Continuous-Depth Bayesian Neural Networks

Winnie Xu [1] **Ricky T. Q. Chen** [1 2] **Xuechen Li** [3] **David Duvenaud** [1 2 3]

Abstract

Taking the infinitesimal limit of residual networks gives neural networks whose hidden unit activations are governed by ordinary differential equations (ODEs). Uncertainty over the weights of each of infinitely many layers endows Bayesian neural networks whose hidden unit activations are governed by stochastic differential equations (SDEs). Building upon efficient algorithms for gradient-based variational inference in SDEs, we explore the use of infinite-dimensional stochastic variational inference in this model. This approach gives arbitrarily-expressive non-Gaussian approximate posteriors. We also extend results from the finite-dimensional case to yield gradient estimators that achieve zero variance as the approximate posterior approaches the true posterior.

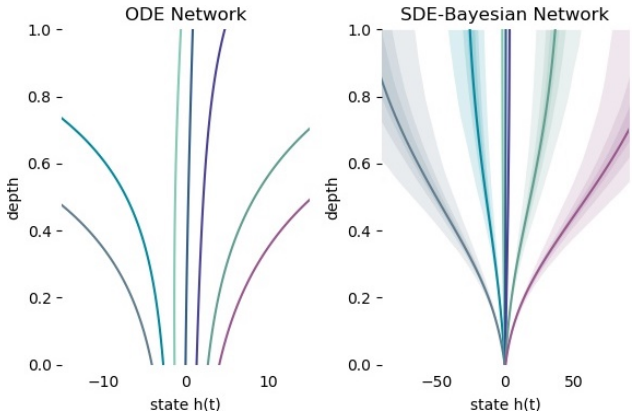

Figure 1. Continuous transformation of the hidden state. $Left$: A continuous-depth ODE network with deterministic transformation. $Right$: Our SDE BNN performs Bayesian inference over time-dependent weights of the ODE network, modeled as a *stochastic process*. $Both$: Each black curve is for a separate initial state. Percentiles are displayed for each initial state.

1. Introduction

Bayesian neural networks are a principled approach to handling uncertainty when training deep neural networks (MacKay, 1992; Hinton & Van Camp, 1993; Neal, 2012). Instead of performing maximum likelihood estimation and obtaining only one set of weights, the Bayesian paradigm frames learning as posterior inference. Specifically, given a data set $\mathcal{D}_N$ of size N, we approximate the posterior distribution over model weights w,

$$p(w|\mathcal{D}_N) \propto p(\mathcal{D}_N|w)p(w) \tag{1}$$

where $p(\mathcal{D}|w)$ is the likelihood of observing the data set. Here, we will assume a supervised learning setting, where each data point consists of an independent variable x and a dependent variable y. Assuming each data point is independent, the likelihood then factors as $p(\mathcal{D}|w) = \prod_{i=1}^{N} p(y_i|x_i, w)$. The posterior predictive distribution then provides us with a distribution over predictions of unobserved (test) data, by marginalizing over all possible parameter settings that fit the training set $\mathcal{D}_N$.

$$p(y|x, \mathcal{D}_N) = \int_w p(y|x, w)p(w|\mathcal{D}_N)\, dw \tag{2}$$

However, the posterior distribution $p(w|\mathcal{D}_N)$ is generally intractable for deep neural networks. In this work, we focus on the variational approach, where we train a variational posterior over the weights. This is typically done by minimizing a Kullback–Leibler (KL) divergence using stochastic gradient descent (SGD). Concretely, the training objective is the evidence lower bound (ELBO):

$$\begin{aligned} L(\phi) =& N\mathbb{E}_{w\sim q(w),(x,y)\sim\mathcal{D}_N}\left[\log p(y|x, w)\right] \\ &- \mathrm{KL}(q(w)||p(w))) \end{aligned} \tag{3}$$

Maximizing this objective is equivalent to minimizing $\mathrm{KL}(q_\phi(w)||p(w|\mathcal{D}_N))$.

Variational inference (VI) turns posterior inference into a stochastic optimization problem. One of the main technical challenges of VI is choosing a parametric family of approximate posteriors $q_\phi(w)$ that is tractable to sample from and

[1]Department of Computer Science, University of Toronto, Canada [2]Vector Institute for Artificial Intelligence [3]Google Research. Correspondence to: Winnie Xu <winnie.xu@mail.utoronto.ca>.

Presented at the ICML 2020 Workshop on Uncertainty and Robustness in Deep Learning.

evaluate, while being flexible enough to approximate the true posterior well. Several works have proposed variational posteriors which take into account a notion of "layers" as building blocks of deep neural nets (Zhang et al., 2017; Mishkin et al., 2018).

In this work, we take the limit of an infinitely-deep Bayesian neural network, with separate weights for each layer. The distribution over weights thus becomes a stochastic process, as does the activation of the hidden units. Our resulting model is a generalization of the Neural ODE framework (Chen et al., 2018). The prior $p(w(\cdot))$ and and variational posterior over weights $q_\phi(w(\cdot))$ are parameterized through stochastic differential equations, resulting in stochastic processes. This approach yields the following potential benefits:

1. The variational posterior can be made arbitrarily expressive, by simply making the dynamics that parameterize the SDE more expressive.

2. We can use adaptive black-box higher-order SDE solvers to evaluate these infinitely-deep neural networks to any desired accuracy, adjustable at test time.

3. Using the recently developed stochastic adjoint method (Li et al., 2020), we can effectively train using $\mathcal{O}(1)$ memory cost.

2. Background

2.1. Neural Ordinary Differential Equations

Neural ODEs (Chen et al., 2018) parameterize the change in activations through an ordinary differential equation (ODE).

$$dh(t) = f(h(t), t)\, dt \tag{4}$$

where an initial value is typically chosen to be the independent variable $h(t_0) = x$. The output of the network is the value of h at time t_1,

$$h(t_1) = h(t_0) + \int_{t_0}^{t_1} f(h(t), t)\, dt \tag{5}$$

If this integral is discretized using the Euler scheme, the discretization is the familiar residual network architecture (He et al., 2016). However, if treated as a continuous transformation, this has some modeling and implementation benefits.

1. Adaptive computation time. With the use of adaptive ODE solvers, the actual amount of compute varies depending on the complexity of the ODE. This allows the model to adapt its computation time to the complexity of the data.

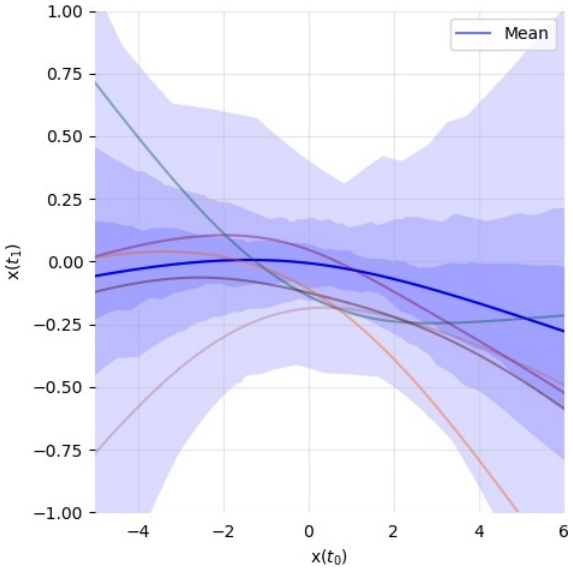

Figure 2. Samples from prior distribution over functions, using an augmented hidden state to allow for non-monotonicity.

2. Constant-memory gradient computation. The adjoint method allows computing gradients by solving a time-reversed ODE. None of the internal activations need to be stored, rendering this approach agnostic to the ODE solver.

2.1.1. Augmented Neural ODEs

Since solutions of ODEs are constrained in the sense that trajectories based on different initial values cannot intersect, Dupont et al. (2019) suggested adding extra dimensions to the ODE. This allows us to keep the benefits of continuous-depth modeling while being expressive enough to model arbitrary transformations (Zhang et al., 2019).

2.2. Stochastic Differential Equations

We specify a prior for the weights of an infinitesimally-layered network using stochastic differential equations (SDEs). SDEs can be written as:

$$dw(t) = f(w(t), t)\, dt + g(w(t), t)\, dB(t) \tag{6}$$

where $B(t)$ is a Brownian motion, intuitively a continuous-time random walk. Dependence of $w(t)$ on the Brownian motion means that $w(t)$ is stochastic given $w(t_0)$.

Li et al. (2020) derived backwards-in-time stochastic adjoint equations for backpropagating through sample paths of SDEs, conditioned on a Brownian motion sample $B(t)$. This provides an infinite-dimensional analogue of the "reparameterization gradient" for training.

Furthermore, it can also be shown (Li et al., 2020) that if $p(w(t))$ and $q(w(t))$ are both SDEs with the same diffusion

coefficient g, then the log-density ratio $\log(p(w(\cdot)/q(w(\cdot))$ can be computed for a given sample path $w(\cdot)$. This lets us compute an unbiased estimate of the KL divergence between the prior and posterior by another SDE integration:

$$\mathrm{KL}(q||p) = \mathbb{E}_{w(\cdot)\sim q}\left[\int_{t_0}^{t_1} \frac{1}{2}\left|u(w(t),t)\right|^2 \, dt\right] \tag{7}$$

where $u(w(t),t)$ satisfies

$$g(w(t),t)u(w(t),t) = f_w(w(t),t) - f_p(w(t),t) \tag{8}$$

where $f_w(w(t),t)$ is the drift for $q_\phi(w(\cdot))$ and $f_p(w(t),t)$ is the drift for $p(w(\cdot))$. This KL divergence is only finite if the prior and posterior share the same diffusion function $g(w(t),t)$.

3. Infinitely deep Bayesian residual nets

Taking the infinitesimal limit of a residual network:

$$h(t+\epsilon) = h(t) + \epsilon f_h(h(t), w(t), t) \tag{9}$$

as $\epsilon \to 0$ gives a differential equation that describes the hidden unit evolution as a function of depth, where depth is indexed by time, t. The dynamics function f_h is given by a small neural network with weights w.

Prior process on weights Putting on a prior on the weights of infinitely-many layers can be done by specificing a stochastic process indexed by depth. For simplicity, we place an Ornstein–Uhlenbeck (OU) process as the prior, i.e.

$$f_p(w(t),t) = -w(t), \qquad g_p(w(t),t) = \sigma \tag{10}$$

where σ is a hyperparameter. The marginal distributions $w(t)$ are Gaussian distributed with zero mean and variance σ^2.

Approximate posterior process on weights We parameterize the approximate posterior on weights $q_\phi(w(\cdot))$ implicitly using another SDE:

$$\begin{aligned} f_w(w(t),t,\phi) &= f_p(w(t),t) + \mathrm{NN}(w(t),t,\phi), \\ g_w(w(t),t) &= \sigma \end{aligned} \tag{11}$$

This SDE shares the same diffusion function as the prior. Its drift function f_w is parameterized by a small neural network with weights ϕ. This posterior is non-Gaussian, and its expressive capacity of the approximate posterior can be made larger by increasing the size of f_w. Figure 3 shows 2-dimensional marginals of the approximate posterior.

Evaluating the network Evaluating our network at a given input requires sampling a weight path $w(\cdot)$ from the approximate posterior, and evaluating the network activations $h(\cdot)$ given those weights and the input. Both steps require solving a differential equation. Luckily, both steps can be done simultaneously in a single call to an SDE solver:

$$d\begin{bmatrix} w(t) \\ h(t) \end{bmatrix} = \begin{bmatrix} f_w(w(t),t,\phi) \\ f_h(h(t),w(t),t) \end{bmatrix} dt + \begin{bmatrix} g_w(w(t),t) \\ \mathbf{0} \end{bmatrix} dB(t) \tag{12}$$

with $h(0) = x$ and $w(0) = w_0$ being learned.

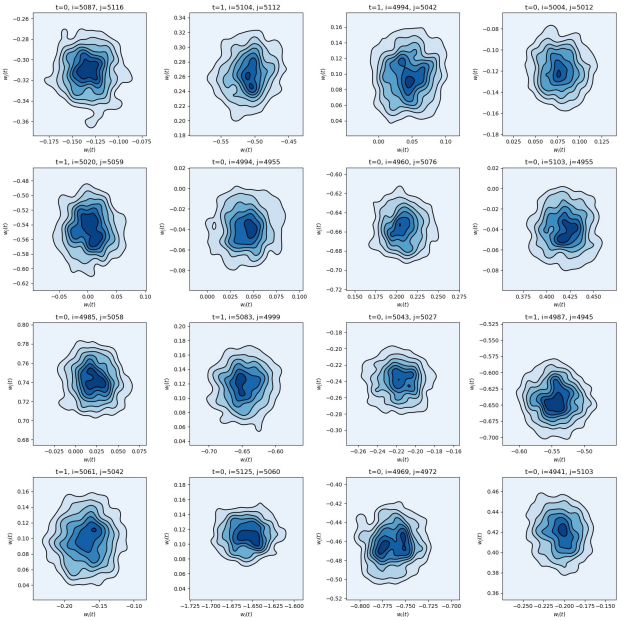

Figure 3. Samples from bias slices fitted to a Gaussian KDE imply non-Gassianity given an intial Gaussian prior (Brownian motion). Multi-modality is difficult to show in high dimensional settings (here, 5132 weights), thus it remains to be investigated whether these features arise from sampling noise or otherwise.

Likelihood Our approach also requires placing a likelihood $\log p(y|x, w(\cdot)) = \log p(y|h(1))$ on the output y given the final activations $h(1)$, for instance a Laplace likelihood for regression, or `softmax`. The parameters of this likelihood can also be learned at training time.

Training objective We use gradient-based optimization to maximize the infinite-dimensional ELBO:

$$\begin{aligned} L(\phi) = \mathbb{E}_{w(\cdot)\sim q_\phi,(x,y)\sim\mathcal{D}_N}[&\log p(y|x,w(\cdot)) \\ &- \int_{t_0}^{t_1} \frac{1}{2}\left|u(w(t),t)\right|^2 \, dt\Big] \end{aligned} \tag{13}$$

where $u(w(t),t) = \frac{1}{\sigma}(f_w(w(t),t,\phi) - f_p(w(t),t))$.

The sampled weights, the hidden activations, and the training objective are all computed simultaneously using a single

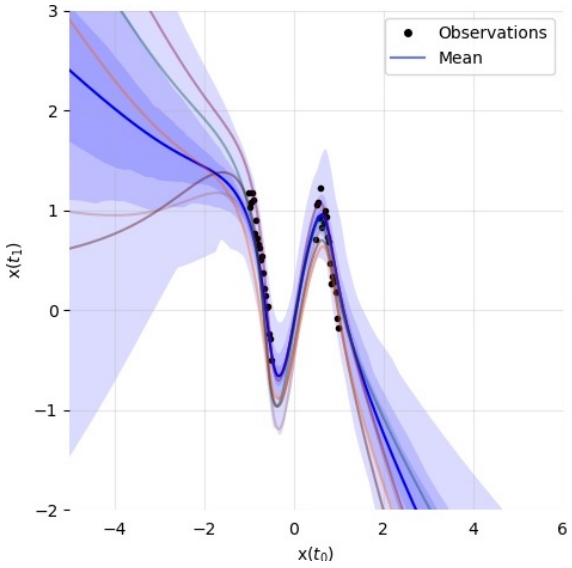

Figure 4. Continuous-depth Bayesian neural networks can have expressive approximate posteriors. Blue shaded areas show posterior percentiles, and colored lines show samples taken from the learned approximate posterior.

call to an SDE solver.

3.1. Variance-reduced Gradients

Roeder et al. (2017) showed that when training with the reparameterization gradient, a lower variance gradient estimator can be constructed by removing a score function term that has expectation zero. The variance of this gradient estimator approaches zero as the approximate posterior approaches the true posterior. We adapt this idea to the SDE setting by replacing the original estimator of the KL with the following surrogate objective

$$\int_{t_0}^{t_1} \frac{1}{2}|u(w(t),t,\phi)|^2 dt + \int_{t_0}^{t_1} u(w(t),t,\texttt{sg}(\phi))dB(t),$$

where $w(\cdot) \sim q_\phi(\cdot)$, and $\texttt{sg}(x)$ (the `stop_gradient` function) renders x a constant and with respect to which gradient accumulation is stopped.

Because our approximate posterior $q_\phi(w(\cdot))$ can be made arbitrarily expressive, we conjecture that our approach can achieve arbitrarily low gradient variance towards the end of training if the network parameterizing f_w is made expressive enough. See Appendix 6.2 for a heuristic derivation.

4. Experiments

We demonstrate the SDE BNN and stochastic variational inference framework on two supervised learning tasks. During training we averaged over 100 weight process samples at each iteration.

4.1. Model Architecture

The approximate posterior drift f_w is parameterized by a two layer network with 64 hidden units. All layers are time dependent and use the Swish activation (Ramachandran et al., 2017). The posterior drift f_w uses the same architecture as the network for the dynamics drift f_h. To stabilize training, we initialized the posterior drift f_w to be equal to the prior drift by setting the parameters of last linear layer in the neural network to zero.

4.2. Toy 1D data

Figure 4 shows that our model learns a reasonable approximate posterior on a synthetic non-monotonic 1D dataset. We augment our input by 2 extra dimensions (Dupont et al., 2019) to provide additional sampling flexibility to our prior process which improves overall performance. Without additional augmentation, the learned SDE-BNN can otherwise only sample monotonic functions.

How expressive is the approximate posterior? The KL divergence is only finite when the approximate posterior diffusion is the same as the prior. This may seem like a large limitation, but the state-dependent drift f_h can make the marginal variance arbitrarily small. Tzen & Raginsky (2019a;b) provide universality results for this setting.

SDE Solvers In this initial work, we use a simple Euler-Maruyama solver step size of 0.01, i.e.

$$s_{t+1} = s_t + h(t)f(h(t),w(t),t) + \sqrt{h(t)}\epsilon g(w(t),t) \tag{14}$$

where $\epsilon \sim \mathcal{N}(0,1)$. We use the Adam optimizer (Kingma & Ba, 2015) with a learning rate of 1e-3. The diffusion function is a constant taken to be 0.2 for the monotonic problem and 0.5 for the non-monotonic.

5. Conclusion

We demonstrated that an infinitely deep Bayesian neural network (Peluchetti & Favaro, 2019) may be trained with an SDE framework. In particular, our method does approximate inference by optimizing the ELBO while utilizing reverse-mode auto-differentiation rather than doing so via a rejection sampling method. We plan to adopt more complicated adaptive SDE solvers with memory-efficient gradient computation in the future for scaling up to higher dimensional data. Our approach gives arbitrarily expressive non-Gaussian approximate posteriors. This approach dovetails with the variance reduction approach (Roeder et al., 2017) to potentially give arbitrarily small gradients towards the end of training. It also allows one to trade off evaluation speed and numerical accuracy at test time.

References

Chen, R. T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. Neural ordinary differential equations. *Advances in Neural Information Processing Systems*, 2018.

Dupont, E., Doucet, A., and Teh, Y. W. Augmented neural odes. In *NeurIPS*, 2019.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hinton, G. E. and Van Camp, D. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pp. 5–13, 1993.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

Li, X., Wong, T.-K. L., Chen, R. T., and Duvenaud, D. Scalable gradients for stochastic differential equations. *arXiv preprint arXiv:2001.01328*, 2020.

MacKay, D. J. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.

Mishkin, A., Kunstner, F., Nielsen, D., Schmidt, M., and Khan, M. E. Slang: Fast structured covariance approximations for bayesian deep learning with natural gradient. In *Advances in Neural Information Processing Systems*, pp. 6245–6255, 2018.

Neal, R. M. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

Peluchetti, S. and Favaro, S. Infinitely deep neural networks as diffusion processes. *arXiv: Machine Learning*, 2019.

Ramachandran, P., Zoph, B., and Le, Q. V. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

Roeder, G., Wu, Y., and Duvenaud, D. K. Sticking the landing: Simple, lower-variance gradient estimators for variational inference. In *Advances in Neural Information Processing Systems*, pp. 6925–6934, 2017.

Tzen, B. and Raginsky, M. Neural stochastic differential equations: Deep latent gaussian models in the diffusion limit. *arXiv preprint arXiv:1905.09883*, 2019a.

Tzen, B. and Raginsky, M. Theoretical guarantees for sampling and inference in generative models with latent diffusions. *arXiv preprint arXiv:1903.01608*, 2019b.

Zhang, G., Sun, S., Duvenaud, D., and Grosse, R. Noisy natural gradient as variational inference. *arXiv preprint arXiv:1712.02390*, 2017.

Zhang, H., Gao, X., Unterman, J., and Arodz, T. Approximation capabilities of neural ordinary differential equations. *arXiv preprint arXiv:1907.12998*, 2019.

6. Appendix

Notation. Denote as ϕ the vector of variational parameters, f_q as the approximate posterior on weights, f_p as the prior on weights, f_h as the dynamics of hidden units, and σ as the diffusion function. Denote the Euclidean norm of a vector u by $|u|$. For function f denote its Jacobian as ∇f.

6.1. Derivation of an Alternative Monte Carlo Estimator

The goal of this section is to derive a Monte Carlo estimator of the KL-divergence on path space that is similar to the *fully Monte Carlo* estimator described in (Roeder et al., 2017). This will serve as the basis for the subsequent heuristic derivation of the continuous-time sticking-the-landing trick.

Let w_0 be a fixed initial state. Let $w_1, ..., w_N$ be states at times $\Delta t, 2\Delta t, \ldots, N\Delta t = T$ generated by the Euler discretization:

$$w_{i+1} = w_i + f_q(w_i)\Delta t + \sigma(w_i)(B(t+\Delta t) - B(t)) \tag{15}$$

$$= w_i + f_q(w_i)\Delta t + \sigma(w_i)\Delta t^{1/2}\epsilon_{i+1}, \quad \epsilon_{i+1} \sim \mathcal{N}(0,1). \tag{16}$$

where $B(\cdot)$ is the Brownian motion. This implies conditional on the previous state, the current state is normally distributed:

$$w_{i+1}|w_i \sim \mathcal{N}(w_i + f_q(w_i)\Delta t, \sigma(w_i)^2\Delta t).$$

Thus, the log-densities can be evaluated as

$$\log q(w_{i+1}|w_i) = -\frac{1}{2}\log(2\pi\sigma(w_i)^2\Delta t) - \frac{1}{2}\frac{(w_{i+1} - (w_i + f_q(w_i)\Delta t))^2}{\sigma(w_i)^2\Delta t}, \quad i = 0, \ldots N-1. \tag{17}$$

On the other hand, if at any time, the next state was generated from the current state based on the prior process, we would have the following log-densities:

$$\log p(w_{i+1}|w_i) = -\frac{1}{2}\log(2\pi\sigma(w_i)^2\Delta t) - \frac{1}{2}\frac{(w_{i+1} - (w_i + f_p(w_i)\Delta t))^2}{\sigma(w_i)^2\Delta t}, \quad i = 0, \ldots N-1. \tag{18}$$

Now, we substitute the form of w_{i+1} based on (15) into the log-density equations (17) and (18) and obtain

$$\log q(w_{i+1}|w_i) = -\frac{1}{2}\log(2\pi\sigma(w_i)^2\Delta t) - \frac{1}{2}\epsilon_{i+1}^2,$$

$$\log p(w_{i+1}|w_i) = -\frac{1}{2}\log(2\pi\sigma(w_i)^2\Delta t) - \frac{1}{2}\left(\frac{(f_q(w_i) - f_p(w_i))^2}{\sigma(w_i)^2}\Delta t + \frac{2(f_q(w_i) - f_p(w_i))\epsilon_{i+1}}{\sigma(w_i)}\Delta t^{1/2} + \epsilon_{i+1}^2\right).$$

The KL divergence on the path space could then be regarded as a sum of infinitely many KL-divergences between Gaussians:

$$\lim_{N\to\infty}\sum_{i=0}^{N}\mathbb{E}_{w_i}\left[\mathrm{KL}\left(q(w_{i+1}|w_i)||p(w_{i+1}|w_i)\right)\right] \tag{19}$$

$$= \lim_{N\to\infty}\sum_{i=0}^{N}\mathbb{E}_{w_i}\left[\mathbb{E}_{w_{i+1}\sim q(w_{i+1}|w_i)}\left[\log\frac{q(w_{i+1}|w_i)}{p(w_{i+1}|w_i)}\right]\right] \tag{20}$$

$$= \lim_{N\to\infty}\sum_{i=0}^{N}\mathbb{E}_{w_i}\left[\mathbb{E}_{\epsilon_{i+1}}\left[\frac{(f_q(w_i) - f_p(w_i))^2}{2\sigma(w_i)^2}\Delta t + \frac{(f_q(w_i) - f_p(w_i))}{\sigma(w_i)}\Delta t^{1/2}\epsilon_{i+1}\right]\right] \tag{21}$$

$$= \mathbb{E}\left[\frac{1}{2}\int_0^T |u_t|^2\,\mathrm{d}t + \int_0^T u_t\,\mathrm{d}B_t\right]. \tag{22}$$

6.2. Sticking-the-landing in Continuous Time

For a non-sequential latent variable model, the sticking-the-landing (STL) trick removes from the fully Monte Carlo ELBO estimator a score function term of the form $\partial \log q(z,\phi)/\partial\phi$, where z is sampled using the reparameterization trick and

may depend on ϕ. The score function term has 0 expectation, but may affect the variance of the gradient estimator for the inference distribution's parameters.

Here, we exploit this intuition and apply it to each step before taking the limit. More precisely, we apply the STL trick to estimate the gradient of $\mathrm{KL}(q(w_{i+1}|w_i)||p(w_{i+1}|w_i))$ for $i = 1, 2, \ldots, N$, and thereafter take the limit as the mesh size of the discretization goes to 0. For each individual term, the score function term to be removed is

$$\begin{aligned}\frac{\partial}{\partial\phi}\log q(w_{i+1}|w_i,\phi) =& -\frac{1}{2\sigma^2(w_i)\Delta t}\frac{\partial}{\partial\phi}\left[(w_{i+1}-(w_i+f_q(w_i,\phi)\Delta t))^2\right]\\ =&\frac{\partial}{\partial\phi}\left[\frac{f_q(w_i,\phi)}{\sigma(w_i)}\right]\epsilon_{i+1}\Delta t^{1/2}.\end{aligned}$$

Now, we sum up all of these terms and take the limit as $\Delta t \to 0$. This gives us

$$\begin{aligned}\lim_{N\to\infty}\sum_{i=0}^{N}\mathbb{E}_{x_i}\left[\mathbb{E}_{x_{i+1}\sim q(x_{i+1}|x_i)}\left[\frac{\partial}{\partial\phi}\log q(x_{i+1}|x_i)\right]\right] &= \lim_{N\to\infty}\sum_{i=0}^{N}\mathbb{E}_{x_i}\left[\mathbb{E}_{\epsilon_{i+1}}\left[\frac{\partial}{\partial\phi}\left[\frac{f_q(x_i,\phi)}{\sigma(x_i)}\right]\epsilon_{i+1}\Delta t^{1/2}\right]\right]\\ &= \mathbb{E}\left[\int_0^T\frac{\partial}{\partial\phi}\left[\frac{f_q(x_t,\phi)}{\sigma(x_t)}\right]\mathrm{d}B_t\right]\\ &= \mathbb{E}\left[\int_0^T\frac{\partial}{\partial\phi}\left[u_t\right]\mathrm{d}B_t\right].\end{aligned}$$

Removing this term from the fully Monte Carlo estimator in (22) gives rise to the following estimator of a surrogate objective that facilitates implementation:

$$\widehat{\mathrm{ELBO}} = \log p(Y|X,\{x_t\}_{t\in[0,T]}) - \int_{t_0}^{t_1}\frac{1}{2}|u(x_t,t,\phi)|^2\,\mathrm{d}t - \int_{t_0}^{t_1}u(x_t,t,\mathtt{sg}(\phi))\,\mathrm{d}B_t, \quad x(\cdot)\sim q_\phi().$$