Deep Discrete Latent Variable Models

Wilker Aziz
ILLC @ UvA

Outline

Implicit distributions

We can specify a stochastic map by using a (deterministic) NN and a source of random numbers with probability density function $s(\epsilon)$. For each (x, ϵ) the mapping is deterministic, but the noise source induces a random variable $Y|\theta, x$. The **implicit likelihood** assigned to an outcome y given x is $p(y|x, \theta) = \int_{\{\epsilon: f(x,\epsilon;\theta)=y\}} s(\epsilon)\mathrm{d}\epsilon$.

In words, we must 'integrate the density of the noise source for every possible way you can map x to y.'

KL divergence

The Kullback-Leibler divergence (or relative entropy) measures the divergence of a distribution q from a distribution p.

- $\mathrm{KL}\left(q(z) \mid\mid p(z)\right) = \mathbb{E}_{q(z)}\left[\log \frac{q(z)}{p(z)}\right]$
- $\mathrm{KL}\left(q(z) \mid\mid p(z)\right) = \int q(z) \log \frac{q(z)}{p(z)} \mathrm{d}z$ (continuous)
- $\mathrm{KL}\left(q(z) \mid\mid p(z)\right) = \sum_z q(z) \log \frac{q(z)}{p(z)}$ (discrete)

KL divergence - Properties

Properties

- $\mathrm{KL}\,(q(z) \,||\, p(z)) \geq 0$ with equality iff $q(z) = p(z)$.
- $-\mathrm{KL}\,(q(z) \,||\, p(z)) = \mathbb{E}_{q(z)}\left[\log \frac{p(z)}{q(z)}\right] \leq 0.$
- We want: $supp(q) \subseteq supp(p)$; otherwise $\mathrm{KL}\,(q(z) \,||\, p(z)) = \infty$

Wake-Sleep Algorithm

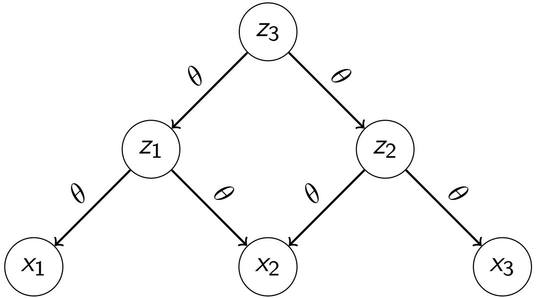
- Generalise latent variables to neural networks.
- Train generative neural model.
- Use variational inference! (kind of)
- Hinton et al. (1995)

Wake-Sleep Architecture

2 neural networks:

- A generation network to model the data (the one we want to optimise) – parameters: θ
- An inference (recognition) network (to model the latent variable) – parameters: λ
- Original setting: binary hidden units
- Training is performed in a "hard EM" fashion

Generator



The 'generator' in wake-sleep is a generative model parameterised by NNs. In the original paper they had an NN with stochastic binary hidden units.

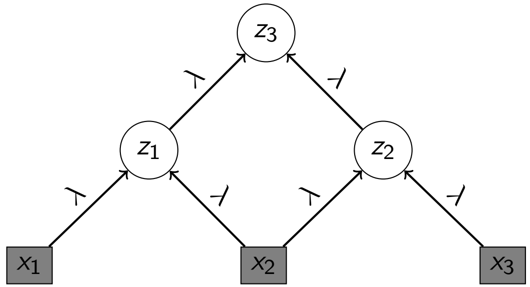
For example, this NN has 3 layers:

- The top one parameterises a distribution over 1 binary random variable, i.e., $Z_3|\theta_0 \sim \text{Bern}(f^{(3)}(\theta_3))$.
- The middle one conditions on a sampled z_3 and parameterises a distribution over 2 binary random variables, i.e., $Z_d|\theta, Z = z_3 \sim \text{Bern}(f_d^{(2)}(z_3; \theta_2))$ for $d = 1, 2$.
- The bottom one conditions on sampled $\langle z_1, z_2\rangle$ and parameterises a distribution over 3 observed random variables. For example, if x is a document we might make an independence assumption: $X_i|\theta, Z_1 = z_1, Z_2 = z_2 \sim \text{Cat}(f^{(1)}(z_1, z_2; \theta_1))$.

The true posterior is clearly intractable, it takes assessing $p(x|\theta) = \sum_{z\in\mathcal{Z}} p(x, z|\theta)$ and $\mathcal{Z}$ is the space of all possible configuration of binary assignments.

I omit arrows from z_2 to x_1 and from z_1 to x_3 to keep the drawing cleaner.

Recognition Network



The recognition network is much like our inference models. It predicts a distribution over Z_1, Z_2, Z_3 given x using an independent model with parameters λ.

This is an NN that predicts as many rvs as there are latent variables in the original model. Think of it as a conditional model of the latent variable.

- We condition on x and parameterise a distribution over two binary random variables, i.e.: $Z_d|\lambda, x \sim \text{Bern}(g^{(1)}(x; \lambda_1))$ for $d = 1, 2$.
- We then condition on sampled $\langle z_1, z_2 \rangle$ and parameterise a distribution over one binary random variable $Z_3|\lambda z_1, z_2 \sim \text{Bern}(g^{(2)}(z_1, z_2; \lambda_2))$

The recognition network specifies an approximate posterior distribution which assumes layer-wise independence, that is, $Z_d^{(\ell)}$ in a layer ℓ is independent on all but the latent variables in the layer below.

I omit arrows from x_2 to z_2 and from x_3 to z_1 to keep the drawing cleaner.

Wake-sleep Training

Wake Phase

- Use inference network to sample hidden unit setting z from $q(z|x, \lambda)$
- Update generation parameters θ to maximize joint log-likelihood of data and latents $p(x, z|\theta)$

Wake-sleep Training

Wake Phase

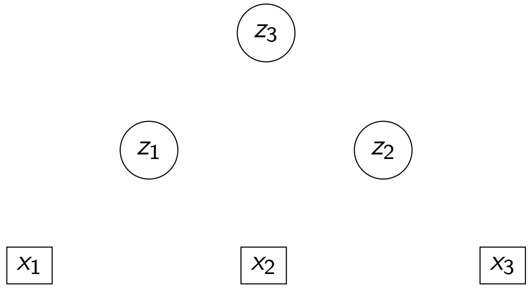
- Use inference network to sample hidden unit setting z from $q(z|x, \lambda)$
- Update generation parameters θ to maximize joint log-likelihood of data and latents $p(x, z|\theta)$

Sleep Phase

- Produce dream sample $z, \tilde{x}$ from the joint distribution
- Update inference parameters λ to maximize probability of latent state $q(z|\tilde{x}, \lambda)$

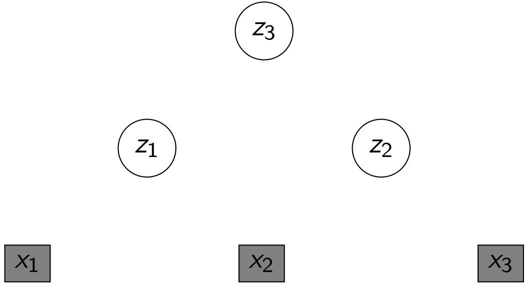
Wake Phase Sampling

Sampling $z \sim q(z|x, \lambda)$



Wake Phase Sampling

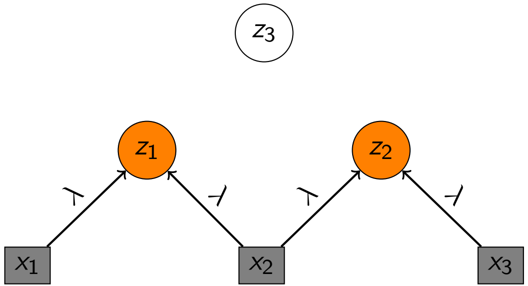
Sampling $z \sim q(z|x, \lambda)$



- Observe x

Wake Phase Sampling

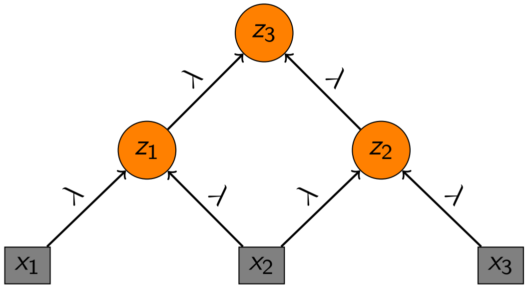
Sampling $z \sim q(z|x, \lambda)$



- Observe x
- Parameterise distributions $Z_d|\theta, X = x$ and sample latent variables z_1, z_2

Wake Phase Sampling

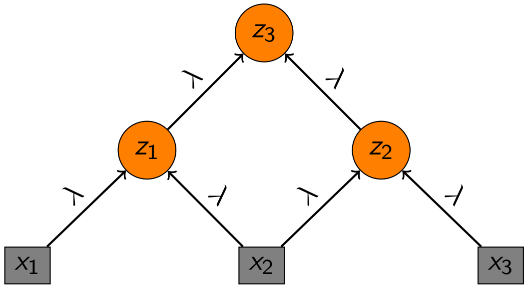
Sampling $z \sim q(z|x, \lambda)$



- Observe x
- Parameterise distributions $Z_d|\theta, X = x$ and sample latent variables z_1, z_2
- Condition on z_1, z_2, parameterise distribution $Z_3|\theta, Z_1 = z_1, Z_2 = z_2$ and sample latent variable z_3.

Wake Phase Sampling

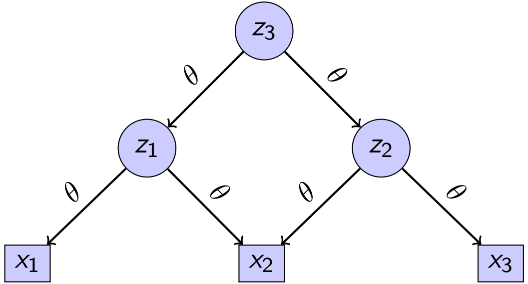
Sampling $z \sim q(z|x, \lambda)$



- Observe x
- Parameterise distributions $Z_d|\theta, X = x$ and sample latent variables z_1, z_2
- Condition on z_1, z_2, parameterise distribution $Z_3|\theta, Z_1 = z_1, Z_2 = z_2$ and sample latent variable z_3.

Wake Phase Update

Compute $\log p(x, z|\theta)$ and update θ



With the sample z we got from the recognition network we can compute the joint probability of z and the observation x. This means we do not need to sample from $p(x, z|\theta)$. The alternative to sampling from the recognition model, would be to fix the observation x and sample from the induced true posterior $p(z|x, \theta)$, which is clearly intractable.

Thus the recognition model plays a role identical to that of the inference model in variational inference.

As in VI, because we sampled from $q(z|x, \lambda)$ it is easy to compute a gradient estimate w.r.t. θ.

The situation is much more difficult w.r.t. λ, as we saw in the section about NVIL. To circumvent difficulties with gradient estimation for λ, in Wake-Sleep, we change the optimisation objective in order to update the recognition model. In particular, we update the recognition model as to maximise the probability of some 'dream data' which we obtain by sampling from the generative model.

Sleep Phase Sampling

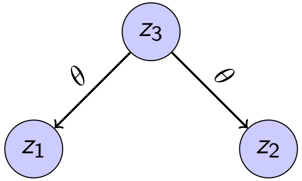
Sampling $(z, \tilde{x}) \sim p(x, z|\theta)$

We do a stochastic forward pass through the generative model sampling our random variables.

- We sample z_3 from the distribution at the top layer.

Sleep Phase Sampling

Sampling $(z, \tilde{x}) \sim p(x, z|\theta)$

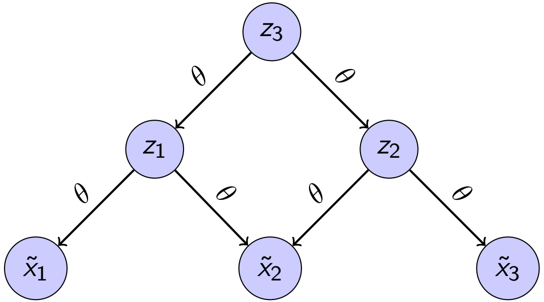


We do a stochastic forward pass through the generative model sampling our random variables.

- We sample z_3 from the distribution at the top layer.
- Then condition on z_3 to parameterise the distribution $Z_1, Z_2|\theta, Z_3 = z_3$, from which we sample z_1 and z_2.

Sleep Phase Sampling

Sampling $(z, \tilde{x}) \sim p(x, z|\theta)$

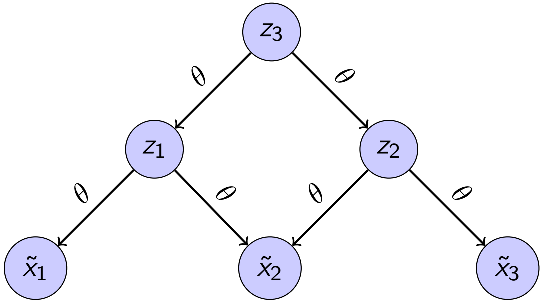


We do a stochastic forward pass through the generative model sampling our random variables.

- We sample z_3 from the distribution at the top layer.
- Then condition on z_3 to parameterise the distribution $Z_1, Z_2|\theta, Z_3 = z_3$, from which we sample z_1 and z_2.
- We condition on z_1 and z_2 to parameterise our output distributions over data space $X_i|\theta, Z_1 = z_1, Z_2 = z_2$, from where we **sample** data. This is crucial, our sample $\tilde{x}$ is not an actual observation (we mark it with tilde to help you track its influence).

Sleep Phase Sampling

Sampling $(z, \tilde{x}) \sim p(x, z|\theta)$

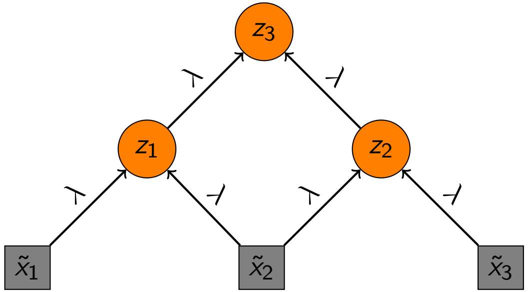


We do a stochastic forward pass through the generative model sampling our random variables.

- We sample z_3 from the distribution at the top layer.
- Then condition on z_3 to parameterise the distribution $Z_1, Z_2|\theta, Z_3 = z_3$, from which we sample z_1 and z_2.
- We condition on z_1 and z_2 to parameterise our output distributions over data space $X_i|\theta, Z_1 = z_1, Z_2 = z_2$, from where we **sample** data. This is crucial, our sample $\tilde{x}$ is not an actual observation (we mark it with tilde to help you track its influence).

Sleep Phase Update

Compute $\log q(z|\tilde{x}, \lambda)$ and update λ



The last ingredient is to assess the likelihood of the sampled z given the sampled $\tilde{x}$ under the recognition model and update λ as to maximise it.

Wake Phase Objective

Objective

$$\arg\min_{\theta} \ \mathbb{E}_{x\sim\mathcal{D}} \left[\mathrm{KL}\left(q(z|x,\lambda) \mid\mid p(z|x,\theta)\right)\right]$$
$$= \arg\max_{\theta} \mathbb{E}_{x\sim\mathcal{D}} \left[\mathrm{ELBO}_x(\theta,\lambda) - \log p(x|\theta)\right]$$

Approximation: optimize the lower-bound alone.

The wake-phase really is identical to VI. It makes the exact same approximation, namely, that optimising a lowerbound on the log-evidence is a good idea.

Wake Phase Objective

Objective

$$\arg\max_{\theta} \ \mathbb{E}_{x\sim\mathcal{D}}\left[\text{ELBO}_x(\theta,\lambda)\right]$$
$$= \arg\max_{\theta} \ \mathbb{E}_{x\sim\mathcal{D}}\left[\mathbb{E}_{q(z|x,\lambda)}\left[\log p(z,x|\theta)\right] + \mathbb{H}[q(z|x,\lambda)]\right]$$

Gradient wrt θ for $x \sim \mathcal{D}$ (an observation)

$$\nabla_\theta \mathbb{E}_{q(z|x,\lambda)}\left[\log p(z,x|\theta)\right] + \nabla_\theta \mathbb{H}[q(z|x,\lambda)]$$
$$= \mathbb{E}_{q(z|x,\lambda)}\left[\nabla_\theta \log p(z,x|\theta)\right]$$
$$\overset{\text{MC}}{\approx} \nabla_\theta \log p(z,x|\theta) \quad \text{where } z \sim q(z|x,\lambda)$$

The gradient of the entropy term is 0 and the first term corresponds to the expected value of a stochastic gradient, thus MC gives us the unbiased estimate we need for optimisation of the generative model.

In this phase z if fixed to a random draw from $q(z|x,\lambda)$, from the point of view of the generative model it is as if z had been observed, so we can maximise $\log p(z,x|\theta)$.

This is simply supervised learning with imputed latent data!

Sleep Phase Objective

Objective

$$\arg\max_{\lambda} \ \mathbb{E}_{x\sim\mathcal{D}}\left[\mathrm{ELBO}_x(\theta,\lambda)\right]$$

$$= \arg\max_{\lambda} \ \mathbb{E}_{x\sim\mathcal{D}}\left[\mathbb{E}_{q(z|x,\lambda)}\left[\log p(z,x|\theta)\right] + \mathbb{H}[q(z|x,\lambda)]\right]$$

Gradient wrt λ for $x \sim \mathcal{D}$ (an observation)

$$\nabla_{\lambda}\mathbb{E}_{q(z|x,\lambda)}\left[\log p(z,x|\theta)\right] + \nabla_{\lambda}\mathbb{H}[q(z|x,\lambda)]$$

Let's change the objective!

When we turn to the gradient of the recognition model, as expected, things are not as easy.

Of course we know that we can re-express both gradients (recall that the entropy term is also an expected value) as expected gradients via the score function method. That's not how WS goes about this problem. Instead, WS changes the objective of optimisation.

This means that for the sleep phase, where we are supposed to learn the recognition model, we are not going to do VI. This is indeed a pity, since maximising the ELBO w.r.t. our choice of λ indeed minimises $\mathrm{KL}\left(q(z|x,\lambda)\ ||\ p(z|x,\theta)\right)$.

Sleep Phase (Convenient) Objective

Flip the direction of the KL

$$\arg\min_{\lambda} \mathbb{E}_{x\sim\mathcal{D}} \left[\mathrm{KL}\left(p(z|x,\theta) \;||\; q(z|x,\lambda) \right) \right]$$

The strategy for the sleep phase is to flip the KL around, that is, to assess the KL divergence of $p(z|x,\theta)$ from $q(z|x,\lambda)$.

Sleep Phase (Convenient) Objective

Flip the direction of the KL

$$
\begin{aligned}
&\arg\min_{\lambda} \mathbb{E}_{x\sim\mathcal{D}}\left[\mathrm{KL}\left(p(z|x,\theta)\;||\;q(z|x,\lambda)\right)\right] \\
&= \arg\min_{\lambda} \mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{p(z|x,\theta)}\left[\log p(z|x,\theta) - \log q(z|x,\lambda)\right]
\end{aligned}
$$

The strategy for the sleep phase is to flip the KL around, that is, to assess the KL divergence of $p(z|x,\theta)$ from $q(z|x,\lambda)$.

- See that this change is in some sense convenient. Assume we are able to sample from the true posterior, then we can get gradient estimates w.r.t. λ. Clearly this is only superficially simple, as we have no means to sample from the true posterior.

Sleep Phase (Convenient) Objective

Flip the direction of the KL

$$\arg\min_{\lambda} \mathbb{E}_{x\sim\mathcal{D}} \left[\mathrm{KL}\left(p(z|x,\theta) \mid\mid q(z|x,\lambda)\right)\right]$$
$$= \arg\min_{\lambda} \mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{p(z|x,\theta)} \left[\log p(z|x,\theta) - \log q(z|x,\lambda)\right]$$
$$\overset{\text{asm}}{=} \arg\max_{\lambda} \mathbb{E}_{p(x,z|\theta)} \left[\log q(z|x,\lambda)\right] - \underbrace{\mathbb{E}_{p(x,z|\theta)} \left[\log p(z|x,\theta)\right]}_{\text{constant}}$$

The strategy for the sleep phase is to flip the KL around, that is, to assess the KL divergence of $p(z|x,\theta)$ from $q(z|x,\lambda)$.

- See that this change is in some sense convenient. Assume we are able to sample from the true posterior, then we can get gradient estimates w.r.t. λ. Clearly this is only superficially simple, as we have no means to sample from the true posterior.
- Here is where WS makes a big assumption, it assumes that sampling from the data $x \sim \mathcal{D}$ is equivalent to sampling from the marginal of the model $x \sim p(x|\theta)$, this can only be true if our model perfectly reproduces the data generating process. This is very unlikely in general, since the data generating process is unknown to us, and it's particularly unlikely at the beginning of training.

Sleep Phase (Convenient) Objective

Flip the direction of the KL

$$\begin{aligned}
&\arg\min_{\lambda} \mathbb{E}_{x\sim\mathcal{D}} \left[\mathrm{KL}\left(p(z|x,\theta) \mid\mid q(z|x,\lambda)\right)\right] \\
&= \arg\min_{\lambda} \mathbb{E}_{x\sim\mathcal{D}} \mathbb{E}_{p(z|x,\theta)} \left[\log p(z|x,\theta) - \log q(z|x,\lambda)\right] \\
&\overset{\text{asm}}{=} \arg\max_{\lambda} \mathbb{E}_{p(x,z|\theta)} \left[\log q(z|x,\lambda)\right] - \underbrace{\mathbb{E}_{p(x,z|\theta)} \left[\log p(z|x,\theta)\right]}_{\text{constant}}
\end{aligned}$$

Gradient wrt λ

$$\begin{aligned}
&\nabla_{\lambda} \mathbb{E}_{p(x,z|\theta)} \left[\log q(z|x,\lambda)\right] \\
&= \mathbb{E}_{p(x,z|\theta)} \left[\nabla_{\lambda} \log q(z|x,\lambda)\right]
\end{aligned}$$

The strategy for the sleep phase is to flip the KL around, that is, to assess the KL divergence of $p(z|x,\theta)$ from $q(z|x,\lambda)$.

- See that this change is in some sense convenient. Assume we are able to sample from the true posterior, then we can get gradient estimates w.r.t. λ. Clearly this is only superficially simple, as we have no means to sample from the true posterior.
- Here is where WS makes a big assumption, it assumes that sampling from the data $x \sim \mathcal{D}$ is equivalent to sampling from the marginal of the model $x \sim p(x|\theta)$, this can only be true if our model perfectly reproduces the data generating process. This is very unlikely in general, since the data generating process is unknown to us, and it's particularly unlikely at the beginning of training.
- With this assumption in place, it's easy to express the gradient as an expected gradient.

Sleep Phase (Convenient) Objective

Flip the direction of the KL

$$
\begin{aligned}
&\arg\min_{\lambda} \mathbb{E}_{x\sim\mathcal{D}}\left[\mathrm{KL}\left(p(z|x,\theta)\ ||\ q(z|x,\lambda)\right)\right] \\
&= \arg\min_{\lambda} \mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{p(z|x,\theta)}\left[\log p(z|x,\theta) - \log q(z|x,\lambda)\right] \\
&\overset{\text{asm}}{=} \arg\max_{\lambda} \mathbb{E}_{p(x,z|\theta)}\left[\log q(z|x,\lambda)\right] - \underbrace{\mathbb{E}_{p(x,z|\theta)}\left[\log p(z|x,\theta)\right]}_{\text{constant}}
\end{aligned}
$$

Gradient wrt λ

$$
\begin{aligned}
&\nabla_{\lambda}\mathbb{E}_{p(x,z|\theta)}\left[\log q(z|x,\lambda)\right] \\
&= \mathbb{E}_{p(x,z|\theta)}\left[\nabla_{\lambda}\log q(z|x,\lambda)\right] \\
&\overset{\text{MC}}{\approx} \nabla_{\lambda}\log q(z|\tilde{x},\lambda) \quad \text{where } z\sim p(z|\theta) \\
&\qquad\qquad\qquad\qquad\qquad\quad \tilde{x}\sim p(x|z,\theta)
\end{aligned}
$$

The strategy for the sleep phase is to flip the KL around, that is, to assess the KL divergence of $p(z|x,\theta)$ from $q(z|x,\lambda)$.

- See that this change is in some sense convenient. Assume we are able to sample from the true posterior, then we can get gradient estimates w.r.t. λ. Clearly this is only superficially simple, as we have no means to sample from the true posterior.
- Here is where WS makes a big assumption, it assumes that sampling from the data $x\sim\mathcal{D}$ is equivalent to sampling from the marginal of the model $x\sim p(x|\theta)$, this can only be true if our model perfectly reproduces the data generating process. This is very unlikely in general, since the data generating process is unknown to us, and it's particularly unlikely at the beginning of training.
- With this assumption in place, it's easy to express the gradient as an expected gradient.
- An MC estimation is possible by ancestral sampling from $p(x,z|\theta)$. This gives us a *dream* (model-generated) observation.

Sleep Phase (Convenient) Objective

Flip the direction of the KL

$$\arg\min_{\lambda} \mathbb{E}_{x\sim\mathcal{D}} \left[\mathrm{KL}\left(p(z|x,\theta) \,||\, q(z|x,\lambda)\right)\right]$$
$$= \arg\min_{\lambda} \mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{p(z|x,\theta)} \left[\log p(z|x,\theta) - \log q(z|x,\lambda)\right]$$
$$\stackrel{\text{asm}}{=} \arg\max_{\lambda} \mathbb{E}_{p(x,z|\theta)} \left[\log q(z|x,\lambda)\right] - \underbrace{\mathbb{E}_{p(x,z|\theta)} \left[\log p(z|x,\theta)\right]}_{\text{constant}}$$

Gradient wrt λ

$$\nabla_\lambda \mathbb{E}_{p(x,z|\theta)} \left[\log q(z|x,\lambda)\right]$$
$$= \mathbb{E}_{p(x,z|\theta)} \left[\nabla_\lambda \log q(z|x,\lambda)\right]$$
$$\stackrel{\text{MC}}{\approx} \nabla_\lambda \log q(z|\tilde{x},\lambda) \quad \text{where } z \sim p(z|\theta)$$
$$\tilde{x} \sim p(x|z,\theta)$$

The strategy for the sleep phase is to flip the KL around, that is, to assess the KL divergence of $p(z|x,\theta)$ from $q(z|x,\lambda)$.

- See that this change is in some sense convenient. Assume we are able to sample from the true posterior, then we can get gradient estimates w.r.t. λ. Clearly this is only superficially simple, as we have no means to sample from the true posterior.
- Here is where WS makes a big assumption, it assumes that sampling from the data $x \sim \mathcal{D}$ is equivalent to sampling from the marginal of the model $x \sim p(x|\theta)$, this can only be true if our model perfectly reproduces the data generating process. This is very unlikely in general, since the data generating process is unknown to us, and it's particularly unlikely at the beginning of training.
- With this assumption in place, it's easy to express the gradient as an expected gradient.
- An MC estimation is possible by ancestral sampling from $p(x,z|\theta)$. This gives us a *dream* (model-generated) observation.

Sleep Phase (Convenient) Objective

Assumes fake data $\tilde{x}$ and latent variables z to be fixed random draws from $p(x, z|\theta)$ via

$$
\begin{aligned}
z &\sim p(z|\theta) \\
\tilde{x} &\sim p(x|z, \theta)
\end{aligned}
$$

and maximises $\log q(z|\tilde{x}, \lambda)$.

This is maximum likelihood estimation for the recognition model as if $z, \tilde{x}$ were observed.

Wake-sleep Algorithm

Advantages

- Simple layer-wise updates
- Amortised inference: all latent variables are inferred from the same weights λ

Drawbacks

- Inference and generative models are trained on different objectives
- Inference weights λ are updated on fake data $\tilde{x}$
- Generative weights are bad initially, giving wrong signal to the updates of λ

Though there are some instances of WS even in modern literature, its drawbacks are generally quite serious.

Frequentist VI

Variational Objective

$$\arg\max_{q(z)} \mathbb{E}_{q(z)} [\log p(x, z)] + \mathbb{H}(q(z))$$

This finds us the best posterior approximation for a **given model**.

Frequentist VI also optimises the model!

$$\arg\max_{q(z), p(x,z)} \mathbb{E}_{q(z)} [\log p(x, z)] + \mathbb{H}(q(z))$$

VI comes from the literature of Bayesian modelling, where it is known as Variational Bayes (VB). VB is concerned with the variational objective, i.e., ELBO maximisation w.r.t. a choice of posterior approximation $q(z)$.

In Frequentism, we make point estimates of model parameters. Whereas we can use the ELBO for that it should be noted that we are not optimising log-likelihood, as customary in MLE, rather we are optimising a lowerbound on it. There's no guarantee that an improvement in the lowerbound correlates with an improvement in log-evidence.

Coordinate Ascent Variational Inference

Frequentist VI can be performed via coordinate ascent. This can be done as a 2-step procedure.

1. Maximise (regularised) expected log-density.

$$\arg\max_{q(z)} \mathbb{E}_{q(z)} [\log p(x, z)] + \mathbb{H}(q(z))$$

2. Optimise generative model.

$$\arg\max_{p(x,z)} \mathbb{E}_{q(z)} [\log p(x, z)] + \underbrace{\mathbb{H}(q(z))}_{\text{constant}}$$

Think of our choice of approximation $q(z)$ and our choice of model $p(x, z)$ as coordinates.

We can keep one fixed an update the other. This is coordinate ascent VI.

Unconstrained (exact) optimisation

What's the solution to the following?

$$\arg\max_{q(z)\in\mathcal{Q}} \mathbb{E}_{q(z)}\left[\log p(x,z)\right] + \mathbb{H}\left(q(z)\right)$$

(assume $\mathcal{Q}$ is large enough a family)

Unconstrained (exact) optimisation

What's the solution to the following?

$$\arg\max_{q(z)\in\mathcal{Q}} \mathbb{E}_{q(z)}\left[\log p(x,z)\right] + \mathbb{H}\left(q(z)\right)$$

(assume $\mathcal{Q}$ is large enough a family)

The true posterior $p(z|x)$! Exactly because

$$\arg\max_{q(z)\in\mathcal{Q}} \mathrm{ELBO} = \arg\min_{q(z)\in\mathcal{Q}} \mathrm{KL}\left(q(z) \,||\, p(z|x)\right)$$

and KL is never negative and 0 iff $q(z) = p(z|x)$.

Recap: EM Algorithm

$$\text{E-step} \quad \arg\max_{q(z)} \mathbb{E}_{q(z)}\left[\log p(x,z)\right] + \mathbb{H}\left(p(z|x)\right)$$
$$= p(z|x)$$
$$\text{M-step} \quad \arg\max_{p(x,z)} \mathbb{E}_{p(z|x)}\left[\log p(x,z)\right] + \underbrace{\mathbb{H}\left(p(z|x)\right)}_{\text{constant}}$$

Expectation Maximisation (EM) is Frequentist variational inference where we solve ELBO maximisation w.r.t. $q(z)$ exactly, that is, we use the true posterior $p(z|x)$.

$$q(z) = p(z|x)$$
$$\mathrm{KL}\left(q(z) \,||\, p(z|x)\right) = 0$$

The implication is that we can only do EM for models whose marginals are already tractable (and thus do not require approximate inference).

When we train a discrete LVM with exact marginals via gradient-based MLE, we solve the marginal exactly (sidestepping the E-step), and the M-step approximately, via iterative gradient-based ascent.

Score Function Estimator: Variance

$$\frac{\partial}{\partial \lambda} \mathbb{E}_{q(z|x,\lambda)} [\log p(x|z,\theta)] = \mathbb{E}_{q(z|x,\lambda)} \left[\log p(x|z,\theta) \frac{\partial}{\partial \lambda} \log q(z|x,\lambda) \right]$$

Empirically this estimator often exhibits high variance.

- the magnitude of $\log p(x|z,\theta)$ varies widely
- the model likelihood does not contribute to direction of gradient

The simplest way to reduce variance of an MC estimator is to sample more times. But it's not very efficient.

Control variates

Intuition
To estimate $\mathbb{E}[f(z)]$ via Monte Carlo we compute the empirical average of $\hat{f}(z)$ where $\hat{f}(z)$ is chosen so that $\mathbb{E}[\hat{f}(z)] = \mathbb{E}[f(z)]$ and $\text{Var}(f) > \text{Var}(\hat{f})$.

Equivalent expectations

Let $\bar{f} = \mathbb{E}[f(z)]$ be an expectation of interest

Equivalent expectations

Let $\bar{f} = \mathbb{E}[f(z)]$ be an expectation of interest

- say we know $\bar{c} = \mathbb{E}[c(z)]$

Equivalent expectations

Let $\bar{f} = \mathbb{E}[f(z)]$ be an expectation of interest

- say we know $\bar{c} = \mathbb{E}[c(z)]$
- then for $\hat{f}(z) \triangleq f(z) - b(c(z) - \mathbb{E}[c(z)])$

Equivalent expectations

Let $\bar{f} = \mathbb{E}[f(z)]$ be an expectation of interest

- say we know $\bar{c} = \mathbb{E}[c(z)]$
- then for $\hat{f}(z) \triangleq f(z) - b(c(z) - \mathbb{E}[c(z)])$
  it holds that $\mathbb{E}[\hat{f}(z)] = \mathbb{E}[f(z)]$

Equivalent expectations

Let $\bar{f} = \mathbb{E}[f(z)]$ be an expectation of interest

- say we know $\bar{c} = \mathbb{E}[c(z)]$
- then for $\hat{f}(z) \triangleq f(z) - b(c(z) - \mathbb{E}[c(z)])$
  it holds that $\mathbb{E}[\hat{f}(z)] = \mathbb{E}[f(z)]$
- and $\operatorname{Var}(\hat{f}) = \operatorname{Var}(f) - 2b \operatorname{Cov}(f, c) + b^2 \operatorname{Var}(c)$

Choosing the control variate

1. $\hat{f}(z) \triangleq f(z) - b(c(z) - \mathbb{E}[c(z)])$
2. $\mathrm{Var}(\hat{f}) = \mathrm{Var}(f) - 2b\,\mathrm{Cov}(f, c) + b^2\,\mathrm{Var}(c)$

How do we choose b and $c(z)$?

Choosing the control variate

1. $\hat{f}(z) \triangleq f(z) - b(c(z) - \mathbb{E}[c(z)])$
2. $\mathrm{Var}(\hat{f}) = \mathrm{Var}(f) - 2b\,\mathrm{Cov}(f, c) + b^2\,\mathrm{Var}(c)$

How do we choose b and $c(z)$?

- If $f(z)$ and $c(z)$ are positively correlated, then we may reduce variance

Choosing the control variate

1. $\hat{f}(z) \triangleq f(z) - b(c(z) - \mathbb{E}[c(z)])$
2. $\mathrm{Var}(\hat{f}) = \mathrm{Var}(f) - 2b\,\mathrm{Cov}(f, c) + b^2\,\mathrm{Var}(c)$

How do we choose b and $c(z)$?

- If $f(z)$ and $c(z)$ are positively correlated, then we may reduce variance
- solving $\frac{\partial}{\partial b}\mathrm{Var}(\hat{f}) = 0$

Choosing the control variate

1. $\hat{f}(z) \triangleq f(z) - b(c(z) - \mathbb{E}[c(z)])$
2. $\mathrm{Var}(\hat{f}) = \mathrm{Var}(f) - 2b\,\mathrm{Cov}(f, c) + b^2\,\mathrm{Var}(c)$

How do we choose b and $c(z)$?

- If $f(z)$ and $c(z)$ are positively correlated, then we may reduce variance
- solving $\frac{\partial}{\partial b}\mathrm{Var}(\hat{f}) = 0$ yields $b^\star = {}^{\mathrm{Cov}(f,c)}/_{\mathrm{Var}(c)}$

Choosing the control variate

1. $\hat{f}(z) \triangleq f(z) - b(c(z) - \mathbb{E}[c(z)])$
2. $\text{Var}(\hat{f}) = \text{Var}(f) - 2b\,\text{Cov}(f,c) + b^2\,\text{Var}(c)$

How do we choose b and $c(z)$?

- If $f(z)$ and $c(z)$ are positively correlated, then we may reduce variance
- solving $\frac{\partial}{\partial b}\text{Var}(\hat{f}) = 0$ yields $b^\star = {}^{\text{Cov}(f,c)}/_{\text{Var}(c)}$

Of course, $\mathbb{E}[c(z)]$ must be known!

MC

We then use the estimate

$$\bar{f} \overset{\text{MC}}{\approx} \frac{1}{S}\left(\sum_{s=1}^{S} f(z^{(s)}) - bc(z^{(s)})\right) + b\bar{c}$$

MC

We then use the estimate

$$\bar{f} \overset{\text{MC}}{\approx} \frac{1}{S}\left(\sum_{s=1}^{S} f(z^{(s)}) - bc(z^{(s)})\right) + b\bar{c}$$

And recall that for us

$$f(z) = \log p(x|z,\theta)\frac{\partial}{\partial\lambda}\log q(z|x,\lambda)$$

and $z^{(s)} \sim q(z|x,\lambda)$

Expected score

The Expectation of the score function is 0.

$$\mathbb{E}_{q(z|x,\lambda)}\left[\frac{\partial}{\partial\lambda}\log q(z|x,\lambda)\right]$$

Expected score

The Expectation of the score function is 0.

$$\mathbb{E}_{q(z|x,\lambda)}\left[\frac{\partial}{\partial\lambda}\log q(z|x,\lambda)\right]$$
$$=\int q(z|x,\lambda)\frac{\partial}{\partial\lambda}\log q(z|x,\lambda)\mathrm{d}z$$

Expected score

The Expectation of the score function is 0.

$$
\begin{aligned}
&\mathbb{E}_{q(z|x,\lambda)}\left[\frac{\partial}{\partial\lambda}\log q(z|x,\lambda)\right] \\
&= \int q(z|x,\lambda)\frac{\partial}{\partial\lambda}\log q(z|x,\lambda)\mathrm{d}z \\
&= \int \frac{\partial}{\partial\lambda}q(z|x,\lambda)\mathrm{d}z
\end{aligned}
$$

Expected score

The Expectation of the score function is 0.

$$
\begin{aligned}
&\mathbb{E}_{q(z|x,\lambda)}\left[\frac{\partial}{\partial\lambda}\log q(z|x,\lambda)\right] \\
&= \int q(z|x,\lambda)\frac{\partial}{\partial\lambda}\log q(z|x,\lambda)\mathrm{d}z \\
&= \int \frac{\partial}{\partial\lambda} q(z|x,\lambda)\mathrm{d}z \\
&= \frac{\partial}{\partial\lambda}\int q(z|x,\lambda)\mathrm{d}z
\end{aligned}
$$

Expected score

The Expectation of the score function is 0.

$$
\begin{aligned}
&\mathbb{E}_{q(z|x,\lambda)}\left[\frac{\partial}{\partial\lambda}\log q(z|x,\lambda)\right] \\
&= \int q(z|x,\lambda)\frac{\partial}{\partial\lambda}\log q(z|x,\lambda)\mathrm{d}z \\
&= \int \frac{\partial}{\partial\lambda} q(z|x,\lambda)\mathrm{d}z \\
&= \frac{\partial}{\partial\lambda}\int q(z|x,\lambda)\mathrm{d}z \\
&= \frac{\partial}{\partial\lambda}1 = 0
\end{aligned}
$$

Baselines

With

$$f(z) = \log p(x|z,\theta)\frac{\partial}{\partial\lambda}\log q(z|x,\lambda)$$

and

$$c(z) = \frac{\partial}{\partial\lambda}\log q(z|x,\lambda)$$

we have

$$\hat{f}(z) =$$

Baselines

With

$$f(z) = \log p(x|z,\theta)\frac{\partial}{\partial\lambda}\log q(z|x,\lambda)$$

and

$$c(z) = \frac{\partial}{\partial\lambda}\log q(z|x,\lambda)$$

we have

$$\hat{f}(z) = (\log p(x|z,\theta) - b)\frac{\partial}{\partial\lambda}\log q(z|x,\lambda)$$

Baselines

With

$$f(z) = \log p(x|z,\theta)\frac{\partial}{\partial\lambda}\log q(z|x,\lambda)$$

and

$$c(z) = \frac{\partial}{\partial\lambda}\log q(z|x,\lambda)$$

we have

$$\hat{f}(z) = (\log p(x|z,\theta) - b)\frac{\partial}{\partial\lambda}\log q(z|x,\lambda)$$

b is known as *baseline* in RL literature.

Examples of baselines

- Moving average of $\log p(x|z, \theta)$ based on previous batches

Examples of baselines

- Moving average of $\log p(x|z, \theta)$ based on previous batches
- A trainable constant b

Examples of baselines

- Moving average of $\log p(x|z, \theta)$ based on previous batches
- A trainable constant b
- A neural network prediction based on x e.g. $b(x; \omega)$

Examples of baselines

- Moving average of $\log p(x|z, \theta)$ based on previous batches
- A trainable constant b
- A neural network prediction based on x e.g. $b(x; \omega)$
- The likelihood assessed at a deterministic point, e.g. $b(x) = \log p(x|z^\star, \theta)$ where $z^\star = \arg\max_z q(z|x, \lambda)$

Trainable baselines

Baselines are predicted by a regression model (e.g. a neural net).

The model is trained using an L_2-loss.

$$\min_{\omega} \left(b(x;\omega) - \log p(x|z,\theta) \right)^2$$

Summary

- In practice the score function estimator leads to high variance gradient estimates.
- We can design control variates that reduce estimator variance, yet do not bias the estimator!

References I

Jasmijn Bastings, Wilker Aziz, and Ivan Titov. Interpretable neural predictions with differentiable binary variables. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 2963–2977, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1284. URL https://www.aclweb.org/anthology/P19-1284.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet Allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=944919.944937. Publisher: JMLR.org.

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017. Publisher: Taylor & Francis.

References II

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2):263–311, 1993. URL https://www.aclweb.org/anthology/J93-2003.

Zoubin Ghahramani and Thomas L. Griffiths. Infinite latent feature models and the Indian buffet process. In Y. Weiss, B. Schölkopf, and J. C. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 475–482. MIT Press, 2006.

Will Grathwohl, Dami Choi, Yuhuai Wu, Geoff Roeder, and David Duvenaud. Backpropagation through the Void: Optimizing control variates for black-box gradient estimation. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=SyzKd1bCW.

References III

Evan Greensmith, Peter L. Bartlett, and Jonathan Baxter. Variance Reduction Techniques for Gradient Estimates in Reinforcement Learning. *Journal of Machine Learning Research*, 5(Nov):1471–1530, 2004. ISSN ISSN 1533-7928. URL https://www.jmlr.org/papers/v5/greensmith04a.html.

Shixiang Gu, Sergey Levine, Ilya Sutskever, and Andriy Mnih. MuProp: Unbiased backpropagation for stochastic neural networks. In *ICLR (poster)*, 2016. URL http://arxiv.org/abs/1511.05176. tex.cdate: 1451606400000 tex.crossref: conf/iclr/2016.

G. E. Hinton, P. Dayan, B. J. Frey, and R. M. Neal. The Wake-Sleep Algorithm for Unsupervised Neural Networks. *Science*, 268:1158–1161, 1995.

References IV

Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul. An Introduction to Variational Methods for Graphical Models. *Machine Learning*, 37(2):183–233, November 1999. ISSN 1573-0565. doi: 10.1023/A:1007665907178. URL https://doi.org/10.1023/A:1007665907178.

Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised Learning with Deep Generative Models. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3581–3589. Curran Associates, Inc., 2014.

References V

Tao Lei, Regina Barzilay, and Tommi Jaakkola. Rationalizing Neural Predictions. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 107–117, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1011. URL https://www.aclweb.org/anthology/D16-1011.

Runjing Liu, Jeffrey Regier, Nilesh Tripuraneni, Michael Jordan, and Jon Mcauliffe. Rao-blackwellized stochastic gradients for discrete distributions. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *ICML*, volume 97 of *Proceedings of machine learning research*, pages 4023–4031, Long Beach, California, USA, June 2019. PMLR. URL http://proceedings.mlr.press/v97/liu19c.html. tex.pdf: http://proceedings.mlr.press/v97/liu19c/liu19c.pdf.

References VI

Thomas P. Minka. Expectation propagation for approximate bayesian inference. In *Proceedings of the seventeenth conference on uncertainty in artificial intelligence*, UAI'01, pages 362–369, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1-55860-800-1. Number of pages: 8 Place: Seattle, Washington.

Andriy Mnih and Karol Gregor. Neural Variational Inference and Learning in Belief Networks. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages II–1791–II–1799. JMLR.org, 2014. event-place: Beijing, China.

Andriy Mnih and Danilo Rezende. Variational inference for monte carlo objectives. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *ICML*, volume 48 of *Proceedings of machine learning research*, pages 2188–2196, New York, New York, USA, June 2016. PMLR. URL http://proceedings.mlr.press/v48/mnihb16.html. tex.pdf: http://proceedings.mlr.press/v48/mnihb16.pdf.

References VII

Shakir Mohamed, Mihaela Rosca, Michael Figurnov, and Andriy Mnih. Monte Carlo Gradient Estimation in Machine Learning. *CoRR*, abs/1906.10652, 2019. URL http://arxiv.org/abs/1906.10652.

Rajesh Ranganath, Sean Gerrish, and David Blei. Black Box Variational Inference. In Samuel Kaski and Jukka Corander, editors, *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, volume 33 of *Proceedings of Machine Learning Research*, pages 814–822, Reykjavik, Iceland, April 2014. PMLR. URL http://proceedings.mlr.press/v33/ranganath14.html.

Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. Self-Critical Sequence Training for Image Captioning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 1179–1195. IEEE Computer Society, 2017. doi: 10.1109/CVPR.2017.131. URL https://doi.org/10.1109/CVPR.2017.131.

References VIII

Miguel Rios, Wilker Aziz, and Khalil Sima'an. Deep Generative Model for Joint Alignment and Word Representation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1011–1023, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1092. URL https://www.aclweb.org/anthology/N18-1092.

John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel. Gradient estimation using stochastic computation graphs. In *Advances in neural information processing systems*, pages 3528–3536, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

References IX

George Tucker, Andriy Mnih, Chris J Maddison, John Lawson, and Jascha Sohl-Dickstein. REBAR: Low-variance, unbiased gradient estimates for discrete latent variable models. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 2627–2636. Curran Associates, Inc., 2017.

Aki Vehtari, Andrew Gelman, Tuomas Sivula, Pasi Jylänki, Dustin Tran, Swupnil Sahai, Paul Blomstedt, John P Cunningham, David Schiminovich, and Christian P Robert. Expectation propagation as a way of life: A framework for bayesian inference on partitioned data. *Journal of Machine Learning Research*, 21(17):1–53, 2020.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. HMM-Based word alignment in statistical translation. In *COLING 1996 volume 2: The 16th international conference on computational linguistics*, 1996. URL `https://www.aclweb.org/anthology/C96-2141`.

References X

Weiyue Wang, Derui Zhu, Tamer Alkhouli, Zixuan Gan, and Hermann Ney. Neural Hidden Markov Model for Machine Translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 377–382, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-2060. URL https://www.aclweb.org/anthology/P18-2060.

Ronald J. Williams. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*, 8(3-4): 229–256, May 1992. ISSN 0885-6125. doi: 10.1007/BF00992696. URL https://doi.org/10.1007/BF00992696.

References XI

Chunting Zhou and Graham Neubig. Multi-space variational encoder-decoders for semi-supervised labeled sequence transduction. In *Proceedings of the 55th annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 310–320, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1029. URL https://www.aclweb.org/anthology/P17-1029.