JMLR: Workshop and Conference Proceedings 41:33–48, 2015 BIGMINE 2015

Scalable Multidimensional Hierarchical Bayesian Modeling on Spark

Robert Ormandi RORMANDI@YAHOO-INC.COM
Hongxia Yang HONGXIA@YAHOO-INC.COM
Quan Lu QLU@YAHOO-INC.COM
Yahoo!, 701 1st Ave, Sunnyvale, CA, 94089.

Editors: Wei Fan, Albert Bifet, Qiang Yang and Philip Yu

Abstract

We consider the problem of estimating occurrence rates of rare events for extremely sparse data using pre-existing hierarchies and selected features to perform inference along multiple dimensions. In particular, we focus on the problem of estimating click rates for {Advertiser, Publisher, User} tuples where both the Advertisers and the Publishers are organized as hierarchies that capture broad contextual information at different levels of granularities. Typically, the click rates are low and the coverage of the hierarchies and dimensions is sparse. To overcome these difficulties, we decompose the joint prior of the three-dimensional Click-Through-Rate (CTR) using tensor decomposition and propose a Multidimensional Hierarchical Bayesian framework (abbreviated as MadHab). We set up a specific framework of each dimension to model dimension-specific characteristics. More specifically, we consider the hierarchical beta process prior for the Advertiser dimension and for the Publisher dimension respectively and a feature-dependent mixture model for the User dimension. Besides the centralized implementation, we propose a distributed algorithm through Spark for inference which make the model highly scalable and suited for large scale data mining applications. We demonstrate that on a real world ads campaign platform our framework can effectively discriminate extremely rare events in terms of their click propensity.

Keywords: Display Advertising, Hierarchical Beta Process Prior, Dependent Mixture Modeling, Spark.

1. Introduction

Display Advertising has been the subject of rigorous research with extremely fast development during the past decade. The area has generated billions of revenue, originated hundreds of scientific papers and patents, seen a broad variety of implementations, yet the accuracy of prediction technologies leaves much to be desired. The expected revenue from displaying each ad is a function of both the bid price and the Click-Through-Rate (CTR). Sponsored search advertising, contextual advertising, display advertising, and real-time bidding auctions have all relied heavily on the ability of learned models to predict ad CTR accurately, quickly and reliably. Note that CTR prediction is not only related to web publishers' revenue but users' experience and advertisers' payment, because this influences ranking, filtering, placement and pricing of ads. Campaign performance directly depends on how well the CTR can be estimated, whereas the performance optimization can be considered as the problem of accurately estimating CTR. If these quantities are over-estimated, bid prices will always be higher than what they should be, the advertiser will waste campaign budget on less valuable impressions; on the other hand, if these quantities

are underestimated, the advertiser will miss high-value impressions that may have led to actions and the campaign will under deliver. That is why CTR prediction plays an important role in the multi-faceted advertising business.

Our work is motivated by the challenges from a world leading advertising platform, YAM+ (Yahoo! Advertising Manager Plus), which is the flagship of Yahoo!'s programmatic ad buying application suite. YAM+ offers efficient Real-Time Bidding (RTB) buying platforms and provides access to Yahoo! and third party inventories. YAM+ capitalizes on relevant billions of user data that has an incomparable breadth, precision and targeting history. Such data access allows us to find and target users across all available inventories including mobile devices. The unparalleled reach across global multi-area exchanges, all major advertisers and an enormous repository of user data facilitates YAM+'s deployment and supports campaign management that capitalizes on maximizing ad campaign's reach, relevance and exposure frequency. To this end, YAM+ relies on incredible user reach, on accuracy of targeting and prediction analytics, on effectiveness of real-time bidding algorithms. However, it is a big challenge to set up a flexible complete model framework that consistently integrates information from different dimensions.

Here we focus on the following challenges. First, CTR generally ranges from 0.001% to 0.5%, which is highly skewed towards the non-clicked class with very high variances. Predictions of CTR for ads are generally based on machine learning or statistical models trained by using the past click data. Examples of such models are logistic regression (Cheng and Cantú-Paz, 2010; Cheng et al., 2012), probit regression (Graepel et al., 2010), and boosted trees (Dave and Varma, 2010; Trofimov et al., 2012).

Second, the complexity of Display Advertising is the huge event space, whose data hierarchies can be expressed as {Advertiser, Publisher, User}. Previous work predominantly describes separate efforts focused on just Advertiser (Menon et al., 2011), or Publisher (Tagami et al., 2013), or User (Agichtein et al., 2006; Richardson et al., 2007), with an integrated multi-dimensional framework being too large and too complex to handle. In our work we consider hierarchical models with various resolutions for all three dimensions using tensor decomposition and discuss how to construct an effective integrated framework that performs inference across all dimensions.

Third, however the dataset we use here is sparse, the amount of data we collect is still extremely large. Therefore, apart from designing a suitable model which captures the data generation process accurately, we need to design a scalable algorithm which can consume the date at the scale we have and fits the model parameters efficiently. For this reason, after proposing our model, we introduce a novel distributed Marcov Chain Monte Carlo (MCMC) sampler relying on in-memory distributed computation engine GraphX (Xin et al., 2013) and Spark (Zaharia et al., 2012).

To the best of our knowledge, we are the first effort to tackle the problem of CTR prediction in such a complete framework. In other words, we encapsulate multiple hierarhicies and dimensions into a unified Bayesian framework. It is also the first time to deploy such a multidimensional hierarchical Bayesian framework on Spark through GraphX. The main advantage is that it allows key information to be shared among the different resolutions and levels.

1.1. Our Contributions

The main contributions of this paper can be summarized as follows:

1. For the first time, in the context of computational advertising, we studied problems where multiple resolutions as well as multiple dimensions may exhibit different degrees of relatedness, and for such problems we consistently integrate information across {Advertiser, Publisher, User};
2. We propose a multidimensional hierarchical Bayesian framework, MadHab, which adaptively learns various multi-resolution relatedness and multi-dimension consistency;
3. We also propose a scalable distributed algorithm for inference that applies graph-parallel processing to provide posterior sampling from the whole dataset, namely MadHab-Spark;
4. We carry out extensive analysis of the model and the proposed algorithms focusing on both the performance and the scalability aspects.

The paper is organized as follows. In Section 2, we briefly review the related work. The dynamic hierarchical Bayesian framework is proposed in Section 3, followed by two algorithm descriptions, which are the centralized MCMC algorithm and the distributed one in popular scalable computational environments on Spark through GraphX in Section 4. Section 5 compares with state-of-the-art methods on real data sets. Finally, we conclude the paper in Section 6.

2. Related Work

With the scale of the data as large as our problem, subset-based communication-free parallel MCMC methods are crucial. There are currently three main directions to accelerate computations. First one is to parallelize computation of the likelihood at each MCMC iteration (Agarwal and Duchi, 2012). Under conditional independence, calculations can be conducted separately for mini-batches of data stored on different machines, with results fed back to a central processor. The bottleneck for this approach is to require communications within each iteration, which limits the overall speed. Second strategy focuses on using random mini-batches of data to accelerate expensive gradient calculations in Langevin and Hamiltonian Monte Carlo algorithms through stochastic approximation (Welling and Teh, 2011). Third direction is to partition the data into mini-batches and run MCMC independently without communications and then combine the chains to mimic draws from the full data posterior distribution (Wang and Dunson, 2014; Scott et al., 2013; Neiswanger et al., 2013). The last direction is obviously more realistic in a very complex Bayesian framework and we will parallelize this algorithm by extending Wang and Dunson (2014).

We selected the Spark (Zaharia et al., 2012) distributed computational environment to implement the proposed algorithms. The Spark framework introduces a richer computational model which supports the in-memory iterative computations as well. Both systems are highly scalable and widely used in the industry.

3. Dynamic Hierarchical Bayesian Framework

3.1. Notations

Figure 1 illustrates the hierarchical multidimensional structure for the {Advertiser, Publisher, User}. Note that this figure does not represent Yam+'s data taxonomy and it is

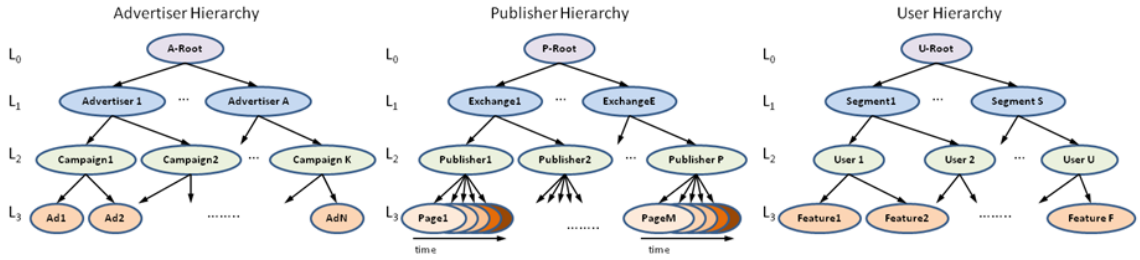

Figure 1: Hierarchical Structure for Ads Campaign Platform

mainly provided for illustration and clarification for the presentation. For example, each ad in the campaign platform can be considered as belonging to an advertising campaign, which in its turn belongs to an advertiser. Similarly, Publisher dimension also embeds such hierarchy: several pages may belong to one publisher and the publisher itself may belong to some category based on its main content. Users can also be segmented hierarchically. Assuming for the advertiser dimension, $a(i, j, k, l)$ stands for the cluster belonging to the ith advertiser, jth campaign, kth line and lth ad. Similarly for the publisher dimension, $p(i, j, k, l)$ stands for the cluster belonging to the ith publisher, jth domain, kth subdomain and lth page. $u(i)$ stands for the ith cluster of the user.

3.2. Model Formulation

We are considering the following general model:

$$y_{a(i,j,k,l),p(i,j,k,l),u(i)} \sim \text{Bernoulli}\big(q_{a(i,j,k,l),p(i,j,k,l),u(i)}\big), \tag{1}$$

where $\mathbf{q} = \{q_{a(i,j,k,l),p(i,j,k,l),u(i)}\}$ is the probability table representing the underlying click probabilities for the tuple of {Advertiser, Publisher, User} and $y_{a(i,j,k,l),p(i,j,k,l),u(i)}$ is the action that has been taken, e.g., click or no-click. We model the probability table $q_{a(i,j,k,l),p(i,j,k,l),u(i)}$ using the following beta distribution and decompose the prior probabilities through rank-one tensor decomposition (Savicky and Vomlel, 2005) to explicitly model the latent CTR distributions from each dimension:

$$\text{Beta}\big(c_q q_{a(i,j,k,l)} q_{p(i,j,k,l,t)} q_{u(i)}, c_q(1 - q_{a(i,j,k,l)} q_{p(i,j,k,l,t)} q_{u(i)})\big),$$

where $q_{a(i,j,k,l)}$ is the marginal CTR for the Advertiser dimension, and similarly $q_{p(i,j,k,l,t)}$ and $q_{u(i)}$ are marginal probabilities for the Publisher and User dimensions. More specifically, we model the prior marginal probabilities of Advertisers, Publishers and Users as follows.

Advertisers and Publishers There are usually millions of advertisers and a few hundred publishers and the main difficulty in such simultaneous rate estimation is the paucity of data and absence of events at fine resolution. Hence rate estimates obtained independently for each cell are often unreliable and noisy. When data is hierarchical, *borrowing strength* from aggregates across multiple resolutions often leads to estimates with a good bias-variance trade-off. We extend the hierarchical Beta priors (Zhou et al., 2011) to perform such borrowing. In general, a "small sample size correction" obtained by *properly* pooling information across different data aggregates provides better estimates.

Instead of choosing the hyper parameters directly for the hierarchical Beta priors, one can choose a central point (mean or mode) and some measure of the spread for the prior distribution. For the prior mean, one can consider the average CTR on different levels, that is the long-term frequency of the observed event of interest: set $\alpha/(\alpha+\beta)$ to be the empirical CTR, where α and β are the corresponding shape parameters for the Beta priors. For the binary variables, this sample mean coincides with the frequency that the event happens in the sample. The choice of the spread is more difficult since sample variance of the observations is not a good choice, especially if the prior distribution is highly skewed. An alternative is to estimate how many extra ensemble members c' the prior information is worth. To conclude, we formulate the hierarchical beta prior for such hierarchical structure as following:

$$\begin{aligned} q_{a_{i,j,k,l}} &\sim \text{Beta}\big(c_3 q_{a_{i,j,k}}, c_3(1-q_{a_{i,j,k}})\big), \\ q_{a_{i,j,k}} &\sim \text{Beta}\big(c_2 q_{a_{i,j}}, c_2(1-q_{a_{i,j}})\big), \\ q_{a_{i,j}} &\sim \text{Beta}\big(c_1 q_{a_i}, c_1(1-q_{a_i})\big), \\ q_{a_i} &\sim \text{Beta}\big(c_0 q_a, c_0(1-q_a)\big). \end{aligned} \tag{2}$$

We estimate the marginal probabilities of Advertisers at multiple resolutions, a_i stands for the ith Advertiser, $a_{i,j}$ stands for the jth campaign belonging to the ith Advertiser, $a_{i,j,k}$ stands for the kth line belonging to the jth campaign and $a_{i,j,k,l}$ denotes the lth ads in the hierarchical tree structures, which are determined by existing hierarchies. These have been created and constantly refined by domain experts; they are well understood and routinely used in applications. Apart from the ease of interpretability, there are other advantages in using such hierarchies. First, they provide a natural and intuitive framework to analyze CTRs at multiple scales, from coarser to successively finer resolutions. The hierarchical Beta prior is expected to induce correlations in CTRs among siblings at each resolution. Also, at coarser resolutions, aggregation induced by the hierarchical Beta prior helps in combating data sparseness and providing reliable estimates for rare events. Note that aggregation can substantially reduce variance but is likely to incur bias. However, if the hierarchy is informative and siblings are homogeneous, the variance reduction will far outweigh the increase in bias. Similar hierarchical set up holds for Publishers dimensions.

Users Agichtein et al. (2006) showed that incorporating user behavior data can significantly improve the ordering of top results in reality, Richardson et al. (2007) also employed logistic regression to predict the CTR for newly created ads, using dimension-specific features. We denote $\mathbf{x}_{u(i)}$ the corresponding user features from cluster $u(i)$ and use the following mixture to capture the user variabilities (Zaman et al., 2014):

$$\begin{aligned} \text{logit}(q_{u(i)}) &\sim \text{Normal}\big(\mathbf{x}'_{u(i)}\beta_{u(i)}, U_i^2\big), \\ (\beta_{u(i)}, U_i^2) &\sim \sum_{j=1}^{J} \pi_j \text{Normal-Inverse-Gamma}(\beta_j^*, U_j^*), \end{aligned} \tag{3}$$

where (β_j^*, U_j^*) are cluster-specific parameters and π_j is the weights for the jth cluster. We can rewrite the above formulation as following:

$$\begin{gathered} \text{logit}(q_i) \sim \text{Normal}\big(\mathbf{x}'_{u(i)}\beta_{u(i)}, U_i^2\big), \\ \{\beta_{u(i)}, U_i^2\} | (Z_i = j) \sim \text{Normal-IG}(\beta_j^*, U_j^*; \beta_0, \Omega_0, \gamma_1, \gamma_2) \\ U_j^* \sim \text{IG}(\gamma_1, \gamma_2), \quad \beta_j^* \sim \text{Normal}(\beta_0, \Omega_0), \quad \Omega_0 \sim \text{Inverse-Wishart}(\gamma_0, I_0) \\ Z_i \sim \text{MultiNormial}(\pi), \quad \pi = \{\pi_1, \ldots, \pi_J\}, \quad \pi \sim \text{Dirichlet}(\theta), \quad \theta = \{\theta_1, \ldots, \theta_J\}, \end{gathered} \tag{4}$$

where $\text{IG}(\gamma_1, \gamma_2)$ stands for inverse gamma distribution with shape and scale parameter γ_1 and γ_2, (β_0, Ω_0) are prior mean and covariance for the latent cluster mean β_j^*, (γ_0, I_0) are the degree of freedom and scale matrix for Ω_0, θs are prior parameters for the Dirichlet distributions. We choose non-informative priors for the above set up, e.g., β_0 is an 0 vector and Ω_0 is a diagonal matrix with 10^3 in the diagonals and θs are chosen to be 1. After the learning, we will further segment the users based on the latent marginal CTR distribution for the user dimension.

3.3. Model Calibration

One of most important paradigms in display advertising is to predict well calibrated probabilities; good accuracy or area under the Receiver Operating Characteristic (ROC) curve are not sufficient. In this scenario, campaign performance directly depends on how well the CTR can be estimated and the performance optimization can be considered as the problem of accurately estimating CTR. If these quantities are overestimated, bid prices will always be higher than what they should be, the advertiser will waste campaign budget on less valuable impressions; on the other hand, if these quantities are underestimated, the advertiser will miss high-value impressions that might have led to actions and the campaign will under deliver. In Niculescu-Mizil and Caruana (2005a) and Niculescu-Mizil and Caruana (2005b) they qualitatively examine what kinds of distortions these calibration methods are suitable for and quantitatively examine how much data they need to be effective.

However, we show in the following sections that no calibration is needed for a Bayesian framework (e.g., forecasts follow a Bernoulli distribution) which corresponds to a linear calibration of the relative frequencies. Graepel et al. (2010) also empirically showed their Bayesian framework adPredictor does not need calibration. For the sake of simplicity, assuming that $p(y_t|p_t) \sim \text{Bernoulli}(p_t)$. Denote $p(p_t|I_t)$ the posterior distribution of p_t and I_t accounts for all left parameters and observed information related to parameter p_t. The predictive distribution of $p(y_t|I_t)$ then can be calculated as following:

$$p(y_t|I_t) = \int p(y_t|I_t, p_t)p(p_t|I_t)dp_t = \int p(y_t|p_t)p(p_t|I_t)dp_t = \int p_t^{y_t}(1-p_t)^(1-y_t)p(p_t|I_t)dp_t$$
$$= \begin{cases} \int p_t p(p_t|I_t)dp_t & \text{if } y_t = 1 \\ \int (1-p_t)p(p_t|I_t)dp_t & \text{if } y_t = 0 \end{cases} = \begin{cases} \int E(p_t|I_t) & \text{if } y_t = 1 \\ 1 - \int E(p_t|I_t) & \text{if } y_t = 0 \end{cases}$$

Thus $y_t|I_t \sim \text{Bernoulli}(E(p_t|I_t))$ and $Pr(y_t = 1|I_t) = E(p_t|I_t)$. So the estimated p_t is the corresponding latent CTR.

4. The Proposed Algorithm

To simplify the description of the algorithm updates, we assume that there are only two levels for the Advertiser dimension and Publisher dimension respectively, however, it is easily extended for multi-level hierarchies.

4.1. Markov Chain Monte Carlo

We use the following Markov Chain Monte Carlo algorithm to update unknown parameters:

Top Level Decomposition Updates

1. Denote $N_{a_{i,j},p_t}$ number of impressions for a specific user and update $q_{a_{i,j},p_t,u_i}$ through:

$$\text{Beta}\Big(\sum_{i,j,p_t}(y_{a_{i,j},p_t,u_i}) + c_q q_{a_{i,j}} q_{p_t} q_{u_i}, N_{a_{i,j},p_t} - \sum_{i,j,p_t}(y_{a_{i,j},p_t,u_i}) + c_q(1 - q_{a_{i,j}} q_{p_t} q_{u_i})\Big) \quad (5)$$

Similarly, $N_{a_{i,j},u_i}$ stands for number of impressions for a specific publisher and update a specific Publisher through:

$$\text{Beta}\Big(\sum_{i,j,u_i}(y_{a_{i,j},p_t,u_i}) + c_q q_{a_{i,j}} q_{p_t} q_{u_i}, N_{a_{i,j},u_i} - \sum_{i,j,u_i}(y_{a_{i,j},p_t,u_i}) + c_q(1 - q_{a_{i,j}} q_{p_t} q_{u_i})\Big) \quad (6)$$

For a specific Advertiser layer through:

$$\text{Beta}\Big(\sum_{p_t,u_i}(y_{a_{i,j},p_t,u_i}) + c_q q_{a_{i,j}} q_{p_t} q_{u_i}, N_{p_t,u_i} - \sum_{p_t,u_i}(y_{a_{i,j},p_t,u_i}) + c_q(1 - q_{a_{i,j}} q_{p_t} q_{u_i})\Big) \quad (7)$$

Advertisers and Publishers Dimension Updates

2. The posterior distribution of Advertiser probability $q_{a_{i,j}}$ is proportional to

$$f(q_{a_{i,j}}|\cdots) \propto \Big(\prod_{p_t,u_i}(\frac{q_{a_{ij}p_t u_i}}{1 - q_{a_{ij}p_t u_i}})^{c_q q_{p_t} q_{u_i}}\Big)^{q_{a_{ij}}} \times q_{a_{ij}}^{c_1 q_{a_i} - 1}\big(1 - q_{a_{ij}}\big)^{c_1(1-q_{a_i})-1}. \quad (8)$$

which cannot be directly sampled from using a closed form posterior distribution. Alternatively, we use the following slice sampling (Damlen1 et al., 2002) by introducing auxiliary variables u_a and v_a:

$$u_a \sim \text{unif}(0, q_{a_{ij}}^{c_1 q_{a_i} - 1}), \quad v_a \sim \text{unif}(0, \big(1 - q_{a_{ij}}\big)^{c_1(1-q_{a_i})-1})$$

and then draw $q_{a_{ij}}$ from the following truncated exponential distribution:

$$\text{Exp}\Big(\prod_{p_t,u_i}(\frac{1 - q_{a_{ij}p_t u_i}}{q_{a_{ij}p_t u_i}})^{c_q q_{p_t} q_{u_i}}\Big) I(q_{a_{ij}}), \quad (9)$$

where $I(q_{a_{ij}})$ represents the range derived from (9).

3. Similarly, the posterior distribution of q_{a_i} is proportional to:

$$\prod_j q_{a_{i,j}}^{c_1 q_{a_i} - 1}(1 - q_{a_{i,j}})^{c_1(1-q_{a_i})-1} \times q_{a_i}^{c_0 q_0 - 1}(1 - q_{a_i})^{c_0(1-q_0)-1}$$
$$\propto \Big(\prod_j(\frac{q_{a_{i,j}}}{1 - q_{a_{i,j}}})\Big)^{q_{a_i}} \times q_{a_i}^{c_0 q_0 - 1}(1 - q_{a_i})^{c_0(1-q_0)-1}$$

and will be updated through slice sampling as well. Updates for Publishers dimension proceed similarly.

Users Dimension Updates

4. The posterior distribution for q_{u_i} is proportional to

$$f(q_{u_i}|\cdots) \propto \prod_{a_{ij},p_t} \Big[\big(\frac{q_{a_{ij}p_t u_i}}{1-q_{a_{ij}p_t u_i}}\big)^{c_q q_{a_{ij}} q_{p_t}} \Big]^{q_{u_i}} \exp\Big\{ - \frac{\big(\log(\frac{q_{u_i}}{1-q_{u_i}}) - x'_{u_i}\beta_{Z_i}\big)^2}{2U^2_{Z_i}} \Big\} \quad (10)$$

We use Metropolis-Hastings Method to update q_{u_i} and our proposal distribution is $g(q_{u_i}|q^*_{u_i}) = \text{Beta}(c_u q^*_{u_i}, c_u(1-q^*_{u_i}))$, where $q^*_{u_i}$ is the update from the last iteration. We accept the new q_{u_i} with the following probability:

$$\frac{f(q_{u_i})g(q^*_{u_i}|q_{u_i})}{f(q^*_{u_i})g(q_{u_i}|q^*_{u_i})} \quad (11)$$

5. The posterior distribution of β^*_j is as following:

$$\prod_{Z_i=j,i=1,\ldots,N} \exp\big\{ - \frac{(\log(\frac{q_{u_i}}{1-q_{u_i}}) - \mathbf{x}'_i\beta^*_j)^2}{2U^2_j}\big\} \times \exp\{-\frac{(\beta^*_j - \beta_0)'\Omega_0^{-1}(\beta^*_j - \beta_0)}{2}\} \quad (12)$$

And we update β^*_j through multivariate normal with mean:

$$\big(\sum_{Z_i=j,i=1,\ldots,N} \frac{\mathbf{x}_i\mathbf{x}'_i}{U^2_j} + \Omega_j^{-1}\big)^{-1} \times \big(\sum_{Z_i=j} \frac{\mathbf{x}_i \log(\frac{q_{u_i}}{1-q_{u_i}})}{U^2_j} + \Omega_j^{-1}\beta_0\big) \quad (13)$$

and the covariance matrix:

$$\Big(\sum_{Z_i=j,i=1,\ldots,N} \frac{\mathbf{x}_i\mathbf{x}'_i}{U^2_j} + \Omega_0^{-1}\Big)^{-1} \quad (14)$$

6. The posterior distribution of U_j is proportional to:

$$\prod_{Z_i=j,i=1,\ldots,N} \exp\big\{ - \frac{(\log(\frac{q_{u_i}}{1-q_{u_i}}) - \mathbf{x}'_i\beta^*_j)^2}{2U^2_j}\big\} \times U_i^{2(\alpha+1)} \exp\{-\frac{\gamma}{U^2_j}\} \quad (15)$$

We update U^2_j through

$$U^2_j \sim \text{IG}\big(\alpha + \frac{\sum I(Z_i == j)}{2}, \gamma + \frac{1}{2}\sum_{Z_i=j} \big(\log(\frac{q_{u_i}}{1-q_{u_i}}) - \mathbf{x}'_i\beta^*_j)^2\big)$$

7. Update Z_i is updated through the multinormial distribution with π^*_j with the constraint that $\sum_{j=1}^J \pi^*_j = 1$

$$\pi^*_j \propto \pi_j \text{lognormal}(q_{u_i}; \mathbf{x}'_i\beta^*_j, U^2_j) \quad (16)$$

8. π_j is updated through

$$\pi \sim \text{Dirichlet}\big(\sum_{i=1}^N I(Z_i == j) + \theta_j\big) \quad (17)$$

Based on the above discussion, the proposed algorithm is summarized in Algorithm 1.

Algorithm 1 Batch Markov Chain Monte Carlo(MCMC) Algorithm

Require: $y_{a_{i,j}p_tu_i}$, $\mathbf{x}_{p,t}$, $\mathbf{x}_{u_i}$, c_q, $c_{0:1}$, q_a,B_t, σ^2, Q_t, γ_0 and θ
Ensure: the initial value for $q_{a_{i,j}}$, $q_{p_{i,j}}$, q_{u_i}, θ_0, Z, β_0 and Ω_0

```
for i = 1 to Total number of iterations do
  for a_{i,j} do
    for p_t do
      for u_i do
        Update q_{a_{i,j},p_{i,j},u_i} through Beta distributions as in Item 1. Section 4 ;
        Update q_{a_{i,j}} and q_{a_i} through truncated exponential distributions as in Item 2.
        and Item 3. of Section 4;
        Draw q_{p_{i,j}} and related parameters similarly to q_{a_{i,j}};
        Update related parameters for user variabilities through the mixture distribu-
        tions in Item 5. to Item 8. of Section 4 .
      end for
    end for
  end for
end for
```

4.2. Scalability

To be able to *successfully* apply the proposed model in real world application, we need to design efficient distributed algorithm that "scales well". That is, it can fit the parameters of the model specified in Section 3.2 *efficiently* using vey large amount of data stored distibutedly in some cluster arhitecture.

In this section, we propose a distributed algorithm for fitting the parameters of the model specified in Section 3.2. We provide an analysis of the algorithm and compare it to the centralized algorithm presented in Algorithm 1 and a Map-Reduce-based naive distributed implementation, detailed later in Sec. 5.2, in terms of accuracy, speed and scalability. However, it is important to note that, from the viewpoint of the scalability, the centrailized algorithm can be seen as a hypothetical algorithm; since it is not possible to run this algorithm at large scale.

4.2.1. The Distributed Implementation

The real challenge of the large scale distributed model building is often that, and this is true in the case of our model too, the size of the model itself is too large to be stored on a single machine. One possible solution to tackle the problem is to decompose the problem into smaller sub-problems, and then apply multiple machines to solve the sub-problems independently, then combine the solutions of the sub-problems to obtain an approximation of the original problems.

Another approach is to find a way to distribute the model and data across multiple machine in a way that a large portion of the computation corresponding to the model building can be performed parallel (maximizing the use of multiple machines) while the synchronization between machines (model components) should be rare or asynchronous. This is a different approach than the one mentioned above; since here parallel executions of "local operations" on model components with lightweight or no (asynchronous) sync between them leads to "global convergence".

Our distributed algorithm, called MadHab-Spark, addresses the problem of model fitting by adopting the second approach above. The distributed representation of algorithm relies on the fact that the MCMC updates, defined in Section 4.1, of a particular node depends only on the nodes located in the *Markov blanket* of the given node in the Bayesian Network representation of the model (Murphy, 2012). Hence, the posterior updating algorithms of two or more nodes can be executed independently and parallel assuming that, they are distant from each other in the Bayesian network.

This observation leads to the following distributed algorithm implemented using the distributed graph computational API called GraphX (Xin et al., 2013, 2014) running on the top distributed framework Spark (Zaharia et al., 2012):

- *Input:* The algorithm receives a general description of the model in the form of *property graph* which describes the model as a Bayesian network, i.e. a directed acyclic graph (DAG), and binds the observations to data pieces stored in the distributed file system HDFS.
- *Initialization:* As a first step, the algorithm builds the in-memory model structure by creating a distributed property graph (Xin et al., 2013), called the *blanket graph*, loads the data into the observation nodes of the model and initializes the remaining nodes.
- *Processing:* The algorithm runs the posterior update algorithms defined in Section 4.1 on each Markov blanket as a distributed graph node program and spread the updated values along the blanket edges within Gather-Apply-Scatter (Xin et al., 2013) cycles iteratively.

In the algorithm description above, the referred *blanket graph* is a DAG $\mathcal{G} = (V, E)$ created from the Bayesian Network representation of the model presented in Figure 2. Each Markov blanket of the original model is represented as a *blanket node* $v \in V$ in the blanket graph. Moreover, we introduce a directed *blanket edge* between two blanket nodes v and w if and only if there is an edge running in the original model between the Bayesian network nodes assigned to v and w respectively.

The initialization of the different type of nodes is as follows: (1) the observation nodes (the one associated with data) receives the their values from the data, (2) the prior nodes (the one without incoming edges in Fig. 2) receives their predefined values coming as part of the input, and (3) the initial values of the stochastic nodes are drawn as random value according to the distribution associated with them using the parent values as parameters.

It is easy to see that the above edge definition of the blanket graph exactly describes the computational dependencies between the Markov blankets. Hence, the above mentioned algorithm computes exactly the same output as the centralized algorithm. As a corollary, the performance of the algorithm is independent from the number of worker machines (unlike in the case of our second baseline detailed in the Section 5.1). Also, one can note that the above defined algorithm is independent from the specified model and one can easily adapt the algorithm to different models. However, the scalability of the algorithm depends on the structure of the blanket graph, meaning that applying it on different model structures may lead to different scalability results.

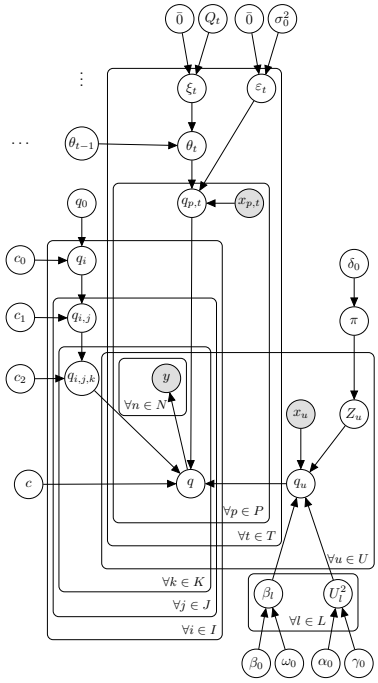

Figure 2: A slightly simplified Bayesian Network representation of the model. Here q is a shorthand of $q_{i,j,k,p,t,u}$ and y is a shorthand of $y_{i,j,k,p,t,u,n}$. The value of θ_t depends on its value from the previous iteration. This is denoted by the arrow from θ_{t-1}.

5. Experiments

In this section, we first describe the features, data sets and baseline algorithms used in the experimental evaluations. Then, we carry out extensive evaluations and present the experimental results using the world leading advertising platform YAM+ to show the effectiveness of the proposed framework MadHab.

5.1. Experimental Setup

We have collected the data of one running campaign with life cycle spanning from 01/19/2015 till 03/31/2015. This campaign has multiple lines with each line consisting of several ads. Each line has its specific targeting criteria with allocated budgets. The campaign itself has a setup cost-per-click (CPC) goal and is bidding with a dynamical click-per-impression (CPM) pricing type. If a CPC goal is set up, then the optimal bid price can be determined from the expected cost-per-impression, which is equal to the CTR for this impression multiplied by the CPC goal (Chen et al., 2011). On average, the dataset contains 8 million instances daily with 134 selected features from the different resolutions for Advertisers, Publishers and Users.

The campaigns are running on YAM+, which is a Demand Side Platform (DSP). YAM+ provides the capabilities to do the real-time bidding for the advertiser on both Yahoo! and 3rd party inventories. The platform's goal is to maximize its campaigns performance. It is roughly 100ms before a time-out for DSP to decide which ad to bid and how much to bid for an incoming ad call. And, the predictive models need to be executed multiple times (once

for each eligible ad) before this time-out threshold. Furthermore, in this environment, there are significant variances on daily actively running campaigns as well as supplied inventories. So, it requires the model to be able to run very fast and also automatically adapt to the external changes by itself.

5.2. Baseline Algorithms

To evaluate the algorithm MadHab, we introduced several baseline algorithms. The first one, is the centralized algorithm introduced in Algorithm 1. This cannot be considered as a distributed algorithm. We use the provided performance as a reference.

The further baseline algorithms utilize the Map-Reduce framework. We applied the popular regularized Generalized linear model using Lasso (Tibshirani, 1996) and Elastic Net (Zou and Hastie, 2005) on the 3 dimensions {Advertiser, Publisher, User} independently with and without calibration (Niculescu-Mizil and Caruana, 2005b) implemented on the top of the Map-Reduce framework.

The next baseline algorithm, called MadHab-MapReduce, is a straightforward distributed implementation of our model on the top of the Map-Reduce framework. The main idea of the algorithm is as follows. Each *mapper* machine in the system is responsible for handling a subset X_i of the whole dataset. Each of them applies independent MCMC sampling from the subset posterior density (i.e from $p(\theta|X_i)$). Because of the nature of the distributed framework Map-Reduce, these subset samplers run in parallel. Then, the subset samples are combined by the only one *reducer* which computes the Weierstrass transformation (Wang and Dunson, 2014). This implementation can be seen as a highly parallelized *approximation* of the batch MCMC algorithm.

The clear advantage of this baseline algorithm is that it is simple and fits well to the widely used Map-Reduce framework. However, the performance-scalability trade-off of the algorithm, as we will show empirically in the Section 5.3, heavily depends on the number of mappers applied. When the number of mappers is too few, we cannot parallelize enough the execution to get a scalable algorithm; on the other hand when it is too large, the dataset X_i per mapper is too small to be able to build an efficient sub-model. As we discussed in Section 4.2.1, the proposed algorithm MadHab does not have this restriction.

5.3. Click Prediction Analysis for Ads

Using the above introduced database and applying baseline algorithms defined in the previous subsection, we experimented in the following scenarios:

1. Regularized Generalized Linear Model using Lasso (Tibshirani, 1996) Elastic Net (Zou and Hastie, 2005) *with* calibration through isotonic regression (Niculescu-Mizil and Caruana, 2005b);
2. MadHab-MapReduce *without* calibration;
3. MadHab-Spark *without* calibration.

We also tested the sensitivity of the performances of algorithm 1 to 3 on different number of mappers, ranging from 80 to 10000. To quantify the quality of the ranking that results from different algorithms according to the predicted probability, we compare the algorithms' areas under the ROC curve (or AUC). Results are listed in Figure 3 (LHS).

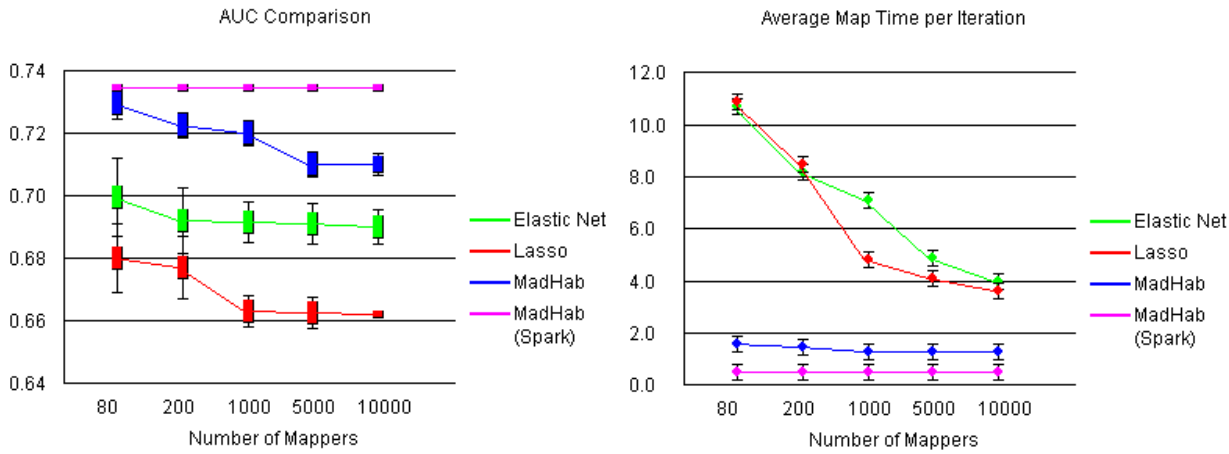

Figure 3: Performance and processing time comparison

As we can see, when the number of mappers increase, the related accuracies decrease. In general, Lasso achieves the worst performances and Elastic Net performs better compared to Lasso. Elastic Net is a better choice when features are correlated and the 134 features selected from the different resolutions of the different hierarchies are somewhat correlated with each. MadHab-MapReduce achieves the 2nd best, however, the performances still get worse as the number of mappers increase. MadHab-Spark is ranked the best over the four algorithms. We also show the average time *per iteration* of the different algorithms in Figure 3 (RHS). Here clearly the algorithm MadHab is the most preferable choice.

5.4. Score Calibration

In this part, we explore the influence of dataset imbalance on the model performance for the campaign. As we have pointed out, we use the AUC as the performance metric. We show in Section 3.3 that it is important for the model calibration for generalized linear model through either Lasso or Elastic Net and theoretically showed that for the Bayesian framework, calibration is not needed. The resulted CTR scores for Lasso and Elastic Net are plotted in the left part of Figure 4. However, they are located far from the underlying true CTR which is around 0.4% on average. The scale of the logistic regression output scores are quite sensitive to the imbalance ratio of the data as illustrated. After the calibration, the scores coincide with the actual campaign CTR rates (see the right part of Figure 4). It is also clear that MadHab does not need the calibration.

6. Conclusions

In this paper we dealt with the problem of CTR prediction in the context of display advertising. We proposed a novel hierarchical Bayesian framework which models the problem along multiple dimensions and with various resolutions jointly. In addition, we developed a highly scalable distributed algorithm for performing inference in this model framework on Spark through the framework GraphX.

Our main conclusion is that the proposed model achieves a significant improvement in terms of prediction performance, while the computational complexity of the parameter inference can be kept at a manageable level by applying the proposed algorithms in a

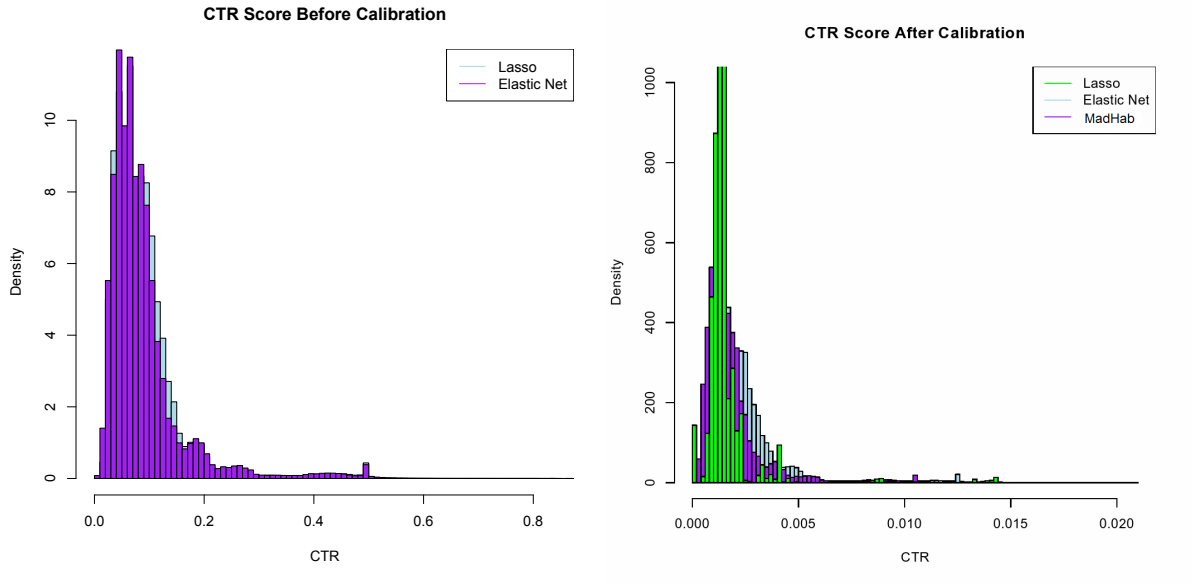

Figure 4: CTR Scores with and without calibration

suitable distributed environment. These two facts confirm that it is worth modeling complex problems, like the CTR prediction, at the original level of complexity by applying e.g. the flexible Bayesian modeling framework within the suitable computational environments.

References

A. Agarwal and J.C. Duchi. Distributed delayed stochastic optimization, decision and control. In *2012 IEEE 51st Annual Conference*, pages 5451–5452, 2012.

Eugene Agichtein, Eric Brill, and Susan Dumais. Improving web search ranking by incorporating user behavior information. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, pages 19–26, New York, NY, USA, 2006. ACM. ISBN 1-59593-369-7. doi: 10.1145/1148170.1148177. URL http://doi.acm.org/10.1145/1148170.1148177.

Ye Chen, Pavel Berkhin, Bo Anderson, and Nikhil R. Devanur. Real-time bidding algorithms for performance-based display ad allocation. In *ACM SIGKDD Conf. on Knowledge Discovery and Data Mining*, San Diego, CA, USA, 2011. ACM.

Haibin Cheng and Erick Cantú-Paz. Personalized click prediction in sponsored search. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, WSDM '10, pages 351–360, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-889-6. doi: 10.1145/1718487.1718531. URL http://doi.acm.org/10.1145/1718487.1718531.

Haibin Cheng, Roelof van Zwol, Javad Azimi, Eren Manavoglu, Ruofei Zhang, Yang Zhou, and Vidhya Navalpakkam. Multimedia features for click prediction of new ads in display advertising. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 777–785, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1462-6. doi: 10.1145/2339530.2339652. URL http://doi.acm.org/10.1145/2339530.2339652.

P. Damlen1, J. Wakefield, and S. Walker. Gibbs sampling for bayesian non-conjugate and hierarchical models by using auxiliary variables. *Journal of the Royal Statistical Society: Series B*, 61:331–344, 2002.

Kushal S. Dave and Vasudeva Varma. Learning the click-through rate for rare/new ads from similar ads. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '10, pages 897–898, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0153-4. doi: 10.1145/1835449.1835671. URL http://doi.acm.org/10.1145/1835449.1835671.

Thore Graepel, Joaquin Quinonero Candela, Thomas Borchert, and Ralf Herbrich. Web-scale bayesian click-through rate prediction for sponsored search advertising in microsoft's bing search engine. In *ICML*, pages 13–20. Omnipress, 2010. ISBN 978-1-60558-907-7. URL http://dblp.uni-trier.de/db/conf/icml/icml2010.html#GraepelCBH10.

Aditya Krishna Menon, Krishna-Prasad Chitrapura, Sachin Garg, Deepak Agarwal, and Nagaraj Kota. Response prediction using collaborative filtering with hierarchies and side-information. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 141–149, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0813-7. doi: 10.1145/2020408.2020436. URL http://doi.acm.org/10.1145/2020408.2020436.

Kevin P Murphy. *Machine learning: a probabilistic perspective*. Cambridge, MA, 2012.

W. Neiswanger, C. Wang, and E. Xing. Asymptotically exact, embarrassingly parallel mcmc. In *arXiv:1311.4780*, 2013.

A. Niculescu-Mizil and R. Caruana. Obtaining calibrated probabilities from boosting. *Proc. 21th Conference on Uncertainty in Artificial Intelligence*, 2005a.

A. Niculescu-Mizil and R. Caruana. Predicting good probabilities with supervised learning. *Proc. 22nd International Conference on Machine Learning*, 2005b.

Matthew Richardson, Ewa Dominowska, and Robert Ragno. Predicting clicks: Estimating the click-through rate for new ads. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 521–530, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-654-7. doi: 10.1145/1242572.1242643. URL http://doi.acm.org/10.1145/1242572.1242643.

P. Savicky and J. Vomlel. Tensor rank-one decomposition of probability tables. *Technical Report DAR- UTIA 2005/26, Institute of Information Theory and Automation, Prague, Czech Republic*, 2005.

S.L. Scott, A.W. Blocker, and F.V. Bonassi. Bayes and big data: The consensus monte carlo algorithm. In *Bayes 250*, 2013.

Yukihiro Tagami, Shingo Ono, Koji Yamamoto, Koji Tsukamoto, and Akira Tajima. Ctr prediction for contextual advertising: Learning-to-rank approach. In *Proceedings of the Seventh International Workshop on Data Mining for Online Advertising*, ADKDD '13, pages 4:1–4:8, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2323-9. doi: 10.1145/2501040.2501978. URL http://doi.acm.org/10.1145/2501040.2501978.

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B*, pages 267–288, 1996.

Ilya Trofimov, Anna Kornetova, and Valery Topinskiy. Using boosted trees for click-through rate prediction for sponsored search. In *Proceedings of the Sixth International Workshop on Data Mining for Online Advertising and Internet Economy*, ADKDD '12, pages 2:1–2:6, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1545-6. doi: 10.1145/2351356.2351358. URL http://doi.acm.org/10.1145/2351356.2351358.

X. Wang and D.B. Dunson. Parallelizing mcmc via weierstrass sampler. In *http://arxiv.org/pdf/1312.4605*, 2014.

M. Welling and Y. Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proc. 28th International Conference on Machine Learning*, pages 681–688, 2011.

Reynold S. Xin, Joseph E. Gonzalez, Michael J. Franklin, and Ion Stoica. Graphx: A resilient distributed graph system on spark. In *First International Workshop on Graph Data Management Experiences and Systems*, GRADES '13, pages 2:1–2:6, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2188-4. doi: 10.1145/2484425.2484427. URL http://doi.acm.org/10.1145/2484425.2484427.

Reynold S. Xin, Daniel Crankshaw, Ankur Dave, Joseph E. Gonzalez, Michael J. Franklin, and Ion Stoica. Graphx: Unifying data-parallel and graph-parallel analytics. *CoRR*, abs/1402.2394, 2014. URL http://arxiv.org/abs/1402.2394.

Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J. Franklin, Scott Shenker, and Ion Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, NSDI'12, pages 2–2, Berkeley, CA, USA, 2012. USENIX Association. URL http://dl.acm.org/citation.cfm?id=2228298.2228301.

T. Zaman, E. Fox, and E. Bradlow. A bayesian approach for predicting the popularity of tweets. *The Annals of Applied Statistics*, 8(3):583–611, 2014.

M. Zhou, H. Yang, G. Sapiro, D. Dunson, and L. Carlin. Dependent hierarchical Beta process for image interpolation and denoising. In G. Gordon, D. Dunson, and M. Dudik, editors, *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, Fort Lauderdale, FL, 2011. JMLR W&CP, vol. 15.

H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society. Series B*, pages 301–320, 2005.