The Bayesian Learning Rule

Mohammad Emtiyaz Khan
RIKEN Center for AI Project
Tokyo, Japan
`emtiyaz.khan@riken.jp`

Håvard Rue
CEMSE Division, KAUST
Thuwal, Saudi Arabia
`haavard.rue@kaust.edu.sa`

Abstract

We show that many machine-learning algorithms are specific instances of a single algorithm called the *Bayesian learning rule*. The rule, derived from Bayesian principles, yields a wide-range of algorithms from fields such as optimization, deep learning, and graphical models. This includes classical algorithms such as ridge regression, Newton's method, and Kalman filter, as well as modern deep-learning algorithms such as stochastic-gradient descent, RMSprop, and Dropout. The key idea in deriving such algorithms is to approximate the posterior using candidate distributions estimated by using natural gradients. Different candidate distributions result in different algorithms and further approximations to natural gradients give rise to variants of those algorithms. Our work not only unifies, generalizes, and improves existing algorithms, but also helps us design new ones.

1 Introduction

1.1 Learning-algorithms

Machine Learning (ML) methods have been extremely successful in solving many challenging problems in fields such as computer vision, natural-language processing and artificial intelligence (AI). The main idea is to formulate those problems as prediction problems, and learn a model on existing data to predict the future outcomes. For example, to design an AI agent that can recognize objects in an image, we can collect a dataset with N images $\boldsymbol{x}_i \in \mathbb{R}^D$ and object labels $y_i \in \{1, 2, \ldots, K\}$, and learn a model $f_{\boldsymbol{\theta}}(\boldsymbol{x})$ with parameters $\boldsymbol{\theta} \in \mathbb{R}^P$ to predict the label for a new image. Learning algorithms are often employed to estimate the parameters $\boldsymbol{\theta}$ using the principle of trial-and-error, e.g., by using Empirical Risk Minimization (ERM),

$$\boldsymbol{\theta}_* = \arg\min_{\boldsymbol{\theta}} \bar{\ell}(\boldsymbol{\theta}) \quad \text{where} \quad \bar{\ell}(\boldsymbol{\theta}) = \sum_{i=1}^{N} \ell(y_i, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)) + R(\boldsymbol{\theta}). \tag{1}$$

Here, $\ell(y, f_{\boldsymbol{\theta}}(\boldsymbol{x}))$ is a loss function that encourages the model to predict well and $R(\boldsymbol{\theta})$ is a regularizer that prevents it from overfitting. A wide-variety of such learning-algorithms exist in the literature to solve a variety of learning problems, for example, ridge regression, Kalman filters, gradient descent, and Newton's method. These algorithms play a key role in the success of modern ML.

Learning-algorithms are often derived by borrowing and combining ideas from a diverse set of fields, such as statistics, optimization, and computer science. For example, the field of probabilistic graphical models [Koller and Friedman, 2009, Bishop, 2006] uses popular algorithms such as ridge-regression [Hoerl and Kennard, 1970], Kalman filters [Kalman, 1960], Hidden Markov Models [Stratonovich, 1965], and Expectation-Maximization (EM) [Dempster et al., 1977]. The field of Approximate Bayesian Inference builds upon such algorithms to perform inference on complex graphical models, e.g., algorithms

such as Laplace's method [Laplace, 1986, Tierney and Kadane, 1986, Rue and Martino, 2007, Rue et al., 2009, 2017], stochastic variational inference (SVI) [Hoffman et al., 2013, Sato, 2001], Variational message passing (VMP) [Winn and Bishop, 2005] etc. Similarly, the field of continuous optimization has its own popular methods such as gradient descent [Cauchy, 1847], Newton's method, and mirror descent [Nemirovski and Yudin, 1978] algorithms, and deep-learning (DL) methods use them to design new optimizers that work with massive models and datasets, e.g., stochastic-gradient descent [Robbins and Monro, 1951], RMSprop [Tieleman and Hinton, 2012], Adam [Kingma and Ba, 2015], Dropout regularization [Srivastava et al., 2014], and Straight-Through Estimator (STE) [Bengio et al., 2013]. Such mixing of algorithms from diverse fields is a strength of the ML community, and our goal here is to provide common principles to unify, generalize, and improve the existing algorithms.

1.2 The Bayesian learning rule

We show that a wide-range of well-known learning-algorithms from a variety of fields are all specific instances of a single learning algorithm derived from Bayesian principles. The starting point, is the variational formulation by Zellner [1988], which is an extension of Eq. 1 to optimize over a well-defined candidate distribution $q(\boldsymbol{\theta})$, and for which the minimizer

$$q_*(\boldsymbol{\theta}) = \operatorname*{arg\,min}_{q(\boldsymbol{\theta})} \quad \mathbb{E}_q \left[\sum_{i=1}^N \ell(y_i, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)) \right] + \mathbb{D}_{KL}[q(\boldsymbol{\theta}) \,\|\, p(\boldsymbol{\theta})] \tag{2}$$

defines a generalized posterior [Zhang, 1999, Catoni, 2007, Bissiri et al., 2016] in lack of a precise likelihood. The prior distribution is related to the regularizer, $p(\boldsymbol{\theta}) \propto \exp(-R(\boldsymbol{\theta}))$, and $\mathbb{D}_{KL}[\cdot \,\|\, \cdot]$ is the Kullback-Leibler Divergence (KLD). In the case where $\exp(-\ell(y_i, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)))$ is proportional to the likelihood for $y_i, \forall i$, then $q_*(\boldsymbol{\theta})$ is the posterior distribution for $\boldsymbol{\theta}$ [Zellner, 1988] (see App. A).

The learning algorithms we will derive from the above Bayesian formulation, needs two components.

1. A (sub-)class of distributions $\mathcal{Q}$ to optimize over. In our discussion we will assume $\mathcal{Q}$ to be the set of a regular and minimal exponential family

$$q(\boldsymbol{\theta}) = h(\boldsymbol{\theta}) \exp\left[\langle \boldsymbol{\lambda}, \boldsymbol{T}(\boldsymbol{\theta}) \rangle - A(\boldsymbol{\lambda})\right]$$

where $\boldsymbol{\lambda} \in \Omega \subset \mathbb{R}^M$ are the natural (or canonical) parameters in a non-empty open set Ω for which the cumulant (or log partition) function $A(\boldsymbol{\lambda})$ is finite, strictly convex and differentiable over Ω. Further, $\boldsymbol{T}(\boldsymbol{\theta})$ is the sufficient statistics, $\langle \cdot, \cdot \rangle$ is an inner product and $h(\boldsymbol{\theta})$ is the base measure. The expectation parameters are $\boldsymbol{\mu} = \mathbb{E}_q[\boldsymbol{T}(\boldsymbol{\theta})] \in \mathcal{M}$, and is a (bijective) function of $\boldsymbol{\lambda}$. Examples later will include the multivariate Normal distribution and the Bernoulli distribution.

2. An optimizing algorithm, called the *Bayesian learning rule* (BLR), that locates the best candidate $q_*(\boldsymbol{\theta})$ in $\mathcal{Q}$, by updating the candidate $q_t(\boldsymbol{\theta})$ with natural parameters $\boldsymbol{\lambda}_t$ at iteration t,

$$\boldsymbol{\lambda}_{t+1} \leftarrow \boldsymbol{\lambda}_t - \rho_t \widetilde{\nabla}_{\boldsymbol{\lambda}} \left[\mathbb{E}_{q_t}\left(\bar{\ell}(\boldsymbol{\theta})\right) - \mathcal{H}(q_t) \right]. \tag{3}$$

Here, $\mathcal{H}(q) = \mathbb{E}_q[-\log q(\boldsymbol{\theta})]$ is the entropy and $\rho_t > 0$ is a sequence of learning rates. The updates use the *natural-gradients* [Amari, 1998] (denoted by $\widetilde{\nabla}_{\boldsymbol{\lambda}}$),

$$\widetilde{\nabla}_{\boldsymbol{\lambda}} \mathbb{E}_{q_t}(\cdot) = \boldsymbol{F}(\boldsymbol{\lambda}_t)^{-1} \left[\nabla_{\boldsymbol{\lambda}} \mathbb{E}_q(\cdot)|_{\boldsymbol{\lambda}=\boldsymbol{\lambda}_t} \right] = \nabla_{\boldsymbol{\mu}} \mathbb{E}_q(\cdot)|_{\boldsymbol{\mu}=\nabla_{\boldsymbol{\lambda}} A(\boldsymbol{\lambda}_t)}, \tag{4}$$

which rescale the vanilla gradients $\nabla_{\boldsymbol{\lambda}}$ with the Fisher information matrix (FIM) $\boldsymbol{F}(\boldsymbol{\lambda}) = \nabla_{\boldsymbol{\lambda}}^2 A(\boldsymbol{\lambda})$ to adjust for the geometry in the parameter space. The second equality follows from the chain

rule to express natural-gradients as vanilla gradients with respect to $\boldsymbol{\mu} = \nabla_{\boldsymbol{\lambda}} A(\boldsymbol{\lambda})$. Throughout, we will use this property to simplify natural-gradient computations. These details are discussed in Sec. 2, where we show that the BLR update can also be seen as a mirror descent algorithm, where the geometry of the Bregman divergence is dictated by the chosen exponential-family. We also assume $\boldsymbol{\lambda}_t \in \Omega$ for all t, which in practice might require a line-search or a projection step.

The main message of this paper, is that many well-known learning-algorithms, such as those used in optimization, deep learning, and machine learning in general, can now be derived directly following the above scheme using a single algorithm (BLR) that optimizes Eq. 2. Different exponential families $\mathcal{Q}$ give rise to different algorithms, and within those, various approximations to natural-gradients that is needed, give rise to many variants.

Our use of natural-gradients here is not a matter of choice. In fact, natural-gradients are inherently present in *all solutions of the Bayesian objective* in Eq. 2. For example, a solution of Eq. 2 or equivalently a fixed point of Eq. 3, satisfies the following,

$$\nabla_{\boldsymbol{\mu}} \mathbb{E}_{q_*}[\bar{\ell}(\boldsymbol{\theta})] = \nabla_{\boldsymbol{\mu}} \mathcal{H}(q_*), \text{ which implies } \widetilde{\nabla}_{\boldsymbol{\lambda}} \mathbb{E}_{q_*}[-\bar{\ell}(\boldsymbol{\theta})] = \boldsymbol{\lambda}_*, \tag{5}$$

for candidate with constant base-measure. This is obtained by setting the gradient of Eq. 2 to 0, then noting that $\nabla_{\boldsymbol{\mu}} \mathcal{H}(q) = -\boldsymbol{\lambda}$ (App. B), and then interchanging $\nabla_{\boldsymbol{\mu}}$ by $\widetilde{\nabla}_{\boldsymbol{\lambda}}$ (because of Eq. 4). In other words, natural parameter of the best $q_*(\boldsymbol{\theta})$ is equal to the natural gradient of the expected negative-loss. The importance of natural-gradients is entirely missed in the Bayesian/variational inference literature, including textbooks, reviews, tutorials on this topic [Bishop, 2006, Murphy, 2012, Blei et al., 2017, Zhang et al., 2018a] where natural-gradients are often put in a special category.

We will show that natural gradients retrieve essential higher-order information about the loss landscape which are then assigned to appropriate natural parameters using Eq. 5. The information-matching is due to the presence of the entropy term there, which is an important quantity for the optimality of Bayes in general [Jaynes, 1982, Zellner, 1988, Littlestone and Warmuth, 1994, Vovk, 1990], and which is generally absent in non-Bayesian formulations (Eq. 1). The entropy term in general leads to exponential-weighting in Bayes' rule. In our context, it gives rise to natural-gradients and, as we will soon see, automatically determines the complexity of the derived algorithm through the complexity of the class of distributions $\mathcal{Q}$, yielding a principled way to develop new algorithms.

Overall, our work demonstrates the importance of natural-gradients for algorithm design in ML. This is similar in spirit to Information Geometric Optimization [Ollivier et al., 2017], which focuses on the optimization of black-box, deterministic functions. In contrast, we derive generic learning algorithms by using the same Bayesian principles. The BLR we use is a generalization of the method proposed in Khan and Lin [2017], Khan and Nielsen [2018] specifically for variational inference. Here, we establish it as a general learning rule to derive many old and new learning algorithms. In this sense, the BLR can be seen as a variant of Bayes' rule, useful for generic ML problems.

1.3 Examples

To fix ideas, we will now use the BLR to derive three classical algorithms: gradient descent, Newton's method, and ridge regression. All of these are derived by choosing $\mathcal{Q}$ to be multivariate Gaussian.

1.3.1 Gradient descent

The gradient descent algorithm uses the following update

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \rho_t \nabla_{\boldsymbol{\theta}} \bar{\ell}(\boldsymbol{\theta}_t),$$

using only the first-order information $\nabla_{\boldsymbol{\theta}}\bar{\ell}(\boldsymbol{\theta}_t)$ (the gradient evaluated at $\boldsymbol{\theta}_t$). We choose $q(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{m},\boldsymbol{I})$, a multivariate Gaussian with unknown mean $\boldsymbol{m}$ and known covariance matrix set to $\boldsymbol{I}$ for simplicity (general covariance case is given in Eq. 35). The natural and expectation parameters are now both $\boldsymbol{m}$, and the base measure is $2\log h(\boldsymbol{\theta}) = -P\log(2\pi) - \boldsymbol{\theta}^T\boldsymbol{\theta}$. We use the following simplified form of the BLR, obtained by using the fact that $\nabla_{\boldsymbol{\mu}}\mathcal{H}(q) = -\boldsymbol{\lambda} - \nabla_{\boldsymbol{\mu}}\mathbb{E}_q(\log h(\boldsymbol{\theta}))$ (App. B), to get

$$\boldsymbol{\lambda}_{t+1} \leftarrow (1 - \rho_t)\boldsymbol{\lambda}_t - \rho_t \nabla_{\boldsymbol{\mu}}\mathbb{E}_{q_t}\left[\bar{\ell}(\boldsymbol{\theta}) + \log h(\boldsymbol{\theta})\right]. \tag{6}$$

By noting that $\nabla_{\boldsymbol{\mu}}\mathbb{E}_q[\log h(\boldsymbol{\theta})] = -\boldsymbol{\lambda}$, the update follows directly from the above BLR

$$\boldsymbol{m}_{t+1} \leftarrow \boldsymbol{m}_t - \rho_t \left.\nabla_{\boldsymbol{m}}\mathbb{E}_q\left[\bar{\ell}(\boldsymbol{\theta})\right]\right|_{\boldsymbol{m}=\boldsymbol{m}_t}. \tag{7}$$

This update is gradient descent but over the expected loss $\mathbb{E}_q\left[\bar{\ell}(\boldsymbol{\theta})\right]$. We can remove the expectation by using the first-order delta method [Dorfman, 1938, Ver Hoef, 2012] (App. C),

$$\nabla_{\boldsymbol{m}}\mathbb{E}_q\left[\bar{\ell}(\boldsymbol{\theta})\right] \approx \left.\nabla_{\boldsymbol{\theta}}\bar{\ell}(\boldsymbol{\theta})\right|_{\boldsymbol{\theta}=\boldsymbol{m}}.$$

With this, the BLR equals the gradient descent algorithm with $\boldsymbol{m}_t$ as the iterate $\boldsymbol{\theta}_t$. The interpretation is that if we give away the posterior averaging in the BLR and resort to a greedy approximation, we are back to a non-Bayesian algorithm to minimize a deterministic function. The two choices, first of the distribution $\mathcal{Q}$ and second of the delta method, give us gradient descent from the BLR.

1.3.2 Newton's method

Newton's method is a second-order method

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \left[\nabla_{\boldsymbol{\theta}}^2\bar{\ell}(\boldsymbol{\theta}_t)\right]^{-1}\left[\nabla_{\boldsymbol{\theta}}\bar{\ell}(\boldsymbol{\theta}_t)\right]. \tag{8}$$

which too can be derived from the BLR by expanding the class $\mathcal{Q}$ to $q(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{m},\boldsymbol{S}^{-1})$ with an unknown precision matrix $\boldsymbol{S}$. This example illustrates a property of the BLR: the complexity of the derived algorithm is directly related to the complexity of the exponential family $\mathcal{Q}$. Fixed covariances previously gave rise to gradient-descent (a first-order method) and by increasing the complexity where we also learn the precision matrix, the BLR reduces to a (more complex) second-order method.

The details are as follows. We use the simplified form Eq. 6 of the BLR. The base measure is a constant, and the natural and expectation parameters divide themselves into natural pairs

$$\begin{array}{llll}
\boldsymbol{\lambda}^{(1)} &= \boldsymbol{Sm}, & \boldsymbol{\mu}^{(1)} &= \mathbb{E}_q[\boldsymbol{\theta}] &= \boldsymbol{m}, \\
\boldsymbol{\lambda}^{(2)} &= -\frac{1}{2}\boldsymbol{S}, & \boldsymbol{\mu}^{(2)} &= \mathbb{E}_q[\boldsymbol{\theta}\boldsymbol{\theta}^{\top}] &= \boldsymbol{S}^{-1} + \boldsymbol{mm}^T.
\end{array} \tag{9}$$

The natural gradients can be expressed in terms of the gradient and Hessian of $\bar{\ell}(\boldsymbol{\theta})$,

$$\nabla_{\boldsymbol{\mu}^{(1)}}\mathbb{E}_q[\bar{\ell}(\boldsymbol{\theta})] = \nabla_{\boldsymbol{m}}\mathbb{E}_q[\bar{\ell}(\boldsymbol{\theta})] - 2\left[\nabla_{\boldsymbol{S}^{-1}}\mathbb{E}_q[\bar{\ell}(\boldsymbol{\theta})]\right]\boldsymbol{m} \qquad = \mathbb{E}_q[\nabla_{\boldsymbol{\theta}}\bar{\ell}(\boldsymbol{\theta})] - \mathbb{E}_q[\nabla_{\boldsymbol{\theta}}^2\bar{\ell}(\boldsymbol{\theta})]\boldsymbol{m}, \tag{10}$$

$$\nabla_{\boldsymbol{\mu}^{(2)}}\mathbb{E}_q[\bar{\ell}(\boldsymbol{\theta})] = \nabla_{\boldsymbol{S}^{-1}}\mathbb{E}_q[\nabla_{\boldsymbol{\theta}}\bar{\ell}(\boldsymbol{\theta})] \qquad = \frac{1}{2}\mathbb{E}_q[\nabla_{\boldsymbol{\theta}}^2\bar{\ell}(\boldsymbol{\theta})]. \tag{11}$$

These are obtained by using the chain-rule, and two identities called Bonnet's and Price's theorem respectively [Bonnet, 1964, Price, 1958, Opper and Archambeau, 2009, Rezende et al., 2014]; see App. D. The expressions show that the natural gradients contain the information about the first and second-order derivatives of $\bar{\ell}(\boldsymbol{\theta})$.

4

The BLR now turns into an online variant of Newton's method where the precision matrix contains an exponential-smoothed Hessian average and is used as a pre-conditioner to update the mean,

$$\boldsymbol{m}_{t+1} \leftarrow \boldsymbol{m}_t - \rho_t \boldsymbol{S}_{t+1}^{-1} \mathbb{E}_{q_t}\left[\nabla_{\boldsymbol{\theta}}\bar{\ell}(\boldsymbol{\theta})\right] \quad \text{and} \quad \boldsymbol{S}_{t+1} \leftarrow (1-\rho_t)\boldsymbol{S}_t + \rho_t \mathbb{E}_{q_t}\left[\nabla_{\boldsymbol{\theta}}^2 \bar{\ell}(\boldsymbol{\theta})\right]. \tag{12}$$

We can recover the classical Newton's method in three steps. First, apply the delta method (App. C),

$$\begin{aligned}
\mathbb{E}_q\left[\nabla_{\boldsymbol{\theta}}\bar{\ell}(\boldsymbol{\theta})\right] &= \nabla_{\boldsymbol{m}}\mathbb{E}_q[\bar{\ell}(\boldsymbol{\theta})] \approx \nabla_{\boldsymbol{\theta}}\bar{\ell}(\boldsymbol{\theta})\big|_{\boldsymbol{\theta}=\boldsymbol{m}} \\
\mathbb{E}_q\left[\nabla_{\boldsymbol{\theta}}^2\bar{\ell}(\boldsymbol{\theta})\right] &= 2\nabla_{\boldsymbol{S}^{-1}}\mathbb{E}_q[\bar{\ell}(\boldsymbol{\theta})] \approx \nabla_{\boldsymbol{\theta}}^2\bar{\ell}(\boldsymbol{\theta})\big|_{\boldsymbol{\theta}=\boldsymbol{m}}.
\end{aligned} \tag{13}$$

Second, set the learning rate to 1 which is justified when the loss is strongly convex or the algorithm is initialized close enough to the solution. Finally, treat the mean $\boldsymbol{m}_t$ as the iterate $\boldsymbol{\theta}_t$.

1.3.3 Ridge regression

Why do we get a second-order method when we increase the complexity of the Gaussian? This is due to the natural gradients which, depending on the complexity of the distribution, retrieve essential higher-order information about the loss landscape. We illustrate this now through the simplest non-trivial case of Ridge regression where the loss is quadratic: $\bar{\ell}(\boldsymbol{\theta}) = \frac{1}{2}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta})^\top(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}) + \frac{1}{2}\delta\boldsymbol{\theta}^\top\boldsymbol{\theta}$ with $\delta > 0$ as the regularization parameter, and the solution is available in closed-form:

$$\boldsymbol{\theta}_* = (\boldsymbol{X}^\top\boldsymbol{X} + \delta\boldsymbol{I})^{-1}\boldsymbol{X}^\top\boldsymbol{y}.$$

We note that the expected-loss is linear in $\boldsymbol{\mu}$, $\mathbb{E}_q\left[\bar{\ell}(\boldsymbol{\theta})\right] = -\boldsymbol{y}^\top\boldsymbol{X}\boldsymbol{\mu}^{(1)} + \text{Tr}\left[\frac{1}{2}\left(\boldsymbol{X}^T\boldsymbol{X} + \delta\boldsymbol{I}\right)\boldsymbol{\mu}^{(2)}\right]$, and therefore the natural-gradients are given by,

$$\nabla_{\boldsymbol{\mu}^{(1)}}\mathbb{E}_q\left[\bar{\ell}(\boldsymbol{\theta})\right] = -\boldsymbol{X}^T\boldsymbol{y} \quad \text{and} \quad \nabla_{\boldsymbol{\mu}^{(2)}}\mathbb{E}_q\left[\bar{\ell}(\boldsymbol{\theta})\right] = \frac{1}{2}\left(\boldsymbol{X}^T\boldsymbol{X} + \delta\boldsymbol{I}\right). \tag{14}$$

It is clear that they already contain parts of the solution $\boldsymbol{\theta}_*$, which is recovered by using Eq. 5

$$\boldsymbol{S}_*\boldsymbol{m}_* = \boldsymbol{X}^T\boldsymbol{y}, \qquad \boldsymbol{S}_* = \boldsymbol{X}^T\boldsymbol{X} + \delta\boldsymbol{I}.$$

and solving to get the mean $\boldsymbol{m}_* = \boldsymbol{\theta}_*$. By increasing the complexity of $\mathcal{Q}$, natural-gradients can retrieve appropriate higher-order information, which are then assigned to the corresponding natural parameters using Eq. 5. This is the main reason why different algorithms are obtained when we change the class $\mathcal{Q}$. We discuss this point further in Sec. 3 when relating Bayesian principles to those of optimization.

1.4 Outline of the rest of the paper

A full list of the learning algorithms derived in this paper is given in Table 1. The rest of the paper is organized as follows. In Sec. 2, we give two derivations of the BLR by using natural-gradient descent and mirror descent, respectively. In Sec. 3, we summarize the main principles for the derivation of existing optimization algorithms, and give guidelines based on entropy for the design of new multimodal-optimization algorithms. In Sec. 4, we discuss the derivation of existing deep-learning algorithms, as well as new algorithms for uncertainty estimation. In Sec. 5, we discuss the derivation of algorithms for Bayesian inference for both conjugate and non-conjugate models. In Sec. 6, we conclude.

2 Derivation of the Bayesian Learning Rule

This section contains two derivations of the BLR. First, we interpret it as a natural-gradient descent using a second order expansion of the KLD, which strengthens its intuition. Secondly, we do a more formal derivation using a mirror-descent algorithm leveraging the connection to Legendre-duality and where we can bypass the need for doing a second-order approximation of the KLD.

Table 1: A summary of learning algorithms derived from the BLR. Each algorithm is derived through specific approximations of the posterior and natural-gradient. New algorithms are marked with "(New)". Abbreviations: cov. → covariance, STE → Straight-Through-Estimator, VI → Variational Inference, VMP → Variational Message Passing.

Learning Algorithm	Posterior Approx.	Natural-Gradient Approx.	Sec.
Optimization Algorithms			
Gradient Descent	Gaussian (fixed cov.)	Delta method	1.3
Newton's method	Gaussian	——"——	1.3
Multimodal optimization (New)	Mixture of Gaussians	——"——	3.2
Deep-Learning Algorithms			
Stochastic Gradient Descent	Gaussian (fixed cov.)	Delta method, stochastic approx.	4.1
RMSprop/Adam	Gaussian (diagonal cov.)	Delta method, stochastic approx., Hessian approx., square-root scaling, slow-moving scale vectors	4.2
Dropout	Mixture of Gaussians	Delta method, stochastic approx., responsibility approx.	4.3
STE	Bernoulli	Delta method, stochastic approx.	4.5
Online Gauss-Newton (OGN) (New)	Gaussian (diagonal cov.)	Gauss-Newton Hessian approx. in Adam & no square-root scaling	4.4
Variational OGN (New)	——"——	Remove delta method from OGN	4.4
BayesBiNN (New)	Bernoulli	Remove delta method from STE	4.5
Approximate Bayesian Inference Algorithms			
Conjugate Bayes	Exp-family	Set learning rate $\rho_t = 1$	5.1
Laplace's method	Gaussian	Delta method	4.4
Expectation-Maximization	Exp-Family + Gaussian	Delta method for the parameters	5.2
Stochastic VI (SVI)	Exp-family (mean-field)	Stochastic approx., local $\rho_t = 1$	5.3
VMP	——"——	$\rho_t = 1$ for all nodes	5.3
Non-Conjugate VMP	——"——	——"——	5.3
Non-Conjugate VI (New)	Mixture of Exp-family	None	5.4

2.1 Bayesian learning rule as natural-gradient descent

Given the objective $\mathcal{L}(\boldsymbol{\lambda}) = \mathbb{E}_q[\bar{\ell}(\boldsymbol{\theta}) + \log q(\boldsymbol{\theta})]$ in Eq. 2, the classical gradient-descent algorithm performs the following update:

$$\boldsymbol{\lambda}_{t+1} \leftarrow \boldsymbol{\lambda}_t - \rho_t \nabla_{\boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{\lambda}_t). \tag{15}$$

The insight motivating natural-gradient algorithms, is that the update Eq. 15 solves

$$\boldsymbol{\lambda}_{t+1} \leftarrow \arg\min_{\boldsymbol{\lambda}} \; \langle \nabla_{\boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{\lambda}_t), \boldsymbol{\lambda} \rangle + \frac{1}{2\rho_t} \|\boldsymbol{\lambda} - \boldsymbol{\lambda}_t\|_2^2, \qquad (16)$$

revealing the implicit Euclidean penalty of changes in the parameters. The parameters $\boldsymbol{\lambda}_t$ parameterize a probability distributions, and therefore their updates should be penalized based on the distance in the space of distributions. Distance between two parameter configurations might be a poor measure of the distance between the corresponding distributions. Natural-gradient algorithms use instead an alternative penalty, and using the KLD [Martens, 2020, Pascanu and Bengio, 2013] we get

$$\boldsymbol{\lambda}_{t+1} \leftarrow \arg\min_{\boldsymbol{\lambda}} \; \langle \nabla_{\boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{\lambda}_t), \boldsymbol{\lambda} \rangle + \frac{1}{\rho_t} \mathbb{D}_{KL}[q(\boldsymbol{\theta}) \, \| \, q_t(\boldsymbol{\theta})], \qquad (17)$$

To obtain a closed form update, we use a second order expansion of the KLD-term using that the Hessian is the Fisher information matrix of $q(\boldsymbol{\theta})$, $\boldsymbol{F}(\boldsymbol{\lambda})$, and we arrive at the natural-gradient descent algorithm [Amari, 1998],

$$\boldsymbol{\lambda}_{t+1} \leftarrow \boldsymbol{\lambda}_t - \rho_t \boldsymbol{F}(\boldsymbol{\lambda}_t)^{-1} \, \nabla_{\boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{\lambda}_t). \qquad (18)$$

The descent direction $\boldsymbol{F}(\boldsymbol{\lambda}_t)^{-1} \, \nabla_{\boldsymbol{\lambda}_t} \mathcal{L}(\boldsymbol{\lambda}_t)$ in this case is referred to as the natural-gradient. This argument here is slightly different from the case maximum-likelihood estimation [Amari, 1998], but the update still holds. This gives us the BLR update Eq. 3.

The following property is useful to compute natural gradients [Malagò et al., 2011, Raskutti and Mukherjee, 2015],

$$\widetilde{\nabla}_{\boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{\lambda}) = \boldsymbol{F}(\boldsymbol{\lambda})^{-1} \, \nabla_{\boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{\lambda}) = \nabla_{\boldsymbol{\mu}} \tilde{\mathcal{L}}(\boldsymbol{\mu}) \qquad (19)$$

which follows directly from the chain-rule using that $\boldsymbol{\mu} = \nabla_{\boldsymbol{\lambda}} A(\boldsymbol{\lambda})$ and $\boldsymbol{F}(\boldsymbol{\lambda}) = \nabla^2_{\boldsymbol{\lambda}\boldsymbol{\lambda}} A(\boldsymbol{\lambda})$ [Nielsen and Garcia, 2009]. The $\tilde{\mathcal{L}}(\boldsymbol{\mu}) = \mathcal{L}(\boldsymbol{\lambda})$ is a reparameterized function expressed in terms of $\boldsymbol{\mu}$, but we do not have to explicitly compute this function; see Eqs. 10 and 11 for example.

2.2 Bayesian learning rule as mirror-descent

In this section we will derive the BLR using a mirror-descent formulation, similarly to [Raskutti and Mukherjee, 2015], but for a Bayesian objective. In short, a mirror-descent step defined in the space of $\boldsymbol{\mu}$ equals a natural-gradient step in the space of $\boldsymbol{\lambda}$ (the update Eq. 18). This is a consequence of the Legendre-duality between the spaces of $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$, and is related to the dual-flat Riemannian structures [Amari, 2016] employed in information geometry. Unlike the previous derivation, this derivation is more direct and obtained without any second-order approximation of the KLD, and it reveals that certain parameterizations should be preferred for a natural-gradient descent over Bayesian objectives.

There is a duality between the space of $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$, since $\boldsymbol{\mu} = \nabla_{\boldsymbol{\lambda}} A(\boldsymbol{\lambda})$ is a bijection. An alternative view of this mapping, is to use the Legendre transform of $A(\boldsymbol{\lambda})$,

$$A^*(\boldsymbol{\mu}) = \sup_{\boldsymbol{\lambda}' \in \Omega} \langle \boldsymbol{\mu}, \boldsymbol{\lambda}' \rangle - A(\boldsymbol{\lambda}'),$$

from which the mapping follows by setting the gradient of the right-hand-side to zero. The reverse mapping is $A(\boldsymbol{\lambda}) = \sup_{\boldsymbol{\mu}' \in \mathcal{M}} \langle \boldsymbol{\mu}', \boldsymbol{\lambda} \rangle - A^*(\boldsymbol{\mu}')$, hence, $\boldsymbol{\lambda} = \nabla_{\boldsymbol{\mu}} A^*(\boldsymbol{\mu})$.

The expectation parameters $\boldsymbol{\mu}$ provide a dual coordinate system to specify the exponential family with natural parameter $\boldsymbol{\lambda}$, and using $\boldsymbol{\mu}$ we express $q(\boldsymbol{\theta})$ as [Banerjee et al., 2005] given below:

$$q(\boldsymbol{\theta}) = h(\boldsymbol{\theta}) \exp\left[-\mathbb{D}_{A^*}(\boldsymbol{T}(\boldsymbol{\theta}) \| \boldsymbol{\mu}) + A^*(\boldsymbol{T}(\boldsymbol{\theta}))\right] \qquad (20)$$

where $\mathbb{D}_{A^*}(\boldsymbol{\mu}_1\|\boldsymbol{\mu}_2) = A^*(\boldsymbol{\mu}_1) - A^*(\boldsymbol{\mu}_2) - \langle \boldsymbol{\mu}_1 - \boldsymbol{\mu}_2, \nabla_{\boldsymbol{\mu}} A^*(\boldsymbol{\mu}_2) \rangle$ is the Bregman divergence defined using the function $A^*(\boldsymbol{\mu})$. The relationship between the Bregman and KL divergence is

$$\mathbb{D}_{KL}[q_1(\boldsymbol{\theta}) \| q_2(\boldsymbol{\theta})] = \mathbb{D}_{A^*}(\boldsymbol{\mu}_1\|\boldsymbol{\mu}_2) = \mathbb{D}_A(\boldsymbol{\lambda}_2\|\boldsymbol{\lambda}_1). \tag{21}$$

The Bregman divergence is equal to the KL divergence between the corresponding distributions, which can also be measured in the dual space using the (swapped order of the) natural parameters. We can then use these Bregman divergences to measure the distances in the two dual spaces.

The relationship with the KLD enables us to express natural gradient descent in one space as a mirror descent in the dual space. More specifically, the following mirror descent update in the space $\mathcal{M}$ is equivalent to the natural gradient descent Eq. 18 in the space of Ω,

$$\boldsymbol{\mu}_{t+1} \leftarrow \arg\min_{\boldsymbol{\mu} \in \mathcal{M}} \langle \nabla_{\boldsymbol{\mu}} \tilde{\mathcal{L}}(\boldsymbol{\mu}_t), \boldsymbol{\mu} \rangle + \frac{1}{\rho_t} \mathbb{D}_{A^*}(\boldsymbol{\mu}\|\boldsymbol{\mu}_t). \tag{22}$$

The proof is straight forward. From the definition for the Bregman divergence, we find that

$$\nabla_{\boldsymbol{\mu}} \mathbb{D}_{A^*}(\boldsymbol{\mu}\|\boldsymbol{\mu}_t) = \boldsymbol{\lambda} - \boldsymbol{\lambda}_t. \tag{23}$$

Using this, if we take the gradient Eq. 22 with respect to $\boldsymbol{\mu}$ and set it to zero, we get the BLR update.

The reverse statement also holds: a mirror descent update in the space Ω leads to a natural gradient descent in the space of $\mathcal{M}$. From a Bayesian viewpoint, Eq. 22 is closer to Bayes' rule since an addition in the natural-parameter space Ω is equivalent to multiplication in the space of $\mathcal{Q}$ (see Sec. 5.1). Natural parameters additionally encode conditional independence (e.g., for Gaussians) which is preferable from a computational viewpoint (see Malagò and Pistone [2015] for a similar suggestion). In any case, a reverse version of BLR can always be used if the need arises.

3 Optimization Algorithms from the Bayesian Learning Rule

3.1 First- and second-order methods by using Gaussian candidates

The examples in Sec. 1.3 demonstrate the key ideas behind our recipe to derive learning algorithms:

1. Natural-gradients retrieve essential higher-order information about the loss landscape and assign them to the appropriate natural parameters,

2. These choices are both dictated by the entropy of the chosen class $\mathcal{Q}$, which automatically determines the complexity of the derived algorithms.

These are concisely summarized in the optimality condition of the Bayes objective (derived in Eq. 5).

$$\nabla_{\boldsymbol{\mu}} \mathbb{E}_{q_*}[\bar{\ell}(\boldsymbol{\theta})] = \nabla_{\boldsymbol{\mu}} \mathcal{H}(q_*) \tag{24}$$

From this, we can derive as a special case the optimality condition of a non-Bayesian problem over $\bar{\ell}(\boldsymbol{\theta})$, as in Eq. 1. For instance, for Gaussians $\mathcal{N}(\boldsymbol{\theta}|\boldsymbol{m}, \boldsymbol{I})$, the entropy is constant. Therefore the right hand side in Eq. 24 goes to zero, giving the condition shown on the left below,

$$\nabla_{\mathbf{m}} \mathbb{E}_{q_*}\left[\bar{\ell}(\boldsymbol{\theta})\right]\big|_{\boldsymbol{m}=\boldsymbol{m}_*} = 0 \xrightarrow{\text{Bonnet's thm.}} \mathbb{E}_{q_*}\left[\nabla_{\boldsymbol{\theta}} \bar{\ell}(\boldsymbol{\theta})\right] = 0 \xrightarrow{\text{delta method}} \nabla_{\boldsymbol{\theta}} \bar{\ell}(\boldsymbol{\theta})\big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_*} = 0. \tag{25}$$

The condition can also be derived from a fixed point of the BLR (in Eq. 7). The simplifications shown at the right above are respectively obtained by using Bonnet's theorem (App. D) and the delta method (App. C) to finally recover the 1st-order optimality condition at a local minimizer $\boldsymbol{\theta}_*$ of $\bar{\ell}(\boldsymbol{\theta})$. Clearly,

the above natural gradient contain the information about the first-order derivative of the loss, to give us an equation that determines the value of the natural parameter $\boldsymbol{m}_*$.

When the complexity of the Gaussian is increased to include the covariance parameters with candidates $\mathcal{N}(\boldsymbol{\theta}|\boldsymbol{m}, \boldsymbol{S}^{-1})$, the entropy is no more a constant but depends on $\boldsymbol{S}$, $\mathcal{H}(q) = -\frac{1}{2}\log|2\pi e \boldsymbol{S}|$. The fixed-point of the BLR (in Eq. 12 now) to yield an additional condition, shown on the left,

$$\nabla_{\boldsymbol{S}^{-1}}\mathbb{E}_{q_*}\left[\bar{\ell}(\boldsymbol{\theta})\right]\big|_{\boldsymbol{S}^{-1}=\boldsymbol{S}_*^{-1}} = \frac{1}{2}\boldsymbol{S}_* \xrightarrow{\text{Price's thm.}} \mathbb{E}_{q_*}\left[\nabla_{\boldsymbol{\theta}}^2\bar{\ell}(\boldsymbol{\theta})\right] = \boldsymbol{S}_* \xrightarrow{\text{delta method}} \nabla_{\boldsymbol{\theta}}^2\bar{\ell}(\boldsymbol{\theta})\big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_*} \succ 0. \quad (26)$$

The simplifications at the right are respectively obtained by using Price's theorem (App. D) and the delta method (App. C) to recover the 2nd-order optimality condition of a local minimizer $\boldsymbol{\theta}_*$ of $\bar{\ell}(\boldsymbol{\theta})$, Here, $\succ 0$ denotes the positive-definite condition which follows from $\boldsymbol{S}_* \succ 0$. The condition above matches the second-order derivatives to the precision matrix $\boldsymbol{S}_*$. In general, more complex sufficient statistics in $q(\boldsymbol{\theta})$ can enable us to retrieve higher-order derivatives of the loss through natural gradients.

The above is a simple method to obtain non-Bayesian solutions $\boldsymbol{\theta}_*$ while optimizing for a Bayesian objective. Often such point estimates are justified as Dirac's-delta posterior approximations, but this is fundamentally flawed because the entropy of such distribution goes to $-\infty$, making the Bayesian objective meaningless; see Welling et al. [2008]. Including such degenerate distributions requires non-trivial modifications to the class of exponential family, for example, see Malagò et al. [2011]. Our use of the delta method is simpler. It is related to Laplace's method (more detail in Sec. 4.4), and also to the online-learning methods where the posterior mean is used for decision making [Hoeven et al., 2018].

3.2 Multimodal optimization by using mixtures-candidate distributions

It is natural to expect that by increasing the complexity of the class $\mathcal{Q}$, we can obtain algorithms that go beyond standard Newton-like optimizers. We illustrate this point by using mixture distribution to obtain a new algorithm for multimodal optimization [Yu and Gen, 2010] where the goal is to simultaneously locate multiple local minima within a single run of the algorithm Wong [2015]. Mixture distributions are ideal for this task where individual components can be tasked to locate different local minima, and diversity among them is encouraged by the entropy, forcing them to take responsibility of different regions. Throughout this section, we should keep in mind that our goal is to simply illustrate the principle, rather than to propose new algorithms that solve multimodal optimization in its entirety.

We will rely on the work of Lin et al. [2019b,a] who derive a natural-gradient update, similar to Eq. 3, but for mixtures. Consider, for example, the finite mixture of Gaussians

$$q(\boldsymbol{\theta}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{m}_k, \boldsymbol{S}_k^{-1})$$

with $\boldsymbol{m}_k$ and $\boldsymbol{S}_k$ as the mean and precision of the k'th Gaussian component and π_k are the component probabilities with $\pi_K = 1 - \sum_{k=1}^{K-1}\pi_k$. In general, the FIM of such mixtures could be singular, making it difficult to use natural-gradient updates. However, the joint $q(\boldsymbol{\theta}, z = k) = \pi_k \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{m}_k, \boldsymbol{S}_k^{-1})$, where $z \in \{1, 2, \ldots, K\}$ a mixture-component indicator, has a well-defined FIM which can be used instead. We will now briefly describe this for the mixture of Gaussian case with fixed π_k.

We start by writing the natural and expectation parameters of the joint $q(\boldsymbol{\theta}, z = k)$, denoted by $\boldsymbol{\lambda}_k$ and $\boldsymbol{\mu}_k$ respectively. Since π_k are fixed, these take very similar form to a single Gaussian case (Eq. 9),

$$\begin{aligned} \boldsymbol{\lambda}_k^{(1)} &= \boldsymbol{S}_k\boldsymbol{m}_k, & \boldsymbol{\mu}_k^{(1)} &= \mathbb{E}_q\left[\mathbb{1}_{[z=k]}\boldsymbol{\theta}\right] = \pi_k\boldsymbol{m}_k, \\ \boldsymbol{\lambda}_k^{(2)} &= -\tfrac{1}{2}\boldsymbol{S}_k, & \boldsymbol{\mu}_k^{(2)} &= \mathbb{E}_q\left[\mathbb{1}_{[z=k]}\boldsymbol{\theta}\boldsymbol{\theta}^T\right] = \pi_k\left(\boldsymbol{S}_k^{-1} + \boldsymbol{m}_k\boldsymbol{m}_k^T\right), \end{aligned} \quad (27)$$

9

where $\boldsymbol{\lambda}_k = \{\boldsymbol{\lambda}_k^{(1)}, \boldsymbol{\lambda}_k^{(2)}\}$ and $\boldsymbol{\mu}_k = \{\boldsymbol{\mu}_k^{(1)}, \boldsymbol{\mu}_k^{(2)}\}$. With this, we can write a natural-gradient update for $\boldsymbol{\lambda}_k, k = 1, \ldots, K$, by using the gradient with respect to $\boldsymbol{\mu}_k$ [Lin et al., 2019b, Theorem 3],

$$\boldsymbol{\lambda}_{k,t+1} \leftarrow \boldsymbol{\lambda}_{k,t} - \rho_t \nabla_{\boldsymbol{\mu}_k} \left[\mathbb{E}_{q_t} \left(\bar{\ell}(\boldsymbol{\theta}) \right) - \mathcal{H}(q_t) \right], \tag{28}$$

where $\boldsymbol{\lambda}_{k,t}$ denotes the natural parameter $\boldsymbol{\lambda}$ at iteration t and $\mathcal{H}(q) = \mathbb{E}_q[-\log q(\boldsymbol{\theta})]$ is the entropy of the mixture $q(\boldsymbol{\theta})$ (not the joint $q(\boldsymbol{\theta}, z)$). This is an extension of the BLR (Eq. 3) to finite mixtures.

As before, the natural gradients retrieve first and second order information. Specifically, as shown in App. E, update for each k takes a Newton-like form,

$$\boldsymbol{m}_{k,t+1} \leftarrow \boldsymbol{m}_{k,t} - \rho \boldsymbol{S}_{k,t+1}^{-1} \mathbb{E}_{q_{k,t}} \left[\nabla_{\boldsymbol{\theta}} \bar{\ell}(\boldsymbol{\theta}) + \nabla_{\boldsymbol{\theta}} \log q(\boldsymbol{\theta}) \right], \tag{29}$$

$$\boldsymbol{S}_{k,t+1} \leftarrow \boldsymbol{S}_{k,t} + \mathbb{E}_{q_{k,t}} \left[\nabla_{\boldsymbol{\theta}}^2 \bar{\ell}(\boldsymbol{\theta}) + \nabla_{\boldsymbol{\theta}}^2 \log q(\boldsymbol{\theta}) \right], \tag{30}$$

where $q_{k,t}(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{m}_{k,t}, \boldsymbol{S}_{k,t}^{-1})$ is the k'th component at iteration t. The mean and precision of a component are updated using the expectations of the gradient and Hessian respectively. The expectation helps to focus on a region where the component has a high probability mass. Similarly to the Newton step of Eq. 12, the first update is preconditioned with the covariance.

The update can locate multiple solutions in a single run, when each component takes the responsibility of an individual minima. For example, for a problem with two local minima $\boldsymbol{\theta}_{1,*}$ and $\boldsymbol{\theta}_{2,*}$, the best candidate with two components $\mathcal{N}(\boldsymbol{\theta}|\boldsymbol{m}_{1,*}, \boldsymbol{S}_{1,*}^{-1})$ and $\mathcal{N}(\boldsymbol{\theta}|\boldsymbol{m}_{2,*}, \boldsymbol{S}_{2,*}^{-1})$ satisfies optimality condition,

$$\mathbb{E}_{\mathcal{N}(\boldsymbol{\theta}|\boldsymbol{m}_{k,*}, \boldsymbol{S}_{k,*}^{-1})} \left[\nabla_{\boldsymbol{\theta}} \bar{\ell}(\boldsymbol{\theta}) + \underbrace{r(\boldsymbol{\theta}) \boldsymbol{S}_{1,*}(\boldsymbol{m}_{1,*} - \boldsymbol{\theta}) + (1 - r(\boldsymbol{\theta})) \boldsymbol{S}_{2,*}(\boldsymbol{m}_{2,*} - \boldsymbol{\theta})}_{\nabla_{\boldsymbol{\theta}} \log q(\boldsymbol{\theta})} \right] = 0, \tag{31}$$

for $k = 1, 2$, where we use expression for $\nabla_{\boldsymbol{\theta}} \log q(\boldsymbol{\theta})$ from Eq. 69 in App. E, and $r(\boldsymbol{\theta})$ is the responsibility function defined as

$$r(\boldsymbol{\theta}) = \frac{\pi \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{m}_{1,*}, \boldsymbol{S}_{1,*}^{-1})}{\pi \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{m}_{1,*}, \boldsymbol{S}_{1,*}^{-1}) + (1 - \pi) \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{m}_{2,*}, \boldsymbol{S}_{2,*}^{-1})}. \tag{32}$$

Suppose that each component takes responsibility of one local minima each, for example, for $\boldsymbol{\theta}$ sampled from the first component, $r(\boldsymbol{\theta}) \approx 1$, and for those sampled from the second component, $r(\boldsymbol{\theta}) \approx 0$. Under this condition, both the second and third term in Eq. 31 are negligible, and the mean $\boldsymbol{m}_{k,*}$ approximately satisfies the first-order optimality condition to qualify as a minimizer of $\bar{\ell}(\boldsymbol{\theta})$,

$$\mathbb{E}_{\mathcal{N}(\boldsymbol{\theta}|\boldsymbol{m}_{k,*}, \boldsymbol{S}_{k,*}^{-1})}[\nabla_{\boldsymbol{\theta}} \bar{\ell}(\boldsymbol{\theta})] \approx 0 \xrightarrow{\text{delta method}} \nabla_{\boldsymbol{\theta}} \bar{\ell}(\boldsymbol{\theta})\big|_{\boldsymbol{\theta}=\boldsymbol{m}_{k,*}} \approx 0, \quad \text{for } k = 1, 2 \tag{33}$$

This roughly implies that $\boldsymbol{m}_{k,*} \approx \boldsymbol{\theta}_{k,*}$, meaning the two components have located the two local minima.

The example illustrates the role of entropy in encouraging diversity through the responsibility function. There remain several practical difficulties to overcome before we are ensured a good algorithmic outcome, like setting the correct number of components, appropriate initialization, and of course degenerate solutions where two components collapse to be the same. Other mixture distributions can also be used as shown in Lin et al. [2019b] who consider the following: finite mixtures of minimal exponential-families, scale mixture of Gaussians [Andrews and Mallows, 1974, Eltoft et al., 2006], Birnbaum-Saunders distribution [Birnbaum and Saunders, 1969], multivariate skew-Gaussian distribution [Azzalini, 2005], multivariate exponentially-modified Gaussian distribution [Grushka, 1972], normal inverse-Gaussian distribution [Barndorff-Nielsen, 1997], and matrix-variate Gaussian distribution [Gupta and Nagar, 2018, Louizos and Welling, 2016].

4 Deep-Learning Algorithms from the Bayesian Learning Rule

4.1 Stochastic gradient descent

Stochastic gradient descent (SGD) is one of the most popular deep-learning algorithms due to its computational efficiency. The computational speedup is obtained by approximating the gradient of the loss by a stochastic-gradient built using a small subset $\mathcal{M}$ of M examples with $M \ll N$. The subset $\mathcal{M}$, commonly referred to as a mini-batch, is randomly drawn at each iteration from the full dataset using a uniform distribution, and the resulting mini-batch gradient

$$\widehat{\nabla}_{\boldsymbol{\theta}}\bar{\ell}(\boldsymbol{\theta}) = \frac{N}{M}\sum_{i \in \mathcal{M}}\nabla_{\boldsymbol{\theta}}\ell(y_i, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)) + \nabla_{\boldsymbol{\theta}}R(\boldsymbol{\theta}), \tag{34}$$

is much faster to compute. Similarly to the gradient-descent case in Sec. 1.3.1, the SGD step can be interpreted as the BLR over a distribution $q(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{m}, \boldsymbol{I})$ with the unknown mean $\boldsymbol{m}$ and the gradient in Eq. 7 being replaced by the mini-batch gradient. The BLR update is extended to candidates $q(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{m}, \boldsymbol{S}^{-1})$ with any fixed $\boldsymbol{S} \succ \boldsymbol{0}$ to get an SGD-like update where $\boldsymbol{S}^{-1}$ is the preconditioner,

$$\boldsymbol{m}_{t+1} \leftarrow \boldsymbol{m}_t - \rho_t \boldsymbol{S}^{-1} \left.\widehat{\nabla}_{\boldsymbol{\theta}}\bar{\ell}(\boldsymbol{\theta})\right|_{\boldsymbol{\theta}=\boldsymbol{m}_t}. \tag{35}$$

This is derived similarly to Sec. 1.3.1 but with $\boldsymbol{\lambda} = \boldsymbol{Sm}$, $\boldsymbol{\mu} = \boldsymbol{m}$, and $2 \log h(\boldsymbol{\theta}) = P \log |2\pi \boldsymbol{S}| - \boldsymbol{\theta}^T \boldsymbol{S}\boldsymbol{\theta}$.

The above BLR interpretation sheds light on the ineffectiveness of first-order methods (like SGD) to estimate posterior approximations, This is a recent trend in Bayesian deep learning, where a preconditioned update is preferred due to the an-isotropic SGD noise [Mandt et al., 2017, Chaudhari and Soatto, 2018, Maddox et al., 2019]. Unfortunately, a first-order optimizer (such as Eq. 35) offers no clue about estimating $\boldsymbol{S}$. SGD iterates can be used [Mandt et al., 2017, Maddox et al., 2019], but iterates and the noise in them are affected by the choice of $\boldsymbol{S}$, which makes the estimation difficult. With our Bayesian scheme, the preconditioner is estimated automatically using a second-order method (Eq. 12) when we allow $\boldsymbol{S}$ as a free parameter. We discuss this next in the context of adaptive learning-rate algorithms.

4.2 Adaptive learning-rate algorithms

Adaptive learning-rate algorithms, motivated from Newton's method, adapt the learning rate in SGD with a scaling vector, and we discuss their relationship to the BLR. Early adaptive variants relied on the diagonal of the mini-batch Hessian matrix to reduce the computation cost [Barto and Sutton, 1981, Becker and LeCun, 1988], and use exponential smoothing to improve stability [LeCun et al., 1998]:

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \alpha \frac{1}{\boldsymbol{s}_{t+1}} \circ \widehat{\nabla}_{\boldsymbol{\theta}}\bar{\ell}(\boldsymbol{\theta}_t), \quad \text{and} \quad \boldsymbol{s}_{t+1} \leftarrow (1-\beta)\boldsymbol{s}_t + \beta \, \text{diag}[\widehat{\nabla}_{\boldsymbol{\theta}}^2\bar{\ell}(\boldsymbol{\theta}_t)], \tag{36}$$

where $\boldsymbol{a} \circ \boldsymbol{b}$ and $\boldsymbol{a}/\boldsymbol{b}$ denote element-wise multiplication and division respectively, and the scaling vector, denoted by $\boldsymbol{s}_t$, uses $\text{diag}[\widehat{\nabla}_{\boldsymbol{\theta}}^2\bar{\ell}(\boldsymbol{\theta}_t)]$, which denotes a vector whose j'th entry is the mini-batch Hessian

$$\widehat{\nabla}_{\theta_j}^2\bar{\ell}(\boldsymbol{\theta}) = \frac{N}{M}\sum_{i \in \mathcal{M}}|\nabla_{\theta_j}^2\ell(y_i, \boldsymbol{f}_{\theta}(\boldsymbol{x}_i))| + \nabla_{\theta_j}^2 R(\boldsymbol{\theta}).$$

Note the use of absolute value to make sure that the entries of $\boldsymbol{s}_t$ stay positive (assuming $R(\boldsymbol{\theta})$ to be strongly-convex). The exponential smoothing, a popular tool from the non-stationary time-series literature [Brown, 1959, Holt et al., 1960, Gardner Jr, 1985], works extremely well for deep learning and

is adopted by many adaptive learning-rate algorithms, e.g., the natural Newton method of Le Roux and Fitzgibbon [2010] and the vSGD method by Schaul et al. [2013] both use it to estimate the second-order information, while more modern variants such as RMSprop [Tieleman and Hinton, 2012], AdaDelta [Zeiler, 2012], and Adam [Kingma and Ba, 2015], use it to estimate the magnitude of the gradient (a more crude approximation of the second-order information [Bottou et al., 2016]).

The BLR naturally gives rise to the update Eq. 36, where exponential smoothing arises as a *consequence* and not something we need to invent. We optimize over candidates of the form $q(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{m}, \boldsymbol{S}^{-1})$ where $\boldsymbol{S}$ is constrained to be a diagonal matrix with a vector $\boldsymbol{s}$ as the diagonal. This choice results in the BLR update similar to the Newton-like update Eq. 12,

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \rho_t \frac{1}{\boldsymbol{s}_{t+1}} \circ \nabla_{\boldsymbol{\theta}} \bar{\ell}(\boldsymbol{\theta}_t), \quad \text{where} \quad \boldsymbol{s}_{t+1} \leftarrow (1 - \rho_t)\boldsymbol{s}_t + \rho_t \, \text{diag}(\nabla_{\boldsymbol{\theta}}^2 \bar{\ell}(\boldsymbol{\theta}_t)), \tag{37}$$

to obtain $\boldsymbol{m}_t = \boldsymbol{\theta}_t$ and $\text{diag}(\boldsymbol{S}_t) = \boldsymbol{s}_t$. Replacing the gradient and Hessian by their mini-batch approximations and then employing different learning-rates for $\boldsymbol{\theta}_t$ and $\boldsymbol{s}_t$, we recover the update Eq. 36.

The similarity of the BLR update can be exploited to compute uncertainty estimates for deep-learning models. Khan et al. [2018] study the relationship between Eq. 37 and modern adaptive learning-rate algorithms, such as RMSprop [Tieleman and Hinton, 2012], AdaDelta [Zeiler, 2012], and Adam [Kingma and Ba, 2015]. These modern variants also use exponential smoothing but over gradient-magnitudes $\widehat{\nabla}_{\boldsymbol{\theta}} \bar{\ell}(\boldsymbol{\theta}_t) \circ \widehat{\nabla}_{\boldsymbol{\theta}} \bar{\ell}(\boldsymbol{\theta}_t)$, instead of the Hessian, e.g., RMSprop uses the following update,

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \alpha \frac{1}{\sqrt{\boldsymbol{v}_{t+1}} + c\mathbf{1}} \circ \left[\widehat{\nabla}_{\boldsymbol{\theta}} \bar{\ell}(\boldsymbol{\theta}_t)\right], \quad \text{where} \quad \boldsymbol{v}_{t+1} \leftarrow (1 - \beta)\boldsymbol{v}_t + \beta \left[\widehat{\nabla}_{\boldsymbol{\theta}} \bar{\ell}(\boldsymbol{\theta}_t) \circ \widehat{\nabla}_{\boldsymbol{\theta}} \bar{\ell}(\boldsymbol{\theta}_t)\right]. \tag{38}$$

There are also some other minor differences to Eq. 37, for example, the scaling uses a square-root to take care of the factor $1/M^2$ in the square of the mini-batch gradient and a small $c > 0$ is added to avoid from (near)-zero $\boldsymbol{v}_{t+1}$ [Khan et al., 2018, Sec. 3.4]. Despite these small differences, the similarly of the two updates in Eqs. 37 and 38 enables us to incorporate Bayesian principles in deep learning. Uncertainty is then computed using a slightly-modified RMSprop update [Osawa et al., 2019] which, which just a slight changes in the code, can exploit all the tricks-of-the-trade of deep learning such as batch normalization, data augmentation, clever initialization, and distributed training (also see Sec. 4.4). The BLR update extends the application of adaptive learning-rate algorithms to uncertainty estimation in deep learning.

The Adam optimizer [Kingma and Ba, 2015] is inarguably one of the most popular optimizer, and this too is closely related to a BLR variant which uses a momentum based mirror-descent. Momentum is a common technique to improve convergence speed of deep learning [Sutskever et al., 2013]. The classical momentum method is based on the Polyak's heavy-ball method [Polyak, 1964],

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \alpha \widehat{\nabla}_{\boldsymbol{\theta}} \bar{\ell}(\boldsymbol{\theta}_t) + \gamma_t (\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t-1})$$

with a fixed momentum coefficient $\gamma_t > 0$, but the momentum used in adaptive learning-rate algorithms, such as Adam, differ in the use of scaling with $\boldsymbol{s}_t$ in front of the momentum term $\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t-1}$ (see Eq. 78 in App. F). These variants can be better explained by including a momentum term in the BLR within the mirror descent framework of Sec. 2, which gives the following modification of the BLR (see App. F):

$$\boldsymbol{\lambda}_{t+1} \leftarrow \boldsymbol{\lambda}_t - \rho_t \widetilde{\nabla}_{\boldsymbol{\lambda}} \left[\mathbb{E}_{q_t}\left(\bar{\ell}(\boldsymbol{\theta})\right) - \mathcal{H}(q_t)\right] + \gamma_t (\boldsymbol{\lambda}_t - \boldsymbol{\lambda}_{t-1}).$$

For a Gaussian distribution, this recovers both SGD and Newton's method with momentum. Variants like Adam can then be obtained by making approximations to the Hessian and by using a square-root scaling, see Eq. 81 in App. F.

Finally, we will also mention Aitchison [2018] who derive deep-learning optimizers using a Bayesian filtering approach with updating equations similar to the ones we presented. They further discuss modifications to justify the use of square-root in Adam and RMSprop. The approach taken by Ollivier [2018] is similar, but exploits the connection of Kalman filtering and the natural-gradient method.

4.3 Dropout

Dropout is a popular regularization technique to avoid overfitting in deep learning where hidden units are randomly dropped from the neural network (NN) during training [Srivastava et al., 2014]. Gal and Ghahramani [2016] interpret SGD-with-dropout as a Bayesian approximation (called MC-dropout), but their crude approximation to the entropy term prohibits extensions to more flexible posterior approximations. In this section, we show that by using the BLR we can improve their approximation and go beyond SGD to derive MC-dropout variants of Newton's method, Adam and RMSprop.

Denoting the j'th unit in the l'th layer by $f_{jl}(\boldsymbol{x})$ for an input $\boldsymbol{x}$, we can write the i'th unit for the $l+1$'th layer as follows:

$$f_{i,l+1}(\boldsymbol{x}) = h\left(\sum_{j=1}^{n_l} \theta_{ijl} z_{jl} f_{jl}(\boldsymbol{x})\right) \tag{39}$$

where the binary weights $z_{jl} \in \{0, 1\}$ are to be defined, n_l is the number of units in the l'th layer, all $\theta_{ijl} \in \mathbb{R}$ are the parameters, and $h(\cdot)$ is a nonlinear activation function. For simplicity, we have not included an offset term.

Without dropout, all weights z_{jl} are all set to be one. With dropout, all z_{jl}'s are independent Bernoulli variables with probability for being 1 equal to π_1 (and fixed). If $z_{jl} = 0$, then the j'th unit of the l'th is dropped from the NN, otherwise it is retained. Let $\tilde{\theta}_{ijl} = \theta_{ijl} z_{jl}$ and let $\boldsymbol{\theta}$ and $\tilde{\boldsymbol{\theta}}$ denote the complete set of parameters. The training can be performed with the following simple modification of SGD where $\tilde{\boldsymbol{\theta}}$ are used during back-propagation,

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \alpha \widehat{\nabla}_{\boldsymbol{\theta}} \bar{\ell}(\tilde{\boldsymbol{\theta}}_t). \tag{40}$$

This simple procedure has proved to be very effective in practice and dropout has become a standard trick-of-the-trade to improve performance of large deep networks trained with SGD.

We can include dropout into the Bayesian learning rule, by considering a spike-and-slab mixture distribution for $\boldsymbol{\theta}_{jl} = (\theta_{1jl}, \ldots, \theta_{n_l jl})$

$$q(\boldsymbol{\theta}_{jl}) = \pi_1 \mathcal{N}(\boldsymbol{\theta}_{jl}|\boldsymbol{m}_{jl}, \boldsymbol{S}_{jl}^{-1}) + (1 - \pi_1)\mathcal{N}(\boldsymbol{\theta}_{jl}|\boldsymbol{0}, s_0^{-1}\boldsymbol{I}_{n_l}) \tag{41}$$

where the means and covariances are unknown, but s_0 is fixed to a small positive value to emulate a spike at zero. Further, we assume independence for every j and l, $q(\boldsymbol{\theta}) = \prod_{jl} q(\boldsymbol{\theta}_{jl})$. For this choice of candidate distribution, we arrive in the end at the following update,

$$\boldsymbol{\theta}_{jl,t+1} \leftarrow \boldsymbol{\theta}_{jl,t} - \rho_t \boldsymbol{S}_{jl,t+1}^{-1}\left[\widehat{\nabla}_{\boldsymbol{\theta}_{jl}} \bar{\ell}(\tilde{\boldsymbol{\theta}}_t)\right]/\pi_1, \tag{42}$$

$$\boldsymbol{S}_{jl,t+1} \leftarrow (1 - \rho_t)\boldsymbol{S}_{jl,t} + \rho_t\left[\widehat{\nabla}_{\boldsymbol{\theta}_{jl}}^2 \bar{\ell}(\tilde{\boldsymbol{\theta}}_t)\right]/\pi_1. \tag{43}$$

The derivation in App. G uses two approximations. First, the delta method (Eq. 84, also used by Gal and Ghahramani [2016]), and second, an approximation to the responsibility function (Eq. 32) which assumes that $\boldsymbol{\theta}_{jl}$ is far apart from 0 (can be ensured by choosing s_0 to be very small; see Eqs. 85 and 86).

The update is a variant of the Newton's method derived in Eq. 12. The difference is that the gradients and Hessians are evaluated at the dropout weights $\tilde{\boldsymbol{\theta}}_t$. By choosing $\boldsymbol{S}_{jl,t}$ to be a diagonal matrix as in Sec. 4.2, we can similarly derive RMSprop/Adam variants with dropout. The BLR derivation enables more flexible posterior approximations than the original derivation by Gal and Ghahramani [2016].

We can build on the mixture candidate model to learn also π_1 and/or allow it to be different across layers, although by doing so we leave the justification leading to dropout. Such extensions can

be obtained using the extension to mixture distribution by Lin et al. [2019b]. The update requires gradients with respect to π_1 which can be approximated by using the Concrete distribution [Maddison et al., 2016, Jang et al., 2016] where the binary z_{jl} is replaced with an appropriate variable in the unit interval, see also Sec. 4.5 for a similar approach. Gal et al. [2017] proposed a similar extension but their approach do not allow for more flexible candidate distributions. Our approach here extends to adaptive learning-rate optimizers, and can be extended to more complex candidate distributions.

4.4 Uncertainty estimation in deep learning

Uncertainty estimation in deep learning is crucial for many applications, such as medical diagnosis and self-driving cars, but its computation is extremely challenging due to a large number of parameters and data examples. A straightforward application of Bayesian methods does not usually work. Standard Bayesian approaches, such as Markov Chain Monte Carlo (MCMC), Stochastic Gradient Langevin Dynamics (SGLD), and Hamiltonian Monte-Carlo (HMC), are either infeasible or slow [Balan et al., 2015]. Even approximate inference approaches, such as Laplace's method [Mackay, 1991, Ritter et al., 2018], variational inference [Graves, 2011, Blundell et al., 2015], and expectation propagation [Hernandez-Lobato and Adams, 2015], do not match the performance of the standard point-estimation methods at large scale. On the ImageNet dataset, for example, these Bayesian methods performed poorly until recently when Osawa et al. [2019] applied the natural-gradient algorithms [Khan et al., 2018, Zhang et al., 2018b]. Such natural-gradient algorithms are in fact variants of the BLR and we will now discuss their derivation, application, and suitability for uncertainty estimation in deep learning.

We start with the Laplace's method which is one of the simplest approaches to obtain uncertainty estimates from point estimates; see Mackay [1991] for an early application to NN. Given a minimizer $\boldsymbol{\theta}_*$ of ERM in Eq. 1 (assume that the loss correspond to a negative log probability over outputs y_i), in Laplace's method, we employ a Gaussian approximation $\mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\theta}_*, \boldsymbol{S}_*^{-1})$ where $\boldsymbol{S}_* = \nabla_{\boldsymbol{\theta}}^2 \bar{\ell}(\boldsymbol{\theta}_*)$. For models with billions of parameters, even forming $\boldsymbol{S}_*$ is infeasible and a diagonal approximation is often employed [Ritter et al., 2018]. Still, Hessian computation after training requires a full pass through the data which could be expensive. The BLR updates solve this issue since Hessian can be obtained during training, e.g., using Eq. 37 with a minibach gradient and Hessian (indicated by $\widehat{\nabla}$ below),

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \rho_t \frac{1}{\boldsymbol{s}_{t+1}} \circ \widehat{\nabla}_{\boldsymbol{\theta}} \bar{\ell}(\boldsymbol{\theta}_t), \quad \text{with} \quad \boldsymbol{s}_{t+1} \leftarrow (1 - \rho_t)\boldsymbol{s}_t + \rho_t \, \text{diag}(\widehat{\nabla}_{\boldsymbol{\theta}}^2 \bar{\ell}(\boldsymbol{\theta}_t)). \tag{44}$$

where it is clear that $\boldsymbol{s}_t$ converges to an unbiased estimate of the diagonal of the Hessian at $\boldsymbol{\theta}_*$.

Using (diagonal) Hessian is still problematic in deep learning because sometimes it can be negative, and the steps can diverge. An additional issue is that its computation is cumbersome in existing software which mainly support first-order derivatives only. Due to this, it is tempting to simply use the RMSprop/Adam update Eq. 38, but using $(\widehat{\nabla}_{\theta_j} \bar{\ell}(\boldsymbol{\theta}_t))^2$ as Hessian approximation results in poor performance as shown by Khan et al. [2018, Thm. 1]. Instead, they propose to use the following Gauss-Newton approximation which is built from first-order derivatives but is more accurate,

$$\widehat{\nabla}_{\theta_j}^2 \bar{\ell}(\boldsymbol{\theta}) \approx \frac{N}{M} \sum_{i \in \mathcal{M}} \left[\nabla_{\theta_j} \ell(y_i, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)) \right]^2 + \nabla_{\theta_j}^2 R(\boldsymbol{\theta}). \tag{45}$$

The resulting algorithm is referred to as the online Gauss-Newton (OGN) in [Osawa et al., 2019, App. C].

The update Eq. 44 is similar to RMSprop, and can be implemented with minimal changes to the existing software and many practical deep-learning tricks can also be applied to boost the performance. Osawa et al. [2019] use OGN with batch normalization, data augmentation, learning-rate scheduling, momentum, initialization tricks, and distributed training. With these tricks, OGN gives similar performance as the Adam optimizer, even at large scale [Osawa et al., 2019], while yielding an estimate of the
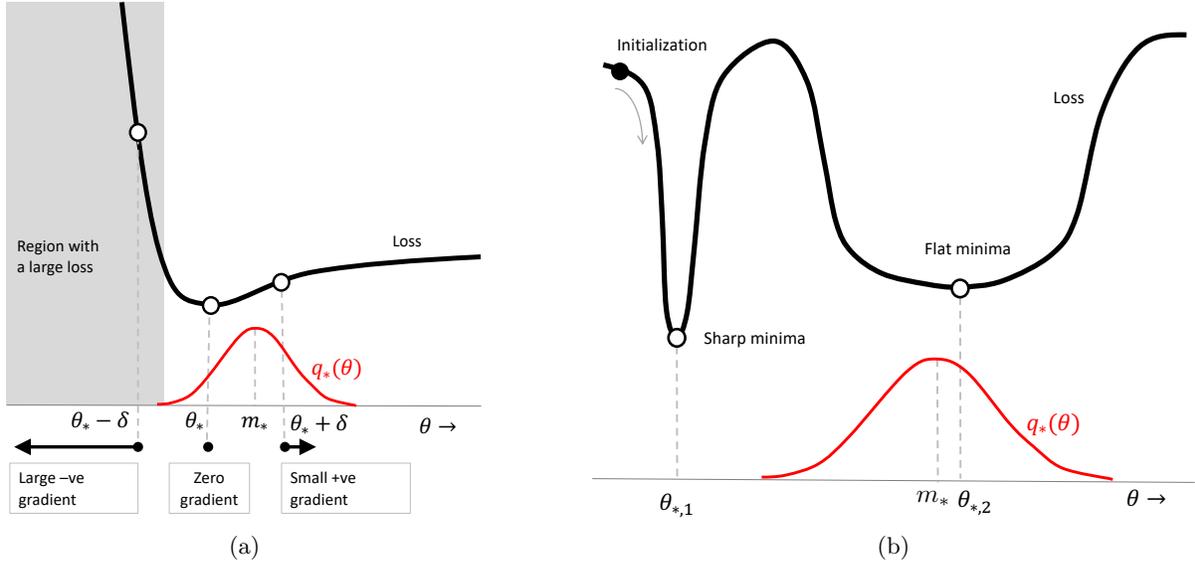
Figure 1: Two examples to illustrate the difference between the solution of the ERM objective (Eq. 1) vs those of the Bayes objective (Eq. 2). Panel (a): When the minima lies right next to a 'wall', the Bayesian solution shifts away from the wall to avoid large losses under small perturbation, which is due to the condition Eq. 25. Panel (b): Given a sharp minima vs a flat minima, the Bayesian solution often prefers the flatter minima, which is again due to the averaging property.

variance. The compute cost is only slightly higher, but it is not an increase in the complexity, rather due to a difficulty of implementing Eq. 45 in the existing deep-learning software. Khan et al. [2019] further propose a slightly better (but costlier) approximation based on the FIM the log-likelihood, to get an algorithm they call the online Generalized Gauss-Newton (OGGN) algorithm. Both OGN and OGGN algorithms are scalable BLR variants for Laplace's method in deep learning.

OGN's uncertainty estimates can be improved by using variational-Bayesian inference. This improves over Laplace's method by using the stationarity conditions (Eqs. 25 and 26) with an expectation over $q(\boldsymbol{\theta})$ [Opper and Archambeau, 2009]. The variational solution is slightly different than Laplace's method and is expected to be more robust due to the averaging over $q_*(\boldsymbol{\theta})$ in Eqs. 25 and 26. Two such situations are illustrated in Fig. 1, one corresponding to an asymmetric loss where the Bayesian solution avoids a region with extremely high loss (Fig. 1(a)), and the other one where it seeks a more stable solution involving a wide and shallow minima compared to a sharp and deep minima (Fig. 1(b)). Such situations are hypothesised to exist in deep-learning problems [Hochreiter and Schmidhuber, 1997, 1995, Keskar et al., 2016], where a good performance of stochastic optimization methods is attributed to their ability to find shallow minima [Dziugaite and Roy, 2017]. Similar strategies exist in stochastic search and global optimization literature where a kernel is used to convolve/smooth the objective function, like in Gaussian homotopy continuation method [Mobahi and Fisher III, 2015], optimization by smoothing [Leordeanu and Hebert, 2008], graduated optimization method [Hazan et al., 2016], and evolution strategies [Huning, 1976, Wierstra et al., 2014]. The benefit of the Bayesian approach is that this kernel corresponds to $q(\boldsymbol{\theta})$ which adapts itself from data and is learned through the BLR.

Variational Bayesian version is obtain by simply removing approximation due to the delta method

from Eq. 44, going back to the original Newton variant of Eq. 12 (now with minibatch Hessians),

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \rho_t \frac{1}{\boldsymbol{s}_{t+1}} \circ \mathbb{E}_{q_t}\left[\widehat{\nabla}_{\boldsymbol{\theta}}\bar{\ell}(\boldsymbol{\theta})\right], \quad \text{with} \quad \boldsymbol{s}_{t+1} \leftarrow (1 - \rho_t)\boldsymbol{s}_t + \rho_t \, \mathbb{E}_{q_t}\left[\text{diag}(\widehat{\nabla}_{\boldsymbol{\theta}}^2 \bar{\ell}(\boldsymbol{\theta}))\right], \qquad (46)$$

where iterates $q_t(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\theta}_t, \boldsymbol{S}_t^{-1})$ are defined with precision matrix $\boldsymbol{S}_t$ as a diagonal matrix with $\boldsymbol{s}_t$ as the diagonal. The iterates converge to an optimal diagonal Gaussian candidate that optimizes Eq. 2.

Similarly to OGN, the update Eq. 46 can be modified to use the Gauss-Newton approximation Eq. 45, which is easier to implement and can also exploit the deep-learning tricks to find good solutions. The expectations can be implemented by a simple 'weight-perturbation',

$$\mathbb{E}_{q_t}[\widehat{\nabla}_{\boldsymbol{\theta}}\bar{\ell}(\boldsymbol{\theta})] \approx \widehat{\nabla}_{\boldsymbol{\theta}}\bar{\ell}(\boldsymbol{\theta}_t + \boldsymbol{\epsilon}_t)$$

where $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\boldsymbol{\epsilon}|0, \text{diag}(\boldsymbol{s}_t))$ (multiple samples can also be used). The resulting algorithm is referred to as variational OGN or VOGN by Khan et al. [2018]) and is shown to match the performance of Adam and give better uncertainty estimates than both Adam, OGN, and other Bayesian alternatives [Osawa et al., 2019, Fig. 1]. A version with the FIM of the log-likelihood is proposed in Khan et al. [2019] (algorithm is called VOGGN). Variants that exploit Hessian structures using Kronecker-factored [Zhang et al., 2018b] and low-rank [Mishkin et al., 2018] are also considered. All these are natural-gradients methods, therefore BLR variants which enable uncertainty estimation in deep learning with scalable Gaussian candidates.

4.5 Binary neural networks

Binary NNs have binary weights $\boldsymbol{\theta} \in \{-1, +1\}^P$, hence we need to consider another class of the candidate distributions than the Gaussian (we will choose Bernoulli). Binary NNs are natural candidates for resource-constrained applications, like mobile phones, wearable and IoT devices, but their training is challenging due to the binary weights for which gradients do not exists. Our Bayesian scheme solves this issue because gradients of the Bayesian objective are always defined with respect to the parameters of a Bernoulli distribution, turning a discrete problem into a continuous one. The discussion here is based on the results of Meng et al. [2020] who used the BLR to derive the BayesBiNN learning algorithm and connect it to the well known Straight-Through-Estimator (STE) [Bengio et al., 2013].

The training objective of binary NN involves a discrete optimization problem

$$\min_{\boldsymbol{\theta} \in \{-1, +1\}^P} \sum_{i=1}^{N} \ell(y_i, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)). \qquad (47)$$

In theory, gradient based methods should not work for such problems since the gradients with respect to discrete variables do not exist. Still, the most popular method to train binary NN, the Straight-Through-Estimator (STE) [Bengio et al., 2013], employs continuous optimization methods and works remarkably well [Courbariaux et al., 2015]. The method is justified based on latent real-valued weights $\tilde{\boldsymbol{\theta}} \in \mathbb{R}^P$ which are discretized at every iteration to get binary weights $\boldsymbol{\theta}$, while the gradients used to update the latent weights $\tilde{\boldsymbol{\theta}}$, are computed at the binary weights $\boldsymbol{\theta}$, as shown below,

$$\tilde{\boldsymbol{\theta}}_{t+1} \leftarrow \tilde{\boldsymbol{\theta}}_t - \alpha \sum_{i \in \mathcal{M}_t} \nabla_{\boldsymbol{\theta}}\ell(y_i, f_{\boldsymbol{\theta}_t}(\boldsymbol{x}_i)), \text{ where } \boldsymbol{\theta}_t \leftarrow \text{sign}(\tilde{\boldsymbol{\theta}}_t) \qquad (48)$$

It is not clear why the gradients computed at the binary weights help the search for the minimum of the discrete problem, and many studies have attempted to answer this question, e.g., see Yin et al. [2019], Alizadeh et al. [2019]. The Bayesian objective in Eq. 2 rectify this issue since the expectation

of a discrete objective is a continuous objective with respect to the parameters of the distributions, justifying the use of gradient based methods.

We will now show that using Bernoulli candidate distributions in the BLR gives rise to an algorithm similar Eq. 48, justifying the application of the STE algorithm to solve a discrete optimization problem. Specifically, we choose $q(\theta_j = 1) = p_j$ where $p_j \in [0, 1]$ is the success probability. We also assume a mean-field distribution to get $q(\boldsymbol{\theta}) = \prod_{j=1}^{P} q(\theta_j)$. Since the Bernoulli distribution is a minimal exponential-family with a constant base measure, we use the BLR in Eq. 6 with $\lambda_j = 1/2\log(p_j/(1 - p_j))$ and $\mu_j = 2p_j - 1$, to get

$$\boldsymbol{\lambda}_{t+1} \leftarrow (1 - \rho_t)\boldsymbol{\lambda}_t - \rho_t \nabla_{\boldsymbol{\mu}} \mathbb{E}_{q_t}\left[\sum_{i \in \mathcal{M}_t} \ell(y_i, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i))\right]. \tag{49}$$

The challenge now is to compute the gradient with respect to $\boldsymbol{\mu}$, but it turns out that approximating it with the Concrete distribution [Maddison et al., 2016, Jang et al., 2016] recovers the STE step.

Concrete distribution provides a way to relax a discrete random variable into a continuous one such that it becomes a deterministic function of a continuous random variable. Following Maddison et al. [2016], the continuous random variable, denoted by $\hat{\theta}^{(\tau)} \in (-1, 1)$, are constructed from the binary variable $\theta \in \{-1, 1\}$ by using iid uniform random variables $\epsilon \in (0, 1)$ and using the function $\tanh(\cdot)$,

$$\hat{\theta}^{(\tau)} = \tanh\left(\frac{\lambda + \delta(\epsilon)}{\tau}\right), \quad \text{where } \delta(\epsilon) = \frac{1}{2}\log\left(\frac{\epsilon}{1 - \epsilon}\right), \tag{50}$$

with $\tau > 0$ as the temperature parameter. As $\tau \to 0$, the function approaches the sign function used in STE step (Eq. 48): $\hat{\theta}^{(0)} = \text{sign}(\lambda + \delta(\epsilon))$.

The gradient too can be approximated by the gradients with respect to the relaxed variables

$$\nabla_{\boldsymbol{\mu}} \mathbb{E}_q\left[\ell(y_i, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i))\right] \approx \nabla_{\boldsymbol{\mu}} \ell(y_i, f_{\hat{\boldsymbol{\theta}}^{(\tau)}}(\boldsymbol{x}_i)) = \boldsymbol{s} \circ \left[\nabla_{\hat{\boldsymbol{\theta}}^{(\tau)}} \ell(y_i, f_{\hat{\boldsymbol{\theta}}^{(\tau)}}(\boldsymbol{x}_i))\right], \tag{51}$$

where we used chain rule in the last step and $\boldsymbol{s}$ is a vector of

$$s_j = \nabla_{\mu_j} \theta_j^{(\tau)} = \frac{1}{\tau}\left[\frac{1 - (\hat{\theta}_j^{(\tau)})^2}{1 - \tanh^2(\lambda_j)}\right].$$

By renaming $\boldsymbol{\lambda}$ by $\tilde{\boldsymbol{\theta}}$ and $\hat{\boldsymbol{\theta}}^{(\tau)}$ by $\boldsymbol{\theta}$, the rule in Eq. 49 becomes

$$\tilde{\boldsymbol{\theta}}_{t+1} \leftarrow (1 - \rho_t)\tilde{\boldsymbol{\theta}}_t - \rho_t \boldsymbol{s}_t \circ \left[\sum_{i \in \mathcal{M}_t} \nabla_{\boldsymbol{\theta}} \ell(y_i, f_{\boldsymbol{\theta}_t}(\boldsymbol{x}_i))\right], \quad \text{where } \boldsymbol{\theta}_t \leftarrow \tanh\left(\frac{1}{\tau}\left(\tilde{\boldsymbol{\theta}}_t + \boldsymbol{\delta}(\boldsymbol{\epsilon}_t)\right)\right) \tag{52}$$

with $\boldsymbol{s}_t \leftarrow (1 - \boldsymbol{\theta}_t^2)/(\tau(1 - \tanh^2(\tilde{\boldsymbol{\theta}}_t)))$, and $\boldsymbol{\epsilon}_t$ is a vector of P iid samples from a uniform distribution. Meng et al. [2020] refer to this algorithm as BayesBiNN.

BayesBiNN is similar to STE shown in Eq. 48, but computes natural-gradients (instead of gradients) at the relaxed parameters $\boldsymbol{\theta}_t$ which are obtained by adding noise $\boldsymbol{\delta}(\boldsymbol{\epsilon}_t)$ to the real-valued weights $\tilde{\boldsymbol{\theta}}_t$. The gradients are approximations of the expected loss and are well defined. By setting noise to zero and letting the temperature $\tau \to 0$, the $\tanh(\cdot/\tau)$ becomes the sign-function and we recover the gradients used in STE. This limit is when the randomness is ignored and is similar to the delta method used in earlier sections. The random noise $\boldsymbol{\delta}(\boldsymbol{\epsilon}_t)$ and the non-zero temperature τ enables a softening of the binary weights, allowing for meaningful gradients. BayesBiNN also provides meaningful latent $\tilde{\boldsymbol{\theta}}$ which are now the natural parameters of the Bernoulli distribution.

Unlike STE, the updates employs a exponential smoothing which is a direct consequence of using the entropy term in the Bayesian objective. With discrete weights, optimum of Eq. 47 could be completely unrelated to an STE solution $\boldsymbol{\theta}_*$, for example, the one that yields zero gradients $\sum_i \nabla_{\boldsymbol{\theta}} \ell(y_i, f_{\boldsymbol{\theta}_*}(\boldsymbol{x}_i)) = 0$. In contrast, the BayesBiNN solution minimizes the well-defined Bayes objective of Eq. 2, whose optimum is characterized by the optimality condition Eq. 24 directly relating the optimal $\tilde{\boldsymbol{\theta}}_*$ to the relaxed variables $\boldsymbol{\theta}_*$:

$$\frac{\tilde{\boldsymbol{\theta}}_*}{\boldsymbol{s}_*} \approx \sum_{i=1}^N \nabla_{\boldsymbol{\theta}} \ell(y_i, f_{\boldsymbol{\theta}_*}(\boldsymbol{x}_i)). \tag{53}$$

The BayesBiNN solution $\boldsymbol{\theta}_*$ is in general different from the one from the STE algorithm, but when temperature $\tau \to 0$ we expect the two solutions to be similar whenever $\boldsymbol{s}_* \to \infty$ (as the left-hand-side in the equation above goes to zero). In general, we expect the BayesBiNN solution to have robustness properties similar to the ones discussed for Fig. 1. The exponential smoothing used in BayesBiNN is similar to the update of Helwegen et al. [2019] but their formulation lacks a well-defined objective.

5 Probabilistic Inference Algorithms from the Bayesian Learning Rule

Algorithms for inference in probabilistic graphical models can be derived from the BLR, by setting the loss function to be the log-joint density of the data and unknowns. Unknowns could include both latent variables $\boldsymbol{z}$ and model parameters $\boldsymbol{\theta}$. The class $\mathcal{Q}$ is chosen based on the form of the likelihood and prior. We will first discuss the case the conjugate case, where BLR reduces to Bayes' rule, and then discuss expectation maximization and variational inference (VI) where structural approximations to the posterior are usually employed.

5.1 Conjugate models and Bayes' rule

We start with one of the most popular category of Bayesian models called exponential-family conjugate models. We consider N iid data $\boldsymbol{y}_i$ with likelihoods $p(\boldsymbol{y}_i|\boldsymbol{\theta})$ (no latent variables) and prior $p(\boldsymbol{\theta})$. The loss is the negative of the log-joint density $\bar{\ell}(\boldsymbol{\theta}) = -\log p(\boldsymbol{y}, \boldsymbol{\theta})$ where $\boldsymbol{y} = (\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_N)$. Conjugacy implies the existence of sufficient statistics $\boldsymbol{T}(\boldsymbol{\theta})$ such that

$$p(\boldsymbol{y}_i|\boldsymbol{\theta}) \propto \exp\left(\langle \widetilde{\boldsymbol{\lambda}}_i(\boldsymbol{y}_i), \boldsymbol{T}(\boldsymbol{\theta})\rangle\right), \quad \text{and} \quad p(\boldsymbol{\theta}) \propto \exp\left(\langle \boldsymbol{\lambda}_0, \boldsymbol{T}(\boldsymbol{\theta})\rangle\right),$$

for some $\widetilde{\boldsymbol{\lambda}}_i(\boldsymbol{y}_i)$ (a function of $\boldsymbol{y}_i$), and $\boldsymbol{\lambda}_0$. The $\widetilde{\boldsymbol{\lambda}}_i(\boldsymbol{y}_i)$ is often interpreted as the sufficient statistics of $\boldsymbol{y}_i$, but it can also be seen as the natural parameter of the likelihood with respect to $\boldsymbol{T}(\boldsymbol{\theta})$. The posterior is available in closed-form obtained by a simple addition of the natural parameters.

$$p(\boldsymbol{\theta}|\boldsymbol{y}) \propto \exp\left(\langle \boldsymbol{\lambda}_0 + \sum_i \widetilde{\boldsymbol{\lambda}}_i(\boldsymbol{y}_i), \boldsymbol{T}(\boldsymbol{\theta})\rangle\right).$$

The natural parameter of the above is equal to the BLR solution in Eq. 5, now available in closed-form,

$$\boldsymbol{\lambda}_* = \boldsymbol{\lambda}_0 + \sum_{i=1}^N \nabla_{\boldsymbol{\mu}} \mathbb{E}_{q_*}[\log p(\boldsymbol{y}_i|\boldsymbol{\theta})] \tag{54}$$

The natural gradient automatically yields $\widetilde{\boldsymbol{\lambda}}_i(\boldsymbol{y}_i)$ because $\nabla_{\boldsymbol{\mu}} \mathbb{E}_{q_*}[\log p(\boldsymbol{y}_i|\boldsymbol{\theta})] = \widetilde{\boldsymbol{\lambda}}_i(\boldsymbol{y}_i)$. As an example, we illustrate these quantities for ridge regression (Sec. 1.3.3) where y_i is a real-valued scalar,

$$\widetilde{\boldsymbol{\lambda}}_i^{(1)}(y_i) = y_i \boldsymbol{x}_i, \quad \widetilde{\boldsymbol{\lambda}}_i^{(2)}(y_i) = -\frac{1}{2}\boldsymbol{x}_i \boldsymbol{x}_i^\top, \quad\quad \boldsymbol{\lambda}_0^{(1)} = 0, \quad \boldsymbol{\lambda}_0^{(2)} = -\frac{1}{2}\delta \boldsymbol{I}.$$

Using these in Eq. 54, the rhs is equal to Eq. 14. Using the definition of $\boldsymbol{\lambda}_*$ and solving a system to recover the mean and covariance yields the well-known Bayesian linear regression posterior.

For conjugate exponential-family models, the BLR yields the posterior obtained by Bayes' rule. This includes popular Gaussian linear models [Roweis and Ghahramani, 1999], such as Kalman filter/smoother, probabilistic Principle Components Analysis (PCA), factor analysis, mixture of Gaussians, and hidden Markov models, as well as nonparameteric models, such as Gaussian process (GP) regression [Rasmussen and Williams, 2006]. The natural-parameters are available in closed form. Note that the solution also correspond to a single step of Eq. 6 with $\rho_t = 1$. From the computational point of view, the BLR does not offer any advice to obtain marginal properties in general, and efficient techniques, such as message passing, are required to compute natural gradients.

5.2 Expectation maximization

We will now derive Expectation Maximization (EM) from the BLR. EM is a popular algorithm for parameter estimation with latent variables [Dempster et al., 1977]. Given the joint distribution $p(\boldsymbol{y}, \boldsymbol{z}|\boldsymbol{\theta})$ with latent variables $\boldsymbol{z}$ and parameters $\boldsymbol{\theta}$, the EM algorithm computes the posterior $p(\boldsymbol{z}|\boldsymbol{y}, \boldsymbol{\theta}_*)$ as well as parameter $\boldsymbol{\theta}_*$ that maximizes the marginal likelihood $p(\boldsymbol{y}|\boldsymbol{\theta})$. The algorithm can be seen as coordinate-wise BLR updates with $\rho_t = 1$ to update candidate distribution that factorizes across $\boldsymbol{z}$ and $\boldsymbol{\theta}$,

$$q_t(\boldsymbol{z}, \boldsymbol{\theta}) = q_t(\boldsymbol{z})q_t(\boldsymbol{\theta}) \ \propto \ e^{\langle \boldsymbol{\lambda}_t, \boldsymbol{T}_z(\boldsymbol{z}) \rangle} \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\theta}_t, \boldsymbol{I}), \tag{55}$$

where $\boldsymbol{\theta}_t$ denotes the mean of the Gaussian. The loss function is set to $\bar{\ell}(\boldsymbol{z}, \boldsymbol{\theta}) = -\log p(\boldsymbol{y}, \boldsymbol{z}|\boldsymbol{\theta})$.

To ease the presentation, assume the likelihood, prior, and joint density can be expressed as

$$p(\boldsymbol{y}|\boldsymbol{z}, \boldsymbol{\theta}) \propto \exp\left(\langle \widetilde{\boldsymbol{\lambda}}_1(\boldsymbol{y}, \boldsymbol{\theta}), \boldsymbol{T}_z(\boldsymbol{z}) \rangle \right), \qquad\qquad p(\boldsymbol{z}|\boldsymbol{\theta}) \propto \exp\left(\langle \boldsymbol{\lambda}_0(\boldsymbol{\theta}), \boldsymbol{T}_z(\boldsymbol{z}) \rangle \right),$$

$$p(\boldsymbol{y}, \boldsymbol{z}|\boldsymbol{\theta}) \propto \exp\left(\langle \boldsymbol{T}_{yz}(\boldsymbol{y}, \boldsymbol{z}), \boldsymbol{\theta} \rangle - A_{yz}(\boldsymbol{\theta}) \right), \tag{56}$$

for some functions $\widetilde{\boldsymbol{\lambda}}_1(\cdot, \cdot)$ and $\boldsymbol{\lambda}_0(\cdot)$ (similar to the previous section), and $\boldsymbol{\theta}$ is assumed to be the natural parameter of the joint. The likelihood and prior are expressed in terms of sufficient statistics $\boldsymbol{T}_z(\boldsymbol{z})$ while the joint is with $\boldsymbol{T}_{yz}(\cdot, \cdot)$ [Winn and Bishop, 2005]. The EM algorithm then takes a simple form where we iterate between updating $\boldsymbol{\lambda}$ and $\boldsymbol{\theta}$ [Banerjee et al., 2005],

$$\text{E-step:} \quad \boldsymbol{\lambda}_{t+1} \leftarrow \widetilde{\boldsymbol{\lambda}}_1(\boldsymbol{y}, \boldsymbol{\theta}_t) + \boldsymbol{\lambda}_0(\boldsymbol{\theta}_t), \qquad \text{M-step:} \quad \boldsymbol{\theta}_{t+1} \leftarrow A_{yz}^*\left(\mathbb{E}_{q_{t+1}}[\boldsymbol{T}_{yz}(\boldsymbol{y}, \boldsymbol{z})] \right).$$

Here, $A_{yz}^*(\cdot)$ is the Legendre transform, and $q_{t+1}(\boldsymbol{z}) = p(\boldsymbol{z}|\boldsymbol{y}, \boldsymbol{\theta}_{t+1})$ is the posterior found in the E-step.

The two steps are obtained from the BLR by using the delta method to approximate the expectation with respect to $q_t(\boldsymbol{\theta})$. For the E-step, we assume $q_t(\boldsymbol{\theta})$ fixed, and use Eqs. 5 and 54 and the delta method,

$$\boldsymbol{\lambda}_{t+1} \leftarrow \nabla_{\boldsymbol{\mu}} \mathbb{E}_{q_t(\boldsymbol{z})q_t(\boldsymbol{\theta})}[-\bar{\ell}(\boldsymbol{z}, \boldsymbol{\theta})]\big|_{\boldsymbol{\mu} = \boldsymbol{\mu}_t} = \mathbb{E}_{q_t(\boldsymbol{\theta})}[\widetilde{\boldsymbol{\lambda}}_1(\boldsymbol{y}, \boldsymbol{\theta}) + \boldsymbol{\lambda}_0(\boldsymbol{\theta})] \approx \widetilde{\boldsymbol{\lambda}}_1(\boldsymbol{y}, \boldsymbol{\theta}_t) + \boldsymbol{\lambda}_0(\boldsymbol{\theta}_t)$$

For the M-step, we use the stationarity condition similar to Eq. 25 but now with the delta method with respect to $q(\boldsymbol{\theta})$ (note that $\boldsymbol{\theta}_{t+1}$ is both the expectation and natural parameter),

$$\nabla_{\boldsymbol{\theta}} \mathbb{E}_{q_{t+1}(\boldsymbol{z})}\left[\bar{\ell}(\boldsymbol{z}, \boldsymbol{\theta}) \right]\big|_{\boldsymbol{\theta} = \boldsymbol{\theta}_{t+1}} = \underline{0}$$

A simple calculation using the joint $p(\boldsymbol{y}, \boldsymbol{z})$ will show that this reduces to the M-step.

The derivation is easily extended to the generalized EM iterations. The coordinate-wise strategy also need not be employed and the BLR steps can also be run in parallel. A such scheme resembles the generalized EM algorithm and usually converge faster. Online schemes can be obtained by using

stochastic gradients, similar to those considered in Titterington [1984], Neal and Hinton [1998], Sato [1999], Cappé and Moulines [2009], Delyon et al. [1999]. More recently, Amid and Warmuth [2020] use a divergence-based approach and Kunstner et al. [2021] prove attractive convergence properties using a mirror-descent formulation similar to ours. Next, we derive a generalization of such online schemes, called the stochastic variational inference.

5.3   Stochastic variational inference and variational message passing

Stochastic variational inference (SVI) is a generalization of EM, first considered by Sato [2001] for the conjugate case and extended by Hoffman et al. [2013] to conditionally-conjugate models (a similar strategy was also proposed by Honkela et al. [2011]). We consider the conjugate case due to its simplicity. The model is assume to be similar to the EM case as in Eq. 56, but with N iid data examples $\boldsymbol{y}_i$ associated with one latent vector $\boldsymbol{z}_i$ each and a conjugate prior $p(\boldsymbol{\theta}|\boldsymbol{\alpha})$ with $\boldsymbol{\alpha} = (\boldsymbol{\alpha}_1, \alpha_2)$ as prior parameters,

$$p(\boldsymbol{y}_i|\boldsymbol{z}_i, \boldsymbol{\theta}) \propto \exp\left(\langle \widetilde{\boldsymbol{\lambda}}_i(\boldsymbol{y}_i, \boldsymbol{\theta}), \boldsymbol{T}_i(\boldsymbol{z}_i)\rangle\right), \qquad p(\boldsymbol{z}_i|\boldsymbol{\theta}) \propto \exp\left(\langle \boldsymbol{\lambda}_0(\boldsymbol{\theta}), \boldsymbol{T}_i(\boldsymbol{z}_i)\rangle\right),$$

$$p(\boldsymbol{y}_i, \boldsymbol{z}_i|\boldsymbol{\theta}) \propto \exp\left(\langle \boldsymbol{T}_{yz}(\boldsymbol{y}_i, \boldsymbol{z}_i), \boldsymbol{\theta}\rangle - A_{yz}(\boldsymbol{\theta})\right), \qquad p(\boldsymbol{\theta}|\boldsymbol{\alpha}) = h_0(\boldsymbol{\theta})\exp\left([\langle \boldsymbol{\alpha}_1, \boldsymbol{\theta}\rangle - \alpha_2 A_{yz}(\boldsymbol{\theta})]\right). \tag{57}$$

Due to conjugacy, each conditional update can be done coordinate wise. A common strategy is to assume a mean-field approximation

$$q(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N, \boldsymbol{\theta}) = q(\boldsymbol{\theta})\prod_i q(\boldsymbol{z}_i) \;\propto\; e^{\boldsymbol{\lambda}_0^{(1)} - \lambda_0^{(2)} A_{yz}(\boldsymbol{\theta})}\prod_i e^{\langle \boldsymbol{\lambda}_i, \boldsymbol{T}_i(\boldsymbol{z}_i)\rangle},$$

where $\boldsymbol{\lambda}_i$ is the natural parameter of $q(\boldsymbol{z}_i)$, and $(\boldsymbol{\lambda}_0^{(1)}, \lambda_0^{(2)})$ is the same for $q(\boldsymbol{\theta})$.

Coordinate-wise updates when applied to general graphical model are known as variational message passing Winn and Bishop [2005]. These updates are a special case of BLR, applied coordinate wise with $\rho_t = 1$. The derivation is almost identical to the one used for the EM case earlier, therefore omitted. One important difference is that, we do not employ the delta method for $q(\boldsymbol{\theta})$ and explicitly carry out the marginalization which has a closed-form expression due to conjugacy. The update for $q(\boldsymbol{\theta})$ is therefore replaced by its corresponding conjugate update.

Stochastic variational inference employs a special update where, after every $q(\boldsymbol{z}_i)$ update with $\rho_t = 1$, we update $q(\boldsymbol{\theta})$ but with $\rho_t < 1$. Clearly, this is also covered as a special case of the BLR. The BLR update is more general than these strategies since it applies to a wider class of non-conjugate models, as discussed next.

5.4   Non-conjugate variational inference

Consider a graphical model $p(\boldsymbol{x}) \propto \prod_{i \in \mathcal{I}} f_i(\boldsymbol{x}_i)$ where $\boldsymbol{x}$ denotes the set of latent variables $\boldsymbol{z}$ and parameters $\boldsymbol{\theta}$ and $f_i(\boldsymbol{x}_i)$ is the i'th factor operating on the subset $\boldsymbol{x}_i$ indexed by the set $\mathcal{I}$. The BLR can be applied to estimate a minimal exponential-family (assuming a constant base measure),

$$\boldsymbol{\lambda}_{t+1} \leftarrow (1 - \rho_t)\boldsymbol{\lambda}_t + \rho_t \sum_{i \in \mathcal{I}} \widetilde{\boldsymbol{\lambda}}_i(\boldsymbol{\mu}_t), \text{ where } \widetilde{\boldsymbol{\lambda}}_i(\boldsymbol{\mu}_t) = \nabla_{\boldsymbol{\mu}} \mathbb{E}_{q_t}\left[\log f_i(\boldsymbol{x}_i)\right]\big|_{\boldsymbol{\mu} = \boldsymbol{\mu}_t}. \tag{58}$$

where $\boldsymbol{\mu}_t = \nabla_{\boldsymbol{\lambda}} A(\boldsymbol{\lambda}_t)$. The setup is quite flexible and can represent for example a Bayesian network, a Markov random field and a Boltzmann machine. We can also view this as an update over distributions, by multiplying by $\boldsymbol{T}(\boldsymbol{x})$ followed by exponentiation,

$$q_{t+1}(\boldsymbol{x}) \propto q_t(\boldsymbol{x})^{(1-\rho_t)}\left[\prod_{i \in \mathcal{I}} \exp\left(\langle \widetilde{\boldsymbol{\lambda}}_i(\boldsymbol{\mu}_t), \boldsymbol{T}(\boldsymbol{x}_i)\rangle\right)\right]^{\rho_t} \tag{59}$$

and at convergence

$$q_*(\boldsymbol{x}) \propto \prod_{i \in \mathcal{I}} \exp\left(\langle \widetilde{\boldsymbol{\lambda}}_i(\boldsymbol{\mu}_*), \boldsymbol{T}(\boldsymbol{x}_i) \rangle\right),$$

where $q_*(\boldsymbol{x})$ is the optimal distribution with natural and expectation parameters as $\boldsymbol{\lambda}_*$ and $\boldsymbol{\mu}_*$. This update has three important features.

1. The optimal $q_*(\boldsymbol{x})$ has the same structure as $p(\boldsymbol{x})$ and each term in $q_*(\boldsymbol{x})$ yields an approximation to those in $p(\boldsymbol{x})$,

$$f_i(\boldsymbol{x}_i) \approx c_i \exp\left(\langle \widetilde{\boldsymbol{\lambda}}_i(\boldsymbol{\mu}_*), \boldsymbol{T}(\boldsymbol{x}_i) \rangle\right)$$

for some constant $c_i > 0$. The quantity $\widetilde{\boldsymbol{\lambda}}_i(\cdot)$ can be interpreted as the natural parameter of the local approximation. Such local parameters are often referred to as the site parameters in expectation propagation (EP) where their computations are often rather involved. In our formulation they are simply the natural gradients of the expected log joint density. See Chang et al. [2020, Sec. 3.4 and 3.5] for an example comparing the two.

2. The update in Eq. 59 yields both the local $\widetilde{\boldsymbol{\lambda}}_i(\cdot)$ and global $\boldsymbol{\lambda}$ parameters simultaneously, which is unlike the message-passing strategies that only compute the local messages. In fact, Eq. 59 can also be reformulated entirely in terms of local parameters (denoted by $\widetilde{\boldsymbol{\lambda}}_{i,t+1}$ below),

$$\boldsymbol{\lambda}_{t+1} \leftarrow \sum_{i \in \mathcal{I}} \widetilde{\boldsymbol{\lambda}}_{i,t+1}, \text{ where } \widetilde{\boldsymbol{\lambda}}_i^{(t+1)} \leftarrow (1 - \rho_t)\widetilde{\boldsymbol{\lambda}}_i^{(t)} + \rho_t \widetilde{\boldsymbol{\lambda}}_i(\boldsymbol{\mu}_t), \tag{60}$$

This formulation was discussed by Khan and Lin [2017] who show this to be a generalization of the non-conjugate variational message passing algorithm [Knowles and Minka, 2011]. The form is suitable for a distributed implementation using a message-passing framework. It is important to note that the local approximations are optimal and therefore inherently superior to the local-variational bounds, for example, quadratic bounds used in logistic regression [Jaakkola and Jordan, 1996, Khan et al., 2010, 2012, Khan, 2012], and also to the augmentation strategies [Girolami and Rogers, 2006, Klami, 2015].

3. The update applies to both conjugate and non-conjugate factors, and probabilistic and deterministic variables. Automatic differentiation tools can be easily be integrated since the update does not make a distinction between those. All this makes the BLR an interesting candidate for practical probabilistic programming [van de Meent et al., 2018].

We have argued that natural gradients play an important role for probabilistic inference, both for conjugate models or within those using message-passing frameworks. Still, their importance is somehow missed in the community. Natural gradients were first introduced in this context by Khan and Lin [2017], Khan et al. [2018], although Salimans and Knowles [2013] also mention a similar condition without an explicit reference to natural gradients. Sheth and Khardon [2016], Sheth [2019] discuss an update similar to the BLR for a specific case of two-level graphical models. Salimbeni et al. [2018] propose natural-gradient strategies for sparse GPs by using automatic differentiation which is often slower (the original work by Hensman et al. [2013], is in fact a special case of the BLR). A recent generalization to structured natural-gradient by Lin et al. [2021] is also worth noting.

We will end the section by discussing the connections of the BLR to the algorithms used in online learning, where the loss functions are observed sequentially at every iteration t,

$$q_{t+1}(\boldsymbol{\theta}) = \underset{q(\boldsymbol{\theta})}{\arg\min} \quad \mathbb{E}_q\left[\ell(y_t, f_{\boldsymbol{\theta}}(\boldsymbol{x}_t))\right] + \frac{1}{\rho_t}\mathbb{D}_{KL}[q(\boldsymbol{\theta}) \,\|\, q_t(\boldsymbol{\theta})].$$

The resulting updates are called Exponential-Weight (EW) updates [Littlestone and Warmuth, 1994, Vovk, 1990], which are closely related to Bayes' rule. [Hoeven et al., 2018] show that, by approximating the first term by a surrogate (linear/quadratic) at the posterior mean (the delta method), many existing online learning algorithms can be obtained as special cases. This includes, online gradient descent, online Newton step, online mirror descent, among many others. This derivation is similar to the BLR which, when applied here, is slightly more general, not only due to the use of expected loss, but also because the surrogate choice is automatically obtained by the posterior approximations.

An advantage of this connection is that the theoretical guarantees derived in online learning can be translated to the BLR, and consequently to all the learning-algorithms derived from it. We will also note that the local and global BLR updates presented in Eqs. 58 and 60 are similar to the 'greedy' and 'lazy' updates used in online learning [Hoeven et al., 2018]. The conjugate case discussed in earlier sections are very similar to the algorithms proposed for online learning by Azoury and Warmuth [2001], which are concurrent to a similar proposal in the Bayesian community by Sato [2001].

6 Discussion

To learn, we need to extract useful information from new data and revise our beliefs. To do this well, we must reject false information and, at the same time, not ignore any (real) information. A good learning algorithm too must possess these properties. We expect the same to be true for the algorithms that have "stood the test of time" in terms of good performance, even if they are derived through empirical experimentation and intuitions. If there exist an optimal learning-algorithm, as it is often hypothesized, then we might be able to view these empirically good learning-algorithms as derived from this common origin. They might not be perfect but we might expect them to be a reasonable approximation of the optimal learning-algorithm. All in all, it is possible that all such algorithms optimize similar objectives and use similar strategies on how the optimization proceeds.

In this paper we argue for two ideas. The first is that the learning-objective is a Bayesian one and uses the variational formulation by Zellner [1988] in Eq. 2. The objective tells us *how* to balance new information with our old one, resulting ultimately in Bayes theorem as the optimal information processing rule. The power of the variational formulation, is that it also shows how to process the information when we have limited abilities, for example, in terms of computation, to extract the relevant information. The role of $q(\boldsymbol{\theta})$ is to define how to represent the knowledge, for which exponential families and mixtures thereof, are natural choices that have shown to balance complexity while still being practical manageable.

The second idea is the role the natural gradients play in the learning algorithms. Bayes' rule in its original form has no opinion about them, but they are inherently present in all solutions of the Bayesian objective. They give us information about the learning-loss landscape, sometimes in the form of the derivatives of the loss and sometimes as the messages in a graphical model. Our work shows that all these seemingly different ideas, in fact, have deep roots in information geometry that is being exploited by the natural gradients and then in the Bayesian Learning Rule (BLR) we proposed.

The two ideas go together as a symbiosis into the BLR. By a long series of examples from optimisation, deep learning, and graphical models, we demonstrate that classical algorithms such as ridge regression, Newton's method, and Kalman filter, as well as modern deep-learning algorithms such as stochastic-gradient descent, RMSprop, and Dropout, all can be derived from the proposed BLR. The BLR gives a unified framework and understanding of what various (good) learning algorithms do, how they differ in approximating the target and then also how they can be improved. The main advantage of the BLR, is that we now have a principled framework to approach new learning problems, both in terms of what to aim for and also how to get there.

Acknowledgement

References

L. Aitchison. Bayesian filtering unifies adaptive and non-adaptive neural network optimization methods. *arXiv preprint arXiv:1807.07540*, 2018.

M. Alizadeh, J. Fernández-Marqués, N. D. Lane, and Y. Gal. An empirical study of binary neural networks' optimisation. *ICLR*, 2019.

S. Amari. *Information geometry and its applications*. Springer, 2016.

S-I. Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

E. Amid and M. K. Warmuth. Divergence-based motivation for online em and combining hidden variable models. In J. Peters and D. Sontag, editors, *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI)*, volume 124 of *Proceedings of Machine Learning Research*, pages 81–90. PMLR, 03–06 Aug 2020. URL `http://proceedings.mlr.press/v124/amid20a.html`.

D. F. Andrews and C. L. Mallows. Scale mixtures of normal distributions. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 99–102, 1974.

Katy S Azoury and Manfred K Warmuth. Relative loss bounds for on-line density estimation with the exponential family of distributions. *Machine Learning*, 43(3):211–246, 2001.

A. Azzalini. The skew-normal distribution and related multivariate families. *Scandinavian Journal of Statistics*, 32(2):159–188, 2005.

A. K. Balan, V. Rathod, K. P. Murphy, and M. Welling. Bayesian dark knowledge. In *Advances in Neural Information Processing Systems*, pages 3438–3446, 2015.

A. Banerjee, S. Merugu, I. S. Dhillon, and J. Ghosh. Clustering with bregman divergences. *Journal of Machine Learning Research*, 6(Oct):1705–1749, 2005.

O. E. Barndorff-Nielsen. Normal inverse Gaussian distributions and stochastic volatility modelling. *Scandinavian Journal of statistics*, 24(1):1–13, 1997.

A. G. Barto and R. S. Sutton. Goal seeking components for adaptive intelligence: An initial assessment. Technical report, MASSACHUSETTS UNIV AMHERST DEPT OF COMPUTER AND INFORMATION SCIENCE, 1981.

S. Becker and Y. LeCun. Improving the convergence of back-propagation learning with second order methods. In *Proceedings of the 1988 connectionist models summer school*, pages 29–37, 1988.

Y. Bengio, N. Léonard, and A. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Z. W. Birnbaum and S. C. Saunders. A new family of life distributions. *Journal of applied probability*, 6(2):319–327, 1969.

C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006. ISBN 0387310738.

P. G. Bissiri, C. C. Holmes, and S. G. Walker. A general framework for updating belief distributions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 78(5):1103–1130, 2016. doi: 10.1111/rssb.12158. URL `https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/rssb.12158`.

D. M. Blei and J. D. Lafferty. A correlated topic model of science. *The Annals of Applied Statistics*, pages 17–35, 2007.

D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight uncertainty in neural networks. In *International Conference on Machine Learning*, pages 1613–1622, 2015.

G. Bonnet. Transformations des signaux aléatoires a travers les systemes non linéaires sans mémoire. In *Annales des Télécommunications*, volume 19, pages 203–220. Springer, 1964.

L. Bottou, F. E. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. *arXiv preprint arXiv:1606.04838*, 2016.

M. Braun and J. McAuliffe. Variational inference for large-scale models of discrete choice. *Journal of the American Statistical Association*, 105(489):324–335, 2010.

R. G. Brown. *Statistical forecasting for inventory control*. McGraw/Hill, 1959.

O. Cappé and E. Moulines. On-line expectation–maximization algorithm for latent data models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(3):593–613, 2009.

O Catoni. Pac-bayesian supervised classification: The thermodynamics of statistical learning. institute of mathematical statistics lecture notes—monograph series 56. *IMS, Beachwood, OH. MR2483528*, 5544465, 2007.

A. Cauchy. Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, 25(1847):536–538, 1847.

P. E. Chang, W. J. Wilkinson, M. E. Khan, and A. Solin. Fast variational learning in state-space gaussian process models. In *2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2020.

P. Chaudhari and S. Soatto. Stochastic gradient descent performs variational inference, converges to limit cycles for deep networks. In *2018 Information Theory and Applications Workshop (ITA)*, pages 1–10. IEEE, 2018.

M. Courbariaux, Y. Bengio, and J-P. David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in neural information processing systems*, pages 3123–3131, 2015.

B. Delyon, M. Lavielle, and E. Moulines. Convergence of a stochastic approximation version of the em algorithm. *The Annals of Statistics*, 27(1):94–128, 1999.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

R. A. Dorfman. A note on the delta-method for finding variance formulae. *Biometric Bulletin*, 1938.

Gintare Karolina Dziugaite and Daniel M Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. *arXiv preprint arXiv:1703.11008*, 2017.

T. Eltoft, T. Kim, and T-W. Lee. Multivariate scale mixture of Gaussians modeling. In *International Conference on Independent Component Analysis and Signal Separation*, pages 799–806. Springer, 2006.

Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pages 1050–1059, 2016.

Y. Gal, J. Hron, and A. Kendall. Concrete dropout. In *Advances in neural information processing systems*, pages 3581–3590, 2017.

E. S. Gardner Jr. Exponential smoothing: The state of the art. *Journal of forecasting*, 4(1):1–28, 1985.

M. Girolami and S. Rogers. Variational Bayesian multinomial probit regression with Gaussian process priors. *Neural Computation*, 18(8):1790–1817, 2006.

A. Graves. Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems*, pages 2348–2356, 2011.

A. Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

E. Grushka. Characterization of exponentially modified Gaussian peaks in chromatography. *Analytical Chemistry*, 44(11):1733–1738, 1972.

A. K. Gupta and D. K. Nagar. *Matrix variate distributions*, volume 104. CRC Press, 2018.

E. Hazan, K. Y. Levy, and S. Shalev-Shwartz. On graduated optimization for stochastic non-convex problems. In *International Conference on Machine Learning*, pages 1833–1841, 2016.

K. Helwegen, J. Widdicombe, L. Geiger, Z. Liu, K-T. Cheng, and R. Nusselder. Latent weights do not exist: Rethinking binarized neural network optimization. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/9ca8c9b0996bbf05ae7753d34667a6fd-Paper.pdf.

J. Hensman, N. Fusi, and N. D. Lawrence. Gaussian processes for big data. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, UAI'13, page 282–290, Arlington, Virginia, USA, 2013. AUAI Press.

J. M. Hernandez-Lobato and R. Adams. Probabilistic backpropagation for scalable learning of Bayesian neural networks. In *International Conference on Machine Learning*, pages 1861–1869, 2015.

S. Hochreiter and J. Schmidhuber. Simplifying neural nets by discovering flat minima. In *Advances in neural information processing systems*, pages 529–536, 1995.

S. Hochreiter and J. Schmidhuber. Flat minima. *Neural Computation*, 9(1):1–42, 1997.

A. E. Hoerl and R. W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

D. Hoeven, T. Erven, and W. Kotłowski. The many faces of exponential weights in online learning. In *Conference On Learning Theory*, pages 2067–2092. PMLR, 2018.

M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

C. C. Holt, F. Modigliani, J. F. Muth, and H. A. Simon. *Planning Production, Inventories, and Work Force.* Englewood Cliffs, 1960.

A. Honkela, T. Raiko, M. Kuusela, M. Tornio, and J. Karhunen. Approximate Riemannian conjugate gradient learning for fixed-form variational Bayes. *Journal of Machine Learning Research*, 11:3235–3268, 2011.

A. Huning. Evolutionsstrategie. optimierung technischer systeme nach prinzipien der biologischen evolution, 1976.

T. Jaakkola and M. Jordan. A variational approach to Bayesian logistic regression problems and their extensions. In *International conference on Artificial Intelligence and Statistics*, 1996.

E. Jang, S Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

E. T. Jaynes. On the rationale of maximum-entropy methods. *Proceedings of the IEEE*, 70(9):939–952, 1982. doi: 10.1109/PROC.1982.12425.

R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 03 1960. ISSN 0021-9223. doi: 10.1115/1.3662552. URL https://doi.org/10.1115/1.3662552.

N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

M. E. Khan. *Variational Learning for Latent Gaussian Models of Discrete Data.* PhD thesis, University of British Columbia, 2012.

M. E. Khan and W. Lin. Conjugate-computation variational inference: converting variational inference in non-conjugate models to inferences in conjugate models. In *International Conference on Artificial Intelligence and Statistics*, pages 878–887, 2017.

M. E. Khan and D. Nielsen. Fast yet simple natural-gradient descent for variational inference in complex models. In *2018 International Symposium on Information Theory and Its Applications (ISITA)*, pages 31–35. IEEE, 2018.

M. E. Khan, B. Marlin, G. Bouchard, and K. Murphy. Variational Bounds for Mixed-Data Factor Analysis. In *Advances in Neural Information Processing Systems*, 2010.

M. E. Khan, S. Mohamed, B. Marlin, and K. Murphy. A stick-breaking likelihood for categorical data analysis with latent gaussian models. In *Artificial Intelligence and Statistics*, pages 610–618. PMLR, 2012.

M. E. Khan, D. Nielsen, V. Tangkaratt, W. Lin, Y. Gal, and A. Srivastava. Fast and scalable Bayesian deep learning by weight-perturbation in Adam. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2611–2620, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/khan18a.html.

M. E. Khan, A. Immer, E. Abedi, and M. Korzepa. Approximate inference turns deep networks into gaussian processes. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

D. P. Kingma and M. Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

A. Klami. Polya-gamma augmentations for factor models. In D. Phung and H. Li, editors, *Proceedings of the Sixth Asian Conference on Machine Learning*, volume 39 of *Proceedings of Machine Learning Research*, pages 112–128, Nha Trang City, Vietnam, 26–28 Nov 2015. PMLR.

D. A. Knowles and T. Minka. Non-conjugate variational message passing for multinomial and binary regression. In *Advances in Neural Information Processing Systems*, pages 1701–1709, 2011.

D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

F. Kunstner, R. Kumar, and M. Schmidt. Homeomorphic-invariance of em: Non-asymptotic convergence in kl divergence for exponential families via mirror descent. In A. Banerjee and K. Fukumizu, editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 3295–3303. PMLR, 13–15 Apr 2021. URL http://proceedings.mlr.press/v130/kunstner21a.html.

P. S. Laplace. Memoir on the probability of the causes of events. *Statistical science*, 1(3):364–378, 1986.

N. Le Roux and A. W. Fitzgibbon. A fast natural newton method. In *International Conference on Machine Learning*, 2010.

Y. LeCun, L. Bottou, and G. B. Orrand K-R. Müller. Efficient backprop. In *Neural Networks: Tricks of the Trade, This Book is an Outgrowth of a 1996 NIPS Workshop*, page 9–50, Berlin, Heidelberg, 1998. Springer-Verlag. ISBN 3540653112.

M. Leordeanu and M. Hebert. Smoothing-based optimization. In *Computer Vision and Pattern Recognition*, pages 1–8, 2008.

W. Lin, M. E. Khan, and M. Schmidt. Stein's lemma for the reparameterization trick with exponential family mixtures. *arXiv preprint arXiv:1910.13398*, 2019a.

W. Lin, M. E. Khan, and M. Schmidt. Fast and simple natural-gradient variational inference with mixture of exponential-family approximations. 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 3992–4002. PMLR, 09–15 Jun 2019b. URL http://proceedings.mlr.press/v97/lin19b.html.

W. Lin, F. Nielsen, M. E. Khan, and M. Schmidt. Tractable structured natural-gradient descent using local parameterizations. In *Proceedings of the 38th International Conference on Machine Learning*, 2021.

N. Littlestone and M. K. Warmuth. The weighted majority algorithm. *Information and computation*, 108(2):212–261, 1994.

C. Louizos and M. Welling. Structured and efficient variational deep learning with matrix gaussian posteriors. In *International Conference on Machine Learning*, pages 1708–1716, 2016.

D. Mackay. *Bayesian Methods for Adaptive Models*. PhD thesis, California Institute of Technology, 1991.

C. J. Maddison, A. Mnih, and Y. W. Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

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*, pages 13153–13164, 2019.

L. Malagò and G. Pistone. Information geometry of the gaussian distribution in view of stochastic optimization. In *Proceedings of the 2015 ACM Conference on Foundations of Genetic Algorithms XIII*, pages 150–162, 2015.

L. Malagò, M. Matteucci, and G. Pistone. Towards the geometry of estimation of distribution algorithms based on the exponential family. In *Proceedings of the 11th workshop proceedings on Foundations of genetic algorithms*, pages 230–242, 2011.

S. Mandt, M. D. Hoffman, and D. M. Blei. Stochastic gradient descent as approximate Bayesian inference. *Journal of Machine Learning Research*, 18:1–35, 2017.

James Martens. New insights and perspectives on the natural gradient method. *Journal of Machine Learning Research*, 21(146):1–76, 2020. URL `http://jmlr.org/papers/v21/17-678.html`.

X. Meng, R. Bachmann, and M. E. Khan. Training binary neural networks using the bayesian learning rule. *arXiv preprint arXiv:2002.10778*, 2020.

A. Mishkin, F. Kunstner, D. Nielsen M. Schmidt, and M. E. Khan. Slang: Fast structured covariance approximations for bayesian deep learning with natural gradient. In *Advances in Neural Information Processing Systems 31*. 2018.

H. Mobahi and J. W. Fisher III. A theoretical analysis of optimization by Gaussian continuation. In *AAAI Conference on Artificial Intelligence*, pages 1205–1211, 2015.

K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012. ISBN 0262018020, 9780262018029.

R. M. Neal and G. E. Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pages 355–368. Springer, 1998.

A. Nemirovski and D. Yudin. On cesaro's convergence of the gradient descent method for finding saddle points of convex-concave functions. *Doklady Akademii Nauk SSSR*, 239(4), 1978.

F. Nielsen and V. Garcia. Statistical exponential families: A digest with flash cards. *arXiv preprint arXiv:0911.4863*, 2009.

Y. Ollivier. Online natural gradient as a kalman filter. *Electronic Journal of Statistics*, 12(2):2930–2961, 2018.

Y. Ollivier, L. Arnold, A. Auger, and N. Hansen. Information-geometric optimization algorithms: A unifying picture via invariance principles. *Journal of Machine Learning Research*, 18(18):1–65, 2017.

M. Opper and C. Archambeau. The variational Gaussian approximation revisited. *Neural Computation*, 21(3):786–792, 2009.

K. Osawa, S. Swaroop, M. E. Khan, A. Jain, R. Eschenhagen, R. E. Turner, and R. Yokota. Practical deep learning with bayesian principles. In *Advances in neural information processing systems*, pages 4287–4299, 2019.

R. Pascanu and Y. Bengio. Revisiting natural gradient for deep networks. *arXiv preprint arXiv:1301.3584*, 2013.

B. T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

R. Price. A useful theorem for nonlinear devices having gaussian inputs. *IRE Transactions on Information Theory*, 4(2):69–72, 1958.

G. Raskutti and S. Mukherjee. The information geometry of mirror descent. *IEEE Transactions on Information Theory*, 61(3):1451–1457, 2015.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, pages 1278–1286, 2014.

H. Ritter, A. Botev, and D. Barber. A scalable laplace approximation for neural networks. In *International Conference on Learning Representations*, 2018.

H. Robbins and S. Monro. A stochastic approximation method. *Ann. Math. Statist.*, 22(3):400–407, 9 1951. doi: 10.1214/aoms/1177729586. URL http://dx.doi.org/10.1214/aoms/1177729586.

S. Roweis and Z. Ghahramani. A unifying review of linear gaussian models. *Neural computation*, 11(2): 305–345, 1999.

H. Rue and S. Martino. Approximate Bayesian inference for hierarchical Gaussian Markov random fields models. *Journal of Statistical Planning and Inference*, 137(10):3177–3192, 2007. Special Issue: Bayesian Inference for Stochastic Processes.

H. Rue, S. Martino, and N. Chopin. Approximate Bayesian inference for latent Gaussian models using integrated nested Laplace approximations (with discussion). *Journal of the Royal Statistical Society, Series B*, 71(2):319–392, 2009.

H. Rue, A. Riebler, S. H. Sørbye, J. B. Illian, D. P. Simpson, and F. K. Lindgren. Bayesian computing with INLA: A review. *Annual Reviews of Statistics and Its Applications*, 4(March):395–421, 2017. doi: 10.1146/annurev-statistics-060116-054045.

T. Salimans and D. A. Knowles. Fixed-form variational posterior approximation through stochastic linear regression. *Bayesian Analysis*, 8(4):837–882, 2013.

H. Salimbeni, S. Eleftheriadis, and J. Hensman. Natural gradients in practice: Non-conjugate variational inference in Gaussian process models. In *International Conference on Artificial Intelligence and Statistics*, pages 689–697. PMLR, 2018.

M-A. Sato. Fast learning of on-line em algorithm. Technical report, ATR Human Information Processing Research Laboratories, 1999.

M-A. Sato. Online model selection based on the variational Bayes. *Neural Computation*, 13(7):1649–1681, 2001.

T. Schaul, S. Zhang, and Y. LeCun. No more pesky learning rates. In *International Conference on Machine Learning*, pages 343–351, 2013.

R. Sheth. *Algorithms and Theory for Variational Inference in Two-Level Non-conjugate Models*. PhD thesis, Tufts University, 2019.

R. Sheth and R. Khardon. Monte carlo structured svi for two-level non-conjugate models. *arXiv preprint arXiv:1612.03957*, 2016.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

R. L. Stratonovich. Conditional markov processes. In *Non-linear transformations of stochastic processes*, pages 427–453. Elsevier, 1965.

I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147, 2013.

T. Tieleman and G. Hinton. Lecture 6.5-RMSprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning 4*, 2012.

L. Tierney and J. B. Kadane. Accurate approximations for posterior moments and marginal densities. *Journal of the American Statistical Association*, 81(393):82–86, 1986.

D. M. Titterington. Recursive parameter estimation using incomplete data. *Journal of the Royal Statistical Society: Series B (Methodological)*, 46(2):257–267, 1984.

J-W. van de Meent, B. Paige, H. Yang, and F. Wood. An introduction to probabilistic programming. *arXiv preprint arXiv:1809.10756*, 2018.

J. M. Ver Hoef. Who invented the delta method? *The American Statistician*, 66(2):124–127, 2012.

V. G. Vovk. Aggregating strategies. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, COLT '90, pages 371–386, San Francisco, CA, USA, 1990. Morgan Kaufmann Publishers Inc. ISBN 1-55860-146-5.

C. Wang and D. M. Blei. Variational inference in nonconjugate models. *Journal of Machine Learning Research*, 14(1):1005–1031, 2013.

M. Welling, C. Chemudugunta, and N. Sutter. Deterministic latent variable models and their pitfalls. In *Proceedings of the 2008 SIAM International Conference on Data Mining*, pages 196–207. SIAM, 2008.

D. Wierstra, T. Schaul, T. Glasmachers, Y. Sun, J. Peters, and J. Schmidhuber. Natural evolution strategies. *Journal of Machine Learning Research*, 15(1):949–980, 2014.

A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, and B. Recht. The marginal value of adaptive gradient methods in machine learning. In *Advances in Neural Information Processing Systems*, pages 4151–4161, 2017.

J. Winn and C. M. Bishop. Variational message passing. *Journal of Machine Learning Research*, 6 (Apr):661–694, 2005.

K-C. Wong. Evolutionary multimodal optimization: A short survey. *arXiv preprint arXiv:1508.00457*, 2015.

P. Yin, J. Lyu, S. Zhang, S. Osher, Y. Qi, and J. Xin. Understanding straight-through estimator in training activation quantized neural nets. *ICLR*, 2019.

X. Yu and M. Gen. *Introduction to evolutionary algorithms*. Springer Science & Business Media, 2010.

M. D. Zeiler. ADADELTA: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

A. Zellner. Optimal information processing and Bayes's theorem. *The American Statistician*, 42(4): 278–280, 1988. doi: 10.1080/00031305.1988.10475585. URL `https://amstat.tandfonline.com/doi/abs/10.1080/00031305.1988.10475585`.

C. Zhang, J. Bütepage, H. Kjellström, and S. Mandt. Advances in variational inference. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):2008–2026, 2018a.

G. Zhang, S. Sun, D. K. Duvenaud, and R. B. Grosse. Noisy natural gradient as variational inference. *arXiv preprint arXiv:1712.02390*, 2018b.

Tong Zhang. Theoretical analysis of a class of randomized regularization methods. In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*, COLT '99, page 156–163, New York, NY, USA, 1999. Association for Computing Machinery. ISBN 1581131674. doi: 10.1145/307400.307433. URL `https://doi.org/10.1145/307400.307433`.

A Bayesian inference as optimization

Bayesian inference is a special case of the Bayesian learning problem. Although this follows directly from the variational formulation by Zellner [1988], it might be useful to provide some more motivation. Bayesian inference corresponds to a log-likelihood loss $\ell(y, f_\theta(\boldsymbol{x})) = -\log p(y|f_\theta(\boldsymbol{x}))$. With conditional independent data and prior $p(\boldsymbol{\theta})$, the posterior distribution is $p(\boldsymbol{\theta}|\mathcal{D}) = p(\boldsymbol{\theta})/\mathcal{Z}(\mathcal{D}) \prod_{i=1}^{N} p(\boldsymbol{y}_i|f_\theta(\boldsymbol{x}_i))$. The minimizer of the Bayesian learning problem recovers this posterior distribution. This follows directly

from reorganizing the Bayesian objective Eq. 2:

$$\mathcal{L}(q) = -\mathbb{E}_{q(\theta)} \left[\sum_{i=1}^{N} \log p(y_i|f_\theta(\boldsymbol{x}_i)) \right] + \mathbb{D}_{KL}[q(\boldsymbol{\theta}) \,\|\, p(\boldsymbol{\theta})] \tag{61}$$

$$= \mathbb{E}_{q(\theta)} \left[\log \frac{q(\boldsymbol{\theta})}{\frac{p(\boldsymbol{\theta})}{\mathcal{Z}(\mathcal{D})} \prod_{i=1}^{N} p(y_i|f_\theta(\boldsymbol{x}_i))} \right] + \log \mathcal{Z}(\mathcal{D}) \tag{62}$$

$$= \mathbb{D}_{KL}[q(\boldsymbol{\theta}) \,\|\, p(\boldsymbol{\theta}|\mathcal{D})] + \log \mathcal{Z}(\mathcal{D}) \tag{63}$$

Choosing $q(\boldsymbol{\theta})$ as $p(\boldsymbol{\theta}|\mathcal{D})$ minimizes the above equation since $\mathcal{Z}(\mathcal{D})$ is a constant, and the KL divergence is zero.

B   Natural gradient of the entropy

Below, we show that $\widetilde{\nabla}_{\boldsymbol{\lambda}} \mathcal{H}(q) = -\boldsymbol{\lambda}$ when $h(\boldsymbol{\theta})$ is a constant, We rewrite the gradient with respect to $\boldsymbol{\mu}$,

$$-\nabla_{\boldsymbol{\lambda}} \mathcal{H}(q) = \boldsymbol{F}(\boldsymbol{\lambda}) \, \nabla_{\boldsymbol{\mu}} \mathbb{E}_q[\log q(\boldsymbol{\theta})],$$

since that gradient has a simpler expression as shown below,

$$\nabla_{\boldsymbol{\mu}} \mathbb{E}_q[\log q(\boldsymbol{\theta})] = \nabla_{\boldsymbol{\mu}} \left[\langle \boldsymbol{\lambda}, \boldsymbol{\mu} \rangle - A(\boldsymbol{\lambda}) \right] + \nabla_{\boldsymbol{\mu}} \mathbb{E}_q[\log h(\boldsymbol{\theta})]$$
$$= \boldsymbol{\lambda} + \left(\nabla_{\boldsymbol{\lambda}}^2 A(\boldsymbol{\lambda}) \right)^{-1} \left[\boldsymbol{\mu} - \nabla_{\boldsymbol{\lambda}} A(\boldsymbol{\lambda}) \right] + \nabla_{\boldsymbol{\mu}} \mathbb{E}_q[\log h(\boldsymbol{\theta})] \tag{64}$$
$$= \boldsymbol{\lambda} + \nabla_{\boldsymbol{\mu}} \mathbb{E}_q[\log h(\boldsymbol{\theta})].$$

By scaling with the inverse of the FIM, we get $-\widetilde{\nabla}_{\boldsymbol{\lambda}} \mathcal{H}(q) = \widetilde{\nabla}_{\boldsymbol{\lambda}} \mathbb{E}_q[\log q(\boldsymbol{\theta})] = \boldsymbol{\lambda} + \widetilde{\nabla}_{\boldsymbol{\lambda}} \mathbb{E}_q[\log h(\boldsymbol{\theta})]$, which for constant $h(\boldsymbol{\theta})$ reduces to $\boldsymbol{\lambda}$.

C   The delta method

In the delta method [Dorfman, 1938, Ver Hoef, 2012], we approximate the expectation of a function of a random variable by the expectation of the function's Taylor expansion. For example, given a distribution $q(\boldsymbol{\theta})$ with mean $\boldsymbol{m}$, we can use the first-order Taylor approximation to get,

$$\mathbb{E}_q[f(\boldsymbol{\theta})] \approx \mathbb{E}_q \left[f(\boldsymbol{m}) + (\boldsymbol{\theta} - \boldsymbol{m})^\top \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{m}} \right] \approx f(\boldsymbol{m})$$

We will often use this approximation to approximate the expectation of gradient/Hessian at their values at a single point (usually the mean). For example, by using the first-order Taylor expansion, we approximate below the expectation of the gradient by the gradient at the mean,

$$\mathbb{E}_q[\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta})] = \nabla_{\boldsymbol{m}} \mathbb{E}_q[f(\boldsymbol{\theta})]$$
$$\approx \nabla_{\boldsymbol{m}} \mathbb{E}_q \left[f(\boldsymbol{m}) + (\boldsymbol{\theta} - \boldsymbol{m})^\top \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{m}} \right]$$
$$= \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{m}}, \tag{65}$$

where the first line is due to Bonnet's theorem (App. D) and the second line is simply the first-order Taylor expansion. We refer to this as the first-order delta method. Similar applications has been used in variational inference [Blei and Lafferty, 2007, Braun and McAuliffe, 2010, Wang and Blei, 2013].

Similarly, by using the second-order Taylor expansion, we approximate below the expectation of the Hessian by the Hessian at the mean,

$$
\begin{aligned}
\mathbb{E}_q[\nabla_{\boldsymbol{\theta}}^2 f(\boldsymbol{\theta})] &= 2\nabla_{\boldsymbol{S}^{-1}}\mathbb{E}_q[f(\boldsymbol{\theta})] \\
&\approx 2\nabla_{\boldsymbol{S}^{-1}}\mathbb{E}_q\left[f(\boldsymbol{m}) + (\boldsymbol{m}-\boldsymbol{\theta})^\top \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{m}} + \frac{1}{2}(\boldsymbol{m}-\boldsymbol{\theta})^\top \nabla_{\boldsymbol{\theta}}^2 f(\boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{m}}(\boldsymbol{m}-\boldsymbol{\theta})\right] \\
&= 2\nabla_{\boldsymbol{S}^{-1}}\mathrm{Tr}\left[\frac{1}{2}\boldsymbol{S}^{-1} \nabla_{\boldsymbol{\theta}}^2 f(\boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{m}}\right] \\
&= \nabla_{\boldsymbol{\theta}}^2 f(\boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{m}},
\end{aligned}
\tag{66}
$$

where the first line is due to Price's theorem (App. D) and the second line is simply the second-order Taylor expansion. We refer to this as the second-order delta method.

We also note that both the first and second-order approximation can be derived in much simpler way by using a zeroth-order expansion where we do not expand the function inside at all,

$$
\mathbb{E}_q[\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta})] \approx \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{m}} \qquad \mathbb{E}_q[\nabla_{\boldsymbol{\theta}}^2 f(\boldsymbol{\theta})] \approx \nabla_{\boldsymbol{\theta}}^2 f(\boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{m}}.
$$

In any case, these can be seen as simply approximating the expectations of a function with the function value at a single point (the mean).

D Newton's method from the BLR

We start with a summary of the Theorem's of Bonnet's and Price's [Bonnet, 1964, Price, 1958], see also Opper and Archambeau [2009], Rezende et al. [2014]. Let $\boldsymbol{x}$ be multivariate normal with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{C}$, $f(\boldsymbol{x})$ a twice differentiable function where $\mathbb{E}|f(\boldsymbol{x})| < \infty$, then the Bonnet and Price's results are

$$
\frac{\partial}{\partial \mu_i}\mathbb{E}\left[f(\boldsymbol{x})\right] = \mathbb{E}\left[\frac{\partial}{\partial x_i}f(\boldsymbol{x})\right] \quad \text{and} \quad \frac{\partial}{\partial C_{ij}}\mathbb{E}\left[f(\boldsymbol{x})\right] = c_{ij}\mathbb{E}\left[\frac{\partial^2}{\partial x_i \partial x_j}f(\boldsymbol{x})\right]
$$

where $c_{ij} = 1/2$ if $i = j$ and $c_{ij} = 1$ if $i \neq j$ (since $\boldsymbol{C}$ is symmetric), respectively. Using these, we obtain Eq. 10 and Eq. 11.

The following derivation is also detailed in Khan et al. [2018]. We now use the definitions of natural and expectation parameters from Eq. 9 in the BLR Eq. 6. The base-measure is constant for the multivariate Gaussian, hence

$$
\begin{aligned}
\boldsymbol{S}_{t+1}\boldsymbol{m}_{t+1} &\leftarrow (1-\rho_t)\boldsymbol{S}_t\boldsymbol{m}_t - \rho_t\nabla_{\boldsymbol{\mu}^{(1)}}\mathbb{E}_{q_t}[\bar{\ell}(\boldsymbol{\theta})] \\
\boldsymbol{S}_{t+1} &\leftarrow (1-\rho_t)\boldsymbol{S}_t + 2\rho_t\nabla_{\boldsymbol{\mu}^{(2)}}\mathbb{E}_{q_t}[\bar{\ell}(\boldsymbol{\theta})].
\end{aligned}
$$

The update for $\boldsymbol{S}_{t+1}$ can be obtained using Eq. 11,

$$
\boldsymbol{S}_{t+1} \leftarrow (1-\rho_t)\boldsymbol{S}_t + \rho_t\mathbb{E}_{q_t}\left[\nabla_{\boldsymbol{\theta}}^2\bar{\ell}(\boldsymbol{\theta})\right].
$$

We can simplify the update for the mean using Eq. 10,

$$
\begin{aligned}
\boldsymbol{S}_{t+1}\boldsymbol{m}_{t+1} &= (1-\rho_t)\boldsymbol{S}_t\boldsymbol{m}_t - \rho_t\left\{\mathbb{E}_{q_t}[\nabla_{\boldsymbol{\theta}}\bar{\ell}(\boldsymbol{\theta})] - \mathbb{E}_{q_t}[\nabla_{\boldsymbol{\theta}}^2\bar{\ell}(\boldsymbol{\theta})]\boldsymbol{m}_t\right\}, \\
&= \boldsymbol{S}_{t+1}\boldsymbol{m}_t - \rho_t\mathbb{E}_{q_t}[\nabla_{\boldsymbol{\theta}}\bar{\ell}(\boldsymbol{\theta})],
\end{aligned}
$$

and the update for $\boldsymbol{m}_{t+1}$ follows.

E Natural-gradient updates for mixture of Gaussians

The joint distribution $q(\boldsymbol{\theta}, z)$ is a minimal-conditional exponential-family distribution [Lin et al., 2019b], i.e., the distribution then takes a minimal form conditioned over $z = k$ which ensures that the FIM is non-singular. Details of the update are in Lin et al. [2019b, Theorem 3]. The natural and expectation parameters are given in Lin et al. [2019b, Table 1]).

The gradients with respect to $\boldsymbol{\mu}_k$ can be written in terms of $\boldsymbol{m}_k$ and $\boldsymbol{S}_k^{-1}$ as

$$\nabla_{\boldsymbol{\mu}_k^{(1)}}\mathbb{E}_q[\bar{\ell}(\boldsymbol{\theta})] = \left[\nabla_{\boldsymbol{\mu}_k^{(1)}}\boldsymbol{m}_k^\top\right]\nabla_{\mathbf{m}_k}\mathbb{E}_q[\bar{\ell}(\boldsymbol{\theta})] = \frac{1}{\pi_k}\nabla_{\mathbf{m}_k}\mathbb{E}_q[\bar{\ell}(\boldsymbol{\theta})]$$

Hence we can utilize from the derivation of Newton's method, Eq. 10 and Eq. 11, and do similarly as in App. D, to achieve a Newton-like update for a mixture of Gaussians

$$\boldsymbol{m}_{k,t+1} \leftarrow \boldsymbol{m}_{k,t} - \frac{\rho}{\pi_k}\boldsymbol{S}_{k,t+1}^{-1}\nabla_{\mathbf{m}_k}\mathbb{E}_{q_t}\left[\bar{\ell}(\boldsymbol{\theta}) + \log q(\boldsymbol{\theta})\right], \tag{67}$$

$$\boldsymbol{S}_{k,t+1} \leftarrow \boldsymbol{S}_{k,t} + \frac{2\rho}{\pi_k}\nabla_{\boldsymbol{S}_{k,t}^{-1}}\mathbb{E}_{q_t}\left[\bar{\ell}(\boldsymbol{\theta}) + \log q(\boldsymbol{\theta})\right]. \tag{68}$$

The gradients can be expressed in terms of the gradient and Hessian of the loss, similar to Bonnet's and Price's Theorem in App. D, see [Lin et al., 2019a, Thm 6 and 7]. This gives

$$\nabla_{\mathbf{m}_k}\mathbb{E}_q[f(\boldsymbol{\theta})] = \pi_k\mathbb{E}_{\mathcal{N}(\boldsymbol{\theta}|\mathbf{m}_k,\mathbf{S}_k^{-1})}[\nabla_{\boldsymbol{\theta}}f(\boldsymbol{\theta})],$$

$$\nabla_{\mathbf{S}_k^{-1}}\mathbb{E}_q[f(\boldsymbol{\theta})] = \frac{\pi_k}{2}\mathbb{E}_{\mathcal{N}(\boldsymbol{\theta}|\mathbf{m}_k,\mathbf{S}_k^{-1})}[\nabla_{\boldsymbol{\theta}}^2 f(\boldsymbol{\theta})],$$

and hence the updates in Eqs. 29 and 30.

The gradient of $\log q(\boldsymbol{\theta}) = \log\sum_{k=1}^K \pi_k\mathcal{N}(\boldsymbol{\theta}|\boldsymbol{m}_k, \boldsymbol{S}_k^{-1})$ were not spelled out explicit in Eqs. 67 and 68, but are as follows,

$$\nabla_{\boldsymbol{\theta}}\log q(\boldsymbol{\theta}) = \sum_{j=1}^K r_j(\boldsymbol{\theta})\boldsymbol{S}_j(\boldsymbol{m}_j - \boldsymbol{\theta}), \tag{69}$$

$$\nabla_{\boldsymbol{\theta}}^2\log q(\boldsymbol{\theta}) = \sum_{j=1}^K r_j(\boldsymbol{\theta})\left[\boldsymbol{A}_{jj}(\boldsymbol{\theta}) - \boldsymbol{S}_j - \sum_{i=1}^K r_i(\boldsymbol{\theta})\boldsymbol{A}_{ij}(\boldsymbol{\theta})\right]. \tag{70}$$

Here, $r_k(\boldsymbol{\theta})$ is the responsibility of the k'th mixture component at $\boldsymbol{\theta}$, $q(z = k|\boldsymbol{\theta})$

$$q(z = k|\boldsymbol{\theta}) = \frac{\pi_k\mathcal{N}(\boldsymbol{\theta}|\boldsymbol{m}_k, \boldsymbol{S}_k^{-1})}{\sum_{j=1}^K \pi_j\mathcal{N}(\boldsymbol{\theta}|\boldsymbol{m}_j, \boldsymbol{S}_j^{-1})}. \tag{71}$$

and $\boldsymbol{A}_{ij}(\boldsymbol{\theta}) = \boldsymbol{S}_i(\boldsymbol{m}_i - \boldsymbol{\theta})(\boldsymbol{m}_j - \boldsymbol{\theta})^\top\boldsymbol{S}_j$.

There are two natural situations that lead to simplifications, which can form the basis for various approximations. First, is when $\boldsymbol{\theta}_k = \boldsymbol{m}_k$, then $\boldsymbol{A}_{ik}(\boldsymbol{\theta}_k) = 0$ for all i, leading to

$$\nabla_{\boldsymbol{\theta}}\log q(\boldsymbol{\theta}_k) = \sum_{j\neq k} r_j(\boldsymbol{\theta}_k)\boldsymbol{S}_j(\boldsymbol{\theta}_j - \boldsymbol{\theta}_k), \tag{72}$$

$$\nabla_{\boldsymbol{\theta}}^2\log q(\boldsymbol{\theta}_k) = -r_k(\boldsymbol{\theta}_k)\boldsymbol{S}_k + \sum_{j\neq k} r_j(\boldsymbol{\theta}_k)\left[\boldsymbol{A}_{ii}(\boldsymbol{\theta}_k) - \boldsymbol{S}_j - \sum_{i=1,i\neq k}^K r_i(\boldsymbol{\theta}_k)\boldsymbol{A}_{ij}(\boldsymbol{\theta}_k)\right], \tag{73}$$

Second, when mixture components are far apart from each other, then the responsibility is approximately zero for all $j \neq k$, leading to

$$\nabla_{\boldsymbol{\theta}}\log q(\boldsymbol{\theta}_k) \approx 0, \qquad \nabla_{\boldsymbol{\theta\theta}}^2\log q(\boldsymbol{\theta}_k) \approx -\boldsymbol{S}_k. \tag{74}$$

F   Deep learning algorithms with momentum

Momentum is a useful technique to improve the performance of SGD [Sutskever et al., 2013]. Modern adaptive-learning algorithms, such as RMSprop and Adam, often employ variants of the classical momentum to increase the learning rate [Sutskever et al., 2013].

The classical momentum method is based on the Polyak's heavy-ball method Polyak [1964], where the current $\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t$ update is pushed along the direction of the previous update $\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t-1}$:

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \alpha \widehat{\nabla}_\theta \bar{\ell}(\boldsymbol{\theta}_t) + \gamma(\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t-1}) \tag{75}$$

with a fixed momentum coefficient $\gamma > 0$. Why this technique can be beneficial when using noisy gradient is more transparent when we rewrite Eq. 75 as

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \alpha \sum_{k=0}^{t} \gamma^k \widehat{\nabla}_\theta \bar{\ell}(\boldsymbol{\theta}_{t-k}) + \text{initial conditions.} \tag{76}$$

Hence, the effective gradient is an average of previous gradients with decaying weights. This will smooth out and reduce the variance of the stochastic gradient estimates.

Adaptive-learning rate algorithms, such as RMSprop and Adam, employ a variant of the classical momentum method where exponential-smoothing is applied to the gradient $\boldsymbol{u}_{t+1} \leftarrow \gamma \boldsymbol{u}_t + (1-\gamma)\widehat{\nabla}_\theta \bar{\ell}(\boldsymbol{\theta}_t)$ which is used in the update while keeping the scaling unchanged [Graves, 2013, Kingma and Ba, 2015]. This gives the update

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \alpha \frac{1}{\sqrt{\boldsymbol{s}_{t+1}} + c\mathbf{1}} \circ \boldsymbol{u}_{t+1}. \tag{77}$$

We can express this similar to Eq. 75, as [Wilson et al., 2017]

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \alpha(1-\gamma)\frac{1}{\sqrt{\boldsymbol{s}_{t+1}} + c\mathbf{1}} \circ \left[\widehat{\nabla}_\theta \bar{\ell}(\boldsymbol{\theta}_t)\right] + \gamma \frac{\sqrt{\boldsymbol{s}_t} + c\mathbf{1}}{\sqrt{\boldsymbol{s}_{t+1}} + c\mathbf{1}} \circ (\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t-1}), \tag{78}$$

showing a dynamic scaling of the momentum depending (essentially) on the ratio of $\boldsymbol{s}_{t+1}$ and $\boldsymbol{s}_t$. Although this adaptive scaling is not counter-intuitive, it is a result of the somewhat arbitrary choices made. Can we justify this from the Bayesian principles, and can derive the corresponding BLR? The problem we face is provoked by the use of stochastic gradient approximations and the principles simply state that we should compute exact gradients instead. This might appear as a deficiency in our main argument but using our free will, we can resolves this issue in other ways.

It is our view that the natural way to include a momentum term in the BLR, is to this within the mirror descent framework in Sec. 2. This will result in an update where the momentum term obey the geometry of the posterior approximation. We propose to augment Eq. 22 with a momentum term

$$\boldsymbol{\mu}_{t+1} \leftarrow \arg\min_{\boldsymbol{\mu} \in \mathcal{M}} \langle \nabla_{\boldsymbol{\mu}} \mathcal{L}(q_t), \boldsymbol{\mu} \rangle + \frac{1+\gamma_t}{\rho_t} \mathbb{D}_{A^*}(\boldsymbol{\mu} \| \boldsymbol{\mu}_t) - \frac{\gamma_t}{\rho_t} \mathbb{D}_{A^*}(\boldsymbol{\mu} \| \boldsymbol{\mu}_{t-1}) \tag{79}$$

The last term penalizes the proximity to $\boldsymbol{\mu}_{t-1}$ where the proximity is measured according to the KL divergence, following the suggestion by Khan et al. [2018] we may interpret this as the natural momentum term. This gives a (revised) BLR learning rule with momentum,

$$\boldsymbol{\lambda}_{t+1} \leftarrow \boldsymbol{\lambda}_t - \rho_t \widetilde{\nabla}_{\boldsymbol{\lambda}} \left[\mathbb{E}_{q_t} \left(\bar{\ell}(\boldsymbol{\theta}) \right) - \mathcal{H}(q_t) \right] + \gamma_t (\boldsymbol{\lambda}_t - \boldsymbol{\lambda}_{t-1}). \tag{80}$$

This update will recover Eq. 75 and an alternative to Eq. 78. Choosing a Gaussian candidate distribution $\mathcal{N}(\boldsymbol{\theta}|\boldsymbol{m}, \boldsymbol{I})$ (the result is invariant to a fixed covariance matrix), we recover Eq. 75 after a derivation

similar as in Sec. 1.3.1. Choosing the candidate distribution $\mathcal{N}(\boldsymbol{\theta}|\boldsymbol{m}, \mathrm{diag}(\boldsymbol{s})^{-1})$, follow again previous derivations (Newton's method in Sec. 1.3.2) with mini-batch gradients, and result in the following update

$$
\begin{aligned}
\boldsymbol{\theta}_{t+1} &\leftarrow \boldsymbol{\theta}_t - \rho_t \frac{1}{\boldsymbol{s}_{t+1}} \circ \left[\widehat{\nabla}_{\boldsymbol{\theta}} \bar{\ell}(\boldsymbol{\theta}_t) \right] + \frac{1}{\boldsymbol{s}_{t+1}} \circ (\boldsymbol{s}_t \boldsymbol{\theta}_t - \boldsymbol{s}_{t-1} \boldsymbol{\theta}_{t-1}), \\
\boldsymbol{s}_{t+1} &\leftarrow (1 - \rho_t) \boldsymbol{s}_t + \rho_t \, \mathrm{diag} \left[\widehat{\nabla}^2_{\boldsymbol{\theta}\boldsymbol{\theta}} \bar{\ell}(\boldsymbol{\theta}_t) \right] + \gamma_t (\boldsymbol{s}_t - \boldsymbol{s}_{t-1}).
\end{aligned}
\tag{81}
$$

Assuming $\boldsymbol{s}_t \approx \boldsymbol{s}_{t-1}$ the last terms in the above two equations simplifies, and then by replacing $\boldsymbol{s}_t$ by $\sqrt{\boldsymbol{s}_t}$ and adding a constant c, we can recover Eq. 78. This momentum-version of Eq. 37, not only justifies the use of scaling vectors for the momentum, but also to the use of exponential-smoothing for the scaling vectors itself.

G   Dropout from the BLR

Deriving updates for $\boldsymbol{m}_{jl}$ and $\boldsymbol{S}_{jl}$ with dropout is similar to the Newton variant derived for mixture of Gaussian discussed in App. E. With dropout there are only two components where one of the component has fixed parameters, hence one set of natural and expectation parameters per a weight vector $\boldsymbol{\theta}_{jl}$ for the j'th unit in l'th layer,

$$
\begin{aligned}
\boldsymbol{\lambda}^{(1)}_{jl} &= \boldsymbol{S}_{jl} \boldsymbol{m}_{jl}, & \boldsymbol{\mu}^{(1)}_{jl} &= \mathbb{E}_q \left[\mathbb{1}_{[z_{jl}=1]} \boldsymbol{\theta}_{jl} \right] = \pi_1 \boldsymbol{m}_{jl}, \\
\boldsymbol{\lambda}^{(2)}_{jl} &= -\tfrac{1}{2} \boldsymbol{S}_{jl}, & \boldsymbol{\mu}^{(2)}_{jl} &= \mathbb{E}_q \left[\mathbb{1}_{[z_{jl}=1]} \boldsymbol{\theta}_{jl} \boldsymbol{\theta}^T_{jl} \right] = \pi_1 \left(\boldsymbol{S}^{-1}_{jl} + \boldsymbol{m}_{jl} \boldsymbol{m}^T_{jl} \right),
\end{aligned}
$$

We can reuse the update Eqs. 67 and 68 from App. E

$$
\boldsymbol{m}_{jl,t+1} \leftarrow \boldsymbol{m}_{jl,t} - \frac{\rho}{\pi_1} \boldsymbol{S}^{-1}_{jl,t+1} \nabla_{\mathbf{m}_{jl}} \left\{ \mathbb{E}_{q_t} \left[\bar{\ell}(\boldsymbol{\theta}) \right] + \mathbb{E}_{q_t} \left[\log q(\boldsymbol{\theta}_{jl}) \right] \right\},
\tag{82}
$$

$$
\boldsymbol{S}_{jl,t+1} \leftarrow \boldsymbol{S}_{jl,t+1} + \frac{2\rho}{\pi_1} \nabla_{\mathbf{S}^{-1}_{jl}} \left\{ \mathbb{E}_{q_t} \left[\bar{\ell}(\boldsymbol{\theta}) \right] + \mathbb{E}_{q_t} \left[\log q(\boldsymbol{\theta}_{jl}) \right] \right\}.
\tag{83}
$$

We have moved the expectation inside the gradient since the entropy term only depends on $\boldsymbol{\theta}_{jl}$. The gradient of the loss $\bar{\ell}(\boldsymbol{\theta})$ may depend on the whole $\boldsymbol{\theta}$, however.

By using appropriate methods to compute gradients, we can recover the dropout method. Specifically, for the loss term, we use the reparameterization trick [Kingma and Welling, 2013] and approximate the gradients at the samples from $q(\boldsymbol{\theta}_{jl})$,

$$
\boldsymbol{\theta}_{jl} = z_{jl}(\boldsymbol{m}_{jl} + \boldsymbol{S}^{-1/2}_{jl} \boldsymbol{\epsilon}_{1,jl}) + (1 - z_{jl}) s_0^{-1/2} \boldsymbol{\epsilon}_{2,jl}
$$

where z_{jl} is a sample from Bernoulli distribution with probability π_1, and $\boldsymbol{\epsilon}_{1,jl}$ and $\boldsymbol{\epsilon}_{2,jl}$ are two independent samples from standard normal distribution. Since weights are deterministic in dropout, we need to ignore $\boldsymbol{\epsilon}_{1,jl}$ and $\boldsymbol{\epsilon}_{2,jl}$ to recover dropout from the BLR. The gradients are then evaluated at the dropout mean $\widetilde{\boldsymbol{m}}_{jl} = z_{jl} \boldsymbol{m}_{jl}$ (the delta method) which leads to the following gradient approximations:

$$
\begin{aligned}
\nabla_{\mathbf{m}_{jl}} \mathbb{E}_{q_t}[\bar{\ell}(\boldsymbol{\theta})] &\approx \nabla_{\boldsymbol{\theta}_{jl}} \bar{\ell}(\tilde{\boldsymbol{\theta}}) \Big|_{\tilde{\boldsymbol{\theta}} = \widetilde{\boldsymbol{m}}_t}, \\
2 \nabla_{\mathbf{S}^{-1}_{jl}} \mathbb{E}_{q_t} \left[\bar{\ell}(\boldsymbol{\theta}) \right] = \nabla^2_{\mathbf{m}_{jl}\mathbf{m}^\top_{jl}} \mathbb{E}_{q_t} \left[\bar{\ell}(\boldsymbol{\theta}) \right] &\approx \nabla^2_{\tilde{\boldsymbol{\theta}}_{jl}\tilde{\boldsymbol{\theta}}^\top_{jl}} \bar{\ell}(\tilde{\boldsymbol{\theta}}) \Big|_{\tilde{\boldsymbol{\theta}} = \widetilde{\boldsymbol{m}}_t}.
\end{aligned}
\tag{84}
$$

We will use these gradients for the loss term in Eqs. 82 and 83.

Gradients of $\log q(\boldsymbol{\theta}_{jl})$ only depends on the variables involved in $q(\boldsymbol{\theta}_{jl})$, and we can use the delta method similar to Eq. 33 in Sec. 3. We make the assumption that the two mixture components are "not close" which is reasonable with dropout. With this assumption then the gradient respect to $\boldsymbol{m}_{jl}$ is approximately zero

$$
\begin{aligned}
\nabla_{\mathbf{m}_{jl}} \mathbb{E}_q \left[\log q(\boldsymbol{\theta}_{jl})\right] &= \nabla_{\mathbf{m}_{jl}} \int d\boldsymbol{\theta}_{jl} \log q(\boldsymbol{\theta}_{jl}) \left[\pi_1 \mathcal{N}(\boldsymbol{\theta}_{jl}|\boldsymbol{m}_{jl}, \boldsymbol{S}_{jl}^{-1}) + (1 - \pi_1)\mathcal{N}(\boldsymbol{\theta}_{jl}|\mathbf{0}, s_0^{-1}\boldsymbol{I}_{n_l})\right] \\
&= \pi_1 \nabla_{\mathbf{m}_{jl}} \int d\boldsymbol{\theta}_{jl} \log q(\boldsymbol{\theta}_{jl}) \mathcal{N}(\boldsymbol{\theta}_{jl}|\boldsymbol{m}_{jl}, \boldsymbol{S}_{jl}^{-1}) \\
&= \pi_1 \mathbb{E}_{\mathcal{N}(\boldsymbol{\theta}_{jl}|\mathbf{m}_{jl,t}, \mathbf{S}_{jl,t}^{-1})} \left[\nabla_{\theta_{jl}} \log q(\boldsymbol{\theta}_{jl})\right] \quad \text{(Using Bonnet's Theorem)} \\
&\approx \pi_1 \nabla_{\mathbf{m}_{jl}} \log q(\boldsymbol{m}_{jl,t}) \text{ (Using the Delta approximation)} \\
&\approx 0.
\end{aligned}
\tag{85}
$$

For the gradient with respect to $\boldsymbol{S}^{-1}$ we get

$$
\begin{aligned}
2\nabla_{\mathbf{S}_{jl}^{-1}} \mathbb{E}_q \left[\log q(\boldsymbol{\theta}_{jl})\right] &= \nabla^2_{\mathbf{m}_{jl}\mathbf{m}_{jl}^\top} \mathbb{E}_q \left[\log q(\boldsymbol{\theta}_{jl})\right] \\
&= \nabla_{\mathbf{m}_{jl}} \left[\pi_1 \mathbb{E}_{\mathcal{N}(\boldsymbol{\theta}_{jl}|\mathbf{m}_{jl,t}, \mathbf{S}_{jl,t}^{-1})} \left[\nabla_{\theta_{jl}} \log q(\boldsymbol{\theta}_{jl})\right]\right] \quad \text{(using first derivative wrt } \mathbf{m}_{jl}) \\
&= \pi_1 \mathbb{E}_{\mathcal{N}(\boldsymbol{\theta}_{jl}|\mathbf{m}_{jl,t}, \mathbf{S}_{jl,t}^{-1})} \left[\nabla^2_{\theta_{jl}\theta_{jl}^\top} \log q(\boldsymbol{\theta}_{jl})\right] \quad \text{(Using Bonnet's theorem)} \\
&\approx \pi_1 \nabla^2_{\mathbf{m}_{jl}\mathbf{m}_{jl}^\top} \log q(\boldsymbol{m}_{jl,t}) \text{ (Using Delta approximation)} \\
&\approx -\pi_1 \boldsymbol{S}_{jl,t},
\end{aligned}
\tag{86}
$$

Using the gradient approximations Eqs. 84 to 86 in Eqs. 82 and 83, we get the

$$
\boldsymbol{m}_{jl,t+1} \leftarrow \boldsymbol{m}_{jl,t} - \frac{\rho}{\pi_1} \boldsymbol{S}_{jl,t+1}^{-1} \nabla_{\mathbf{m}_{jl}} \left\{\nabla_{\mathbf{m}_{jl}} \bar{\ell}(\widetilde{\boldsymbol{m}}_t) + 0\right\},
$$

$$
\boldsymbol{S}_{jl,t+1} \leftarrow \boldsymbol{S}_{jl,t+1} + \frac{\rho}{\pi_1} \nabla_{\mathbf{S}_{jl}^{-1}} \left\{\nabla^2_{\mathbf{m}_{jl}\mathbf{m}_{jl}^\top} \bar{\ell}(\widetilde{\boldsymbol{m}}_t) - \pi_1 \boldsymbol{S}_{jl,t}\right\}.
$$

which reduces to the updates Eqs. 42 and 43.