
Facilitating Bayesian Continual Learning by Natural Gradients and Stein Gradients

Yu Chen*
University of Bristol
yc14600@bristol.ac.uk

Tom Diethe
Amazon
tdiethe@amazon.com

Neil Lawrence
Amazon
lawrennd@amazon.com

Abstract

Continual learning aims to enable machine learning models to learn a general solution space for past and future tasks in a sequential manner. Conventional models tend to forget the knowledge of previous tasks while learning a new task, a phenomenon known as *catastrophic forgetting*. When using Bayesian models in continual learning, knowledge from previous tasks can be retained in two ways: (i) posterior distributions over the parameters, containing the knowledge gained from inference in previous tasks, which then serve as the priors for the following task; (ii) *coresets*, containing knowledge of data distributions of previous tasks. Here, we show that Bayesian continual learning can be facilitated in terms of these two means through the use of natural gradients and Stein gradients respectively.

1 Background

There are several existing approaches for preventing *catastrophic forgetting* of regular (non-Bayesian) Neural Networks (NNs) by constructing a regularization term from parameters of previous tasks, such as Elastic Weight Consolidation (EWC) [1] and Synaptic Intelligence (SI) [2]. In the Bayesian setting, Variational Continual Learning (VCL) [3] proposes a framework that makes use of Variational Inference (VI).

$$\mathcal{L}_{\text{VCL}}(\theta) = \mathbb{E}_{q_t(\theta)} [\log p(D_t|\theta)] - \text{KL}(q_t(\theta)||q_{t-1}(\theta)). \quad (1)$$

The objective function is as in Equation (1), where t is the index of tasks, $q_t(\theta)$ represents the approximated posterior of parameters θ of task t and D_t is the data of task t . This is the same as the objective of conventional VI except the prior is from the previous task, which produces the regularization by Kullback-Leibler (KL)-divergence between parameters of current and previous tasks. In addition, VCL [3] proposes a predictive model trained by coresets of seen tasks to perform prediction for those tasks, where the coresets consist of data samples from the dataset of each seen task except the training data D_t of each task.

$$\hat{q}_t = \arg \max_{q_t} \mathbb{E}_{q_t(\theta)} [\log p(C_t|\theta)] - \text{KL}(q_t(\theta)||q_t^*(\theta)). \quad (2)$$

As shown in Equation (2), $C_t = \{c_1, c_2, \dots, c_t\}$ represents the collection of coresets at task t and $q_t^*(\theta)$ is the optimal posterior obtained by Equation (1). VCL shows promising performance comparing to EWC [1] and SI [2], which demonstrates that effectiveness of Bayesian approaches to continual learning.

2 Facilitating Bayesian continual learning by natural gradients

In order to prevent *catastrophic forgetting* in Bayesian continual learning, we would prefer the posteriors of a new task stay as close as possible to the posteriors of the previous task. Conventional gradient methods give the direction of steepest descent of parameters in Euclidean space, which might

*This work was conducted during Yu Chen's internship at Amazon.

cause a large difference in terms of distributions following a small change in terms of parameters. We posit that natural gradient methods may be a better choice than the conventional gradient descent. The definition of the natural gradient is the direction of steepest descent in Riemannian space rather than Euclidean space, which means the natural gradient would prefer the smallest change in terms of distribution while optimizing some objective function [4]. The formulation is as below:

$$\hat{\nabla} \mathcal{L}(\beta) = \nabla \mathcal{L}(\beta) \mathbb{E}_{\theta} \left[(\nabla \log p(\theta|\beta))^T (\nabla \log p(\theta|\beta)) \right]^{-1} \stackrel{\text{def}}{=} \nabla \mathcal{L}(\beta) \mathbf{F}_{\beta}^{-1}, \quad (3)$$

where \mathbf{F}_{β} is the Fisher information of β .

2.1 Natural gradients of the exponential family

Specifically, when the posterior of a parameter θ is from the exponential family, we can write it in the form of $p(\theta|\beta) = h(\theta) \exp\{\eta(\beta)^T u(\theta) - a(\eta(\beta))\}$, where $h(\cdot)$ is the base measure and $a(\cdot)$ is log-normalizer, $\eta(\cdot)$ is natural parameter and $u(\cdot)$ are sufficient statistics. Then the Fisher information of β is the covariance of the sufficient statistics which is the second derivative of $a(\cdot)$ [5]:

$$\mathbf{F}_{\beta} = \mathbb{E}_q[(u(\theta) - \mathbb{E}_q[u(\theta)])^T (u(\theta) - \mathbb{E}_q[u(\theta)])] = \nabla_{\beta}^2 a(\eta(\beta)). \quad (4)$$

In this case, the natural gradient of β is the transformation of the Euclidean gradient by the precision matrix of the sufficient statistics $u(\theta)$.

2.2 Gaussian natural gradients and the Adam optimizer

In the simplest (and most common) formulation of Bayesian NNs (BNNs), the weights are drawn from Gaussian distributions, with a mean-field factorization which assumes that the weights are independent. Hence, we have an approximate posterior for each weight $q(\theta_i|\mu_i, \sigma_i) = \mathcal{N}(\mu_i, \sigma_i^2)$, where μ_i, σ_i are the parameters to be optimized, and their Fisher information has an analytic form:

$$\mathbf{F}_{\mu_i} = 1/\sigma_i^2, \quad \mathbf{F}_{\mathbf{v}_i} = 2, \quad \text{where } \mathbf{v}_i = \log \sigma_i. \quad (5)$$

Consequently, the natural gradient of the mean of posterior can be computed as follows, where \mathcal{L} represents the objective (loss) function:

$$\hat{g}_{\mu_i} = \sigma_i^2 g_{\mu_i}, \quad g_{\mu_i} = \nabla_{\mu_i} \mathcal{L}. \quad (6)$$

Equation (6) indicates that small σ_i can cause the magnitude of natural gradient to be much reduced. In addition, BNNs usually need very small variances in initialization to obtain a good performance at prediction time, which brings difficulties when tuning learning rates when applying vanilla Stochastic Gradient Descent (SGD) to this Gaussian Natural Gradient (GNG). As shown in Figure 1 and 2 in the supplementary materials we can see how the scale of variance in initialization changes the magnitude of GNG.

Meanwhile, Adam optimization [6] provides a method to ignore the scale of gradients in updating steps, which could compensate this drawback of GNG. More precisely, Adam optimization uses the second moment of gradients to reduce the variance of updating steps:

$$\theta_k \leftarrow \theta_{k-1} - \alpha_k * \mathbb{E}_k[\mathbf{g}_{\theta,k}] / \sqrt{\mathbb{E}_k[\mathbf{g}_{\theta,k}^T \mathbf{g}_{\theta,k}]}, \quad \mathbf{g}_{\theta} = \nabla_{\theta} \mathcal{L}, \quad (7)$$

where k is index of updating steps, \mathbb{E}_k means averaging over updating steps, α_k is the adaptive learning rate at step k . Considering the first and second moments of $\hat{g}_{\mu_i,k}$,

$$\begin{aligned} \mathbb{E}_k[\hat{g}_{\mu_i,k}] &= \mathbb{E}_k[\sigma_{i,k}^2] \mathbb{E}_k[g_{\mu_i,k}] + \text{cov}(\sigma_{i,k}^2, g_{\mu_i,k}) \\ \mathbb{E}_k[\hat{g}_{\mu_i,k}^2] &= (\mathbb{E}_k[\sigma_{i,k}^2]^2 + \text{var}(\sigma_{i,k}^2)) \mathbb{E}_k[g_{\mu_i,k}^2] + \text{cov}(\sigma_{i,k}^4, g_{\mu_i,k}^2) \end{aligned} \quad (8)$$

We can see that only when $\text{var}(\sigma_{i,k}^2) = 0$ and $g_{\mu_i,k}$ are independent from $\sigma_{i,k}^2$, the updates of GNG are equal to the updates by Euclidean gradients in the Adam optimizer. It also that shows larger $\text{var}(\sigma_{i,k}^2)$ will result in smaller updates when applying Adam optimization to GNG.

We show comparison between different gradient descent algorithms in the supplementary materials. More experimental results are shown in Section 4.

In **non-Bayesian** models, natural gradients may have problems with the Adam optimizer because there is no posterior $p(\theta|\beta)$ defined. The distribution measured in natural gradient is the conditional probability $p(x|\theta)$ [4] and the loss function is usually $\mathcal{L}_\theta = \mathbb{E}_x[\log p(x|\theta)]$. In this case the natural gradient of θ becomes:

$$\hat{\mathbf{g}}_\theta = \frac{\mathbb{E}_x[g_\theta]}{\mathbb{E}_x[g_\theta^T g_\theta]}, \quad g_\theta = \nabla_\theta \log p(x|\theta), \quad \nabla_\theta \mathcal{L} = \mathbb{E}_x[g_\theta]. \quad (9)$$

If we apply this to the Adam optimizer, which means replacing \mathbf{g}_θ in Equation (7) by $\hat{\mathbf{g}}_\theta$, the formulation is duplicated and involves the fourth moment of the gradient, which is undesirable for both Adam optimization and natural gradients. One example is EWC [1] which uses Fisher information to construct the penalty of changing previous parameters, hence, it has a similar form with Adam and it works worse with Adam than with vanilla SGD in our experience. However, this is not the case for **Bayesian** models, where Equation (9) does not hold because the parameter θ has its posterior $p(\theta|\beta)$ and then the loss function \mathcal{L} is optimized w.r.t. β , meanwhile $\nabla_\beta \mathcal{L} \neq \mathbb{E}_\theta[\nabla_\beta \log p(\theta|\beta)]$ in common cases.

3 Facilitating Bayesian Continual Learning with Stein Gradients

In the context of continual learning, ‘‘coresets’’ are small collections of data samples of every learned task, used for task revisiting when learning a new task [3]. The motivation is to retain summarized information of the data distribution of learned tasks so that we can use this information to construct an optimization objective for preventing parameters from drifting too far away from the solution space of old tasks while learning a new task. Typically, the memory cost of coresets will increase with the number of tasks, hence, we would prefer the size of a coreset as small as possible. There are some existing approaches to Bayesian coreset construction for scalable machine learning [7, 8], the idea is to find a sparse weighted subset of data to approximate the likelihood over the whole dataset. In their problem setting the coreset construction is also crucial in posterior approximation, and the computational cost is at least $O(MN)$ [8], where M is the coreset size and N is the dataset size. In Bayesian continual learning, the coreset construction does not play a role in the posterior approximation of a task. For example, we can construct coresets without knowing the posterior, i.e. random coresets, K -centre coresets [3]. However, the information of a learned task is not only in its data samples but also in its trained parameters, so we consider constructing coresets using our approximated posteriors, yet without intervening the usual Bayesian inference procedure.

3.1 Stein gradients

Stein gradients [9] can be used to generate samples of a known distribution. Suppose we have a series of samples $\mathbf{x}^{(l)}$ from the empirical distribution $p(\mathbf{x})$, and we update them iteratively to move closer to samples from the conditional distribution $p(\mathbf{x}|\theta)$ by $\mathbf{x}_{k+1}^{(l)} = \mathbf{x}_k^{(l)} + \epsilon \phi^*(\mathbf{x}_k^{(l)})$, where

$$\phi^* = \arg \max_{\phi \in \mathcal{F}} -\nabla_\epsilon D_{KL}(p(\mathbf{x})||p(\mathbf{x}|\theta))|_{\epsilon=0} = \arg \max_{\phi \in \mathcal{F}} \mathbb{E}_q[\mathcal{A}_p \phi(\mathbf{x})]. \quad (10)$$

$\phi^*(\cdot)$ is chosen to decrease the KL-divergence between $p(\mathbf{x})$ and $p(\mathbf{x}|\theta)$ in the steepest direction; \mathcal{F} is chosen to be the unit ball of the Reproducing Kernel Hilbert Space to give a closed form update of samples; and \mathcal{A}_p is Stein operator. Thus, the Stein gradient can be computed by:

$$\phi^*(\mathbf{x}_k^{(l)}) = \frac{1}{M} \sum_{j=1}^M \left[k(\mathbf{x}_k^{(j)}, \mathbf{x}_k^{(l)}) \nabla_{\mathbf{x}_k^{(j)}} \log p(\mathbf{x}_k^{(j)}|\theta) + \nabla_{\mathbf{x}_k^{(j)}} k(\mathbf{x}_k^{(j)}, \mathbf{x}_k^{(l)}) \right]. \quad (11)$$

In the mean-field BNN model introduced in Section 2.2, we can just replace θ by μ, σ in Equation (11). The computational complexity of the Stein gradient method is $O(M^2)$, which is significantly cheaper than $O(MN)$ when $M \ll N$.

4 Experiments

We tested GNG with Adam in the framework of VCL on permuted MNIST [1], split MNIST [2], and split fashion MNIST [10] tasks. We applied a BNN with two hidden layers, each layer with

100 hidden units, all split tasks tested using multi-head models [2]. The results are displayed in Figure 1 (left column) and the error bars are from 5 runs by different random seeds. In the permuted MNIST task, GNG with Adam outperforms standalone Adam. There is no significant difference in split tasks. More details and further analysis can be found in the supplementary material.

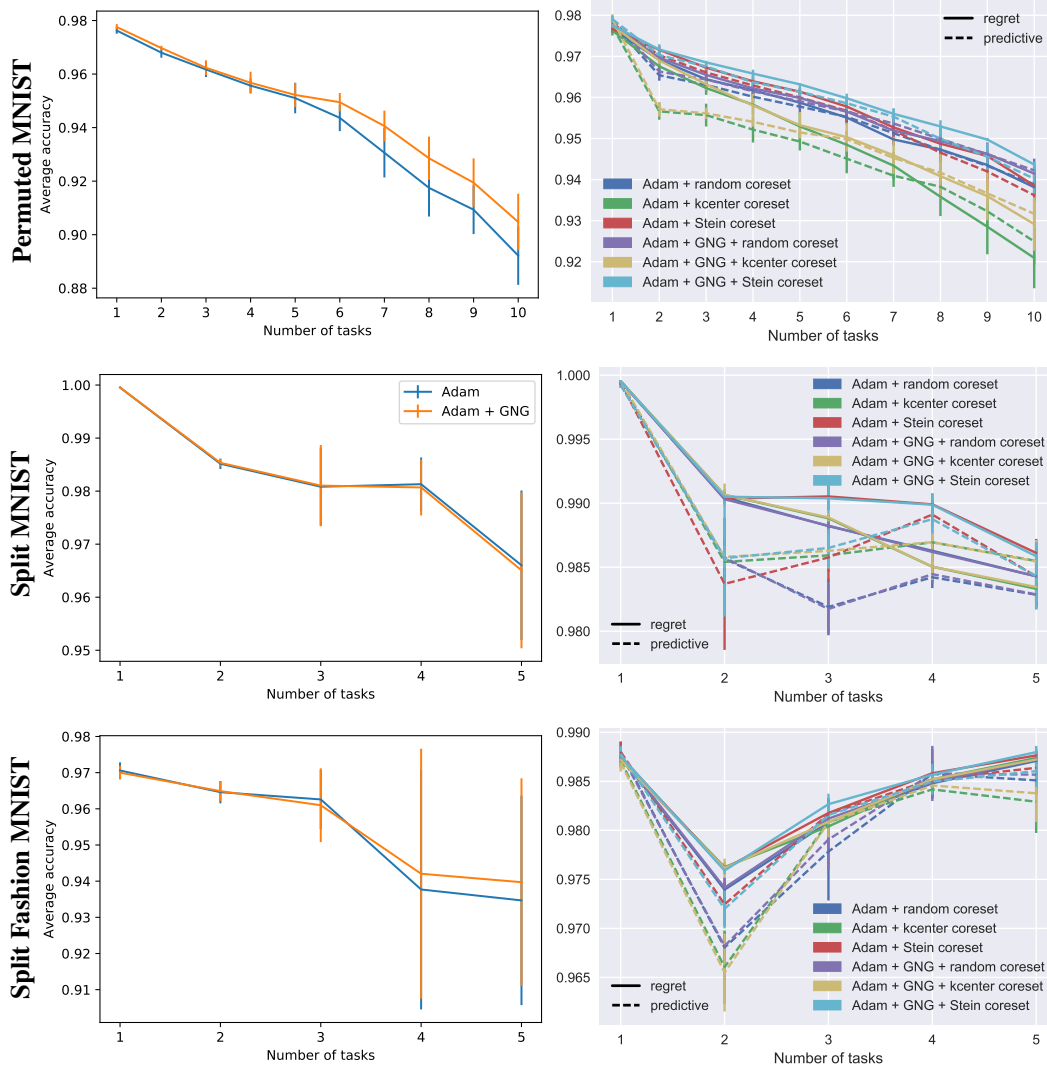


Figure 1: Average accuracy on permuted and split tasks without (left) and with (right) coresets

For experiments with coresets, we tested two different usages of coresets. In the first we use coresets to train a predictive model as introduced in [3] (Equation (2)); in the second we add a regret loss from the coresets to the objective function, which does not require a separate predictive model:

$$\mathcal{L}_t = \mathbb{E}_{q_t(\theta)} [\log p(\mathcal{D}_t|\theta)] + \mathbb{E}_{q_t(\theta)} [\log p(\mathcal{C}_{t-1}|\theta)] - \text{KL}(q_t(\theta)||q_{t-1}(\theta)), \quad (12)$$

where the second term in Equation (12) is regret loss constructed by coresets of previous tasks \mathcal{C}_{t-1} . We applied a RBF kernel in the same manner as described in [9] to the Stein gradients and tested the Stein coresets in both permuted and split tasks, comparing with random and K -center coresets. The coreset size is 200 per task in permuted MNIST and 40 in split tasks, which is the same as used in [3]. The results are shown in Figure 1 (right column). The regret usage of coresets gives better performance in general, and Stein coresets also outperform other two types of coresets in most cases.

References

- [1] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, page 201611835, 2017.
- [2] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, pages 3987–3995, 2017.
- [3] Cuong V Nguyen, Yingzhen Li, Thang D Bui, and Richard E Turner. Variational continual learning. In *International Conference on Learning Representations*, 2018.
- [4] Razvan Pascanu and Yoshua Bengio. Revisiting natural gradient for deep networks. In *International Conference on Learning Representations 2014 (Conference Track)*, April 2014.
- [5] Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.
- [6] Diederik P Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *Proceeding 3rd International Conference on Learning Representations*, 2014.
- [7] Jonathan Huggins, Trevor Campbell, and Tamara Broderick. Coresets for scalable Bayesian logistic regression. In *Advances in Neural Information Processing Systems*, pages 4080–4088, 2016.
- [8] Trevor Campbell and Tamara Broderick. Bayesian coreset construction via greedy iterative geodesic ascent. *arXiv preprint arXiv:1802.01737*, 2018.
- [9] Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose Bayesian inference algorithm. In *Advances In Neural Information Processing Systems*, pages 2378–2386, 2016.
- [10] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Facilitating Bayesian Continual Learning by Natural Gradients and Stein Gradients

Supplementary Material

Yu Chen
University of Bristol
yc14600@bristol.ac.uk

Tom Diethe
Amazon
tdiethe@amazon.com

Neil Lawrence
Amazon
lawrennd@amazon.com

A Comparing different gradient descent algorithms

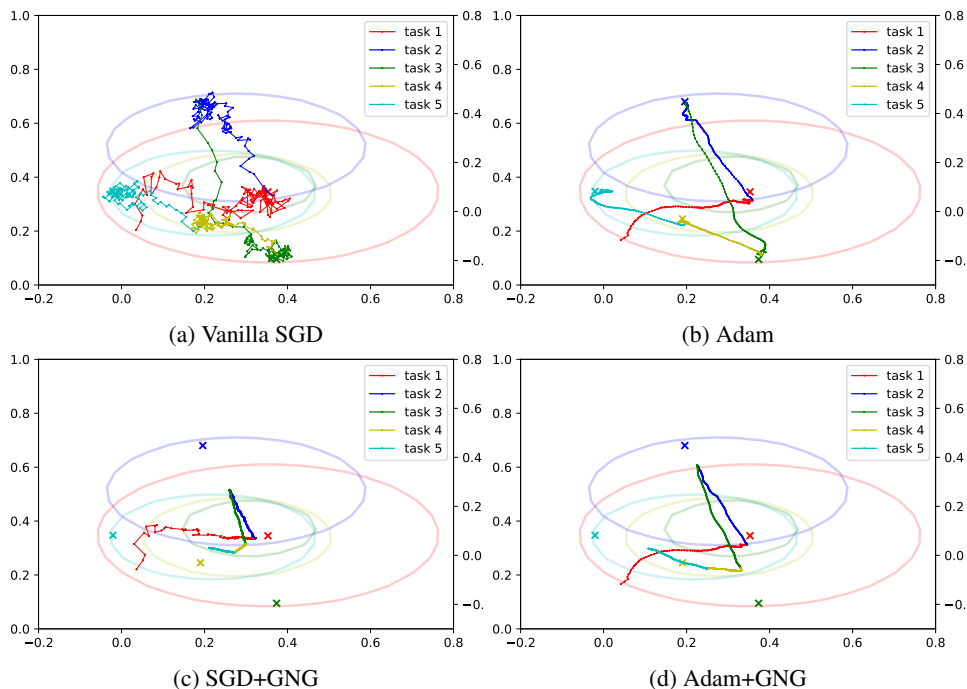


Figure 1: Updating trajectory of parameters of 1-dimensional Bayesian linear regression in continual learning. The model is defined as $y \sim \mathcal{N}(wx + b, 0.1)$, $w \sim \mathcal{N}(\mu_w, \sigma_w^2)$, $b \sim \mathcal{N}(\mu_b, \sigma_b^2)$. The x -axis is μ_w and y -axis is μ_b . The contour depicts the average MSE over seen tasks, the cross-mark indicates the position of true parameters of each task, different colours represent different tasks. The learning rate is set to 0.001 for vanilla SGD and 0.01 for all other methods. The initialization of σ_w and σ_b is set to $\sigma_0 = e^{-1}$.

Figures 1 and 2 demonstrate how the optimization methods and scale of variance affect parameter updates in an 1-dimensional Bayesian linear regression model of continual learning. In Figure 1d the updating steps are smaller than in Figure 2d, even when the scale of variance is larger, which is because larger value of initialization σ_0 results in a larger variance of gradients (see the difference between Figure 1a and Figure 2a, Figure 1b and Figure 2b), and consequently larger $\text{var}(\sigma_{i,t}^2)$ as well, meaning that the step size of Gaussian Natural Gradient (GNG) decreases according to Equa-

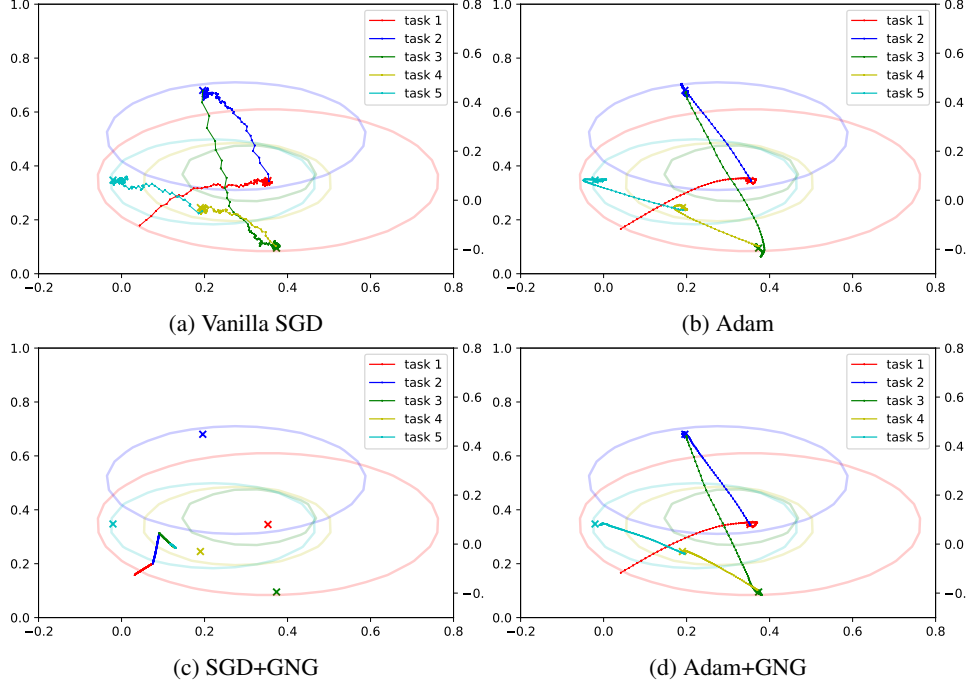


Figure 2: Parameter trajectory of 1-dimensional Bayesian linear regression in continual learning. All configurations are the same as in Figure 1 except $\sigma_0 = e^{-3}$.

tion 7 and 8 in the main content. In general, GNG shows lower variance in parameter updates, and it works better with Adam than with SGD.

B Further analysis of Gaussian Natural Gradients and Adam experiments

As one model has a limited capacity, and each different task contains some different information, the ideal case for continual learning is that each new task shares as much information as possible with previous tasks, and occupying as little extra capacity within the Neural Network as possible. This is analogous to model compression [1], but one key difference is we want more *free* parameters instead of parameters that are set to zero. For example, there are k independent parameters in a model and the log-likelihood of current task is factorized as:

$$\log p(D_t | \theta_1, \theta_2, \dots, \theta_k) = \sum_{i=1}^k \log p(D_t | \theta_i). \quad (1)$$

If θ_1 is absolutely free for this task, it indicates the following conditional probability is a constant w.r.t. θ_1 :

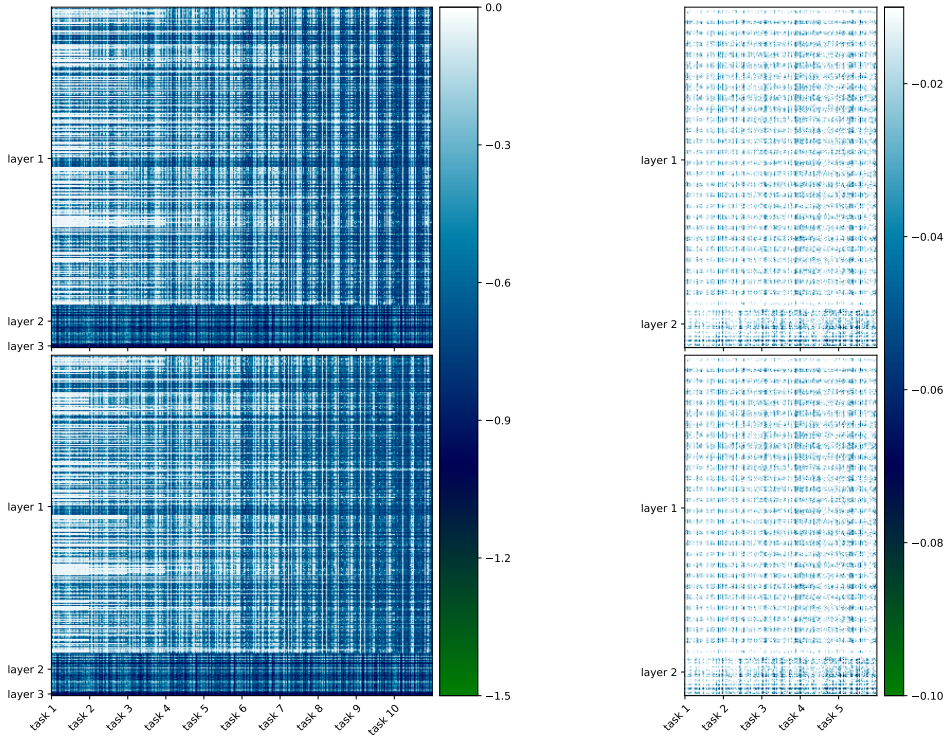
$$\log p(D_t, \theta_2, \dots, \theta_k | \theta_1) = \log p(D_t | \theta_1) + \sum_{j=2}^k \log p(D_t, \theta_j) = \text{const}, \quad \forall \theta_1. \quad (2)$$

This would require

$$\nabla_{\theta_1} \log p(D_t | \theta_1) = 0, \quad \forall \theta_1. \quad (3)$$

Therefore, θ_1 is free to move, then no matter what value of θ_1 is set to in future tasks, it will not affect the loss of previously learned tasks. In realistic situations, θ_1 is very unlikely to be absolutely free. However, it is feasible to maximize the entropy of θ_1 , larger entropy indicating more freedom of θ_1 . For instance, minimizing KL divergence includes maximizing the entropy of parameters:

$$KL(q_t(\theta) || q_{t-1}(\theta)) = \mathbb{E}_{q_t} [q_{t-1}(\theta)] - H_{q_t}(\theta). \quad (4)$$



(a) permuted MNIST

(b) split MNIST

Figure 3: Variance changes w.r.t. first task, top row is from models trained by Adam, bottom row is from models trained by Adam + GNG, tested on permuted and split MNIST without coresets. The x -axis is concatenated by tasks, the y -axis is concatenated by BNN layers, as split tasks are tested on multi-head models, so there is no layer 3 in Figure 3b.

On the other hand, it is undesirable to change parameters with lower entropy instead of those with higher entropy while learning a new task, since it could cause a larger loss on previous tasks.

The entropy of a Gaussian distribution is defined by its variance alone. In this sense, a larger decrease of variance indicates larger decrease of entropy. To understand why GNG works better on permuted MNIST tasks, we visualized how the variances of the weights change in Figure 3 where we normalized all values as below:

$$\Delta\sigma_{i,t} = \frac{\sigma_{i,t} - \max_i \sigma_{i,1}}{\max_i \sigma_{i,1}}, \quad (5)$$

where $\max_i \sigma_{i,1}$ is the maximal variance of the first task. When the variance of parameters is decreased by learning a new task, the entropy of the model is decreased as well. We can think of it as new information written into the model, so when the model has learned more tasks, the variances of more parameters will have shrunk as shown in Figure 3.

In an ideal case, a parameter with larger variance should be chosen to write new information preferentially to avoid erasing information of previous tasks. Therefore, it would be preferred if the dark colour spread more evenly in latter tasks in Figure 3, and Adam + GNG appears to have this property for the permuted MNIST task (Figure 3a). However, there is no notable difference caused by GNG for split MNIST tasks (Figure 3b), which consistent with their performance in terms of average accuracy over tasks. The underlying reason needs further investigation.

References

- [1] Christos Louizos, Karen Ullrich, and Max Welling. Bayesian compression for deep learning. In *Advances in Neural Information Processing Systems*, pages 3288–3298, 2017.