Project Report - Variational Inference

Shubham Gupta
Department of Computer Science and Automation
Indian Institute of Science
Bangalore, India 560012
shubham.gupta@csa.iisc.ernet.in

Abstract

Variational Inference is an approximate inference technique. Since the problem of probabilistic inference is NP hard in general, techniques like variational inference have found application in many problems involving probability models. In this project we explore the basics of variational inference by means of toy examples. We also implement variational autoencoder as an example application of variational inference to a non-trivial problem on large scale.

1 Introduction

The GitHub repository which contains the code and other resources for this project can be found at:

https://github.com/sh-gupta/VariationalInference

Let us suppose that we are given a probabilistic graphical model. Inference refers to the problem of finding a probability distribution over one or more random variables given the value of some other random variables. If we denote the *observed* random variable by x and *unobserved* random variable by z, then finding $p(z|x)$ is an inference problem. Exact inference, i.e. finding $p(z|x)$ exactly is a NP hard problem for general graphical models. Hence one is left with two choices:

1. Impose structural constraints on graphical model so that inference becomes tractible
2. Switch to approximate inference instead of exact inference

Variational inference [1][2] is a technique which falls in the second category. The aim is to find a *good* approximation to $p(z|x)$.

By using Bayes' theorem we get:

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)} \tag{1}$$

Where $p(x)$ can be calculated using total probability theorem as:

$$p(x) = \int_z p(x,z)dz \tag{2}$$

The integral in equation 2 is intractable in most cases mostly because of the high dimensionality of z. The key idea behind variational inference is to turn an inference problem into an optimization problem. Let $q(z;\theta)$ be probability distribution parameterized by θ then the goal is to find the optimal parameters θ^* such that $KL(q||p)$ is minimized. That is:

$$\theta^* = arg\min_\theta KL[q(z;\theta)||p(z|x)] \tag{3}$$

In equation 3, we use the KL divergence to calculate the objective function. This is currently the most widely used choice although one can use other measures like α-divergence as well [3].

2 Evidence Lower Bound Objective (ELBO)

The optimization problem posed in equation 3 can not be solved in that form since it requires one to evaluate $p(z|x)$ which is not known. One can arrive at an upper bound to the objective given in equation 3 and minimize that upper bound as follows:

$$\begin{aligned}
KL[q(z;\theta)||p(z|x)] &= \mathrm{E}_{z\sim q}[\log q(z;\theta) - \log p(z|x)] \\
&= \mathrm{E}_{z\sim q}[\log q(z;\theta) - \log p(x,z) + \log p(x,z) - \log p(z|x)] \\
&= \mathrm{E}_{z\sim q}[\log q(z;\theta) - \log p(x,z)] + \log p(x) \\
\Rightarrow \log p(x) &= \mathrm{E}_{z\sim q}[\log p(x,z) - \log q(z;\theta)] + KL[q(z;\theta)||p(z|x)] \\
&\geq \mathrm{E}_{z\sim q}[\log p(x,z) - \log q(z;\theta)] \qquad (4)
\end{aligned}$$

Equation 4 follows from the fact that KL divergence is always positive. The quantity on the right hand side of equation 4 is known as Evidence Lower Bound Objective or $ELBO$ since it is a lower bound on the log probability of observed data (evidence). Thus $ELBO(q) = \mathrm{E}_{z\sim q}[\log p(x,z) - \log q(z;\theta)]$. Since $\log p(x)$ is not dependent on z, maximizing $ELBO(q)$ with respect to θ results in minimizing $KL[q(z;\theta)||p(z|x)]$, which is what was desired in the first place.

$ELBO(q)$ is sometimes also written in a different form as follows:

$$\begin{aligned}
ELBO(q) &= \mathrm{E}_{z\sim q}[\log p(x|z) + \log p(z) - \log q(z;\theta)] \\
&= \mathrm{E}_{z\sim q}[\log p(x|z)] - KL[q(z;\theta)||p(z)] \qquad (5)
\end{aligned}$$

Equation 5 reveals an important insight. Maximizing $ELBO(q)$ leads to the setting of parameters θ where $\mathrm{E}_{z\sim q}[\log p(x|z)]$ is high, while forcing $KL[q(z;\theta)||p(z)]$ to be small. This is the general trade-off which is seen in Bayes' theorem. The posterior should explain the observed data while remaining close to the prior.

3 Experiment 1: Univariate Gaussian

We are given a set $\mathcal{D} = \{x^{(1)}, x^{(2)}, ..., x^{(N)}\}$ where each $x^{(i)} \in \mathbb{R}$. The task is to model this data using a univariate Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$. One can adopt a maximum likelihood approach to find point estimates for μ and σ^2. In the Bayesian setting the task is to determine $p(\mu, \sigma^2|\mathcal{D})$. For pedagogical purpose, we adopt a variational inference approach for finding μ and σ^2.

Let $p(\mu)$ and $p(\sigma^2)$ be the priors on μ and σ^2 respectively. Following the variational inference recipe we approximate $p(\mu, \sigma^2|\mathcal{D})$ by $q(\mu, \sigma^2;\theta)$. The **mean field approach** involves the assumption that $q(\mu, \sigma^2;\theta)$ can be factorized as: $q(\mu, \sigma^2;\theta) = q_1(\mu;\theta)q_2(\sigma^2;\theta)$. We use the following modeling choices:

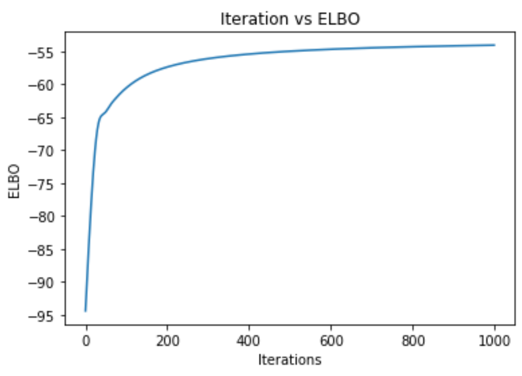
$$\begin{aligned}
p(\mu) &= \mathcal{N}(\mu|m_p, s_p^2) \\
p(\sigma^2) &= Gamma(\sigma^2|\alpha_p, \beta_p) \\
q_1(\mu;\theta) &= \mathcal{N}(\mu|m, s^2) \\
q_2(\sigma^2;\theta) &= Gamma(\sigma^2|\alpha, \beta)
\end{aligned}$$

$\theta = [m, s^2, \alpha, \beta]$ is the vector of **variational parameters**. We need to optimize $ELBO(q)$ with respect to these variational parameters. Using the modeling choices given above, the following expression for $ELBO(q)$ can be derived:

$$\begin{aligned}
ELBO(q) = K &+ \frac{\Gamma(\alpha-1)}{\beta\Gamma(\alpha)}\Big(m\sum_{i=1}^{N} x_i - \frac{N}{2}(m^2+s^2) - \frac{1}{2}\sum_{i=1}^{N} x_i^2\Big) \\
&+ (\alpha_p - \frac{N}{2})(\psi(\alpha) + \log\beta) - \frac{\alpha\beta}{\beta_p} + \alpha + \log\Gamma(\alpha) \\
&- \alpha\psi(\alpha) + \frac{\log 2s^2\pi e}{2} \qquad (6)
\end{aligned}$$

K is a constant that contains all the terms which are not dependent on variational parameters or data. The function $\psi(\alpha) = \frac{\Gamma'(\alpha)}{\Gamma(\alpha)}$ is called the digamma function. $\alpha_p, \beta_p, m_p, s_p^2$ are hyper-parameters.

Figure 1 shows the results obtained from this experiment. More details about this experiment and other results can be found in the Basic Concepts notebook from the GitHub repository.



(a) Iteration vs ELBO

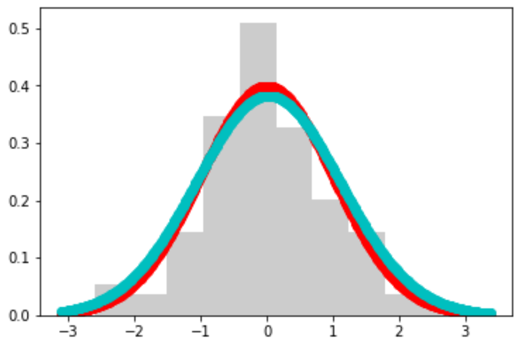
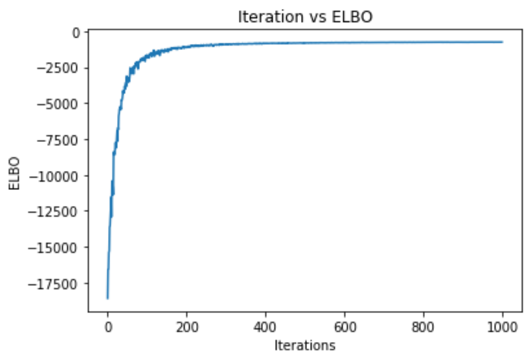

(b) Approximated Distribution (in blue)

Figure 1: Experiment 1: Univariate Gaussian



(a) Iteration vs ELBO

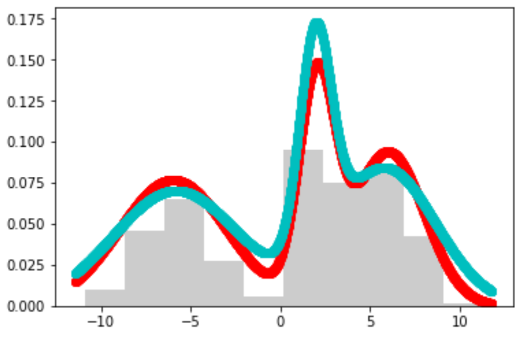

(b) Approximated Distribution (in blue)

Figure 2: Experiment 2: Gaussian Mixture Model

4 Experiment 2: Gaussian Mixture Model

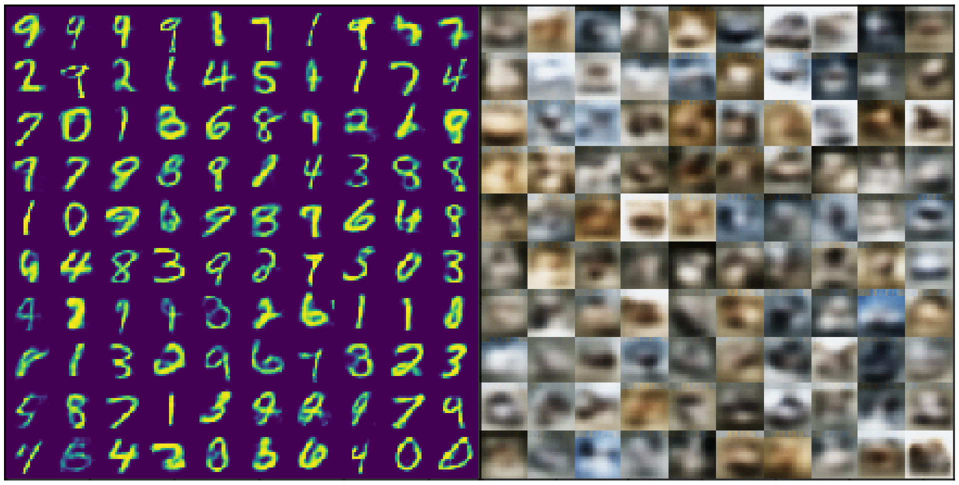
The setup of this experiment is same as that of experiment 1, except that we are going to model $\mathcal{D}$ by a mixture of Gaussian. Details about this experiment have been presented in appendix A. In this section we present the main results.

It can be seen in figure 2b that the approximated distribution (in blue) is very close to the true distribution (in red). While in experiment 1, we obtained an analytical form for the expectation in $ELBO$ in this case a Monte-Carlo approximation for the expectation was used (although it is easy to derive the analytical formula as well). This is a well known practice which is used when the expectation can not be calculated analytically.

For generating the plots shown in figure 2, only one sample was used to estimate the expectation at each iteration. Although compared to figure 1a, the figure in 2a is not as smooth because of Monte-Carlo approximation, nonetheless the variance is low enough and we reach a local minima.

In experiment 1, Gamma prior was used for variance. However it is hard to re-parameterize the Gamma distribution. Re-parameterization [4] is needed to use Monte-Carlo estimation along with an automatic differentiation software package such as TensorFlow. In this experiment, we use a log-normal prior for variance. It is very easy to re-parameterize the log-normal distribution. To the best of our knowledge, this is the first time that a log-normal prior has been used in this context.

An interested reader may refer to Basic Concepts notebook in the GitHub repository for more details and to view comparison with standard techniques such as EM algorithm.



(a) Generated MNIST Images (b) Generated CIFAR-10 Images

Figure 3: Case Study: Variational Autoencoder

5 Case Study: Variational Autoencoder

A generative model is a probabilistic model that explains the process of generation of observed data. In this experiment, we are given a dataset $\mathcal{D} = \{x^{(1)}, x^{(2)}, ..., x^{(N)}\}$ where each $x^{(i)}$ is an image. Variational autoencoder [4] is a generative model that uses variational inference to model $p(x)$ using latent variables. As a generative model it can be used to generate new images from the same distribution. More details and other results can be found in appendix B and in the Variational Autoencoder notebook in the GitHub repository. In this section we present the results obtained by training a variational autoencoder on two datasets - MNIST [5] and CIFAR-10 [6].

Figure 3 shows the images that were generated using variational autoencoders trained using MNIST & CIFAR-10 datasets. The images for CIFAR-10 are very blurry. This is a general problem with images generated using variational autoencoders. Many modifications have been proposed to vanilla variational autoencoders for example the DRAW network [7] which combines attention with a variational autoencoder. Such models are beyond the scope of this project.

6 Other Topics

Variational inference is an active area of research both in machine learning and statistics. Many successful models have been built using variational inference. Two such models - variational autoencoder [4] (appendix B) and Latent Dirichlet allocation (LDA) [8] (appendix C) have been discussed in this report.

Variational inference can be seen as an alternative to more traditional Markov Chain Monte-Carlo (MCMC) [9] approaches. Appendix D highlights certain differences between variational inference and the MCMC approach. Questions related to which technique is more suitable in which case have also been answered.

There are many open problems in the area of variational inference. Some of them are:

1. Using measures other than KL-divergence
2. Relaxing the mean field assumption in general case
3. Dealing with the problem of underestimated variance
4. Understanding various statistical properties of variational inference

These and other open problems have been mentioned in detail in [1]. It will be exciting to see where the field progresses.

Acknowledgements

I would like to thank all the members of Statistics and Machine Learning group and especially Gaurav for insightful discussions at different stages during the project.

References

[1] Blei, D.M., Kucukelbir, A. & McAuliffe, J.D. (2016). Variational Inference: A Review for Statisticians. 2016arXiv160100670B.

[2] Jordan, M. I., Ghahramani, Z., Jaakkola, T., & Saul, L. (1999). Introduction to variational methods for graphical models. Machine Learning, 37:183233.

[3] Minka, T. (2005). Divergence measures and message passing. Technical Report TR-2005-173, Microsoft Research.

[4] Kingma, D.P & Welling, M. (2013). Auto-Encoding Variational Bayes. 2013arXiv1312.6114K.

[5] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition, Proceedings of the IEEE, 86(11):2278-2324

[6] Krizhevsky, A. (2009). Learning Multiple Layers of Features from Tiny Images

[7] Gregor, K., Danihelka, I., Graves, A. & Wierstra, D. (2015). DRAW: A Recurrent Neural Network For Image Generation, Proceedings of the 32 nd International Conference on Machine Learning. JMLR: W&CP volume 37.

[8] Blei, D., Ng, A., & Jordan, M. I. (2003). Latent Dirichlet allocation. Journal of Machine Learning Research, 3:9931022.

[9] Robert, C. & Casella, G. (2011). A Short History of Markov Chain Monte Carlo: Subjective Recollections from Incomplete Data. Statist. Sci. 26 (2011), no. 1, 102–115.

A Experiment 2: Details

Given $\mathcal{D} = \{x^{(1)}, x^{(2)}, ..., x^{(N)}\}$, where each $x^{(i)} \in \mathbb{R}$, we model this data using a mixture of Gaussian distribution. We assume that the number of mixture components M is known. Corresponding to each observation $x^{(i)}$ there is a hidden random variable $z^{(i)}$. $z^{(i)}$ is represented by using $1 - of - M$ encoding such that if $x^{(i)}$ belongs to j^{th} mixture component then $z^{(1)}_j = 1$ and all the other components of $z^{(i)}$ are set to zero. Let $\mu = [\mu_1, \mu_2, ..., \mu_M]^\intercal$ be a vector such that μ_j is the mean of j^{th} mixture component. Similar to μ we define $\sigma^2 = [\sigma_1^2, \sigma_2^2, ..., \sigma_M^2]$ for the variances. Now, given the value of $z^{(i)}$, the probability distribution of $x^{(i)}$ is given by:

$$p(x^{(i)}|z^{(i)}, \mu, \sigma^2) = \mathcal{N}(x^{(i)}|z^{(i)\intercal}\mu, z^{(i)\intercal}\sigma^2)$$

In order to proceed with a Bayesian treatment we also need prior distributions. We impose the following priors:

$$p(z^{(i)}) = 1/M$$

$$p(\mu_j) = \mathcal{N}(\mu_j|0, \sigma_\mu^2); \quad p(\mu) = \prod_{j=1}^{M} p(\mu_j)$$

$$p(\sigma_j^2) = ln\mathcal{N}(\sigma_j^2|0, \sigma_\sigma^2); \quad p(\sigma^2) = \prod_{j=1}^{M} p(\sigma_j^2)$$

We assume that the components of μ and σ^2 are mutually independent. σ_μ^2 and σ_σ^2 are hyper-parameters. Next we use the mean-field approach and propose a distribution q to approximate the posterior distribution $p(Z, \mu, \sigma^2|\mathcal{D})$. Here $Z = \{z^{(1)}, z^{(2)}, ..., z^{(N)}\}$.

$$q(Z, \mu, \sigma^2) = \Big(\prod_{i=1}^{N} q_{1i}(z^{(i)})\Big)\Big(\prod_{j=1}^{M} q_{2j}(\mu_j)\Big)\Big(\prod_{j=1}^{M} q_{3j}(\sigma_j^2)\Big)$$

We model $q_{1i}(z^{(i)})$ by a multinomial distribution. More specifically $q_{1i}(z^{(i)} = e_j) = \pi_{ij}$ such that $\forall i, j \ \pi_{ij} \geq 0$ and $\forall i \ \sum_{j=1}^{M} \pi_{ij} = 1$. The distribution over mean variables $q_{2j}(\mu_j)$ is modelled by a normal distribution i.e. $q_{2j}(\mu_j) = \mathcal{N}(\mu_j|m_j, s_j^2)$. Finally, we model $q_{3j}(\sigma_j^2) = ln\mathcal{N}(\sigma_j^2|\alpha_j, \beta_j^2)$, where $ln\mathcal{N}(\alpha, \beta^2)$ is a log-normal distribution with mean α and scale parameter β^2. Thus we have a total of $NM + 4M$ variational parameters that should be learned from data.

Using the modeling choices given above, one can arrive at the following expression for $ELBO(q)$. The derivation of this expression can be found in the Basic Concepts notebook in the GitHub repository.

$$\begin{aligned}
ELBO(q) &= \mathrm{E}_{Z\sim q}\Big[\log P(X, Z, \mu, \sigma^2) - \log q(Z, \mu, \sigma^2)\Big] \\
&= \mathrm{E}_{Z\sim q}\Big[K - \frac{1}{2\sigma_\mu^2}\sum_{j=1}^{M}\mu_j^2 - \frac{1}{2\sigma_\sigma^2}\sum_{j=1}^{M}(\log\sigma_j^2)^2 - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} z_j^{(i)}\log\sigma_j^2 \\
&- \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M}\pi_{ij}\frac{(x_i - \mu_j)^2}{\sigma_j^2} + \frac{1}{2}\sum_{j=1}^{M}\log s_j^2 + \frac{1}{2}\sum_{j=1}^{M}\frac{(\mu_j - m_j)^2}{s_j^2} + \frac{1}{2}\sum_{j=1}^{M}\log\beta_j^2 \\
&+ \frac{1}{2}\sum_{j=1}^{M}\frac{(\log\sigma_j^2 - \alpha_j)^2}{\beta_j^2} - \sum_{i=1}^{N}\sum_{j=1}^{M}\pi_{ij}\log\pi_{ij}\Big]
\end{aligned}$$

B Variational Autoencoder: Details

Figure 4 shows the model of image generation that is used by our variational autoencoder implementation. A hidden random vector $z \in \mathbb{R}^m$ is sampled from the prior distribution $p(z|\theta)$. Based on a sampled value of z, an image $x \in \mathbb{R}^d$ is sampled from the distribution $p(x|z, \theta)$. These distributions are modeled as follows:

$$p(z|\theta) = \mathcal{N}(z|0, I)$$

$$\log p(x^{(i)}|z, \theta) = \sum_{j=1}^{d}\Big(x_j^{(i)}\log y_j^{(i)} + (1 - x_j^{(i)})\log(1 - y_j^{(i)})\Big) \tag{7}$$

In equation 7, $y^{(i)} \in \mathbb{R}^d$ is the reconstructed image. $p(x^{(i)}|z, \theta)$ is modeled as a multivariate Bernoulli distribution based on the image $y^{(i)}$ that is reconstructed using $z \sim q(z|x^{(i)}, \phi)$. $y^{(i)}$ obtained from z using the

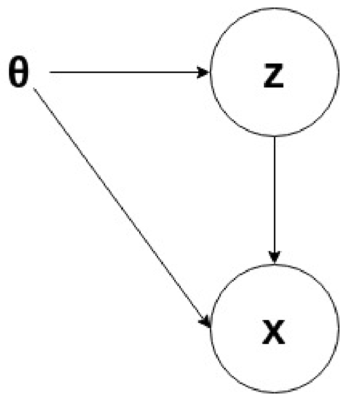


Figure 4: Image Generation Process

neural network parameterized by θ. This neural network is known as a decoder. $q(z|x^{(i)}, \phi)$ is the distribution that we use to approximate the true posterior $p(z|x^{(i)})$. The distribution q is modeled as follows:

$$q(z|x^{(i)}; \phi) = \mathcal{N}(z|x^{(i)}, \mu^{(i)}, \Sigma^{(i)}) \tag{8}$$

In equation 8, $\mu^{(i)} \in \mathbb{R}^m$ and $\Sigma^{(i)}$ are obtained by using outputs from a neural network called encoder. Encoder is parameterized by ϕ. The final expression obtained for lower bound on the log-likelihood of observed data (ELBO) is:

$$\begin{aligned} ELBO(q) &= \mathcal{L}(\theta, \phi) \\ &= \sum_{i=1}^{N} \Big(E_{z \sim q(z|x^{(i)};\phi)}[\log p(x^{(i)}|z;\theta)] - KL[q(z|x^{(i)};\phi)||p(z;\theta)] \Big) \\ &= \sum_{i=1}^{N} \sum_{j=1}^{d} \Big(x_j^{(i)} \log y_j^{(i)} + (1 - x_j^{(i)}) \log(1 - y_j^{(i)}) \Big) \\ &+ \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{m} \Big(1 + \sigma_j^{(i)} - (\mu_j^{(i)})^2 - (e^{\sigma_j^{(i)}})^2 \Big) \end{aligned} \tag{9}$$

To train the variational autoencoder $\mathcal{L}(\theta, \phi)$ is maximized with respect to parameters θ and ϕ. This can be done in an end to end fashion by using the re-parameterization trick while sampling $z = \mu + \epsilon^\intercal \Sigma$, where $\epsilon \sim U[0, 1]^m$.

Detailed derivation of equation 9 and other details can be found in the Variational Autoencoder notebook in the GitHub repository.

C Latent Dirichlet Allocation (LDA)

Latent Dirichlet Allocation [8] is a probabilistic model that describes the process of generation of documents. Each document is modeled as a mixture of topics and each topic is modeled as a mixture of words. Hence corresponding to each document we have a probability distribution over topics and corresponding to each topic we have a probability distribution over words.

A document $d^{(j)} = \{x_1^{(j)}, x_2^{(j)}, ..., x_{n_j}^{(j)}\}$ is represented using a bag of words approach. $x_i^{(j)}$ represents the i^{th} word of j^{th} document. The generation of each document is modeled as follows:

1. Sample a multinomial distribution over topics from a Dirichlet distribution as: $\theta^{(j)} \sim Dirichlet(\alpha)$.
2. Sample the number of words in the document as: $n_j \sim Poisson(\lambda)$.
3. For each of the n_j words in the document do the following:
   (a) Sample a topic $t \sim \theta^{(j)}$.
   (b) Sample a word $x_i^{(j)} \sim \tau_t$, where τ_t refers to the probability distribution over words corresponding to topic t.

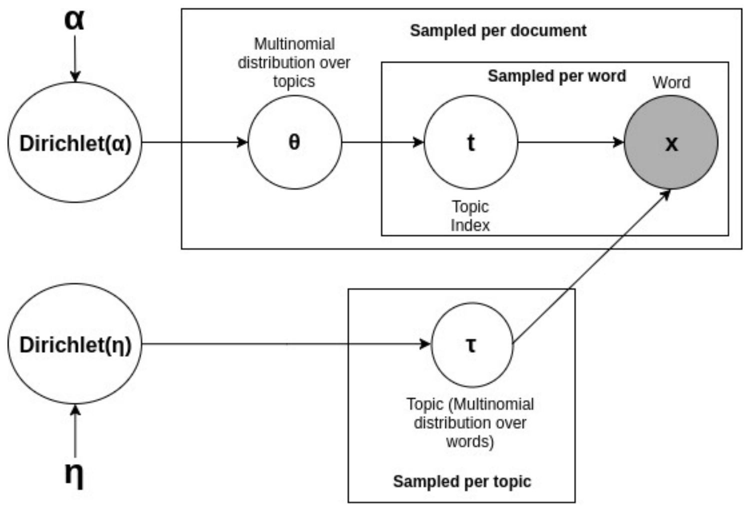


Figure 5: Generation Model for LDA

The parameters λ and α are a hyper-parameters. A $Dirichlet(\alpha)$ prior is used for the distribution of topics in a document. $\theta^{(j)} \sim Dirichlet(\alpha)$ represents a multinomial probability distribution over topics and hence if we have K topics, then $\theta^{(j)} \in \mathbb{R}^K$. We use a Poisson distribution to model the prior distribution over the number of words in a document.

We also need to model the prior distribution over words in a topic. Since the samples from this distribution must represent a multinomial distribution (topics are represented as multinomial distribution over words), the prior is modeled using a Dirichlet distribution, i.e. $p(\tau_t) = Dirichlet(\tau_t|\eta)$, $t = 1, 2, ..., K$. The parameter η is again a hyper-parameter. Since τ_t is a multinomial distribution over the vocabulary, $\tau_t \in \mathbb{R}^{|V|}$, where $|V|$ represents the size of vocabulary. Figure 5 shows the model of generation process.

Given a document $\mathcal{D}$, one can obtain $p(\mathcal{D}|\theta, \tau, T)$ using the model. Here $T = \{t_1^{(j)}, t_2^{(j)}, ..., t_{n_j}^{(j)}\}$ where t_i^j is the topic index chosen for word i in document j. We are interested in finding $p(\theta, \tau, T|\mathcal{D})$. This is the problem of inferring a posterior distribution. One can employ mean-field variational inference approach to approximate the posterior as we have done in other cases. More details on formulation can be found in [8].

D Comparison with MCMC approach

Variational inference provides a deterministic approximation to the true posterior. By deterministic we mean that the parametric form of the approximated distribution $q(x)$ is fixed which determines the family to which $q(x)$ belongs. Variational inference chooses a member from this family which best approximates the true posterior (under KL divergence). Markov Chain Monte-Carlo methods [9] on the other hand provide stochastic approximations.

The main idea behind MCMC is to construct a Markov chain whose stationary distribution is equal to the posterior distribution. After the “burn in” phase of the Markov Chain one starts getting unbiased samples from the true posterior distribution.

While it can be asymptotically guaranteed that MCMC can model the true posterior distribution, such guarantees do not exist for variational inference. The accuracy of approximation depends on the expressive power of the parametric class of functions q that is being used.

On the other hand, variational inference methods tend to be faster than MCMC methods which are more computationally intensive due to the “burn in” phase. We have good tools for optimization. With the emergence of stochastic optimization and automatic differentiation softwares variational inference becomes particularly well suited for large datasets. MCMC on the other hand is more applicable when more exact samples are required and computational expenses are not the primary concern. This is because MCMC provide asymptotic guarantees.

This issue has been explored in more detail in [1].