A Brief History of the Wake-Sleep Algorithm

Ronak Mehta

1 Introduction

The goal of this note is to review the basic details of the wake-sleep algorithm [7] and its modern variants. Wake-sleep-like optimization procedures have shown striking success in generative modeling tasks including highly structured latent variable models with discrete latent codes and/or observations, from natural language to program induction. [8, 6, 11] Interestingly, the method is different from nearly all prominent approaches to generative models in that it involves a multi-objective optimization. As such, many of the algorithm's empirical and theoretical properties remain elusive, and further study could pioneer a promising and novel line of research. The remainder of this note fits the wake-sleep algorithm into the larger context of generative modeling via latent variables.

2 Generative Models

Generative modeling is the problem of sampling new observations from a probability distribution p over $\mathcal{X}$ given only access to a sample $x_1, ..., x_n \sim p$.

One way to frame this statistical problem is as the composition of density estimation and simulation, as a density estimation approach can return estimate $\hat{p}$, and then methods from probability can be used to sample from this distribution. However, an implicit assumption is that the space $\mathcal{X}$ is high-dimensional and richly structured, such as in the case of images, text, and audio. In these scenarios, representing the density $\hat{p}$ may be a difficult task, without even considering the question of estimating it effectively.

Instead, we assume that have the ability to sample observations over from a "simple" distribution (say Uniform$(0, 1)$ or $\mathcal{N}(0, I)$) over some space $\mathcal{Z}$, and would like to learn a *generator* $g : \mathcal{Z} \to \mathcal{X}$ such that if $g(z) \sim p$ when z follows the simple distribution. The generator is often selected from a parametrized function class $\{g_\theta : \theta \in \Theta\}$.

There are a number of classes of algorithms that are popular for generative modeling, namely generative adversarial networks (GANs) [4], energy-based models (EBMs) [2], and latent variable models [3]. To build up to the wake-sleep approach we focus here on latent variable models.

3 Latent Variable Models and the Evidence Lower Bound

Latent variable models impose additional structure by assuming that the random x is sampled by first drawing a unobserved variable $z \sim r$, and then drawing $x \sim p(\cdot|z)$. The density of x is then written

$$p(x) = \int_{\mathcal{Z}} p(x|z) \cdot r(z) \; dz.$$

This can be useful because by choosing a good z, the likelihood $p(x|z)$ and prior $r(z)$ might have much simpler forms than the marginal $p(x)$, both from the perspective of representation and estimation. Additionally, the latent variables might better capture global codependencies between features of x, whereas other models (such as autoregressive models) may struggle give the examples

global coherence. For example, an autoregressive model might model the top left and bottom right of an image as independent, whereas a latent variable can disperse information throughout the image via $p(x|z)$.

We will often assume that $p(x|z)$ is of some parametric form $p_\theta(x|z)$ (we can assume the same of the prior as well). In this case, we can define the optimal $\theta^* \in \Theta$ by minimization of KL divergence:

$$\theta^* = \arg\min_\theta \; D(p \,||\, p_\theta).$$

The sample estimate of the above yields the the method of maximum likelihood. An example of such a parametric family is the Gaussian Mixture Model (GMM), in which $\mathcal{X} = \mathbb{R}^d$, $|\mathcal{Z}| = K$, $\pi \in \Delta^{K-1}$, and $\mu_z \in \mathbb{R}^d, \Sigma_z \in \mathbb{R}^{d\times d}$. The observations follow

$$\begin{aligned} z &\sim \text{Categorical}(\pi) \\ x &\sim \mathcal{N}(\mu_z, \Sigma_z). \end{aligned}$$

Here, $\theta = (\pi, \{\mu_z\}_z, \{\Sigma_z\}_z)$. For the GMM, we can easily write down and estimate the likelihood.

$$\begin{aligned} p_\theta(x) &= \sum_{z\in\mathcal{Z}} \pi_z \cdot \mathcal{N}(\mu_z, \Sigma_z) \\ \log p_\theta(x) &= \text{LogSumExp}_{z\in\mathcal{Z}} \left(\log \pi_z + \log \mathcal{N}(\mu_z, \Sigma_z)\right) \end{aligned}$$

More generally, if the latent variable z is continuous or is discrete with a vary large number of categories (such as a bit vector), computing the integral will be intractable (as will computing the gradient of the log-likelihood). One approach is to sample Monte Carlo iterates $z_j \sim r$ and estimate

$$p_\theta(x) = \mathbb{E}_{z\sim r}\left[p_\theta(x|z)\right],$$

but this estimate will have prohibitively high variance in most cases. For that reason, we typically employ *importance sampling* to estimate this integral. Using draws from a *proposal distribution* $q(z|x)$ that depends on x (the point at which we would like to evaluate the log-likelihood), we can estimate

$$\log p_\theta(x) = \log \mathbb{E}_{z\sim q(\cdot|x)}\left[\frac{p_\theta(x,z)}{q(z|x)}\right] = \log \mathbb{E}_{z\sim q(\cdot|x)}\left[\frac{p_\theta(x|z)\cdot r(z)}{q(z|x)}\right],$$

by plugging in the sample average into the expectation. We are almost there! The estimate will be biased (it is a sample mean inside the log), and numerically unstable, as the integrand has probabilities multiplied and divided by one another. We could solve both of these issues by pulling the log inside the expected value; the integrand would now be $\log p_\theta(x|z) + \log r(z) - \log q(z|x)$, and because the expected value will be on the outside of the log, we can produce an unbiased estimate of $\mathbb{E}_{z\sim q(\cdot|x)}\left[\log p_\theta(x|z) + \log r(z) - \log q(z|x)\right]$. Of course, we are now estimating something that is not the log marginal likelihood of x. However, noticing that

$$\log \mathbb{E}_{z\sim q(\cdot|x)}\left[\frac{p_\theta(x,z)}{q(z|x)}\right] = \mathbb{E}_{z\sim q(\cdot|x)}\left[\log \frac{p_\theta(x,z)}{q(z|x)}\right] + D(q(z|x) \,||\, p_\theta(z|x)), \tag{1}$$

we are in business! Because the KL-divergence $D(q(z|x) \,||\, p_\theta(z|x)) \geq 0$ and is zero when $q(z|x) = p_\theta(z|x)$, we can claim that

$$\log p_\theta(x) = \max_q \; \mathbb{E}_{z\sim q(\cdot|x)}\left[\log \frac{p_\theta(x,z)}{q(z|x)}\right].$$

The term inside the maximum is called the evidence lower bound (ELBO). Repeating the above with an expected value over x, we can also claim that

$$-H(p, p_\theta) = \mathbb{E}_{x\sim p}\left[\log p_\theta(x)\right] = \max_q \ \mathbb{E}_{x\sim p}\left[\mathbb{E}_{z\sim q(\cdot|x)}\left[\log \frac{p_\theta(x,z)}{q(z|x)}\right]\right],$$

where $H(p, p_\theta)$ is the cross entropy of p to p_θ. If we use the posterior distribution $p_\theta(z|x)$ as our proposal distribution, then we can estimate the log marginal likelihood effectively. Unfortunately, that requires computing the integral over $\mathcal{Z}$ anyway, which puts as back at square one. The key insight of *variational inference* is that if we let the proposal belong to an expressive function class $\{q_\varphi(z|x) : \varphi \in \Phi\}$, then we can approximate the log marginal likelihood, as per

$$\max_q \ \mathbb{E}_{z\sim q(\cdot|x)}\left[\log \frac{p_\theta(x,z)}{q(z|x)}\right] \approx \max_\varphi \ \mathbb{E}_{z\sim q_\varphi(\cdot|x)}\left[\log \frac{p_\theta(x,z)}{q_\varphi(z|x)}\right]$$

Now, we have an optimization problem that we can use to find the KL minimizer, which we can efficiently solve with stable, unbiased stochastic gradient updates.

$$\begin{aligned}\theta^* &= \arg\min_\theta \ D(p \,||\, p_\theta) \\ &= \arg\max_\theta \ \mathbb{E}_{x\sim p}\left[\log p_\theta(x)\right]\end{aligned}$$

Using the posterior approximator which is often called the *inference*, *recognition*, or *encoder* network, we have

$$\theta^* \approx \arg\max_\theta \ \max_\varphi \ \mathcal{L}(\theta, \phi) = \arg\max_\theta \ \max_\varphi \ \mathbb{E}_{x\sim p}\left[\mathbb{E}_{z\sim q_\varphi(\cdot|x)}\left[\log \frac{p_\theta(x,z)}{q_\varphi(z|x)}\right]\right] \tag{2}$$

The following methods will differ on how to perform this optimization, a fork that will lead to variational autoencoders (VAEs) on one side, and the wake-sleep algorithm and Helmholtz machines on the other. VAEs will attempt to perform this optimization directly via coordinate ascent, estimating the gradients of the objective in Equation 2 and updating with respect to θ and φ in an alternating fashion. As we will see, the gradient with respect to φ will cause some problems for estimation, and a host of methods have been introduced to alleviate these problems in VAEs. The wake-sleep algorithm takes a different approach; it will introduce a new objective for φ that will directly measure closeness to $p_\theta(z|x)$. We will first review the VAE approach to learning latent variable models before returning to that of wake-sleep.

4 Gradient Estimates for Latent Variable Models

The objective in Equation 2 is relatively straightforward to optimize with respect to θ for any fixed φ, by estimating the gradient with data x and proposal samples $z \sim q(\cdot|x)$. The gradient with respect to φ is a little trickier to estimate, but given both of these quantities, one can optimize the function by coordinate ascent on θ and φ. This is the approach taken by varational autoencoders. The difficulty arises because expected value is over a distribution which depends on φ, so we can't pass a gradient directly over φ through. The policy gradient theorem, a.k.a. REINFORCE [15], gets around this with a mathematics trick, and we have:

$$\begin{aligned}\nabla_\varphi \ \mathcal{L}(\theta, \phi) &= \nabla_\varphi \ \mathbb{E}_{x\sim p}\left[\mathbb{E}_{z\sim q_\varphi(\cdot|x)}\left[\log \frac{p_\theta(x,z)}{q_\varphi(z|x)}\right]\right] \\ &= \mathbb{E}_{x\sim p}\left[\mathbb{E}_{z\sim q_\varphi(\cdot|x)}\left[\left(\log p_\theta(x|z) - \log q_\varphi(z|x) - 1\right)\nabla_\varphi \log q_\varphi(z|x)\right]\right]\end{aligned}$$

While the sample average analog of this provides a consistent estimate of the gradient, the variance is known to be extremely high. [14] Thus, a significant line of work is dedicated to lowering the variance of the REINFORCE estimator, thus making this method a viable way to fit latent variable models.

One effective method is known as the *reparametrization trick*. [10] Given a generic function h, say we would like to estimate

$$\nabla_\varphi \, \mathbb{E}_{z\sim q_\varphi} \left[h(z)\right].$$

The reparametrization trick introduces a random variable $\epsilon : \Omega \to \mathcal{E}$ and a function $f : \mathcal{E} \times \Phi \to \mathcal{Z}$ that is differentiable with respect to $\varphi \in \Phi$, such that

$$h(z) = f_\varphi(\epsilon).$$

Usually, the dependence on φ is written as a subscript. Thus, the gradient with respect to φ "passes through":

$$\nabla_\varphi \, \mathbb{E}_{z\sim q_\varphi} \left[h(z)\right] = \mathbb{E}_\epsilon \left[\nabla_\varphi f_\varphi(\epsilon)\right].$$

In the VAE context, f will also depend on data x. This results in a low-variance estimate that can be applied to the ELBO objective. Unfortunately, this method can only be used for continuous $\mathcal{Z}$, as no differentiable function can map to a finite set. For discrete latents, adjustments must be made.

Once such adjustment is *continuous relaxation* of the discrete z. Assuming our estimand is of the form

$$\nabla_\varphi \, \mathbb{E}_{z\sim q_\varphi} \left[h(\text{OneHot}(z))\right],$$

we can replace the $\text{OneHot}(z)$ argument with an element from the $|\mathcal{Z}| - 1$-simplex, using samples from the Gumbel distribution. Specifically, if $\mathcal{Z} = \{z_1, ..., z_k\}$, and we draw $g_1, ..., g_k \sim \text{Gumbel}(0, 1)$, then we have

$$\arg\max_{i=1,...,k} \left(g_i + \log q_\varphi(z_i)\right) \sim q_\varphi.$$

Replacing the "argmax" with a "softmax", this form of continuous relaxation yields

$$\nabla_\varphi \, \mathbb{E}_{z\sim q_\varphi} \left[h\left(\text{SoftMax}_\tau\left(\left[g_i + \log q_\varphi(z_i)\right]_{i\in[k]}\right)\right)\right],$$

This is a biased gradient estimate, and as the temperature $\tau \to 0$, while it approaches unbiasedness, the variance of the gradients explode. Thus, the temperature parameter must be tuned to use this method. [12]

The final method to alleviate variance in these gradient estimates is *control variate* or *baseline* methods. Here, we choose an auxilary random variable b, and let q_φ now represent the joint distribution of (z, b). We choose b such that the expectation $\mathbb{E}_{b\sim q_\varphi}[b]$ is known and b is highly positively correlated with z. Assume we have already applied the policy gradient theorem, and now just want to estimate the function $\mathbb{E}_{z\sim q_\varphi}[h(z)]$. Letting each $z_j \sim q_\varphi$, we then revise our estimate from

$$\frac{1}{K}\sum_{j=1}^{K} h(z_j)$$

to

$$\frac{1}{K}\sum_{j=1}^{K}(h(z_j)-b)+\mathbb{E}_{b\sim q_\varphi}[b]$$

Clearly, this is still an unbiased estimate. When the known expectation $\mathbb{E}_{b\sim q_\varphi}[b]=0$, the b is referred to as a baseline. Because of the correlation between b and $h(z)$, the variance of this estimate decreases, as $\mathrm{Var}(h(z)-b)=\mathrm{Var}(h(z))+\mathrm{Var}(b)-2\mathrm{Cov}(h(z),b)$. The art of these methods comes down to cleverly designing control variates. Mnih and Rezende [13] gives an example of such an engineered control variate, whereas Tucker et al. [14] and Grathwohl et al. [5] presents methods that parametrize and learn the control variate to minimize a variance objective. As expected, this method comes with designing, tuning, and training challenges, as another whole neural network might have to be optimized in addition to the latent variable model being fitted.

5 Helmholtz Machines and the Wake-Sleep Algorithm

Returning to the objective in Equation 2, one can learn a latent variable model by successively maximizing estimates of a lower bound on $\mathbb{E}_{x\sim p}[\log p_\theta(x)]$, a bound that becomes tighter when the proposal distribution $q_\varphi(z|x)$ gets closer to the posterior $p_\theta(z|x)$. The VAE methods try to make the bound tighter by just simply maximizing the ELBO over q_φ via gradient-based optimization. As we saw in the previous section, the gradients will have high variance and require extra care (and hyperparameter tuning) to be effective. This especially comes up when z is discrete. Averaging Equation 1 over x, we have:

$$\begin{aligned}
\mathbb{E}_{x\sim p}[\log p_\theta(x)] &= \mathbb{E}_{x\sim p}\left[\log \mathbb{E}_{z\sim q_\varphi(\cdot|x)}\left[\frac{p_\theta(x,z)}{q_\varphi(z|x)}\right]\right] \\
&= \mathbb{E}_{x\sim p}\left[\mathbb{E}_{z\sim q_\varphi(\cdot|x)}\left[\log\frac{p_\theta(x,z)}{q_\varphi(z|x)}\right]+D(q_\varphi(z|x)\,||\,p_\theta(z|x))\right] \\
&= \mathbb{E}_{x\sim p}\left[\mathbb{E}_{z\sim q_\varphi(\cdot|x)}\left[\log\frac{p_\theta(x,z)}{q_\varphi(z|x)}\right]\right]+\mathbb{E}_{x\sim p}[D(q_\varphi(z|x)\,||\,p_\theta(z|x))]
\end{aligned}$$

Increasing the first term $\mathbb{E}_{x\sim p}\left[\mathbb{E}_{z\sim q_\varphi(\cdot|x)}\left[\log\frac{p_\theta(x,z)}{q_\varphi(z|x)}\right]\right]$ decreases the second term $\mathbb{E}_{x\sim p}[D(q_\varphi(z|x)\,||\,p_\theta(z|x))]$, as the left hand side does not depend on q_φ. In the (original) wake-sleep algorithm, we eschew increasing the ELBO directly in favor of indirectly making $q_\varphi(z|x)$ closer to $p_\theta(z|x)$ using a new objective. [7] How do we measure closeness? We could use the divergence term from above,

$$\mathbb{E}_{x\sim p}[D(q_\varphi(z|x)\,||\,p_\theta(z|x))],$$

and try to minimize this. Unfortunately, this requires knowledge of the posterior. Instead, we will make two important changes. First, we will change the expected value over $x\sim p$ to $x\sim p_\theta$, the current model. Intuitively, if q_φ is expressive enough, this should not matter much, as the KL minimizer will have $q_\varphi(z|x)\approx p_\theta(z|x)$ for *every* value of x (although, if q_φ is restricted, it will fit $p_\theta(z|x)$ well in regions where $p_\theta(x)$ has most mass). The second change that we will make is switching the order of the arguments to the KL divergence. Again, if q_φ is expressive enough, this should still find a distribution close to the posterior. If not, the implications might be a mode-covering approximation instead of a mode-seeking one. Thus, our final objective for the inference

network will be

$$\varphi_\theta^* = \arg\min_{\varphi} \ \mathbb{E}_{x\sim p_\theta}\left[D(p_\theta(z|x) \ || \ q_\varphi(z|x))\right] \tag{3}$$

Why did we make these changes? There is still the pesky posterior. It turns out that objective in Equation 3 has easily estimable gradients! To see this, write

$$\begin{aligned}
\varphi_\theta^* &= \arg\min_{\varphi} \ \mathbb{E}_{x\sim p_\theta}\left[D(p_\theta(z|x) \ || \ q_\varphi(z|x))\right] \\
&= \arg\min_{\varphi} \ \mathbb{E}_{x\sim p_\theta}\left[H(p_\theta(z|x), q_\varphi(z|x))\right] \\
&= \arg\min_{\varphi} \ -\mathbb{E}_{x\sim p_\theta}\left[\mathbb{E}_{z\sim p_\theta(\cdot|x)}\left[\log q_\varphi(z|x)\right]\right] \\
&= \arg\max_{\varphi} \ \mathbb{E}_{(x,z)\sim p_\theta}\left[\log q_\varphi(z|x)\right].
\end{aligned}$$

This is just maximum likelihood estimation of φ_θ^*, using joint samples drawn from the generator distribution! We no longer need to sample from the posterior, instead we can draw $z \sim r$ and $x \sim p_\theta(x|z)$ to form our "data", to which we fit q_φ. Note that this update does not require any of the real $x \sim p$, and only that drawn from the model p_θ. For this reason, the φ update is called the "sleep" phase, where as the θ update (essentially the same as in VAEs) is called the "wake" update. On a more technical note, the wake-sleep algorithm was originally introduced the fit Helmholtz machines, in which both $p_\theta(x|z)$ and $q_\varphi(z|x)$ are stochastic neural networks. Precisely, let $\mathcal{Z} = \{0,1\}^{d_z}$ and $\mathcal{X} = \{0,1\}^{d_x}$. For $p_\theta(x|z)$, z travels through a sequence of neural network layers with sampling operations in between to produce the final x, as per

$$\begin{aligned}
h_2 &\sim \sigma(W_1^p z + b_1^p) \\
h_{l+1} &\sim \sigma(W_l^p h_l + b_l^p) \text{ for } l = 2, ..., L-2 \\
x &\sim \sigma(W_{L-1}^p h_{L-1} + b_{L-1}^p),
\end{aligned}$$

where $\sigma(\cdot)$ is the element-wise sigmoid function and the W_l^p's and b_l^p's are weights and biases. Here, $\theta = \left(\{W_l^p\}_l, \{b_l^p\}_l\right)$. Instead of just pairs (x, z), we end up sampling $(x, h_{L-1}, ..., h_2, z)$. Similarly, the inference distribution $q_\varphi(z|x)$ is sampled as per

$$\begin{aligned}
h_{L-1} &\sim \sigma(W_L^q x + b_L^q) \\
h_{l-1} &\sim \sigma(W_l^q h_l + b_l^q) \text{ for } l = L-1, ..., 3 \\
z &\sim \sigma(W_2^q h_2 + b_2^q),
\end{aligned}$$

with $\varphi = \left(\{W_l^q\}_l, \{b_l^q\}_l\right)$. Thus, the "wake" update takes an estimated gradient step of the ELBO objective

$$\begin{aligned}
\nabla_\theta \ \mathcal{J}_\varphi(\theta) &= \nabla_\theta \ \mathbb{E}_{x\sim p}\left[\mathbb{E}_{z,h_1,...,h_{L-1}\sim q_\varphi(\cdot|x)}\left[\log \frac{p_\theta(x, h_{L-1}, ..., h_1, z)}{q_\varphi(z, h_{L-1}, ..., h_1|x)}\right]\right] \\
&= \nabla_\theta \ \mathbb{E}_{x\sim p}\left[\mathbb{E}_{z,h_1,...,h_{L-1}\sim q_\varphi(\cdot|x)}\left[\sum_{l=2}^{L} \log p_\theta(h_l|h_{l-1})\right]\right],
\end{aligned}$$

letting $h_1 = z$ and $h_L = x$. The q_φ terms do not depend on θ. A standard gradient with respect to θ can be taken here. Similarly, for the "sleep" update, we take a gradient update with respect to

φ of:

$$\nabla_\varphi \ \mathcal{J}_\theta(\varphi) = \nabla_\varphi \ \mathbb{E}_{z\sim r}\left[\mathbb{E}_{x,h_{L-1},\ldots,h_1\sim p_\theta(\cdot|z)}\left[\sum_{l=2}^{L}\log q_\varphi(h_{l-1}|h_l))\right]\right].$$

This alternating optimization has an appealing symmetry to it, and can learn latent variable models that circumvent the difficulty of inference network optimization in VAEs. For this reason, a modified version of the algorithm, called Reweighted Wake-Sleep (RWS) is a competitive alternative to discrete VAEs, which is covered in the next section. This being said, the convergence properties of wake-sleep has received little attention, except in the case of a single latent variable with no hidden layers. [9] For general purpose neural networks, there is no such guarantee. For these reasons, the wake-sleep algorithm is often critiqued for optimizing KL in the "wrong" direction.

6 Reweighted Wake Sleep

Reweighted wake-sleep is another variant of the general method that takes another route earlier in the decision tree. For this method, we skip the variational bound entirely, and optimize the objective directly, appealing to importance sampling to reduce variance. [1] First, note that the marginal likelihood $p_\theta(x)$ admits an unbiased, importance sampled estimate, as per

$$p_\theta(x) = \sum_{z\in\mathcal{Z}} p_\theta(x,z) = \mathbb{E}_{z\sim q(\cdot|x)}\left[\frac{p_\theta(x,z)}{q(z|x)}\right].$$

This will be true for any proposal distribution q. Thus, the gradient of the log marginal likelihood follows

$$\begin{aligned}
\nabla_\theta \ \log p_\theta(x) &= \frac{\nabla_\theta \ p_\theta(x)}{p_\theta} \\
&= \frac{\nabla_\theta \ \mathbb{E}_{z\sim q(\cdot|x)}\left[\frac{p_\theta(x,z)}{q(z|x)}\right]}{\mathbb{E}_{z\sim q(\cdot|x)}\left[\frac{p_\theta(x,z)}{q(z|x)}\right]} \\
&= \frac{\mathbb{E}_{z\sim q(\cdot|x)}\left[\frac{p_\theta(x,z)\nabla_\theta \ \log p_\theta(x,z)}{q(z|x)}\right]}{\mathbb{E}_{z\sim q(\cdot|x)}\left[\frac{p_\theta(x,z)}{q(z|x)}\right]} \\
&= \frac{\mathbb{E}_{z\sim q(\cdot|x)}\left[\frac{p_\theta(x,z)}{q(z|x)}\nabla_\theta \ \log p_\theta(x,z)\right]}{\mathbb{E}_{z\sim q(\cdot|x)}\left[\frac{p_\theta(x,z)}{q(z|x)}\right]}.
\end{aligned}$$

The third step follows from the policy gradient theorem argument. Note that regardless of the choice of q, the above expression is still a formula for the gradient of the log marginal likelihood, unlike the ELBO, which is a lower bound that gets tighter as q approaches $p_\theta(z|x)$. The expression suggests a ratio estimator for the gradient of θ, by sampling $z_j \sim q$, and plugging in the empirical average for the numerator and denominator. This is the approach in Reweighted Wake-Sleep,

yielding

$$\begin{aligned}\nabla_\theta \ \log p_\theta(x) &\approx \frac{\frac{1}{K}\sum_{j=1}^K \frac{p_\theta(x,z_j)}{q(z_j|x)}\nabla_\theta \ \log p_\theta(x,z_j)}{\frac{1}{K}\sum_{k=1}^K \frac{p_\theta(x,z_k)}{q(z_k|x)}} \\ &= \frac{1}{\sum_{k=1}^K w_k}\sum_{j=1}^K w_j \nabla_\theta \ \log p_\theta(x,z_j) \\ &=: \nabla_\theta \ \widehat{\log p_\theta}(x)\end{aligned}$$

with $w_j := \frac{p_\theta(x,z_j)}{q(z_j|x)}$. While $q(z|x) = p_\theta(z|x)$ is the minimum variance estimate for $\mathbb{E}_{z\sim q(\cdot|x)}\left[\frac{p_\theta(x,z)}{q(z|x)}\right]$, *there is no guarantee* that this choice will be the minimum variance estimator for the above estimator, whether it is a ratio of averages or an average of ratios. Additionally, for any finite K this is a *biased* estimator, being a ratio of sample averages to estimate a ratio of expectations. The bias goes away when $q(z|x) = p_\theta(z|x)$, because

$$w_j = \frac{p_\theta(x,z_j)}{q(z_j|x)} = \frac{p_\theta(z_j|x)\cdot p_\theta(x)}{q(z_j|x)} = p_\theta(x),$$

and

$$\begin{aligned}\mathbb{E}_{z_{1:K}\sim p_\theta(\cdot|x)}\left[\nabla_\theta \ \widehat{\log p_\theta}(x)\right] &= \mathbb{E}_{z_{1:K}\sim p_\theta(\cdot|x)}\left[\frac{1}{\sum_{k=1}^K w_k}\sum_{j=1}^K w_j \nabla_\theta \ \log p_\theta(x,z)\right] \\ &= \mathbb{E}_{z_{1:K}\sim p_\theta(\cdot|x)}\left[\frac{1}{K}\sum_{j=1}^K \nabla_\theta \ \log p_\theta(x,z)\right] \\ &= \mathbb{E}_{z\sim p_\theta(\cdot|x)}\left[\nabla_\theta \ \log p_\theta(x,z)\right] \\ &= \sum_{z\in\mathcal{Z}} \frac{\nabla_\theta \ p_\theta(x,z)}{p_\theta(x,z)} p_\theta(z|x) \\ &= \sum_{z\in\mathcal{Z}} \frac{\nabla_\theta \ p_\theta(x,z)}{p_\theta(x,z)} p_\theta(z|x) \\ &= \frac{\nabla_\theta \ p_\theta(x,z)}{p_\theta(x)} \\ &= \frac{\nabla_\theta \ \sum_{z\in\mathcal{Z}} p_\theta(x,z)}{p_\theta(x)} \\ &= \nabla_\theta \log p_\theta(x)\end{aligned}$$

which is no longer reweighted. Thus the authors still parametrize the proposal distribution as $q_\varphi(z|x)$ and optimize it to be close to $p_\theta(z|x)$, using a similar objective as before.

$$\begin{aligned}\varphi_\theta^* &= \arg\min_\varphi \ \mathbb{E}_x\left[D(\tilde{p}_\theta(z|x) \ || \ q_\varphi(z|x))\right] \\ &= \arg\max_\varphi \ \mathbb{E}_x\left[\mathbb{E}_{z\sim\tilde{p}_\theta(\cdot|x)}\left[\log q_\varphi(z|x)\right]\right]\end{aligned}$$

Note that the distribution of x by which to take the expectation is unspecified, and $p_\theta(z|x)$ is modified to be $\tilde{p}_\theta(z|x)$. Bornschein and Bengio [1] propose two variants of this update which specify both of these options. The *wake-phase* φ update sample $x \sim p$ (the real data distribution), and samples $z \sim q_\varphi(\cdot|x) = \tilde{p}_\theta(z|x)$, letting q be considered an estimate of the posterior. For this operation q_φ is detached from the computation graph, in the sense that the gradient is not backpropagated through it. The familiar *sleep-phase* update is the same as the original wake-sleep algorithm, in which $x \sim p_\theta$ and $\tilde{p}_\theta(z|x) = p_\theta(z|x)$, meaning we can just use dream samples $(x, z) \sim p_\theta$ to estimate. Interestingly, the authors leave the q updates as optional, as because this is not a variational method, "any" q will do, and the updates can alleviate bias (as can increasing K). Of note is that variational autoencoders, original wake-sleep, and reweighted wake-sleep all use a biased estimate of the gradient of the log marginal likelihood, and optimize q to alleviate the bias.

7 Applications and Results

Recent works have shown the applicability of wake-sleep-like algorithms to modern problems in generative modeling. Le et al. [11] compares Reweighted Wake Sleep to many of the methods mentioned above in a series of experiments, showing superior performance in learning discrete latent models and having competitive performance for continuous models as well. Hu et al. [8] show success in conditional generative models for text, in which the goal is sample $x \mid z, c$, where z is an "unstructured" latent code sampled from a prior, and c is a user-specified "structered" attribute vector that controls the nature of the text. z and c are conditionally independent given x. Their approach combines the VAE method with wake-sleep, with an encoder $q_E(z \mid x)$ that infers the unstructured code, a discriminator $q_D(c|x)$ that infers the structured code, and a generator $p_G(x|z, c)$ that produces samples. The discriminator $q_D(c|x)$ is trained with a combination of labelled real data and generated examples, and the generator is trained to produce confident discriminator predictions. As a final note, the term "wake-sleep" is applied somewhat loosely to many of these methods, and could even be a blanket phrase for many semi-supervised methods. One could assume that "wake-phase" training is that of the generation network, and "sleep-phase" training is that of the inference network, but [1] uses a wake-phase to update the inference network as a variant. Similarly, one might organize wake updates as that which use real data, and sleep updates as those which use generated data, but [8] use both in the sleep-phase update of the discriminator. This suggests that wake-sleep is a more general denomination that describes any learning procedure that optimizes multiple objectives and makes use of model-generated samples for inference.

References

[1] J. Bornschein and Y. Bengio. Reweighted wake-sleep, 2015.

[2] Y. Du and I. Mordatch. Implicit generation and generalization in energy-based models, 2020.

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

[4] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks, 2014.

[5] W. Grathwohl, D. Choi, Y. Wu, G. Roeder, and D. Duvenaud. Backpropagation through the void: Optimizing control variates for black-box gradient estimation, 2018.

[6] L. Hewitt, T. Anh Le, and J. Tenenbaum. Learning to learn generative programs with memoised wake-sleep. In J. Peters and D. Sontag, editors, *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI)*, volume 124 of *Proceedings of Machine Learning Research*, pages 1278–1287, Virtual, 03–06 Aug 2020. PMLR. URL http://proceedings.mlr.press/v124/hewitt20a.html.

[7] G. Hinton, P. Dayan, B. Frey, and R. Neal. The "wake-sleep" algorithm for unsupervised neural networks. *Science*, 268(5214):1158–1161, 1995. ISSN 0036-8075. doi: 10.1126/science.7761831. URL https://science.sciencemag.org/content/268/5214/1158.

[8] Z. Hu, Z. Yang, X. Liang, R. Salakhutdinov, and E. P. Xing. Toward controlled generation of text. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1587–1596, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL http://proceedings.mlr.press/v70/hu17e.html.

[9] S. Ikeda, S.-I. Amari, and H. Nakahara. Convergence of the wake-sleep algorithm. In *Proceedings of the 11th International Conference on Neural Information Processing Systems*, NIPS'98, page 239–245, Cambridge, MA, USA, 1998. MIT Press.

[10] D. Kingma and M. Welling. Auto-encoding variational bayes. 12 2014.

[11] T. A. Le, A. R. Kosiorek, N. Siddharth, Y. W. Teh, and F. Wood. Revisiting reweighted wake-sleep for models with stochastic control flow. In R. P. Adams and V. Gogate, editors, *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, volume 115 of *Proceedings of Machine Learning Research*, pages 1039–1049, Tel Aviv, Israel, 22–25 Jul 2020. PMLR. URL http://proceedings.mlr.press/v115/le20a.html.

[12] C. J. Maddison, A. Mnih, and Y. W. Teh. The concrete distribution: A continuous relaxation of discrete random variables, 2017.

[13] A. Mnih and D. J. Rezende. Variational inference for monte carlo objectives. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 2188–2196. JMLR.org, 2016.

[14] G. Tucker, A. Mnih, C. J. Maddison, D. Lawson, and J. Sohl-Dickstein. Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 2624–2633, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

[15] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 2004.