local move from X to X' that makes a relatively small change to the current partitioning and ordering. If the change is large, then the rejection rate is high, thus leading to high cost (typically the computational cost increases with the step size of the local moves). On the other hand, if the change is small, then the random walks will explore the state-space too slowly.

We propose two operators to enable the proposal move: the `split` operator takes a non-singleton subset X_t and randomly splits it into two sub-subsets $\{X_t^1, X_t^2\}$, where X_t^2 is inserted *right* next to X_t^1; and the `merge` operator takes two *consecutive* subsets $\{X_t, X_{t+1}\}$ and merges them. This dual procedure will guarantee exploration of all possible configurations of partitioning and ordering, given enough time (See Figure 1 for an illustration).

2.4.1. Split Operator

Assume that among the T subsets, there are T_{split} non-singleton subsets from which we randomly select one subset to split, and let this be X_t. Since we want the resulting sub-subsets to be non-empty, we first randomly draw two distinct objects from X_t and place them into the two subsets. Then, for each remaining object, there is an equal chance going to either X_t^1 or X_t^2. Let $N_t = |X_t|$, the probability of this drawing is $\left(N_t(N_t-1)2^{N_t-2}\right)^{-1}$. Since the probability that these two sub-subsets will be merged back is T^{-1}, the proposal probability ratio p_{split} can be computed as in Eq (6). Since our potential functions depend only on the relative orders between subsets and between objects in the same set, the likelihood ratio l_{split} due to the `split` operator does not depend on other subsets, it can be given as in Eq (7). This is because the members of X_t^1 are now ranked higher than those of X_t^2 while they are of the same rank previously.

$$p_{\mathsf{split}} = \frac{T_{\mathsf{split}} N_t (N_t - 1) 2^{N_t - 2}}{T} \tag{6}$$

$$l_{\mathsf{split}} = \prod_{x_i \in X_t^1} \prod_{x_j \in X_t^2} \frac{\psi(x_i \succ x_j)}{\varphi(x_i \sim x_j)} \tag{7}$$

2.4.2. Merge Operator

For T subsets, the probability of merging two consecutive ones will be $(T-1)^{-1}$ since there are $T-1$ pairs, and each pair can be merged in exactly one way. Let T_{merge} be the number of

non-singleton subsets after the merge, and let N_t and N_{t+1}be the sizes of the two subsets X_t and X_{t+1}, respectively. Let $N_t^* = N_t + N_{t+1}$, the probability of recovering the state before the merge (by applying the `split` operator) is $\left(T_{\mathsf{merge}} N_t^*(N_t^* - 1)2^{N_t^*-2}\right)^{-1}$. Consequently, the proposal probability ratio p_{merge} can be given as in Eq (8), and the likelihood ratio l_{merge} is clearly the inverse of the split case as shown in Eq (9).

$$p_{\mathsf{merge}} = \frac{T-1}{T_{\mathsf{merge}} N_t^*(N_t^*-1)2^{N_t^*-2}} \quad (8) \qquad l_{\mathsf{merge}} = \prod_{x_i \in X_t} \prod_{x_j \in X_{t+1}} \frac{\varphi(x_i \sim x_j)}{\psi(x_i \succ x_j)} \quad (9)$$

Finally, the pseudo-code of the `split-and-merge` Metropolis-Hastings procedure for the OSM is presented in Algorithm 1.

1. Given an initial state X.
2. **Repeat** until convergence
   - 2a. Draw a random number $\eta \in [0, 1]$.
   - 2b. **If** $\eta < 0.5$ {`Split`}
     - i. Randomly choose a non-singleton subset.
     - ii. Split into two sub-subsets and insert one sub-subset right after the another.
     - iii. Evaluate the acceptance probability P_{accept} using Eqs.(6,7,5).
     - iv. Accept the move with probability P_{accept}.
   - **Else** {`Merge`}
     - i. Randomly choose two consecutive subsets.
     - ii. Merge them in one, keeping the relative orders with other subsets unchanged.
     - iii. Evaluate the acceptance probability P_{accept} using Eqs.(8,9,5).
     - iv. Accept the move with probability P_{accept}.
   - **End**

End

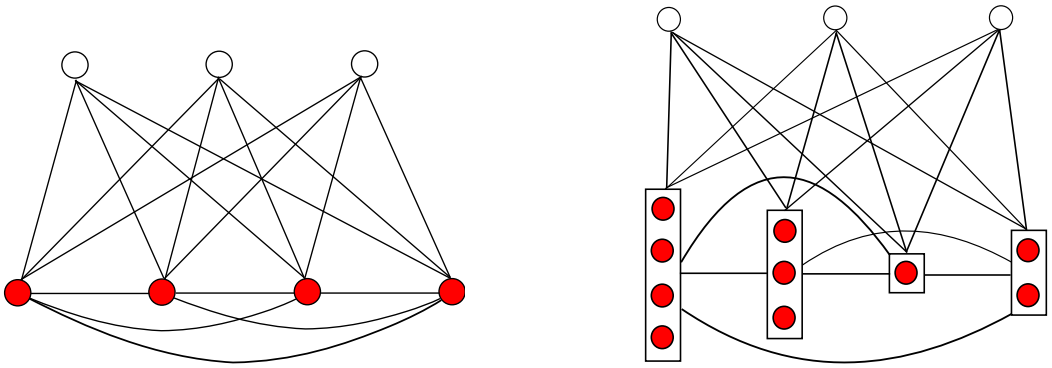
Algorithm 1: Pseudo-code of the `split-and-merge` Metropolis-Hastings for OSM.

2.5. Estimating Partition Function

To estimate the normalisation constant Z, we employ an efficient procedure called Annealed Importance Sampling (AIS) proposed recently (Neal, 2001). More specifically, AIS introduces the notion of inverse-temperature τ into the model, that is $P(X|\tau) \propto \Omega(X)^\tau$.

Let $\{\tau_s\}_{s=0}^S$ be the (slowly) increasing sequence of temperature, where $\tau_0 = 0$ and $\tau_S = 1$, that is $\tau_0 < \tau_1 ... < \tau_S$. At $\tau_0 = 0$, we have a uniform distribution, and at $\tau_S = 1$, we obtain the desired distribution. At each step s, we draw a sample X^s from the distribution $P(X|\tau_{s-1})$ (e.g. using the `split-and-merge` procedure). Let $P^*(X|\tau)$ be the unnormalised distribution of $P(X|\tau)$, that is $P(X|\tau) = P^*(X|\tau)/Z(\tau)$. The final weight after the annealing process is computed as

$$w = \frac{P^*(X^1|\tau_1)}{P^*(X^1|\tau_0)} \frac{P^*(X^2|\tau_2)}{P^*(X^2|\tau_1)} \cdots \frac{P^*(X^S|\tau_S)}{P^*(X^S|\tau_{S-1})}$$



(a) (b)

Figure 2: (a) A Semi-Restricted Boltzmann Machine representation of vectorial data: each shaded node represents a visible variable and empty nodes the hidden units. (b) A Latent OSM for representing ordered sets: each box represents a subset of objects.

The above procedure is repeated R times. Finally, the normalisation constant at $\tau = 1$ is computed as $Z(1) \approx Z(0) \left(\sum_{r=1}^{R} w^{(r)}/R \right)$ where $Z(0) = \text{Fubini}(N)$, which is the number of configurations of the model state variables X.

2.6. Log-linear Parameterisation and Learning

Here we assume that the model is in the log-linear form, that is $\varphi(x_i \sim x_j) = \exp\{\sum_a \alpha_a f_a(x_i, x_j)\}$ and $\psi(x_i \succ x_j) = \exp\{\sum_b \beta_b g_b(x_i, x_j)\}$, where $\{f_a(\cdot),\, g_b(\cdot)\}$ are sufficient statistics (or feature functions) and $\{\alpha_a,\, \beta_b\}$ are free parameters.

Learning by maximising (log-)likelihood in log-linear models with respect to free parameters often leads to computing the expectation of sufficient statistics. For example, $\langle f_a(x_i, x_j) \rangle_{P(x_i \sim x_j)}$ is needed in the gradient of the log-likelihood with respect to α_a, where $P(x_i \sim x_j)$ is the pairwise marginal. Unfortunately, computing $P(x_i \sim x_j)$ is inherently hard, and running a full MCMC chain to estimate it is too expensive for practical purposes. Here we follow the stochastic approximation proposed in (Younes, 1989), in that we iteratively update parameters after very short MCMC chains (e.g., using Algorithm 1).

3. Introducing Latent Variables to OSMs

In this section, we further extend the proposed OSM by introducing latent variables into the model. The latent variables serve multiple purposes. For example, in collaborative filtering, each person chooses only a small subset of objects, thus the specific choice of objects and the ranking reflects personal taste. This cannot be discovered by the standard OSM. Second, if we want to measure the distance or similarity between two ordered partitioned sets, e.g. for clustering or visualisation, it may be useful to first transform the data into some vectorial representation.

3.1. Model Specification

Denote by $\boldsymbol{h} = (h_1, h_2, ..., h_K) \in \{0,1\}^K$ the hidden units to be used in conjunction with the ordered sets. The idea is to estimate the posterior $P(h_k = 1|X)$ - the probability that the k^{th} hidden unit will be activated by the input X. Thus, the requirement is that the model should allow the evaluation of $P(h_k = 1 \mid X)$ efficiently. Borrowing from the Restricted Boltzmann Machine architecture (Smolensky, 1986; Welling et al., 2005), we can extend the model potential function as follows:

$$\hat{\Omega}(X, \boldsymbol{h}) = \Omega(X) \prod_k \Omega_k(X)^{h_k} \tag{10}$$

where $\Omega_k(X)$ admits the similar factorisation as $\Omega(X)$, i.e. $\Omega_k(X) = \prod_t \Phi_k(X_t) \prod_{t'>t} \Psi_k(X_t \succ X_{t'})$, and

$$\Phi_k(X_t) \;=\; \prod_{i,j \in X_t | j>i} \varphi_k(x_i \sim x_j); \qquad \Psi_k(X_t \succ X_{t'}) = \prod_{i \in X_t} \prod_{j \in X_{t'}} \psi_k(x_i \succ x_j) \tag{11}$$

where $\varphi_k(x_i \sim x_j)$ and $\psi_k(x_i \succ x_j)$ capture the events of tie and relative ordering between objects x_i and x_j *under the presence of the* k^{th} *hidden unit*, respectively.

We then define the model with hidden variables as $P(X, \boldsymbol{h}) = \hat{\Omega}(X, \boldsymbol{h})/Z$, where $Z = \sum_{X,\boldsymbol{h}} \hat{\Omega}(X, \boldsymbol{h})$. A graphical representation is given in Figure 2b. Hereafter, we shall refer to this proposed model as the Latent OSM.

3.2. Inference

The posteriors are indeed efficient to evaluate:

$$P(\boldsymbol{h} \mid X) \;=\; \prod_k P(h_k \mid X), \quad \text{where } P(h_k = 1 \mid X) = \frac{1}{1 + \Omega_k(X)^{-1}} \tag{12}$$

Denote by h_k^1 as the shorthand for $h_k = 1$, the vector $(P(h_1^1 \mid X), P(h_2^1 \mid X), ..., P(h_K^1 \mid X))$ can then be used as a latent representation of the configuration X.

The generation of X given $\boldsymbol{h}$ is, however, much more involved as we need to explore the whole subset partitioning and ordering space:

$$P(X \mid \boldsymbol{h}) = \frac{\hat{\Omega}(X, \boldsymbol{h})}{\sum_X \hat{\Omega}(X, \boldsymbol{h})} = \frac{\Omega(X) \prod_k \Omega_k(X)^{h_k}}{\sum_X \Omega(X) \prod_k \Omega_k(X)^{h_k}} \tag{13}$$

For inference, since we have two layers X and $\boldsymbol{h}$, we can alternate between them in a Gibbs sampling manner, that is, sampling X from $P(X \mid \boldsymbol{h})$ and then $\boldsymbol{h}$ from $P(\boldsymbol{h} \mid X)$. Since sampling from $P(\boldsymbol{h} \mid X)$ is straightforward, it remains to sample from $P(X|\boldsymbol{h}) = \hat{\Omega}(X, \boldsymbol{h}) / \sum_X \hat{\Omega}(X, \boldsymbol{h})$. Since $\hat{\Omega}(X, \boldsymbol{h})$ has the same factorisation structure into a product of pairwise potentials as $\Omega(X)$, we can employ the `split-and-merge` technique described in the previous section in a similar manner.

To see how, let $\hat{\varphi}(x_i \sim x_j, \boldsymbol{h}) = \varphi(x_i \sim x_j) \prod_k \varphi_k(x_i \sim x_j)^{h_k}$ and $\hat{\psi}(x_i \succ x_j, \boldsymbol{h}) = \psi_k(x_i \succ x_j) \prod_k \psi_k(x_i \succ x_j)^{h_k}$, then from Eqs.(4,10,11). We can see that $\hat{\Omega}(X, \boldsymbol{h})$ is now

factorised into products of $\hat{\varphi}(x_i \sim x_j, \boldsymbol{h})$ and $\hat{\psi}(x_i \succ x_j, \boldsymbol{h})$ in the same way as $\Omega(X)$ into products of $\varphi(x_i \sim x_j)$ and $\psi(x_i \succ x_j)$:

$$\hat{\Omega}(X, \boldsymbol{h}) \quad = \quad \Omega(X) \prod_k \Omega_k(X)^{h_k} = \prod_t \hat{\Phi}(X_t, \boldsymbol{h}) \prod_{t'>t} \hat{\Psi}(X_t \succ X_{t'}, \boldsymbol{h})$$

where

$$\hat{\Phi}(X_t, \boldsymbol{h}) \quad = \quad \prod_{i,j \in X_t | j>i} \hat{\varphi}(x_i \sim x_j, \boldsymbol{h}); \qquad \hat{\Psi}(X_t \succ X_{t'}, \boldsymbol{h}) = \prod_{i \in X_t} \prod_{j \in X_{t'}} \hat{\psi}(x_i \succ x_j, \boldsymbol{h})$$

Estimating the normalisation constant Z can be performed using the AIS procedure described earlier (cf. Section 2.5), except that the unnormalised distribution $P^*(X|\tau)$ is given as:

$$P^*(X \mid \tau) \quad = \sum_{\boldsymbol{h}} \hat{\Omega}(X, \boldsymbol{h})^\tau = \quad \Omega(X)^\tau \prod_k (1 + \Omega_k(X)^\tau)$$

which can be computed efficiently for each X.

For sampling X^s from $P(X \mid \tau_{s-1})$, one way is to sample directly from the $P(X \mid \tau_{s-1})$ in a Rao-Blackwellised fashion (e.g. by marginalising over $\boldsymbol{h}$ we obtain the unnormalised $P^*(X|\tau)$). A more straightforward way is alternating between $X \mid \boldsymbol{h}$ and $\boldsymbol{h} \mid X$ as usual. Although the former would give lower variance, we implement the latter for simplicity. The remaining is similar to the case without hidden variables, and we note that the base partition function $Z(0)$ should be modified to $Z(0) = \text{Fubini}(N)2^K$, taking into account of K binary hidden variables. A pseudo-code for the `split-and-merge` algorithm for Latent OSM is given in Algorithm 8.

1. Given an initial state X.
2. **Repeat** until convergence
   - 2a. Sample $\boldsymbol{h}$ from $P(\boldsymbol{h} \mid X)$ using Eq.(12).
   - 2b. Sample X from $P(X \mid \boldsymbol{h})$ using Eq.(13) and Algorithm 1.
   - **End**

End

Algorithm 2: Pseudo-code of the `split-and-merge` Gibbs/Metropolis-Hastings for Latent OSM.

3.3. Parameter Specification and Learning

Like the OSM, we also assume log-linear parameterisation. In addition to those potentials shared with the OSM, here we specify hidden-specific potentials as follows: $\varphi_k(x_i \sim x_j)^{h_k} = \exp\{\sum_a \lambda_{ak} f_a(x_i, x_j) h_k\}$ and $\psi_k(x_i \succ x_j)^{h_k} = \exp\{\sum_b \mu_{bk} g_b(x_i, x_j) h_k\}$. Now $\{f_a(x_i, x_j)h_k,\ g_b(x_i, x_j)h_k\}$ are new sufficient statistics. As before, we need to estimate the expectation of sufficient statistics, e.g., $\langle f_a(x_i, x_j)h_k \rangle_{P(x_i, x_j, h_k)}$. Equipped with Algorithm 8, the stochastic gradient trick as in Section 2.6 can then be used, that is, parameters are updated after very short chains (with respect to the model distribution $P(X, \boldsymbol{h})$).

4. Application in Collaborative Filtering

In this section, we present one specific application of our Latent OSM in collaborative filtering. Recall that in this application, each user has usually expressed their preferences over a set of items by rating them (e.g., by assigning each item a small number of stars). Since it is cumbersome to rank all the items completely, the user often joins items into groups of similar ratings. As each user often rates only a handful of items out of thousands (or even millions), this creates a sparse ordering of subsets. Our goal is to first discover the latent taste factors for each user from their given ordered subsets, and then use these factors to recommend new items for each individual.

4.1. Rank Reconstruction and Completion

In this application, we are limited to producing a *complete ranking* over objects instead of subset partitioning and ordering. Here we consider two tasks: (i) *rank completion* where we want to rank unseen items given a partially ranked set[3], and (ii) *rank reconstruction*[4] where we want to reconstruct the complete rank $\hat{X}$ from the posterior vector $(P(h_1^1 \mid X), P(h_2^1 \mid X)..,, P(h_K^1 \mid X))$.

Rank completion. Assume that an unseen item x_j might be ranked higher than any seen item $\{x_i\}_{i=1}^N$. Let us start from the mean-field approximation

$$P(x_j \mid X) \quad = \quad \sum_{\boldsymbol{h}} P(x_j, \boldsymbol{h} \mid X) \approx Q_j(x_j \mid X) \prod_k Q_k(h_k \mid X)$$

From the mean-field theory, we arrive at Eq (14), which resembles the factorisation in (10).

$$Q_j(x_j \mid X) \quad \propto \quad \Omega(x_j, X) \prod_k \Omega_k(x_j, X)^{Q_k(h_k^1 \mid X)} \tag{14}$$

Now assume that X is sufficiently informative to estimate $Q_k(h_k \mid X)$, we make further approximation $Q_k(h_k^1 \mid \boldsymbol{x}) \approx P(h_k^1 \mid \boldsymbol{x})$. Finally, due to the factorisation in (11), this reduces to

$$Q_j(x_j \mid X) \propto \prod_i \left[\psi(x_j \succ x_i) \prod_k \psi_k(x_j \succ x_i)^{P_k(h_k^1 \mid X)} \right]$$

The RHS can be used for the purpose of ranking among new items $\{x_j\}$.

Rank reconstruction. The *rank reconstruction* task can be thought as estimating $\hat{X} = \arg\max_{X'} Q(X'|X)$ where $Q(X'|X) = \sum_{\boldsymbol{h}} P(X'|\boldsymbol{h}) P(\boldsymbol{h}|X)$. Since this maximisation is generally intractable, we may approximate it by treating X' as state variable of *unseen* items, and apply the mean-field technique as in the completion task.

4.2. Models Implementation

To enable fast recommendation, we use a rather simple scheme: Each item is assigned a worth $\phi(x_i) \in \mathbb{R}^+$ which can be used for ranking purposes. Under the Latent OSM, the

3. This is important in recommendation, as we shall see in the experiments.
4. This would be useful in data compression setting.

worth is also associated with a hidden unit, e.g. $\phi_k(x_i)$. Then the events of grouping and ordering can be simplified as

$$\varphi_k(x_i \sim x_j) \quad = \quad \theta\sqrt{\phi_k(x_i)\phi_k(x_j)}; \quad \text{and} \quad \psi_k(x_i \succ x_j) = \phi_k(x_i)$$

where $\theta > 0$ is a factor signifying the contribution of item compatibility to the model probability. Basically the first equation says that if the two items are compatible, their worth should be positively correlated. The second asserts that if there is an ordering, we should choose the better one. This reduces to the tie model of (Davidson, 1970) when there are only two items.

For learning, we parameterise the models as follows

$$\theta = e^{\nu}; \qquad \phi(x_i) \quad = \quad e^{u_i}; \qquad \phi_k(x_i) = e^{W_{ik}}$$

where ν, $\{u_i\}$ and $\{W_{ik}\}$ are free parameters. The Latent OSM is trained using stochastic gradient with a few samples per user to approximate the gradient (e.g., see Section 3.3). To speed up learning, parameters are updated after every block of 100 users. Figure 3(a) shows the learning progress with learning rate of 0.01 using parallel *persistent* Markov chains, one chain per user (Younes, 1989). The samples get closer to the observed data as the model is updated, while the acceptance rates of the `split-and-merge` decrease, possibly because the samplers are near the region of attraction. A notable effect is that the `split-and-merge` dual operators favour sets of small size due to the fact that there are far more many ways to split a big subset than to merge them. For the AIS, we follow previous practice (e.g. see (Salakhutdinov and Murray, 2008)), i.e. $S = \{10^3, 10^4\}$ and $R = \{10, 100\}$.

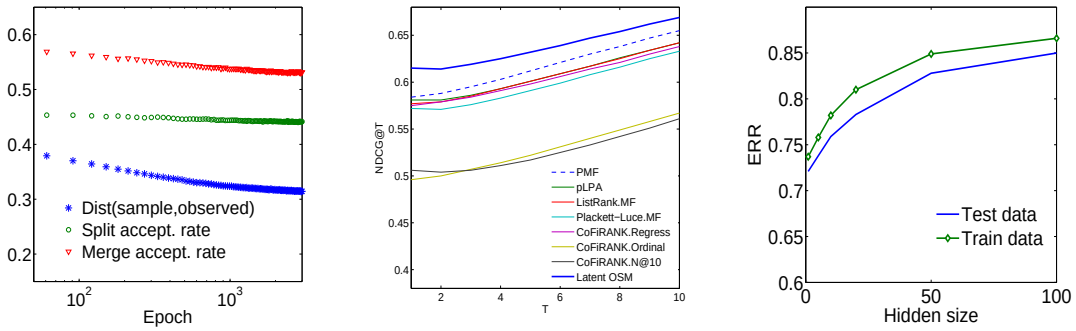
For comparison, we implemented existing methods including the Probabilistic Matrix Factorisation (*PMF*) (Salakhutdinov and Mnih, 2008) where the predicted rating is used as scoring function, the Probabilistic Latent Preference Analysis (*pLPA*) (Liu et al., 2009), the ListRank.MF (Shi et al., 2010) and the matrix-factored Plackett-Luce model (Truyen et al., 2011) (*Plackett-Luce.MF*). For the pLPA we did not use the MM algorithm but resorted to simple gradient ascent for the inner loop of the EM algorithm. We also ran the CoFi^{RANK} variants (Weimer et al., 2008) with code provided by the authors[5]. We found that the ListRank-MF and the Plackett-Luce.MF are very sensitive to initialisation, and good results can be obtained by randomly initialising the user-based parameter matrix with non-negative entries. To create a rank for Plackett-Luce.MF, we order the ratings according to `quicksort`.

The performance will be judged based on the correlation between the predicted rank and ground-truth ratings. Two performance metrics are reported: the Normalised Discounted Cumulative Gain at the truncated position T (NDCG@T) (Järvelin and Kekäläinen, 2002), and the Expected Reciprocal Rank (ERR) (Chapelle et al., 2009):

$$\texttt{NDCG@}T \quad = \tfrac{1}{\kappa(T)} \quad \sum_{i=1}^{T} \frac{2^{r_i} - 1}{\log_2(1+i)}; \quad \texttt{ERR} = \sum_i \frac{1}{i} V(r_i) \prod_{j=1}^{i-1} (1 - V(r_j)) \quad \texttt{for } V(r) = \frac{2^{r-1} - 1}{16}$$

where r_i is the relevance judgment of the movie at position i, $\kappa(T)$ is a normalisation constant to make sure that the gain is 1 if the rank is correct. Both the metrics put more emphasis on top ranked items.

5. http://cofirank.org



(a) (b) (c)

Figure 3: Results with MovieLens data. (a) Learning progress with time: Dist(sample,observed) is the portion of pairwise orders being incorrectly sampled by the `split-and-merge` Markov chains ($N = 10$,$K = 20$). (b) Rank completion, as measured in NDCG@T ($N = 20$, $K = 50$). (c) Rank reconstruction ($N = 10$) - trained on $9,000$ users and tested on $1,000$ users.

	$M = 10$	$M = 20$	$M = 30$
Plackett-Luce.MF	-14.7	-41.3	-72.6
Latent OSM	-9.8	-37.9	-73.4

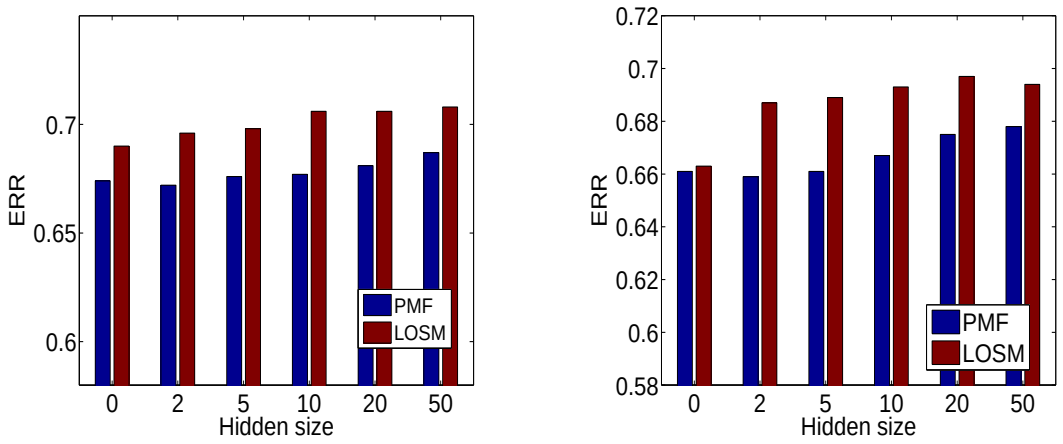
Table 1: Average log-likelihood over 100 users of test data (Movie Lens 10M dataset), after training on the $N = 10$ movies per user ($K = 10$). M is the number of test movies per user. The Plackett-Luce.MF and the Latent OSM are comparable because they are both probabilistic in ranks and can capture latent aspects of the data. The main difference is that the Plackett-Luce.MF does not handle groupings or ties.

4.3. Results

We evaluate our proposed model and inference on large-scale collaborative filtering datasets: the MovieLens[6] 10M and the Netflix challenge[7]. The MovieLens dataset consists of slightly over 10 million half-integer ratings (from 0 to 5) applied to $10,681$ movies by $71,567$ users. The ratings are from 0.5 to 5 with 0.5 increments. We divide the rating range into 5 segments of equal length., and those ratings from the same segment will share the same rank. The Netflix dataset has slightly over 100 million ratings applied to $17,770$ movies by $480,189$ users, where ratings are integers in a 5-star ordinal scale.

Data likelihood estimation. Table 1 shows the log-likelihood of test data averaged over 100 users with different numbers of movies per user. Results for the Latent OSM are estimated using the AIS procedure.

6. http://www.grouplens.org/node/12
7. http://www.netflixprize.com



(a) MovieLens, $N = 20$ (b) Netflix $N = 10$

Figure 4: Rank completion quality vs. number of hidden units - note that since the PMF is not defined when hidden size is 0, we substitute using the result for hidden size 1.

Rank reconstruction. Given the posterior vector, we ask whether we can reconstruct the original rank of movies for that data instance. For simplicity, we only wish to obtain a complete ranking, since it is very efficient (e.g. a typical cost would be $N \log N$ per user). Figure 3(c) indicates that high quality rank reconstruction (on both training and test data) is possible given enough hidden units. This suggests an interesting way to store and process rank data by using vectorial representation.

Rank completion. In collaborative filtering settings, we are interested in ranking unseen movies for a given user. To highlight the disparity between user tastes, we remove movies whose qualities are inherently good or bad, that is when there is a general agreement among users. More specifically, we compute the movie entropy as $H_i = -\sum_{r=1}^{5} P_i(r) \log P_i(r)$ where $P_i(r)$ is estimated as the proportion of users who rate the movie i by r points. We then remove half of the movies with lowest entropy. For each dataset, we split the data into a training set and a test set as follows. For each user, we randomly choose 10, 20 and 50 items for training, and the rest for testing. To ensure that each user has at least 10 test items, we keep only those users with no less than 20, 30 and 60 ratings, respectively.

Figs. 3(b), 4(a) and Table 2 report the results on the MovieLens 10M dataset; Figs. 4(b) and Table 3 show the results for the Netflix dataset. It can be seen that the Latent OSM performs better than rivals when N is moderate. For large N, the rating-based method (PMF) seems to work better, possibly because converting rating into ordering loses too much information in this case, and it is more difficult for the Latent OSM to explore the hyper-exponential state-space .

5. Related Work

This work is closely related to the emerging concept of *preferences over sets* in AI (Brafman et al., 2006; Wagstaff et al., 2010) and in social choice and utility theories (Barberà et al.,

	$N = 10$		$N = 20$		$N = 50$	
	ERR	N@5	ERR	N@5	ERR	N@5
PMF	0.673	0.603	0.687	0.612	**0.717**	**0.638**
pLPA	0.674	0.596	0.684	0.601	0.683	0.595
ListRank.MF	0.683	0.603	0.682	0.601	0.684	0.595
Plackett-Luce.MF	0.663	0.586	0.677	0.591	0.681	0.586
CoFi^{RANK}.Regress	0.675	0.597	0.681	0.598	0.667	0.572
CoFi^{RANK}.Ordinal	0.623	0.530	0.621	0.522	0.622	0.515
CoFi^{RANK}.N@10	0.615	0.522	0.623	0.517	0.602	0.491
Latent OSM	**0.690**	**0.619**	**0.708**	**0.632**	0.710	0.629

Table 2: Model comparison on the MovieLens data for rank completion ($K = 50$). N@T is a shorthand for NDCG@T.

	$N = 10$				$N = 20$			
	ERR	N@1	N@5	N@10	ERR	N@1	N@5	N@10
PMF	0.678	0.586	0.607	0.649	0.691	0.601	0.624	0.661
ListRank.MF	0.656	0.553	0.579	0.623	0.658	0.553	0.577	0.617
Latent OSM	**0.694**	**0.611**	**0.628**	**0.666**	**0.714**	**0.638**	**0.648**	**0.680**

Table 3: Model comparison on the Netflix data for rank completion ($K = 50$).

2004). However, most existing work has focused on representing preferences and computing the optimal set under preference constraints (Binshtok et al., 2007). These differ from our goals to model a distribution over all possible set orderings and to learn from example orderings. Learning from expressed preferences has been studied intensively in AI and machine learning, but they are often limited to pairwise preferences or complete ordering (Cohen et al., 1999; Weimer et al., 2008).

On the other hand, there has been very little work on learning from ordered sets (Yue and Joachims, 2008; Wagstaff et al., 2010). The most recent and closest to our is the PMOP which models ordered sets as a locally normalised *high-order Markov chain* (Truyen et al., 2011). This contrasts with our setting which involves a globally normalised log-linear solution. Note that since the high-order Markov chain involves all previously ranked subsets, while our OSM involves pairwise comparisons, the former is not a special case of ours. Our additional contribution is that we model the space of partitioning and ordering directly and offer sampling tools to explore the space. This ease of inference is not readily available for the PMOP. Finally, our solution easily leads to the introduction of latent variables, while their approach lacks that capacity.

Our `split-and-merge` sampling procedure bears some similarity to the one proposed in (Jain and Neal, 2004) for mixture assignment. The main difference is that we need to handle the extra orderings between partitions, while it is assumed to be exchangeable in (Jain and Neal, 2004). This causes a subtle difference in generating proposal moves. Likewise, a similar method is employed in (Ranganathan et al., 2006) for mapping a set of observations into a set of landmarks, but again, ranking is not considered.

With respect to collaborative ranking, there has been work focusing on producing a set of items instead of just ranking individual ones (Price and Messinger, 2005). These can be considered as a special case of OSM where there are only two subsets (those selected and the rest).

6. Conclusion and Future Work

We have introduced a latent variable approach to modelling ranked groups. Our main contribution is an efficient `split-and-merge` MCMC inference procedure that can effectively explore the hyper-exponential state-space. We demonstrate how the proposed model can be useful in collaborative filtering. The empirical results suggest that proposed model is competitive against state-of-the-art rivals on a number of large-scale collaborative filtering datasets.

References

S. Barberà, W. Bossert, and P.K. Pattanaik. Ranking sets of objects. *Handbook of Utility Theory: Extensions*, 2:893, 2004.

M. Binshtok, R.I. Brafman, S.E. Shimony, A. Martin, and C. Boutilier. Computing optimal subsets. In *Proceedings of the National Conference on Artificial Intelligence*, volume 22, page 1231. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007.

R.I. Brafman, C. Domshlak, S.E. Shimony, and Y. Silver. Preferences over sets. In *Proceedings of the National Conference on Artificial Intelligence*, volume 21, page 1101. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006.

O. Chapelle, D. Metlzer, Y. Zhang, and P. Grinspan. Expected reciprocal rank for graded relevance. In *CIKM*, pages 621–630. ACM, 2009.

W.W. Cohen, R.E. Schapire, and Y. Singer. Learning to order things. *J Artif Intell Res*, 10:243–270, 1999.

R.R. Davidson. On extending the Bradley-Terry model to accommodate ties in paired comparison experiments. *Journal of the American Statistical Association*, 65(329):317–328, 1970.

O. Dekel, C. Manning, and Y. Singer. Log-linear models for label ranking. *Advances in Neural Information Processing Systems*, 16, 2003.

S. Jain and R.M. Neal. A split-merge Markov chain Monte Carlo procedure for the Dirichlet process mixture model. *Journal of Computational and Graphical Statistics*, 13(1):158–182, 2004.

K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):446, 2002.

N.N. Liu, M. Zhao, and Q. Yang. Probabilistic latent preference analysis for collaborative filtering. In *CIKM*, pages 759–766. ACM, 2009.

M. Mureşan. *A concrete approach to classical analysis*. Springer Verlag, 2008.

R.M. Neal. Annealed importance sampling. *Statistics and Computing*, 11(2):125–139, 2001.

R. Price and P.R. Messinger. Optimal recommendation sets: Covering uncertainty over user preferences. In *Proceedings of the National Conference on Artificial Intelligence*, volume 20, page 541. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2005.

A. Ranganathan, E. Menegatti, and F. Dellaert. Bayesian inference in the space of topological maps. *Robotics, IEEE Transactions on*, 22(1):92–107, 2006.

R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. *Advances in neural information processing systems*, 20:1257–1264, 2008.

R. Salakhutdinov and I. Murray. On the quantitative analysis of deep belief networks. In *Proceedings of the 25th international conference on Machine learning*, pages 872–879. ACM, 2008.

Y. Shi, M. Larson, and A. Hanjalic. List-wise learning to rank with matrix factorization for collaborative filtering. In *ACM RecSys*, pages 269–272. ACM, 2010.

P. Smolensky. Information processing in dynamical systems: Foundations of harmony theory. *Parallel distributed processing: Explorations in the microstructure of cognition*, 1:194–281, 1986.

T. Truyen, D.Q Phung, and S. Venkatesh. Probabilistic models over ordered partitions with applications in document ranking and collaborative filtering. In *Proc. of SIAM Conference on Data Mining (SDM)*, Mesa, Arizona, USA, 2011. SIAM.

J.H. van Lint and R.M. Wilson. *A course in combinatorics*. Cambridge Univ Pr, 1992.

S. Vembu and T. Gärtner. Label ranking algorithms: A survey. *Preference Learning*, page 45, 2010.

K.L. Wagstaff, Marie desJardins, and E. Eaton. Modelling and learning user preferences over sets. *Journal of Experimental & Theoretical Artificial Intelligence*, 22(3):237–268, 2010.

M. Weimer, A. Karatzoglou, Q. Le, and A. Smola. CoFi^{RANK}-maximum margin matrix factorization for collaborative ranking. *Advances in neural information processing systems*, 20:1593–1600, 2008.

M. Welling, M. Rosen-Zvi, and G. Hinton. Exponential family harmoniums with an application to information retrieval. In *Advances in NIPS*, volume 17, pages 1481–1488. 2005.

L. Younes. Parametric inference for imperfectly observed Gibbsian fields. *Probability Theory and Related Fields*, 82(4):625–645, 1989.

Y. Yue and T. Joachims. Predicting diverse subsets using structural SVMs. In *Proceedings of the 25th international conference on Machine learning*, pages 1224–1231. ACM, 2008.