INFERRING LATENT STATES IN A NETWORK INFLUENCED BY NEIGHBOR ACTIVITIES: AN UNDIRECTED GENERATIVE APPROACH

Buddhika L. Samarakoon, Manohar N. Murthi, Kamal Premaratne

Department of Electrical and Computer Engineering
University of Miami, Coral Gables, Florida, USA
Email: buddhika@umiami.edu, {mmurthi,kamal}@miami.edu

ABSTRACT

The problem of inferring the hidden state of individual nodes in social/sensor networks in which node activities affect their neighbors is growing in importance. We present an undirected generative model, a type of probabilistic model that has so far not been used for modeling latent variables influenced by neighbors in a network. We also propose an efficient inference method based on variational inference principles which, in contrast to sampling methods used in most existing models, is scalable to larger networks. While training is intractable in general, by using stochastic methods to approximate the intractable derivative, we show that our model can be trained using the maximum likelihood method by formulating the model as an exponential family distribution. The results demonstrate that the proposed undirected model can accurately infer latent states compared to baseline methods.

Index Terms— Network influence, latent sentiments, factor models, exponential family, stochastic gradient methods.

1. INTRODUCTION

Modern networks (e.g., online social networks) routinely generate large amount of signals/data, and computationally efficient methods are essential to analyze and understand these datasets. The temporal evolution of data in these networks is often influenced by their neighbors. Furthermore, there can be hidden states or latent sentiments that are not observable [1, 2, 3]. For example, a sensor may update its belief regarding a hidden state depending on the measurements of other sensors, or users in an online social network may change their political biases depending on their neighbors' postings but they may not wish to express these changes explicitly. The activities or observations of the agents in a network may in turn depend on their latent sentiment or hidden state [4]. Such phenomena call for modeling networked agents using probabilistic models in which an agent's latent/hidden state is coupled to the observed activities of its neighbors.

This work is based on research supported by the U.S. Office of Naval Research (ONR) via grant #N00014-10-1-0140, and the U.S. National Science Foundation (NSF) via grant #1343430.

We assume that observed data are in the form of 'counts' (e.g., the number of postings of a certain category or number of infected people in a contact network). Hidden states are taken as discrete variables. While various types of hidden variable models abound, their application to networked settings brings up special challenges. With directed models [5], inference using exact methods such as those used in Hidden Markov Models (HMMs) becomes intractable when hidden states are coupled across a network. Coupled HMMs, such as the model proposed in [6], where the latent state probabilities are obtained by sampling can be computationally expensive in larger networks. While one can model the neighbor influence by changing the direction of dependency of observations of a HMM as in [7], due to normalization of the transition probabilities at each time step, such a strategy can suffer from label bias [8]. An extension of an HMM which uses neighbor activities as an input was presented in [4], where a threshold determines how the neighbors' actions affect the latent variables. [4] has limitations with estimating the thresholds and generalizing to handle an arbitrary number of hidden states.

We address these limitations via an undirected generative model. To the best of our knowledge, no undirected models have been applied to modeling neighbor influence and hidden variables in networks. In contrast to discriminative undirected models [8, 9], a generative model does not rely on labeled latent states being always available for training. While belief propagation can be used for inference in an undirected model, such a method may not converge for a general network with loops [10, 11]. To overcome this, we propose a variational inference method [5] which is also scalable compared to sampling methods. After forming our model as a member of the exponential family, we show that we can use stochastic approximation to compute the intractable derivatives. This strategy was used to estimate the parameters of our model. In summary, the main contributions of our work include a novel undirected generative model for inferring latent states in a network influenced by neighbors' actions and a computationally efficient inference method scalable to larger networks. We employ sampling only for training the model. The paper is organized as follows: Section 2 formulates the

model, Section 3 presents the variational inference approach, Section 4 provides a training method, Section 5 details the experimental results, and Section 6 concludes.

2. MODELING WITH EXPONENTIAL FACTORS

We assume a directed network $\mathfrak{N}(\mathcal{V}, \mathcal{E})$ with $N =\mid \mathcal{V} \mid$ users. While the proposed model is valid for a dynamic network, for convenience, we assume a static network in this paper. User i's set of neighbors, $\mathcal{N}(i)$ is defined as

$$\mathcal{N}(i) = \{j \in \mathcal{V} \setminus i \mid A_{i,j} = 1\},\ i \in \mathcal{V}, \tag{1}$$

where A is the adjacency matrix. The time duration in which the user activities are observed is divided into T equal time periods with time stamps $\tau_0 \dots \tau_T$. Let X_t^i denote the latent state and Z_t^i be the C dimensional vector of user activity counts of user i during the time period $[\tau_{t-1}, \tau_t)$. Let $\mathcal{I} = \{1, 2, \dots, K\}$ be the set of latent states. We use 1-of-K encoding to denote this latent state. We model the joint distribution of all latent variables $X_{1:T}^{1:N}$ and the observed user activity counts $Z_{1:T}^{1:N}$ through two exponential factors ψ_A and ψ_B. As we need to model temporal dynamics of the latent states influenced by $\mathcal{N}(i)$, the factor ψ_A includes X_{t-1}^i, X_t^i and its neighbor activity counts at $t-1$. We also assume the frequency of activity counts of a user is dependent on its latent state. This is modeled through ψ_B which includes X_t^i and Z_t^i. We combine these two factors for each user across all time durations to obtain the full joint distribution $P(X_{1:T}^{1:N}, Z_{1:T}^{1:N})$.

2.1. Factor for Latent Sentiments and Neighbor Influence

Factor ψ_A is defined as

$$\psi_A(X_{t-1}^i = a,\ X_t^i = b,\ Z_{t-1}^{j \in \mathcal{N}(i)}) = \exp\left(\theta(i)_{a,b}^T \sum_{j \in \mathcal{N}(i)} \frac{Z_{t-1}^j}{\mid \mathcal{N}(i) \mid}\right) \exp(\theta(i)_{a,b,0}), \tag{2}$$

where $\theta(i)_{a,b,0} \in \mathbb{R}$, $\theta(i)_{a,b} \in \mathbb{R}^C$, and $a, b \in \mathcal{I}, t = 2, \dots, T$. The influence of the agents in $\mathcal{N}(i)$ on user i's latent states is incorporated into this factor through the vector parameter $\theta(i)_{a,b}$. The dependency on latent states a, b of this parameter models how neighbor actions affect the dynamics of different latent states. This factor also models how user i's own latent state affects the temporal evolution of future latent states by the parameter $\theta(i)_{a,b,0}$, e.g., for a stubborn user i who disregards $\mathcal{N}(i)$'s actions this can be large.

2.2. Factor for Latent Sentiments and User Activity

Factor ψ_B is defined as

$$\psi_B(X_t^i, Z_t^i) = \frac{1}{\prod_{\nu=1}^{C} Z_t^i(\nu)!} \exp({Z_t^i}^T \boldsymbol{\lambda}(i) X_t^i), \tag{3}$$

where $\boldsymbol{\lambda}(i) \in \mathbb{R}^{C \times |\mathcal{I}|}$ and $t = 1, \dots, T$. The parameter $\boldsymbol{\lambda}(i)$ contributes to the changes in the intensity of each user i's activity Z_t^i depending on X_t^i. Each value in $\boldsymbol{\lambda}(i)$ affects Z_t^i depending on $\nu = 1 \dots C$ and latent state $X_t^i \in \mathcal{I}$. In contrast to the self or mutually exciting point process in recent work [12, 13], we do not assume that the user activities will always trigger more activities. Depending on $\boldsymbol{\lambda}(i)$ and X_t^i, the frequency of these activities can decrease or increase. The form of ψ_B also makes the distribution of user i's activity counts at time t a Poisson distribution conditioned on all the other variables $X_t^j, Z_t^j,\ i \neq j$. This is useful for training as we can directly sample Z_t^i from a Poisson distribution.

2.3. Joint Distribution of Complete Data

Combining the two factors in (2) and (3), the complete joint distribution of $Z_{1:T}^{1:N}$ and $X_{1:T}^{1:N}$ can be obtained as

$$\begin{aligned} &P(X_{1:T}^{1:N}, Z_{1:T}^{1:N} \mid \boldsymbol{\theta}, \boldsymbol{\lambda}) \\ &= \frac{\prod_{i=1}^{N} \psi_b(X_1^i, Z_1^i)}{\exp(A(\boldsymbol{\theta}, \boldsymbol{\lambda}))} \\ &\quad \times \prod_{i=1}^{N} \left\{ \prod_{t=2}^{T} \psi_A(X_t^i, X_{t-1}^i, Z_{t-1}^{j \in \mathcal{N}(i)})\, \psi_B(X_t^i, Z_t^i) \right\}, \end{aligned} \tag{4}$$

where $\boldsymbol{\theta} = \{\theta(1)_{a,b}, \theta(1)_{a,b,0} \dots \theta(N)_{a,b}, \theta(N)_{a,b,0}\}$ and $\boldsymbol{\lambda} = \{\boldsymbol{\lambda}(1) \dots \boldsymbol{\lambda}(N)\}$; $A(\boldsymbol{\theta}, \boldsymbol{\lambda})$ is the logarithm of the normalizing constant, while $\{\boldsymbol{\theta}, \boldsymbol{\lambda}\}$ denotes the set of parameters of our model for all users. Note that the use of linear terms in the exponents without complex feature transformations allows the other relevant tasks such as inference and training to be computationally feasible. Finally, the exponential terms ψ_A and ψ_B can be combined to form an exponential family distribution, i.e.,

$$P(X_{1:N}^{1:T}, Z_{1:N}^{1:T}) = h(Z) \exp\{E_p(X, Z) - A(\boldsymbol{\theta}, \boldsymbol{\lambda})\}, \tag{5}$$

where

$$\begin{aligned} E_p(X, Z) &= \sum_{i=1}^{N} \sum_{t=1}^{T} {Z_t^i}^T \boldsymbol{\lambda}(\boldsymbol{i}) X_t^i \\ &\quad + \sum_{i=1}^{N} \sum_{t=2}^{T} {X_{t-1}^i}^T \Gamma_{t-1}^i(\theta, Z) X_t^i; \\ h(Z) &= \left[\prod_{i=1}^{N} \prod_{t=1}^{T} \prod_{\nu=1}^{C} Z_t^i(\nu)!\right]^{-1}; \\ \Gamma_{t-1}^i(\theta, Z)_{\{a,b\}} &= \hat{\theta}(i)_{a,b}^T \hat{Z}_{t-1}^i,\ a, b \in \mathcal{I}. \end{aligned} \tag{6}$$

Here, $\hat{\theta}_{a,b}(i) \in \mathbb{R}^{C+1}$ and $\hat{Z}_{t-1}^i$, for all i, are defined as

$$\hat{\theta}(i)_{a,b} = \begin{bmatrix} \theta(i)_{a,b,0} \\ \theta(i)_{a,b} \end{bmatrix};\ \hat{Z}_{t-1}^i = \begin{bmatrix} 1 \\ \sum_{j \in \mathcal{N}(i)} \frac{Z_{t-1}^j}{\mid \mathcal{N}(i) \mid} \end{bmatrix}. \tag{7}$$

3. INFERRING LATENT STATES

How can we efficiently compute the probabilities of latent states after observing user activities? The posterior distribution of latent variables conditioned on observed user activity counts is

$$\begin{aligned} P(X_{1:T}^{1:N} \mid Z_{1:T}^{1:N}) &= \frac{P(X_{1:T}^{1:N}, Z_{1:T}^{1:N})}{P(Z_{1:T}^{1:N})} \\ &= \frac{h(Z)\exp(E_p(X,Z))}{\sum_{X_{1:T}^{1:N}} h(Z)\exp(E_p(X,Z))} \\ &= \exp\{E_p(X,Z) - A(\theta,\lambda,Z)\}, \end{aligned} \quad (8)$$

where $A(\theta, \lambda, Z)$ is the normalizing constant of the posterior distribution. While belief propagation algorithms can be used to compute the exact probabilities of a factor model, it cannot be used here since our model is in general not a tree structure [10]. As loopy belief propagation is also not guaranteed to converge [11], we propose a variational inference method [14] as an approximate solution to our inference problem.

3.1. Mean Field Variational Inference

For the variational distribution we assume a fully factored approximation over the latent variables as given in (9), i.e.,

$$Q(X_{1:T}^{1:N} \mid \boldsymbol{\gamma}) = \prod_{i=1}^{N}\prod_{t=1}^{T} q(X_t^i \mid \gamma_t^i), \quad (9)$$

where $q(X_t^i = k \mid \gamma_t^i)$ denotes the probability that the latent state is equal to $k \in \mathcal{I}$ and γ_t^i is the multinomial variational parameter. We then make this approximate distribution as close as possible to (8) (in the KL divergence sense). The KL divergence between $P(X_{1:N}^{1:T} \mid Z_{1:N}^{1:T})$ and $Q(X_{1:T}^{1:N})$ is

$$\begin{aligned} \mathbb{KL}(Q \mid\mid P) &= \mathbb{E}_q\left[\log \frac{Q(X_{1:T}^{1:N})}{P(X_{1:T}^{1:N} \mid Z_{1:T}^{1:N})}\right] \\ &= \sum_{t=1}^{T}\left\{\sum_{i=1}^{N} \mathbb{E}_q[\log q(X_t^i \mid \gamma_t^i)]\right\} \\ &\quad - \mathbb{E}_q[E_p(X,Z)] + A(\theta,\lambda,Z) \end{aligned} \quad (10)$$

where $\mathbb{E}_q$ is the expectation with respect to the variational distribution. We minimize (10) with respect to γ_t^i for all i and t to obtain the best approximation. Substituting (5) for E_p and ignoring constant terms with respect to γ, we get

$$\begin{aligned} \mathbb{KL}(Q||P) = &\sum_{i=1}^{N}\sum_{t=1}^{T} \mathbb{E}_q \log q(X_t^i \mid \gamma_t^i) - {Z_t^i}^T \boldsymbol{\lambda}(i) \mathbb{E}_q[X_t^i] \\ &- \sum_{i=1}^{N}\sum_{t=2}^{T} \mathbb{E}_q[X_{t-1}^i]^T \Gamma_{t-1}^i(\theta, Z) \mathbb{E}_q[X_t^i] \\ &- \sum_{i=1}^{N}\sum_{t=1}^{T} {Z_t^i}^T \boldsymbol{\lambda}(i) \mathbb{E}_q[X_t^i]. \end{aligned} \quad (11)$$

Note that, since we use a fully factored approximation, the expectation terms in the second line above can be separated into two terms. To obtain the coordinate descent equations for the mean field update of $q(X_t^i \mid \gamma_t^i)$ we replace the other variables with their expectations and write the remaining terms as

$$\mathbb{KL}(Q||P) = \mathbb{E}_q \log[q(X_t^i \mid \gamma_t^i)] - \mathbb{E}_q \log f(X_t^i) + \kappa, \quad (12)$$

where κ is a constant and

$$f(X_t^i) = \begin{cases} \exp(\mathbb{E}_q[X_{t-1}^i]^T \Gamma_{t-1}^i(\theta, Z) X_t^i) \\ \times \exp\left({X_t^i}^T \Gamma_t^i(\theta, Z) \mathbb{E}_q[X_{t+1}^i] + {Z_t^i}^T \boldsymbol{\lambda}(\boldsymbol{i}) X_t^i\right), \\ \qquad \text{if } t \geq 2; \\ \exp\left({X_t^i}^T \Gamma_t^i(\theta, Z) \mathbb{E}_q[X_{t+1}^i] + {Z_t^i}^T \boldsymbol{\lambda}(\boldsymbol{i}) X_t^i\right), \\ \qquad \text{otherwise.} \end{cases} \quad (13)$$

Since (12) is again in the form of the KL divergence where the normalizing constant is absorbed into the constant term, it will be minimized when

$$q(X_t^i \mid \gamma_t^i) \propto f(X_t^i). \quad (14)$$

All the variational parameters are updated using (14) with the expectations in $f(X_t^i)$ computed by using the parameters of the most recent updates [5].

4. ESTIMATING PARAMETERS

The model parameters can be estimated using a maximum likelihood strategy. The method proposed in this paper is a supervised training method assuming both latent variables X_t^i and user activity counts Z_t^i are available in the training data. One challenge of our model is computing the derivatives of the normalizing factor $A(\boldsymbol{\theta}, \boldsymbol{\lambda})$ which is also known as the *log partition function* of an exponential family distribution. Since our model is a member of the exponential family, we can approximate the derivatives with respect to the model parameters using stochastic methods. The complete data log-likelihood under this model can be written as

$$\log P(X_{1:T}^{1:N}, Z_{1:T}^{1:N}) = E_p(X,Z) - A(\boldsymbol{\theta}, \boldsymbol{\lambda}) + \log h(Z). \quad (15)$$

Denoting the parameter vector as Λ and defining $L(\Lambda)$ as the negative log-likelihood terms in (15), we get

$$L(\Lambda) = A(\boldsymbol{\theta}, \boldsymbol{\lambda}) - E_p(X,Z). \quad (16)$$

Computing the gradient with respect to Λ we get

$$\frac{\partial}{\partial \Lambda} L(\Lambda) = \frac{\partial}{\partial \Lambda} A(\boldsymbol{\theta}, \boldsymbol{\lambda}) - \frac{\partial}{\partial \Lambda} E_p(X,Z). \quad (17)$$

While the gradients of $E_p(X, Z)$ can be computed, we cannot directly compute the gradients of $A(\boldsymbol{\theta}, \boldsymbol{\lambda})$ due to infinite summations. However we can write it as an expectation [14]:

$$\frac{\partial}{\partial \Lambda} A(\boldsymbol{\theta}, \boldsymbol{\lambda}) = \mathbb{E}_p\left[\frac{\partial}{\partial \Lambda} E_p(X,Z)\right], \quad (18)$$

where $\mathbb{E}_p$ is the expectation with respect to the true joint distribution in (5). Then we approximate this expectation using MCMC methods. Specifically we can write

$$\mathbb{E}_p\left[\frac{\partial}{\partial\Lambda}E_p(X,Z)\right] \approx \frac{1}{S}\sum_s \frac{\partial}{\partial\Lambda}E_p(X_s,Z_s), \qquad (19)$$

where X_s and Z_s are the samples drawn from (5). We employ Gibbs sampling to generate these samples. Finally, we minimize $L(\Lambda)$ using the following form of update:

$$\Lambda_{n+1} = \Lambda_n - \epsilon_n \frac{\partial}{\partial\Lambda}L(\Lambda_n),\ \epsilon_n = \frac{0.01}{1+n*0.01}. \qquad (20)$$

Since $E_p(X,Z)$ is linear in parameters and $A(\boldsymbol{\theta},\boldsymbol{\lambda})$ is convex in Λ [14], this type of an update is guaranteed to converge to a global minimum under certain conditions on ϵ_n [15].

5. EXPERIMENTS

To evaluate our model, we use a synthetic dataset where the ground truth is known. We generate samples of user activity counts and latent states from (5) and this data was used as the training dataset. another set of samples was generated as the testing dataset. The observed user activity counts were used to infer unobserved/hidden states of the agents in a network. We implemented our work using C++ and Python libraries [16, 17]. We compare our method with two baseline methods: **(1)** Cox process [18] which treats each category of user activity is a Poisson process where the underlying intensity is stochastic. The intensity is assumed to be a Markov chain where states correspond to latent states of agents. **(2)** Support vector machine (SVM) was also used to classify latent states. For SVM, the input feature for each variable X_t^i is a $2C$-dimensional vector in which we include the sum of the neighbor activities and user i's own activity counts at time t.

The data for the experiments were generated on an Erdős-Rényi random graph with an edge probability of 0.2. We set $|\mathcal{I}| = 2$ and $C = 2$. The parameters used are

$$\theta(i)_{a,b} = \begin{bmatrix} -0.5(a+b) \\ 0.5(a-2b) \end{bmatrix};\ \boldsymbol{\lambda}(i) = \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}, \qquad (21)$$

where $\theta(i)_{a,b,0} = 0.1,\ \forall a,b \in \mathcal{I},\ i = 1,\ldots,N$.

For the first experiment, we changed the number of users in the network from 100 to 400. Fig. 1 includes the ROC curves for all three methods for $N = 100$ and $T = 10$. The area under the curve of the ROC (AUC-ROC) values appear in Table 1. The AUC-ROC for SVM is close to 0.5 in all the experiments because the SVM assumes all samples are independent. While the Cox process models temporal dynamics thus giving better performance than the SVM, it does not model the coupling among users. The proposed method has AUC-ROC values greater than 0.9 in all the experiments, an indication of its scalability for larger networks.

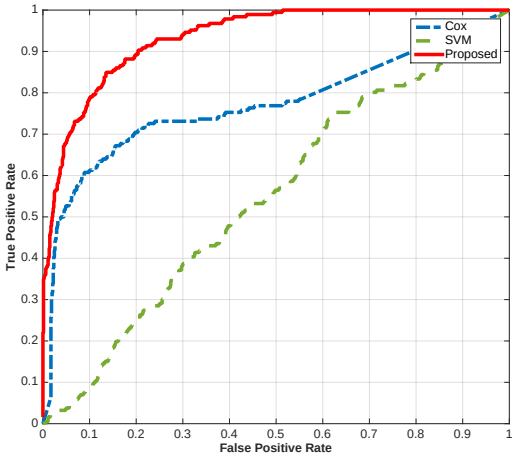

Fig. 1. The ROC curves for $N = 100$ and $T = 10$.

Table 1. AUC-ROC for Different Network Sizes

N	SVM	Cox	Proposed
100	0.55768	0.76397	**0.93413**
200	0.55115	0.74995	**0.93085**
300	0.49945	0.80338	**0.92186**
400	0.50534	0.75932	**0.93517**

For the second experiment, we increased $|\mathcal{I}|$ to 3 keeping the other parameters constant; $\theta(i)_{a,b}$ and $\boldsymbol{\lambda(i)}$ were set to

$$\theta(i)_{a,b} = \begin{bmatrix} \frac{-0.5}{|\mathcal{I}|^2}(|\mathcal{I}|a+b+1) \\ \frac{0.5}{|\mathcal{I}|^2}(|\mathcal{I}|a+b+1) \end{bmatrix};\ \boldsymbol{\lambda}(i) = \begin{bmatrix} -1 & 1 \\ 0.5 & -0.5 \\ 1 & -1 \end{bmatrix}. \qquad (22)$$

The ROC-AUC values computed using the one-vs-rest approach for each latent state appear in Table 2. While there is a variation in performance across different states, the proposed method always performs better than the Cox process.

6. CONCLUSION

We have proposed an undirected generative model for modeling networked data which addresses some of the limitation in existing models of its kind. An efficient inference method and a training method with stochastic approximation were also presented despite the intractabilities involved with this model. The results demonstrate that this model can infer latent states outperforming other baseline methods.

Table 2. ROC Performance for $|\mathcal{I}| = 3$

State	1	2	3
Cox	0.83945	0.60181	0.77642
Proposed	0.91683	0.67330	0.83976

7. REFERENCES

[1] D. Wang, L. Kaplan, and T. F. Abdelzaher, "Maximum likelihood analysis of conflicting observations in social sensing," *ACM Transactions on Sensor Networks*, vol. 10, no. 2, pp. 1–27, Jan. 2014.

[2] V. Krishnamurthy and H. V. Poor, "A tutorial on interactive sensing in social networks," *IEEE Transactions on Computational Social Systems*, vol. 1, no. 1, pp. 3–21, Mar. 2014.

[3] M. M. Mostafa, "More than words: Social networks' text mining for consumer brand sentiments," *Expert Systems with Applications*, vol. 40, no. 10, pp. 4241–4251, Aug. 2013.

[4] V. Raghavan, G. V. Steeg, A. Galstyan, and A. G. Tartakovsky, "Modeling temporal activity patterns in dynamic social networks," *IEEE Transactions on Computational Social Systems*, vol. 1, no. 1, pp. 89–107, Mar. 2014.

[5] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*, MIT Press, 2012.

[6] W. Dong, A. Pentland, and K. Heller, "Graph-coupled HMMs for modeling the spread of infection," in *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, Corvallis, Oregon, 2012, pp. 227–236.

[7] A. McCallum, D. Freitag, and F. C. N. Pereira, "Maximum entropy Markov models for information extraction and segmentation," in *Proceedings of the International Conference on Machine Learning (ICML)*, San Francisco, CA, 2000, pp. 591–598.

[8] J. D. Lafferty, A. McCallum, and F. C. N. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proceedings of the International Conference on Machine Learning (ICML)*, San Francisco, CA, 2001, pp. 282–289.

[9] S. B. Wang, A. Quattoni, L.-P. Morency, and D. Demirdjian, "Hidden conditional random fields for gesture recognition," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, Washington, DC, 2006, vol. 2, pp. 1521–1527.

[10] J. S. Yedidia, W. T. Freeman, and Y. Weiss, "Generalized belief propagation," in *Advances in Neural Information Processing Systems*, T. K. Leen, T. G. Dietterich, and V. Tresp, Eds., vol. 13, pp. 689–695. MIT Press, 2001.

[11] K. P. Murphy, Y. Weiss, and M. I. Jordan, "Loopy belief propagation for approximate inference: An empirical study," in *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, San Francisco, CA, 1999, pp. 467–475.

[12] C. Blundell, K. A. Heller, and J. M. Beck, "Modelling reciprocating relationships with Hawkes processes," in *Proceedings of the International Conference on Neural Information Processing Systems (NIPS)*, Lake Tahoe, NV, Dec. 2012, pp. 2600–2608.

[13] Q. Zhao, M. A. Erdogdu, H. Y. He, A. Rajaraman, and J. Leskovec, "SEISMIC: a self-exciting point process model for predicting tweet popularity," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, 2015.

[14] M. J. Wainwright and M. I. Jordan, "Graphical models, exponential families, and variational inference," *Foundations and Trends in Machine Learning*, vol. 1, no. 1/2, pp. 1–305, 2008.

[15] L. Bottou, "Stochastic gradient descent tricks," in *Neural Networks: Tricks of the Trade, Lecture Notes in Computer Science (LNCS)*, G. Montavon, G. B. Orr, and K.-R. Müller, Eds., vol. 7700, chapter 18, pp. 421–436. Springer, Berlin Heidelberg, Jan. 2012.

[16] Conrad Sanderson and Ryan Curtin, "Armadillo: a template-based C++ library for linear algebra," *The Journal of Open Source Software*, vol. 1, no. 2, June 2016.

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, Oct. 2011.

[18] D. R. Cox, "Some statistical methods connected with series of events," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 17, no. 2, pp. 129–164, 1955.