Approximate Inference

Zhiwei Han
Faculty of Electrical Engineering and Information Technology
Technical University of Munich
Arcisstr. 21, Munich, 80333
Email: hanzw356255531@icloud.com

***Abstract*—This is a seminar work from Chair of Medientechnik, Technical University of Munich. Most of this work is based on the deep learning book from MIT. [1] In this seminar work, an overview of approximation inference and how this technique is applied to solve intractable problem in inference is given. We begin with the introduction of the intractable problems raised in the inference of a probabilistic model and then present a new objective function for the optimization problem. After that, we show the great simplification of the optimization problem with the new objective function instead of log-likelihood with mathematical proof. At the end, a CNN-based variational auto encoder is presented to show that with this technique a deep generative model, whose latent variables are centered isotropic multivariate Gaussian distributed, can be train in a feasible time.**

***Index Terms*—Approximate Inference, Variational Inference, Deep Learning, Generative Model, Auto Encoder.**

I. Introduction

In many statistical learning problems, especially training of a generative model, inference can be thought as a part of likelihood maximization problem. For some simple models like Restricted Boltzmann Machines (RBM) and probabilistic PCA, inference can be simply done by computing the posterior and taking the expectation over it [1]. Those computations are critical process since they are the basis of the training step afterwards. However, within some intractable graphical models have multiple layers like Deep Belief Network, exact inference will cost an exponential amount of time. Consequently, an exact inference is not necessary if we can find an approach to do it with enough accuracy in an acceptable time amount.

In the context of deep learning, the problem setting can be organized in a more specific way. We assume that we have a set of visible variables $\boldsymbol{v}$, which can be seen as the input of a one layer RBM, and a set of latent variables $\boldsymbol{h}$, the corresponding output. The goal of an exact inference on such model is to compute the posterior $p(\boldsymbol{h}|\boldsymbol{v})$ analytically. Unfortunately, due to several interactions between latent variables in a model like structured graphical model, it's definitely inefficient, if we still calculate the posterior in the traditional way. In other words, when the latent variables are not independent anymore, the computation of posterior $p(\boldsymbol{h}|\boldsymbol{v})$ can be very hard.

One possibility how we deal with these kind of intractable inference problems is variational inference. Instead of trivially integral over the visible variables and latent variables, we are going to find a distribution to approximate the posterior as much as possible and new objective function (a lower bound of log likelihood function with respect to this approximate posterior). Finally, finish the inference by taking the expectation over this distribution. For a perfect approximation q of posterior $p(\boldsymbol{h}|\boldsymbol{v})$, the new objective function is exactly the log likelihood function.

The goal of this seminar work is to present an overview about the approximate inference and effective method to confront these issues in term of statistics. In the second section, we show the basic concept of inference and the main problem through an intuitive example and introduce several techniques for solving intractable inference problem and learning with structured probabilistic models. As the learned approximate posterior inference model can be used in a huge amount of tasks, we present a $variational\ auto$-$encoder$ to show that a CNN-based generative model can learn efficient and benefit from the techniques introduced above.

II. Background

A. Inference

In machine learning community, discriminant method and generative method are two main approaches to solve specific learning tasks with large data sets, their models are therefore named as discriminant models (SVM, Logistic Regression) and generative models (GMM, HMM), respectively.

The goal of discriminant models is for example classification, in other words the discriminant model learns the **conditional probability distribution** $p(\boldsymbol{c}|\boldsymbol{o})$, which is the conditional probability of class vector $\boldsymbol{c}$ given observation vector $\boldsymbol{o}$. The learned model should be able to classify the input data into the exact class the input data belongs to, according to some predefined criterias (e.g. the conditional probability is higher than a threshold). While the generative model does inference, which means to learn the **joint distribution** $p(\boldsymbol{c},\boldsymbol{o})$ of the given data sets. Since the generative model knows the joint distribution of the data sets, so conditional probability can be easily derived by dividing with prior according to Bayes rule,

$$p(\boldsymbol{c}|\boldsymbol{o}) = \frac{p(\boldsymbol{c},\boldsymbol{o})}{p(\boldsymbol{o})} \tag{1}$$

From the above example, we can see that inference is a generalization form of classification task. Therefore, generative model has better representation ability and a faster convergence. The drawbacks is that the training of generative model is more computational complex. [2]

For convenience, we define here the problem setting for the training task in rest of this seminar work. Our inference problems are built so that, the models are consisting of visible variables $\boldsymbol{v}$ and latent variables $\boldsymbol{h}$. And we want to find a proper model parameter $\boldsymbol{\theta}^*$, which maximize likelihood over given dataset.

Since in this seminar work we are more interested in inference, we will focus on the application of inference, especially approximate inference in rest of this work. Consider the standard training procedure of a generative model, which has visible variables $\boldsymbol{v}$, latent variables $\boldsymbol{h}$ and model parameter $\boldsymbol{\theta}$, as first step we need to compute the likelihood by marginalize over the posterior $p(\boldsymbol{h} \mid \boldsymbol{\theta})$ as follows. It is equal to take the expectation of likelihood give latent variable $\boldsymbol{h}$ with respect to posterior.

$$L(\boldsymbol{v} \mid \boldsymbol{\theta}) = \int_{\boldsymbol{h}} p(\boldsymbol{v} \mid \boldsymbol{h}, \boldsymbol{\theta}) p(\boldsymbol{h} \mid \boldsymbol{\theta}) \mathrm{d}\boldsymbol{h}. \tag{2}$$

Simple graphical models remain the computation of posterior $p(\boldsymbol{h} \mid \boldsymbol{\theta})$ still solvable e.g. RBM (see fig. 1[1]), so a standard EM algorithm could be used to solve these problems. Unfortunately, most applicable graphical model usually have interactions between their latent variables and thus also have intractable posterior distribution (see fig. 2), it means that the v-structure and the intractable edge between the latent variables $p(\boldsymbol{h} \mid \boldsymbol{\theta})$ make the posterior distribution intractable. Consequently, the computational expense rises dramatically and it is almost impossible to finish the computations of posterior in a feasible training time. However, with approximation inference, we are given a powerful weapon to solve this problem.

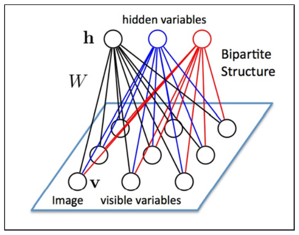


Fig. 1. RBM: Every latent variable is independent to each other since there is no connection between them. The posteriors are through factorization solvable.

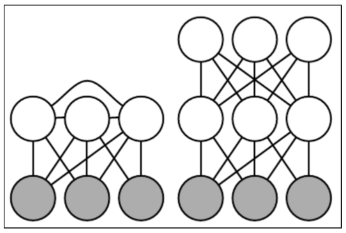


Fig. 2. Models with v-structure and edge between latent variables: The posterior distribution are intractable because of the interaction between latent variables[1]

B. MAP Inference

As shown in (2), when training a probabilistic model, we are always interested in computing the data distribution by integral over a set latent variables, namely inference. However, integral over all latent variables could be computationally expensive and should be therefore avoided in the real implementation. A solution to this problem is to use the most likely latent variable $\boldsymbol{h}^*$ and ignore other unlikely latent variables, rather than integral over all possible latent variables. Because in practice, for most $\boldsymbol{h}$, $P(\boldsymbol{v}|\boldsymbol{h})$ will be nearly zero, and hence contribute almost nothing to the calculation of likelihood $p(\boldsymbol{v})$. [1][3]

[1]Salakhutdinov. R., Deep Learning lecture slides, CMU, US

Mathematically, it is equal to an optimization problem as follows,

$$\boldsymbol{h}^* = \arg\max_{\boldsymbol{h}} p(\boldsymbol{h} \mid \boldsymbol{\theta}), \tag{3}$$

and approximate (2) as

$$\begin{aligned} L(\boldsymbol{v} \mid \boldsymbol{\theta}) &= \int_{\boldsymbol{h}} p(\boldsymbol{v} \mid \boldsymbol{h}, \boldsymbol{\theta}) p(\boldsymbol{h} \mid \boldsymbol{\theta}) \mathrm{d}\boldsymbol{h} \\ &= p(\boldsymbol{v} \mid \boldsymbol{h}^*, \boldsymbol{\theta}) p(\boldsymbol{h}^* \mid \boldsymbol{\theta}) \end{aligned} \tag{4}$$

This method is known as maximum a posteriori inference (MAP).

C. EM Algorithm

Expectation Maximization (EM) [4] algorithm is a standard iterative learning algorithm, which is based on maximum likelihood estimation and especially designed for models with latent variables.

EM algorithm includes two steps and runs until the predefined convergent criteria is satisfied,

1) Initialization: Initialize the model parameters $\boldsymbol{\theta}_0$

2) Expectation Step: Compute the objective function (sum of ELBO on all data index) according to (3),

$$\sum \mathcal{L}(\boldsymbol{v}^{(i)}, \boldsymbol{\theta}, q). \tag{5}$$

Note, set $q(\boldsymbol{h}^{(i)} \mid \boldsymbol{v}^{(i)})$ for all the index of the data set we need to train on and remain distribution $q(\boldsymbol{h} \mid \boldsymbol{v})$ always equal to $p(\boldsymbol{h} \mid \boldsymbol{v}, \boldsymbol{\theta}_0)$ while updating $p(\boldsymbol{h} \mid \boldsymbol{v}, \boldsymbol{\theta}_t)$ with $\boldsymbol{\theta_t}$, where t is the number of current iteration.

3) Maximization Step: Maximize the objective function (sum of ELBO on all data index) over model parameters $\boldsymbol{\theta}_t$ with arbitrary optimization algorithm.

4) Repeat 2), 3) until converge:

III. Variational Inference

A. Objective function

Many sample based training problems which make use of observations can be reconstructed as optimization problems and they maximize the log-likelihood function of the given datasets. [3][5] Approximate Inference algorithm will then simplify the underlying optimization problems by using the approximation of posteriors.

While the intractability between latent variables make the likelihood computation much more difficult (because of integral), instead of directly calculating and optimizing the log-likelihood, we introduce a new objective function here, which is easy to compute and optimize if a distribution $q(\boldsymbol{h} \mid \boldsymbol{v})$ could be found. This means that we need to find a new distribution $q(\boldsymbol{h} \mid \boldsymbol{v})$ which can make a good approximation of the posterior. Or in other words, it is a function, which takes a vector $\boldsymbol{v}$ as visible variable and give us a distribution over the latent variable $\boldsymbol{h}$ that are likely to produce the visible variable $\boldsymbol{v}$. Ideally, the distribution of $\boldsymbol{h}$ under $q(\boldsymbol{h} \mid \boldsymbol{v})$ is much simpler than the one under the posterior $p(\boldsymbol{h} \mid \boldsymbol{v})$. This trick makes the computation of $\mathbb{E}_{\boldsymbol{h}\sim q}[p(\boldsymbol{v} \mid \boldsymbol{h})]$ significantly much easier.

We begin with the derivation of the new objective function from the definition of KL Divergence between the distribution $q(\boldsymbol{h} \mid \boldsymbol{v})$ and the posterior $p(\boldsymbol{h} \mid \boldsymbol{v})$ for some arbitrary q, then we get the $\log p(\boldsymbol{v} \mid \boldsymbol{h})$ and $\log p(\boldsymbol{v})$ term after applying Bayes rule to $\log p(\boldsymbol{h} \mid \boldsymbol{v})$. Here we can take the $\log p(\boldsymbol{v})$ term out of the expectation since it has no dependency with $\boldsymbol{h}$.

$$\begin{aligned}
&D_{KL}\Big[q(\boldsymbol{h} \mid \boldsymbol{v}) \mid\mid p(\boldsymbol{h} \mid \boldsymbol{v})\Big] \\
&= \mathbb{E}_{\boldsymbol{h}\sim q}\Big[\log \frac{q(\boldsymbol{h} \mid \boldsymbol{v})}{p(\boldsymbol{h} \mid \boldsymbol{v})}\Big] \\
&= \mathbb{E}_{\boldsymbol{h}\sim q}\Big[\log q(\boldsymbol{h} \mid \boldsymbol{v}) - \log \frac{p(\boldsymbol{h}, \boldsymbol{v})}{p(\boldsymbol{v})}\Big] \\
&= \mathbb{E}_{\boldsymbol{h}\sim q}\Big[\log q(\boldsymbol{h} \mid \boldsymbol{v}) - \log p(\boldsymbol{v}, \boldsymbol{h}) + \log p(\boldsymbol{v})\Big] \\
&= \mathbb{E}_{\boldsymbol{h}\sim q}\Big[\log q(\boldsymbol{h} \mid \boldsymbol{v}) - \log p(\boldsymbol{v}, \boldsymbol{h})\Big] + \log p(\boldsymbol{v})
\end{aligned} \tag{6}$$

After reforming the expectation term into KL divergence and rearranging the formula, it yields a lower bound for the log-likelihood function (the new objective function),

$$\begin{aligned}
J &= \log p(\boldsymbol{v}) - D_{KL}\Big[q(\boldsymbol{h} \mid \boldsymbol{v}) \mid\mid p(\boldsymbol{h} \mid \boldsymbol{v})\Big] \\
&= -\mathbb{E}_{\boldsymbol{h}\sim q}\Big[\log q(\boldsymbol{h} \mid \boldsymbol{v}) - \log p(\boldsymbol{v}, \boldsymbol{h})\Big] \\
&= \mathbb{E}_{\boldsymbol{h}\sim q}\Big[\log q(\boldsymbol{h}, \boldsymbol{v})\Big] + H(q)
\end{aligned} \tag{7}$$

where $H(q)$ is the information entropy of distribution q.

The left part of the objectvie function has the log-likelihood $p(\boldsymbol{v})$ term we want to maximize plus an error term, which measures the difference of the distribution $q(\boldsymbol{h} \mid \boldsymbol{v})$ and the posterior $p(\boldsymbol{h} \mid \boldsymbol{v})$. Hence, the difference between our new objective function and the log-likelihood is decided by the KL divergence and these two are equal if and only if distribution q is exactly the same as $p(\boldsymbol{h} \mid \boldsymbol{v})$. Because the KL divergence is always non-negative and this new objective function is therefore smaller than or at most equal to the real log-likelihood. In this case, the new objective function is defined as a lower bound $\mathcal{L}(\boldsymbol{v}, \boldsymbol{\theta}, q)$ of log-likelihood function and this lower bound is called the *evidence lower bound* (ELBO).

It is not hard to see, our new objective function is much easier to compute for some appropriate choice of distribution q. For any distribution q, our objective function is guaranteed to be the lower bound of log-likelihood and with a better approximation of posterior the lower bound will be closer to the real log-likelihood.

B. Core Idea

As shown in previous subsection, we can transfer the inference to a new optimization problem that we should find a proper distribution q that maximize our objective function derived in the previous subsection.

In other words, the original intractable learning problem, which explicitly maximize the log-likelihood, can be thought of as a new procedure that is less computational expensive by using an approximated version of posterior from a restricted search family, which is imperfect approximation and may not exactly maximize $\mathcal{L}$ but can improve it with a significantly amount.

To summarize, the core idea behind variational inference is that we can maximize the objective function over a restricted space of distribution q. This space should be chosen so that it makes the posterior computation efficient but not necessarily exact.

C. Structure of Approximated Posterior

The critical task of variational inference is to design a reparameteric distribution q such that it can make a good approximation of posterior q while still remains $\mathbb{E}_{\boldsymbol{h}\sim q}[p(\boldsymbol{v} \mid \boldsymbol{h})]$ feasible to compute. It could be very difficult in terms of reparameterization tricks, because an irrational reparameterization would make the distribution q either a bad approximation to posterior or remain $\mathbb{E}_{\boldsymbol{h}\sim q}[p(\boldsymbol{v} \mid \boldsymbol{h})]$ unsolvable.

In this subsection we provide two efficient approaches to form a reasonable distribution q:

1) Mean Field Approach: The idea of Mean Field Approach is to compute the distribution q given $\boldsymbol{v}$ by multiplying all the conditional probabilistic of single latent variable together.

$$q(\boldsymbol{h} \mid \boldsymbol{v}) = \prod q(h_i \mid \boldsymbol{v}) \tag{8}$$

This approach is based on the assumption that all the relationships between the latent variables have been removed. Hence, we can impose the restriction that distribution q is a factorial distribution. With this technique, the engineer only need to design how does distribution q factorize rather than guess an accurate approximation distribution q, which is very likely to posterior. It helps to significantly reduce the work of algorithm designer.

2) Deep Neural Network Approach: A deep neural network is built in this case as a part of the distribution approximator, which takes the visible variable $\boldsymbol{v}$ as the input and output the latent variable $\boldsymbol{h}$ with respect to some distribution q (e.g. Gaussian Distribution).

The advantage is that this approach is a purely model-free approach and the designer almost doesn't need to concern about any structure issues. Therefore, it is also the most state-of-the-art approach in variational inference. With this end-to-end solution, the algorithm is even able to do the inference and job at the same time using backpropagation. At last, we give a variational auto-encoder example to show that a CNN-based generative model with the technique of approximation inference can efficiently overcome the difficulty in both inference and learning problem.

D. Variational Auto-Encoder

In this subsection, we give an example of variational auto-encoder (VAE) presented in [6]. In this example, a CNN is used for the probabilistic encoder $q_{\boldsymbol{\theta},\phi}(\boldsymbol{h} \mid \boldsymbol{v})$, where $\boldsymbol{\theta}, \boldsymbol{\phi}$ are the model parameters (in our case $\boldsymbol{\theta}$ is the weights and bias of a network and $\boldsymbol{\phi}$ is the variance of latent variables) and they are updated jointly with AEVB algorithm derived also in [6].

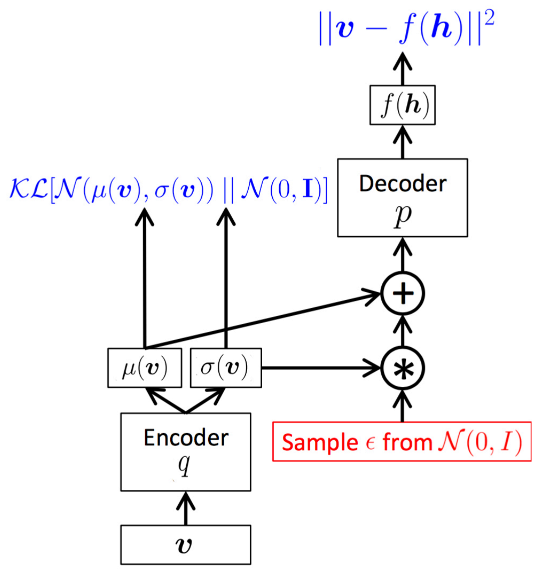
We assume the prior over the latent variables is centered isotropic multivariate Gaussian distribution $p_{\boldsymbol{\theta}}(\boldsymbol{h}) = \mathcal{N}(\boldsymbol{z}; \mathbf{0}, \mathbf{I})$ and likelihood of visible variables $p_{\boldsymbol{\theta}}(\boldsymbol{v} \mid \boldsymbol{h})$ (decoder) given latent variables $\boldsymbol{h}$ is also multivariate Gaussian distribution, which is modelled as a reverse convolutional neural network. Since we have much freedom to design an approximated posterior $q_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{h} \mid \boldsymbol{v})$, so we just assume the real posterior takes a multivariate Gaussian form.

More specifically,

$$\log q_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{h} \mid \boldsymbol{v}^{(\boldsymbol{i})}) = \log \mathcal{N}(\boldsymbol{h}; \boldsymbol{\mu}^{(i)}, {\boldsymbol{\sigma}^{(i)}}^2 \mathbf{I}), \tag{9}$$

where the mean and standard deviation of the approximated posterior $q_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{h} \mid \boldsymbol{v})$, $\boldsymbol{\mu}$, $\boldsymbol{\sigma}$ are the outputs of the encoder (CNN), which maps the visible variable $\boldsymbol{v}$ to the latent variable $\boldsymbol{h}$ nonlinearly with the model parameter $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$.

A training-time variational auto-encoder implementation can be described in the following figure.



$||\boldsymbol{v} - f(\boldsymbol{h})||^2$
$f(\boldsymbol{h})$
$\mathcal{KL}[\mathcal{N}(\mu(\boldsymbol{v}), \sigma(\boldsymbol{v})) \,||\, \mathcal{N}(0, \mathbf{I})]$
Decoder p
$\mu(\boldsymbol{v})$
$\sigma(\boldsymbol{v})$
Encoder q
Sample ϵ from $\mathcal{N}(0, I)$
$\boldsymbol{v}$

Fig. 3. VAE: $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are the mean and variance of the approximated posterior. The red part is "reparameterization trick" and the blue part is the loss function of the model.[3]

We sample from the approximated posterior $\boldsymbol{h}^{(i)} \sim q_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{h} \mid \boldsymbol{v})$ using

$$\boldsymbol{h}^{(i)} = \boldsymbol{\mu}^{(i)} + \boldsymbol{\sigma}^{(i)} \otimes \boldsymbol{\epsilon}^{(i)}, \tag{10}$$

where $\boldsymbol{\epsilon}^{(i)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\otimes$ means element-wise multiplication. Since in this model the prior and the likelihood are both Gaussian, we can therefore use the objective function derived in this paper [6], where the KL divergence can be computed without estimation, and run the standard backpropagation algorithm to update the model parameter $\boldsymbol{\theta}$, $\boldsymbol{\phi}$ jointly.

IV. Experiment

As experiment of this seminar work we reimplement the variational auto-encoder introduced in the paper [6] from the existing tensorflow[2] based framework[3] includes a generative model and a variational approximation model on the dataset of MNIST, which is finally able to randomly generate MNIST digits from the learned data manifold after training procedure.

In this implementation, the CNN-based generative model (encoder) and variational approximation model (decoder) are trained with respect to the loss function in Fig. 3, where the described encoder and decoder have an equal number of hidden units. After training, the generative model and variational approximation model are separated. Then the visualization of the learned dataset manifold is obtained by sampling the reconstructed images according to a two-dimensional Gaussian noise.

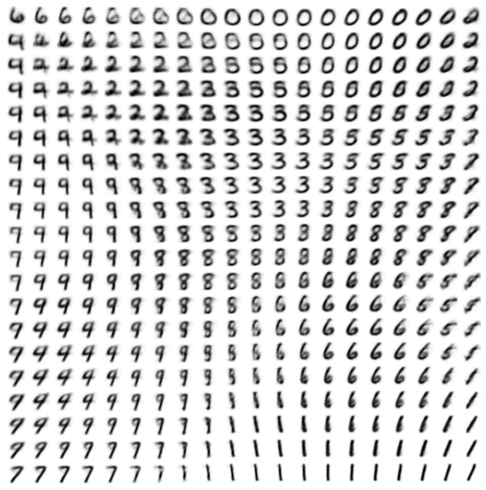


Fig. 4. Visualization of Learned MNIST dataset manifold for the generative model with two-dimensional latent space using AEVB [6]

V. Conclusion

From this seminar work, it is not hard to see that for intractable inference and learning problem, directly applying optimization method over the log-likelihood could be computationally expensive and time consuming. Approximate Inference contributes thus a approximated form of posterior that compute a new objective function (a lower bound of log-likelihood). Finally, with this new objective function we are able to solve intractable inference problems properly. Furthermore, a deep generative model can even merge the inference and learning in one step.

[2]M. A. et al., TensorFlow: Large-scale machine learning on heterogeneous systems, 2015, software available from tensorflow.org. [Online]. Available: http://tensorflow.org/

[3]https://github.com/ikostrikov/TensorFlow-VAE-GAN-DRAW

References

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[2] A. Y. Ng and M. I. Jordan, “On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes,” *Advances in neural information processing systems*, vol. 2, pp. 841–848, 2002.

[3] C. Doersch, “Tutorial on variational autoencoders,” *arXiv preprint arXiv:1606.05908*, 2016.

[4] T. K. Moon, “The expectation-maximization algorithm,” *IEEE Signal processing magazine*, vol. 13, no. 6, pp. 47–60, 1996.

[5] Y. Anzai, *Pattern recognition and machine learning*. Elsevier, 2012.

[6] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” *arXiv preprint arXiv:1312.6114*, 2013.