SURROGATED ASSISTED BAYESIAN NEURAL NETWORK FOR GEOLOGICAL MODELS

Sean Luo

Supervisors: Rohitash Chandra, Richard Scalzo

School of Mathematics and Statistics
UNSW Sydney

June 2022

Submitted in partial fulfillment of the requirements of the degree of
Bachelor of Science with Honours

Plagiarism statement

I declare that this thesis is my own work, except where acknowledged, and has not been submitted for academic credit elsewhere.

I acknowledge that the assessor of this thesis may, for the purpose of assessing it:

- Reproduce it and provide a copy to another member of the University; and/or,
- Communicate a copy of it to a plagiarism checking service (which may then retain a copy of it on its database for the purpose of future plagiarism checking).

I certify that I have read and understood the University Rules in respect of Student Academic Misconduct, and am aware of any potential plagiarism penalties which may apply.

By signing this declaration I am agreeing to the statements and conditions above.

Signed: _____ Date: _____

Acknowledgements

By far the greatest thanks must go to my supervisor for the guidance, care and support they provided.

Thanks to my family for supporting me to advance in higher education.

Thanks go to Fred Flintstone and Robert Taggart for allowing this thesis style to be shamelessly copied.

Sean Luo, Day Month Year.

Abstract

Bayesian learning methods natively offer insights into dependencies between parameters and posterior probability density estimation in complex systems. Markov Chain Monte Carlo methods compare to variational inference methods offers an asymptotically exact approximation to the posterior. However it suffers from exponetially high computational cost in multimodal high dimensional distributions due to high rejection rate from proposal distributions such as Langevin Dynamic or network architecture. This thesis propose to sample from a surrogate proposal distribution approximated by a Generative Adversarial Network trained from previously accepted proposals from parallelized tempered training such that it increases the acceptance rate and avoid the expensive forward pass that is required for Langevin Dynamics.

The results hopefully don't prove to be inconclusive.

Finally this method is demonstrated on a large geological model on radar data.

Contents

Chapter 1

Introduction (Currently a collection of messy notes)

Application of bayesian methods

In the past decade there has being some progression in the field of bayesian deep learning. Bayesian deep learning offers an intrinsic way to ensemble models that helps to quantify the posterior uncertainty in model parameters and prediction. Applying tradition bayesian inference techniques such as Monte Carlo sampling in deep neural networks have multiple challenges, mainly revolving around computational expenses due to the evaluation speed of large dataset, exponentially increasing rejection rate in MCMC sampler as the parameter space dimensions grows with large model, difficulty to evaluate uncertainty in realtime data, efficient proposal convergence and others. Variational bayesian inference methods are faster than sampling, but it does not offer an exact approximation of the target distribution.
Ideas explored:

Pure Bayesian:

1. Using stochastic mini batches with Langevin dynamic by Welling et al [1].
2. Using langevin dynamic Dynamic as proposal to predict time series by chandra et al [2]
3. Using parallel tempering to speed up exploration and exploitation by chandra [3]
4. Using surrogate models to simulate the posterior likelihood $p(D|\theta)$ to speed up evaluation, by chandra. [4]
5. Surrogate models are selected from variantional inference category, including, GAN, VAE, and Normaling Flow.
6. Train a WGAN to approximate the joint prior distribution of parameter and state, and sample from latent space instead. This is useful when the domain has high dimension of state and input parameters that each have different prior distributions. [5]. youtube video of paper presentation
7. Using GAN to learn target optimization function's landscape, specifically around mode/modes (unsure) optimum. [6]

Modification to frequentist models that turns into bayesian:

1. Using dropout as intrinsic ensemble (bayesian?).
2. Using stochastic weight averaging withe gaussian noise (SWAG) for bayesian model averaging by approximating a normal weight posterior initialized on a pretrained solution. Bayesian model averaging in general prevents overfitting and overconfidence - Maddox et al [7]

Other assumptions and constraints

1. weight decay as normal prior on weights

1.1 Optimization

Neural networks are typically optimized through variants of the gradient descent optimizers. While gradient descent is stable and efficient to train, it is very easy for it to be stuck at a high dimensional local optimum or saddle point. Various gradient free methods such as evolutionary algorithms genetic, neuroevolution, metaheuristic particle swarm, bayesian optimization have being applied to neural networks and each have their weaknesses.

1.1.1 Benchmark Functions by Landscape group

Ill-conditioned,

Gaussian Type: Lunacek Bi-Rastrigin, Sphere, Schaffers F7 ill-conditioned

Non Gaussian (Valley? and other?): Shifted and Rotated Expanded Schaffer F6 and the Shifted and Rotated Weierstrass

1.1.2 Genetic Algorithms

1.1.3 Metaheuristic Algorithms

1.1.4 Bayesian Optimization

1.1.5 Neural Network Based Optimizers

Neural networks have being applied as optimizers as well. Namely Opt-GAN proposed by Lu et al 2021 [6] uses a GAN to generate samples from latent spaces that transforms into a multivariate uniformly distribution high dimensional space that is gradually transformed to approximate the target function's landscape and eventually area around the optimum via a transition between exploration and exploitation phase controlled by hyperparameter.

1.2 Bayesian Deep Learning

Bayesian deep learning offers insight into uncertainty of model parameters and predictive distribution.

The predictive uncertainty can be decomposed further to model and data uncertainty to make inference about the data space that requires more sampling for training to reduce model uncertainty. Chai 2018 [8] uses variational inference to approximate the posterior distribution of model weights and thus inference about the uncertainty with respect to specific input features such as a patch of an image. By learning which input feature region produces more reducible (epistemic) uncertainty we can adjust how to expand our training dataset to fully cover the dataset's event/probability space.

1.3 Variational Inference

1.4 Markov Chain Monte Carlo

1.4.1 Metropolis Hasting

Assume a distribution of a random variable $p(\theta|Y), \theta = (\boldsymbol{w}, \tau)$ that we'd like to sample from that is hard to evaluate. Specifically:

$$Y \sim \mathrm{N}(f(\boldsymbol{x}), \tau^2)$$
$$p(y) = \frac{1}{\sqrt{2\pi\tau^2}} \exp(-\frac{1}{2\tau^2}(y - f_{\boldsymbol{w}}(\boldsymbol{x}))^2)$$
$$\boldsymbol{w} \sim \mathrm{N}(\mathbf{0}, \sigma^2)$$
$$\tau^2 \sim \mathrm{LogNorm}(\alpha, \beta)$$
$$p(\boldsymbol{\theta}|Y) \propto p(Y|\boldsymbol{\theta})p(\tau^2)p(\boldsymbol{w})$$

The Metropolis Hasting algorithm allows us to sample this distribution by using a proposal distribution $q(\boldsymbol{\theta}|\boldsymbol{\theta}_t)$ that is rejected according to the probability

$$\min(1, \frac{p(\boldsymbol{\theta}^*|x)q(\boldsymbol{\theta}_t|\boldsymbol{\theta}^*)}{p(\boldsymbol{\theta}_t|x)q(\boldsymbol{\theta}^*|\boldsymbol{\theta}_t)})$$

Criterion for suitable prroposal distribution include uniqueness of distribution and existance of stationary distribution.

1.4.2 Hamiltonian Dynamics

Hamiltonian Monte Carlo is a vaiant of Metropolis Hasting Algorithm that utilizes Hamiltonian dynamics to propose a new point that incorporates the energy information of the current state to increase acceptance probability while reducing correlation between the points. it samples according to the following acceptance distribution:

$$\alpha = \min(1, \frac{\exp[-H(x_n(L\delta t), \mathbf{p}_n(L\delta t)]}{\exp[-H(x_n(0), \mathbf{p}_n(0)]})$$

1.4.3 Langevin Dynamics

Langevin Dynamic is a single leap frog step Hamiltonian Monte Carlo that can be alternatively considered as adding gaussian noise to weights proposed by gradient descent algorithms. In practice it still maintains the ability to increase acceptance probability. It proposes weights according to the following rule:

a stub equation

In standard LD or 1 Leap frog step of Hamilton Dynamics, there is 1 distinctive forward+backward pass in a leap frog step:

1. w_last forward (for posterior likelihood $p(data|w\_last)$)
2. w_last backward (for proposal distribution mean w_last_bar,$w\_star \sim N(w\_last\_bar, \sigma)$)
3. w_star forward (for posterior likelihood $p(data|w\_star)$)
4. w_star backward (for proposal distribution mean w_star_bar,$w\_last \sim N(w\_star\_bar, \sigma)$)

The empirical results agrees thats the time complexity with respect to epoch of frequentist training and langevin proposal is identical, but langevin is about 3-4 times slower in my impleentation. The suspected reason is the heavy cost of weight replacement/ duplication and copy for statistic analysis purpose. I need to confirm by how much this can be optimized.

w_last are reused from last iteration.

By using a surrogate model to approximate the posterior likelihood $p(data|w_star)$ we replaced step 3's forward pass.

1.4.4 Application to Large Dataset

Stochastic Langevin Dynamics

Welling 2011 [1] proposed stochastic langevin gradient using mini-batched training samples that does not evaluate the full dataset during each training step to accomodate big data. This results in the removal of accept reject step in the training as it is argured that as $\Delta\boldsymbol{\theta}$ is dominated by gradient descent early and should always result in high probability of acceptance. Whereas once the network reaches a local optimum, the gaussian noise would sample its surroundings.

Minibatch Metropolis Hasting

Seita et al 2018 [9] followed the work of others (not sure whether to cite) and propose to increase the amount of samples inside a current mini-batch if it fails to satisify a certain constraint. However, the proposed algorithm relies on evaluated the additional samples as the minibatch increases in size. This means that if the initial minibatch size is not sufficient, it might take a really long time to find the optimal batch that approximates the full dataset. It is not tested with langevin gradient (still reading).

1.5 Generative Models

1.5.1 Vartiational Autoencoder (VAE)

1.5.2 Normalizing Flow

1.5.3 Generative Adversarial Network (GAN)

The target loss function for generative model in continuous case (we care for continuous case only as we plan to use it to generate the weights) in unsupervised setting is $L(D) \approxeq \frac{1}{N}\sum_{i=1}^{N} -\log(p_\theta(\tilde{x}^{(i)})) + c$. Where $\tilde{x}^{(i)} = x^i + u$ with $u \sim U(0, a)$ and $c = -M \times \log a$, a is dependent on data discretization and M is the dimension of x. This loss function encourages the network to learn to distribute data across the domain of the standard normal distribution. When there is an underlying distribution to approximate, we need to utilizes distance/divergence measures in loss function to have convergent distribution.

VAE is a shrinkage training model that has two shrinker that trains to a latent space of mean and variance of normal distribution. from the two latent vector it then trains to simulate a proper input.

GAN have discriminator and generator, generator samples from standard normal and maps it via a network to output space, discriminator compare the overall generated result to dataset. The training is conducted via a maximizing loss for discriminator and minimizing loss on generator.

Normalizing Flow is composed of multiple inversible layers that helps to provide a tractable posterior likelihood on weights. Due to the decomposition of Jacobian matrix during gradient calculation, there is a significant reduction in memory requirements. Specifically the GLOW model proposed by OpenAI [10] uses 1x1 convolution to simulate channel switching to boost training performance. Another

paper from USTC improves upon this by using matrix exponentials (no idea what it means at the moment)

In flow and autoencoder we are dealing with distributions directly in the latent space (flow is autoencoder with only one network) and Arjovsky (NYU) [11] et al proposed in 2017 to simulate the Wasserstein distance (in general Earth Mover Distance) instead as a loss measure when training GAN.(may be it can be applied to VAE or NF? Or maybe it already is, base on how I see the implementation).

It is stated that divergence measures such as Jensen or KL fails to pickup the low dimensional manifold of a lot of problems' distributions (Q: are the distribution of weights of DNN parameters supported by low dimensional manifolds?), further in practice, other measures when training GAN produces in saturated discriminator that fails to provided any sensible gradient afterwards, that leads to mode collapse. WGAN resolves this as further training of critic/(score based discriminator instead of class based discriminator) still provides reasonable gradient. WGAN's weakness include failure when using momentum based optimizer (use RMSProp instead), and the clamping of hyperparameters to maintain lipschitz (what does this mean?) seems to be troublesome as it is problem specific. This is addressed by Gulrajani [12] using gradient penalty normalisation to replace the clipping that allows the loss to be normally distributed instead of bernoulli distrbuted at the two clipping points. This paper also states that WGAN has comparable epoch performance to WCGAN while offering better stability, the caveat is a slower wallclock time.

More info here GAN vs WGAN

short read on improved WGAN - Gradient Penalty

Turner et al 2019 [13] proposed to use Metropolis Hasting sampling from the generator with the acceptance ratio based on calibrated discriminator. It was proven to generate better quality sample as the discriminator is better at rating than the generator at samping due to the latter being a harder task. It was also shown that the discriminator are generally not properly calibrated.

1.6 low dimensional manifold

(Imagine your patterned bedsheets. They've got a nice plaid grid look to them, very easy to predict what the next few centimeters of material look like.

Now imagine someone tied them in a knot, tore them a little, balled them up, and made them a crumpled mess, then handed them to you. Now if you were to look at the plaid pattern, it would be very difficult to discern how the pattern is, or predict what will be the color of any point of the arbitrary 3D ball of mess you've got. But if you untangled it, you could clearly see the pattern again.

The data is a bunch of colors in an arbitrary 3D ball of mess with very difficult to discern patterns. But the data lies on a low dimensional manifold (the 2D bedsheet). If you could figure out the manifold (flatten the bedsheet), the data will be easy to model.) *explaination on reddit*

(stil have problem understanding the math behind it, but implementation wise i think it just removes the bounded final activation and uses the score as a loss)

1.7 Application:

1.7.1 Stochastic Weather Generation

Besombes et al 2021 [14] trained a WGAN on 100 years data simulated from the general circulation model PlaSIM, in that they used batch normalization to stabilize the gan training. However, batch normalisation induces dependencies between items that are suppose to be independent within a batch. This ""will be"" replaced with (pixel norm or layer norm) to stabilize the training. In the published code they also normalized each timeframe independently, it will be replaced with normalization over the whole dataset.

1.7.2 Weather Forcasting

1.8 Notes:

Swag : approximates local loss distribution (posterior?) using standard normal momentum distribution. Propose -¿ Accept Reject: replacing forward pass for proposal Surrogate by Chandra et al: evaluate proposal and spit out its posterior likelihood i.e.

$min(1, pi(w*|x)/pi(wi|x) * q(wi|w*)/q(w*|wi))$

$min(1, p(x|w*)/p(x|wi) * p(w*|x)/p(wi|x) * q(wi|w*)/q(w*|wi))$

$p(w|x)$ and $q(wi|w*)$ are both easy to compute, the first is a loss, the second is proposal distribution (using tractable distribution like normal that s easy to evaluate) $p(x|w)$ requires forward pass, so we evaluate it using surrogate
$https://www.lpsm.paris/pageperso/merle/slides_3.1.TV.pdf$ stuff for distance understanding

list to read Tractable Approximate Gaussian Inference for Bayesian Neural Networks

Adversarial Variational Bayes: Unifying Variational Autoencoders and Generative Adversarial Networks

GAN vs VAE

Distributed Bayesian optimisation framework for deep neuroevolution

Bridging the Gap Between f-GANs and Wasserstein GANs Stronger and Faster Wasserstein Adversarial Attacks

Modulating Surrogates for Bayesian Optimization

GENERATIVE ADVERSARIAL LEARNING OF MARKOV CHAINS

on the quantitative analysis of decoder based generative models

Packages of interest include pymc, pyro, pytorch, (maybe sydney-machine-learning/pingala?)

Chapter 2

Challenges

As we consider the weights in a neural network encoded sequentially, this means the weight space will have an exponential amount of "quasi-symmetric" (is this the right word?) local optimal modes as switching the order of weights in a layer and its corresponding connections to the next layer does not change the network's state.

overfitting of GAN that leads to no escape of local optimum during training or heatup.

Potential Solutions:

- use exploration/ exploitation
- maybe solvable with parallel tempering with different loss penalty in wgan?)

Solving the detailed balance condition of using GAN. because of generating from normal. we need to consider the probability transformation. Tasks and potential solutions:

- use flow based GAN or flow
- calculate the invertible functions and the respective post transform distribution of GAN
- embarassingly ignore it?
- look up stuff on invertible GAN, invertible conditional GAN, and other stuff.
- use different acceptance test like the one in Metropolis Hasting GAN
- generate the lanvegin $q(\theta * |\theta_l)$ probability and the corresponding weight together.

Chapter 3

Proposal

3.1 Proposal

Now we propose to replace step 4's backward pass with a GAN that generates the weights.

The believed benefits includes:

1. faster evaluation.

For this GAN to be benificial, it has several constraints:

1. shallower than original network.
2. approximates the full posterior $p(w|data)$.

3.2 Random Ideas

Multiple GANs, each for different module/layers of network such that it works for larger network like resnet50.

Federated learning - consider each parallel process to have its own minibatch as localized individual data.

Few shot learning - consider a potentially limited amount of accepted samples for training.

Convert a random walk in latent space to a langevin gradient in weight space

Use Stochastic Langevin Gradient when in initial warm up/ burn phase, later during reaching of a plateau, use GAN to escape the local optimum. Since the GAN is learning a multimodal distribution through paralleled tempering, it should helps to explore the different weights and learn the global optimum.

To break the symmetric mode in joint weight distribution, is it possible to use tempering or treat individual layer's with separate learnable random variable variance?

3.2.1 Potential uses

Using the posterior on weights (more accurate than variation inference) for network connection pruning? (need to look up)

Chapter 4

Algorithm

Chapter 5

Experiments

Chapter 6

Analysis

Chapter 7

Discussion

Chapter 8

Conclusion

References

[1] Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient langevin dynamics. In *ICML*, 2011.

[2] Rohitash Chandra, Lamiae Azizi, and Sally Cripps. Bayesian neural learning via langevin dynamics for chaotic time series prediction. In *ICONIP*, 2017.

[3] Rohitash Chandra, Konark Jain, Ratneel Vikash Deo, and Sally Cripps. Langevin-gradient parallel tempering for bayesian neural learning. *CoRR*, abs/1811.04343, 2018.

[4] Rohitash Chandra, Konark Jain, and Arpit Kapoor. Surrogate-assisted parallel tempering for bayesian neural learning. *CoRR*, abs/1811.08687, 2018.

[5] Nikolaj T. Mücke, Benjamin Sanderse, Sander Bohté, and Cornelis W. Oosterlee. Markov chain generative adversarial neural networks for solving bayesian inverse problems in physics applications, 2021.

[6] Minfang Lu, Fengyang Sun, Lin Wang, Bo Yang, and Shuangrong Liu. Black-box optimization via generative adversarial nets. *CoRR*, abs/2102.03888, 2021.

[7] Wesley J Maddox, Pavel Izmailov, Timur Garipov, Dmitry P Vetrov, and Andrew Gordon Wilson. A simple baseline for bayesian uncertainty in deep learning. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[8] Lucy R. Chai. Uncertainty estimation in bayesian neural networks and links to interpretability. 2018.

[9] Daniel Seita, Xinlei Pan, Haoyu Chen, and John Canny. An efficient minibatch acceptance test for metropolis-hastings. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 5359–5363. International Joint Conferences on Artificial Intelligence Organization, 7 2018.

[10] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[11] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223. PMLR, 06–11 Aug 2017.

[12] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[13] Ryan Turner, Jane Hung, Eric Frank, Yunus Saatchi, and Jason Yosinski. Metropolis-Hastings generative adversarial networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6345–6353. PMLR, 09–15 Jun 2019.

[14] C. Besombes, O. Pannekoucke, C. Lapeyre, B. Sanderson, and O. Thual. Producing realistic climate data with generative adversarial networks. *Nonlinear Processes in Geophysics*, 28(3):347–370, 2021.