The Bayesian Learning Rule

Sacha Braun, Alexandre François and Hippolyte Pilchen
MVA ENS Paris-Saclay

Abstract

The Bayesian Learning Rule (BLR) is a principle that unifies a wide array of Bayesian and non-Bayesian algorithms by focusing on the minimization of expected loss within a geometric framework defined by a selected posterior distribution. This framework incorporates natural gradients, which provide insights into the curvature of the loss function, facilitating the optimization process. By adopting a Bayesian approach and applying appropriate approximations, it becomes possible to derive established algorithms, with their complexity being directly linked to the chosen posterior's complexity. Our contribution starts with an analysis of the increased solutions stability and robustness provided by the BLR. Additionally, we apply natural gradients based optimizer to refine existing algorithms, notably NoisyNet and expose the robustness property of the BLR. We also propose an extension of the unifying perspective of the BLR from which we derived Stochastic Average Gradient.

1 Introduction

Learning algorithms often aim to address the Empirical Risk Minimization (ERM) problem, formulated as follows, where l represents a loss function, and R denotes a regularization term that mitigates overfitting.

$$\theta^* = \arg\min_{\theta} l(\theta), \quad \text{where} \quad l(\theta) = \sum_i l(y_i, f_\theta(x_i)) + R(\theta). \tag{1}$$

The objective of numerous algorithms developed is to efficiently minimize the ERM (1) problem. Designing efficient algorithm is therefore a wide area of research. What if all algorithms that have "stood-the-test-of-time" could be seen as a derivation of a specific instance, of a single learning algorithm ? We explain why and how the Bayesian Learning Rule (BLR) [7], enables the derivation of state-of-the-art algorithms and how even non-Bayesian ones can be conceptualized as solving a two-step process:

- Setup: Writting a Bayesian objective as a function of a parameterize posterior.
- Non-naïve resolution: Utilizing natural-gradient descent for optimization.

This optimization framework, using Bayesian inference, enables to estimate the confidence or uncertainty in the predictions of the algorithms which can be very valuable in advanced ML [6]. The practical utilization of natural gradients has also been studied for black-box functions optimization and it has been established that it naturally derives adaptive learning rate for different components of the parameter vector we want to optimize [13]. Adaptive learning rate are widely used in modern optimizers for deep learning such as Adam [8]. From the BLR we derive a unifying framework for all these algorithms, leveraging information geometry.

Our work begins with an introduction on the theoretical setting of the Bayesian Learning Rule. We describe the method designed by [7] and give our critical point of view on using the Bayesian setting described to reproduce learning algorithms. We propose some original experiments to highlight the smoothing effect and the stability of the solutions that the Bayesian versions of algorithms can provide. We establish a direct connection between the BLR and NoisyNet [11], a renowned reinforcement learning algorithm designed to minimize the loss incurred by a neural network perturbed by Gaussian noise. In the original paper, the parameter updates are derived from Euclidean geometry. However, upon reviewing [7], we claim that the use of natural gradients could improve the updating process and proposed another optimizer for this specific neural network. As an additional contribution, we also underline the universality of the BLR by deriving a learning algorithm which have not been described in the paper of interest: Stochastic Average Gradient Descent. All our codes are available on GitHub[1].

[1] `https://github.com/HipPilchen/bayesian_learning_rule`

2 The Bayesian Learning Rule (BLR)

2.1 How to derive BLR learning algorithms ?

Let us define this Bayesian Learning Rule (BLR), highlighting its key points. The initial step involves reformulating the problem 1 in a Bayesian framework . Upon selecting a candidate class of distributions Q, one might aim to minimize the following Bayes objective, with $H(q)$ the entropy of q.

$$q^* = \arg\min_{q \in \mathrm{Q}} L(q) := \mathbb{E}_{\theta \sim q}[l(\theta)] - H(q), \tag{2}$$

Considering the posterior parameterized by a parameter λ, the challenge becomes identifying this optimal parameter, which can iteratively be performed by the optimization step

$$\lambda_{t+1} \leftarrow \arg\min_{\lambda} \langle \nabla_\lambda L(\lambda_t), \lambda \rangle + \frac{1}{2} D(\lambda, \lambda_t), \tag{3}$$

where D represents a metric that penalizes optimization and prevent from taking too large updates. Setting D to be the euclidean distance, we recover a classic gradient step. The essence of the BLR is to use the Kullback-Leibler Divergence (KLD), which delineates the geometry of the optimization space.

Since this optimization unfolds within the geometry shaped by the KLD, the update formula proposed is

$$\lambda_{t+1} \leftarrow \lambda_t - \rho_t F(\lambda_t)^{-1} \nabla_\lambda (\mathbb{E}_{q_{\lambda_t}}[l(\theta)] - \mathrm{H}(q_{\lambda_t})), \tag{4}$$

where F denotes the Fisher Information Matrix (FIM). The emergence of the FIM and its intricacies are further elaborated in Appendix, but is just a minimization of the loss penalized by the KLD. We should then see the inverse of its associated KL tensor as a descent normalized, that happens to be well approximated by the FIM [3]. The key insight is that the FIM contains all the information about the curvature in our loss function. Nonetheless, the FIM often poses computational challenges, rendering the update process cumbersome.

It has been shown [2] that this obstacle can be circumvented with a judicious choice of posterior. Specifically, consider a posterior within the minimal exponential family, i.e.,

$$q_\lambda(\theta) = h(\theta) \exp\left[\langle \lambda, T(\theta) \rangle - A(\lambda)\right].$$

In this scenario, the gradient within the geometry induced by the KLD equates to a mere gradient with respect to the natural parameters μ, with $\mu = \mathrm{E}_{q_\lambda}[T(\theta)] = \nabla_\lambda A(\lambda)$. Formally,

$$F(\lambda_t)^{-1}[\nabla_\lambda \mathbb{E}_q[\cdot]|_{\lambda=\lambda_t}] = \nabla_\mu \mathbb{E}_q[\cdot]|_{\mu=\nabla_\lambda A(\lambda_t)}.$$

This revelation is significant: it implies that leveraging the KLD-induced geometry does not necessitate computing the FIM; instead, one only needs to compute gradients concerning the natural parameters, termed natural gradients. Further mathematical justifications of this derivation and on the form of $\mu = \nabla_\lambda A(\lambda)$ are provided in Appendix.

Now that the Bayesian Learning Rule is specified, one can observe that the solution q^* in equation 2 highly depends on the size of the class $\mathcal{Q}$. In fact, the nature and the complexity of the algorithm derived by the Bayesian Rule vary with the complexity of the class $\mathcal{Q}$. At optimality, the Bayes objective must respect

$$\nabla_\mu \mathbb{E}_{q^*}[l(\theta)] = \nabla_\mu \mathcal{H}(q^*)$$

Therefore, when the posterior distribution is a Gaussian with mean m and fixed covariance, id est $q \sim \mathcal{N}(\theta|m, I)$ the entropy is constant and optimality condition boils down to a 1^{st} optimality condition. If the complexity increases with learnable covariance, $q \sim \mathcal{N}(\theta|m, S^{-1})$, the entropy is no longer constant and the Bayesian Learning Rule update is more complex. Finally, using some approximation methods like the delta method enables to retrieve many of the classical optimization and deep learning algorithms. [7] uses varying complexity posterior classes to retrieve many of these algorithms.

2.2 The Bayesian Learning Rule provides more robust solutions

We would like to take a step back and try to further understand how going back to the expectation in the Bayesian Learning Rule's updates actually helps. To that extent, we display the actual effect of taking the expectation of the loss in a specific model (Fig 1)

Instead of optimizing the original loss, the Bayesian approach optimizes the expectation of the loss with respect to a Gaussian parameter for example. We illustrate the smoothing effect of the expectation, Fig. 1, that acts in a way like a convolution (a Gaussian convolution in that case), by adding some noise to the loss. By doing

Figure 1: Taking the expectation into account smooths the loss, which becomes a function of the mean and the standard deviation.

so, the Bayesian Learning Rule no longer works with the original loss, which can be very irregular or full of local minima that hampers training, but rather modifies it to stabilize its variations. This implicit regularization is not new in the field of deep learning, and has been extensively used recently in reinforcement learning, or robust optimization [12]. By implicit regularization we mean that no concrete regularizing term is added, like a penalization term for example. This principle is the basis of the stochastic gradient descent implicit regularization [14]. The introduction of noise smooths the loss and avoids local minima, towards the quest of more dominant ones. This same natural idea governs the Bayesian Learning Rule principle, as the stochastic noise is akin to the the Bayesian noise injected through the distribution q_λ.
However, the BLR provides another strong point that sets it apart from the typical gradient descent, by providing solutions that are more robust and can control better the uncertainties in the prediction, proving very useful and efficient in deep networks for instance as proved by the winning team of NeurIPS Challenge [4]. To understand that, let's come back to the expression of the updates in the case of the Bayesian Learning Rule: contrarily to classical gradient descent, the BLR puts the expectation back in, thus modifying the optimality condition, and for a gaussian posterior with fixed covariance to 1 and mean m:

$$m \leftarrow m - \rho \nabla_m \mathbb{E}_q[l(\theta)] \Rightarrow \nabla_m \mathbb{E}_{q*}[l(\theta)] = 0$$

This is slightly different from the gradient descent, as it is the gradient of the convoluted loss that must be set to 0. The BLR setting has for effect to push the optimal solution towards flatter regions of the loss, therefore providing more robust solutions. Let's make it more clear using the Bonnet's theorem: in fact, under some regular assumptions on the loss function, Bonnet's theorem gives us $\nabla_m \mathbb{E}_{q*}[l(\theta)] = \mathbb{E}_{q*}[\nabla_\theta l(\theta)]$. Thus, at optimality, the expected gradient of the loss with respect to the parameter must be set to 0, which forces the optimal solution to shift towards flatter directions of the loss, to compensate for the 'walls' of the loss. The gradient with respect to the parameter needs somehow to cancel out, and regions with high absolute gradients force to give more importance to flat regions. Therefore, Bayesian solution tends to avoid sharp minimum, rather preferring potentially worst but more robust ones. The variational solution is expected to be more robust due to the averaging over $q_*(\theta)$.

The simple observation that putting the expectation back in the classical algorithms framework enables to gain more stability and robustness in the solution incites to derive new Bayesian versions of deep learning algorithms to better control the uncertainties. In fact, let's take a look at classical adaptive learning-rate algorithms, see e.g., [15] who uses exponential smoothing to improve stability, and a learnable scaling vector s_t to improve learning:

$$\theta_{t+1} \leftarrow \theta_t - \alpha \frac{1}{s_{t+1}} \cdot \hat{\nabla}_\theta l(\theta_t) \text{ with } s_{t+1} \leftarrow (1-\beta) s_t + \beta \text{diag}[\hat{\nabla}^2_\theta l(\theta_t)]$$

These adaptive learning approach gave birth to many variants algoritms, such as RMSProp, or Adam.
These versions can be retrieved pretty easily with the Bayesian Learning Rule, optimizing over candidates of the form $q(\theta) = \mathcal{N}(\theta|m, S^{-1})$, with S constrained to be a diagonal matrix with vector s for diagonal. In that case, it is possible to show that the BLR gives birth to the following updates:

$$m_{t+1} \leftarrow m_t - \rho_t \frac{1}{s_{t+1}} \cdot \mathbb{E}_{q_t}[\hat{\nabla}_\theta l(\theta)], \text{ with } s_{t+1} \leftarrow (1-\rho_t) s_t + \rho_t \mathbb{E}_{q_t}[\text{diag}(\hat{\nabla}^2_\theta l(\theta))]$$

Therefore, as a classical gradient descent, or even any non-bayesian algorithm would simply give one 'optimal' value per iteration, Bayesian approach provides us with a mean and a variance, which enables to sample many versions of the parameters, and therefore knowing where the data is or is not. We illustrate this point below in a very toy example on Fig.2, knowing an MLP classification problem, tackled with a 'deterministic' gradient descent optimization on the one hand, and the Bayesian version on the other hand.

2.3 Application to NoisyNet

We have explored the relationship between the Bayesian Learning Rule (BLR) and the NoisyNet approach [11] within the context of Reinforcement Learning (RL). In RL, it is crucial for an agent to explore its environment to devise strategies. This exploration involves taking sub-optimal actions to uncover new states. A common technique involves selecting actions randomly with a gradually decreasing probability to encourage exploration. NoisyNet introduces a method to embed noise directly into the decision-making process, moving beyond simple random actions. Specifically, it adds Gaussian noise $\mathcal{N}(m, S^{-1})$ to the neural network's output, where this noise parameter is learned. As a result, the network's output becomes a probabilistic action choice, aligning with Bayesian principles for loss minimization: $\min \mathrm{E}_{q_\lambda}[l(y, f_\lambda(\theta)]$. However, the optimization strategy outlined in [11] utilizes straightforward gradient descent, namely $\mathbb{E}\left[\nabla_{m,S^{-1}} L\left(m + (S^{-1} \odot \epsilon)\right)\right]$. We argue that this may not be the most effective approach since the goal is to optimize parameters of a distribution within the minimal exponential family. Instead, we propose using a natural gradient update, which could be more suitable for achieving this objective, that is $\mathbb{E}\left[\nabla_\mu L\left(m + (S^{-1} \odot \epsilon)\right)\right] |_{\mu=(m,S^{-1}+mm^T)}$.

Our experimental comparison involved using a NoisyNet model with two different optimizers: Adam [8], and our adapted version of VAdam [5], an optimizer based on the (BLR). Our adaptation of VAdam was necessary to accommodate direct noise injection into the model, rather than adding noise during model re-evaluation. The pseudo-code of our optimizer is provided in Appendix. The algorithm used is a classic DQN without target network. We tested our optimizers on the CartPole's Gym environnement. Even though VAdam is not a state of the art bayesian optimizer (compared to VOGN [4]), it matches the results of Adam (Fig 3). Furthermore, it provides more stable solutions compared to Adam, suggesting that the optimizer converged to more stable parameters.

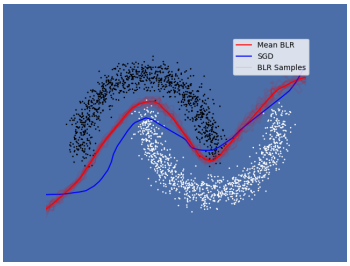

Figure 2: The Bayesian approach optimizes over the mean and the standard deviation of the parameters, enabling to better control approximations and uncertainties. Here, the weights of a from-scratch-MLP with two layers have Gaussian posterior with learnable mean and standard deviation, deriving a variation of the RMSProp algorithm.

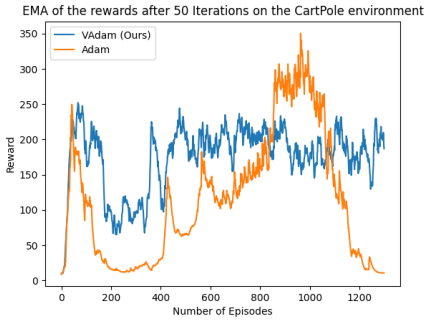

Figure 3: Convergence of the Agent on a CartPole environment based on the used optimizer.

3 On the universality of BLR

3.1 Retrieving the Stochastic Average Gradient

As previously described many learning-algorithms can be derived from the BLR. Most of them are described in the studied paper [7]. However this framework enables to derive other algorithms and can be seen as a methodology to derive algorithms in the Bayesian framework. To illustrate the universality of this method and its adaptation to new algorithms we decided to derive a slightly modified version of Stochastic Gradient Descent (SGD): Stochastic Average Gradient (SAG).

The advantage of SGD compared to standard gradient descent is its computational efficiency and its better generalization because of the noise introduced by the random sampling which can help escape local minima. Furthermore, full gradient descent is computationally more expensive than SGD. However, SGD suffers from a high variance in parameter updates due to noisy gradients. Since the bayesian framework also adds variance to the algorithm, SGD may become even more unstable. SAG algorithm has been thought to be less computationally costly than GD and less noisy (with a smaller variance) than SGD. In the particular case of the bayesian framework, having a learning-algorithm with less variance could be very interesting in order not to add too much noise while keeping the advantages of the stochastic approximation.

To derive SAG using the Bayesian Learning Rule we take a Gaussian, $q(\theta) = \mathbf{N}(\theta|\mathbf{m}, \mathbf{S^{-1}})$ with fixed covariance matrix as posterior approximation, as for SGD and GD. The mean m remains unknown and is learned through iterations with natural gradients. Then, the natural-gradient approximation that we apply can be called "average

stochastic approximation". Indeed, instead of taking the mini-batch gradient (the gradient calculated on a mini-batch randomly drawn at each iteration from the full dataset using a uniform distribution), we take the previously calculated gradient and update just the equivalent of the mini-batch gradient. As for SGD, the natural parameter λ is equal to Sm and $\mu = m$. We first define the mini-batch gradient with $\mathcal{M}$ the uniformly sampled minibatch.

$$\hat{\nabla}_\theta \bar{l}(\theta) = \frac{N}{M} \sum_{i \in \mathcal{M}} \nabla_\theta l(y_i, f_\theta(x_i)) + \nabla_\theta R(\theta)$$

Then, the update of SAG using the Bayesian Learning Rule becomes:

$$m_{t+1} \leftarrow m_t - \rho_t S^{-1}(g_{t-1} - \hat{\nabla}_\theta \bar{l}(\theta)|^{\mathcal{M}_t}_{\theta=m_{t-1}} + \hat{\nabla}_\theta \bar{l}(\theta)|^{\mathcal{M}_t}_{\theta=m_t}) \tag{5}$$

, with ρ_t the learning rate, $\mathcal{M}_t$ the randomly sampled mini-batch at time step t and g_{t-1} a buffer which store the sum of the already computed gradient at this stage. If gradients over some of the samples of $\mathcal{M}_t$ have not already been computed then these gradients are set to 0 in $\hat{\nabla}_\theta \bar{l}(\theta)|^{\mathcal{M}_t}_{\theta=m_{t-1}}$.

In practice, SAG, SGD and GD were tested on a classification task. The dataset used is the cancer dataset from Scikit-learn and a Support-Vector Machine algorithm is used. Then, our learning algorithms are used to minimize the smoothed Hinge loss in order to find the best hyperplane separating the dataset in the two accurate classes ('malignant' and 'benign'). The bayesian framework means that parameters are updated according to the formula above and sampled from the updated distribution at each step.

We illustrate (Fig.2) that the characteristics of these three algorithms, what differentiates them, are also conserved when they are implemented using the bayesian framework. Indeed, GD converges in fewer iterations than the two other. On the other hand, GD runs in $3m54s$, SGD in $4.6s$ and SAG in $11.3s$. Since the setting of our experiment is easy to learn from, other characteristics of these algorithms have not been tested. However, BLR enables to reproduce the algorithms in theory but also to implement algorithms with identical characteristics.

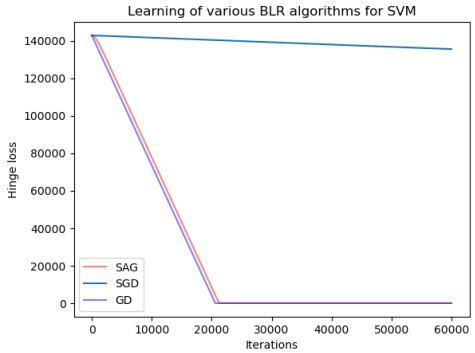

Figure 4: Evolution of the loss during learning for the three studied algorithms, implemented using the Bayesian Learning Rule. The mini-batch size has been fixed to 10 (1.6% of the datapoints), $S^{-1} = 10^{-4}\mathbf{I}_n$ and the learning rate is set to 0.001.

4 Conclusion - Limitations

We highlight a significant limitation of the approach of the BLR, specifically its reliance on minimal exponential families. The feasibility of updates, such as those described by Equation 4, is greatly facilitated by the tractability of minimal exponential families. However, this reliance narrows the scope of applicability of the methods discussed. For example, deriving similar updates for a uniform distribution over $[0, \lambda]$ is not feasible. This observation underscores a promising yet largely unexplored research avenue, suggesting that significant efforts are required to extend these methodologies to broader classes of posterior distributions. Further work could try to derive such approximations for less restricted classes of posteriors. For instance, [9] extends the natural gradient update to mixture of minimal exponential families.

Another critical aspect that draws our attention is the theoretical iterative update on the natural gradient λ as mentioned in Equation 4, which does not consider the constraints inherent to the parameters. Taking the example of a posterior assumed to be $\mathcal{N}(m, S^{-1})$, the original formulation for the derivation of the Gauss-Newton algorithm. Specifically, the update formula for S is expressed as $S_{t+1} \leftarrow (1 - \rho_t)S_t + \rho_t \mathbb{E}_{q_t}[\nabla_\theta l(\theta)]$. This update overlooks the crucial constraint that the covariance matrix must remain positive definite. This oversight has been recently

addressed for positive constraints [10], ensuring the practical applicability and theoretical soundness of these update mechanisms.

Nevertheless, we show through our work that the Bayesian Learning Rule enables to reproduce several algorithms while improving their results by enhancing their robustness. Furthermore, we test the advantages of the BLR in practice and propose new derived algorithms or applications with SAG and NoisyNet.

References

[1] Shun-ichi Amari. Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5(4-5):185–196, 1993.

[2] Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

[3] Roger Grosse. Natural gradient. 2017.

[4] Anirudh Jain Runa Eschenhagen Richard E. Turner Rio Yokota Mohammad Emtiyaz Khan Kazuki Osawa, Siddharth Swaroop. Practical deep learning with bayesian principles. *NeurIPS*, 2019.

[5] Mohammad Khan, Didrik Nielsen, Voot Tangkaratt, Wu Lin, Yarin Gal, and Akash Srivastava. Fast and scalable bayesian deep learning by weight-perturbation in adam. In *International conference on machine learning*, pages 2611–2620. PMLR, 2018.

[6] Mohammad Emtiyaz Khan and Didrik Nielsen. Fast yet simple natural-gradient descent for variational inference in complex models, 2018.

[7] Mohammad Emtiyaz Khan and Håvard Rue. The bayesian learning rule, 2023.

[8] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[9] Wu Lin, Mohammad Emtiyaz Khan, and Mark Schmidt. Fast and simple natural-gradient variational inference with mixture of exponential-family approximations. In *International Conference on Machine Learning*, pages 3992–4002. PMLR, 2019.

[10] Wu Lin, Mark Schmidt, and Mohammad Emtiyaz Khan. Handling the positive-definite constraint in the bayesian learning rule. In *International conference on machine learning*, pages 6116–6126. PMLR, 2020.

[11] Bilal Piot Jacob Menick Ian Osband Alex Graves Vlad Mnih Remi Munos Demis Hassabis Olivier Pietquin Charles Blundell Shane Legg Meire Fortunato, Mohammad Gheshlaghi Azar. Noisy networks for exploration. In *International Conference on Learning Representations*, 2018. arXiv:1706.10295 [cs.LG].

[12] Behnam Neyshabur. Implicit regularization in deep learning. *arXiv preprint arXiv:1709.01953*, 2017.

[13] Yann Ollivier, Ludovic Arnold, Anne Auger, and Nikolaus Hansen. Information-geometric optimization algorithms: A unifying picture via invariance principles, 2017.

[14] David G. T. Barrett Soham De Samuel L. Smith, Benoit Dherin. On the origin of implicit regularization in stochastic gradient descent. *ICLR*, 2021.

[15] L. Bottou Y. LeCun and G. B. Orrand K-R. M¨uller. Efficient backprop. *Neural Networks: Tricks of the Trade,*, 1998.

5 Appendix

5.1 Results in the Exponential Family

Justification of $\mu = \nabla_\lambda A(\lambda)$

Given $q_\lambda(\theta) = h(\theta)\exp(\langle\lambda, T(\theta)\rangle - A(\lambda))$, and considering that $\int q_\lambda(\theta)\,d\theta = 1$, we find that $A(\lambda) = \ln\left(\int h(\theta)\exp(\langle\lambda, T(\theta)\rangle)\,d\theta\right)$.

From this, we deduce that $\frac{\partial A(\lambda)}{\partial \lambda_i} = \frac{\int h(\theta)\exp(\langle\lambda,T(\theta)\rangle)T_i(\theta)\,d\theta}{\int h(\theta)\exp(\langle\lambda,T(\theta)\rangle)\,d\theta}$.

This leads to the conclusion that $\frac{\partial A(\lambda)}{\partial \lambda_i} = \mathbb{E}_{q_\lambda}[T_i(\theta)] = (\mu)_i$, effectively demonstrating the justification for $\mu = \nabla_\lambda A(\lambda)$.

Justification of the use of the Fisher matrix

Suppose λ_t is a vector of the current parameter values, and we would like to find a new set of parameters λ_{t+1}. To do so we solve the following problem, where we add a constraint on the geometry of the data that we are studying, that is the KullBack-Leiger Divergence to avoid taking an infinite step in the direction of the negative gradient.

$$\lambda_{t+1} = \text{argmin}\left(f(\lambda_t) + \nabla f(\lambda_t)^\top(\lambda - \lambda_t) + D_{KL}(q_\lambda \| q_{\lambda_t}\right).$$

In the general framework where we replace D_{KL} by another metric G, it has been shown [2] that the steepest descent direction in a space with metric tensor G is given by $G^{-1}(\lambda_t)\nabla f(\lambda_t)$. Furthermore, deriving a second-order Taylor expansion of KL divergence turns out to be the Fisher information matrix F. This means that locally around λ_t, we have

$$D_{KL}(q_\lambda \| q_{\lambda_t}) \approx F(\lambda).$$

where

$$F = \mathbb{E}_{\theta\sim q_\lambda}\left[(\nabla_\lambda \log q_\lambda(\theta)(\nabla_\lambda \log q_\lambda(\theta))^\top\right].$$

Thus, our update for the natural gradient in this setting is then

$\lambda_{t+1} = \lambda_t - F^{-1}(\lambda_t)\nabla f(\lambda_t)$.

Algorithm 1 Our VAdam Optimizer

while not converged **do**
  Randomly sample a data example D_i
  for j in NB_MC_eval **do**
    $\theta \leftarrow \mu + \sigma \circ \epsilon,$ where $\epsilon \sim \mathcal{N}(0, I), \sigma \leftarrow \frac{1}{\sqrt{Ns+\lambda}}$
    $f_{\theta_t} \leftarrow f_\theta$ ▷ f is the NoisyNetwork
    $g \leftarrow -\nabla \log p(D_i|\theta)$
    $m \leftarrow \gamma_1 m + (1-\gamma_1)(g + \lambda\mu/N)$
    $s \leftarrow \gamma_2 s + (1-\gamma_2)(g \circ g)$
    $\hat{m} \leftarrow \frac{m}{1-\gamma^t}, \hat{s} \leftarrow \frac{s}{\sqrt{1-\gamma^t}}$
    $\mu \leftarrow \mu - \alpha\frac{\hat{m}}{\sqrt{\hat{s}+\lambda/N}}$
    $t \leftarrow t+1$
  end for
end while