Deep Generative Models and Biological Applications

by

Kai Fan

Graduate Program in Computational Biology and Bioinformatics
Duke University

Date: ____________________

Approved:

________________________
Katherine Heller, Supervisor

________________________
Lawrence Carin

________________________
Raluca Gordan

________________________
Chris Woods

Dissertation submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in the Graduate Program in Computational Biology and
Bioinformatics
in the Graduate School of Duke University
2017
Abstract

Deep Generative Models and Biological Applications

by

Kai Fan

Graduate Program in Computational Biology and Bioinformatics
Duke University

Date: ______________________

Approved:

__________________________
Katherine Heller, Supervisor

__________________________
Lawrence Carin

__________________________
Raluca Gordan

__________________________
Chris Woods

An abstract of a dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Graduate Program in Computational Biology and Bioinformatics in the Graduate School of Duke University 2017
Abstract

High-dimensional probability distributions are important objects in a wide variety of applications. Generative models provide an excellent manipulation method for training from rich available unlabeled data set and sampling new data points from underlying high-dimensional probability distributions. The recent proposed Variational auto-encoders (VAE) framework is an efficient high-dimensional inference method to modeling complicated data manifold in an approximate Bayesian way, i.e., variational inference. We first discuss how to design fast stochastic backpropagation algorithm for the VAE based amortized variational inference method. Particularly, we propose second order Hessian-free optimization method for Gaussian latent variable models and provide a theoretical justification to the convergence of Monte Carlo estimation in our algorithm. Then, we apply the amortized variational inference to a dynamic modeling application in flu diffusion task. Compared with traditional approximate Gibbs sampling algorithm, we make less assumption to the infection rate.

Differing from the maximum likelihood approach of VAE, Generative Adversarial Networks (GAN) is trying to solve the generation problem from a game theoretical way. From this viewpoint, we design a framework VAE+GAN, by placing a discriminator on top of auto-encoders based model and introducing an extra adversarial loss. The adversarial training induced by the classification loss is to make the discriminator believe the sample from the generative model is as real as the one from the true dataset. This trick can practically improve the quality of generation samples,
demonstrated on images and text domains with elaborately designed architectures. Additionally, we validate the importance of generative adversarial loss with the conditional generative model in two biological applications: approximate Turing pattern PDEs generation in synthetic/system biology, and automatic cardiovascular disease detection in medical imaging processing.
To my family.
Contents

Abstract iv
List of Tables xii
List of Figures xiii
List of Abbreviations and Symbols xv
Acknowledgements xvi

1 Introduction 1

1.1 Probabilistic Generative Models 2
1.1.1 Variational Auto-Encoders 3
1.1.2 Generative Adversarial Nets 5

1.2 Research Questions and Contributions 7
1.2.1 Can the stochastic backpropagation in VAE be more efficient? 7
1.2.2 How to apply VAE in dynamic modeling, such as flu diffusion? 8
1.2.3 To what extent, adversarial loss will help training? 9
1.2.4 How to use adversarial loss in synthetic biology and medical imaging processing? 9

2 Second Order Stochastic Backpropagation for Variational Inference 11

2.1 Stochastic Backpropagation 11
2.1.1 Second Order Gaussian Backpropagation 11
2.1.2 Covariance Parameterization for Optimization 12
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1.3 Apply Reparameterization on Second Order Algorithm</td>
<td>13</td>
</tr>
<tr>
<td>2.1.4 Estimator Variance</td>
<td>16</td>
</tr>
<tr>
<td>2.2 Application on Variational Auto-encoder</td>
<td>17</td>
</tr>
<tr>
<td>2.2.1 Model Description</td>
<td>18</td>
</tr>
<tr>
<td>2.2.2 Deep Neural Networks with Hybrid Hidden Layers</td>
<td>19</td>
</tr>
<tr>
<td>2.3 Experiments</td>
<td>20</td>
</tr>
<tr>
<td>2.3.1 Bayesian Logistic Regression</td>
<td>21</td>
</tr>
<tr>
<td>2.3.2 Variational Auto-encoder</td>
<td>22</td>
</tr>
<tr>
<td>2.4 Summary</td>
<td>26</td>
</tr>
<tr>
<td>3 A Unifying Variational Inference Framework for Hierarchical Graph-</td>
<td>27</td>
</tr>
<tr>
<td>Coupled HMM with an Application to Influenza Infection</td>
<td></td>
</tr>
<tr>
<td>3.1 Background</td>
<td>28</td>
</tr>
<tr>
<td>3.1.1 Notation</td>
<td>28</td>
</tr>
<tr>
<td>3.1.2 Graph-coupled Hidden Markov Model</td>
<td>29</td>
</tr>
<tr>
<td>3.1.3 Generative Modeling</td>
<td>30</td>
</tr>
<tr>
<td>3.1.4 Generalized Baum-Welch Algorithm</td>
<td>32</td>
</tr>
<tr>
<td>3.1.5 Forward-backward Algorithm</td>
<td>33</td>
</tr>
<tr>
<td>3.1.6 Approximate EM Algorithm</td>
<td>34</td>
</tr>
<tr>
<td>3.1.7 A Simulation Study</td>
<td>37</td>
</tr>
<tr>
<td>3.1.8 Extending GCHMMs to Hierarchies</td>
<td>38</td>
</tr>
<tr>
<td>3.1.9 Model Evolution</td>
<td>40</td>
</tr>
<tr>
<td>3.1.10 Approximate Conjugacy</td>
<td>42</td>
</tr>
<tr>
<td>3.1.11 Gibbs Sampling for the Conjugate Part</td>
<td>45</td>
</tr>
<tr>
<td>3.1.12 Burn-in Gibbs EM Algorithm</td>
<td>46</td>
</tr>
<tr>
<td>3.1.13 Short Discussion on Sigmoid link</td>
<td>51</td>
</tr>
<tr>
<td>3.1.14 Further Discussion on $\beta_n$</td>
<td>51</td>
</tr>
</tbody>
</table>
3.2 Variational Inference Approach ........................................ 52
   3.2.1 Limitation of Gibbs Sampling for φ Approximation by Auxiliary Variable ........................................ 53
   3.2.2 Variational Inference ........................................ 55
   3.2.3 Dynamic Recognition Model ........................................ 56
   3.2.4 Parametrization and Optimization ........................................ 57
3.3 Experiments ........................................ 61
   3.3.1 Data Description ........................................ 61
   3.3.2 Evaluation of Gibbs-EM ........................................ 64
   3.3.3 Variational Inference for Different Infection Rates ........................................ 68
   3.3.4 Short Pattern Capture on Real Flu Case ........................................ 71
3.4 Summary ........................................ 72
4 On the Importance of Generative Adversarial Loss ........................................ 74
   4.1 Text GAN ........................................ 75
       4.1.1 Model Description ........................................ 75
       4.1.2 Matching Methods ........................................ 80
       4.1.3 Experimental Evaluation ........................................ 81
   4.2 Image Enhancement via GAN Loss ........................................ 85
       4.2.1 ADMM ........................................ 87
   4.3 Inner-loop free ADMM ........................................ 89
       4.3.1 Amortized inference for x using a conditional proximity operator ........................................ 89
       4.3.2 Inversion-free update of z ........................................ 90
       4.3.3 Adversarial training of cPSDAE ........................................ 92
       4.3.4 Image super-resolution and motion deblurring ........................................ 94
       4.3.5 Joint super-resolution and colorization ........................................ 96
   4.4 Summary ........................................ 97
## List of Tables

2.1 Comparison on misclassification error ............................................. 22
3.1 Notations ....................................................................................... 28
3.2 Coefficients Estimation on exFlu Dataset ................................. 67
3.3 Recall and Accuracy of exFlu Epidemics ...................................... 71
4.1 Quantitative results using BLEU-2,3,4 and KDE. ......................... 83
4.2 Sentences generated by textGAN. ................................................. 84
4.3 Sentences generated by interpolation of latent space .................. 86
List of Figures

1.1 The auto-encoders interpretation of VAE .................................. 3
1.2 The graphical representation of GAN ...................................... 6
2.1 Auto-encoder Model by Deep Neural Nets ............................... 18
2.2 Results for sparse logistic regression ................................. 23
2.3 Results on FreyFace dataset for (a) convergence of lower bound, (b) reconstruction and manifold ................................. 25
2.4 Results on MNIST dataset for (a) convergence of lower bound, (b) reconstruction and manifold ................................. 25
2.5 Results on OliveFace for (a) running time and (b) reconstruction comparison ................................................. 26
3.1 Graphical Representation of GCHMMs ................................. 30
3.2 Factor Graph of Figure 3.1(b) ........................................ 33
3.3 Accuracy with respect to iterations .................................. 37
3.4 The graphical representation of hGCHMM extended from Figure 3.1(b) ................................................. 41
3.5 Evolution from LDA model ............................................ 42
3.6 Comparison between different approximation methods ................. 44
3.7 The error visualization .................................................... 54
3.8 Recognition models, where (b) and (c) omit some side information nodes ................................................. 60
3.9 Overall Social Network ................................................. 62
3.10 Visualization of iEpi Bluetooth Network ............................... 64
3.11 Left is true Onset and its Duration. Right is predicted by Sigmoid Model. ................................................................. 64
3.12 Results of infection rate estimation. ............................................... 65
3.13 Scatter plot of infection rate estimation. ....................................... 66
3.14 (a) Accuracy v.s threshold. (b) ROC curve comparison ................. 69
3.15 Results of latent variable estimation. ........................................ 70
4.1 VAE+GAN framework ........................................................... 75
4.2 Model scheme of TextGAN ...................................................... 76
4.3 Network structure comparison between auto-encoders and Unets (borrowed from Isola et al. (2016)). ................................. 87
4.4 Conditional face generation from line drawing, training with (a) only reconstruction loss (b) reconstruction loss and GAN loss ........... 87
4.5 Network Architecture of Inf-ADMM-ADNN. .............................. 92
4.6 Results of super-resolution and motion deblurring ......................... 95
4.7 Results of joint super-resolution and colorization .......................... 97
5.1 Left: Circuit logic. Right: Self-organized pattern formation in engineered bacteria. ....................................................... 98
5.2 PDE model to account for the circuit dynamics. ............................ 100
5.3 Model structure for ring pattern generation. ................................. 101
5.4 (a) Variation of peak values of RFP concentration. (b) Conditional Generator. ......................................................... 102
5.5 Peak and shape prediction on test dataset. .................................. 103
5.6 (a) Suspected lesions. (b) Contour of lesion blood vessel. ............... 103
List of Abbreviations and Symbols

Symbols

\[ \mathbb{E} \] The expectation operation.
\[ \mathcal{N}(\cdot, \cdot) \] Multivariate Gaussian distribution.
\[ \mathbf{x} \] A column vector.

Abbreviations

Long lines in the symbolist environment are single spaced, like in the other front matter tables.

MCMC Markov chain Monte Carlo
KL Kullback-Leibler
EM Expectation-Maximization
L-BFGS Limited-memory Broyden-Fletcher-Goldfarb-Shanno
HMMs Hidden Markov Models
CDC Center of Disease Control
ADMM Alternative Direction Method of Multipliers
PDEs Partial Differential Equations
SDEs Stochastic Differential Equations
YOLO You Only Look Once
LSTM Long Short Term Memory
Acknowledgements

I felt myself being so blessed for the past four years. First of all, I would like to express my deepest gratitude to my advisor, Professor Katherine Heller, for her invaluable mentoring and support. She is a young but active Bayesian statistician of this generation, and supervised my first machine learning project to guide my next few years research in Bayesian deep generative model. Her generous support for my academic conference traveling provided me many opportunities to communicate with other talent researchers all over the world. Her way of supervision style and friendly communication shaped our research lab to a relatively relaxed atmosphere, allowing a large degree of freedom for my research direction, collaboration, and internship. I was able to develop my own research style and published a lot of papers within four years. These are beneficial not only for my study and career development, but also for my whole life.

Fortunately, I also met a lot of incredible people, making my Ph.D. life an invaluable memory. I would like to express my deep gratefulness to all these people. I will thank Professor Scott Schmidler and Professor Raluca Gordan gave me two rotation opportunities to work in their labs. It was my first time to dive into Markov Chain Monte Carlo research in Scott’s group, and it was my first attempt to use deep learning model to biological application. I would also like to my other thesis committee members, Professor Sayan Mukherjee and Professor Chris Woods. Sayan was the first professor I met and talked with in a machine learning conference in
China. Without him and the nice conversation, I won’t come to Duke. Dr. Woods also gave me many suggestions on epidemiology for my first project on flu tracking, and some insightful advice on my gene circuit simulation project.

Many special thanks go to my fellows and colleagues. I want to thank other Qi Wei, Xiangyu Wang, Fangjian Guo, Yizhe Zhang, and Ricardo Henao to be my frequent co-authors. I really enjoy the collaboration and brainstorming with such talent colleagues. I would like to thank all the extremely talented people in the Heller’s group for numerous conversations and discussions that inspired my research. I want to thank many master students in my machine learning class. Their curiosity and questions made me to fully prepare my teaching notes, that may be very beneficial to my future academic career. I also want to thank everyone at Information Initiative at Duke and Department of Statistical Science.

I would thank my internship mentors at Microsoft Research, Nicolo Fusi, Hao Ma, and Jianfeng Gao. They directed my research from a more applicable way, broadening my industrial horizon. I also thank the sponsors of KDD, NIPS, ICML, and ICDM to support my student travel award.

Finally, I want to express my deepest thankfulness to my parents. I thank everything they have done for me and filling my life with happiness.
Generative models, usually belonging to the unsupervised learning domain, have become ubiquitous in machine learning and statistics, and are now widely used in fields such as bioinformatics, computer vision, or natural language processing, since the reachable large scale unlabeled data. Bayesian statistics is a natural way to utilize the framework of generative models. However, posterior inference usually brings an intractable integration due to the normalizing constant in many complicated non-conjugate models, while Markov chain Monte Carlo (MCMC) Neal (1993), such as Metropolis-Hastings (MH) algorithm, is commonly adopted to obtain a sequence random samples from the probability distribution for which direct sampling is difficult. Though MH eventually converges to the desired distribution and is an asymptotic exact inference, it is often suffering slow-mixing rate and requires large samples for model averaging. Variational Bayes, sometimes referred to as variational inference (VI), is an alternative for approximate inference. In this thesis, we will focus on a particular VI method, amortized variational inference, where variational auto-encoders (VAE) Kingma and Welling (2014) is the classic probabilistic model that will be extensively discussed. Whether MCMC or VI is developed to provide
an explicit density function, another type of generative models can be trained without even needing to explicitly define a density function. Generative adversarial nets (GAN) Goodfellow et al. (2014), that will be discussed in this thesis as well, are designed to be an implicit generative model with many nice theoretical and practical properties, such as scaling to high dimensional space, or efficient sampling scheme. These two models benefit from being more interpretable than other deep learning models and easily extended to other downstream semi-supervised/supervised learning tasks. In other words, the flexibility of generative models allows them to become discriminative models with the advantage of pre-training and simple modifications.

1.1 Probabilistic Generative Models

In the field of machine learning, we often prefer to define a probabilistic model that can characterize our interests with respect to the data, based on our prior knowledge. From the perspective of Bayesian statistics, we believe the data generation process follows a hierarchical structure, indicating the observed data coming from an unknown high level latent variable. Concretely, suppose \( x \) and \( z \) represent the observed variable and the latent variable respectively, the mathematical description of the joint distribution of them is

\[
p_{\theta}(x, z) = p_{\theta}(x|z)p_{\theta}(z)
\] (1.1)

where \( p_{\theta}(z) \) and \( p_{\theta}(x|z) \) represent the prior distribution for latent variable and the data generation process. Based on the observations, we usually interest the posterior distribution of latent variable,

\[
p_{\theta}(z) = \frac{p_{\theta}(x|z)p_{\theta}(z)}{p_{\theta}(x)} = \frac{p_{\theta}(x|z)p_{\theta}(z)}{\int p_{\theta}(x|z)p_{\theta}(z)dz}
\] (1.2)

Notice that the marginal distribution over the observed variable \( p_{\theta}(x) \), taking as a function of \( \theta \), involves an integration over latent variable. This distribution is not
identical to the underlying true data distribution \( p_{data}(x) \), however, it can be very complex such as the expressivity can arbitrarily approximate any desired distribution. For example, if the prior of \( z \) follows categorical distribution and \( x|z \) follows a Gaussian distribution, we obtain a common generative model, Gaussian mixture models; if \( z \) follows continuous distribution, more advanced and efficient generative models are developed recently.

The main difficulty to do inference on the posterior is typically the intractability of marginal probability of data under many models, i.e., the integral for computing model evidence. If there is no analytical form of the denominator in Eq (1.2), we cannot differentiate it with respect to the parameters \( \theta \). However, when we care more about the generative model itself, we can just obtain the point estimate by maximizing \( p_{\theta}(x|z) \) directly, which is fast but often suffers from the overfitting problem. An alternative is to use Markov Chain Monte Carlo (MCMC), which is asymptotically unbiased, but it is expensive and slow to assess convergence in high dimensional case. This is also the reason why this thesis will focus on the VAE and GAN frameworks, that are suitable for both large scale model and data.

1.1.1 Variational Auto-Encoders

![Figure 1.1: The auto-encoders interpretation of VAE.](image)

Besides MAP and MCMC, another popular Bayesian inference method is Variational Bayes (VB), being able to provide an analytical approximation to the posterior probability of the latent variables. VAE usually refers to deep latent variable model
with VB inference framework, while sometimes it also indicates the amortized variational inference method that can be applied to many other machine learning models. In this section, we will provide a brief review of the VAE, and in the main thesis we will discuss a novel stochastic backpropagation algorithm that can be applied to the VAE model and use this amortized variational inference to solve a dynamic modeling problem of flu diffusion application.

In VAE, we introduce another approximate posterior (or recognition model corresponding to generative model) \( q_\phi(z|x) \) indicated by variational parameters \( \phi \). If we consider this model as an encoder and the generative model as a decoder, the two joint models forms the structure of auto-encoders in Fig 1.1.1. The objective is to minimize the KL divergence

\[
\arg\min_{\phi, \theta} D_{KL} \left( q_\phi(z|x) \| p_\theta(z|x) \right)
\]

where the KL divergence is non-negative and the minimum is achieved if and only if \( q_\phi(z|x) = p_\theta(z|x) \). However, directly optimization on Eq (1.3) is intractable due to the existence of \( p_\theta(z|x) \). Fortunately, by the non-negative property of KL divergence, we can derive the following inequality (which can also be derived through Jensen’s Inequality) and a surrogate optimization objective.

\[
\log p_\theta(x) = D_{KL} \left( q_\phi(z|x) \| p_\theta(z|x) \right) + \mathbb{E}_{q_\phi(z|x)} \left[ \log \frac{p_\theta(x, z)}{q_\phi(z|x)} \right] \geq \mathbb{E}_{q_\phi(z|x)} \left[ \log \frac{p_\theta(x, z)}{q_\phi(z|x)} \right]
\]

(1.4)

By convention, the lower bound of data log-likelihood in Eq (1.4) is call evidence lower bound (ELBO), denoted as \( \mathcal{L}(\theta, \phi; x) \).

Therefore, our objective is to maximize the ELBO with respect to generative model parameters and variational parameters concurrently. The gap between the ELBO and the marginal likelihood is controlled by the KL divergence in Eq (1.3). The larger we can push the ELBO up, the smaller the gap between \( q_\phi(z|x) \) and true
posterior distribution is. The nice property of $\mathcal{L}(\theta, \phi; x)$ is that it only involved probability we have already defined, if we make another transformation.

$$
\mathcal{L}(\theta, \phi; x) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{\text{KL}}(q_\phi(z|x) \| p_\theta(z))
$$

where the first term is expected log-likelihood and the second term is the KL distance between approximate proposed posterior and generative model prior. The connection to traditional auto-encoders is that the first term is actually the reconstruction error while the second term plays the role of regularization for latent representation.

The traditional VB inference falls into the EM algorithm by alternatively updating the generative model parameters and the variational parameters. However, in VAE framework, if the prior and approximate posterior are chosen appropriately, these parameters can be jointly and efficiently learned by stochastic backpropagation, due to the reparameterization trick. Since the numerical approximation of $\mathcal{L}(\theta, \phi; x)$ involves two Monte Carlo integration with respect to $q_\phi(z|x)$, this trick is used to draw samples from approximate posterior and obtain unbiased and low variance estimator of ELBO. Another main difference from traditional VB is that the variational distribution $q_\phi(z|x)$ is in the amortized formulation that depends on every single data point, whereas the $q_\phi(z)$ is frequently defined. The variation of latent variable in the latter is directly controlled by model parameters $\phi$, thus having the disadvantage that the number of variational parameter is scaled with the data size. In contrast, the amortized variational inference enables one set of shared parameter for all data points, indicating the variation of latent variable is controlled by the variation of the data.

1.1.2 Generative Adversarial Nets

It is noticed that VAE does not directly minimize the distance between marginalized distribution $p_\theta(x)$ and the true data distribution $p_{\text{data}}(x)$. By explicitly defining highly flexible priors or highly flexible approximate posteriors, VAE can empirically
obtain the result close to their own log-likelihood. However, the main drawback of variational methods is that, it is difficult to know whether the model we used is complicated enough. If not, even with a perfect optimization algorithm and infinite training data, the gap between $\mathcal{L}$ and the true likelihood can result in $p_\theta$ learning something other than the true $p_{\text{data}}$. By this motivation, GAN is proposed by implicitly defining a model, and offers a way to train the model while interacting only indirectly with $p_\theta$, usually by generating one sample from it at one single step.

![Figure 1.2: The graphical representation of GAN.](image)

Notice the generator $G_\theta(z)$ of GAN shown in Fig 1.2 shares the same architecture with VAE. However, the recognition model from samples in dataset to latent variable is removed. Instead, another discriminator $D_\psi(x)$ is introduced to distinguish the sample fed from generative model or real dataset. The objective is to confusing the discriminator s.t. it believes the sample from generator is statistically identical to the one from dataset. Therefore, the problem can be formulate a minimax game.

$$\min_{\theta} \max_{\psi} V(\theta, \psi) = \mathbb{E}_{p_{\text{data}}(x)}[\log D_\psi(x)] + \mathbb{E}_{p(z)}[\log(1 - D_\psi(G_\theta(z)))] \quad (1.6)$$

For above problem, if we define $C(\theta) = \max_{\psi} V(\theta, \psi)$, the global minimum of the virtual training criterion $C(\theta)$ is achieved if and only if $p_\theta = p_{\text{data}}$. The theoretical
guarantee of this claim is that \( C(\theta) = -\log 4 + D_{JSD}(p_{\text{data}}\|p_\theta) \), where JSD is the Jensen-Shannon divergence \(^1\) that is another measurement between the distance of two distributions.

In practice, the optimization of the loss function can alternatively update the discriminator by ascending its stochastic gradient and update the generator by descending its stochastic gradient. Particularly, the original loss function may not provide sufficient gradient for \( G \) to learn well in the early learning stage. Since \( G \) is usually poor but \( D \) can be learned easily and reject samples from \( G \) with high probability, it will result in saturated \( \log(1 - D(G(z))) \). Instead, in order to provide much stronger gradients during training, maximizing \( \log D(G(z)) \) makes the algorithm more efficient and converges faster, but also leads to the same fixed point of the defined dynamics.

1.2 Research Questions and Contributions

In this section, we formulate the research questions and contributions of this thesis.

1.2.1 Can the stochastic backpropagation in VAE be more efficient?

In Chapter 2, the standard reparameterization trick for VAE inference will be generalized to the second order case. Inspired by the property of location scale families, we show how to reduce the computational cost of the Hessian or Hessian-vector product, thus allowing for a second order stochastic optimization scheme for variational inference under Gaussian approximation. In conjunction with the Hessian-free optimization, we propose an efficient and scalable second order stochastic Gaussian backpropagation for variational inference. Alternately, L-BFGS version, a quasi-Newton method merely using the gradient information, is a natural generalization of first order variational inference. The most immediate application would be to look at

\(^1\) \( D_{JSD}(p\|q) = D_{KL}(\frac{p + q}{2}) + D_{KL}(\frac{p + q}{2}) \)
obtaining better optimization algorithms for variational auto-encoders. At the time this paper published, the model applying second order information in variational inference is LDA, where the Hessian is easy to compute. In general, for non-linear factor models like non-linear factor analysis or the deep latent Gaussian models this is not the case. There has very less systematic investigation into the properties of various optimization algorithms and how they might impact the solutions to optimization problem arising from variational approximations. This chapter is based on our research Fan et al. (2015b, 2016a).

1.2.2 How to apply VAE in dynamic modeling, such as flu diffusion?

As we discussed in previous section, VAE also represents the methodology of amortized variational inference. In Chapter 3, we will investigate the power of VAE as an inference method in dynamic temporal modeling, such as flu diffusion. The mainstay of computational epidemiology research for predicting patterns or discovering the risks of infectious disease has been at the population level. A prominent example is the Google flu trend website, which, leveraging the temporal searching key words associating with flu, forecasted the flu outbreak two weeks ahead of CDC. However, the advent of personal health apps in mobile or wearable devices allows disease diffusion to be modeled in an individual level, e.g. data from cell phones is harnessed to identify the probability of contracting flu for every single person in a relatively close community. They both incorporate the face-to-face contact information within a local area to adjust the transmission function and construct their models, thus constructing a hierarchical model with extra features with an enriched epidemiology dataset for more accurate prediction. The main idea in these works is to build a model in a fully Bayesian setting taking the advantage of efficient Gibbs sampling. However, the nice conjugacy property of such complicated models, to a great extent, relies on a small infection rate since a Taylor expansion trick is applied. We
are aiming to break this assumption and develop a unifying inference method for a non-conjugate disease diffusion model with the help of VAE. This chapter is based on our research Fan et al. (2015c,a, 2016b).

1.2.3 To what extent, adversarial loss will help training?

One of main contribution of GAN is to introduce the adversarial loss that may help any generative models to produce more realistic samples. For example, on top of VAE loss, an extra adversarial loss will make the generated images become less blur and sharper. In Chapter 4, we also validate this claim in a neural ADMM framework, a learning-to-learn algorithm by utilizing auxiliary deep neural networks to solve the inverse problem in signal processing. The applications, such as super-resolution, motion-deblurring or colorization, are very suitable for incorporating GAN loss for further image enhancement, since synthesizing sharper and clearer samples based on degraded input coincides with the advantage of GAN model. This chapter is based on our research Zhang et al. (2017); Wei et al. (2017).

1.2.4 How to use adversarial loss in synthetic biology and medical imaging processing?

Mechanism-based mathematical models are the foundation for diverse applications in science and engineering. It is often critical to explore the massive parametric space for each model. For certain models, e.g., agent-based, PDEs, and SDEs, this practice can impose a prohibitive barrier for practical applications even when computer clusters are used. In Chapter 5, we develop a fundamentally new framework to improve the computational efficiency by orders of magnitude, to overcome the limitation. The key concept is to train a conditional GAN using a limited number of simulations generated by well-defined mechanism-based models. The number of simulations is small enough such that the simulations are still manageable, but large enough to train the GAN sufficiently well to make reliable predictions. Then, the trained GAN
will be used to explore the system dynamics in a much larger parametric space. In addition to the framework, we also illustrate the application of this approach using several hand-on examples in synthetic biology design and in exploring stochastic dynamics of complex networks. Our work can potentially be a platform for faster pattern screening, cell strain identification as well as new drug development.

Besides the system biology application, we use the conditional generative model to provide a down-stream task for cardiovascular lesion detection in Chapter 6. We first apply YOLO object detection algorithm to extract a rectangle area that contains suspect lesion blood vessel. Then an image-to-image translation model based on conditional GAN is applied to extract the contour or the edge of blood vessel, thus providing a diagnosis reference to the doctor.

The thesis concludes in Chapter 7 with some summary comments and points to open questions and potential future research, and all the technical proofs are deferred to the appendix.
Second Order Stochastic Backpropagation for Variational Inference

2.1 Stochastic Backpropagation

In this section, we extend the Bonnet and Price theorem Bonnet (1964); Price (1958) to develop 2nd order Gaussian backpropagation. Specifically, we consider how to optimize an expectation of the form $\mathbb{E}_{q_\theta}[f(z|x)]$, where $z$ and $x$ refer to latent variables and observed variables respectively, and expectation is taken w.r.t distribution $q_\theta$ and $f$ is some smooth loss function (e.g. it can be derived from a standard variational lower bound Beal (2003)). Sometimes we abuse notation and refer to $f(z)$ by omitting $x$ when no ambiguity exists. To optimize such expectation, gradient decent methods require the 1st derivatives, while Newton’s methods require both the gradients and Hessian involving 2nd order derivatives.

2.1.1 Second Order Gaussian Backpropagation

If the distribution $q$ is a $d_z$-dimensional Gaussian $\mathcal{N}(z|\mu, C)$, the required partial derivative is easily computed with a lower algorithmic cost $\mathcal{O}(d_z^2)$ Rezende et al. (2014). By using the property of Gaussian distribution (proof in supplementary), we
can compute the 2nd order partial derivative of $\mathbb{E}_{N(z|\mu,C)}[f(z)]$ as follows:

\begin{align}
\nabla_{\mu_{i},\mu_{j}}^{2} \mathbb{E}_{N(z|\mu,C)}[f(z)] &= \mathbb{E}_{N(z|\mu,C)}[\nabla_{z_{i},z_{j}}^{2} f(z)] = 2 \nabla_{C_{i,j}} \mathbb{E}_{N(z|\mu,C)}[f(z)], \quad (2.1) \\
\nabla_{C_{i,j},C_{k,l}}^{2} \mathbb{E}_{N(z|\mu,C)}[f(z)] &= \frac{1}{4} \mathbb{E}_{N(z|\mu,C)}[\nabla_{z_{i},z_{j},z_{k},z_{l}}^{4} f(z)], \quad (2.2) \\
\nabla_{\mu_{i},C_{k,l}}^{2} \mathbb{E}_{N(z|\mu,C)}[f(z)] &= \frac{1}{2} \mathbb{E}_{N(z|\mu,C)}[\nabla_{z_{i},z_{j}}^{3} f(z)]. \quad (2.3)
\end{align}

Eq. (2.1), (2.2), (2.3) have the nice property that a limited number of samples from $q$ are sufficient to obtain unbiased gradient estimates. However, note that Eq. (2.2), (2.3) needs to calculate the third and fourth derivatives of $f(z)$, which is highly computationally inefficient. To avoid the calculation of high order derivatives, we use a co-ordinate transformation.

### 2.1.2 Covariance Parameterization for Optimization

By constructing the linear transformation (a.k.a. reparameterization) $z = \mu + R\epsilon$, where $\epsilon \sim \mathcal{N}(0, I_{d_{e}})$, we can generate samples from any Gaussian distribution $\mathcal{N}(\mu, C)$ by simulating data from a standard normal distribution, provided the decomposition $C = RR^{\top}$ holds. This fact allows us to derive the following theorem indicating that the computation of 2nd order derivatives can be scalable and programmed to run in parallel.

**Theorem 1 (Fast Derivative).** If $f$ is a twice differentiable function and $z$ follows Gaussian distribution $\mathcal{N}(\mu, C)$, $C = RR^{\top}$, where both the mean $\mu$ and $R$ depend on a $d$-dimensional parameter $\theta = (\theta_{l})_{l=1}^{d}$, i.e. $\mu(\theta), R(\theta)$, we have \[ \nabla_{\mu,R}^{2} \mathbb{E}_{N(\mu,C)}[f(z)] = \mathbb{E}_{\epsilon \sim \mathcal{N}(0,I_{d_{e}})}[\epsilon^{T} \otimes H], \] and \[ \nabla_{R}^{2} \mathbb{E}_{N(\mu,C)}[f(z)] = \mathbb{E}_{\epsilon \sim \mathcal{N}(0,I_{d_{e}})}[(\epsilon \epsilon^{T}) \otimes H]. \] This then implies

\begin{align}
\nabla_{\theta_{l}} \mathbb{E}_{N(\mu,C)}[f(z)] &= \mathbb{E}_{\epsilon \sim \mathcal{N}(0,1)}[g^{\top} \frac{\partial(\mu + R\epsilon)}{\partial \theta_{l}}], \quad (2.4) \\
\nabla_{\theta_{l_{1}},\theta_{l_{2}}}^{2} \mathbb{E}_{N(\mu,C)}[f(z)] &= \mathbb{E}_{\epsilon \sim \mathcal{N}(0,1)} \left[ \frac{\partial(\mu + R\epsilon)^{\top}}{\partial \theta_{l_{1}}} H \frac{\partial(\mu + R\epsilon)}{\partial \theta_{l_{2}}} + g^{\top} \frac{\partial^{2}(\mu + R\epsilon)}{\partial \theta_{l_{1}} \partial \theta_{l_{2}}} \right]. \quad (2.5)
\end{align}
where $\otimes$ is Kronecker product, and gradient $g$, Hessian $H$ are evaluated at $\mu + \Re e$ in terms of $f(z)$.

If we consider the mean and covariance matrix as the variational parameters in variational inference, the first two results w.r.t $\mu, R$ make parallelization possible, and reduce computational cost of the Hessian-vector multiplication due to the fact that $(A^T \otimes B)\text{vec}(V) = \text{vec}(AVB)$. If the model has few parameters or a large resource budget (e.g. GPU) is allowed, Theorem 1 launches the foundation for exact 2nd order derivative computation in parallel. In addition, note that the 2nd order gradient computation on model parameter $\theta$ only involves matrix-vector or vector-vector multiplication, thus leading to an algorithmic complexity that is $O(d^2)$ for 2nd order derivative of $\theta$, which is the same as 1st order gradient Rezende et al. (2014). The derivative computation at function $f$ is up to 2nd order, avoiding to calculate 3rd or 4th order derivatives. One practical parametrization assumes a diagonal covariance matrix $C = \text{diag}\{\sigma_1^2, ..., \sigma_d^2\}$. This reduces the actual computational cost compared with Theorem 1, albeit the same order of the complexity ($O(d^2)$) (see supplementary material). Theorem 1 holds for a large class of distributions in addition to Gaussian distributions, such as student $t$-distribution. If the dimensionality $d$ of embedded parameter $\theta$ is large, computation of the gradient $G_\theta$ and Hessian $H_\theta$ (differ from $g$, $H$ above) will be linear and quadratic w.r.t $d$, which may be unacceptable. Therefore, in the next section we attempt to reduce the computational complexity w.r.t $d$.

2.1.3 Apply Reparameterization on Second Order Algorithm

In standard Newton’s method, we need to compute the Hessian matrix and its inverse, which is intractable for limited computing resources. Martens (2010) applied Hessian-free (HF) optimization method in deep learning effectively and efficiently. This work largely relied on the technique of fast Hessian matrix-vector multiplication Pearlmutter (1994). We combine reparameterization trick with Hessian-free or quasi-
Newton to circumvent matrix inverse problem.

**Hessian-free** Unlike quasi-Newton methods HF doesn’t make any approximation on the Hessian. HF needs to compute \( \nabla F(\theta) \), where \( \nabla F(\theta) \) is any vector that has the matched dimension to \( \nabla \), and then uses conjugate gradient algorithm to solve the linear system \( \nabla F(\theta) v = -\nabla F(\theta)^\top v \), for any objective function \( F \). Martens (2010) gives a reasonable explanation for Hessian free optimization. In short, unlike a pre-training method that places the parameters in a search region to regularize Erhan et al. (2010), HF solves issues of pathological curvature in the objective by taking the advantage of rescaling property of Newton’s method. By definition \( \nabla F(\theta) v = \lim_{\gamma \to 0} \frac{\nabla F(\theta + \gamma v) - \nabla F(\theta)}{\gamma} \) indicating that \( \nabla F(\theta) v \) can be numerically computed by using finite differences at \( \gamma \). However, this numerical method is unstable for small \( \gamma \).

In this section, we focus on the calculation of \( \nabla F(\theta + \gamma v) \) by leveraging a reparameterization trick. Specifically, we apply an \( \mathcal{R} \)-operator technique Pearlmutter (1994) for computing the product \( \nabla F(\theta + \gamma v) \) exactly. Let \( F = \mathbb{E}_{N(\mu, C)}[f(z)] \) and reparametrize \( z \) again as Sec. 2.1.2, we do variable substitution \( \theta \leftarrow \theta + \gamma v \) after gradient Eq. (2.4) is obtained, and then take derivative on \( \gamma \). Thus we have the following analytical expression for Hessian-vector multiplication:

\[
\nabla F(\theta + \gamma v) \bigg|_{\gamma=0} = \mathbb{E}_{N(\mu, C)} \left[ \frac{\partial}{\partial \gamma} \left( \nabla F(\theta + \gamma v) \bigg|_{\gamma=0} \right) \right].
\]

Eq. (2.6) is appealing since it does not need to store the dense matrix and provides an unbiased \( \nabla F(\theta) v \) estimator with a small sample size. In order to conduct the 2\textsuperscript{nd} order optimization for variational inference, if the computation of the gradient for variational lower bound is completed, we only need to add one extra step for gradient evaluation via Eq. (2.6) which has the same computational complexity as Eq. (2.4).
This leads to a Hessian-free variational inference method described in Algorithm 1.

For the worst case of HF, the conjugate gradient (CG) algorithm requires at most \(d\) iterations to terminate, meaning that it requires \(d\) evaluations of \(H_\theta v\) product. However, the good news is that CG leads to good convergence after a reasonable number of iterations. In practice we found that it may not necessary to wait CG to converge. In other words, even if we set the maximum iteration \(K\) in CG to a small fixed number (e.g., 10 in our experiments, though with thousands of parameters), the performance does not deteriorate. The early stoping strategy may have the similar effect of Wolfe condition to avoid excessive step size in Newton’s method. Therefore we successfully reduce the complexity of each iteration to \(O(Kd^2)\), whereas \(O(dd^2)\) is for one SGD iteration.

**L-BFGS** Limited memory BFGS utilizes the information gleaned from the gradient vector to approximate the Hessian matrix without explicit computation, and we can readily utilize it within our framework. The basic idea of BFGS approximates Hessian by an iterative algorithm

\[
B_{t+1} = B_t + \Delta G_t \Delta G_t^\top / \Delta \theta_t \Delta \theta_t^\top - B_t \Delta \theta_t \Delta \theta_t^\top B_t / \Delta \theta_t^\top B_t \Delta \theta_t,
\]

where \(\Delta G_t = G_t - G_{t-1}\) and \(\Delta \theta_t = \theta_t - \theta_{t-1}\). By Eq. (2.4), the gradient \(G_t\) at each iteration can be obtained without any difficulty. However, even if this low rank approximation to the Hessian is easy to invert analytically due to the Sherman-Morrison formula, we still need to store the matrix. L-BFGS will further implicitly approximate this dense \(B_t\) or \(B_t^{-1}\) by tracking only a few gradient vectors and a short history of parameters and therefore has a linear memory requirement. In general, L-BFGS can perform a sequence of inner products with the \(K\) most recent \(\Delta \theta_t\) and \(\Delta G_t\), where \(K\) is a predefined constant (10 or 15 in our experiments). Due to the space limitations, we omit the details here but none-the-less will present this algorithm in experiments section.
Algorithm 1 Hessian-free Algorithm on Stochastic Gaussian Variational Inference (HFSGVI)

**Parameters:** Minibatch Size $B$, Number of samples to estimate the expectation $M$ (= 1 as default).

**Input:** Observation $X$ (and $Y$ if required), Lower bound function $L = \mathbb{E}_{\mathcal{N}(\mu, \Sigma)}[f_L]$

**Output:** Parameter $\theta$ after having converged.

for $t = 1, 2, \ldots$ do

1. Randomly draw $B$ datapoints from full data set $X$;
2. Sample $M$ times from $\mathcal{N}(0, I)$ for each $x_b$;
3. Define gradient $G(\theta) = \frac{1}{M} \sum_b \sum_{m_b} g_{b,m} \frac{\partial (\mu + R\epsilon_{m_b})}{\partial \theta}$,
4. Define function $B(\theta, v) = \nabla_{\gamma} G(\theta + \gamma v)|_{\gamma=0}$, where $v$ is a $d$-dimensional vector;

Using Conjugate Gradient algorithm to solve linear system: $B(\theta_t, p_t) = -G(\theta_t)$;

end for

return: $\hat{\theta}$

2.1.4 Estimator Variance

The framework of stochastic backpropagation Kingma et al. (2014); Kingma and Welling (2014); Mnih and Gregor (2014); Rezende et al. (2014) extensively uses the mean of very few samples (often just one) to approximate the expectation. Similarly we approximate the left side of Eq. (2.4), (2.5), (2.6) by sampling few points from the standard normal distribution. However, the magnitude of the variance of such an estimator is not seriously discussed. Rezende et al. (2014) simply explored the variance quantitatively for separable functions. Mnih and Gregor (2014) merely borrowed the variance reduction technique from reinforcement learning by centering the learning signal in expectation and performing variance normalization. Here, we will generalize the treatment of variance to a broader family, Lipschitz continuous function.

**Theorem 2 (Variance Bound).** If $f$ is an $L$-Lipschitz differentiable function and $\epsilon \sim \mathcal{N}(0, I_d)$, then $\mathbb{E}[(f(\epsilon) - \mathbb{E}[f(\epsilon)])^2] \leq \frac{L^2 \pi^2}{4}$.

The proof of Theorem 2 (see Appendix) employs the properties of sub-Gaussian...
distributions and the duplication trick that are commonly used in learning theory. Significantly, the result implies a variance bound independent of the dimensionality of Gaussian variable. Note that from the proof, we can only obtain the $\mathbb{E} \left[ e^{\lambda (f(\epsilon) - \mathbb{E}[f(\epsilon)])} \right] \leq e^{L^2 \lambda^2 \sigma^2 / 8}$ for $\lambda > 0$. Though this result is enough to illustrate the variance independence of $d_z$, we can in fact tighten it to a sharper upper bound by a constant scalar, i.e. $e^{\lambda^2 L^2 / 2}$, thus leading to the result of Theorem 2 with $\text{Var}(f(\epsilon)) \leq L^2$. If all the results above hold for smooth (twice continuous and differentiable) functions with Lipschitz constant $L$ then it holds for all Lipschitz functions by a standard approximation argument. This means the condition can be relaxed to Lipschitz continuous function.

**Corollary 3 (Bias Bound).** $\mathbb{P} \left( \left| \frac{1}{M} \sum_{m=1}^{M} f(\epsilon_m) - \mathbb{E}[f(\epsilon)] \right| \geq t \right) \leq 2e^{-\frac{2M^2 \sigma^2}{\pi^2 L^2}}$.

It is also worth mentioning that the significant corollary of Theorem 2 is probabilistic inequality to measure the convergence rate of Monte Carlo approximation in our setting. This tail bound, together with variance bound, provides the theoretical guarantee for stochastic backpropagation on Gaussian variables and provides an explanation for why a unique realization ($M = 1$) is enough in practice. By reparametrization, Eq. (2.4), (2.5), (2.6) can be formulated as the expectation w.r.t. the isotropic Gaussian distribution with identity covariance matrix leading to Algorithm 1. Thus we can rein in the number of samples for Monte Carlo integration regardless dimensionality of latent variables $z$. This seems counter-intuitive. However, we notice that larger $L$ may require more samples, and Lipschitz constants of different models vary greatly.

### 2.2 Application on Variational Auto-encoder

Note that our method is model free. If the loss function has the form of the expectation of a function w.r.t latent Gaussian variables, we can directly use Algorithm 1.
In this section, we put the emphasis on a standard framework VAE model Kingma and Welling (2014) that has been intensively researched.

### 2.2.1 Model Description

Suppose we have \( N \) i.i.d. observations \( X = \{x^{(i)}\}_{i=1}^N \), where \( x^{(i)} \in \mathbb{R}^D \) is a data vector that can take either continuous or discrete values. In contrast to a standard auto-encoder model constructed by a neural network with a bottleneck structure, VAE describes the embedding process from the prospective of a Gaussian latent variable model. Specifically, each data point \( x \) follows a generative model \( p_\psi(x|z) \), where this process is actually a decoder that is usually constructed by a non-linear transformation with unknown parameters \( \psi \) and a prior distribution \( p_\psi(z) \). The encoder or recognition model \( q_\phi(z|x) \) is used to approximate the true posterior \( p_\psi(z|x) \), where \( \phi \) is similar to the parameter of variational distribution. As suggested in Kingma et al. (2014); Kingma and Welling (2014); Rezende et al. (2014), multi-layered perceptron (MLP) is commonly considered as both the probabilistic encoder and decoder. We will later see that this construction is equivalent to a variant deep neural networks under the constrain of unique realization for \( z \). For this model and each datapoint,
the variational lower bound on the marginal likelihood is,

$$\log p_\psi(x^{(i)}) \geq \mathbb{E}_{q_\psi(z|x^{(i)})} [\log p_\psi(x^{(i)} | z)] - D_{KL}(q_\psi(z|x^{(i)}) || p_\psi(z)) = \mathcal{L}(x^{(i)}) (2.7)$$

We can actually write the KL divergence into the expectation term and denote \((\psi, \phi)\) as \(\theta\). By the previous discussion, this means that our objective is to solve the optimization problem \(\arg \max_\theta \sum_i \mathcal{L}(x^{(i)})\) of full dataset variational lower bound. Thus L-BFGS or HF SGVI algorithm can be implemented straightforwardly to estimate the parameters of both generative and recognition models. Since the first term of reconstruction error appears in Eq. (3.33) with an expectation form on latent variable, Kingma and Welling (2014); Rezende et al. (2014) used a small finite number \(M\) samples as Monte Carlo integration with reparametrization trick to reduce the variance. This is, in fact, drawing samples from the standard normal distribution. In addition, the second term is the KL divergence between the variational distribution and the prior distribution, which acts as a regularizer.

### 2.2.2 Deep Neural Networks with Hybrid Hidden Layers

In the experiments, setting \(M = 1\) can not only achieve excellent performance but also speed up the program. In this special case, we discuss the relationship between VAE and traditional deep auto-encoder. For binary inputs, denote the output as \(y\), we have \(\log p_\psi(x|z) = \sum_{j=1}^D x_j \log y_j + (1 - x_j) \log(1 - y_j)\), which is exactly the negative cross-entropy. It is also apparent that \(\log p_\psi(x|z)\) is equivalent to negative squared error loss for continuous data. This means that maximizing the lower bound is roughly equal to minimizing the loss function of a deep neural network (see Figure 1 in supplementary), except for different regularizers. In other words, the prior in VAE only imposes a regularizer in encoder or generative model, while \(L_2\) penalty for all parameters is always considered in deep neural nets. From the perspective of deep neural networks with hybrid hidden nodes, the model consists of two Bernoulli
layers and one Gaussian layer. The gradient computation can simply follow a variant of backpropagation layer by layer (derivation given in supplementary). To further see the rationale of setting $M = 1$, we will investigate the upper bound of the Lipschitz constant under various activation functions in the next lemma. As Theorem 2 implies, the variance of approximate expectation by finite samples mainly relies on the Lipschitz constant, rather than dimensionality. According to Lemma 4, imposing a prior or regularization to the parameter can control both the model complexity and function smoothness. Lemma 4 also implies that we can get the upper bound of the Lipschitz constant for the designed estimators in our algorithm.

**Lemma 4.** For a sigmoid activation function $g$ in deep neural networks with one Gaussian layer $z, z \sim N(\mu, C), C = R^\top R$. Let $z = \mu + R\epsilon$, then the Lipschitz constant of $g(W_i(\mu + R\epsilon) + b_i)$ is bounded by $\frac{1}{2\epsilon} \|W_i R\|_2$, where $W_i$ is $i$th row of weight matrix and $b_i$ is the $i$th element bias. Similarly, for hyperbolic tangent or softplus function, the Lipschitz constant is bounded by $\|W_i R\|_2$.

### 2.3 Experiments

We apply our 2nd order stochastic variational inference to two different non-conjugate models. First, we consider a simple but widely used Bayesian logistic regression model, and compare with the most recent 1st order algorithm, doubly stochastic variational inference (DSVI) Titsias and Lázaro-Gredilla (2014), designed for sparse variable selection with logistic regression. Then, we compare the VAE model with our algorithms. All the experiments are conducted on a 3.2GHz CPU computer with X-Intel 32G RAM. For fair comparison, the algorithms and datasets we referred to as the baseline remain the same as in the previously cited work and software was downloaded from the website of relevant papers.
2.3.1 Bayesian Logistic Regression

Given a dataset \( \{x_i, y_i\}_{i=1}^N \), where each instance \( x_i \in \mathbb{R}^D \) includes the default feature 1 and \( y_i \in \{-1, 1\} \) is the binary label, the Bayesian logistic regression models the probability of outputs conditional on features and the coefficients \( \beta \) with an imposed prior. The likelihood and the prior can usually take the form as \( \prod_{i=1}^N g(y_i x_i^T \beta) \) and \( \mathcal{N}(0, \Lambda) \) respectively, where \( g \) is sigmoid function and \( \Lambda \) is a diagonal covariance matrix for simplicity. We can propose a variational Gaussian distribution \( q(\beta | \mu, C) \) to approximate the posterior of regression parameter. If we further assume a diagonal \( C \), a factorized form \( \prod_{j=1}^D q(\beta_j | \mu_j, \sigma_j) \) is both efficient and practical for inference. Unlike iteratively optimizing \( \Lambda \) and \( \mu, C \) as in variational EM, Titsias and Lázaro-Gredilla (2014) noticed that the calculation of the gradient w.r.t the lower bound indicates the updates of \( \Lambda \) can be analytically worked out by variational parameters, thus resulting a new objective function for the representation of lower bound that only relies on \( \mu, C \) (details refer to Titsias and Lázaro-Gredilla (2014)). We apply our algorithm to this variational logistic regression on three appropriate datasets:\footnote{From \url{http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html}.}

**DukeBreast** and **Leukemia** are small size but high-dimensional for sparse logistic regression, and **a9a** which is large. See Table 2.1 for additional dataset descriptions.

Fig. 2.2 shows the convergence of Gaussian variational lower bound for Bayesian logistic regression in terms of running time, where the first row is the convergence rate and the second row is the estimated regression coefficients. It is worth mentioning that the lower bound of HFSGVI converges within 3 iterations on the small datasets **DukeBreast** and **Leukemia**. This is because all data points are fed to all algorithms and the HFSGVI uses a better approximation of the Hessian matrix to proceed 2nd order optimization. L-BFGS-SGVI also take less time to converge and yield slightly larger lower bound than DSVI. In addition, as an SGD-based algorithm, it is clearly
Table 2.1: Comparison on misclassification error

<table>
<thead>
<tr>
<th>#train/test/feature</th>
<th>DSVI</th>
<th>L-BFGS-SGVI</th>
<th>HFSGVI</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>train</td>
<td>test</td>
<td>train</td>
</tr>
<tr>
<td>DukeBreast(38/4/7129)</td>
<td>0/38</td>
<td>2/4</td>
<td>0/38</td>
</tr>
<tr>
<td>Leukemia(38/34/7129)</td>
<td>0/38</td>
<td>3/34</td>
<td>0/38</td>
</tr>
<tr>
<td>A9a(32561/16281/123)</td>
<td>4948/32561</td>
<td>2455/16281</td>
<td>4936/32561</td>
</tr>
</tbody>
</table>

seen that DSVI is less stable for small datasets and fluctuates strongly even at the later optimized stage. For the large a9a, we observe that HFSGVI also needs 1000 iterations to reach a good lower bound and becomes less stable than the other two algorithms. However, L-BFGS-SGVI performs the best both in terms of convergence rate and the final lower bound. The misclassification report in Table 2.1 reflects the similar advantages of our approach, indicating a competitive prediction ability on various datasets. Finally, it is worth mentioning that all three algorithms learn a set of very sparse regression coefficients on the three datasets.

The optimized lower bound function when the covariance matrix C is diagonal is as following.

\[
\mathcal{L}(\mu, \sigma) = \mathbb{E}_{z \sim \mathcal{N}(0, I)}[\log l(\mu + \sigma \odot z)] + \frac{1}{2} \sum_{i=1}^{d} \log \frac{\sigma_i^2}{\mu_i^2 + \mu_i^2},
\]

where \( l \) is the likelihood function.

2.3.2 Variational Auto-encoder

We also apply the 2nd order stochastic variational inference to train a VAE model (setting \( M = 1 \) for Monte Carlo integration to estimate expectation) or the equivalent deep neural networks with hybrid hidden layers. The datasets we used are images from the Frey Face, Olivetti Face and MNIST\(^2\). We mainly learned three tasks by maximizing the variational lower bound: parameter estimation, images reconstruction and images generation. Meanwhile, we compared the convergence rate (running

\(^2\)From [http://www.cs.nyu.edu/~roweis/data.html](http://www.cs.nyu.edu/~roweis/data.html)
Figure 2.2: Results for sparse logistic regression.

time) of three algorithms, where in this section the compared SGD is the Ada version Duchi et al. (2011) that is recommended for VAE model in Kingma and Welling (2014); Rezende et al. (2014). The experimental setting is as follows. The initial weights are randomly drawn from $\mathcal{N}(0, 0.01^2 \mathbf{I})$ or $\mathcal{N}(0, 0.001^2 \mathbf{I})$, while all bias terms are initialized as 0. The variational lower bound only introduces the regularization on the encoder parameters, so we add an $L_2$ regularizer on decoder parameters with a shrinkage parameter 0.001 or 0.0001. The number of hidden nodes for encoder and decoder is the same for all auto-encoder model, which is reasonable and convenient to construct a symmetric structure. The number is always tuned from 200 to 800 with 100 increment. The mini-batch size is 100 for L-BFGS and Ada, while larger mini-batch is recommended for HF, meaning it should vary according to the training size.

The detailed results are shown in Fig. 2.3 and 2.5. Fig. 2.3(a) shows how lower bound increases w.r.t program running time for different algorithms, and (b) illustrates the reconstruction ability of this auto-encoder model when $d_z = 20$ (left 5 columns are randomly sampled from dataset), and the learned manifold of generative
model when \( d_z = 2 \): two coordinates of latent variables \( z \) take values that were transformed through the inverse CDF of the Gaussian distribution from equal distance grid on the unit square. \( p_\theta(x|z) \) is used to generate the images. Both Hessian-free and L-BFGS converge faster than Ada in terms of running time. HFSGVI also performs competitively with respect to generalization on testing data. Ada takes at least four times as long to achieve similar lower bound. Theoretically, Newton’s method has a quadratic convergence rate in terms of iteration, but with a cubic algorithmic complexity at each iteration. However, we manage to lower the computation in each iteration to linear complexity. Thus considering the number of evaluated training data points, the 2\textsuperscript{nd} order algorithm needs much fewer step than 1\textsuperscript{st} order gradient descent (see visualization in supplementary on MNIST). The Hessian matrix also replaces manually tuned learning rates, and the affine invariant property allows for automatic learning rate adjustment. Technically, if the program can run in parallel with GPU, the speed advantages of 2\textsuperscript{nd} order algorithm should be more obvious Ngiam et al. (2011).

Fig. 2.3(b) and Fig. 2.5(b) are reconstruction results of input images. Notice that Fig. 2.5(b) illustrates reconstruction comparison \textbf{without} patch sampling, where \( d_z = 100 \) and top 5 rows are original faces. From the perspective of deep neural network, the only difference is the Gaussian distributed latent variables \( z \). By corollary of Theorem 2, we can roughly tell the mean \( \mu \) is able to represent the quantity of \( z \), meaning this layer is actually a linear transformation with noise, which looks like dropout training Dahl et al. (2013). Specifically, Olivetti includes 64×64 pixels faces of various persons, which means more complicated models or pre-processing Hinton and Salakhutdinov (2006) (e.g. nearest neighbor interpolation, patch sampling) is needed. However, even when simply learning a very bottlenecked auto-encoder, our approach can achieve acceptable results. Note that although we have tuned the hyperparameters of Ada by cross-validation, the best result is still a
bunch of mean faces. For manifold learning, Fig. 2.3(c) represents how the learned generative model can simulate the images by HFSGVI. To visualize the results, we choose the 2D latent variable \( z \) in \( p_\psi(x|z) \), where the parameter \( \psi \) is estimated by the algorithm. The two coordinates of \( z \) take values that were transformed through the inverse CDF of the Gaussian distribution from equal distance grid (10×10 or 20×20) on the unit square. Then we merely use the generative model to simulate the images. Besides these learning tasks, denoising, imputation Rezende et al. (2014) and even generalizing to semi-supervised learning Kingma et al. (2014) are possible application of our approach.

(b) Reconstruction and Manifold

Figure 2.3: Results on FreyFace dataset for (a) convergence of lower bound, (b) reconstruction and manifold.

(b) Reconstruction and Manifold

Figure 2.4: Results on MNIST dataset for (a) convergence of lower bound, (b) reconstruction and manifold.
2.4 Summary

In this Chapter we proposed a scalable 2\textsuperscript{nd} order stochastic variational method for generative models with continuous latent variables. By developing Gaussian back-propagation through reparametrization we introduced an efficient unbiased estimator for higher order gradients information. Combining with the efficient technique for computing Hessian-vector multiplication, we derived an efficient inference algorithm (HFSGVI) that allows for joint optimization of all parameters. The algorithmic complexity of each parameter update is quadratic w.r.t the dimension of latent variables for both 1\textsuperscript{st} and 2\textsuperscript{nd} derivatives. Furthermore, the overall computational complexity of our 2\textsuperscript{nd} order SGVI is linear w.r.t the number of parameters in real applications just like SGD or Ada. However, HFSGVI may not behave as fast as Ada in some situations, e.g., when the pixel values of images are sparse.
The Hierarchical Graph-Coupled Hidden Markov Model (hGCHMM) is a useful tool for tracking and predicting the spread of contagious diseases, such as influenza, by leveraging social contact data collected from individual wearable devices. However, the existing inference algorithms depend on the assumption that the infection rates are small in probability, typically close to 0. The purpose of this paper is to build a unified learning framework for latent infection state estimation for the hGCHMM, regardless of the infection rate and transition function. We derive our algorithm based on a dynamic auto-encoding variational inference scheme, thus potentially generalizing the hGCHMM to models other than those that work on highly contagious diseases. We experimentally compare our approach with previous Gibbs EM algorithms and standard variational method mean-field inference, on both semi-synthetic data and app collected epidemiological and social records.
Table 3.1: Notations

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n \in [N]$</td>
</tr>
<tr>
<td>$t \in [T]$</td>
</tr>
<tr>
<td>$s \in [S]$</td>
</tr>
<tr>
<td>$z_n$</td>
</tr>
<tr>
<td>$G_{t-1}$</td>
</tr>
<tr>
<td>$\gamma(\gamma_n)$</td>
</tr>
<tr>
<td>$\alpha(\alpha_n)$</td>
</tr>
<tr>
<td>$\beta(\beta_n)$</td>
</tr>
<tr>
<td>$\pi$</td>
</tr>
<tr>
<td>$x_{n,t} \in \mathcal{X} = {0, 1}$</td>
</tr>
<tr>
<td>$y_{n,t} \in \mathcal{Y} = {0, 1}^S$</td>
</tr>
<tr>
<td>$\theta_{x_{n,t},s}$</td>
</tr>
</tbody>
</table>

3.1 Background

In this section, we first introduce hierarchical Graph-Coupled Hidden Markov Models (hGCHMMs) to simultaneously track the spread of infection in a small cell phone community and capture person-specific infection parameters by leveraging a link prior that incorporates additional covariates, and investigate two link functions, the beta-exponential link and sigmoid link, both of which allow the development of a principled Bayesian hierarchical framework for disease transmission. The results of our model allow us to predict the probability of infection for each persons on each day, and also to infer personal physical vulnerability and the relevant association with covariates.

3.1.1 Notation

Let $N$ be the number of participants in the social community, and $T$ be the days being tracked. The health record for each participant can be simulated as an HMM with $T$ timestamps. Let $\mathcal{Y}$ be the observation space of a Markov chain with hidden state space $\mathcal{X}$, and initial probabilities $\pi$. We also refer to the infection rate related parameters as $\gamma$, $\alpha$ and $\beta$. In particular, $\gamma$ gives the probability that a
previously-infectious individual recovers and again becomes susceptible. \( \alpha \) represents the probability that an infectious person from outside the community infects a previously-susceptible person within the community. \( \beta \) represents the probability that an infectious person from the community infects a previously-susceptible person. These parameters are used to construct the transition probability of HMMs, whereas the emission probability \( \theta_X \) merely depends on the hidden state.

Additionally, we introduce the notation associated with the mobile app survey or sensor logs. \( S \) is the number of symptoms in self report record. Temporal features in our survey are denoted by \( z \). Since our discussion of HMMs varies between parameter sharing and the inhomogeneous setting, we temporarily did not include any subscripts to avoid ambiguity. Clarification will be given in the subsequent sections. However, the overall notation description is concisely summarized in Table 3.1.

3.1.2 Graph-coupled Hidden Markov Model

We can briefly review the graph-coupled hidden Markov model (GCHMM), evolving from coupled hidden Markov model (CHMM) Brand et al. (1997), a dynamic representation for analyzing the discrete-time series data by considering the interactions between Markov chains (see Figure 3.1 for an example, where filled nodes are observed). The top panel in (a) is an HMM graphical model, and the bottom one is a dynamic social network; the right panel illustrates the formation of GCHMMs involving 3 people, where 1 and 3 have social contact between \( t - 1 \) and \( t \). The infection states of 1 and 3 at \( t \) are then both influenced by each others’ infection states at time \( t - 1 \). The standard CHMM is typically fully connected, between hidden nodes at successive timestamps, whereas the intrinsic sparsity of a dynamic social network can couple multiple HMMs with the possibility for fast inference. The number of parameters needed to be inferred will decrease dramatically from \( \mathcal{O}(N^N) \) to \( \mathcal{O}(N^{n_{\text{max}}}) \) where \( n_{\text{max}} \) is the maximum degree of hidden nodes. This advantage
will further benefit our message passing algorithm and hGCHMMs in the subsequent section.

3.1.3 Generative Modeling

Let $G_t = (N, E_t)$ be a network structure snapshot between timestamps $t - 1$ and $t$, where each agent or participant is represented by a node $n \in \mathcal{N} = [N]^1$ in graph $G_t$, and $E_t$ is a set of undirected edges in $G_t$, where unordered pair $(n_i, n_j) \in E_t$ if two participants $n_i$ and $n_j$ have a valid contact during the time interval $(t - 1, t]$. The bottom in Figure 3.1(a) illustrates an example of the dynamic social networks. Assuming that each participant $n$ is represented a HMM with binary hidden state $x_{n,t}$ shown as the top in Figure 3.1(a), where 0 and 1 indicate susceptible and infectious respectively. The observed node $y_{n,t}$ is an $S$ dimensional binary vector $(y_{n,t,1}, y_{n,t,2}, \ldots, y_{n,t,S})$ as an indicator for $S$ symptoms. Thus, the generative model

\[ [N] \text{ means a set including integers from 1 to } N. \]
of GCHMMs is given in a fully bayesian way.

\[
\begin{align*}
\pi &\sim \text{Beta}(a_\pi, b_\pi) \\
\alpha &\sim \text{Beta}(a_\alpha, b_\alpha) \\
\beta &\sim \text{Beta}(a_\beta, b_\beta) \\
\gamma &\sim \text{Beta}(a_\gamma, b_\gamma) \\
\theta_{0,s} &\sim \text{Beta}(a_0, b_0) \\
\theta_{1,s} &\sim \text{Beta}(a_1, b_1) \\
x_{n,0} &\sim \text{Bernoulli}(\pi) \\
x_{n,t} &\sim \text{Bernoulli}(\phi_{n, x_{n',t}:(n,n')\in G_t}(\alpha, \beta, \gamma)) \\
y_{n,t,s} &\sim \text{Bernoulli}(\theta_{x_{n,t},s})
\end{align*}
\]  

(3.1)

where the transition probability \( \phi_{n, x_{n',t}:(n,n')\in G_t}(\alpha, \beta, \gamma) \) is a function of the infection parameters and the dynamic graph structure. This homogenous setting means all HMMs share the same parameters set or similar transition function. The difference is reflected as Figure 3.1 indicated, the transition of the hidden state is not only dependent on the previous state of its own Markov chain but also may be influenced by states from other HMMs that have edges connected to it. One undirected edge in \( G_t \) indicates a valid contact in time interval \( (t-1, t] \), thus leading to a directed edge in GCHMMs. Recalling the definition in terms of \( \gamma, \alpha, \beta \) (See Table 3.1), it is natural to construct the transition probability function as follows:

\[
\phi_{n, x_{n',t}:(n,n')\in G_t}(\gamma, \alpha, \beta) = \begin{cases} 
\gamma & x_{n,t} = 1, x_{n,t+1} = 0; \\
1 - \gamma & x_{n,t} = 1, x_{n,t+1} = 1; \\
1 - (1 - \alpha)(1 - \beta)C_{n,t} & x_{n,t} = 0, x_{n,t+1} = 1; \\
(1 - \alpha)(1 - \beta)C_{n,t} & x_{n,t} = 0, x_{n,t+1} = 0.
\end{cases}
\]

(3.2)

where \( \mathbb{I}_{\{t\}} \) is the indicator function and \( C_{n,t} = \sum_{n':(n,n')\in E_t} \mathbb{I}_{\{x_{n',t}=1\}} \) is the count of possible infectious sources for node \( n \) in \( G_t \), in other words, it means that besides participant \( n \), the number of other infectious nodes that have social contacts with \( n \) at the previous day. The epidemiological intuition is very simple, the more infectious people one comes in contact with, the more probable one is to be contaminated.

This Bayesian formulation of GCHMMs can be applied to fit homogenous susceptible-infectious-susceptible (SIS) epidemic dynamics. Additionally, the Bayesian inference
of this model (e.g. Dong et al. (2012)) can be reduced to a special case from our heterogeneous model by getting rid of link hierarchy and personalization of infection parameters. We do not go into details here, but we generalize the Baum-Welch algorithm to sparse-coupled HMMs and relax the assumption of near-zero parameters, i.e. $\alpha$, $\beta$ and $\gamma$ are all $\approx 0$.

### 3.1.4 Generalized Baum-Welch Algorithm

HMMs usually model independent sequenced or discrete time-series data, and represent long-range dependencies between observations for each data point, mediated via latent variables. We are inspired by the fact (Loeliger (2004)) that the forward-backward, Viterbi and EM for HMMs (see Murphy (2012)’s review) learning algorithms have their equivalent message passing formulation in factor graph representation. The independency property allows the three algorithms to be efficient and effective, while the blownup parameter space of standard coupled HMMs makes analogous treatment impossible and impractical. Thus different assumptions have been imposed on CHMM, such as Zhong and Ghosh (2001). Fortunately, in our sparse coupling setting, a Generalized Baum-Welch Algorithm can be developed via a similar factor graph representation and enables efficient forward-backward, Viterbi and approximate EM algorithms.

First, we describe the factor graph in Figure 3.2 derived from the Bayesian network in Figure 3.1(b). Commonly, factor graphs contain two types of nodes, variable nodes and factor nodes. In our model, the factor nodes can be categorized as emission probability functions and transition probability functions,

\[
\phi_{n,t,s}(y_{n,t,s}|x_{n,t}) = p(y_{n,t,s}|x_{n,t}) = \theta_{y_{n,t,s}}(1 - \theta_{x_{n,t,s}})y_{n,t,s} \tag{3.3}
\]

\[
\phi_{n,t-1,t}(x_{n,t-1}, x_{n,t-1};(n',n)\in E_t, x_{n,t}) = p(x_{n,t}|x_{n,t-1}, x_{n',t-1};(n',n)\in E_t) \tag{3.4}
\]

where the second equation defines the same function (3.2). Notice that the factor graph is undirected and direction information is comprised in the factor node. The
factor graph is still shown with plates; each plate represents a participant while the interaction is captured by transition factor. In fact, during the running of the EM algorithm, all parameters are unknown, so \( \alpha, \beta, \gamma \) can go at the top of the current graph, adding edges to all transition factors. For simplicity, we did not include them in our factor graph, though this widely used trick is introduced in independent HMMs (Loeliger (2004)).

3.1.5 Forward-backward Algorithm

Notice that even if the Bayesian network is not exactly a directed polytree (cyclic paths exist if omitting edge direction), marginal inference on each hidden node can still be approximated with belief propagation on the factor graph for the sake of efficiency. In this section, we will derive the single node belief propagation rules (Pearl (2014)) on Figure 3.2. Denote the message passing to the child, i.e. from \( x_{m,t-1} \) to \( x_{n,t} \) as \( \pi_{x_{m,t-1}}(x_{n,t}) \), and the message to parent, i.e. from \( x_{k,t+1} \) to \( x_{n,t} \) as \( \lambda_{x_{k,t+1}}(x_{n,t}) \). Then, the belief or probability distribution passed with all evidence shown is denoted as \( \text{BEL}(x_{n,t}) = P(x_{n,t}|Y) \). Our derivation is based on Pearl’s belief
propagation algorithm. All $\pi$ in the following imply the term should be normalized to 1 as a valid probability distribution. For notation simplicity, we further denote $\pi_{x_{n,t}}(x_{.,t-1}) = p(x_{n,t}|x_{.,t-1})$ and $\lambda_{x_{.,t+1}}(x_{n,t}) = p(x_{.,t+1}|x_{n,t})$. The principal part of forward-backward algorithm can be summarized as follows, where the detailed update for $\lambda$ and $\pi$ is derived in Appendix A.

$$\pi^{(i)}(x_{n,t}) = \sum_{x_{n,t-1},x_{n',t-1}:(n',n)\in E_t} \phi_{n,t-1,t} \prod_{n\cup\{n':(n',n)\in E_t\}} \pi^{(i)}(x_{.,t-1}) \tag{3.5}$$

$$\lambda^{(i)}(x_{n,t}) = \prod_{s=1}^S \lambda_{y_{n,t,s}}(x_{n,t}) \prod_{n\cup\{n':(n',n)\in E_t\}} \lambda^{(i)}_{x_{.,t+1}}(x_{n,t}) \tag{3.6}$$

$$\text{BEL}^{(i)}(x_{n,t}) = \pi^{(i)}(x_{n,t}) \lambda^{(i)}(x_{n,t}) \tag{3.7}$$

Even if it has been proven that belief propagation on standard HMMs is equivalent to the forward-backward algorithm, the update step for message passing in our case is a little more complicated since the single node dependence is generalized to multi-nodes. Let the maximum degree of all $G_t$s be $M$, then the update of sum-product for message can be computed with complexity $O(2^M)$ at each iteration. This is the reason why the sparsity assumption is required in our algorithm. In addition, the initialization for all messages from variable nodes to factor nodes, such as $\pi_{x_{.}}(x_{.}), \lambda_{x_{.}}(x_{.})$, can be set to all 1s.

**Viterbi** algorithm can be derived in a straightforward way, if the sum-product $\sum \prod$ in forward-backward is substitute by max-product $\max \prod$. Since the message updating step is almost the same, it won’t be discussed here.

### 3.1.6 Approximate EM Algorithm

In this section, we will put forward a parameter learning scheme via the generalized Baum-Welch Algorithm. It is straightforward to derive the expected complete data
log-likelihood given by
\[ Q(\Theta, \Theta^{old}) = \sum_{X} \sum_{n=1}^{N} \left\{ x_{n,0} \log \pi + (1 - x_{n,0}) \log (1 - \pi) + \sum_{t=1}^{T} \log \phi_{n,t-1, t} + \sum_{t=1}^{T} \sum_{s=1}^{S} \log \phi_{n,t, y|x,s} \right\} \Pr(X|Y, \Theta^{old}) \] (3.8)

This is exactly the E-step in EM algorithm, and the non-approximate M-step can be optimized for parameters \( \pi, \theta, \) and \( \gamma \). In fact, due to the conjugacy of these parameters, their posterior distribution can also be analytically computed. Taking the partial derivative on \( Q \), i.e.
\[ \frac{\partial Q(\Theta, \Theta^{old})}{\partial \xi} = 0 \quad \frac{\partial Q(\Theta, \Theta^{old})}{\partial \theta_{0,s}} = 0 \quad \frac{\partial Q(\Theta, \Theta^{old})}{\partial \theta_{1,s}} = 0 \quad \frac{\partial Q(\Theta, \Theta^{old})}{\partial \gamma} = 0 \] (3.9)

By solving above equations, we obtain the update formula for corresponding parameters.
\[ \pi = \frac{\sum_{n=1}^{N} \mathbb{E}[x_{n,0}]}{N} \] (3.10)
\[ \theta_{i,s} = \frac{\sum_{x_{1,N,1:T} \in \{0,1\} \times \mathcal{N}^2} \left( \sum_{n=1}^{N} \sum_{t=1}^{T} y_{n,t,s} \mathbb{I}[x_{n,t} = i] \right) p(X|Y, \Theta^{old})}{\sum_{x_{1,N,1:T} \in \{0,1\} \times \mathcal{N}^2} \left( \sum_{n=1}^{N} \sum_{t=1}^{T} \mathbb{I}[x_{n,t} = i] \right) p(X|Y, \Theta^{old})}, \quad i = 0, 1 \]
\[ \gamma = \frac{\sum_{x_{1,N,1:T} \in \mathcal{X} \times \mathcal{N}^2} \left( \sum_{n=1}^{N} \sum_{t=2}^{T} \mathbb{I}[x_{n,t-1} = 1, x_{n,t} = 0] \right) p(X|Y, \Theta^{old})}{\sum_{x_{1,N,1:T} \in \mathcal{X} \times \mathcal{N}^2} \left( \sum_{n=1}^{N} \sum_{t=2}^{T} \mathbb{I}[x_{n,t-1} = 1] \right) p(X|Y, \Theta^{old})} \]

Notice that it does not matter if we change the \( p(X|Y, \Theta^{old}) \) to \( p(X, Y|\Theta^{old}) \), and \( \frac{\partial Q(\Theta, \Theta^{old})}{\partial \alpha} = 0, \frac{\partial Q(\Theta, \Theta^{old})}{\partial \beta} = 0 \) can result in an analogous computation as for \( \gamma \) for the iteration. However, except for \( \pi \), the exact computational complexity of the iteration step for other parameters is intractable, exponentially increasing with \( N \) or \( N^2 \). Since we did not assume near-zero parameters, the induced non-conjugacy requires further approximation in the M-step. If we approximate \( P(X|Y, \Theta^{old}) = \prod_{n,t} p(x_{n,t}|Y, \Theta^{old}) \) in a fully factorized form, then all the optimized results for \( \theta \)
would update analytically, because \( p(x_{n,t}|Y, \Theta^{old}) \) (i.e. BEL\((x_{n,t})\)) can be computed by the *forward-backward* algorithm as derived before.

\[
\theta_{i,s} = \frac{\sum_{n=1}^{N} \sum_{t=1}^{T} y_{n,t,s} \mathbb{E}[\mathbb{I}_{i=0}(1-x_{n,t}) + \mathbb{I}_{i=1}x_{n,t}]}{\sum_{n=1}^{N} \sum_{t=1}^{T} \mathbb{E}[\mathbb{I}_{i=0}(1-x_{n,t}) + \mathbb{I}_{i=1}x_{n,t}]} , i = 0, 1 \quad (3.11)
\]

Updating \( \gamma, \alpha, \beta \) is a little tricky. First we introduce the approximation for \( \gamma \), which will make the other two updates more understandable. Even if we use full factorization in the approximation, the update of \( \gamma \) is associated with variables \( x_{n,t-1} \) and \( x_{n,t} \). A natural idea is to use Monte Carlo methods to sample \( \{\tilde{x}_{1:N,1:T}\} \in X^{N^2} \) from \( p(X|Y, \Theta^{old}) = \prod_{n,t} p(x_{n,t}|Y, \Theta^{old}) \). Then we can count the number of times that event \( x_{n,t-1} = 1, x_{n,t} = 0 \) happens. For simplicity, we can directly assign the simulated sample by Bayesian decision strategy according to each \( p(x_{n,t}|Y, \Theta^{old}) \) instead of sampling. That is to say, we only need to set the sample \( x_{n,t} = \arg \max_{\tilde{x}_{n,t}=(0,1)} p(x_{n,t}|Y, \Theta^{old}) \) and give the following result.

\[
\gamma = \frac{\sum_{n=1}^{N} \sum_{t=1}^{T} \mathbb{I}_{\tilde{x}_{n,t-1}=1, \tilde{x}_{n,t}=0}}{\sum_{n=1}^{N} \sum_{t=1}^{T} \mathbb{E}[\tilde{x}_{n,t-1}]} \approx \frac{\sum_{n=1}^{N} \sum_{t=1}^{T} \mathbb{I}_{\tilde{x}_{n,t-1}=1, \tilde{x}_{n,t}=0}}{\sum_{n=1}^{N} \sum_{t=1}^{T} \mathbb{I}_{\tilde{x}_{n,t-1}=1}} \quad (3.12)
\]

The same trick can be applied to update \( \alpha, \beta \). However, along with this approximation trick, we also need variable substitution. Let \( \tau_j = (1-\alpha)(1-\beta)^j \), then \( \alpha = 1-\tau_0 \). Therefore the update of \( \alpha \) and \( \tau_i, i = 1, ..., M \) is analogous to \( \gamma \).

\[
\alpha = \frac{\sum_{n=1}^{N} \sum_{t=1}^{T} \mathbb{I}_{\tilde{x}_{n,t-1}=0, \tilde{x}_{n,t}=1, \sum_{n',n''} \tilde{x}_{n',t-1} = 0}}{\sum_{n=1}^{N} \sum_{t=1}^{T} \mathbb{E}[1-x_{n,t-1}]} \quad (3.13)
\]

\[
\tau_i = \frac{\sum_{n=1}^{N} \sum_{t=1}^{T} \mathbb{I}_{\tilde{x}_{n,t-1}=0, \tilde{x}_{n,t}=0, \sum_{n',n''} \tilde{x}_{n',t-1} = j}}{\sum_{n=1}^{N} \sum_{t=1}^{T} \mathbb{E}[1-x_{n,t-1}]} \quad (3.14)
\]

\[\beta = 1 - \left( \frac{\tau_i}{1-\alpha} \right)^\frac{1}{j}, \text{ thus meaning we have } M \text{ estimations for } \beta \text{s. How to combine these } \beta \text{s to obtain a better estimation may vary for different applications. We suggest one possibility using averaging that can be adjusted under various circumstances. In}
\]

36
Algorithm 2 Generalized Baum-Welch Algorithm

Input: $Y$, $G$,

Output: parameter set $\Theta = \{\pi, \theta, \alpha, \beta, \gamma\}$ and hidden matrix $X$

Initialize coefficient parameter $\Theta^{(0)}$

while $\Theta^{new}$ NOT Convergence do

Run one iteration forward-backward algorithm on $\Theta^{old}$ to obtain $p(x_{n,t}|Y, \Theta^{old})$

Update $\Theta^{new}$ based on equations from (10) to (15)

end while

3.1.7 A Simulation Study

Synthesized data based on Real Social Networks  We leveraged a dynamic social network dataset with 84 nodes over 107 days in Madan et al. (2012) denoted as $G_{84 \times 107}$; we modified it to make its maximum degree be bounded by constant $M = 11$. Based on the generative model, we simulated infection states $X_{84 \times 108}$ (including $X_{t,0}$ without emission) and observed symptoms data $Y_{84 \times 107 \times 6}$. We ran inference using two algorithms Generalized BW (GBW) and the Gibbs sampling developed by Dong et al. (2012) in two different settings, when parameters are known and

\[
\beta = 1 - \frac{1}{M} \sum_{j=1}^{M} \left( \frac{\tau_i}{1 - \alpha} \right)\frac{1}{j}
\]  

(3.15)
unknown. Notice that GBW with known parameters is reduced to forward-backward
belief propagation. Both algorithms won’t take $X$ as arguments but predict an $X$.
Gibbs sampling and GBW are run for 500 and 15 iterations respectively. The Gibbs
sampling implemented in the experiments is burns-in half of the total iterations.
Since $X$ is a binary matrix, a threshold of 0.5 is used for prediction.

In order to observe the performance with respect to the number of iterations,
we conduct experiments testing the predictive accuracy of $X$ with results shown in
Figure 3.3. Notice that the axis of Gibbs sampling is scaled by logarithm and both
algorithms can achieve an accuracy above 98.5%. With parameters known, both al-

gorithms can achieve good performance within a few iterations, while in the unknown


case, the generalized BW algorithm gives better performance than Gibbs sampling
over fewer iterations. However, the excellent performance of GBW is dependent on
the initialization of parameters. If they are chosen inappropriately, GBW may not
perform well, a common disadvantage of EM algorithm. In addition, with more
iterations, Gibbs sampling will give more robust prediction results.

3.1.8 Extending GCHMMs to Hierarchies

Though we successfully design a generalized EM algorithm for GCHMMs, we still
overlook temporal covariates $z_n$. In epidemics, it is commonly assumed that personal
health features (covariates) are relevant to influenza vulnerability. In our model,
$z_n \in \mathcal{R}^K$, where $K$ is the dimension of the covariate feature space. Without loss of
generalization, the feature is a constant of 1 in the feature space we are looking at.
This relevance is captured by the mapping $f : \mathcal{R}^K \rightarrow [0, 1]$ or the transformation
from the feature space to infection parameters. In this section, we propose two
constructions using two different link functions. A natural way to go is to extend
the beta prior of the standard GCHMM to a beta-exponential link.
**Beta-Exponential link**

\[ \eta_r \sim \mathcal{N}(\mu, \Sigma) \]  \hspace{1cm} (3.16)

\[ \gamma_n \sim \text{Beta}(\exp(z_n^T \eta_r, 1), \exp(z_n^T \eta_r, 2)) \]

\[ \alpha_n \sim \text{Beta}(\exp(z_n^T \eta_a, 1), \exp(z_n^T \eta_a, 2)) \]

\[ \beta_n \sim \text{Beta}(\exp(z_n^T \eta_b, 1), \exp(z_n^T \eta_b, 2)) \]

where \( \eta_r \) is distributed as a multivariate Gaussian, playing the role of the regression coefficients, since the expectation \( \frac{1}{1 + e^{z_n^T (\eta_r, 1) - \eta_r, 2}} \) can be considered as an approximation for logistic regression with coefficients \( (\eta_r, 1) - (\eta_r, 2) \). This link also enables the exponential term \( \exp(z_n^T \eta_r) \) to take the place of the hyper-parameter of the beta prior. The usual count update to the hyper-parameter will implicitly update \( \eta_r \) via our EM algorithm.

Once \( \gamma, \alpha, \beta \) are allowed to be indexed by \( n \), a new *transmission function* merely needs an index modification of the arguments in (3.2), but otherwise remains the same, i.e. \( \phi_{n, n':(n, n') \in G_t} (\gamma_n, \alpha_n, \beta_n) \). The advantage of this setting is that it allows for the approximate Gibbs sampling of infection parameters in a way that still holds for GCHMMs, except for \( \eta_r \), so that the original Gibbs sampling scheme to update the beta distribution by event counts is the same in the later E-step. Another advantage is, when \( X \) is generalized to categorical variables, a similar construction also works. We use an individual level distribution with the transition but not with the emission matrix because it makes more sense that everyone has the same probability of a physical behavior given an infection state. Patients should have corresponding symptoms, such as cough or throat pain, or the flu cannot be discovered or diagnosed. Instead of the Beta-Exponential distribution, we can introduce a deterministic link.

**Sigmoid link**

\[ \eta_r \sim \mathcal{N}(\mu, \Sigma), \quad \gamma_n = \sigma(z_n^T \eta_r), \quad \alpha_n = \sigma(z_n^T \eta_a), \quad \beta_n = \sigma(z_n^T \eta_b) \]  \hspace{1cm} (3.17)
In this generative process, less $\eta$s are present, thus leading to a simpler model. Instead of sampling, the equation (3.17) will actually make infection parameters vanish in the model. In other words, $\phi_{n,x':(n,n')\in G_1}(\gamma_n, \alpha_n, \beta_n)$ is replaced by $\phi_{n,x':(n,n')\in G_2}(z_n, \eta, \sigma(\cdot))$. From an implementation perspective, the EM derivation will be easier, and the experimental results imply its performance is more competitive. Additionally, for both link constructions, the generative model (3.1) is also individually indexed by $n$.

**Another interpretation of $\beta_n$** In above two extensions, it is implicitly assumed that $\beta_n$ means the individual infection probability from another person within the network, is as given in Equation (3.18). From a biological perspective, the contagiousness of the infected person varies, meaning that $\beta_n$ can be interpreted as the probability of spreading illness to any other person in the social network. This heterogeneous inconsistency will not appear in the previously discussed homogeneous setting. However, the second interpretation results in a slightly complicated mathematical calculation (details in inference section), since both the total count of infectious contacts $C_{n,t}$ and their individual features are required. Thus, the probability of infection has two different definitions.

$$P(x_{n,t+1} = 1|x_{n,t} = 0) = 1 - (1 - \alpha_n)(1 - \beta_n)^{C_{n,t}}$$  \hspace{1cm} (3.18)

$$P(x_{n,t+1} = 1|x_{n,t} = 0) = 1 - (1 - \alpha_n) \prod_{n' \in S_{n,t}} (1 - \beta_{n'})$$  \hspace{1cm} (3.19)

where the node set of infectious contacts is defined as $S_{n,t} = \{n' \in [N] : (n, n') \in E_t, x_{n',t} = 1\}$.

**3.1.9 Model Evolution**

**HMM** Along the direction of graphical representation in Figure 3.1, the hierarchical GCHMMs shown in Figure 3.4 can be seen as a two-step evolution. Notice that for plotting simplification, the edges from $\theta_X$ to other observed nodes are blanked out; the two step evolution, GCHMMs→heterogenous GCHMMs→hierarchical GCHMMs. First, each HMM is allowed to contain its own infection parameters, thus evolv-
Figure 3.4: The graphical representation of hGCHMM extended from Figure 3.1(b).

ing to a heterogeneous GCHMMs. Second, the link introduced previously can be used to associate with covariates. This scheme also inspires another corresponding two-separated-segment training: Gibbs sampling to estimate the posterior mean of \((\gamma_n, \alpha_n, \beta_n)\) for heterogeneous GCHMMs, and fitting a standard logistic regression between the posterior mean and covariates.

However, there are some things that we need to notice. (1) The beta-exponential generative model is not well defined for unique dataset simulation, because we need only one sample from this generative model, which actually uses a set of fixed parameters \((\alpha^f_n, \beta^f_n, \gamma^f_n)_{n=1}^N\) to simulate \(X_{N \times T}\) and \(Y_{N \times T \times S}\). Take \(\gamma_n\) for example, \(\gamma^f_n \neq E[\gamma_n]\), since \(\gamma^f_n\) is one realization of the generative model. What our algorithm aims to learn is a generative distribution Beta\((e^{x_n^1 \eta_n^1}, e^{x_n^2 \eta_n^2})\) with expectation equal to \(\gamma^f_n\), rather than the real \(E[\gamma_n]\). (2) Therefore, in experiments our simulation dataset is always generated from the sigmoid link model. However, it is reasonable to use the beta-exponential link model for inference to eliminate the inconsistency. It has been mentioned previously the expectation of the beta-exponential is basically an approximate logistic regression. An EM like algorithm can perform point estimation well for this expectation, which in turn would be an estimator of the sigmoid link.
Another way to make the beta-exponential generative and the inference process work is to sample $\alpha_n, \beta_n, \gamma_n$ both individually and dynamically, i.e. $\gamma_{n,t}, \alpha_{n,t}, \beta_{n,t}$. A number of samples are sufficient to learn the true generative distribution, though the new inference algorithm will necessarily become more difficult.

**LDA** Notice that as shown in Figure 3.5, (a) is the commonly represented LDA model; (b) is an equivalent but different graphical representation of LDA; (c) is our proposed model, adding topic dependency, document dependency and document-specific features. If we impose a Markov dependency to model topics changing, an embedded HMM appears with respect to the latent topics. Furthermore, we construct the topic changing function relying on the document relationship and a link associated with document-specific covariates, thus resulting our hGCHMMs. A similar variant is LDA-HMM Griffiths et al. (2005), which requires extra hidden nodes essentially to model syntactic or function words, such as ”and” or ”however”, introducing a sentence-level dependency, while the HMM imposed on topic nodes gives a word-level dependency.

### 3.1.10 Approximate Conjugacy

The inference process is designed to invert the generative model and to discover the $\eta$ and $X$ that best explain $G$ and $Y$. In our hierarchical extension, however, a fully
conjugate prior is not present and knowing what the right prior is can be difficult. Thus an approximate conjugacy is developed by introducing the auxiliary variable $R_{n,t}$, representing the non-specific infection source (inside or outside networks). The idea is to decompose infection probability $I_{n,t} = 1 - (1 - \alpha_n)(1 - \beta_n)^{C_{n,t}}$ into the summation of three terms, $\alpha_n(1 - \beta_n)^{C_{n,t}}$, $(1 - \alpha_n)(1 - (1 - \beta_n)^{C_{n,t}})$ and $\alpha_n(1 - (1 - \beta_n)^{C_{n,t}})$, indicating infection from outside, inside and both respectively, thus following a categorical distribution:

$$P(R_{n,t}) = \begin{cases} \frac{\alpha_n(1-\beta_n)^{C_{n,t}}}{1-(1-\alpha_n)(1-\beta_n)^{C_{n,t}}}, & \text{if outside infection, } R_{n,t} = 1 \\
\frac{(1-\alpha_n)(1-\beta_n)^{C_{n,t}}}{1-(1-\alpha_n)(1-\beta_n)^{C_{n,t}}}, & \text{if inside infection, } R_{n,t} = 2 \\
\frac{\alpha(1-\beta_n)^{C_{n,t}}}{1-(1-\alpha_n)(1-\beta_n)^{C_{n,t}}}, & \text{if both, } R_{n,t} = 3, \end{cases}$$ (3.20)

The exact expression still does not have Beta-Bernoulli conjugacy except for the case where $R_{n,t} = 1$. However, using a taylor expansion we have $P(R_{n,t} = 2)P(x_{n,t+1} = 1|x_{n,t} = 0) \approx C_{n,t}(1 - \alpha_n)\beta_n$ and $P(R_{n,t} = 3)P(x_{n,t+1} = 1|x_{n,t} = 0) \approx C_{n,t}\alpha_n\beta_n$. The two approximations have the property that local full conditionals can be analytically obtained by discarding $\eta$ temporarily. In practice the term involving $P(R_{n,t} = 3)$ can be approximated as 0 for Gibbs sampling. Because of the biological application, $\alpha_n$ and $\beta_n$ are both a positive real value close to 0, resulting in their product being quite small. Even if this probability is taken into consideration in Gibbs sampling, there is a very small chance that $R_{n,t} = 3$. This approximation allows the posterior distribution of $\alpha_n, \beta_n$ to be much easier to compute given the current value of $\eta$.

In addition, our approximation works better than the proposed decomposition in Dong et al. (2012), i.e. $I_{n,t} = \alpha_n + C_{n,t}\beta_n$. In Figure 3.6, a quantitative comparison between our proposed approximation and previous work indicates less error achieved by our decomposition. Specifically, if $C_{n,t} = 1$, our decomposition recovers $I$ exactly (blue line and red line are overlapped); with $C_{n,t}$ increasing, both of the two approximations are biased, but our approach has constant error regardless of varying $\alpha$, and
shows less error for relatively bigger $\beta$; moreover, the induced approximate distribution through our three terms is almost in line with the true distribution (3.20). The main advantage of the novel approximation is the possibility of deriving the fully conjugate posterior. Specifically, we have the following posteriors, which will benefit from the EM algorithm described later:

$$
\alpha_n \sim \text{Beta} \left( e^{Z_n^{\top} \eta_{n,1}} + C_{n,R_n=1,3}, e^{Z_n^{\top} \eta_{n,2}} + C_{n,R_n=2} + C_{n,0\rightarrow0} \right)
$$

$$
\beta_n \sim \text{Beta} \left( e^{Z_n^{\top} \eta_{n,1}} + C_{n,R_n=2,3}, e^{Z_n^{\top} \eta_{n,2}} + C_{n,R\neq2,3} \right)
$$

$$
\gamma_n \sim \text{Beta}(e^{Z_n^{\top} \eta_{r,1}} + C_{n,1\rightarrow0}, e^{Z_n^{\top} \eta_{r,2}} + C_{n,1\rightarrow1})
$$

where the count notations are defined as follows.

$$
\begin{align*}
C_{n,i\rightarrow j} &= \sum_t \mathbb{I}\{x_{n,t}=i, x_{n,t+1}=j\}, i, j \in \{0, 1\} \\
C_{n,R_n=1,3} &= \sum_t \mathbb{I}\{R_{n,t}=1,3\} \approx C_{n,R_n=1} = \sum_t \mathbb{I}\{R_{n,t}=1\} \\
C_{n,R_n=2,3} &= \sum_t \mathbb{I}\{R_{n,t}=2,3\} \approx C_{n,R_n=2} = \sum_t \mathbb{I}\{R_{n,t}=2\} \\
C_{n,R\neq2,3} &= \sum_t C_{n,t} \left[ \mathbb{I}\{R_{n,t}=1\} + \mathbb{I}\{x_{n,t}=0, x_{n,t+1}=0\} \right]
\end{align*}
$$

Note that the auxiliary variable did not appear in the posterior of $\gamma_n$, which can be exactly computed due to conjugacy. Utilizing these approximate posteriors, the complete likelihood $P(X, R, \eta|Z)$ is obtained by integrating out the infection
parameters.

\[
\int P(\mathbf{R}| \mathbf{X}, \alpha_{n=1}^{N}, \beta_{n=1}^{N}) P(\mathbf{X}| \gamma_{n=1}^{N}, \alpha_{n=1}^{N}, \beta_{n=1}^{N}) P(\gamma_{n=1}^{N}, \alpha_{n=1}^{N}, \beta_{n=1}^{N}| \eta, \mathbf{Z}) P(\eta) d\gamma_{n=1}^{N} d\alpha_{n=1}^{N} d\beta_{n=1}^{N} = P(\eta) \prod_{n} \left( \frac{B(e^{x_{n-1}^{\eta} + C_{n,1-0}}, e^{x_{n-1}^{\eta} + C_{n,1-1}})}{B(e^{x_{n-1}^{\eta}}, e^{x_{n-1}^{\eta}})} \cdot \frac{B(e^{x_{n-1}^{\eta}} + C_{n,R_{n}=1,3}, e^{x_{n-1}^{\eta} + C_{n,1-2}} + C_{n,R_{n}=2,3})}{B(e^{x_{n-1}^{\eta}}, e^{x_{n-1}^{\eta}})} \right)
\]

where \( B(\cdot) \) is the beta function, and \( P(\eta) \) is a multinomial Gaussian distribution. The integral result enables the analytical computation of the gradient \( \nabla_{\eta} \) and 2\(^{nd} \) derivative \( \partial^2 \) of the log-likelihood, used in optimization via Newton’s method. Under such \( \beta_{n} \) interpretation, the derivatives of the log likelihood (3.21) are straightforward.

### 3.1.11 Gibbs Sampling for the Conjugate Part

#### Sampling Infection States

Given \( \alpha_{n}, \beta_{n}, \gamma_{n} \), the generative model implies a conjugate prior for \( x_{n,t} \). The unnormalized posterior probability of \( x_{n,t} = i \) can be represented as \( p_{n,t}^{i}, i = 0,1 \).

\[
p_{n,t}^{0} \propto \gamma_{n}^{1_{x_{n,t}=1}1_{x_{n,t+1}=1}} \gamma_{n}^{1_{x_{n,t-1}=0}1_{x_{n,t+1}=0}} (1-\alpha_{n})^{1_{x_{n,t-1}=0}1_{x_{n,t+1}=0}} (1-\beta_{n})^{C_{n,t-1}1_{x_{n,t-1}=0}+C_{n,t}1_{x_{n,t+1}=0}} \prod_{s} (1-\theta_{0,s}^{y_{n,t,s}=0})^{y_{n,t,s}=0}
\]

\[
p_{n,t}^{1} \propto \gamma_{n}^{1_{x_{n,t+1}=0}} (1-\gamma_{n})^{1_{x_{n,t-1}=1}1_{x_{n,t+1}=1}} \gamma_{n}^{1_{x_{n,t-1}=0}1_{x_{n,t+1}=0}} \prod_{s} (1-\theta_{1,s}^{y_{n,t,s}=0})^{y_{n,t,s}=0}
\]

where the normalized posterior of \( p(x_{n,t} = 1) \) is \( \frac{p_{n,t}^{1}}{p_{n,t}^{0}+p_{n,t}^{1}} \). We need to be careful of the boundary condition since \( x_{n,1} \) and \( x_{n,T} \) do not have this form. \( x_{n,1} \) is generated by \( x_{n,1} \sim \text{Bernoulli}(\pi) \), where \( \pi \sim \text{Beta}(a_{\pi}, b_{\pi}) \). The full conditional depends on the initial event occurrence rate \( \pi \), further requiring some mild modification. The full
conditional of $\pi$ can be efficiently derived.

$$\pi | X \sim \text{Beta} \left( a_\pi + \sum_n \mathbb{I}(x_{n,1}=1), b_\pi + N - \sum_n \mathbb{I}(x_{n,1}=1) \right)$$

For the state $x_{n,T}$, the posterior is easily computed since terms associated with $t+1$ cancel out immediately.

**Sampling Missing Observations** For real world data, a missing value problem commonly arises because of underreporting in data collection. Bayesian schemes can successfully fill in missing values by drawing $y_{n,t,s}$ according to the distribution $\text{Bernoulli}(\theta_{x_{n,t,s}})$, if they are NA. Given $y_{n,t,s}$, the posterior of $\theta_{i,s}$, $(i = 0, 1)$ is from a beta distribution.

$$\theta_{i,s} | X, Y \sim \text{Beta} \left( a_i + \sum_{n,t} \mathbb{I}(y_{n,t,s}=1,x_{n,t}=i), b_i + \sum_{n,t} \mathbb{I}(y_{n,t,s}=0,x_{n,t}=i) \right)$$

### 3.1.12 Burn-in Gibbs EM Algorithm

Previous works on CHMMs or GCHMMs seldom include the parameter $\eta$, let alone a sampling scheme for inference. In hGCHMMs, the Gaussian prior makes the posterior of $\eta$ not conjugate. One possible solution is the Metropolis Hastings (MH) algorithm due to the approximate likelihood (3.21); however, the transition kernel is difficult to choose for MH, and running large numbers of iterations is usually required to achieve good mixing. Another thing to try may be an augmentation trick based on the Poyla-Gamma distribution Polson et al. (2013), which is mentioned for the network imputation in the introduction. The drawback of this scheme is that it can be straightforward to fit the sigmoid link, while the beta-exponential prior may need further generalization. Variational Bayesian inference (see Beal (2003) for a detailed introduction) is commonly used for approximate inference by optimizing a lower bound. If the readers are familiar with variational methods, it is obvious that for the conjugate-exponential family the update for parameters (in our model, $\pi, \theta_X, \gamma$)
can be written out analytically, equivalent to the posterior derivation. However, for other parameters associated with non-conjugacy, a gradient based method is the first option to explore, such as stochastic variational inference (SVI, Hoffman et al. (2013)), unless another lower bound with respect to previous lower bound can be found (e.g. as an illustrative example in Figure 3.6(c), \( p(R = 2) \) is approximated by lower bound but \( p(R = 1) \) by upper bound. Strictly speaking, this approximation cannot be used as the lower bound of lower bound).

In this section, we propose a fast algorithm based on expectation-maximization. In hGCHMMs, expected sufficient statistics are computationally intractable since there is no closed form for integrating out the latent variables. Stochastic Approximation (SA) or Monte Carlo (MC) EM by Delyon et al. (1999); Wei and Tanner (1990) is an alternative introduced to simulate the expectation, and it is able to obtain convergence to a local minimum with a theoretical guarantee under mild conditions. The basic idea is to use a Monte Carlo sampling approximation; however, we replace this step with Gibbs sampling by utilizing the approximate conjugacy property.

**E-step:** Sampling \( \{X^{(j)}\}_{j=1}^J \) and \( \{R^{(j)}\}_{j=1}^J \) follows

\[
\begin{align*}
\alpha_n, \beta_n, \gamma_n | Z, \eta^{(k-1)} \\
X | \alpha_n, \beta_n, \gamma_n, Y \\
R | X, \alpha_n, \beta_n, \gamma_n, G
\end{align*}
\]

The true expectation or intractable integration \( Q^{(k)}(\eta) \) is approximately calculated by a stochastic averaging in a burn-in representation \( \hat{Q}^{(k)}(\eta) \) defined as (3.26), taking advantage of Gibbs sampling. During each E-step, infection parameters are in fact always updated at each inner iteration of Gibbs Sampling, thus making the latent variables \( X, R \) update based on different posterior distributions at each sampling, which disagrees with SAEM or MCEM, sampling latent variable from a fixed distri-
bution based on estimated parameter by previous M-step. Therefore the samples at later Gibbs sampling iterations are closer to the true posterior given current $\eta^{(k-1)}$. From this perspective, the Gibbs sampling in E-step may essentially accelerate the convergence rate in the next maximization step.

**M-step:** Maximizing with respect to $\eta$, i.e. $\arg\max Q^{(k)}(\eta)$. However, directly optimizing $\hat{Q}^{(k)}(\eta)$ will suffer from the same drawback as in standard EM. Pathological surfaces of the log-likelihood may be present via saddle points and local optima, meaning that the algorithm is sensitive to initialization. Delyon et al. (1999) argued that the augmented objective function $Q^{(k)}(\eta) \triangleq (1 - \delta^{(k)})Q^{(k-1)}(\eta) + \delta^{(k)}\hat{Q}^{(k)}(\eta)$ can avoid this problem partially, where $\hat{Q}^{(k)}(\eta)$ usually takes few samples to introduce a stochastic property, and $\delta^{(k)}$ is a small positive step size, essentially requiring the conditions in (3.25)

$$\lim_{k \to \infty} \delta^{(k)} = 0, \lim_{k \to \infty} \delta^{(k)}/\delta^{(k+1)} = 1, \sum_{k} \delta^{(k)} = \infty \tag{3.25}$$

The intuition to solve this intractable objective lies in Celeux et al. (1995), showing that this optimization can be updated by $\eta^{(k+1)} = (1 - \delta^{(k+1)})\eta^{(k+1)}_{EM} + \delta^{(k+1)}\eta^{(k+1)}_{SEM}$, where $\eta^{(k+1)}_{EM}$ is the true EM result approximated by MCEM with large sampling size, and $\eta^{(k+1)}_{SEM}$ is the special case of MCEM in very few samples or even unique one sometimes.

Generalizing this scheme to the Gibbs sampling setting, we formalize Algorithm 3, where $\hat{Q}_{bGEM}$ takes the sample average of the Gibbs algorithm and $\hat{Q}_{SEM}$ takes the last sample. $\hat{Q}_{SEM}$ is a stochastic perturbation of EM, and is expected to search more stable points. The algorithm starts by optimizing $\hat{Q}_{SEM}$ with $\delta^{(1)} = 1$, making the search area large for the first few steps. Then it focuses more weight on optimizing $\hat{Q}_{bGEM}$. A theoretical guarantee for this algorithm can be illustrated by using two convergence bounds; Birkhoff Ergodic theory Durrett (2010) and Theorem 7 in Delyon et al. (1999).
Algorithm 3 burn-in Gibbs EM Algorithm

**Input**: $Z$, $Y$, $G$, sampling size $J$, burn-in iteration $B$, step size series $\{\delta^{(k)}\}_{k=1}^{\infty}$

**Output**: $\eta$ and $X$

Initialize coefficient parameter $\eta^{(0)}$

while $\eta^{(k)}$ NOT Convergence do

for $i = 1, 2, \ldots, J$ do

sampling $\{X^{(j)}, R^{(j)}\}_{j=1}^{J}$ according to (3.24)

end for

Compute

$$\hat{Q}_{\text{bGEM}}^{(k)}(\eta) = \frac{1}{J-B} \sum_{j=B+1}^{J} \log \left( P(X^{(j)}, R^{(j)}, \eta | Z, \eta^{(k-1)}) \right)$$  \hspace{1cm} (3.26)

$$\hat{Q}_{\text{SEM}}^{(k)}(\eta) = \log \left( P(X^{(J)}, R^{(J)}, \eta | Z, \eta^{(k-1)}) \right)$$  \hspace{1cm} (3.27)

Optimization

$$\eta^{(k)}_{\text{bGEM}} = \arg \max \hat{Q}_{\text{bGEM}}^{(k)}(\eta)$$

$$\eta^{(k)}_{\text{SEM}} = \arg \max \hat{Q}_{\text{SEM}}^{(k)}(\eta)$$

Combination $\eta^{(k)} = (1 - \delta^{(k)})\eta^{(k)}_{\text{bGEM}} + \delta^{(k)}\eta^{(k)}_{\text{SEM}}$

end while

**Faster version for binary latent variables** Because taking the first order derivative with respect to $\eta$ and setting it equal to 0 will obtain a non-analytical root, gradient descent based optimization is necessary, and we adopt Newton’s Method by taking the advantage of curvature information. Taking the inverse of the Hessian matrix usually requires algorithmic complexity $O(K^3)$. The dimensionality of $\eta$ is $K$ which is independent of HMMs scale $N$, and a PCA preprocessing will reduce it significantly, where the necessity is illustrated in experiments section. Though $K$ representing the number of temporal health feature is less scalable in most application, there may still be a high cost to computing the Hessian with $O(JK^2)$ complexity due to matrix addition, unless there is a parallelized implementation with reduce operation Dean and Ghemawat (2008). Thus we need to seek for more efficient algorithm. For Gaussian variable, Price (1958) prove a theorem to address the exchangeability of the derivatives and expectations. Rezende et al. (2014) implemented this idea in a
non-Gaussian posterior likelihood and obtained good performance by approximating expectation with unique delicately designed sample. An improved SAEM coupled with MCMC is discussed in Kuhn and Lavielle (2004), which also argues that only one sample is required in the E-step if an appropriate Markov transition kernel is also used.

Consequently, we mimic these two ideas to design our MC integration with a single sample. Technically, if we omit the probability of tracking a source belonging to both inside and outside the network, latent variable $R_{n,t}$ can be considered as binary variable as well. Then, we can use the posterior mean of the latent variable as the sample we have been looking for. Therefore, at the $k$th iteration of EM, the pseudo-sample is constructed via a Bayesian decision rule based on the burn-in posterior mean in Gibbs sampling, i.e. $\hat{x}_{n,t} = \mathbb{I}\{\frac{1}{J-B} \sum_{j=B+1}^{J} x_{n,t}^{(j)} > 0.5\}$ and $\hat{R}_{n,t} = \mathbb{I}\{\frac{1}{J-B} \sum_{j=B+1}^{J} R_{n,t}^{(j)} > 0.5\}$. This means that a unique set $(\hat{X}, \hat{R})$ is sufficient to approximate $\hat{Q}^{(k)}(\eta)$, that is to say, $\log(P(\hat{X}, \hat{R}, \eta)|Z, \eta^{(k-1)})$ substitutes for $Q_{bGEM}^{(k)}$. This trick applied on non-Gaussian variables is not theoretically guaranteed but has been broadly used in EM or other optimization problems, by assuming a fully factorized joint distribution. In our binary variable case we found that it made no significant difference on accuracy whenever this trick is applied, in practice.

**Optimization** To optimize $\eta_{*EM}^{(k)}$ at the $k$th M-step, the update formula by the Newton-Raphson Method is briefly outlined in this paragraph, excluding the analytical gradient $G$ and Hessian $H$ computation. For efficiency, we update parameters as follows, with a few iterations.

$$\eta_{*EM, new}^{(k)} = \eta_{*EM, old}^{(k)} - \delta H^{-1}G$$

where *EM varies according to different estimators, bGEM or SEM. It is unnecessary for there to be complete convergence in order to guarantee $Q(\eta^{(k)}) > Q(\eta^{(k-1)})$. A similar idea with a single iteration is mentioned in Lange (1995). The step size $\delta$
ensures that the Wolfe conditions (Nocedal and Wright (2006)) are satisfied. The intuition in adding in step size here is, compared with gradient descent, is that Newton's Method tends to make more progress in the right direction of the local optima, due to the property of affine invariance. This probably leads to an update where the step size is too large, so it is better for stochastic algorithms to enlarge the search domain at first then shrink later.

3.1.13 Short Discussion on Sigmoid link

A sigmoid link function benefits from model simplicity and hiding the infection parameters without the necessity to integrate them out. The likelihood $P(\eta, X|Z)$ shown in (3.28) can thus be exactly computed. It means that we can estimate parameters throughout either standard SAEM by getting rid of latent variable $R$ immediately, or bGEM by introducing $R$ in E-step and a faster M-step by keeping $\hat{X}$ alone.

$$P(\eta) \prod_{n=1}^{N} P(x_{n,1}) \times \prod_{n=1}^{N} \prod_{t=1}^{T-1} \sigma(z_n^T \eta_r)^{I[z_n,t=1, x_{n,t+1}=0]} (1 - \sigma(z_n^T \eta_r))^{I[z_n,t=1, x_{n,t+1}=1]}
\cdot (1 - (1 - \sigma(z_n^T \eta_0))(1 - \sigma(z_n^T \eta_b))^{C_{n,t}})^{I[x_{n,t}=0, x_{n,t+1}=1]}
\cdot ((1 - \sigma(z_n^T \eta_0))(1 - \sigma(z_n^T \eta_b))^{C_{n,t}})^{I[x_{n,t}=0, x_{n,t+1}=0]}$$

(3.28)

3.1.14 Further Discussion on $\beta_n$

In the second biological interpretation of $\beta_n$ (the probability of infecting others), transition function $\phi_n, x_n; (n,n') \in G_t$ will become dependent on a parameter set $\{\beta_{n'} : n' \in S_{n,t}\}$. Consequently, the posterior of each $\beta_n$ requires both a count number and source tracking (conceptually, this is like a "pointer" in the C programming language). However, the likelihood of the beta-exponential model can be simplified to integrate out these parameters due to the auxiliary variable $R_{n,t}$ as well, corre-
sponding to a new approximate categorical distribution, though \( P(R_{n,t}) \) in previous (3.20) actually aggregates the probability with respect to all equal \( \beta_n \)s. The new categorical distribution and its induced completed likelihood can be represented as follows.

\[
P(R_{n,t}) \approx \text{Categorical} \left( \frac{\alpha_n \prod_{n' \in S_{n,t}} (1 - \beta_{n'})}{1 - (1 - \alpha_n) \prod_{n' \in S_{n,t}} (1 - \beta_{n'})} \frac{(1 - \alpha_n) \beta_{n'}}{1 - (1 - \alpha_n) \prod_{n' \in S_{n,t}} (1 - \beta_{n'})}, \ldots \right)
\]

(3.29)

\[
P(X, R, \eta | Z) = P(\eta) \prod_n \frac{B(e^{z_n} \eta_{n,1} + C_{n,1} - 0, e^{z_n} \eta_{n,2} + C_{n,1} - 1)}{B(e^{z_n} \eta_{n,1}, e^{z_n} \eta_{n,2})} \frac{B(e^{z_n} \eta_{n,0} + C_{n,R_n = 0}, e^{z_n} \eta_{n,0} + C_{n,R_n = 0} - 0)}{B(e^{z_n} \eta_{n,0}, e^{z_n} \eta_{n,0})} \frac{B(e^{z_n} \eta_{n,0} + C_{n,R_n = n}, e^{z_n} \eta_{n,2} + C_{n,R_n = n})}{B(e^{z_n} \eta_{n,0}, e^{z_n} \eta_{n,2})}
\]

(3.30)

where \( R_{n,t} \) takes the value \( \{0, 1, \ldots, C_{n,t}\} \), and 0 means there is an outside network source and other integers refer to specific infection in-network sources. The categorical distribution makes the beta prior for the infection parameters conjugate in the posterior. However, the integral for the likelihood is actually difficult and needs some tricks, especially for \( \beta_n \) because of the source tracking (see Appendix B for details). Note the likelihood for sigmoid link can be derived analogously. The new count notations are listed below.

\[
\begin{align*}
C_{n,R_n = 0} &= \sum_t \mathbb{I}[R_{n,t} = 0] \\
C_{n,R_n \neq 0} &= \sum_t \sum_{n' \in S_{n,t}} \mathbb{I}[R_{n,t} = n'] \\
C_{n,R = n} &= \sum_{n', t, m \in S_{n', t}} \mathbb{I}[R_{n', t} = n] \\
C_{n,R \neq n} &= \sum_{n', t, m \in S_{n', t}} [\mathbb{I}[R_{n', t} = 0] + \mathbb{I}[x_{n', t} = 0, x_{n', t+1} = 0]]
\end{align*}
\]

3.2 Variational Inference Approach

In our section, we propose a novel learning approach in the variational inference (VI) framework. Differing from the traditional VI methods, such as mean-field Beal
(2003), we get rid of the complicated gradient computation for non-conjugate probabilistic models but build a tractable dynamic recognition model corresponding to the generative process. The recognition model is equivalent to the variational distribution in VI parlance. Therefore we can minimize the Kullback-Leibler (KL) divergence or maximize the corresponding lower bound for parameter estimation. Taking advantage of the binary variables in the hGCHMM, the recognition model can be completely constructed by a sigmoid belief network (SBN) Sutskever and Hinton (2008), where it is straightforward to generalize to the categorical case by inducing softmax output, thus allowing our approach to reproduce the potential tasks introduced in Fan et al. (2015c). In addition, we overcome the major drawback – lacking the capability to simulate highly contagious disease, by proposing a dynamic auto-encoding variational inference method. In the experiments section, we compare our algorithm with the Gibbs EM Fan et al. (2015c) and mean-field Beal (2003) versions, and achieve competitive performance in the small infection rate case but outperform them in the large rate scenario.

### 3.2.1 Limitation of Gibbs Sampling for \( \phi \) Approximation by Auxiliary Variable

A critical approximation of transition function is

\[
p_n^t(0 \rightarrow 1) = 1 - (1 - \alpha_n)(1 - \beta_n)C_n^{t-1}.
\]  

Eq. (3.31) is an exception unsatisfying the Bernoulli-Beta conjugate in generative model (3.1). By inducing auxiliary variable \( R \), which indicates the infection source from outside, inside or both of the surveyed community, an approximate Gibbs scheme can be developed. Particularly, Dong et al. (2012) used a simple Taylor expansion \( \alpha_n + C_n^t\beta_n \) to represent (3.31), and Fan et al. (2015c) further applied a polynomial decomposition trick by rewriting it as the summation of three terms,

\[
\alpha_n(1 - \beta_n)C_n^{t-1} + C_n^{t-1}(1 - \alpha_n)\beta_n + C_n^{t-1}\alpha_n\beta_n
\]  

(3.32)
where \((1 - (1 - \beta_n)^{C_{t-1}^n})\) is approximated by \(C_{t-1}^n \beta_n\). Admittedly, (3.32) is a better estimation of (3.31) than \(\alpha_n + C_{t}^n \beta_n\), and also favors the Bernoulli-Beta conjugate for potential Gibbs sampling. We have to notice that if the condition \(\alpha, \beta \approx 0\) does not hold, the resulted error of (3.32) will be crucially non-negligible and violate the entire Bayesian scheme, thus leading a biased estimation. In Fig. 3.7, we plot the infection probability as the function of \(\alpha, \beta\) with difference approximation. Notice that (a) The bottom surface is the exact plot for \(p_{n}^t(0 \rightarrow 1)\); the middle one is the approximation by (3.32); the top one is the approximation by \(\alpha + C \beta\). In this plot, we set \(C = 2\). (b) Rescaled plot of (a), where the value is divided by the maximum of approximated Taylor expansion, i.e. 2 and 3 respectively.

\[\text{Figure 3.7: The error visualization.}\]

The visualized comparison shows when \(\alpha\) or \(\beta\) is sufficiently large, the approximation is not even a probability at all, which should be bounded by 1. One plausible solution is to rescale, but this hack trick lacks any theoretical or intuitive explanation (the rescaled figure is shown in supplementary). Therefore, the Taylor expansion restricts the generalization ability of this model, so we are intent to modify the inference method by using variational inference.
3.2.2 Variational Inference

Suppose we are interested in a latent variable model represented as distribution $P_\Phi(X, Y)$ parametrized by $\Phi$, where $X$ is latent variable and $Y$ is observed variable. The purpose of learning task is to estimate the posterior of latent variable $P_\Phi(X|Y)$ and $\Phi$. In most case, the exact inference is intractable and thus a variational lower bound on the marginal log-likelihood is often derived to be maximized, because optimization problem can be naively solved by gradient based algorithm. Particularly, we need induce a variational distribution $Q_\Psi(X|Y)$ with parameters $\Psi$, which is selected with the intention of being similar to the true posterior $P_\Phi(X|Y)$. A prevalent construction of $Q$ will usually be assumed to factorize over some partition of the latent variables, i.e mean-field method.

With simple mathematical derivation, we have

$$
\log P(Y) = \mathbb{E}_Q \left[ \log \frac{P_\Phi(X, Y)}{Q_\Psi(X|Y)} \right] + KL(Q_\Psi(X|Y)||P_\Phi(X|Y))
$$

$$
\geq \mathbb{E}_Q \left[ \log P_\Phi(X, Y) - \log Q_\Psi(X|Y) \right] = \mathcal{L}(Y, \Phi, \Psi)
$$

For the lower bound $\mathcal{L}$, maximizing it with respect to $\Phi, \Psi$ is equivalent to minimizing the KL divergence between proposed distribution and true posterior. The tightness of this bound holds when $Q$ exactly recovers true posterior. By examining the bound, it does not rely on the form of $Q$. In our work, the variational distribution is restricted to belong to a family of distributions of simpler form than $P_\Phi(X|Y)$, but preferably flexible enough to contain or close the true posterior as a solution. To be specific, we reverse the directed edge between latent and observed variable in generative process and enforce a sigmoid belief nets to simulate this link.
3.2.3 Dynamic Recognition Model

Before we describe the details of our recognition model, we first revisit the generative model and reformulate part of the process to be deterministic. Due to the fact that the expectation of distribution \( \text{Beta} \left( e^{x^T \eta_{.1}}, e^{x^T \eta_{.1}} \right) \) is \( \sigma \left( (\eta_{.1} - \eta_{.2})^T z_n \right) \), where \( \sigma(x) = \frac{1}{1+e^{-x}} \) is sigmoid function, we can simplify the parameterization by substituting a single \( \eta \) for \( \eta_{.1} - \eta_{.2} \), if we apply a sigmoid belief net with input \( z_n \). One advantage of this approximation is to reduce the number of parameters in generative model since \( \alpha_n, \beta_n, \gamma_n \) can implicitly vanish from the arguments in \( \phi \), leading to a more direct form \( \phi(\eta, z_n) \). In the contrary, the disadvantage is apparently the deficiency of uncertainty for these intermediate variables. However, point estimation of associating parameters is also acceptable in the medical application, since the most concerned issue is to discover heterogenous infection probability on every single day and how the individual covariates \( z_n \) influence the personal physical constitution. The advantage of sigmoid formulation is to benefit our subsequent gradient based algorithm. Analogously, we can rewrite the generative process of \( y_{n,t,s} \) by using sigmoid function, i.e. \( P(y_{n,t} = 1) = \theta_{0,s} 1_{(x_{n,t} = 0)} + \theta_{1,s} 1_{(x_{n,t} = 1)} = \sigma(w_s x_{n,t} + b_s) \), where \( w_s \) and \( b_s \) become model parameters in \( \Phi \).

With the reformulated generative process, we can readily construct the a simple dynamic recognition model \( Q_\psi(X|Y, Z) \) by applying sigmoid belief nets. Notice that for clarity we write the observed variables \( Y, Z \) separately, which is a slightly different from the general discussion.

\[
x_{n,t} \sim \text{Bernoulli} \left( \sigma(\omega^T \tilde{x}^{t-1}_n + \nu^T y^t_n + \kappa^T z_n + b) \right)
\]  

(3.34) where the vector \( \tilde{x}^{t-1}_n = g^{t-1}_n \odot \mathbb{1}_{x_{n,t-1}=1} \), and \( g^{t-1}_n \) is the \( n \)th column of \( \mathbb{G}^{t-1} \) and \( \odot \) is element-wise multiplication operator. (3.34) indicates the graphical representation of recognition model has only two modifications of Fig.3.4: getting rid of redundant intermediate variable by inducing link between \( z_n \) and \( x_{n,t} \); reversing the direction.
from $x^t_n$ to $y^t_n$. The powerful approximation ability of sigmoid belief nets allows it to obtain perfect estimation of any transition function. This is key reason why we did not make the same assumption as Fan et al. (2015c). Additionally, we can further plug in one more hidden layer $h^t_n$ following the convention of (3.34), and then let $x_{n,t} \sim$ Bernoulli $(\sigma(\omega^+_h h^t_n + b_h))$. The deep architecture of networks can enlarge the representative ability, and succeeds with great improvement in many machine learning areas, such as collaborative filtering Salakhutdinov et al. (2007), or document modeling Srivastava et al. (2013). In our paper, deep structure is not the main issue we discuss, since we found it had no significant improvement and brought more variational parameters to estimate.

### 3.2.4 Parametrization and Optimization

In previous section, we did not specify the index or subscript of variational parameters $\omega, \nu, \kappa, b$ in recognition model. By observing the factorized structure of generative model (3.35), we analyze different parameterization possibilities for (3.34).

$$P_\Phi(X, Y) = \prod_{n=1}^{N} \prod_{t=1}^{T} p(y^t_n| x^t_n) p(x^t_n| \tilde{x}^{t-1}_n)$$  \hspace{1cm} (3.35)

where we hide the dependence on $z_n$ for simplicity but without ambiguousness, and $x^0_n$ can either be non-exsited node or the initial node without emission. Therefore, according to previous discussion, the corresponding recognition model for approximating $P_\Phi(X|Y)$ has the following form.

$$Q_\Psi(X|Y) = \prod_{n=1}^{N} \prod_{t=1}^{T} q(x^t_n| \tilde{x}^{t-1}_n, y^t_n)$$  \hspace{1cm} (3.36)

In order to take the advantage of variational inference principle established in (3.33), an ideal factor by factor optimization between the logarithm form of $P$ and $Q$ inspires three main parameterization methods.
Temporal Parametrization

We emphasize the description on the most natural temporal parametrization and demonstrate our learning signal Sutton and Barto (1998) based optimization algorithm. In this setting, each factor $q$ can be formulated as

$$q(x^t_n = 1) = \sigma(\omega^T \tilde{x}^{t-1}_n + \nu^T y^t_n + \kappa^T z_n + b_t)$$ (3.37)

This allows the dynamic parameters shared by different chains, and is equivalent to train the model with a unique datapoint $Y_{N \times T \times S}$. The reason is the temporal parameters $\Psi_t = \{\omega_t, \nu_t, \kappa_t, b_t\}$ can be updated locally. Each set of temporal parameters is merely associated with the correspondent observed data $y^t_{1:N}$. To make this argument concrete, it can be shown by integration by parts for the derivative of lower bound (3.33). For notation simplicity, we denote the learning signal $l_\Psi = \log P_\Psi - \log Q_\Psi$, and then have the following temporal decomposition.

$$\nabla_{\Psi} \mathcal{L} = \mathbb{E}_Q[l_\Psi \nabla_{\Psi} \log q(x^t_n = 1|\tilde{x}^{t-1}_n, y^t_n)]$$

$$= \mathbb{E}_q(x^t_{1:N} | y^t_{1:N}) \left[ \mathbb{E}_q(x^{t-1}_{1:N} | y^{t-1}_{1:N}, y^t_{1:N}) \mathbb{E}_q \nabla_{\Psi} \log q(x^t_n = 1|\tilde{x}^{t-1}_n, y^t_n) | x^{t-1}_{1:N} \right]$$ (3.38)

Additionally, (3.38) can also enable the possibility of temporal learning signal, $l'_\Psi = \log P(x^{t:T}_{1:N}, y^{t:T}_{1:N} | x^{t-1}_{1:N}) - \log Q(x^{t:T}_{1:N}, y^{t:T}_{1:N} | x^{t-1}_{1:N})$. Thus, $\Psi_t$ can be locally updated by $l'_\Psi$. To fully adopt the reinforcement learning trick, we also induce an observation dependent but latent variable independent signal baseline $B_{\Xi_t}(y^t_{1:N})$ temporally parametrized by $\Xi_t$, which is also implemented by SBN. Since the identity property

$$\mathbb{E}_Q[(l_\Psi - B) \nabla_{\Psi} \log q_\Psi] = \mathbb{E}_Q[l_\Psi \nabla_{\Psi} \log q_\Psi]$$ (3.39)

it works practically in gradient variance reduction Mnih and Gregor (2014); Mnih et al. (2015). Therefore, we summarize the Algorithm 6. It is noticed that we only provide the basic gradient ascent algorithm, whereas many existed advanced trick
can be explored as well, such as the adaptive learning rate, RMSprop, or Adagrad. Compared with mean-field algorithm, the burden of derivative computation in our algorithm is relatively simple. Due to the limited space, the detailed derivation shows in the supplementary materials. Even if we deploy a deep SBN, the convenience of deriving sigmoid function makes our algorithm efficient. As we discussed before, if a hidden layer plugs into the SBN, we can actually sample this layer similarly as Line 3 in Algorithm 6. When it comes to compute the gradient associated with hidden layer, it is not necessary to use the back-propagation, which is a standard method in feed forward neural networks. We merely need to compute the gradient of sigmoid function twice, since given the sampled hidden layer the input and the output are independent.

**Personal Parametrization**

Analogously, the subscript of variational parameters can be heterogeneously indexed by $n$.

$$q(x_n^t = 1) = \sigma(\omega_n^T x_n^{t-1} + \nu_n^T y_n^t + \kappa_n^T z_n + b_n) \quad (3.40)$$

The setting means the variational parameters $\Psi = \{\omega_n, \nu_n, \kappa_n, b_n\}_{n=1}^N$ would not change dynamically but differ chain by chain, thus being equivalent to train the model with dataset $\{y_{1:N}\}_{t=1}^T$. It also indicates $\Psi$ can not be trained temporally as previous discussion but allows centralization and normalization of learning signal for variance reduction. For Algorithm 5, the baseline becomes global, and $l^t_{\Psi}$ has the different meaning from previous one.

**Node-wise Parametrization**

More specifically, we can even construct the node-wise parameterization $\Psi_n^t = \{\omega_n^t, \nu_n^t, \kappa_n^t, b_n^t\}$. The formulation and the detailed algorithm framework are similar to a combination of previous two settings, so we include the Algorithm in the supplementary. All
Algorithm 4 Temporal Sigmoid Variational Inference

Initialization: By normal distribution with small variance;
1: while \((\Phi, \Psi, \Xi)\) Not Converge do
2: for \(t = 1, \ldots, T\) do
3: Sample \(x_{1:N}^t \sim q(x_{1:N}^t|x_{1:N}^{t-1}, y_{1:N}^t)\);
4: Compute temporal learning signal \(l^t_{\Psi}\);
5: Subtract baseline \(l^t_{\Psi} \leftarrow l^t_{\Psi} - B_{\Xi}(y_{1:N}^t)\);
6: \(\Psi_t \leftarrow \Psi_t + \epsilon \cdot l^t_{\Psi} \nabla_{\Psi_t} \log q(x_{1:N}^t|x_{1:N}^{t-1}, y_{1:N}^t);\)
7: \(\Xi_t \leftarrow \Xi_t + \epsilon \cdot l^t_{\Psi} \nabla_{\Xi_t} \log B_{\Xi}(y_{1:N}^t);\)
8: end for
9: \(\Phi \leftarrow \Phi + \epsilon \cdot \sum_{t=1}^{T} \nabla_{\Phi} \log p(x_{1:N}^t, y_{1:N}^t);\)
10: end while

three parameterized recognition models do not assume any quantitative magnitude on the parameters, and the only approximation induced in this framework is due to the inequality (3.33). This gap is hopefully filled by the gradient based optimization algorithm. Furthermore, the recognition model of hGCHMM is particularly counter-intuitive, since its parameterization shows a trade-off between the complexity of graphical model and the number of variational parameters. To approximate the same true posterior, our approach is to propose either a simple model with more parameters or complex one with less parameters (See Fig. 3.8 for a graphcial illustration). In our experiments, the \(L_2\) norm penalty on parameters is also implemented.
Algorithm 5 Personal Sigmoid Variational Inference

1: while (Φ, Ψ, Ξ) Not Converge do
2:   for t = 1, ..., T do
3:     Sample \( x_{t}^{d} \sim q(x_{t}^{d} | x_{t-1}^{d}, y_{t}^{d}) \);
4:     Compute temporal learning signal \( l_{t}^{\psi} \);
5:     Subtract baseline \( l_{t}^{\psi} \leftarrow l_{t}^{\psi} - B_{\Xi}(y_{t}^{d}) \);
6:   end for
7:   Normalize \( \{l_{t}^{\psi}\}_{t=1}^{T} \);
8:   \( \nabla_{\psi} L = \sum_{t=1}^{T} l_{t}^{\psi} \nabla_{\psi} \log q(x_{t}^{d} | x_{t-1}^{d}, y_{t}^{d}) \);
9:   \( \nabla_{\Xi} L = \sum_{t=1}^{T} l_{t}^{\psi} \nabla_{\Xi} \log B_{\Xi}(y_{t}^{d}) \);
10:  Update \( \Phi \) (as Alg. 1), \( \Xi \) and \( \Psi \) by gradient ascent;
11: end while

Algorithm 6 Node-wise Sigmoid Variational Inference

Initialization: By normal distribution with small variance;
1: while (Φ, Ψ, Ξ) Not Converge do
2:   for \( n = 1, \ldots, N \) do
3:     for \( t = 1, \ldots, T \) do
4:       Sample \( x_{n}^{t} \sim q(x_{n}^{t} | x_{n-1}^{t}, y_{n}^{t}) \);
5:       Compute temporal learning signal \( l_{n}^{t} \);
6:       Subtract baseline \( l_{n}^{t} \leftarrow l_{n}^{t} - B_{\Xi}(y_{t}^{1:N}) \);
7:     end for
8:     \( c_{n} = \text{mean}\{l_{n}^{t}\}_{t=1}^{T} \), \( v_{n} = \text{variance}\{l_{n}^{t}\}_{t=1}^{T} \);
9:     Normalize \( \{l_{n}^{t}\}_{t=1}^{T} \);
10:   end for
11:   for \( n = 1, \ldots, N \) do
12:     for \( t = 1, \ldots, T \) do
13:       \( \Psi_{n}^{t} \leftarrow \Psi_{n}^{t} + \epsilon \cdot l_{n}^{t} \nabla_{\psi_{n}} \log q(x_{n}^{t} | x_{n-1}^{t}, y_{n}^{t}) \);
14:     end for
15:   end for
16:   \( \Xi_{t} \leftarrow \Xi_{t} + \epsilon \cdot \left( \sum_{n=1}^{N} l_{n}^{t} \right) \nabla_{\Xi_{t}} \log B_{\Xi_{t}}(y_{t}^{1:N}) \);
17:   \( \Phi \leftarrow \Phi + \epsilon \cdot \sum_{t=1}^{T} \nabla_{\Phi} \log p(x_{1:N}^{t}, y_{1:N}^{t}) \);
18: end while

3.3 Experiments

3.3.1 Data Description

We apply our inference method to two flu diffusion datasets. The first experiment is based on MIT social evolution data. The dynamic social contacts can be summarized from the daily bluetooth data, thus resulting in \( G_{t}, t = 1, \ldots, 107 \). In addition, the personal health habits \( z_{n} \) for each person contain 9 features, weight, height, salads per week, veggies fruits per day, healthy diet level, aerobics per week, sports per...
week, smoking indicator, and default feature 1. Since the ground truth of infectious state is latent variable or unknown, we need to simulate consistent $X$ and $Y$ for evaluation by generative model, though self-reported symptom $Y$ is in fact provided in the dataset.

Another dataset is exFlu survey. This study is conducted in a college dormitory during a chain referral recruitment process carried out from September 2012 to January 2013. 103 enrolled students participated flu survey by the smartphone installed with health apps. Besides the covariates described in MIT data, the temporally unchanged features of exFlu also include gender, age, average times of hand washing by sanitizer, and indicator for vaccination or flu shot. The data type for dynamic social networks $G$ and daily symptoms $Y$ is exactly the same as above dataset. However, this survey has a special treatment on participants with severe symptoms to diagnose whether the specific person is infected and record the flu duration since onsite. Thus, it allows us to evaluate the performance of our approach comparing with expertise.

![Overall Social Network](image)

**Figure 3.9: Overall Social Network**

103 (17.5%) students of the 590 enrolled participants were equipped with provided smartphones and joined the iEpi sub-study. They were required to use their iEpi smartphone and could report their symptoms, meeting the study criteria for ILI. A total of 4843 contextually-based surveys were administered on all sub-study smartphones (mean 62.09/day), 1743 (36.0%) of which were responded to by iEpi
sub-study participants (mean 22.35/day). There were a total of 60131 Bluetooth contacts between smartphones within the iEpi sub-study, and 148,333 total Bluetooth contacts with other devices of any kind, averaging 7.48 contacts/phone/day and 20.95 contacts/person/day, respectively.

Notice that in the Figure 3.10, (a) iEpi Bluetooth network (N=103): Network of Bluetooth contacts between smartphones in the iEpi sub-study; (b) Sampled dynamic social networks from (a): 103 dots uniformly distributed as a large circle. Contacts within the network account for edges between solid dots. The bluetooth detector can automatically collect contacts occurring between iEpi installed smartphones, or to other smart devices. Each node (circle) in Figure 3.10(a) represents an individual in the sub-study, and the links (edges) between nodes represent bluetooth detections between smartphones of individuals within the sub-study networks. Node size is proportional to the total number of contacts detected in the bluetooth data (equivalent to degree), and the link thickness indicates the contact duration between the two nodes (equivalent to weight on edge). During the experiment period, we also conducted a comparison test. Some participants (yellow nodes in Figure 3.10(a)) were isolated for three days at the onset of illness, which means no social contacts were made during this period.

The next step is to extract daily social networks. We use the 77 days of the iEpi survey data which is relatively complete, and its corresponding bluetooth data to construct dynamic networks. Figure 3.10(b) illustrates 4 independently sampled sub-networks, i.e. $G_t$, $t = 2, 27, 52, 77$. To make more sense of the edges, only the bluetooth data showing the total contact duration between two participants lasting more than 10 minutes will contribute to an edge on that day. The threshold of 10 minutes can be adjusted to make the graph denser or sparser, thus leading to a higher or lower computational cost.
3.3.2 Evaluation of Gibbs-EM

We ran the algorithm 10 times. The prediction performance on latent variable $X$ is the byproduct of the E-step, and when $x_{n,t}$ is larger than the threshold 0.5 the person is diagnosed as being infected. Since $X$ is completely unknown to the algorithm, held-out test data prediction is unnecessary, while the whole matrix $X$ is used to evaluate prediction accuracy. Notice in Figure 3.15, it shows the posterior $P(X|Y)$ estimation, where $x$-axis represents day and $y$-axis means id number of participants. (a) A binary matrix with 1 indicating infected; (b) Posterior mean estimated by SigVI. (c) Resulted binary matrix by applying a 0.5 threshold on (v); (d) Posterior mean estimated by mean-filed; (e) Posterior mean estimated by Gibbs EM. (a-c) shows the difference between the truth and the inferred results for each of the two
linked models. The posterior mean from Gibbs sampling for prediction, in both beta-exponential and sigmoid models, leads to a real value in the interval \([0, 1]\). Notice that in Figure 3.12, (a) is accuracy comparison on latent variable prediction, revealing a quantitative measurement on accuracy with a standard deviation; (b-d) shows the error bar comparison on parameter estimation, where the \(N\)-dimensional error vector is normalized to a scalar for statistical comparison. As mentioned before, the baseline model is a two-step algorithm including the standard GCHMM and logistic regression is also implemented and compared to. The rightmost barplot in Figure 3.12(a) shows its predictive performance. The GCHMM needs to run at least 2000 iterations of Gibbs sampling to obtain good mixing, while in our approach, we only run about 50 inner Gibbs sampling iterations in E step and less than 10 outer EM iterations.

![Figure 3.12: Results of infection rate estimation.](image)

Figure 3.12(b-d) displays the predictive error of the forecasted infection parameters. Since the infection parameters are individual specific, the estimation is in fact a vector of length \(N\). Therefore we used the 2-norm of the error vector for statistic summarization and further comparison. It is apparent that the sigmoid model gives the best performance on latents \(X\) or \(\gamma_n, \alpha_n, \beta_n\), in terms of the generative model. The defined Beta-exponential Model, as a probabilistic substitute to the approximate sigmoid transformation on infection parameters, proves it is competitive for parameter estimation. However, standard GCHMM with logistic regression, as
two independent parts of the sigmoid model, provides an unreliable prediction on individual-specific parameters, albeit its excellent latent variable inference. All three inference methods use Gibbs sampling to infer the posterior mean of $X$. This is most likely the reason why they share similar performance.

![Figure 3.13: Scatter plot of infection rate estimation.](image)

**Individual Parameter Analysis**

From the perspective of general health care or disease control for large populations, $\eta$ is of concern (discussion on a real biological dataset later). However, as for individual treatment and personal medical advice, $\gamma_n, \alpha_n, \beta_n$ are more significant for physical health. Better immunity usually indicates a smaller $\alpha_n, \beta_n$ but a larger $\gamma_n$. In our model, these parameters are designed to correlate with personal health habits by using a link with influence coefficient $\eta$. The prediction of the infection parameters on raw data $Z$ is shown in Figure 3.13(a-c). Notice that in (e-g) the horizontal axis of the scatter plot illustration is estimation, while the vertical axis is ground truth. The colinearity elimination on $\gamma_n$ is observed, and PCA justification $\gamma$ is to obtain the regressed slope close to 1. This illustration is consistent with the error bar plot in Figure 3.12(b-d). The predicted values of our proposed models are distributed with higher concentration on the diagonal of 2D-coordinate plane, while standard GCHMM + logistic regression has relatively larger variance. The underlying linear slope for $\gamma_n$ seems inconsistent with the diagonal. This phenomenon can be blamed on the colinearity of $Z$. Looking at the names of the covariates (BMI$^2$, weight, 2 Body mass index$= \frac{m}{h^2}$, where $m$ is mass/kg and $h$ is height/m.

66
height, salads per week, veggies and fruits per week, healthy diet or not, aerobic per week, sports per week, smoking or not), correlation obviously exists. Thus, we apply Principal component analysis (PCA) on $\mathbf{Z}$, and then select the first 4 main components (explanatory power 99.9%) and the default feature 1. We next obtain the scatter plot of PCA justified $\gamma_n$ in Figure 3.13(d). Results imply that PCA can eliminate colinearity effectively; however, the interpretability may not as effortless as Figure 3.13(a).

Table 3.2: Coefficients Estimation on exFlu Dataset

<table>
<thead>
<tr>
<th>Feature $^3$</th>
<th>Recovery $\eta_r$</th>
<th>Outside Infect $\eta_a$</th>
<th>Inside Infect $\eta_b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default=1</td>
<td>-1.3022 ± 0.0146</td>
<td>-5.1517 ± 0.0024</td>
<td>-4.1619 ± 0.0281</td>
</tr>
<tr>
<td>Gender</td>
<td>-0.1575 ± 0.0118</td>
<td>-0.2428 ± 0.0074</td>
<td>-0.1457 ± 0.0078</td>
</tr>
<tr>
<td>Age</td>
<td>0.0074 ± 0.0082</td>
<td>-0.2376 ± 0.0051</td>
<td>-0.0181 ± 0.0017</td>
</tr>
<tr>
<td>Alc_Day</td>
<td>0.1090 ± 0.0078</td>
<td>-0.1534 ± 0.0003</td>
<td>-0.0410 ± 0.0018</td>
</tr>
<tr>
<td>Vacc_Ever</td>
<td>-0.0698 ± 0.0104</td>
<td>0.1092 ± 0.0095</td>
<td>0.0382 ± 0.0085</td>
</tr>
<tr>
<td>Flushot_Yr</td>
<td>0.0769 ± 0.0092</td>
<td>-0.3209 ± 0.0073</td>
<td>0.0837 ± 0.0055</td>
</tr>
<tr>
<td>Smoker</td>
<td>-0.1080 ± 0.0029</td>
<td>-0.0536 ± 0.0008</td>
<td>0.0773 ± 0.0021</td>
</tr>
<tr>
<td>Drinker</td>
<td>-0.1335 ± 0.0092</td>
<td>0.0628 ± 0.0030</td>
<td>0.1408 ± 0.0029</td>
</tr>
<tr>
<td>Act_Days</td>
<td>0.0356 ± 0.0099</td>
<td>0.0054 ± 0.0063</td>
<td>-0.0622 ± 0.0078</td>
</tr>
<tr>
<td>Sleep_Qual</td>
<td>0.0225 ± 0.0069</td>
<td>-0.3686 ± 0.0051</td>
<td>-0.0162 ± 0.0077</td>
</tr>
<tr>
<td>Wash_Opt</td>
<td>0.0024 ± 0.0103</td>
<td>0.0816 ± 0.0132</td>
<td>-0.0714 ± 0.0048</td>
</tr>
<tr>
<td>High_Risk</td>
<td>-0.1274 ± 0.0116</td>
<td>-0.1252 ± 0.0058</td>
<td>-0.0727 ± 0.0007</td>
</tr>
</tbody>
</table>

Available illness onset diagnoses in our experiment allows for the evaluation of inferred infection states. We tried all three models, Sigmoid link, Beta-exponential link and standard GCHMMs+LogReg on this dataset. Because of the specific quantized distribution of diagnosed flu onset (see red short pattern of left graph in Figure 3.11), the three methods perform stably, but give different results over 10 runs with no standard deviation. Though they all heavily rely on Gibbs sampling, the sigmoid link model can detect more short term patterns than the other two. Table ?? gives

$^3$ Gender 1 means female; Alc_Day: average number of times hand washing with sanitizer; Vacc_Ever: previous vaccination; Flushot_Yr: vaccination this year; Act_Days: exercise in broad sense per day; Wash_Opt: whether wash hands exceeding 20s; High_Risk: contact with impaired immunity patient.
both precision and recall for prediction, since the proportion of positive instances, unlike our simulation, is about one tenth. Even the sigmoid model missed some very short patterns. Two reasons may contribute to this phenomena; the first is that HMMs are a long distance dependent models; the second is that we find symptom reports for short period disease courses are always low severity.

In contrast to other models, and serving as the mainstay and novelty of this paper, we aimed to learn how personal features (first column in Table 3.2) were associated with individual flu vulnerability, i.e. coefficients $\eta$. A Sigmoid transform on $\eta$ will immediately give infection parameters. Larger $\gamma_n$ implies better resistance, while larger $\alpha_n, \beta_n$ indicates increased vulnerability. Because resistance or vulnerability is not an experimental quantity (difficult to measure in a real world dataset), we prefer to evaluate coefficients $\eta$ (Table 3.2) rather than actual infection parameters. The right three columns are the estimated $\eta$s associated with different biological meanings (indicated by their subscripts) in the Sigmoid model—possessing the best performance in both the simulation and real cases. Looking at the feature column, we can see that females seems suffer from a slower recovery but are not as likely to catch a cold. Another important factor is whether participants are addicted to alcohol. Drinkers significantly aggravate body immunity. However, whether or not one washes their hands for more than 20s, interestingly, seems not to be significant to the model, especially to the recovery rate. This may be blamed on an overly long washing duration–20s in the experimental design. Overall, the sign consistency with respect to $\eta$ makes sense, with the exception of a few relationships. For the sigmoid function, positive coefficients will enlarge infection parameters, and vice versa.

3.3.3 Varitional Inference for Different Infection Rates

In this experiment, we mainly study the generalization ability of our approach for various diseases by synthesizing 3 different infection rates. The data we used in this
section is partially simulated MIT data with true \( G_t \) and \( z_n \). Since the infection rate is crucially heterogenous in our model, the magnitude mentioned later or the severity of contagiosity is virtually the mean value of these person-specific rates. Basically, the three contagious diseases are usual flu with recovery rate 0.3 and low infection rate 0.01 (outside) and 0.02 (inside), severe flu (such as H1N1) with lower recovery rate 0.2 and high infection rate 0.34 and 0.24, and a completely artificial flu with high recovery rate 0.6 but relatively high infection rate 0.06 and 0.13. The demo of dynamic prediction is recorded in the .avi files of supplementary.

![Figure 3.14](image.png)

**Figure 3.14**: (a) Accuracy v.s threshold. (b) ROC curve comparison

In this setting, we evaluate the predicting performance on simulated infected states \( X_{N \times T} \), with \( N \) participants in \( T \) days. The algorithms we tested are node-wise sigmoid variational infection, mean-field variational bayesian inference and Gibbs sampling with EM Fan et al. (2015c). Mean-field is the most standard variational inference approach; however, its derivation is extremely complicated for the non-conjugate model. In our case, some variables like \( \gamma \), \( Y \), belonging to conjugate exponential (CE) family, have a nice variational EM updating formula, which is actually similar to the full conditional in Gibbs sampling. However, for other variables, we need to derive the gradient based optimization method. Unlike much simpler gradient computation with respect to sigmoid function, the gradient of some logarithm term may become quite annoying (we provide the mean field VB derivation in the
supplementary material).

Fig. 3.15: Results of latent variable estimation.

Fig. 3.14(a) and (b) display the measurement of accuracy and ROC curve. For usual flu setting when $\alpha, \beta$ are extremely small ($\approx 0$), the performance on three algorithms has no significant difference, no matter on which criterion. However, as mentioned previously, the flaw of Taylor expansion will be exaggeratedly amplified if $\alpha, \beta$ increases. On this circumstance, the advantage of variational inference is exemplified on the other two flu settings. Both the accuracy and ROC illustrate that Gibbs EM almost deteriorates as a random classifier, while mean field and sigmoid variational infection can still obtain reasonable result. Additionally, the behavior of SigVI is smoother than mean-field. This phenomenon is more obvious in yellow circle of Fig.3.14 in the setting with low recovery and high infection rate, thus empirically leading to a reasonable hard threshold for decision. Since mean-field VBEM is similar to full conditional of Gibbs sampling, its curves behave analogously as Gibbs EM. The 1000 thresholds we used to plot are uniformly distributed on interval $[0, 1]$, but
the corresponding points of ROC except SigVI are almost located in the leftmost region of coordinates.

3.3.4 Short Pattern Capture on Real Flu Case

In the second experiment, we validate our approach in the real flu diffusion dataset with professional diagnosis. In previous work, the Gibbs sampling based algorithm tends to find the long duration flu while the short pattern is usually omitted by averaging when computing posterior mean. However, we found the node-wise sigmoid variational inference can eliminate this deficiency due to the structure of parameterization. It favors a node-wise estimation rather than temporal dependency. Fig.3.15 illustrates the comparison between the expertise diagnosis and different inference results. We notice that the short duration after onsite of flu does exist in accordance to Fig.3.15(a). The posterior mean of SigVI in Fig.3.15(b) shows that even the short duration is less than 5 days, such patterns can be captured by assigning high risk on these days, accompanying with less risk of infection appearing before or after these days. The finding actually reflects the common sense. For example, Participant 13 was diagnosed as flu during day 13 to 15. Comparing with other algorithms, only SigVI detected this patient with prediction during the whole week. The first and last 2 days are predicted as low risk period, while the middle 3 days are high risk period which is corresponding to expertise.

<table>
<thead>
<tr>
<th>Model</th>
<th>Recall</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>SigVI</td>
<td><strong>0.9548</strong></td>
<td>0.9946</td>
</tr>
<tr>
<td>MfVI</td>
<td>0.9032</td>
<td><strong>0.9979</strong></td>
</tr>
<tr>
<td>GibEM Fan et al. (2015c)</td>
<td>0.8974</td>
<td>0.9978</td>
</tr>
<tr>
<td>GCHMMs+LogReg Dong et al. (2012)</td>
<td>0.7436</td>
<td>0.9912</td>
</tr>
</tbody>
</table>

For the overall predication accuracy, all algorithms achieve a more than 99%
accuracy. This is mainly due to two reasons. The underlying infection rate should be close to 0, since this dataset is collected during a normal flu season, not from SARS or HxNx outbreaking period. Another obvious reason is that the negative cases (0) dominate the binary matrix, thus meaning that a more than 95% accuracy can be obtained even if we predict all zeros. Therefore, we also prefer to examine recall (equivalently, sensitivity or true positive rate). Table 3.3 shows our results compared with other papers’ report. Mean-field variational inference has the highest accuracy while it is not significantly better than the others. However, SigVI achieved at least 5% improvement on recall than any other algorithm. This is, in turn, consistent with the short pattern capture property, since more positive cases are detected by SigVI.

3.4 Summary

In this chapter, we propose a unified variational inference framework to learn a general flu diffusion model – hGCHMMs. Our VI algorithm is based on minimizing the KL divergence between true posterior of generative model and the proposed recognition model. Differing from standard variations EM, our approach can learn the parameters of both models simultaneously even in a dynamic and heterogenous set-up. In particular, the experimental results imply that our inference method is possible to generalize the application of hGCHMMs to more broader diseases, such as high contagious avian influenza, which proves difficult to model previously. Like deep neural networks, our developed variational inference may suffer the problem of blow-up parameters even regularization is imposed during training. The recent success of deep learning lies on the sufficient training data, which is usually impossible to obtain in the research of health informatics. An important avenue of future research might explore the MCMC method which can likewise overcome the problem of non-conjugacy, especially the efficient Hamiltonian Monte Carlo Neal (2011). In addition, Salimans et al. (2015) bridges the gap between variational inference and MCMC, thus
making it more possible to develop robust inference method.
On the Importance of Generative Adversarial Loss

In previous two chapters, we mainly discuss the power of VAE framework as either a model or an inference method, being able to learn a representative latent feature. However, in the introduction, we also mentioned several disadvantages of VAE over GAN in terms of generation.

- Explicitly probability density model is pre-defined, thus leading to maximize the likelihood estimation.

- Complicated generative model $p(x|z)$ may be problematic (e.g., the RNN, autoregressive decoding distribution can be less dependent on the latent code).

- The result is not asymptotically consistent unless variational distribution is perfect. (e.g., better proposed reference model, such as normalizing flow, inverse autoregressive transformations).

- Generated samples are tend to conservative. (e.g., images are blurred, sentences prefer to contain most frequent words.)

In this chapter, we will illustrate that the generative adversarial loss can improve
the generation results on top of VAE-like models in both natural language processing and image processing. The graphical representation of the VAE+GAN framework is shown in Fig 4. The final desired objective function usually can be written into the following formulation

$$L_G = L_{VAE} + \lambda L_{GAN:G}$$

(4.1)

$$L_D = L_{GAN:D}$$

(4.2)

where $\lambda$ is a tuning parameter controlling the trade-off between the equivalent MLE loss and generative loss in GAN, and the $L_{GAN}$ can refer to the definition in introduction.

4.1 Text GAN

4.1.1 Model Description

In this section, we propose a framework for generating realistic text via adversarial training, and employ a long short-term memory network as generator, and a convolutional network as discriminator. Instead of using the standard objective of GAN, we propose matching the high-dimensional latent feature distributions of real and synthetic sentences, via a kernelized discrepancy metric. This eases adversarial training by alleviating the mode-collapsing problem.

Notice that we use $s$ instead of $x$ to represent sentence as more meaningful notation. Given a sentence corpus $S = p_{data}$, instead of directly optimizing the
objective from standard GAN, we adopt an approach that is similar to the feature matching scheme of Salimans et al. (2016). Specifically, we consider the objective

\begin{align}
L_D &= L_{GAN:D} - \lambda_r L_{recon} + \lambda_m L_{MMD^2} \quad (4.3) \\
L_G &= L_{MMD^2} \quad (4.4) \\
L_{GAN} &= \mathbb{E}_{s \sim p_{data}} \log D(s) + \mathbb{E}_{p(z)} \log[1 - D(G(z))] \\
L_{recon} &= ||\hat{z} - z||^2,
\end{align}

where \(L_D\) and \(L_G\) are iteratively maximized w.r.t \(D(\cdot)\) and minimized w.r.t. \(G(\cdot)\), respectively. \(L_{recon}\) is the Euclidean distance between the reconstructed latent code, \(\hat{z}\), and the original code, \(z\), drawn from prior distribution \(p(z)\). We denote the synthetic sentences as \(\tilde{s} \triangleq G(z)\), where \(z \sim p(z)\). \(L_{MMD^2}\) represents the Maximum Mean Discrepancy (MMD) Gretton et al. (2012) between the empirical distribution of sentence embeddings \(\tilde{f}\) and \(f\), for synthetic and real data, respectively. The model framework is illustrated in Fig 4.2. The latent codes \(z\) are fed through a generator \(G(\cdot)\), to produce synthetic sentence \(\tilde{s}\). Synthetic and real sentences (\(\tilde{s}\) and \(s\)) are fed into a binary discriminator \(D(\cdot)\), for real vs. fake (synthetic) prediction, and also for latent code reconstruction \(\hat{z}\). \(\tilde{f}\) and \(f\) represent features of \(\tilde{s}\) and \(s\), respectively.

We first consider \(L_G\) in (4.4). The generator \(G(\cdot)\) attempts to adjust itself to produce synthetic sentence \(\tilde{s}\), with features \(\tilde{f}\), encoded by \(D(\cdot)\), to mimic the real sentence features \(f\) (also encoded by \(D(\cdot)\)). This is achieved by matching the empirical distributions of \(\tilde{f}\) and \(f\) via the MMD objective.

76
Concisely, MMD measures the mean squared difference between two sets of samples $\mathcal{X}$ and $\mathcal{Y}$, where $\mathcal{X} = \{x_i\}_{i=1:N_x}$, $x_i \in \mathbb{R}^d$, $\mathcal{Y} = \{y_i\}_{i=1:N_y}$, $y_i \in \mathbb{R}^d$, $d$ is the dimensionality of the samples, and $N_x$ and $N_y$ are sample sizes for $\mathcal{X}$ and $\mathcal{Y}$, respectively.

The MMD metric characterizes the differences between $\mathcal{X}$ and $\mathcal{Y}$ over a Reproducing Kernel Hilbert Space (RKHS), $\mathcal{H}$, associated with kernel function $k(\cdot) : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$. The kernel can be written as an inner product over $\mathcal{H}$:

$$k(\cdot) = \langle k(\cdot), k(\cdot, \cdot) \rangle_{\mathcal{H}}$$

and $\phi(x) \triangleq k(x, \cdot) \in \mathcal{H}$ is denoted as the feature mapping Gretton et al. (2012). Formally, the MMD for two empirical distributions $\mathcal{X}$ and $\mathcal{Y}$ is given by

$$L_{MMD}^2 = \|E_{x \sim \mathcal{X}} \phi(x) - E_{y \sim \mathcal{Y}} \phi(y)\|^2_{\mathcal{H}}$$

$$= E_{x \sim \mathcal{X}} E_{x' \sim \mathcal{X}} [k(x, x')] + E_{y \sim \mathcal{Y}} E_{y' \sim \mathcal{Y}} [k(y, y')] - 2E_{x \sim \mathcal{X}} E_{y \sim \mathcal{Y}} [k(x, y)].$$

Note that $L_{MMD}^2$ reaches its minimum when the two empirical distributions $\mathcal{X}$ and $\mathcal{Y}$ (in general) match exactly. For example, with a polynomial kernel, minimizing $L_{MMD}^2$ can be understood as matching moments of two empirical distributions up to per-defined order of polynomial. With a universal kernel like the Gaussian kernel, with bandwidth $\sigma$, minimizing the MMD objective will match moments of all orders Gretton et al. (2012). Here, we use MMD to match the empirical distribution of $\tilde{f}$ and $f$ using a Gaussian kernel.

The adversarial discriminator $D(\cdot)$ associated with the loss in (4.3) aims to produce sentence features that are most discriminative, representative and challenging. These aims are explicitly represented as the three components of (4.3), namely, (i) $L_{GAN}$ requires $\tilde{f}$ and $f$ to be discriminative of real and synthesized sentences; (ii) $L_{recon}$ requires $\tilde{f}$ and $f$ to preserve maximum reconstruction information for the latent code $z$ that generates synthetic sentences; and (iii) $L_{MMD}^2$ forces $D(\cdot)$ to select the most challenging features for the generator to match.

In the situation for which simple features are enough for the discrimination/reconstruction task, this additional loss seeks to estimate complex features that are difficult for the
current generator, thus improving in terms of generation ability. In our experience, we find the reconstruction and MMD loss in D serve as regularizer to the binary classification loss, in that by adding these losses, discriminator features tend to be more spread-out in the feature space.

In summary, the adversarial game associated with (4.3) and (4.4) is the following: \( D(\cdot) \) attempts to select informative sentence features, while \( G(\cdot) \) aims to match these features. Parameters \( \lambda_r \) and \( \lambda_m \) act as trade-off between discrimination ability, and reconstruction and moment matching precision, respectively. We argue that this framework has several advantages over the standard GAN objective.

The original GAN objective has been shown to be prone to mode collapsing, especially when the so-called log \( D \) alternative for the generator loss is used Metz et al. (2016), i.e., replacing the second term of standard GAN objective by \( -E_{p(z)} \log[D(G(z))] \).

This is because when log \( D \) is used, fake-looking samples are penalized more severely than less diverse samples Arjovsky and Bottou (2017), thus grossly underestimating the variance of latent features. The loss in (4.4), on the other hand, forces the generator to produce highly diverse sentences to match the variation of real sentences, by latent moment matching, thus alleviating the mode-collapsing problem. We believe that leveraging MMD is general enough to be useful as a framework in other data domains, e.g., images. Presumably, the discrete nature of text data makes standard GAN prone to mode-collapsing. This is manifested by close neighbors in latent code space producing the same text output. In our approach, MMD and feature matching are introduced to alleviate mode collapsing with text data as motivating domain. However, whether such an objective is free from the convergence issues of the standard GAN, due to vanishing gradient from the generator, is known to be problem specific Arjovsky and Bottou (2017).

Arjovsky and Bottou (2017) demonstrated that JSD yields weak gradient signals when the real and synthetic data are far apart. To deliver stable gradients, a
smoother distance metric over the data domain is required. In (4.5), we are essentially employing a Neural Network (NN) embedding via Gaussian kernel for matching \( s \) and \( \tilde{s} \), i.e., \( k_s(s,s') = \phi(g(s))^\top \phi(g(s')) \), where \( g(\cdot) \) denotes the NN embedding that maps from the data to the feature domain. Under the assumption that \( g(\cdot) \) is a bijective mapping, i.e., distinct sentences have different embedded feature vectors, it is proven that if the original kernel function is universal, the composed kernel is also universal. As shown in Gretton et al. (2012), the MMD is a proper metric when the kernel is universal. In fact, if the kernel function is universal, the MMD metric will be no worse than TVD in terms of vanishing gradients Arjovsky et al. (2017). However, if the bandwidth of the kernel is too small, much smaller than the average distance between data points, the vanishing gradient problem remains Arjovsky et al. (2017).

Additionally, seeking to match the sentence features provides a more achievable and informative objective than directly trying to mislead the discriminator as in standard GAN. Specifically, the loss in (4.4) implies a clearer aim for the generator, as it requires matching the latent features (distribution-wise) as opposed to uniquely trying to fake a binary classifier.

Note that if the latent features from real and synthetic data have similar distributions it is unlikely that the discriminator, that uses these features as inputs, will be able to tell them apart. Implementation-wise, the updating signal from the generator does not need to propagate all the way back from the discriminator, but rather directly from the features layer, thus less prone to fading. We believe there may be other possible approaches for text generation using GAN, however, we hope to provide a first attempt toward overcoming some of the difficulties associated with it.
4.1.2 Matching Methods

One limitation of the proposed approach is that the dimensionality of features $\tilde{f}$ and $f$ could be much larger than the size of the subset of data (minibatch) used during learning, hence the empirical distribution may not be sufficiently representative. In fact, a reliable Gaussian kernel MMD two-sample test generally requires the size of the minibatch to be proportional to the number of dimensions Ramdas et al. (2014). To alleviate this issue, we consider two strategies.

**Compressing network** We map $\tilde{f}$ and $f$ into a lower-dimensional feature space using a *compressing network* with fully connected layers, also learned by $D(\cdot)$. This is sensible because the discriminator will still encourage the most challenging features to be abstracted (compressed) from the original features $\tilde{f}$ and $f$. This approach provides significant computational savings, as computation of the MMD in (4.5) scales with $O(d^2d_f)$, where $d_f$ denotes the dimensionality of the feature vector. However, a lower-dimensional mapping may miss valuable information. Besides, finding the optimal mapping dimension may be difficult in practice. There exists a tradeoff between fast estimation and a richer feature vector, by setting $d_f$ appropriately.

**Gaussian covariance matching** We could also avoid using the *kernel trick*, as was used in (4.5). Instead, we can replace $\mathcal{L}_{MMD}^2$ by $\mathcal{L}_G^{(c)}$ (below), where we accumulate (Gaussian) sufficient statistics from multiple minibatches, thus alleviating the inadequate-minibatch-size issue. Specifically,

$$\mathcal{L}_G^{(c)} = \text{Tr}(\tilde{\Sigma}^{-1}\Sigma + \Sigma^{-1}\tilde{\Sigma})$$

$$+ (\tilde{\mu} - \mu)^T(\tilde{\Sigma}^{-1} + \Sigma^{-1})(\tilde{\mu} - \mu),$$

(4.6)

where $\tilde{\Sigma}$ and $\Sigma$ represent the covariance matrices of synthetic and real sentence feature vectors $\tilde{f}$ and $f$, respectively. $\tilde{\mu}$ and $\mu$ denote the mean vectors of $\tilde{f}$ and $f$.
, respectively. By setting \( \tilde{\Sigma} = \Sigma = I \), (4.6) reduces to the first-moment feature matching technique from Salimans et al. (2016). Note that this loss \( L_{\mathcal{G}}^{(c)} \) is an upper bound of the JSD (omitting constant, proved in the Supplementary Material) between two multivariate Gaussian distribution \( \mathcal{N}(\mu, \Sigma) \) and \( \mathcal{N}(\tilde{\mu}, \tilde{\Sigma}) \), which is more tractable than directly minimizing JSD. The feature vectors used in (4.6) are the neural net outputs before applying any non-linear activation function. We note that the Gaussian assumption may still be strong in many cases. In practice, we use a moving average of the most recent \( m \) minibatches for estimating all sufficient statistics \( \tilde{\Sigma}, \Sigma, \tilde{\mu} \) and \( \mu \). Further, \( \tilde{\Sigma} \) and \( \Sigma \) are initialized to be \( I \) to prevent numerical problems.

### 4.1.3 Experimental Evaluation

We use a CNN discriminator and an LSTM generator, however, we initialize the LSTM parameters of the generator by pre-training a standard CNN-LSTM autoencoder. For the discriminator/encoder initialization, we use a permutation training strategy. For each sentence in the corpus, we randomly swap two words to construct a slightly tweaked sentence counterpart. The discriminator is pre-trained to distinguish the tweaked sentences from the true sentences. We empirically found this provides a better initialization (compared to no pre-training) for the discriminator to learn good features. We also utilized other training techniques to stabilize training, such as soft-labeling Salimans et al. (2016).

#### Data and Experimental Setup

Our model is trained using a combination of two datasets: (i) the BookCorpus dataset Zhu et al. (2015), which consists of 70 million sentences from over 7000 books; and (ii) the ArXiv dataset, which consists of 5 million sentences from abstracts of papers from various subjects, obtained from the arXiv website. The motivation for merging two different corpora is to investigate whether
the model can generate sentences that integrate both scientific and informal writing styles. We randomly choose 0.5 million sentences from BookCorpus and 0.5 million sentences from arXiv to construct training and validation sets, i.e., 1 million sentences for each. For testing, we randomly select 25,000 sentences from both corpus, for a total of 50,000 sentences.

We train the generator and discriminator/encoder iteratively. Provided that the LSTM generator typically involves more parameters and is more difficult to train than the CNN discriminator, we perform one optimization step for the discriminator for every $K = 5$ steps of the generator. We use a mixture of 5 isotropic Gaussian (RBF) kernels with different bandwidths $\sigma$ as in Li et al. (2015). Bandwidth parameters are selected to be close to the median distance (in our case around 20) of feature vectors encoded from real sentences. $\lambda_r$ and $\lambda_m$ are selected based on the performance on the validation set. The validation performance is evaluated by loss of generator and corpus-level BLEU score Papineni et al. (2002), described below.

For the CNN discriminator/encoder, we use filter windows ($h$) of sizes \{3,4,5\} with 300 feature maps each, hence each sentence is represented as a 900-dimensional vector. The dimensionality of $z$ and $\hat{z}$ is also 900. The feature vector is then fed into a 900-200-2 fully connected network for the discriminator and 900-900-900 for encoder, with sigmoid activation units connecting the intermediate layers and softmax/tanh units for the top layer of discriminator/encoder. We did not observe performance changes by adding dropout. For the LSTM sentence generator, we use one hidden layer of 500 units.

Gradients are clipped if the norm of the parameter vector exceeds 5 Sutskever et al. (2014). Adam Kingma and Ba (2014) with learning rate $5 \times 10^{-5}$ for both discriminator and generator is utilized for optimization. The size of the minibatch is set to 256.

Both the generator and the discriminator are pre-trained using the strategies
Table 4.1: Quantitative results using BLEU-2,3,4 and KDE.

<table>
<thead>
<tr>
<th>Method</th>
<th>BLEU-4</th>
<th>BLEU-3</th>
<th>BLEU-2</th>
<th>KDE(nats)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AE</td>
<td>0.01±0.01</td>
<td>0.11±0.02</td>
<td>0.39±0.02</td>
<td>2727±42</td>
</tr>
<tr>
<td>VAE</td>
<td>0.02±0.02</td>
<td>0.16±0.03</td>
<td>0.54±0.03</td>
<td>1892±25</td>
</tr>
<tr>
<td>seqGAN</td>
<td>0.04±0.04</td>
<td>0.30±0.08</td>
<td>0.67±0.04</td>
<td>2019±53</td>
</tr>
<tr>
<td>textGAN(MM)</td>
<td>0.09±0.04</td>
<td>0.42±0.04</td>
<td>0.77±0.03</td>
<td>1823±50</td>
</tr>
<tr>
<td>textGAN(CM)</td>
<td>0.12±0.03</td>
<td>0.49±0.06</td>
<td>0.84±0.02</td>
<td>1686±41</td>
</tr>
<tr>
<td>textGAN(MMD)</td>
<td>0.13±0.05</td>
<td>0.49±0.06</td>
<td>0.83±0.04</td>
<td>1688±38</td>
</tr>
<tr>
<td>textGAN(MMD-L)</td>
<td>0.11±0.05</td>
<td><strong>0.52±0.07</strong></td>
<td><strong>0.85±0.04</strong></td>
<td><strong>1684±44</strong></td>
</tr>
</tbody>
</table>

described in Section ???. We also employed a **warm-up** training during the first two epochs, as we found it improves convergence during the initial stage of learning. Specifically, we use a mean-matching objective for the generator loss, i.e., $||E_f - E\tilde{f}||^2$, as in Salimans et al. (2016). Further details of the experimental design are provided in the the Supplementary Material. All experiments are implemented in Theano Bastien et al. (2012), using one NVIDIA GeForce GTX TITAN X GPU with 12GB memory. The model was trained for 50 epochs in roughly 3 days.

For BLEU score evaluation, we follow the strategy in ?? of using the entire test set as the reference. For KDE evaluation, the lengths of the generated sentences are different, thus we first embed all the sentences to a 900-dimensional vector. Since no standard sentence encoder is available, we use the encoder learned from AE. The covariance matrix for the Parzen kernel in KDE is set to be the covariance of feature vectors for real tested sentences. Despite the fact that the KDE approach, as a log-likelihood estimator tends to have high variance Theis et al. (2015), the KDE score tracks well with our BLEU score evaluation. The results are shown in Table 4.1. MMD and MMD-L generally score higher in sentences quality. MMD-L seems better at capturing 2-grams (BLEU-2), while MMD outperforms MMD-L in 4-grams (BLEU-4). We also observed that when using CM, the generated sentences tend to be shorter than MMD (not shown).
Table 4.2: Sentences generated by textGAN.

| a | we show the joint likelihood estimator (in a large number of estimating variables embedded on the subspace learning). |
| b | this problem achieves less interesting choices of convergence guarantees on turing machine learning. |
| c | in hidden markov relational spaces, the random walk feature decomposition is unique generalized parametric mappings. |
| d | i see those primitives specifying a deterministic probabilistic machine learning algorithm. |
| e | i wanted in alone in a gene expression dataset which do n’t form phantom action values. |
| f | as opposite to a set of fuzzy modelling algorithm, pruning is performed using a template representing network structures. |

Generated sentences  Table 4.2 shows six sentences generated by textGAN. Note that the generated sentences seem to be able to produce novel phrases by imagining concept combinations, e.g., in Table 4.2(b,c,f), or to borrow words from a different corpus to compose novel sentences, e.g., in Table 4.2(d,e). In many cases, it learns to automatically match the parentheses and quotation marks, e.g., in Table 4.2(a), and can synthesize relatively long sentences, e.g., in 4.2(a,f). In general, the synthetic sentences seem syntactically reasonable. However, the semantic meaning is less well preserved especially in sentence of more than 20 words, e.g., in Table 4.2(e,f).

We observe that the discriminator can still sufficiently distinguish the synthetic sentences from the real ones (the probability to predict synthetic data as real is around 0.05), even when the synthetic sentences seems to perserve reasonable grammatical structure and use proper wording. It is likely that the CNN is able to accurately characterize the semantic meaning and differentiate sentences, while the generator may get trapped into a local optimum, where any slight modification would result in a higher loss (4.4) for the generator. Presumably, long-range distance features are not difficult to abstract by the discriminator/encoder, however, is less likely to be imitated by the generator. One promising direction is to leverage reinforce-
ment learning strategies as in ?, where the updating for LSTM can be more effectively steered. Nevertheless, investigation on how to improve the long-range behavior is left as interesting future work.

Latent feature space trajectories Following Bowman et al. (2015), we further empirically evaluate whether the latent variable space can “densely” encode sentences. We visualize the transition from one sentence to another by constructing a linear path between two randomly selected points in latent feature space, to then generate the intermediate sentences along the linear trajectory. For comparison, a baseline autoencoder (AE) is trained for 20 epochs. The results for textGAN and AE are presented in Table 4.3, illustrating the intermediate sentences produced from linear transition between two points (A and B) in the latent feature space, where each sentence is generated from a latent point on a linear path. Compared to AE, the sentences produced by textGAN are generally more syntactically and semantically reasonable. The transition suggest “smoothness” and interpretability, however, the wording choices and sentence structure showed dramatic changes in some regions in the latent feature space. This seems to indicate that local “transition smoothness” varies from region to region.

4.2 Image Enhancement via GAN Loss

Besides the text domain, we will in this section demonstrate GAN loss will help improve the quality of image generation. We first test the importance of GAN loss on a very simple model, Unets Ronneberger et al. (2015), which is used for image segmentation task. Fig 4.2 shows an image-to-image translation example from simple line drawing to color faces. In the case without GAN loss, Unets is basically a conditional generation model or fully supervised model, where the input is the simple line drawing derived from edge detection technique, and the output is desired color
images. When GAN loss is imposed, the graphical representation is almost identical to Fig 4, but the structure of auto-encoders is adapted to the Unets (Fig 4.2).

Moreover, we found that GAN loss can potentially help a variety of tasks in image processing, such as super-resolution, motion deblurring and colorization. We first propose a learning-to-learn framework for ADMM with the auxiliary deep neural networks, resulting in an inner-loop free ADMM algorithm. In addition, we build a discriminator and a comparator on top of the learning-to-learn ADMM framework, to incorporate both GAN loss and feature matching loss (which has been proposed as an improved GAN loss).

<table>
<thead>
<tr>
<th>textGAN</th>
<th>AE</th>
</tr>
</thead>
<tbody>
<tr>
<td>our methods apply novel approaches to solve modeling tasks.</td>
<td>our methods apply to train UNK models involving complex.</td>
</tr>
<tr>
<td>our methods apply two different approaches to solve computing.</td>
<td>our methods solve use to train ) .</td>
</tr>
<tr>
<td>our methods achieve some different approaches to solve computing.</td>
<td>our approach show UNK to models exist.</td>
</tr>
<tr>
<td>our methods achieve the best expert structure detection.</td>
<td>that supervised algorithms show to UNK speed.</td>
</tr>
<tr>
<td>the methods have been different related tasks.</td>
<td>that address algorithms to handle ).</td>
</tr>
<tr>
<td>the guy is the minimum of UNK.</td>
<td>that address versions to be used in .</td>
</tr>
<tr>
<td>the guy is n’t easy tonight.</td>
<td>i believe the means of this attempt to cope .</td>
</tr>
<tr>
<td>i believe the guy is n’t smart okay?</td>
<td>i believe it ’s we be used to get .</td>
</tr>
<tr>
<td>i believe the guy is n’t smart .</td>
<td>i believe it i ’m a way to belong .</td>
</tr>
</tbody>
</table>

B i believe i ’m going to get out.
4.2.1 ADMM

**Notation:** trainable networks by calligraphic font, e.g., $A$, fixed networks by italic font e.g., $A$. As mentioned in the last section, the low dimensional measurement is denoted as $y \in \mathbb{R}^m$, which is reduced from high dimensional ground truth $x \in \mathbb{R}^n$ by a linear operator $A$ such that $y = Ax$. Note that usually $n \geq m$, which makes the number of parameters to estimate no smaller than the number of data points in hand. This imposes an ill-posed problem for finding solution $x$ on new observation $y$, since $A$ is an underdetermined measurement matrix. For example, in a super-resolution set-up, the matrix $A$ might not be invertible, such as the strided Gaussian convolution in Shi et al. (2016); Sønderby et al. (2016). To overcome this difficulty, several computational strategies, including Markov chain Monte Carlo (MCMC) and tailored variable splitting under the ADMM framework, have been proposed and
applied to different kinds of priors, e.g., the empirical Gaussian prior Wei et al. (2015a); Zhao et al. (2016), the Total Variation prior Simoes et al. (2015); Wei et al. (2015b, 2016), etc. In this paper, we focus on the popular ADMM framework due to its low computational complexity and recent success in solving large scale optimization problems. More specifically, the optimization problem is formulated as

Thus, it is principal to consider the approach as regularized signal recovery problems in the form of linear inverse solution by optimization technique alternating direction method of multipliers method (ADMM).

\[ \hat{x} = \arg \min_{x,z} \|y - Az\|^2 + \lambda R(x), \quad s.t. \quad z = x \] (4.7)

where the introduced auxiliary variable \(z\) is constrained to be equal to \(x\), and \(R(x)\) captures the structure promoted by the prior/regularization. If we design the regularization in an empirical Bayesian way, by imposing an implicit data dependent prior on \(x\), i.e., \(R(x; y)\) for amortized inference Sønderby et al. (2016), the augmented Lagrangian for (4.7) is

\[ \mathcal{L}(x, z, u) = \|y - Az\|^2 + \lambda R(x; y) + \langle u, x - z \rangle + \beta \|x - z\|^2 \] (4.8)

where \(u\) is the Lagrange multiplier, and \(\beta > 0\) is the penalty parameter. The usual augmented Lagrange multiplier method is to minimize \(\mathcal{L}\) w.r.t. \(x\) and \(z\) simultaneously. This is difficult and does not exploit the fact that the objective function is separable. To remedy this issue, ADMM decomposes the minimization into two subproblems that are minimizations w.r.t. \(x\) and \(z\), respectively. More specifically, the iterations are as follows:

\[ x^{k+1} = \arg \min_x \beta \|x - z^k + u^k/2\beta\|^2 + \lambda R(x; y) \] (4.9)

\[ z^{k+1} = \arg \min_z \|y - Az\|^2 + \beta \|x^{k+1} - z + u^k/2\beta\|^2 \] (4.10)

\[ u^{k+1} = u^k + 2\beta(x^{k+1} - z^{k+1}). \] (4.11)
If the prior $\mathcal{R}$ is appropriately chosen, such as $\|x\|_1$, a closed-form solution for (4.9), i.e., a soft thresholding solution is naturally desirable. However, for some more complicated regularizations, e.g., a patch based prior Elad and Aharon (2006), solving (4.9) is nontrivial, and may require iterative methods. To solve (4.10), a matrix inversion is necessary, for which conjugate gradient descent (CG) is usually applied to update $z$ Chang et al. (2017). Additionally, in many cases there are no closed form solutions, more auxiliary variables are essentially required. Even worse, there is no guarantee of convergence for the algorithm with more than two variables ?. Thus, solving (4.9) and (4.10) is in general cumbersome. Inner loops are required to solve these two sub-minimization problems due to the intractability of the prior and the inversion, resulting in large computational complexity. Most of computational complexity comes from the proximity operator due to its intractability and the matrix inversion which is not easy diagonalized. To avoid such drawbacks or limitations, we propose an inner loop-free update rule with two pre-trained deep convolutional architectures.

4.3 Inner-loop free ADMM

4.3.1 Amortized inference for $x$ using a conditional proximity operator

Solving sub-problem (4.9) is equivalent to finding the solution of the proximity operator

$$
P_{\mathcal{R}}(v; y) = \arg \min_x \frac{1}{2} \|x - v\|^2 + \mathcal{R}(x; y)$$

(4.12)

where we incorporate the constant $\frac{\lambda}{2}$ into $\mathcal{R}$ without loss of generality. If we impose the first order necessary conditions Maurer and Zowe (1979), we have

$$
x = P_{\mathcal{R}}(v; y) \iff 0 \in \partial \mathcal{R}(\cdot; y)(x) + x - v \iff v - x \in \partial \mathcal{R}(\cdot; y)(x)$$

(4.13)

where $\partial \mathcal{R}(\cdot; y)$ is a partial derivative operator. For notational simplicity, we define another operator $\mathcal{F} = I + \partial \mathcal{R}(\cdot; y)$. Thus, the last condition in (4.13) indicates that
\( x^{k+1} = F^{-1}(v) \). Note that the inverse here represents the inverse of an operator, i.e., the inverse function of \( F \). Thus our objective is to learn such an inverse operator which projects \( v \) into the prior subspace. For simple priors like \( \| \cdot \|_1 \) or \( \| \cdot \|_2^2 \), the projection can be efficiently computed. In this work, we propose an implicit example-based prior, which does not have a truly Bayesian interpretation, but aids in model optimization. In line with this prior, we define the implicit proximity operator \( G_\theta(x; v, y) \) parameterized by \( \theta \) to approximate unknown \( F^{-1} \). More specifically, we propose a neural network architecture referred to as conditional Pixel Shuffling Denoising Auto-Encoders (cPSDAE) as the operator \( G \), where pixel shuffling Shi et al. (2016) means periodically reordering the pixels in each channel mapping a high resolution image to a low resolution image with scale \( r \) and increase the number of channels to \( r^2 \) (see Shi et al. (2016) for more details). This allows us to transform \( v \) so that it is the same scale as \( y \), and concatenate it with \( y \) as the input of cPSDAE easily. The architecture of cPSDAE is shown in Figure 4.5 (d).

4.3.2 Inversion-free update of \( z \)

While it is straightforward to write down the closed-form solution for sub-problem (4.10) w.r.t. \( z \) as is shown in (4.14), explicitly computing this solution is nontrivial.

\[
    z^{k+1} = K \left( A^T y + \beta x^{k+1} + u^k / 2 \right), \quad \text{where} \quad K = (A^T A + \beta I)^{-1} \tag{4.14}
\]

In (4.14), \( A^T \) is the transpose of the matrix \( A \). As we mentioned, the term \( K \) in the right hand side involves an expensive matrix inversion with computational complexity \( O(n^3) \). Under some specific assumptions, e.g., \( A \) is a circulant matrix, this matrix inversion can be accelerated with a Fast Fourier transformation, which has a complexity of order \( O(n \log n) \). Usually, the gradient based update has linear complexity in each iteration and thus has an overall complexity of order \( O(n_{\text{int}} \log n) \), where \( n_{\text{int}} \) is the number of iterations. In this work, we will learn this matrix inversion explicitly by designing a neural network. Note that \( K \) is only dependent on \( A \), and
thus can be computed in advance for future use. This problem can be reduced to a smaller scale matrix inversion by applying the Sherman-Morrison-Woodbury formula:

\[ K = \beta^{-1} (I - A^T B A), \text{ where } B = (\beta I + A A^T)^{-1}. \]  

(4.15)

Therefore, we only need to solve the matrix inversion in dimension \( m \times m \), i.e., estimating \( B \). We propose an approach to approximate it by a trainable deep convolutional neural network \( C_\phi \approx B \) parameterized by \( \phi \). Note that \( B^{-1} = \lambda I + A A^T \) can be considered as a two-layer fully-connected or convolutional network as well, but with a fixed kernel. This inspires us to design two auto-encoders with shared weights, and minimize the sum of two reconstruction losses to learn the inversion \( C_\phi \) :

\[
\arg \min_{\phi} \mathbb{E}_\varepsilon \left[ \| \varepsilon - C_\phi B^{-1} \varepsilon \|^2_2 + \| \varepsilon - B^{-1} C_\phi \varepsilon \|^2_2 \right] \tag{4.16}
\]

where \( \varepsilon \) is sampled from a standard Gaussian distribution. The loss in (4.16) is clearly depicted in Figure 4.5 (a) with the structure of \( B^{-1} \) in Figure 4.5 (b) and the structure of \( C_\phi \) in Figure 4.5 (c). Since the matrix \( B \) is symmetric, we can reparameterize \( C_\phi \) as \( W_\phi W_\phi^T \), where \( W_\phi \) represents a multi-layer convolutional network and \( W_\phi^T \) is a symmetric convolution transpose architecture using shared kernels with \( W_\phi \), as shown in Figure 4.5 (c) (the blocks with the same colors share the same network parameters). This loss can be optimized using stochastic gradient descent.

By plugging the learned \( C_\phi \) in (4.15) , we obtain a reusable deep neural network \( K_\phi = \beta^{-1} (I - A^T C_\phi A) \) as a surrogate for the exact inverse matrix \( K \). The update of \( z \) at each iteration can be done by applying the same \( K_\phi \) as follows:

\[
z^{k+1} \leftarrow \beta^{-1} (I - A^T C_\phi A) \left( A^T y + \beta x^{k+1} + u^k / 2 \right). \tag{4.17}
\]
4.3.3 Adversarial training of cPSDAE

In this section, we will describe the proposed adversarial training scheme for cPSDAE to update $x$. Suppose that we have the paired training dataset $(x_i, y_i)_{i=1}^N$, a single cPSDAE with the input pair $(\tilde{x}, y)$ is trying to minimize the reconstruction error $L_r(\mathcal{G}_{\theta}(\tilde{x}, y), x)$, where $\tilde{x}$ is a corrupted version of $x$, i.e., $\tilde{x} = x + n$ where $n$ is random noise. Notice $L_r$ in traditional DAE is commonly defined as $\ell_2$ loss, however, $\ell_1$ loss is an alternative in practice. Additionally, we follow the idea in Nguyen et al. (2016); Dosovitskiy and Brox (2016) by introducing a discriminator and a comparator to help train the cPSDAE, and find that it can produce sharper or higher quality images than merely optimizing $\mathcal{G}$. This will wrap our conditional generative model $\mathcal{G}_{\theta}$ into the conditional GAN Goodfellow et al. (2014) framework with an extra feature matching network (comparator). Recent advances in representation
learning problems have shown that the features extracted from well-pre-trained neural networks on supervised classification problems can be successfully transferred to other tasks, such as zero-shot learning Lei Ba et al. (2015), style transfer learning Gatys et al. (2016). Thus, we can simply use pre-trained AlexNet Krizhevsky et al. (2012) or VGG-16 Model Simonyan and Zisserman (2014) on ImageNet as the comparator without fine-tuning in order to extract features that capture complex and perceptually important properties. The feature matching loss \( \mathcal{L}_{fp} \) is usually the \( \ell_2 \) distance of high level image features, where \( C \) represents the pre-trained network. Since \( C \) is fixed, the gradient of this loss can be back-propagated to \( \theta \).

For the adversarial training, the discriminator \( \mathcal{D}_\psi \) is a trainable convolutional network. We can keep the standard discriminator loss as in a traditional GAN, and add the generator loss of the GAN to the previously defined DAE loss and comparator loss. Thus, we can write down our two objectives,

\[
\mathcal{L}_D(x, y) = -\log \mathcal{D}_\psi(x) - \log (1 - \mathcal{D}_\psi(G_\theta(\tilde{x}, y)))
\]

\[
\mathcal{L}_G(x, y) = \lambda_r \|G_\theta(\tilde{x}, y) - x\|_2^2 + \lambda_f \|C(G_\theta(\tilde{x}, y)) - C(x)\|_2^2 - \lambda_a \log \mathcal{D}_\psi(G_\theta(\tilde{x}, y))
\]

The optimization involves iteratively updating \( \psi \) by minimizing \( \mathcal{L}_D \) keeping \( \theta \) fixed, and then updating \( \theta \) by minimizing \( \mathcal{L}_G \) keeping \( \psi \) fixed. Notice that \( \log \frac{\mathcal{D}_\psi(G_\theta(\tilde{x}, y))}{1 - \mathcal{D}_\psi(G_\theta(\tilde{x}, y))} \) is a practical and theoretical alternative in \( \mathcal{L}_G \). The proposed method, including training and inference has been summarized in Algorithm 7. Note that each update of \( x \) or \( z \) using neural networks in an ADMM iteration has a complexity of linear order w.r.t. the data dimensionality \( n \).

In summary, the Figure 4.5 shows the network architecture of our algorithm, where the network for updating \( z \) (in black) has three components: (a) loss function (4.16), (b) structure of \( B^{-1} \), (c) structure of \( C_\phi \). Additionally, the input \( \epsilon \) is random noise independent from the training data. Thus, the network for updating \( z \) (in blue)
is represented as (d) structure of cPSDAE $G_{\theta}(x; \tilde{x}, y)$ ($\tilde{x}$ plays the same role as $v$ in training). (e) shows the adversarial training for $R(x; y)$. Note again that (a)(b)(c) describes the network for inferring $z$, which is data-independent and (d)(e) describes the network for inferring $x$, which is data-dependent.

**Algorithm 7** Inner-loop free ADMM with Auxiliary Deep Neural Nets (Inf-ADMM-ADNN)

**Training stage:**
1. Train net $K_{\phi}$ for inverting $A^T A + \beta I$
2. Train net cPSDAE for proximity operator of $R(x; y)$

**Testing stage:**
1. for $t = 1, 2, \ldots$ do
2. Update $x$ cf. $x^{k+1} = F^{-1}(v)$;
3. Update $z$ cf. (4.17);
4. Update $u$ cf. (4.11);
5. end for

4.3.4 Image super-resolution and motion deblurring

In this section, we apply the proposed Inf-ADMM-ADNN to solve the popular image super-resolution problem. We have tested our algorithm on the MNIST dataset LeCun et al. (1998) and the 11K images of the Caltech-UCSD Birds-200-2011 (CUB-200-2011) dataset Wah et al. (2011). In the first two rows of Figure 4.6, high resolution images, as shown in the last column, have been blurred (convolved) using a Gaussian kernel of size $3 \times 3$ and downsampled every 4 pixels in both vertical and horizontal directions to generate the corresponding low resolution images as shown in the first column. The bicubic interpolation of LR images and results using proposed Inf-ADMM-ADNN on a 20% held-out test set are displayed in column 2 and 3. Visually, the proposed Inf-ADMM-ADNN gives much better results than the bicubic interpolation, recovering more details including colors and edges. A similar task to super-resolution is motion deblurring, in which the convolution kernel is a directional kernel and there is no downsampling. The motion deblurring results using
Inf-ADMM-ADNN are displayed in the bottom of Figure 4.6 and are compared with the Wiener filtered deblurring result (the performance of Wiener filter has been tuned to the best by adjusting the regularization parameter). Obviously, the Inf-ADMM-ADNN gives visually much better results than the Wiener filter. In summary, the top row of Figure 4.6 includes (column 1) LR images, (column 2) bicubic interpolation ($\times 4$), (column 3) results using proposed method ($\times 4$), and (column 4) HR image. The middle row includes the same as the top row except for last column which is the SRGAN result. The bottom row includes (column 1) motion blurred images, (column 2) results using Wiener filter with the best performance by tuning regularization parameter, (column 3) results using proposed method, and (column 4) the ground-truth.
4.3.5 Joint super-resolution and colorization

While image super-resolution tries to enhance spatial resolution from spatially degraded images, a related application in the spectral domain exists, i.e., enhancing spectral resolution from a spectrally degraded image. One interesting example is the so-called automatic colorization, i.e., hallucinating a plausible color version of a colorless photograph. To the best knowledge of the authors, this is the first time we can enhance both spectral and spatial resolutions from one single band image. In this section, we have tested the ability to perform joint super-resolution and colorization from one single colorless LR image on the celebA-dataset Liu et al. (2015). The LR colorless image, its bicubic interpolation and $\times 2$ HR image are displayed in the top row of Figure 4.7. The ADMM updates in the 1st, 4th and 7th iterations (on held-out test set) are displayed in the bottom row, showing that the updated image evolves towards higher quality from the bottom left to right in Figure 4.7. Note that the colorless LR images and bicubic interpolations are visually similar but different in details noticed by zooming out.
4.4 Summary

In this Chapter, we demonstrate that Generative adversarial loss can pratically improve the quality of generation in text or image domain. Basically, we build the discriminator on top of various generative models to introduce another extra game theoretic loss. Another alternative optimization is further derived in such setting for both generative and discriminative loss. Thus, in the next chapter, we will further explore the possibility of benefits in the biological image processing.
In this chapter, we will further explore how VAE+GAN framework can be applied to synthetic/system biology and medical image processing tasks.

5.1 Turing Pattern Synthesis

5.1.1 Background

Recent research has shown that morphogen gradients may not be necessary for pattern formation. However, it remains unknown whether and how scale invariance will

Figure 5.1: Left: Circuit logic. Right: Self-organized pattern formation in engineered bacteria.
emerge in most biological systems. The difficulty of this question is mainly because the complexity of natural systems may have many confounding functional elements that indicate complicate quantitative experiments and data interpretation. To address this limitation, some researchers propose to examine the scaling dynamics of pattern formation in Escherichia coli programmed by a synthetic gene circuit that was previously developed by Payne et al. (2013) in Figure 5.1.1. The circuit consists of a mutant T7 RNA polymerase Tan et al. (2009) (T7RNAP) that activates its own expression and that of LuxR and LuxI. LuxI synthesizes an acyl-homoserine lactone (AHL), a membrane-diffusible chemical that upon binding and activating LuxR, can induce expression of T7 lysozyme. Lysozyme inhibits T7RNAP by forming a stable complex with it, and the resulting T7-lysozyme complex inhibits the transcription of T7RNAP as well Stano and Patel (2004). It is also found that in the reported circuit dynamics, CFP and mCherry fluorescent proteins are co-expressed with T7RNAP and lysozyme, respectively. Since AHL is highly diffusible, it is usually negligible for AHL spatial gradient over the length scale of pattern formation in this system. When other factors are controlled, the time required for AHL to reach a critical concentration is inversely proportional to the size of the growth environment. To this end, it is concluded that the accumulation and production of AHL can help the growing colony to sense the domain size and time the pattern formation accordingly Payne et al. (2013). The property of sensing ability indicates that we can launch a well-defined experimental platform to quantify the pattern scale. In addition, it is experimentally found that in such system, a simple mechanism of scale invariance can emerge without the requirement of scaling a diffusible morphogen gradient. As shown in Figure 5.1, the left panel is the circuit that contains a T7 RNA polymerase which activates its own expression as well as the expression of LuxI and LuxR. Once its activation by T7RNAP (T7), LuxI mediates synthesis of AHL (A) that can diffuse through the cell membrane. When the global AHL concentration is larger than a threshold,
LuxR is bound by intracellular AHL and then further activates the synthesis of T7 lysozyme (L). Lysozyme can bind to the T7RNAP and form a T7-lysozyme complex, inhibiting the T7RNAP binding to the T7 promoter and T7RNAP transcription. In this process, the AHL concentration is controlled by its initial concentration and the domain size. The expression rates of T7RNAP, lysozyme, and AHL are all controlled by the spatially dependent gene expression capacity. The described circuit dynamics can be mathematically simulated as partial differential equations in Figure 5.1.1.

5.1.2 Fast Simulation

The main drawback of solving such PDEs is very time consuming. Therefore, we propose a fast approximate solver for the PDEs by conditional deep generative models. The basic idea is to generate desire pattern based on the coefficients of PDEs. When training the neural network, we may use some randomly generated coefficients (uniformly distributed in the coefficients space, within the reasonable range), say about $10^6$ combination of parameters. Then ideally we can efficiently predict $10^{10}$ or even more combinations of parameters using trained Neural network with high efficiency.
Figure 5.1.2 shows the basic architecture of our generative model. However, we did not directly generate the pattern as image; we make an assumption that the ring pattern is centrosymmetric. Thus, we can reduce our generation from matrix space to vector space. First, since the peak value of RFP concentration over the space (output) varies significantly, we need to predict a scalar representing the peak value for each ring pattern. Notice the scale of peak value can range from $10^{-10}$ to 10, so we prefer to use the logarithm transformation as the target value. Then, we normalize other concentration value to the interval $[0, 1]$ by dividing the peak value. This data preprocessing is visualized in Figure 5.1.2(a). In the end, we also need to predict the shape of pattern in 1D form. Due to our assumption, we only need to characterize the shape with respect to the diameter to the ring pattern center. In our model, we use a recurrent neural networks (RNN) with LSTM units Hochreiter and Schmidhuber (1997) to deal with this task. Additionally, we initialize our RNN model from the farthest distance to the ring center, since the concentration will always be 0 there.

5.1.3 Ring Pattern Results

We simulate 100000 sets of coefficients of PDEs, and their resulted ring patterns. 80%, 10% and 10% of data are used as training, validation and test dataset, respectively. We only show the results on held-out test dataset in Figure 5.1.3. We can observe that except for several very irregular shape most of the shape can correctly predicted. Similarly, for the peak value, very few predictions stand as outliers beyond the ground truth. If we consider the $R^2$ for these two prediction on test data, we achieve 0.95 and 0.994 for peak value and shape values.
5.2 Cardiovascular Lesion Detection

In this section, we will test the model Unets with generative adversarial loss in the medical imaging application. The diameter of blood vessel can measure to what extent, the patient has the risk of cardiovascular disease. In negative case, the diameter of blood vessel should change in a smooth way; however, if there exists a sudden narrow area, it is highly suspected that a blood cot happens. Due the variation of doctors’ professional ability, the diagnosis may differ a lot even if the same x-ray image is presented. So, we propose to build an automatic diagnosis system than can provide a consistent diagnosis as reference for the doctor. Basically, our system has two step: first, we use the object detection algorithm YOLO to locate the suspected lesion area by a rectangle bounding box; then, we run our unets
to extract the contour of the blood vessel within the bounding box, which plays the role of edge detection. In this way, we can provide a global lesion detection result and a local contour extraction of suspected region as the diagnosis reference. Figure 5.2 shows an example of our result. Our next step is to require some doctors to validate our findings, which will be our future work.

5.3 Summary

Besides traditional machine learning tasks, the deep generative can be further deployed to the biological domain if we can collect enough data. Especially in the medical area, the automatic system can even achieve the professional diagnosis level, potentially being able to benefit thousands of people who cannot afford the experi-
sive health care fee. At least, from the perspective of efficiency, reservation system is always a big issue due to high volume patients, but we can help to discover the suspected disease as soon as possible.
6

Conclusion

6.1 Summary

This dissertation mainly addresses the important unsupervised learning problem in modeling high-dimensional data with complicated manifold. Motivated by fact that non-linear transformation of base distribution can represent any desired complex distribution, we explore the problem from the perspective of deep generative models. However, inference and learning with Bayesian generative models is often intractable, especially for models that employ continuous latent variables, and so fast approximate methods are needed. Variational Bayesian (VB) methods deal with this problem by approximating the true posterior that has a tractable parametric form and then identifying the set of parameters that maximize a variational lower bound on the marginal likelihood. Thus, we first discuss the Variational Auto-encoders framework and its inference method. Indeed, efficient stochastic gradient variational Bayesian (SGVB) estimators have been developed. We provide a complementary perspective by using second order stochastic backpropagation that is equivalent to SGVB with natural gradient and apply it to deep latent Gaussian models. In addition, we describe a clever approach to obtain curvature information with low computational
cost, thus making the Newton’s method both scalable and efficient. We also show
that the variance of the lower bound estimator can be bounded by a dimension-free
constant, extending the work of Rezende et al. (2014) that discussed a specific bound
for univariate function. Third, we demonstrate the performance of our method for
Bayesian logistic regression and the VAE model in comparison to commonly used
algorithms. Convergence rate is shown to be competitive or faster.

Stochastic backpropagation overcomes many limitations of traditional inference
methods such as the mean-field or wake-sleep algorithms due to the existence of
efficient computations of an unbiased estimate of the gradient of the variational
lower bound. A joint learning for both generative and proposal distributions in
VAE framework is the main advantage, learning an amortized variational inference
method. A detailed comparison among many Bayesian inference methods, including
MCMC, mean-filed and amortized variational inference, is illustrated for an epi-
demiological application modeling by hierarchical Graph-Coupled Hidden Markov
Model (hGCHMM). The reason we emphasize this model is that the hGCHMM is a
general and flexible framework to heterogeneously simulate the disease spread in a
dynamic social network. It simulates a discretized Suspicious-Infections-Suspicious
(SIS) model Cooper and Lipsitch (2004), evolved from standard hidden Markov pro-
cess, and is able to reduce to a homogeneous version, the Graph-coupled HMM or
even latent Dirichlet allocation (LDA) Gruber et al. (2007). In this dissertation, we
propose to use the amortized variational inference. Differing from the traditional
VI methods, such as mean-field Beal (2003), we get rid of the complicated gradient
computation for non-conjugate probabilistic models but build a tractable dynamic
recognition model corresponding to the generative process. The recognition model
is equivalent to the variational distribution in VI parlance. Taking advantage of the
binary variables in the hGCHMM, the recognition model can be completely con-
structed by a sigmoid belief network (SBN) Sutskever and Hinton (2008), where it
is straightforward to generalize to the categorical case by inducing softmax output, thus allowing our approach to reproduce the potential tasks introduced in Fan et al. (2015c). In addition, we overcome the major drawback—lacking the capability to simulate highly contagious disease, by proposing a dynamic auto-encoding variational inference method.

Though it is highly suggested that VAEs with highly flexible priors or highly flexible approximate posteriors can obtain values of lower bound that are near their own log-likelihood, we still need to fill the gap between the pre-defined explicit model and real data distribution. Thus, we utilize the advantage of implicit deep generative model, placing GAN on top of traditional auto-encoder based generative models. Practically, we found the additional generative adversarial loss can improve the quality of generated samples. We validate our claim in both text and image generation tasks with some benchmark machine learning datasets. Additionally, we also try to solve biological problems. The first is to learn an approximate PDEs solver that can simulate the pattern generation in synthetic/system biology. The second is to build an automatic cardiovascular lesion detection system and provide a consistent diagnosis reference to doctors.

6.2 Future Works

There are many potential extensions of our previous works. Some of the extensions are described below.

The first future research of VAE inference method can focus on some difficult or large scale deep models such as LSTM Sutskever et al. (2014) or RNNs Martens and Sutskever (2011). Because our experiments had higher training variational lower bounds, we infer that our second order algorithms may be capable of addressing the under-fitting problem in stochastic RNN Gregor et al. (2015). Another possible area of future research will be reinforcement learning (RL) Mnih et al. (2015); Williams
(1992). Many RL problems can be reduced to how to compute gradients of expectations (e.g., in policy gradient methods) and there has been series of exploration in this area for natural gradients. However, we would suggest that it might be interesting to consider where stochastic backpropagation fits in our framework and how 2nd order computations can help.

The further discussion on $\beta_n$ in Chapter 3 provides another biological interpretation. Therefore, one possible area of exploration of hGCHMMs would be to implement this setting or investigate infection network learning by detecting the disease spread path (auxiliary variable $R$). We can further relax the heterogeneity assumption to a cluster assumption, where this tradeoff can be realized by constructing a nonparametric version GCHMMs, enforcing similar HMMs to share the same parameters.

In this end, we will discuss the future work of powerful generative adversarial loss. Since GAN itself suffers so many problems theoretically, the most fundamental research will be exploring better training algorithm with respect to the optimization towards the Nash equilibrium in high dimensional space. Practically, since GAN can use the supervised ratio estimation technique to approximate many loss functions (e.g., the KL divergence), we can also develop many interesting architecture to help improve existed models that maximum likelihood estimation.
Appendix A

A.1 Proofs of the Extending Gaussian Gradient Equations in Chapter 2

Lemma 5. Let $f(z) : \mathbb{R}^d \rightarrow \mathbb{R}$ be an integrable and twice differentiable function. The second gradient of the expectation of $f(z)$ under a Gaussian distribution $\mathcal{N}(z|\mu, C)$ with respect to the mean $\mu$ can be expressed as the expectation of the Hessian of $f(z)$:

$$
\nabla^2_{\mu_i, \mu_j} \mathbb{E}_{\mathcal{N}(z|\mu, C)}[f(z)] = \mathbb{E}_{\mathcal{N}(z|\mu, C)}[\nabla^2_{z_i, z_j} f(z)] = 2 \nabla_{C_{ij}} \mathbb{E}_{\mathcal{N}(z|\mu, C)}[f(z)]. \quad (A.1)
$$

Proof. From Bonnet’s theorem Bonnet (1964), we have

$$
\nabla_{\mu_i} \mathbb{E}_{\mathcal{N}(z|\mu, C)}[f(z)] = \mathbb{E}_{\mathcal{N}(z|\mu, C)}[\nabla_{z_i} f(z)]. \quad (A.2)
$$
Moreover, we can get the second order derivative,

\[
\nabla^2_{\mu_i, \mu_j} \mathbb{E}_{N(z|\mu, C)}[f(z)] = \nabla_{\mu_i} \left( \mathbb{E}_{N(z|\mu, C)}[\nabla_{z_j} f(z)] \right)
\]

\[
= \int \nabla_{\mu_i} N(z|\mu, C) \nabla_{z_j} f(z) \, dz
\]

\[
= - \int \nabla_{z_i} N(z|\mu, C) \nabla_{z_j} f(z) \, dz
\]

\[
= - \left[ \int N(z|\mu, C) \nabla_{z_j} f(z) \, dz \right]_{z_i = -\infty}^{z_i = +\infty} + \int N(z|\mu, C) \nabla_{z_i, z_j} f(z) \, dz
\]

\[
= \mathbb{E}_{N(z|\mu, C)}[\nabla^2_{z_i, z_j} f(z)]
\]

\[
= 2 \nabla_{C_{ij}} \mathbb{E}_{N(z|\mu, C)}[f(z)]
\]

where the last equality we use the equation

\[
\nabla_{C_{ij}} N(z|\mu, C) = \frac{1}{2} \nabla^2_{z_i, z_j} N(z|\mu, C).
\] (A.3)

\[
\Box
\]

**Lemma 6.** Let \( f(z) : \mathcal{R}^d \rightarrow \mathcal{R} \) be an integrable and fourth differentiable function. The second gradient of the expectation of \( f(z) \) under a Gaussian distribution \( N(z|\mu, C) \) with respect to the covariance \( C \) can be expressed as the expectation of the fourth gradient of \( f(z) \)

\[
\nabla^2_{C_{1,j}, C_{k,l}} \mathbb{E}_{N(z|\mu, C)}[f(z)] = \frac{1}{4} \mathbb{E}_{N(z|\mu, C)}[\nabla^4_{z_i, z_j, z_k, z_l} f(z)].
\] (A.4)

**Proof.** From Price’s theorem Price (1958), we have

\[
\nabla_{C_{1,j}} \mathbb{E}_{N(z|\mu, C)}[f(z)] = \frac{1}{2} \mathbb{E}_{N(z|\mu, C)}[\nabla^2_{z_i, z_j} f(z)].
\] (A.5)
\[ \nabla_{C_{i,j},C_{k,l}}^2 \mathbb{E}_{N(z|\mu, C)}[f(z)] = \frac{1}{2} \nabla_{C_{k,l}} \left( \mathbb{E}_{N(z|\mu, C)}[\nabla_{z_{i},z_{j}}^2 f(z)] \right) \]

\[ = \frac{1}{2} \int \nabla_{C_{k,l}} \mathcal{N}(z|\mu, C) \nabla_{z_{i},z_{j}}^2 f(z) \, dz \]

\[ = \frac{1}{4} \int \nabla_{z_{k},z_{l}} \mathcal{N}(z|\mu, C) \nabla_{z_{i},z_{j}}^2 f(z) \, dz \]

\[ = \frac{1}{4} \int \mathcal{N}(z|\mu, C) \nabla_{z_{i},z_{j},z_{k},z_{l}}^2 f(z) \, dz \]

\[ = \frac{1}{4} \mathbb{E}_{N(z|\mu, C)}[\nabla_{z_{i},z_{j},z_{k},z_{l}}^2 f(z)]. \]

In the third equality we use the Eq.(A.3) again. For the fourth equality we use the product rule for integrals twice.

From Eq.(A.2) and Eq.(A.5) we can straightforward write the second order gradient of interaction term as well:

\[ \nabla_{\mu_{i},C_{k,l}}^2 \mathbb{E}_{N(\mu, C)}[f(z)] = \frac{1}{2} \mathbb{E}_{N(\mu, C)}[\nabla_{z_{i},z_{j}}^3 f(z)]. \quad (A.6) \]

## A.2 Proof of Theorem 1 in Chapter 2

By using the linear transformation \( z = \mu + R\epsilon \), where \( \epsilon \sim N(0, I_d) \), we can generate samples from any Gaussian distribution \( \mathcal{N}(\mu, C) \), \( C = RR^\top \), where \( \mu(\theta), R(\theta) \) are both dependent on parameter \( \theta = (\theta_i)_{i=1}^d \).

Then the gradients of the expectation with respect to \( \mu \) and (or) \( R \) is

\[ \nabla_{R} \mathbb{E}_{N(\mu, C)}[f(z)] = \nabla_{R} \mathbb{E}_{N(0, I)}[f(\mu + R\epsilon)] = \mathbb{E}_{N(0, I)}[\epsilon g^\top] \]

\[ \nabla_{R_{i,j},R_{k,l}}^2 \mathbb{E}_{N(\mu, C)}[f(z)] = \nabla_{R_{i,j}} \mathbb{E}_{N(0, I)}[\epsilon_i \epsilon_j g_k] = \mathbb{E}_{N(0, I)}[\epsilon_i \epsilon_j H_{ik}] \]

\[ \nabla_{\mu_{i},R_{k,l}}^2 \mathbb{E}_{N(\mu, C)}[f(z)] = \nabla_{\mu_{i}} \mathbb{E}_{N(0, I)}[\epsilon_i g_k] = \mathbb{E}_{N(0, I)}[\epsilon_i H_{ik}] \]

\[ \nabla_{\mu_{i}}^2 \mathbb{E}_{N(\mu, C)}[f(z)] = \mathbb{E}_{N(0, I)}[H] \]
where $g = \{g_j\}_{j=1}^{d_z}$ is the gradient of $f$ evaluated at $\mu + R\epsilon$, $H = \{H_{ij}\}_{d_z \times d_z}$ is the Hessian of $f$ evaluated at $\mu + R\epsilon$.

Furthermore, we write the second order derivatives into matrix form:

$$
\nabla^2_{\mu, R} \mathbb{E}_{N(\mu, C)}[f(z)] = \mathbb{E}_{N(0, I)}[\epsilon^\top \otimes H],
$$

$$
\nabla^2_{R} \mathbb{E}_{N(\mu, C)}[f(z)] = \mathbb{E}_{N(0, I)}[(\epsilon \epsilon^T) \otimes H].
$$

For a particular model, such as deep generative model, $\mu$ and $C$ are depend on the model parameters, we denote them as $\theta = (\theta_i)_{i=1}^{d}$, i.e. $\mu = \mu(\theta), C = C(\theta)$. Combining Eq.A.2 and Eq.A.5 and using the chain rule we have

$$
\nabla_{\theta_i} \mathbb{E}_{N(\mu, C)}[f(z)] = \mathbb{E}_{N(\mu, C)} \left[ g^\top \frac{\partial \mu}{\partial \theta_i} + \frac{1}{2} \text{Tr} \left( H \frac{\partial C}{\partial \theta_i} \right) \right],
$$

where $g$ and $H$ are the first and second order gradient of $f(z)$ for abusing notation. This formulation involves matrix-matrix product, resulting in an algorithmic complexity $O(d_z^2)$ for any single element of $\theta$ w.r.t $f(z)$, and $O(dd_z^2), O(d^2d_z^2)$ for overall gradient and Hessian respectively.

Considering $C = RR^\top, z = \mu + R\epsilon$,

$$
\nabla_{\theta_i} \mathbb{E}_{N(\mu, C)}[f(z)] = \mathbb{E}_{N(0, I)} \left[ g^\top \frac{\partial \mu}{\partial \theta_i} + \text{Tr} \left( eg^\top \frac{\partial R}{\partial \theta_i} \right) \right]
$$

$$
= \mathbb{E}_{N(0, I)} \left[ g^\top \frac{\partial \mu}{\partial \theta_i} + g^\top \frac{\partial R}{\partial \theta_i} \epsilon \right].
$$
For the second order, we have the following separated formulation:

\[
\nabla^2_{\theta_1, \theta_2} E_{\mathcal{N}(\mu, \Sigma)}[f(z)] = \nabla_{\theta_1} E_{\mathcal{N}(0, \Sigma)} \left[ \sum_i g_i \frac{\partial \mu_i}{\partial \theta_1} + \sum_{i,j} \epsilon_{ij} g_i \frac{\partial^2 \mu_i}{\partial \theta_1 \partial \theta_2} \right]
\]

\[
= E_{\mathcal{N}(0, \Sigma)} \left[ \sum_{i,j} H_{ij} \left( \frac{\partial \mu_j}{\partial \theta_1} + \sum_k \epsilon_{jk} \frac{\partial R_{ik}}{\partial \theta_1} \right) \frac{\partial \mu_i}{\partial \theta_2} + \sum_i g_i \frac{\partial^2 \mu_i}{\partial \theta_1 \partial \theta_2} \right]
\]

\[
+ \sum_{i,j} \epsilon_{ij} \left( \sum_k H_{ik} \left( \frac{\partial \mu_k}{\partial \theta_1} + \sum_l \epsilon_{kl} \frac{\partial R_{jl}}{\partial \theta_1} \right) \right) \frac{\partial R_{ij}}{\partial \theta_2} + \sum_{i,j} \epsilon_{ij} g_i \frac{\partial^2 R_{ij}}{\partial \theta_1 \partial \theta_2}
\]

\[
= E_{\mathcal{N}(0, \Sigma)} \left[ \frac{\partial \mu^\top}{\partial \theta_1} H \frac{\partial \mu}{\partial \theta_2} + \left( \frac{\partial R}{\partial \theta_1} \epsilon \right)^\top H \frac{\partial \mu}{\partial \theta_2} + g^\top \frac{\partial^2 \mu}{\partial \theta_1 \partial \theta_2} \right]
\]

\[
+ \left( \frac{\partial R}{\partial \theta_2} \epsilon \right)^\top H \frac{\partial \mu}{\partial \theta_1} + \left( \frac{\partial R}{\partial \theta_1} \epsilon \right)^\top H \frac{\partial R}{\partial \theta_2} \epsilon + g^\top \frac{\partial^2 R}{\partial \theta_1 \partial \theta_2} \epsilon
\]

\[
= E_{\mathcal{N}(0, \Sigma)} \left[ \frac{\partial (\mu + R \epsilon)}{\partial \theta_1}^\top H \frac{\partial (\mu + R \epsilon)}{\partial \theta_2} + g^\top \frac{\partial^2 (\mu + R \epsilon)}{\partial \theta_1 \partial \theta_2} \right].
\]

It is noticed that for second order gradient computation, it only involves matrix-vector or vector-vector multiplication, thus leading to an algorithmic complexity \(O(d^2)\) for each pair of \(\theta\).

One practical parametrization is \(C = \text{diag}\{\sigma_1^2, ..., \sigma_d^2\}\) or \(R = \text{diag}\{\sigma_1, ..., \sigma_d\}\), which will reduce the actual second order gradient computation complexity, albeit
the same order of $O(d_z^2)$. Then we have

$$
\nabla_{\theta_i} \mathbb{E}_{N(\mu, C)}[f(z)] = \mathbb{E}_{N(0, I)} \left[ g^T \frac{\partial \mu}{\partial \theta_i} + \sum_i \epsilon_i g^T \frac{\partial \sigma_i}{\partial \theta_i} \right]
$$

$$
= \mathbb{E}_{N(0, I)} \left[ g^T \frac{\partial \mu}{\partial \theta_i} + (\epsilon \otimes g)^\top \frac{\partial \sigma}{\partial \theta_i} \right], \quad (A.7)
$$

$$
\nabla^2_{\theta_1, \theta_2} \mathbb{E}_{N(\mu, C)}[f(z)] = \mathbb{E}_{N(0, I)} \left[ \frac{\partial \mu}{\partial \theta_{i_1}}^\top H \frac{\partial \mu}{\partial \theta_{i_2}} + \left( \epsilon \otimes \frac{\partial \sigma}{\partial \theta_{i_1}} \right)^\top H \left( \epsilon \otimes \frac{\partial \sigma}{\partial \theta_{i_2}} \right) + \frac{\partial^2 \mu}{\partial \theta_{i_1} \partial \theta_{i_2}} \right]
$$

$$
+ \left( \epsilon \otimes \frac{\partial \sigma}{\partial \theta_{i_2}} \right)^\top H \left( \frac{\partial \mu}{\partial \theta_{i_1}} + \epsilon \otimes \frac{\partial \sigma}{\partial \theta_{i_1}} \right) H \left( \epsilon \otimes \frac{\partial \sigma}{\partial \theta_{i_2}} \right)
$$

$$
+ (\epsilon \otimes g)^\top \frac{\partial^2 \sigma}{\partial \theta_{i_1} \partial \theta_{i_2}} \right], \quad (A.8)
$$

where $\otimes$ is Hadamard (or element-wise) product, and $\sigma = (\sigma_1, ..., \sigma_{d_z})^\top$.

**Derivation for Hessian-Free SGVI without $\theta$ Plugging** This means $(\mu, R)$ is the parameter for variational distribution. According the derivation in this section, the Hessian matrix with respect to $(\mu, R)$ can represented as $H_{\mu, R} = \mathbb{E}_{N(0, I)} \left[ \left( \begin{array}{c} 1 \\ \epsilon \end{array} \right) \left[ 1, \epsilon^\top \right] \otimes \left[ \begin{array}{c} \mu \\ \sigma \end{array} \right] \right]$.

For any $d_z \times (d_z + 1)$ matrix $V$ with the same dimensionality of $[\mu, R]$, we also have the Hessian-vector multiplication equation.

$$
H_{\mu, R} vec(V) = \mathbb{E}_{N(0, I)} \left[ vec \left( HV \left[ \begin{array}{c} 1 \\ \epsilon \end{array} \right] \left[ 1, \epsilon^\top \right] \right) \right]
$$

where $vec(\cdot)$ denotes the vectorization of the matrix formed by stacking the columns into a single column vector. This allows an efficient computation both in speed and storage.
A.3 Forward-Backward Algorithm for Special Variation Auto-encoder Model in Chapter 2

We illustrate the equivalent deep neural network model (Figure 2.1) by setting $M = 1$ in VAE, and derive the gradient computation by lawyer-wise backpropagation. Without generalization, we give discussion on the binary input and diagonal covariance matrix, while it is straightforward to write the continuous case. For binary input, the parameters are $\{(W_i, b_i)\}_{i=1}^5$.

The feedforward process is as follows:

\[
\begin{align*}
    h_e &= \tanh(W_1x + b_1) \\
    \mu_e &= W_2h_e + b_2 \\
    \log \sigma_e &= 0.5 \times (W_3h_e + b_3) \\
    \epsilon &\sim \mathcal{N}(0, I_{d_z}) \\
    z &= \mu_e + \sigma_e \odot \epsilon \\
    h_d &= \tanh(W_4z + b_4) \\
    y &= \text{sigmoid}(W_5h_d + b_5).
\end{align*}
\]

Considering the cross-entropy loss function, the backward process for gradient
backpropagation computation is:

\[ \delta_5 = x \odot (1 - y) + (1 - x) \odot y \]
\[ \nabla W_5 = \delta_5 h_d^\top, \quad \nabla b_5 = \delta_5 \]
\[ \delta_4 = (W_5^\top \delta_5) \odot (1 - h_d \odot h_d) \]
\[ \nabla W_4 = \delta_4 z^\top, \quad \nabla b_4 = \delta_4 \]
\[ \delta_3 = 0.5 \ast (W_4^\top \delta_4) \odot (z - \mu_e) + 1 - \sigma_e^2 \]
\[ \nabla W_3 = \delta_3 h_e^\top, \quad \nabla b_3 = \delta_3 \]
\[ \delta_2 = W_4^\top \delta_4 - \mu_e \]
\[ \nabla W_2 = \delta_2 h_e^\top, \quad \nabla b_2 = \delta_2 \]
\[ \delta_1 = (W_2^\top \delta_2 + W_3^\top \delta_3) \odot (1 - h_e \odot h_e) \]
\[ \nabla W_1 = \delta_1 x^\top, \quad \nabla b_1 = \delta_1. \]

Notice that when we compute the differences \( \delta_2, \delta_3 \), we also include the prior term which acts as the role of regularization penalty. In addition, we can add the \( \mathcal{L}_2 \) penalty to the weight matrix as well. The only modification is to change the expression of \( \nabla W_i \) by adding \( \lambda W_i \), where \( \lambda \) is a tunable hyper-parameter.

### A.4 \( \mathcal{R} \)-Operator Derivation in Chapter 2

Define \( \mathcal{R}_v \{ f(\theta) \} = \frac{\partial}{\partial \gamma} f(\theta + \gamma v) \bigg|_{\gamma=0} \), then \( H_\theta v = \mathcal{R}_v \{ \nabla_\theta F(\theta) \} \). First, mapping \( v \) to \( \{ \mathcal{R}(W_i), \mathcal{R}(b_i) \}_{i=1}^5 \). Then, we derive the \( Hv \) to \( \{ \mathcal{R}(DW_i), \mathcal{R}(Db_i) \}_{i=1}^5 \), where \( \mathcal{D} \)-operator means take derivative with respect to objective function.

Denote \( s_i = W_i u + b_i \), where \( u \) can represent any vector used in neural networks.
Forward Pass:

\[
\mathcal{R}\{s_1\} = \mathcal{R}\{W_1\}x + \mathcal{R}\{b_1\} \quad (\text{Since } \mathcal{R}\{x\} = 0)
\]

\[
\mathcal{R}\{h_e\} = \mathcal{R}\{s_1\} \tanh'(s_1)
\]

\[
\mathcal{R}\{\mu_e\} = \mathcal{R}\{s_2\} = \mathcal{R}\{W_2\}h_e + W_2\mathcal{R}\{h_e\} + \mathcal{R}\{b_2\}
\]

\[
\mathcal{R}\{s_3\} = \mathcal{R}\{W_3\}h_e + W_3\mathcal{R}\{h_e\} + \mathcal{R}\{b_3\}
\]

\[
\mathcal{R}\{\sigma_e\} = \mathcal{R}\{s_3\} \exp\{s_3/2\} \quad (\text{Since } (e^x)' = e^x)
\]

\[
\mathcal{R}\{z\} = \mathcal{R}\{\mu_e\} + \mathcal{R}\{\sigma_e\} \odot \epsilon
\]

\[
\mathcal{R}\{s_4\} = \mathcal{R}\{W_4\}z + W_4\mathcal{R}\{z\} + \mathcal{R}\{b_4\}
\]

\[
\mathcal{R}\{h_d\} = \mathcal{R}\{s_4\} \tanh'(s_4)
\]

\[
\mathcal{R}\{s_5\} = \mathcal{R}\{W_5\}h_d + W_5\mathcal{R}\{h_d\} + \mathcal{R}\{b_5\}
\]

\[
\mathcal{R}\{y\} = \mathcal{R}\{s_5\} \text{sigmoid}'(s_5)
\]

Backwards Pass:

\[
\mathcal{R}\{Dy\} = \mathcal{R} \left\{ \frac{\partial \mathcal{L}(x, y)}{\partial y} \right\} = \frac{\partial^2 \mathcal{L}(x, y)}{\partial^2 y} \mathcal{R}\{y\}
\]

\[
\mathcal{R}\{Ds_5\} = \mathcal{R}\{Dy\} \odot \text{sigmoid}'(s_5) + \mathcal{D}y \odot \text{sigmoid}''(s_5) \odot \mathcal{R}\{s_5\}
\]

\[
\mathcal{R}\{DW_5\} = \mathcal{R}\{Ds_5\}h_d^\top + Ds_5\mathcal{R}\{h_d\}^\top
\]

\[
\mathcal{R}\{Db_5\} = \mathcal{R}\{Ds_5\}
\]

\[
\mathcal{R}\{Dh_d\} = \mathcal{R}\{W_5\}^\top Ds_5 + W_5^\top \mathcal{R}\{Ds_5\}
\]

where the rest can follow the same recursive computation which is similar to gradient derivation.

A.5 Variance Analysis (Proof of Theorem 2) in Chapter 2

In this part we analyze the variance of the stochastic estimator.
Lemma 7. For any convex function $\phi$,

$$
\mathbb{E}[\phi(f(\epsilon) - \mathbb{E}[f(\epsilon)])] \leq \mathbb{E}\left[ \phi\left( \frac{\pi}{2} \langle \nabla f(\epsilon), \eta \rangle \right) \right],
$$

(A.9)

where $\epsilon, \eta \sim \mathcal{N}(0, \mathbf{I}_d)$ and $\epsilon, \eta$ are independent.

Proof. Using interpolation $\gamma(\omega) = \epsilon \sin(\omega) + \eta \cos(\omega)$, then $\gamma'(\omega) = \epsilon \cos(\omega) - \eta \sin(\omega)$, and $\gamma(0) = \eta, \gamma(\pi/2) = \epsilon$. Furthermore, we have the equation,

$$
f(\epsilon) - f(\eta) = \int_{0}^{\pi} \frac{d}{d\omega} f(\gamma(\omega)) d\omega = \int_{0}^{\pi} \langle \nabla f(\gamma(\omega)), \gamma'(\omega) \rangle d\omega.
$$

Then

$$
\mathbb{E}_\epsilon[\phi(f(\epsilon) - \mathbb{E}[f(\epsilon)])] = \mathbb{E}_\epsilon[\phi(f(\epsilon) - \mathbb{E}_\eta[f(\eta)])] \leq \mathbb{E}_{\epsilon, \eta}[\phi(f(\epsilon) - f(\eta))]
$$

$$
= \mathbb{E}\left[ \phi\left( \frac{2}{\pi} \int_{0}^{\pi} \frac{\pi}{2} \langle \nabla f(\gamma(\omega)), \gamma'(\omega) \rangle d\omega \right) \right]
$$

$$
\leq \frac{2}{\pi} \mathbb{E}\left[ \int_{0}^{\pi} \phi\left( \frac{\pi}{2} \langle \nabla f(\gamma(\omega)), \gamma'(\omega) \rangle \right) d\omega \right]
$$

$$
= \frac{2}{\pi} \int_{0}^{\pi} \mathbb{E}\left[ \phi\left( \frac{\pi}{2} \langle \nabla f(\gamma(\omega)), \gamma'(\omega) \rangle \right) \right] d\omega
$$

$$
= \mathbb{E}\left[ \phi\left( \frac{\pi}{2} \langle \nabla f(\epsilon), \eta \rangle \right) \right].
$$

The above two inequalities use the Jensen’s Inequality. The last equation holds because both $\gamma$ and $\gamma'$ follow $\mathcal{N}(0, \mathbf{I}_d)$, and $\mathbb{E}[\gamma \gamma^T] = 0$ implies they are independent. □

Before giving a dimensional free bound, we first let $\phi(x) = x^2$ and can obtain a relatively loosen bound of variance for our estimators. Assuming $f$ is a $L$-Lipschitz differentiable function and $\epsilon \sim \mathcal{N}(0, \mathbf{I}_d)$, the following inequality holds:

$$
\mathbb{E}[(f(\epsilon) - \mathbb{E}[f(\epsilon)])^2] \leq \frac{\pi^2 L^2 d_z}{4}.
$$

(A.10)
To see the reason, we only need to reuse the double sample trick and the expectation of Chi-squared distribution, we have
\[
E \left[ \left( \frac{\pi}{2} \langle \nabla f(\epsilon), \eta \rangle \right)^2 \right] \leq \frac{\pi^2 L^2}{4} E[\|\eta\|^2] = \frac{\pi^2 L^2 d_\epsilon}{4}.
\]

Then by Lemma 7, Eq.(A.11) holds. To get a tighter bound as in Theorem 2, we give the following Lemma 8 and Lemma 9 first.

**Lemma 8** (?). A random variable \(X\) with mean \(\mu = E[X]\) is sub-Gaussian if there exists a positive number \(\sigma\) such that for all \(\lambda \in \mathcal{R}^+\)
\[
E \left[ e^{\lambda(X-\mu)} \right] \leq e^{\sigma^2 \lambda^2/2},
\]
then we have
\[
E \left[ (X - \mu)^2 \right] \leq \sigma^2.
\]

**Proof.** By Taylor’s expansion,
\[
E \left[ e^{\lambda(X-\mu)} \right] = E \left[ \sum_{i=1}^{\infty} \frac{\lambda^i}{i!} (X - \mu)^i \right] \leq e^{\sigma^2 \lambda^2/2} = \sum_{i=0}^{\infty} \frac{\sigma^2 \lambda^2}{2^i i!}.
\]

Thus \(\frac{\lambda^2}{2} E[(X - \mu)^2] \leq \frac{\sigma^2 \lambda^2}{2} + o(\lambda^2).\) Let \(\lambda \to 0\), we have \(\text{Var}(X) \leq \sigma^2.\)

**Lemma 9.** If \(f(x)\) is a \(L\)-lipschitz differentiable function and \(\epsilon \in \mathcal{N}(0, I_{d_\epsilon})\) then the random variable \(f(\epsilon) - E[f(\epsilon)]\) is sub-Gaussian with parameter \(L\), i.e. for all \(\lambda \in \mathcal{R}^+\)
\[
E \left[ e^{\lambda(f(\epsilon)-E[f(\epsilon)])} \right] \leq e^{L^2 \lambda^2 \pi^2/8}.
\]

**Proof.** From Lemma 7, we have
\[
E \left[ e^{\lambda(f(\epsilon)-E[f(\epsilon)])} \right] \leq E_{\epsilon, \eta} \left[ e^{\lambda \frac{\pi}{2} \langle \nabla f(\epsilon), \eta \rangle} \right]
\]
\[
= E_{\epsilon, \eta} \left[ e^{-\lambda \sum_{i=1}^{d_\epsilon} \frac{\eta_i}{\sqrt{\epsilon_i}} f(\epsilon) \left( \sum_{i=1}^{d_\epsilon} \frac{\eta_i}{\sqrt{\epsilon_i}} \right)^2} \right] = E_{\epsilon} \left[ e^{-\lambda \sum_{i=1}^{d_\epsilon} \frac{\eta_i}{\sqrt{\epsilon_i}} f(\epsilon) \left( \sum_{i=1}^{d_\epsilon} \frac{\eta_i}{\sqrt{\epsilon_i}} \right)^2} \right]
\]
\[
\leq \exp \left( \frac{\lambda^2 \pi^2 L^2}{8} \right).
\]
Proof of Theorem 2 Combining Lemma 8 and Lemma 9 we complete the proof of Theorem 2.

In addition, we can also obtain a tail bound,
\[
\Pr_{\epsilon \sim \mathcal{N}(0, \mathbf{I}_d)} (|f(\epsilon) - \mathbb{E}[f(\epsilon)]| \geq t) \leq 2e^{-\frac{2t^2}{\pi^2 L^2}}. \tag{A.11}
\]

For \( \lambda > 0 \), let \( \epsilon_1, \ldots, \epsilon_M \) be i.i.d random variables with distribution \( \mathcal{N}(0, \mathbf{I}_d) \),
\[
\Pr\left( \frac{1}{M} \sum_{m=1}^{M} f(\epsilon_m) - \mathbb{E}[f(\epsilon)] \geq t \right) = \Pr\left( \sum_{m=1}^{M} f(\epsilon_m) - M\mathbb{E}[f(\epsilon)] \geq Mt \right) = \Pr\left( e^{\lambda \left( \sum_{m=1}^{M} f(\epsilon_m) - M\mathbb{E}[f(\epsilon)] \right)} \geq e^{\lambda Mt} \right) \leq \mathbb{E}\left[ e^{\lambda \left( \sum_{m=1}^{M} f(\epsilon_m) - M\mathbb{E}[f(\epsilon)] \right)} \right] e^{-\lambda Mt} = (\mathbb{E}\left[ e^{\lambda f(\epsilon_m) - \mathbb{E}[f(\epsilon)]} \right] e^{-t})^M.
\]

According to Lemma 9, let \( \lambda = \frac{M}{\pi^2 L^2} \), we have \( \Pr\left( \frac{1}{M} \sum_{m=1}^{M} f(\epsilon_m) - \mathbb{E}[f(\epsilon)] \geq t \right) \leq e^{-\frac{2Mt^2}{\pi^2 L^2}} \). The other side can apply the same trick. Let \( M = 1 \) we have Inequality (A.11). Thus Theorem 2 and Inequality (A.11) provide the theoretical guarantee for stochastic method for Gaussian variables.

A.6 Proof of Lemma 3

Proof. Since \( g(x) = \frac{1}{1+e^x} \), we have \( g'(x) = g(x)(1-g(x)) \leq \frac{1}{4} \).
\[
|f(\epsilon) - f(\eta)| = |g(h_i(\epsilon)) - g(h_i(\eta))| \leq \frac{1}{4}|h_i(\epsilon) - h_i(\eta)| \leq \frac{1}{4}\|W_i \mathbf{R}\|_2 \|\epsilon - \eta\|_2.
\]

Since \( \tanh(x) = 2g(2x) - 1 \) and \( (1 + e^x)' \leq 1 \), the bound is trivial. \( \square \)
A.7 Update of Message Passing in Chapter 3

Updating $\lambda$

$$
\lambda_{x_{n,t}}^{(i+1)}(x_{n,t-1}) \propto \sum_{x_{n,t}} \lambda^{(i)}(x_{n,t}) \sum_{x_{n',t-1}:(n',n) \in E_{i-1}} \phi_{n,t-1} \prod_{n':(n',n) \in E_{i-1}} \pi^{(i)}_{x_{n,t}}(x_{n,t-1})
$$

$$
\lambda_{x_{n,t}}(x_{n,t-1} = 1) \propto \sum_{x_{n,t}} \lambda(x_{n,t}) \gamma^\beta x_{n,t}^{-\beta}(1 - \gamma)^{x_{n,t}-1}
$$

$$
\lambda_{x_{n,t}}(x_{n,t-1} = 0) \propto \sum_{x_{n,t}} \lambda(x_{n,t}) \sum_{x_{n',t-1}:(n',n) \in E_{i-1}} \left[ 1 - (1 - \alpha)(1 - \beta)^{x_{n',t-1}} \right]^{x_{n,t}-1} \prod_{n':(n',n) \in E_{i-1}} \pi_{x_{n,t}}(x_{n,t-1})
$$

$$
\lambda_{x_{n,t}}^{(i+1)}(x_{n',t-1}) \propto \sum_{x_{n,t}} \lambda^{(i)}(x_{n,t}) \sum_{x_{n,t-1}:(n',n) \in E_{i-1} \neq x_{n',t-1}:n' \in E_{i-1}} \phi_{n,t-1} \prod_{n':(n',n) \in E_{i-1}} \pi^{(i)}_{x_{n,t}}(x_{n,t-1})
$$

Updating $\pi$

$$
\pi_{x_{n,t+1}}^{(i+1)}(x_{n,t}) \propto \prod_{s=1}^{S} \lambda_{y_{n,t,s}}(x_{n,t}) \prod_{n':(n',n) \in E_{i}} \lambda_{x_{n,t+1}}^{(i)}(x_{n,t}) \pi^{(i)}(x_{n,t}) = \frac{BEL^{(i)}(x_{n,t})}{\lambda_{x_{n,t+1}}^{(i)}(x_{n,t})}
$$

$$
\pi_{x_{n,t+1}}^{(i+1)}(x_{n,t}) \propto \prod_{s=1}^{S} \lambda_{y_{n,t,s}}(x_{n,t}) \prod_{n' \in E_{i}} \lambda_{x_{n,t+1}}^{(i)}(x_{n,t}) \pi^{(i)}(x_{n,t}) = \frac{BEL^{(i)}(x_{n,t})}{\lambda_{x_{n,t+1}}^{(i)}(x_{n,t})}
$$

$$
\pi_{x_{n,t},s}^{(i+1)}(x_{n,t}) \propto \prod_{s' \neq s} \lambda_{y_{n,t,s'}}(x_{n,t}) \prod_{n' \in E_{i}} \lambda_{x_{n,t+1}}^{(i)}(x_{n,t}) \pi^{(i)}(x_{n,t}) = \frac{BEL^{(i)}(x_{n,t})}{\lambda_{y_{n,t,s}}^{(i)}(x_{n,t})}
$$

Boundary Conditions

- Root nodes, i.e. $x_{n,0}$. The prior distribution is $p(x_{n,0}) = \pi^{x_{n,0}}(1 - \pi)^{1-x_{n,0}}$. 

• Evidence nodes, i.e. \( y_{n,t,s} \)

\[
\lambda(y_{n,t,s}) = (I_{y_{n,t,s}=0}, I_{y_{n,t,s}=1})
\]

\[
\pi(y_{n,t,s}) = \sum_{x_{n,t}} \phi_{n,t,y|x,s}(y_{n,t,s}|x_{n,t}) \pi_{y_{n,t,s}}(x_{n,t})
\]

\[
BEL(y_{n,t,s}) \propto \lambda(y_{n,t,s}) \pi(y_{n,t,s})
\]

\[
\lambda_{y_{n,t,s}}(x_{n,t}) \propto \sum_{y_{n,t,s}} \lambda(y_{n,t,s}) \phi_{n,t,y|x,s}(y_{n,t,s}|x_{n,t}) = I_{y_{n,t,s}=0}(1 - \theta_{x_{n,t,s}}) + I_{y_{n,t,s}=1}\theta_{x_{n,t,s}}
\]

A.8 On Likelihood computation for 2nd Interpretation on \( \beta_n \) in Chapter 3

Notice when \( \beta_n \) to obtain (3.30), we use the following identity.

\[
\prod_{n=1}^{N} \prod_{t=1}^{T-1} \prod_{n'\in S_{n,t}} (1 - \beta_{n'})^I(R_{n,t}=0) + I(X_{n,t}=0, X_{n,t+1}=0) \beta_{n'}^{I(R_{n,t}=n')} I(n'\in S_{n,t})
\]

\[
= \prod_{n=1}^{N} \prod_{t=1}^{T-1} \prod_{n=1}^{N} \prod_{n'\in S_{n,t}} (1 - \beta_{n'})^I(R_{n,t}=0) + I(X_{n,t}=0, X_{n,t+1}=0) \beta_{n'}^{I(R_{n,t}=n')} I(n'\in S_{n,t})
\]

\[
= \prod_{n=1}^{N} \prod_{t=1}^{T-1} \prod_{n'=1}^{N} \prod_{n=1}^{N} \prod_{n'\in S_{n,t}} (1 - \beta_{n'})^I(R_{n',t}=0) + I(X_{n',t}=0, X_{n',t+1}=0) \beta_{n'}^{I(R_{n',t}=n')} I(n\in S'_{n',t})
\]

A.9 Derivation of Gradient in Chapter 3

A.9.1 \( W.R.T \log Q_{\Psi} \)

we only need to consider the general form of factors, since no matter what parameterization we apply the gradient of parameter is just a summation over the gradient derived from single factor with respect to \( t \) or \( n \).

\[
q(x_{n,t} = 1|\tilde{X}_{n}^{t-1}, y_{n}^{t}) = \sigma(\omega' \tilde{X}_{n}^{t-1} + \nu' y_{n}^{t} + \kappa' z_{n} + b)
\]
Taking the advantage of sigmoid function, we have

\[
\frac{\partial \log q}{\partial \omega} = (x_n^t - q_n^t) \hat{x}_{n}^{t-1}
\]

\[
\frac{\partial \log q}{\partial \nu} = (x_n^t - q_n^t) y_n^t
\]

\[
\frac{\partial \log q}{\partial \kappa} = (x_n^t - q_n^t) z_n
\]

\[
\frac{\partial \log q}{\partial b} = (x_n^t - q_n^t)
\]

If we generalize it to deep SBN, we add a hidden layer.

\[
q(h_{n,t} = 1|\tilde{x}_{t}^{t-1}, y_n^t) = \sigma(\omega^\top \tilde{x}_{t}^{t-1} + \nu^\top y_n^t + \kappa^\top z_n + b)
\]

\[
q(x_n^t|h_{n,t}) = \sigma(\omega_h^\top h_{n,t} + b_h)
\]

where \(\omega, \nu, \kappa\) have become matrix and \(b\) is a vector. In our variational inference framework, we can first sample \(h_{n,t}\) given \(\tilde{x}_{t}^{t-1}, y_n^t\), and then sample \(x_n^t\) given \(h_{n,t}\). This means back-propagation is unnecessary for gradient computation. We only need to compute the gradient \(\omega, \nu, \kappa, b\) similarly as non-deep SBN w.r.t the sampled \(h_{n,t}\), then compute the gradient \(\omega_h, b_h\) in the same manner w.r.t the sampled \(x_n^t\).
A.9.2 W.R.T log $P_\Phi$

To save space, we only provide the results.

$$\frac{\partial \log P_\Phi}{\partial \eta_b}$$

$$= \sum_{n,t} \left\{ \left[ \mathbb{I}_{\{x_{n,t}^{t+1} \neq 0\}} \frac{C_{n,t}p_{00}}{1 - p_{00}} - \mathbb{I}_{\{x_{n,t}^{t+1} = 0\}} C_{n,t} \right] \sigma(z_n^T \eta_b) z_n \right\}$$

$$\frac{\partial \log P_\Phi}{\partial \eta_r}$$

$$= \sum_n (C_{n,1\rightarrow 0} \cdot (1 - \sigma(z_n^T \eta_r)) - C_{n,1\rightarrow 1} \cdot \sigma(z_n^T \eta_r)) z_n$$

$$\frac{\partial \log P_\Phi}{\partial \eta_a}$$

$$= \sum_n \left( \sum_t \mathbb{I}_{\{x_{n,t}^{t+1} \neq 0\}} \frac{p_{00}}{1 - p_{00}} - C_{n,0\rightarrow 0} \right) \sigma(z_n^T \eta_a) z_n$$

where

$$p_{00} = (1 - \sigma(z_n^T \eta_a))(1 - \sigma(z_n^T \eta_b)) C_n^{t-1}$$

$$C_{n,1\rightarrow 0} = \sum_t \mathbb{I}(x_n^t = 1, x_n^{t+1} = 0)$$

$$C_{n,1\rightarrow 1} = \sum_t \mathbb{I}(x_n^t = 1, x_n^{t+1} = 1)$$

$$C_{n,0\rightarrow 0} = \sum_t \mathbb{I}(x_n^t = 0, x_n^{t+1} = 0)$$

In addition, the gradient of parameter for emission distribution is trivial.
A.10 Derivation of Mean-field Method in Chapter 3

Notice that since the mean-filed method is quite complicated, we need more parameters and adopt the new notation system. Define model parameters as $\Phi = \{\alpha, \beta, \gamma, \pi, \theta\}$, hyper-parameters as $\Psi = \{\eta, a, h\}$, where we hidden the index. The posterior distribution of hidden variable is given.

$$p(X, \Phi|Y, \Psi) = \frac{p(Y|X, \Phi)p(X, \Phi|\Psi)}{p(Y|\Psi)}$$

which is intractable for the normalization term $p(Y|\Psi) = \int P(\Phi|\Psi) (\sum_X p(Y|X, \Phi)p(X|\Phi) d\Phi$ being difficult to compute directly. However, for any variational distribution $q(X, \Phi)$, we have

$$p(Y|\Psi) = \frac{p(Y, X, \Phi|\Psi)}{q(X, \Phi)} \frac{q(X, \Phi)}{p(X, \Phi|Y, \Psi)}$$

Thus, we can derive

$$\log p(Y|\Psi) \geq \mathbb{E}_q \left[ \log \frac{p(Y, X, \Phi|\Psi)}{q(X, \Phi)} \right]$$

$$= \mathbb{E}_q [\log p(Y, X, \Phi|\Psi)] - \mathbb{E}_q [\log q(X, \Phi)]$$

$$= \mathbb{E}_q [\log p(Y|X, \Phi)] - D_{KL} (q(X, \Phi)||p(X, \Phi|\Psi))$$

$$= \mathcal{L}(\Xi; \Psi)$$

where $\Xi$ is the parameter set of variational distribution $q$.

Assuming that $q(X, \Phi)$ can be factorized as $q(X)q(\Phi)$, we have the lower bound.

$$\mathcal{L}(\Xi; \Psi) = \mathbb{E}_{q(\Phi)} \left[ \log \frac{p(\Phi|\Psi)}{q(\Phi)} + \mathbb{E}_{q(X)} \left[ \log \frac{p(Y|X, \Phi)}{q(X)} \right] \right]$$
On taking functional derivatives of $\mathcal{L}$ with respect to $q(\Phi)$, we obtain

$$
\log q(\Phi) = \log p(\Phi|\Psi) + \mathbb{E}_{q(X)} [\log p(Y, X|\Phi)] + c
$$

$$
= \log p(\pi|\alpha) + \log p(\alpha|\eta_a) + \log p(\beta|\eta_b)
$$

$$
+ \log p(\gamma|\eta_r) + \log p(\theta|h) + \mathbb{E}_{q(x_1)} [\log p(X_1|\pi)]
$$

$$
+ \mathbb{E}_{q(x_2)} [\log p(X_2, \alpha, \beta, \gamma)]
$$

$$
+ \mathbb{E}_{q(X)} [\log p(Y|X, \theta)] + c
$$

A.10.1 Variational Bayesian E-step

The variational posterior over the parameters can be factorized without further approximation $q(\Phi) = q(\pi)q(\alpha)q(\beta)q(\gamma)q(\theta)$. Taking functional derivatives with re-
pect to $q(\cdot)$ and equating them to zero, we obtain

$q(\pi) = \Beta \left( a_\pi + \mathbb{E}_{q(X,1)} \left[ \sum_{n=1}^{N} \mathbf{I}(X_{n,1} = 1) \right] \right),$

$b_\pi + \mathbb{E}_{q(X,1)} \left[ \sum_{n=1}^{N} \mathbf{I}(X_{n,1} = 0) \right])$

$q(\theta_{1,s}) = \Beta \left( h_{1,s} + \mathbb{E}_{q(X)} \left[ \sum_{n,t} \mathbf{I}(Y_{n,t,s} = 1, X_{n,t} = 1) \right] \right),$

$h_{1,s} + \mathbb{E}_{q(X)} \left[ \sum_{n,t} \mathbf{I}(Y_{n,t,i} = 0, X_{n,1} = 1) \right])$

$q(\theta_{0,s}) = \Beta \left( h_{0,s} + \mathbb{E}_{q(X)} \left[ \sum_{n,t} \mathbf{I}(Y_{n,t,s} = 1, X_{n,t} = 0) \right] \right),$

$h_{0,s} + \mathbb{E}_{q(X)} \left[ \sum_{n,t} \mathbf{I}(Y_{n,t,i} = 0, X_{n,1} = 0) \right])$

$q(\gamma_n) = \Beta \left( e^{Z_n^{T} \eta_{r,1}} + \mathbb{E}_{q(X_n)} \left[ \sum_{t} \mathbf{I}(X_{n,t} = 1, X_{n,t+1} = 0) \right] \right),$

$e^{Z_n^{T} \eta_{r,2}} + \mathbb{E}_{q(X_n)} \left[ \sum_{t} \mathbf{I}(X_{n,t} = 1, X_{n,t+1} = 1) \right])$
However, for $\alpha_n, \beta_n$ do not belong to conjugate exponential family, we have to explicitly set the variational parameters, i.e $q(\alpha_n) = \text{Beta}(a_{1,n}, a_{2,n}), q(\beta_n) = \text{Beta}(b_{1,n}, b_{2,n})$.

$$\frac{\partial \mathcal{L}}{\partial a_{1,n}} = \int [\ln \alpha_n - \psi(a_{1,n}) + \psi(a_{1,n} + a_{2,n})] q(\alpha)q(\beta)q(X)$$

$$\times \left\{ \sum_{n=1}^{N} \sum_{t=1}^{T-1} \left[ I_{X_{n,t,t+1}^0} \ln (1 - (1 - \alpha_n)(1 - \beta_n)^{C_{n,t}}) + I_{X_{n,t,t+1}^0} \ln (1 - \alpha_n)(1 - \beta_n)^{C_{n,t}} \right] \right\} \, d\alpha d\beta dX$$

$$+ \int [\ln \alpha_n - \psi(a_{1,n}) + \psi(a_{1,n} + a_{2,n})]$$

$$\times q(\alpha_n) \ln p(\alpha_n|\eta_{a,Z_n}) \, d\alpha_n$$

$$- \int [\ln \alpha_n - \psi(a_{1,n}) + \psi(a_{1,n} + a_{2,n})] q(\alpha_n)[1 + \ln q(\alpha_n)] \, d\alpha_n$$

The gradients for $a_{2,n}, b_{1,n}, b_{2,n}$ are similar. We need Monte Carlo (MC) integration for this term and apply gradient descent based algorithm to find the optima.

$$\mathcal{L} = \int q(\Phi) \ln \frac{p(\Phi|\Psi)}{q(\Phi)} \, d\Phi$$

$$+ \int q(\Phi) \int q(X) \ln \frac{p(Y|X, \theta)p(X|\alpha, \beta, \gamma, \pi)}{q(X)} \, dX \, d\Phi$$

On taking functional derivative with respect to $q(X)$, we obtain

$$\ln q(X) = \mathbb{E}_{q(\Phi)} [\ln p(Y|X, \theta)p(X|\alpha, \beta, \gamma, \pi)] - \ln Z(Y)$$

Furthermore, we assume $q(X) = \prod_n \prod_t q(X_{n,t})$, where $q(X_{n,t}) = \text{Bernoulli}(\phi_{n,t})$. Easy to verify this part is conjugate thus inducing the similar result as before (except including the expectation on $q(\Phi)$).
A.10.2 Variational Bayesian M-step

\[
\mathcal{L} = \int q(\Phi) \ln p(\Phi|\Psi) d\Phi - \mathbb{E}_{q(\Phi)}[\ln q(\Phi)] \\
+ \mathbb{E}_{q(X)q(\Phi)} \left[ \log \frac{p(Y, X|\Phi)}{q(X)} \right]
\]

We need to perform \( \frac{\partial \mathcal{L}}{\partial \Phi} = 0 \) to find the optimal hyperparameter given current \( \Xi^* \) or \( q(X, \Phi|\Xi^*) \).

\[
\frac{\partial \mathcal{L}}{\partial h_{1,s}} = \int q(\theta_{1,s}^*)[\ln(\theta_{1,s}) - \psi(h_{1,s}) + \psi(h_{1,s} + h_{2,s})] d\theta_{1,s} \\
\Rightarrow \psi(h_{1,s}^*) - \psi(h_{1,s}^* + h_{2,s}^*) = \psi(h_{1,s}) - \psi(h_{1,s} + h_{2,s})
\]

\[
\frac{\partial \mathcal{L}}{\partial h_{2,s}} = 0 \\
\Rightarrow \psi(h_{2,s}^*) - \psi(h_{1,s}^* + h_{2,s}^*) = \psi(h_{2,s}) - \psi(h_{1,s} + h_{2,s})
\]

This means \( h_{1,s} = h_{1,s}^* \) and \( h_{2,s} = h_{2,s}^* \). The results for \( h_{0,s}, h_{0,s}, a_\pi, b_\pi \) are similar.

\[
0 = \frac{\partial \mathcal{L}}{\partial \eta_{r,1}} = \int q(\gamma) \sum_{n=1}^{N} \left[ \ln \gamma_{n} - \psi \left( e^{Z_{n}^{T} \eta_{r,1}} \right) + \psi \left( e^{Z_{n}^{T} \eta_{r,1} + e^{Z_{n}^{T} \eta_{r,2}}} \right) \right] e^{Z_{n}^{T} \eta_{r,1}} Z_{n} d\gamma \\
= \sum_{n=1}^{N} \left[ \psi(r_{1,n}^*) - \psi(r_{1,n}^* + r_{2,n}^*) - \psi \left( e^{Z_{n}^{T} \eta_{r,1}} \right) + \psi \left( e^{Z_{n}^{T} \eta_{r,1} + e^{Z_{n}^{T} \eta_{r,2}}} \right) \right] Z_{n} \\
\Rightarrow \psi(r_{1,n}^*) - \psi(r_{1,n}^* + r_{2,n}^*) = \psi \left( e^{Z_{n}^{T} \eta_{r,1}} \right) - \psi \left( e^{Z_{n}^{T} \eta_{r,1} + e^{Z_{n}^{T} \eta_{r,2}}} \right)
\]

\[
0 = \frac{\partial \mathcal{L}}{\partial \eta_{r,2}} \Rightarrow \psi(r_{2,n}^*) - \psi(r_{1,n}^* + r_{2,n}^*) = \psi \left( e^{Z_{n}^{T} \eta_{r,2}} \right) - \psi \left( e^{Z_{n}^{T} \eta_{r,1} + e^{Z_{n}^{T} \eta_{r,2}}} \right)
\]

Setting the gradient equals to zero is sufficient to recover \( \eta \), since we can find one possible solution \( e^{Z_{n}^{T} \eta_{r,1}} = r_{1,n}^* \). Then we only need to solve a linear system \( Z_{\eta_{r,1}} = (\ln r_{1,1}^*, \ldots, \ln r_{1,n}^*)^T \), where \( Z = [Z_{1}, \ldots, Z_{n}]^T \). The results for \( \eta_{a,}, \eta_{b,} \) are similar.
Bibliography


Murphy, K. P. (2012), Machine learning: a probabilistic perspective.


135


Biography

Kai Fan was born on Nov 10, 1988 in Xingping, Shaanxi, China. He received a B.S. in Mathematics from Peking University in July 2010, an M.S. in Computer Science from Peking University in 2013. He graduated with a Doctor of Philosophy in Dec 2017 from Duke University.

He served as the program committee member for Neural Information Processing Systems (NIPS 2016, 2017), International Conference on Machine Learning (ICML 2016, 2017), Association for the Advancement of Artificial Intelligence (AAAI 2018), and International Conference on Artificial Intelligence and Statistics (AISTATS 2018).

He had two internships at Microsoft Research Redmond and New England, and he will join Alibaba Institute of Data Science and Technologies as a researcher at San Francisco Bay Area.