Learning-Algorithms from Bayesian Principles

Mohammad Emtiyaz Khan
RIKEN center for Advanced Intelligence Project
Tokyo, Japan

Haavard Rue
CEMSE Division
King Abdullah University of Science and Technology
Thuwal, Saudi Arabia

August 16, 2020
Version 0.7

Abstract

Machine-learning algorithms are commonly derived using ideas from optimization and statistics, followed by an extensive empirical efforts to make them practical as there is a lack of underlying principles to guide this process. In this paper, we present a learning rule derived from Bayesian principles, which enables us to connect a wide-variety of learning algorithms. Using this rule, we can derive a wide-range of learning-algorithms in fields such as probabilistic graphical models, continuous optimization, and deep learning. This includes classical algorithms such as least-squares, Newton's method, and Kalman filter, as well as modern deep-learning algorithms such as stochastic-gradient descent, RMSprop and Adam. Overall, we show that Bayesian principles not only unify, generalize, and improve existing learning-algorithms, but also help us design new ones. [**This is a working draft and a work in progress**]

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 recongnize objects, we collect a dataset $\mathcal{D} := \{\mathbf{x}_i, y_i\}_{i=1}^N$ with images $\mathbf{x}_i \in \mathbb{R}^D$ and object labels $y_i \in \{1, 2, 3, \dots, K\}$, and learn a model $f_{\mathbf{w}}(\mathbf{x})$ with parameters $\mathbf{w} \in \mathbb{R}^P$ to predict the labels for a new image $\mathbf{x}_*$. *Learning algorithms* are often employed to estimate the mode parameters $\mathbf{w}$ using the *princinple of trial-and-error*. For example, the well-known *Empricial Risk Minimization (ERM)* approach uses the following criteria:

$$\min_{\mathbf{w}} \bar{\ell}(\mathbf{w}) := \sum_{i=1}^{N} \ell(y_i, \mathbf{f}_{\mathbf{w}}(\mathbf{x}_i)) + \mathcal{R}(\mathbf{w}). \tag{1}$$

where $\ell(y, \mathbf{f}_{\mathbf{w}}(\mathbf{x}))$ is a loss function that encourages the model to predict well and $\mathcal{R}(\mathbf{w})$ is a regularizer that prevents it from overfitting. A wide-variety of such learning-algorithms exist in the literature that minimize different types of loss functions, e.g., least-squares, Kalman filters, stochastic-gradient descent,

and Newton's method etc. Design of such algorithms plays a key role behind the recent success of modern machine-learning algorithms.

Often, learning-algorithms are derived by borrowing ideas from a diverse set of fields, such as, statistics, optimization, statistical physics, information theory, and computer science. Borrowing and combinging ideas from such diverse fields is a strong point of the ML community, and has given rise to a wide-variety of learning algorithm. This diversity, however, comes with a price. The very nature of machine learning is data centric and the charasteristics of data is problem dependent. A successful application of learning algorithms, in many situations, becomes an art involving many tricks-of-the-trade which need subtle tuning for every specific pair of the problem and model. Discovery of such tricks is a costly process and typically requires extensive empirical investigations. For every new algorithm, researchers are often forced to reinvent these tricks from scratch. Deployment of these new algorithms also results in a huge change in practice, e.g., changes during data collection and preprocessing, as well as in codebases. Such costs discourage algorithmic innovations and forces researchers and practitioners to resort to simpler approximations, even when the approximations are suboptimal, brittle, and theoretically unsound. Currently, there is a lack of common principles that can guide the design and tuning process, and it is our opinion that this widens the gap between theory and practice, which in the long run would not be benefitial for ML research community.

1.2 Learning-algorithms from Bayesian principles

The goal of this paper is to introduce Bayesian principles as a common set of principle to derive a wide-variety of learning algorithms. Our key idea is to use the following Bayesian reformulation of the learning problem such as (1) where the optimization is over distributions $q(\mathbf{w})$ instead of $\mathbf{w}$:

$$\min_{q(\mathbf{w})\in\mathcal{Q}} \mathcal{L}(q) := \mathbb{E}_{q(\mathbf{w})}\left[\sum_{i=1}^{N} \ell(\mathbf{y}_i, \mathbf{f}_{\mathbf{w}}(\mathbf{x}_i))\right] + \mathbb{D}_{KL}[q(\mathbf{w}) \,\|\, p(\mathbf{w})], \tag{2}$$

where $\mathcal{Q} \subseteq \mathcal{P}$ is a set of probability distribution, $p(\mathbf{w}) \propto \exp\left[-\mathcal{R}(\mathbf{w})\right]$ is a prior distribution derived from the regularizer $\mathcal{R}(\mathbf{w})$ and assumed to be well-defined, and $\mathbb{D}_{KL}[\,\|\,]$ is the Kullback-Leibler divergence. We refer to this problem as the *Bayesian learning problem*. The solutions obtained for this problem are typically different from the ones obtained with the ERM approach (1), however our goal is to show that with a specific choice of an algorithm we can recover the ERM solutions in many cases.

The Bayesian learning problem is a generalization of Bayesian inference which requires a probabilistic model. When the loss corresponds to the log of a probability distribution $\ell(\mathbf{y}, f_w(\mathbf{x})) := -\log p(\mathbf{y}|f_w(\mathbf{x}))$, then the the solution of (2) is equal to the posterior distribution of a probabilistic model with the likelihood $p(\mathbf{y}|f_w(\mathbf{x}))$ and prior $p(\mathbf{w})$; see Appendix A for a proof. The objective (2) generalizes Bayesian inference to a general class of loss functions and prior distributions (Zellner, 1988; Bissiri et al., 2016). In such cases, when $\mathcal{Q} \equiv \mathcal{P}$, the optimum is the following the following *generalized* posterior, which is sometime refered to as the *Gibbs posterior* (Geman and Geman, 1984; Jiang and Tanner, 2008):

$$q_{\text{Gibbs}}(\mathbf{w}) \propto p(\mathbf{w}) \prod_{i=1}^{N} e^{-\ell(y_i, \mathbf{f}_{\mathbf{w}}(\mathbf{x}_i))} \tag{3}$$

We show that by restricting the set $\mathcal{Q} \subset \mathcal{P}$ to a carefully chosen approximation, we can obtain a variety of existing learning algorithms from the above Bayesian principle.

Throughout the paper, we will focus on cases where $\mathcal{Q}$ is the set of a class of *minimal* exponential family approximations:

$$q(\mathbf{w}) := h(\mathbf{w}) \exp\left[\langle \boldsymbol{\lambda}, \mathbf{T}(\mathbf{w})\rangle - A(\boldsymbol{\lambda})\right] \tag{4}$$

where $\boldsymbol{\lambda} \in \Omega \subset \mathbb{R}^M$ is a natural parameter with Ω being the set of valid[1] natural parameterization, $\mathbf{T}(\mathbf{w})$ is a vector containing sufficient statistics, $\langle \cdot, \cdot \rangle$ is an inner product, $A(\boldsymbol{\lambda})$ is the log-partition function, and $h(\mathbf{w})$ is the base measure. We assume that the family is regular, i.e., the set of constraints Ω is an open set. More details on the *minimality* condition is given in Section 2. Our result indeed holds in more general setting where $\mathcal{Q}$ can be a class of mixture distribution as well as kernel exponential family. These extensions are also discussed in Section **??**. Since minimal exponential families are sufficient to demonstrate the generality of our Bayesian principle, most of our examples based on them.

1.3 Bayesian learning rule

Given a learning problem, such as (1), and an exponential-family approximation, such as (4), but with a constant base measure $h(\mathbf{w}) \equiv 1$, a variety of learning algorithms can be obtained as special cases of the following update, which we call the *Bayesian learning rule*:

$$\boldsymbol{\lambda}_{t+1} \leftarrow (1-\rho_t)\boldsymbol{\lambda}_t - \rho_t \nabla_{\boldsymbol{\mu}} \mathbb{E}_{q_t(\mathbf{w})}\left[\bar{\ell}(\mathbf{w})\right], \tag{5}$$

where t denotes the iteration, $q_t(\mathbf{w})$ denotes $q(\mathbf{w})$ with natural parameters $\boldsymbol{\lambda}_t$, $\rho_t > 0$ is a sequence of scalar step-sizes (learning rates), $\boldsymbol{\mu} = \boldsymbol{\mu}(\boldsymbol{\lambda}) := \mathbb{E}_{q(\mathbf{w})}[\mathbf{T}(\mathbf{w})]$ is the *expectation parameter* of the exponential family $q(\mathbf{w})$, and the derivative $\nabla_{\boldsymbol{\mu}} \mathbb{E}_{q_t(\mathbf{w})}\left[\bar{\ell}(\mathbf{w})\right]$ is taken at $\boldsymbol{\mu} = \boldsymbol{\mu}_t$ which is the expectation parameter of $q_t(\mathbf{w})$. The above learning rule updates the natural parameter while computing the gradient with respect to the expectation parameter. This is valid since the expectation parameter $\boldsymbol{\mu}$ is a function of $\boldsymbol{\lambda}$, and for a class of exponential-family distribution there is a one-to-one mapping between the two; see Section 2.1. We will assume that the update $\boldsymbol{\lambda}$ remains within Ω which is the space of valid natural parameters. A line-search strategy, a projection step, or approximations might be necessary in practice to ensure this (see Section (4) for an example).

The Bayesian learning rule optimizes the objective (2) and is derived by using techniques from information geometry. The rule is originally proposed in (Khan and Lin, 2017) for nonconjugate variational inference, where it is referred to as the *conjugate-computation variational inference* algorithm. Our goal is to extend its application beyond variational inference and establish it as a generic learning rule that can be applied to a wide-variety of learning problems. To reflect this fact, we rename the algorithm as the Bayesian learning rule.

A full derivation of the rule, based on the work of Khan and Lin (2017), is given in Section 2. Specifically, the rule is a natural-gradient algorithm which uses the Fisher information matrix of $q(\mathbf{w})$ to precondition the gradient with respect to $\boldsymbol{\lambda}$. Surprisingly, this preconditioned natural-gradient is equal to the gradient with respect to the expectation parameter, and in many cases is easy to compute or approximate. The rule is also a specific instance of a mirror-descent algorithm where the geometry of the Bregman divergence is dictated by the exponential-family approximation. This connection is discussed in Section **??**.

The rule is not restricted to exponential-family approximation of the form (4), and a similar version can be derived for other approximations. For example, when the base measure $h(\mathbf{w})$ in (4) is not a constant, the rule can still be applied although it takes a slightly different form with an additional term added to account for the base measure:

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

Detailed derivation of this case is given in Section 2. A similar update holds for mixture of exponential-family and kernel exponential-family approximations, as discussed in Section 2.

[1] More precisely, Ω is the relative interior of the set of all $\boldsymbol{\lambda}$ for which the partition function $A(\boldsymbol{\lambda}) < \infty$.

Even though the algorithm is designed to optimize the Bayesian learning problem, with an appropriate choice of the approximation $q(\mathbf{w})$, it reduces to learning algorithms that optimize the ERM loss (1). This is a surprising and remarkable result, and similar results of this type show that the Bayesian principle indeed is a fundamental principle behind a variety of learning algorithms. Not only this, but the complexity of the algorithm is directly related to the complexity of the approximation, e.g., by choosing a more flexible approximation, we can obtain algorithms that go beyond second-order methods, such as Newton's method. We will now give several examples to demonstrate these points.

1.4 Illustrative example (least-squares)

The simplest learning problem to illustrate this point is linear regression where the Bayesian learning rule reduces to ridge regression. The loss function is given as follows:

$$\bar{\ell}(\mathbf{w}) := \tfrac{1}{2}\left[\sum_{i=1}^{N}(y_i - \mathbf{x}_i^\top \mathbf{w})^2 + \delta \mathbf{w}^\top \mathbf{w}\right], \tag{7}$$

with $\delta > 0$ as a scalar regularization parameter. The minimizer is $\mathbf{w}_* := (\mathbf{X}^\top\mathbf{X} + \delta\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{y}$, where $\mathbf{X}$ is a $N \times D$ matrix with $\mathbf{x}_i^\top$ as rows and $\mathbf{y}$ is the vector of all y_i. We can derive this from Bayesian principle by choosing $q(\mathbf{w}) := \mathcal{N}(\mathbf{w}|\mathbf{m}, \mathbf{S}^{-1})$ with mean $\mathbf{m}$ and precision matrix $\mathbf{S}$ (i.e., inverse of the covariance) and estimate its parameters using Bayesian learning rule.

To apply the rule, we must first identify the natural and expectation parameters of the Gaussian approxiation. This can be done by writing the Gaussian in an exponential-form:

$$\begin{aligned}\mathcal{N}(\mathbf{w}|\mathbf{m}, \mathbf{S}^{-1}) &:= |2\pi\mathbf{S}|^{-1/2}\exp\left[-\tfrac{1}{2}(\mathbf{w}-\mathbf{m})^T\mathbf{S}(\mathbf{w}-\mathbf{m})\right] &(8)\\ &= \exp\left[\mathbf{m}^\top\mathbf{S}\mathbf{w} - \tfrac{1}{2}\mathrm{Tr}(\mathbf{S}\mathbf{w}\mathbf{w}^\top) - \tfrac{1}{2}\left(\mathbf{m}^\top\mathbf{m} + \log|2\pi\mathbf{S}|\right)\right]. &(9)\end{aligned}$$

Comparing this expression with (4), we can determine the natural parameters, sufficient statistics, and corresponding expectation parameters. They come in pairs as shown below:

$$\begin{array}{llllll} \boldsymbol{\lambda}^{(1)} & := \mathbf{S}\mathbf{m}, & \mathbf{T}^{(1)}(\mathbf{w}) & := \mathbf{w}, & \boldsymbol{\mu}^{(1)} & = \mathbb{E}_{q(\mathbf{w})}[\mathbf{w}] \quad = \mathbf{m}, \\ \boldsymbol{\lambda}^{(2)} & := -\tfrac{1}{2}\mathbf{S}, & \mathbf{T}^{(2)}(\mathbf{w}) & := \mathbf{w}\mathbf{w}^\top, & \boldsymbol{\mu}^{(2)} & = \mathbb{E}_{q(\mathbf{w})}[\mathbf{w}\mathbf{w}^\top] \quad = \mathbf{S}^{-1} + \mathbf{m}\mathbf{m}^\top, \end{array} \tag{10}$$

The inner product for the first set of natural parameter $\boldsymbol{\lambda}^{(1)}$ and sufficient statistics $\mathbf{T}^{(1)}(\mathbf{w})$ is the dop product, while for the second set is the trace operator. It is also easy to establish that the mapping between $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ is one-to-one, which is essential for the Bayesian learning rule to be valid.

The Bayesian learning rule requires the gradient of $\mathbb{E}_{q(\mathbf{w})}[\bar{\ell}(\mathbf{w})]$ with respect to $\boldsymbol{\mu}$. This can be easily obtained by noticing that the expectated loss is in fact linear in $\boldsymbol{\mu}$, as shown below:

$$\begin{aligned}\mathbb{E}_{q(\mathbf{w})}\left[\bar{\ell}(\mathbf{w})\right] &= \mathbb{E}_{q(\mathbf{w})}\left[-\mathbf{y}^\top\mathbf{X}\mathbf{w} + \tfrac{1}{2}\mathbf{w}^\top\left(\mathbf{X}\mathbf{X}^\top + \delta\mathbf{I}\right)\mathbf{w}\right] + \mathrm{cnst} &(11)\\ &= -\mathbf{y}^\top\mathbf{X}\underbrace{\mathbb{E}_{q(\mathbf{w})}(\mathbf{w})}_{\boldsymbol{\mu}^{(1)}} + \tfrac{1}{2}\mathrm{Tr}\Big[\left(\mathbf{X}^\top\mathbf{X} + \delta\mathbf{I}\right)\underbrace{\mathbb{E}_{q(\mathbf{w})}\left(\mathbf{w}\mathbf{w}^\top\right)}_{\boldsymbol{\mu}^{(2)}}\Big] + \mathrm{cnst}. &(12)\end{aligned}$$

Therefore the gradients are simply given as follows:

$$\begin{aligned}\nabla_{\boldsymbol{\mu}^{(1)}}\mathbb{E}_{q(\mathbf{w})}\left[\bar{\ell}(\mathbf{w})\right] &= -\mathbf{X}^\top\mathbf{y} &(13)\\ \nabla_{\boldsymbol{\mu}^{(2)}}\mathbb{E}_{q(\mathbf{w})}\left[\bar{\ell}(\mathbf{w})\right] &= \tfrac{1}{2}\left(\mathbf{X}^\top\mathbf{X} + \delta\mathbf{I}\right). &(14)\end{aligned}$$

The right-hand side above already contains the components requires to obtain $\mathbf{w}_*$. Since the gradient does not depend on $\boldsymbol{\lambda}$, we can use a learning-rate $\rho_t = 1$ and set $\boldsymbol{\lambda}_* = -\nabla_{\boldsymbol{\mu}} \mathbb{E}_{q(\mathbf{w})}\left[\bar{\ell}(\mathbf{w})\right]$ to get the following solution:

$$\mathbf{m}_* := \left(\mathbf{X}^\top \mathbf{X} + \delta \mathbf{I}\right)^{-1} \mathbf{X}^\top \mathbf{y}, \qquad \mathbf{S}_* := \left(\mathbf{X}^\top \mathbf{X} + \delta \mathbf{I}\right), \tag{15}$$

As promised, the mean $\mathbf{m}_*$ is equal to $\mathbf{w}_*$ recovering the ridge regression solution.

This simple examples illustrates the main idea of the Bayesian learning rule. The loss (7) is quadratic and therefore is linear in sufficient statistics of q (which contains both linear and quadratic terms in $\mathbf{w}$). Due to this, the expected loss is linear in $\boldsymbol{\mu}$. Due to this, the gradient $\nabla_{\boldsymbol{\mu}} \mathbb{E}_{q(\mathbf{w})}\left[\bar{\ell}(\mathbf{w})\right]$ is independent of $\boldsymbol{\lambda}$, and the fixed-point of the Bayesian learning rule can be obtained in closed-form by setting $\rho_t = 1$. The Bayesian learning rule therefore recovers recovers the minimizer of the ERM loss (7).

This example illustrate the derivation of conjugate Bayesian inference from Bayesian learning rule, which is discussed in detail in Section **??**. In such cases, the loss functions correspond to the log-likelihood of a *conjugate* exponential-family model, and the loss can be written as $\bar{\ell}(\mathbf{w}) = \langle \boldsymbol{\lambda}_{\mathcal{D}}, \mathbf{T}(\mathbf{w}) \rangle$ for some $\boldsymbol{\lambda}_{\mathcal{D}}$ which only depends on the dataset $\mathcal{D}$ and is independent of $\mathbf{w}$. For example, for the least-square case, $\boldsymbol{\lambda}_{\mathcal{D}}$ consists of the following two parameters:

$$\boldsymbol{\lambda}_{\mathcal{D}}^{(1)} := -\mathbf{X}^\top \mathbf{y}, \qquad \boldsymbol{\lambda}_{\mathcal{D}}^{(2)} := \tfrac{1}{2}\left(\mathbf{X}^\top \mathbf{X} + \delta \mathbf{I}\right). \tag{16}$$

The solution is then recovered by setting $\boldsymbol{\lambda}_* = \boldsymbol{\lambda}_{\mathcal{D}}$.

As discussed in Section 5, the classical *forward-backward* algorithms (Koller and Friedman, 2009), such as Kalman filters and hidden Markov model, can all be seen as specific instances of the Bayesian leanring rule. They all solve an equation of the type $\boldsymbol{\lambda}_* = \boldsymbol{\lambda}_{\mathcal{D}}$ where the choices of $\boldsymbol{\lambda}$ and $\boldsymbol{\lambda}_{\mathcal{D}}$ are governed by the choices of approximation and the probabilistic model respectively. The classical expectation-maximization algorithm can also be derived from this procedure. Not only this, the solutions of approximate Bayesian inference algorithms, such as Laplace approximation and variational inference, can also be recovered with the same rule. This is discussed in Section 5. In summary, many learning-algorithms used in Bayesian inference can be derived from the Bayesian learning rule.

1.5 Illustrative example (gradient-descent and Newton's method)

Gradient-based algorithms, such as gradient descent (GD) and Newton's method, are the most popular algorithms for continuous, unconstrained optimization problems. Such algorithms can be derived from the Bayesian learning rule by choosing $q(\mathbf{w})$ to be Gaussian. We will now demonstrate this.

The GD update takes the following form:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \rho_t \nabla_{\mathbf{w}} \bar{\ell}(\mathbf{w}_t) \tag{17}$$

To derive this algorithm from Bayesian learning rule, we let $q(\mathbf{w}) := \mathcal{N}(\mathbf{m}, \mathbf{I})$ where $\mathbf{m}$ is the parameter that needs to be estimate and the covariance is set to $\mathbf{I}$. The choice of covariance as well as prior can be arbitrary and does not change the form of the update. A more general derivation is given in Appendix **??**, and here we consider a simpler case to demonstrate the derivation.

To apply Bayesian leanring rule, we first need to express the Gaussian approximation $q(\mathbf{w}) := \mathcal{N}(\mathbf{m}, \mathbf{I})$ in the exponential-form as shown below:

$$\mathcal{N}(\mathbf{w}|\mathbf{m}, \mathbf{I}) := (2\pi)^{-P/2} e^{-\frac{1}{2}(\mathbf{w}-\mathbf{m})^T(\mathbf{w}-\mathbf{m})} = (2\pi)^{-P/2} e^{-\frac{1}{2}\mathbf{w}^\top \mathbf{w}} \exp\left(\mathbf{m}^\top \mathbf{w} - \tfrac{1}{2}\mathbf{m}^\top \mathbf{m}\right) \tag{18}$$

Comparing this expression with (4), we clearly see that both the natural and expectation parameters are equal to the mean $\mathbf{m}$, i.e., $\boldsymbol{\lambda} := \mathbf{m}$ and $\boldsymbol{\mu} := \mathbb{E}_{q(\mathbf{w})}(\mathbf{w}) = \mathbf{m}$, and the base measure is $h(\mathbf{w}) := (2\pi)^{-P/2} \exp\left(-\frac{1}{2}\mathbf{w}^\top \mathbf{w}\right)$, which is not equal to 1. The inner-product between $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ is the dot product. Our goal is to estimate the unknown parameter $\boldsymbol{\lambda}$.

Since the base measure is not equal to 1, we need to use the update (6) as discussed in Section 1.3:

$$\boldsymbol{\lambda}_{t+1} \leftarrow (1-\rho_t)\boldsymbol{\lambda}_t - \rho_t \nabla_{\boldsymbol{\mu}} \mathbb{E}_{q_t(\mathbf{w})}\left[\bar{\ell}(\mathbf{w}) + \log h(\mathbf{w})\right]. \tag{19}$$

By simply substituting the definition of $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ and noting that the derivative of $\mathbb{E}_{q(\mathbf{w})}[\log h(\mathbf{w})]$ is equal to $-\mathbf{m}$, we get the following update:

$$\mathbf{m}_{t+1} \leftarrow \mathbf{m}_t - \rho_t \nabla_{\mathbf{m}} \mathbb{E}_{q_t(\mathbf{w})}\left[\bar{\ell}(\mathbf{w})\right], \tag{20}$$

which is similar to the GD update but now the gradients are obtained at the expectation $\mathbb{E}_{q(\mathbf{w})}\left[\bar{\ell}(\mathbf{w})\right]$ instead of the loss itself. To recover the GD update, we can approximate the gradient at the mean $\mathbf{m}$ as shown below:

$$\nabla_{\mathbf{m}} \mathbb{E}_{q(\mathbf{w})}\left[\bar{\ell}(\mathbf{w})\right] = \mathbb{E}_{q(\mathbf{w})}\left[\nabla_{\mathbf{w}} \bar{\ell}(\mathbf{w})\right] \approx \nabla_{\mathbf{m}} \bar{\ell}(\mathbf{m}), \tag{21}$$

where the first step is obtained using Bonnet's theorem Bonnet (1964); Opper and Archambeau (2009); Rezende et al. (2014), and the second step is using the delta approximation (Dorfman, 1938; Ver Hoef, 2012) where we approximate the expectation of a function by the value at the mean $\mathbf{m}$. The latter implies that we resort to a greedy approximation that does not employ the averaging property of the Bayesian principle. Using this approximation and then denoting $\mathbf{m}_t$ as the iterate $\mathbf{w}_t$, we recover the gradient descent update (17). This derivation essentially shows that, by giving away the averaging property of the Bayesian principles, i.e., by approximating the expectation in (21) by a point estimate, we obtain a non-Bayesian algorithm that converges to a minimum of $\bar{\ell}(\mathbf{w})$.

The above example illustrates one of the key ideas behind the Bayesian learning rule – the gradient $\nabla_{\boldsymbol{\mu}} \mathbb{E}_{q(\mathbf{w})}[\bar{\ell}(\mathbf{w})]$ enables us to probe gradient information of the $\bar{\ell}(\mathbf{w})$. The gradient with respect to the mean $\mathbf{m}$ of the Gaussian, we get first-order information about the $\bar{\ell}(\mathbf{w})$. In general, by increasing the complexity of the approximating distribution, we can dig-out higher-order information about $\bar{\ell}(\mathbf{w})$. In fact, as we discuss in Section 3, if we use a Gaussian approximation $\mathcal{N}(\mathbf{w}|\mathbf{m}, \mathbf{S}^{-1})$ and estimate both its mean $\mathbf{m}$ and precision $\mathbf{S}$ (i.e., inverse of the covariance), then we recover an online version of Newton's method:

$$\mathbf{m}_{t+1} \leftarrow \mathbf{m}_t - \rho_t \mathbf{S}_{t+1}^{-1} \mathbb{E}_{q_t(\mathbf{w})}\left[\nabla_{\mathbf{w}} \bar{\ell}(\mathbf{w})\right], \qquad \text{where } \mathbf{S}_{t+1} \leftarrow (1-\rho_t)\mathbf{S}_t + \rho_t \mathbb{E}_{q_t(\mathbf{w})}\left[\nabla^2_{\mathbf{ww}} \bar{\ell}(\mathbf{w})\right]. \tag{22}$$

and $q_t(\mathbf{w}) := \mathcal{N}(\mathbf{w}|\mathbf{m}_t, \mathbf{S}_t)$. This algorithm is derived by Khan et al. (2018) where the updates use gradients and Hessians evaluated at samples from q_t. By making the approximation (21) over both Hessian and Gradients: $\mathbb{E}_{q(\mathbf{w})}\left[\nabla_{\mathbf{w}} \bar{\ell}(\mathbf{w})\right] \approx \nabla_{\mathbf{m}} \bar{\ell}(\mathbf{m})$ and $\mathbb{E}_{q(\mathbf{w})}\left[\nabla^2_{\mathbf{ww}} \bar{\ell}(\mathbf{w})\right] \approx \nabla^2_{\mathbf{mm}} \bar{\ell}(\mathbf{m})$, and then denoting $\mathbf{m}_t$ by $\mathbf{w}_t$, we get the following update that resorts to a greedy approximation, rather than the Bayesian one:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \rho_t \mathbf{S}_{t+1}^{-1}\left[\nabla_{\mathbf{w}} \bar{\ell}(\mathbf{w}_t)\right], \qquad \text{where } \mathbf{S}_{t+1} \leftarrow (1-\rho_t)\mathbf{S}_t + \rho_t \nabla^2_{\mathbf{ww}} \bar{\ell}(\mathbf{w}_t). \tag{23}$$

The precision $\mathbf{S}_t$ contains a moving average of the past Hessians. Newton's method can be obtained as a special case where the learning-rate is set to be 1, which corresponds to a perfect Newton step:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \left[\nabla^2_{\mathbf{ww}} \bar{\ell}(\mathbf{w}_t)\right]^{-1}\left[\nabla_{\mathbf{w}} \bar{\ell}(\mathbf{w}_t)\right], \tag{24}$$

This is valid when the loss is strongly convex and the algorithm is initialized close to the solution.

In situations, when it is not possible to compute exact gradients, it is typical to use stochastic gradients, e.g., in deep learning. In such cases, we should resort to a step size of less than 1 since we do not have correct second-order information. Application of Bayesian learning rule to such cases results in an algorithm similar to the popular algorithms used in deep learning, e.g., RMSprop (Tieleman and Hinton, 2012) and Adam (Kingma and Ba, 2015). We discuss such variants in Section 4, based on the work of Khan et al. (2018).

The updates are also closely related to methods used for online learning methods, e.g., the online gradient-descent (OGD) and online Newton step (ONS) (Hazan et al., 2007). The Bayesian learning rule generalizes such methods and connects them to other methods used in online learning, such as the Follow the Regularized Leader method, multiplicative weight update (Vovk, 1990; Warmuth, 2010), and the Follow the Perturbed Leader method (Kalai and Vempala, 2005). Such connections are are obtained by using a general version of mirror descent McMahan (2011).

Furthermore, when we increase the complexity of the approximation from a unimodal distribution to a multimodal one, we can derive more complex optimization algorithms. Lin et al. (2019) consider mixture of Gaussian distributions to derive many variants of the Newton's method. In particular, when we choose finite mixture of Gaussians, a variant of the Bayesian learning rule results in an *ensemble* method where multiple chains of Newton-like updates are run in parallel. Bayesian principles in this case enable a targeted exploration of the optimization landscape – the KL divergence term in (2) forces the chains to be different from each other.

The above two examples illsutrate a strong link between the complexity of the approximation $q(\mathbf{w})$ and that of the learning algorithms. The Gaussian approximation with unknown mean parameter results in a first-order method, while adding the covariance parameter results in a second-order method. By increasing the number of parameters of $q(\mathbf{w})$ or equivalently a richer sufficient statistics, we can therefore obtain learning-algorithm that go beyond traditional second-order methods, such as Newton's method. This is detailed in Section 3 in the context of continuous optimization algorithms. In conclusion, Bayesian principles carry an immense potential to enable researchers invent new optimization algorithms that go beyond first and second-order algorithms.

1.6 Outline of the Rest of the Paper

We start with a detailed derivation of the Bayesian learning rule using natural-gradient algorithms in Section 2. We also discuss an alternate derivation based on mirror-descent which provides the alternate and more complete perspective using the duality of exponential family. Section 3, we discuss derivation of Newton's method, and use it to illustrate the fundamental differences between the optimization and Bayesian solutions. Implications on robustness of the Bayesian solution is illustrated through simple examples. We end the section by illustrating advantages of the Bayesian approach in the derivation of optimization methods that go beyond Newton's method. Section 4 discusses derivation of deep learning algorithms. We consider SGD, adaptive learning-rate algorithms, momentum, dropout, and algorithms for binary neural network. Section 5 (which is incomplete right now) discusses relationship to Bayesian inference and its approximations, such as variational inference and Laplace approximation.

2 Bayesian Learning Rule

2.1 Bayesian learning rule as natural-gradient descent

In this section, we derive the Bayesian learning rule using the *natural-gradient* algorithm, which differs from methods such as gradient-descent because it exploits the information geometry of a posterior distribution. Given the objective $\mathcal{L}(q)$ in (2), a standard gradient-descent algorithm performs the

following update:

$$\boldsymbol{\lambda}_{t+1} \leftarrow \boldsymbol{\lambda}_t - \rho_t \nabla_{\boldsymbol{\lambda}} \mathcal{L}(q_t), \tag{25}$$

where $\nabla_{\boldsymbol{\lambda}} \mathcal{L}(q_t)$ denotes the derivatives at $\boldsymbol{\lambda} = \boldsymbol{\lambda}_t$ which is the natural parameter of the distribution $q_t(\mathbf{w})$. Natural-gradient algorithms exploit the fact that $\boldsymbol{\lambda}_t$ parameterize a probability distributions, and therefore their updates should focus to optimize the changes directly in the space of distributions rather than space of parameters. The update (25) ignore this, which is clear when we rewrite (25) as the following equivalent maximization problem where the changes between the iterates is measured with a Euclidean distance (denoted by $\|\cdot\|_2$):

$$\boldsymbol{\lambda}_{t+1} \leftarrow \arg\min_{\boldsymbol{\lambda}} \langle \nabla_{\boldsymbol{\lambda}} \mathcal{L}(q_t), \boldsymbol{\lambda} \rangle + \frac{1}{2\rho_t} \|\boldsymbol{\lambda} - \boldsymbol{\lambda}_t\|_2^2, \tag{26}$$

The equivalence can be established by taking the derivative of the objective above. Euclidean distance in the parameter space might be a poor measure of the distance between the distributions (see Fig. 1a in Khan and Nielsen (2018) for an example).

Natural-gradient algorithms address this issue by replacing the Euclidean distance by a more appropriate distance measure, e.g., we can use the KL divergence (Martens, 2014; Pascanu and Bengio, 2013), as shown below:

$$\boldsymbol{\lambda}_{t+1} \leftarrow \arg\min_{\boldsymbol{\lambda} \in \Omega} \langle \nabla_{\boldsymbol{\lambda}} \mathcal{L}(q_t), \boldsymbol{\lambda} \rangle + \frac{1}{\rho_t} \mathbb{D}_{KL}[q(\mathbf{w}) \,\|\, q_t(\mathbf{w})], \tag{27}$$

To obtain a closed form solution, a second-order approximation of the KL divergence is utilized (Martens, 2014): $\mathbb{D}_{KL}[q(\mathbf{w}) \,\|\, q_t(\mathbf{w})] \approx (\boldsymbol{\lambda} - \boldsymbol{\lambda}_t)^\top \mathbf{F}(\boldsymbol{\lambda}_t)(\boldsymbol{\lambda} - \boldsymbol{\lambda}_t)$, where $\mathbf{F}(\boldsymbol{\lambda}) := \mathbb{E}_{q(\mathbf{w})}\left[\nabla_\lambda \log q(\mathbf{w}) \nabla_\lambda \log q(\mathbf{w})^\top\right]$ is the Fisher information matrix of $q(\mathbf{w})$. Assuming that the FIM is invertible, this approximation gives us the following update which is referred to as the natural-gradient descent algorithm (Amari, 1998):

$$\boldsymbol{\lambda}_{t+1} \leftarrow \boldsymbol{\lambda}_t - \rho_t \underbrace{\mathbf{F}(\boldsymbol{\lambda}_t)^{-1} \nabla_{\boldsymbol{\lambda}} \mathcal{L}(q_t)}_{\text{Natural Gradient}}, \tag{28}$$

The *descent direction* $\mathbf{F}(\boldsymbol{\lambda}_t)^{-1} \nabla_{\boldsymbol{\lambda}} \mathcal{L}(q_t)$ in this case is referred to as the natural-gradient, which is different from the *Euclidean* gradients which assume a Eulidean distance. The update above is slightly different from the standard derivation of natural-gradients for maximum-likelihood estimation where the loss $\mathcal{L}(q)$ is equal to the negative log-probability (Amari, 1998). In our Bayesian setup, this is not the case but the update still holds. In fact, as we discussed in Section 2.2, the second-order approximation is not required for exponential-family approximations, and the update (28) is in fact equaivalent to an update similar to (27) but performed in a *dual space*. Below, we present an easier derivation and differ the connection to duality in Section 2.2.

The Bayesian learning rule is obtained from (28) by using a special property of the exponential family: the natural-gradient is equal to the (Euclidean) gradient with respect to the expectation parameter $\boldsymbol{\mu}$:

$$\mathbf{F}(\boldsymbol{\lambda})^{-1} \nabla_{\boldsymbol{\lambda}} \mathcal{L}(q) = \nabla_{\boldsymbol{\mu}} \mathcal{L}(q) \tag{29}$$

This can be shown by simply using the chain rule as shown below:

$$\nabla_{\boldsymbol{\lambda}} \mathcal{L}(q) = [\nabla_{\boldsymbol{\lambda}} \boldsymbol{\mu}] \nabla_{\boldsymbol{\mu}} \mathcal{L}(\boldsymbol{\mu}) = \left[\nabla^2_{\boldsymbol{\lambda}\boldsymbol{\lambda}} A(\boldsymbol{\lambda})\right] \nabla_{\boldsymbol{\mu}} \mathcal{L}(\boldsymbol{\mu}) = \mathbf{F}(\boldsymbol{\lambda}) \nabla_{\boldsymbol{\lambda}} \mathcal{L}(q), \tag{30}$$

where the first equality is obtained using chain rule, the second equality is obtained using the definition of the expectation parameter: $\boldsymbol{\mu} = \nabla_{\boldsymbol{\lambda}} A(\boldsymbol{\lambda})$, and the last equality is obtained using the fact that for

exponential-family distributions the second derivative of $A(\boldsymbol{\lambda})$ is equal to the FIM, i.e., $\mathbf{F}(\boldsymbol{\lambda}) := \nabla^2_{\boldsymbol{\lambda}\boldsymbol{\lambda}} A(\boldsymbol{\lambda})$ (Nielsen and Garcia, 2009). Using this simplification, we can rewrite the update (28) as follows:

$$\boldsymbol{\lambda}_{t+1} \leftarrow \boldsymbol{\lambda}_t - \rho_t \nabla_{\boldsymbol{\mu}} \mathcal{L}(q_t). \tag{31}$$

For exponential-family approximations, we can further simplify the update. We first note that the objective $\mathcal{L}(q) = \mathbb{E}_{q(\mathbf{w})}[\bar{\ell}(\mathbf{w}) - \log q(\mathbf{w})]$ contains the entropy of the distribution $q(\mathbf{w})$ in the second term. For exponential-family distribution with constant base measure, the gradient of the entropy with respect to $\boldsymbol{\mu}$ is equal to $\boldsymbol{\lambda}$:

$$\begin{aligned}\nabla_{\boldsymbol{\mu}} \mathbb{E}_{q(\mathbf{w})}\left[\log q(\mathbf{w})\right] &= \nabla_{\boldsymbol{\mu}} \mathbb{E}_{q(\mathbf{w})}\left[\langle \boldsymbol{\lambda}, \mathbf{T}(\mathbf{w})\rangle + A(\boldsymbol{\lambda})\right] = \nabla_{\boldsymbol{\mu}}\left[\langle \boldsymbol{\lambda}, \boldsymbol{\mu}\rangle - A(\boldsymbol{\lambda})\right] \\ &= \boldsymbol{\lambda} + \langle \nabla_{\boldsymbol{\mu}} \boldsymbol{\lambda}, \nabla_{\boldsymbol{\lambda}} A(\boldsymbol{\lambda})\rangle - \nabla_{\boldsymbol{\mu}} A(\boldsymbol{\lambda}) = \boldsymbol{\lambda} + \cancel{\nabla_{\boldsymbol{\mu}} A(\boldsymbol{\lambda})} - \cancel{\nabla_{\boldsymbol{\mu}} A(\boldsymbol{\lambda})} = \boldsymbol{\lambda}\end{aligned} \tag{32}$$

where in the second line we use the fact that for exponential family $\boldsymbol{\mu} = \nabla_{\boldsymbol{\lambda}} A(\boldsymbol{\lambda})$ (Nielsen and Garcia, 2009). Substituting this in (31) gives us the Bayesian learning rule (5). When the base measure is not constant, the gradient of the entropy term contains an additional term due to the base measure, which then results in the update (6)

The derivation above assumes that the FIM of the exponential family is invertible, but under what condition is this true? A sufficient condition to ensure the invertibility is that the exponential-family is regular, i.e., Ω is an open set and that it assumes a *minimal* representation as defined[2] below (Wainwright and Jordan, 2008).

Definition 1 (Minimal EF) *A regular exponential-family representation is said to be minimal when there does not exist a nonzero vectors $\boldsymbol{\lambda}$ such that $\langle \boldsymbol{\lambda}, \mathbf{T}(\mathbf{w})\rangle$ is equal to a constant.*

This essentially means that there are no linear dependencies in the parameterization of the distribution. When such a nonzero vector exists, we can add/subtract it from $\boldsymbol{\lambda}$ without changing the distribution. Minimality ensures that this never happens and the parametrization $\boldsymbol{\lambda}$ is unique and identifiable up to multiplication with a nonsingular affine transformation. Under a minimal representation, the log-partition function $A(\boldsymbol{\lambda})$ is *strictly* convex which implies that the FIM is positive-definite (since the FIM is second derivative of $A(\boldsymbol{\lambda})$ Wainwright and Jordan (2008). For other types of representation, this may not be always true. Minimality ensures that the FIM is positive-definite and that update (28) is well defined [3].

A direct consequence of minimality is that the mapping $\boldsymbol{\mu} = \nabla_{\boldsymbol{\lambda}} A(\boldsymbol{\lambda})$ is one-to-one. Defining the space of all expectation parameters by $\mathcal{M} := \{\boldsymbol{\mu} \in \mathbb{R}^M | \exists\ q(\mathbf{w})\ s.t.\ \mathbb{E}_{q(\mathbf{w})}[\mathbf{T}(\mathbf{z})] = \boldsymbol{\mu},\ \forall \boldsymbol{\mu}\}$, the following theorem by Wainwright and Jordan (2008) states the result.

Theorem 1 *The representation (4) is minimal if and only if the mapping $\nabla_{\boldsymbol{\lambda}} A(\boldsymbol{\lambda}) : \Omega \to \mathcal{M}$ is one-to-one.*

This property ensures that the update (31) and (28) are valid and equivalent. It also gives rise a new interpretation of natural-gradient descent as mirror descent which we discuss next.

2.2 Bayesian learning rule as mirror-descent

Bayesian learning rule can also be derived using a mirror-descent formulation. This derivation provides a beautiful connection between methods used in continuous optimization and information geometry. In summary, a mirror-descent step defined in the space of $\boldsymbol{\mu}$ results in the natural-gradient step in the space

[2] A more precise and complete definition is given in Definition 1.3 of Johansen (1979).

[3] In some cases, when FIM is not invertible, it is still possible to perform NGD by, for example, ignoring the zero eigenvalues or using damping, but the simplification shown in (31) may not be possible.

of $\boldsymbol{\lambda}$ (the update (28)). This connection is a result of the *Legendre-duality* between the space of $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$, and is related to the *dually-flat* Riemannien structures employed in information geometry. The derivation also avoids the use of crude second-order approximations used in (27) to derive the natural-gradient update, and is expected to generalize to other types of divergences. Due to its importance, this section is dedicated to the derivation using mirror descent framework.

We begin by defining the duality between space of $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$. Recall that given any $\boldsymbol{\lambda}$ we can obtain the corrsepnoding expectation parameter using the following operation: $\boldsymbol{\mu} = \nabla_{\boldsymbol{\lambda}} A(\boldsymbol{\lambda})$. An alternate way to obtain this mapping is to use the *Legendre transform* at a point $\boldsymbol{\mu}$:

$$A^*(\boldsymbol{\mu}) := \sup_{\boldsymbol{\lambda}' \in \Omega} \left\{ \langle \boldsymbol{\mu}, \boldsymbol{\lambda}' \rangle - A(\boldsymbol{\lambda}') \right\}, \tag{33}$$

The solution $\boldsymbol{\lambda}$ is then obtained by taking the derivative of the objective at the right hand side and setting it to zero. This results in the mapping $\boldsymbol{\mu} = \nabla_{\boldsymbol{\lambda}} A(\boldsymbol{\lambda})$. Note that the solution can be obtained by setting the derivative to zero because $A(\boldsymbol{\lambda})$ is strictly convex and differentiable over Ω.

Theorem 1 ensures that this operation is one-to-one, therefore for every $\boldsymbol{\lambda} \in \Omega$ there exists one and only one $\boldsymbol{\mu} \in \mathcal{M}$. Similarly to the mapping from $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$, we can define the reverse mapping using the Legendre transform $A^*(\boldsymbol{\mu})$:

$$A(\boldsymbol{\lambda}) := \sup_{\boldsymbol{\mu}' \in \mathcal{M}} \left\{ \langle \boldsymbol{\mu}', \boldsymbol{\lambda} \rangle - A^*(\boldsymbol{\mu}') \right\}, \tag{34}$$

which results in the reverse map: $\boldsymbol{\lambda} = \nabla_{\boldsymbol{\mu}} A^*(\boldsymbol{\mu})$.

The expectation parameters therefore constitute a *dual* coordinate system to specify the distribution. Any exponential family distribution specified with a natural parameter $\boldsymbol{\lambda}$, as in (4), can be expressed in terms of the expectation parameter $\boldsymbol{\mu}$ (Banerjee et al., 2005). The exact expression is given below:

$$q(\mathbf{w}) := h(\mathbf{w}) \exp\left[\langle \boldsymbol{\lambda}, \mathbf{T}(\mathbf{w}) \rangle - A(\boldsymbol{\lambda})\right] = h(\mathbf{w}) \exp\left[-\mathcal{D}_{A^*}(\mathbf{T}(\mathbf{w}), \boldsymbol{\mu}) + A^*(\mathbf{T}(\mathbf{w}))\right] \tag{35}$$

where

$$\mathcal{D}_{A^*}(\boldsymbol{\mu}_1, \boldsymbol{\mu}_2) := A^*(\boldsymbol{\mu}_1) - A^*(\boldsymbol{\mu}_2) - \langle \boldsymbol{\mu}_1 - \boldsymbol{\mu}_2, \nabla_{\boldsymbol{\mu}} A^*(\boldsymbol{\mu}_2) \rangle \tag{36}$$

is the Bregman divergence defined using the function $A^*(\boldsymbol{\mu})$. Surprisingly, there is an elegant relationship between the Bregman divergence and KL divergence. Specifically,

$$\mathbb{D}_{KL}[q_1(\mathbf{w}) \,\|\, q_2(\mathbf{w})] = \mathbb{D}_{A^*}(\boldsymbol{\mu}_1 \| \boldsymbol{\mu}_2) = \mathbb{D}_A(\boldsymbol{\lambda}_2 \| \boldsymbol{\lambda}_1), \tag{37}$$

i.e., the above Bregman divergence is equal to the KL divergence between the distributions, and is also equal to the Bregman divergence defined in the dual space $\boldsymbol{\lambda}$ using the function $A(\boldsymbol{\lambda})$, but with the arguments swapped. Bregman divergence therefore provides a way for us to measure the distances in the two dual spaces using the Lengdre duals A and $A*$. The two distances are equal to the KL divergence which enables us to write natural gradient descent as a mirror descent, as we explain below.

We are now ready to state the final result:

Theorem 2 *The following* mirror descent *update in the space $\mathcal{M}$ is equivalent to the natural gradient descent* (28) *in the space of Ω,*

$$\boldsymbol{\mu}_{t+1} \leftarrow \arg\min_{\boldsymbol{\mu} \in \mathcal{M}} \langle \nabla_{\boldsymbol{\mu}} \mathcal{L}(q_t), \boldsymbol{\mu} \rangle + \frac{1}{\rho_t} \mathbb{D}_{A^*}(\boldsymbol{\mu} \| \boldsymbol{\mu}_t). \tag{38}$$

Proof We can show by simply taking the derivative of the objective and setting it to zero. First we note that the derivative of the Bregman divergence with respect to $\boldsymbol{\mu}$ is equal to the difference in the natural parameters:

$$\begin{aligned}
\nabla_{\boldsymbol{\mu}} \mathbb{D}_{A^*}(\boldsymbol{\mu} \| \boldsymbol{\mu}_t) &= \nabla_{\boldsymbol{\mu}} \left[A^*(\boldsymbol{\mu}) - A^*(\boldsymbol{\mu}_t) - \langle \boldsymbol{\mu} - \boldsymbol{\mu}_t, \nabla_{\boldsymbol{\mu}} A^*(\boldsymbol{\mu}_t) \rangle \right], && (39) \\
&= \nabla_{\boldsymbol{\mu}} A^*(\boldsymbol{\mu}) - \nabla_{\boldsymbol{\mu}} \langle \boldsymbol{\mu}, \nabla_{\boldsymbol{\mu}} A^*(\boldsymbol{\mu}_t) \rangle, && (40) \\
&= \nabla_{\boldsymbol{\mu}} A^*(\boldsymbol{\mu}) - \nabla_{\boldsymbol{\mu}} A^*(\boldsymbol{\mu}_t), && (41) \\
&= \boldsymbol{\lambda} - \boldsymbol{\lambda}_t, && (42)
\end{aligned}$$

where in the first line we use the definition (36) and in the last line we used the relationship $\boldsymbol{\lambda} = \nabla_{\boldsymbol{\mu}} A^*(\boldsymbol{\mu})$ and $\boldsymbol{\lambda}_t = \nabla_{\boldsymbol{\mu}} A^*(\boldsymbol{\mu}_t)$. Therefore, when we take the derivative of the objective (38) with respect to $\boldsymbol{\mu}$, we get: $\nabla_{\boldsymbol{\mu}} \mathcal{L}(q_t) + \frac{1}{\rho_t} (\boldsymbol{\lambda} - \boldsymbol{\lambda}_t) = 0$, which gives us the update: $\boldsymbol{\lambda}_{t+1} \leftarrow \boldsymbol{\lambda}_t - \rho \nabla_{\boldsymbol{\mu}} \mathcal{L}(q_t)$ which is then equivalent to the natural-gradient update (31). The $\boldsymbol{\lambda}_{t+1}$ can be mapped to $\boldsymbol{\mu}_{t+1}$ to get the mirror-descent step, which is valid since the mapping is one-to-one. ■

The theorem uses Lengendre-duality do derive natural-gradient descent. The converse of the above theorem is also true, i.e., a natural-gradient descent step in the space of $\mathcal{M}$ is equivalent to mirror-descent step in Ω. The proof is almost identical to the above and we omit it. To optimize the Bayesian objective (2), the natural-gradient descent in $\boldsymbol{\lambda}$-space is more convenient. This is because the gradient of $\mathcal{L}(q_t)$ is usually easier to compute since often the objective involves expectations of log-probabilities of exponential family distributios. This is easy because such expectations are linear in $\boldsymbol{\mu}$, for example, see 12 in case of least-squares.

The above result is derived using the Legendre-duality, but it has a deep connection with a duality concept in information geometry, called the *dually-flat* Riemmanien manifold (Amari, 2008). There, the two spaces of $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ are two Riemanniand manifolds whose FIMs are inverses of each other: $\mathbf{F}(\boldsymbol{\lambda}) = \mathbf{F}(\boldsymbol{\mu})^{-1}$. The divergences are related through $\mathbb{D}_A(\boldsymbol{\lambda}_1 \| \boldsymbol{\lambda}_2) = \mathbb{D}_{A^*}(\boldsymbol{\mu}_2 \| \boldsymbol{\mu}_1)$ with their arguments swapped. The dually-flat structure is not limited to KL divergences. Such structures exist for many other types of divergences. It is likley that updates similar to the Bayesian learning rule can be derived for more general divergences, which dictates a more general principles that the Bayesian one we used in (2).

A small but important technical point here is that, unlike (27), we do not have to approximate the KL divergence term (which is the Bregman divergence term here) by a second-order approximation. The result above is exact, i.e., a mirror-descent step in the $\boldsymbol{\mu}$-space is exactly equivalent to the natural-gradient step in the $\boldsymbol{\lambda}$-space. The mirror-descent framework is therefore a more natural way to derive the natural-gradient update for exponential families.

3 Optimization Algorithms from the Bayesian Learning Rule

The key idea in the derivation of optimization algorithms is choose an appropriate posterior approximation and then make use of a gradient approximation (21) at the mean of the posterior approximation. In Section 1.5, we chose $q(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{m}, \mathbf{I})$ which resulted in gradient descent, a first-order method. By increasing the complexity of the approximation, we can obtain higher-order methods. Specifically, we get a Newton-like method, when we increase the complexity of the approximation by including the covariance: $q(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{m}, \mathbf{S}^{-1})$ where $\mathbf{S}$ is the precision matrix (i.e., inverse of the covariance). This again can be shown by simply using the Bayesian learning rule (5).

3.1 Newton's Method from the Bayesian Learning Rule

To derive Newton's method, we use the definition of the natural parameter, sufficient statistics, and expectation parameters of a multivariate Gaussian distribution $\mathcal{N}(\mathbf{w}|\mathbf{m}, \mathbf{S}^{-1})$ as defined in (10). Using

these in the Bayesian learning rule 5, we get the following updates:

$$\begin{aligned}\mathbf{S}_{t+1}\mathbf{m}_{t+1} &\leftarrow (1-\rho_t)\mathbf{S}_t\mathbf{m}_t - \rho_t \nabla_{\boldsymbol{\mu}^{(1)}} \mathbb{E}_{q_t(\mathbf{w})}\left[\bar{\ell}(\mathbf{w})\right], \\ \mathbf{S}_{t+1} &\leftarrow (1-\rho_t)\mathbf{S}_t + \rho_t 2\nabla_{\boldsymbol{\mu}^{(2)}} \mathbb{E}_{q_t(\mathbf{w})}\left[\bar{\ell}(\mathbf{w})\right].\end{aligned} \tag{43}$$

For now, we assume that ρ_t are chosen appropriately to ensure that the $\mathbf{S}_t \succ 0$ which is required for a valid Gaussian distribution (see Section 3 in Khan et al. (2018).

As before, the difficulty of the update is now in computing the natural gradients. Fortunately, we can express the two gradients in terms of the gradient with respect to $\mathbf{m}$ and $\mathbf{S}^{-1}$, i.e., for a differentiable function f, we have,

$$\nabla_{\boldsymbol{\mu}^{(1)}} f = \nabla_{\mathbf{m}} f - 2\left[\nabla_{\mathbf{S}^{-1}} f\right]\mathbf{m}, \qquad \nabla_{\boldsymbol{\mu}^{(2)}} f = \nabla_{\mathbf{S}^{-1}} f. \tag{44}$$

This can be derived using the chain rule using the definition of the expectation parameters in (10) which is expressed in terms of $\mathbf{m}$ and $\mathbf{S}$ (a derivation for the scalar case is given in Appendix B.1 of Khan and Lin (2017)). The gradient with respect to $\mathbf{m}$ and $\mathbf{S}^{-1}$ is easy to compute since it can be expressed in terms of the gradient and Hessian of the function as shown below,

$$\nabla_{\mathbf{m}} \mathbb{E}_{q(\mathbf{w})}[\bar{\ell}(\mathbf{w})] = \mathbb{E}_{q(\mathbf{w})}[\nabla_{\boldsymbol{w}} \bar{\ell}(\mathbf{w})], \qquad \nabla_{\mathbf{S}^{-1}} \mathbb{E}_{q(\mathbf{w})}[\bar{\ell}(\mathbf{w})] = \tfrac{1}{2}\mathbb{E}_{q(\mathbf{w})}[\nabla^2_{\boldsymbol{ww}} \bar{\ell}(\mathbf{w})]. \tag{45}$$

These two relationships are referred to as Bonnet's and Price's theorem respectively (Bonnet, 1964; Price, 1958; Opper and Archambeau, 2009; Rezende et al., 2014). Using these relationships, we can simplify the update (43) to get a Newton-like step. The update of $\mathbf{S}$ is straightforwardly obtained by plugging in the derivative with respect to $\mathbf{S}^{-1}$ to get the derivative with respect to $\boldsymbol{\mu}^{(2)}$ which is substituted in (43) to get the following:

$$\mathbf{S}_{t+1} \leftarrow (1-\rho_t)\mathbf{S}_t + \rho_t \mathbb{E}_{q_t(\mathbf{w})}[\nabla^2_{\boldsymbol{ww}} \bar{\ell}(\mathbf{w})]. \tag{46}$$

The update of $\mathbf{m}_t$ can be obtained in a similar way after rearranging some of the terms after substitutions:

$$\begin{aligned}\mathbf{S}_{t+1}\mathbf{m}_{t+1} &\leftarrow (1-\rho_t)\mathbf{S}_t\mathbf{m}_t - \rho_t\left\{\mathbb{E}_{q_t(\mathbf{w})}[\nabla_{\boldsymbol{w}} \bar{\ell}(\mathbf{w})] - \mathbb{E}_{q_t(\mathbf{w})}[\nabla^2_{\boldsymbol{ww}} \bar{\ell}(\mathbf{w})]\mathbf{m}_t\right\}, &(47)\\ &\leftarrow \underbrace{\left[(1-\rho_t)\mathbf{S}_t^{-1} + \rho_t \mathbb{E}_{q_t(\mathbf{w})}[\nabla^2_{\boldsymbol{ww}} \bar{\ell}(\mathbf{w})]\right]}_{\mathbf{S}_{t+1}} \mathbf{m}_t - \rho_t \mathbb{E}_{q_t(\mathbf{w})}[\nabla_{\boldsymbol{w}} \bar{\ell}(\mathbf{w})], &(48)\end{aligned}$$

to get the following update:

$$\mathbf{m}_{t+1} \leftarrow \mathbf{m}_t - \rho_t \mathbf{S}_{t+1} \mathbb{E}_{q_t(\mathbf{w})}[\nabla_{\boldsymbol{w}} \bar{\ell}(\mathbf{w})], \qquad \text{where } \mathbf{S}_{t+1} \leftarrow (1-\rho_t)\mathbf{S}_t + \rho_t \mathbb{E}_{q_t(\mathbf{w})}[\nabla^2_{\boldsymbol{ww}} \bar{\ell}(\mathbf{w})]. \tag{49}$$

This is the update shown in (22). By using the delta approximation (21) over both Hessian and Gradients:

$$\mathbb{E}_{q(\mathbf{w})}\left[\nabla_{\mathbf{w}} \bar{\ell}(\mathbf{w})\right] \approx \nabla_{\mathbf{m}} \bar{\ell}(\mathbf{m}), \qquad \mathbb{E}_{q(\mathbf{w})}\left[\nabla^2_{\mathbf{ww}} \bar{\ell}(\mathbf{w})\right] \approx \nabla^2_{\mathbf{mm}} \bar{\ell}(\mathbf{m}), \tag{50}$$

and then denoting $\mathbf{m}_t$ by $\mathbf{w}_t$, we get the update shown in (23).

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \rho_t \mathbf{S}_{t+1}^{-1}\left[\nabla_{\mathbf{w}} \bar{\ell}(\mathbf{w}_t)\right], \qquad \text{where } \mathbf{S}_{t+1} \leftarrow (1-\rho_t)\mathbf{S}_t + \rho_t \nabla^2_{\mathbf{ww}} \bar{\ell}(\mathbf{w}_t). \tag{51}$$

These updates are very similar to Newton's method, but now involve an *exponential smoothing* of the Hessian over the past iterates. This step has implications for stochastic training in deep learning and has connections to deep-learning optimizers such as RMSprop and Adam. We will explore this in Section 4. This is also closely related to the Online Newton Step (ONS) used in online learning.

3.2 Optimization vs Bayes

An important step in deriving optimization algorithms from Bayesian principles is the use of the delta approximation (50). At first, this might seem like a heuristic, but a closer look will highlight the fundamental difference between learning by optimization vs learning by Bayesian principles. Table 1 compares the optimality conditions of the fixed-point of (51) and (49) respectively. The 1^{st}-order condition in the first column is obtained by setting $\mathbf{w}_{t+1} = \mathbf{w}_t = \mathbf{w}_*$ in (51), and the 2^{nd}-order condition follows from the stationarity condition. In the second column, the 1^{st}-order condition is obtained by setting $\mathbf{m}_{t+1} = \mathbf{m}_t = \mathbf{m}_*$. The 2^{nd}-order condition is obtained $\mathbf{S}_{t+1} = \mathbf{S}_t = \mathbf{S}_*$ to get $\mathbf{S}_* = \mathbb{E}_{q_*(\mathbf{w})}[\nabla^2_{\boldsymbol{ww}} \bar{\ell}(\mathbf{w})]$ and then using the fact that $\mathbf{S}_* \succ 0$ for a valid Gaussian distribution.

Table 1: Optimality conditions for solutions found by an optimization vs Bayesian problem.

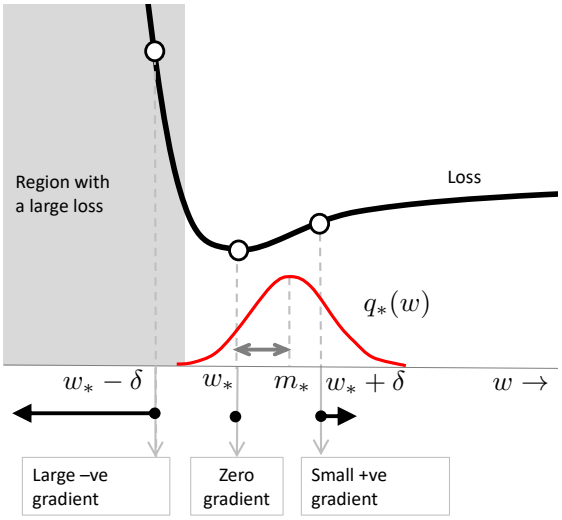
	Optimization	Bayes
1^{st}-order condition:	$\nabla_{\boldsymbol{w}} \bar{\ell}(\mathbf{w}_*) = 0$	$\mathbb{E}_{q_*(\mathbf{w})}[\nabla_{\boldsymbol{w}} \bar{\ell}(\mathbf{w})] = 0$
2^{nd}-order condition:	$\nabla^2_{\boldsymbol{ww}} \bar{\ell}(\mathbf{w}_*) \succ 0$	$\mathbb{E}_{q_*(\mathbf{w})}[\nabla^2_{\boldsymbol{ww}} \bar{\ell}(\mathbf{w})] \succ 0$

We can see that the solutions $\mathbf{w}_*$ found by optimization algorithms such as (51), gradient descent, or Newton's method, require the gradient at $\mathbf{w}_*$ to be zero and the respective Hessian to be positive definite. In constrast, the solutions $q_*(\mathbf{w}) := \mathcal{N}(\mathbf{w}|\mathbf{m}_*, \mathbf{S}_*^{-1})$ found by the Bayesian learning rule (49) require the *expectation of the gradient* to be zero and *expectation of the Hessian* to be positive definite (a similar result is discussed in Opper and Archambeau (2009) in the context of variational inference and Laplace's method). The optimization solution can be obtained as a special case of the Bayesian solution by using the delta approximation to approximate the expectation at the mean and ignore the higher order statistics of $q_*(\mathbf{w})$.

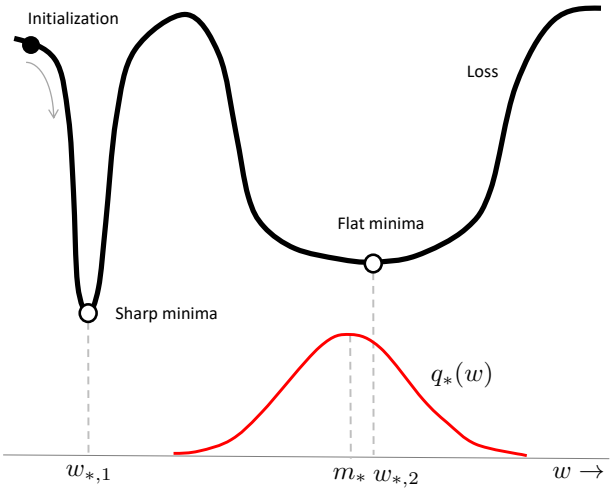
When do we expect the Bayesian solution to be better? The two solutions differ fundamentally: the optimization solution only focus on the optimal point $\mathbf{w}_*$, therefore is a *greedy, local* solution. The Bayesian solution, on the other hand, is built using the average of the gradients and Hessians at samples $\mathbf{w} \sim q_*(\mathbf{w})$. The solution therefore attemptes to exploit the information about the local neightborhood, which may lie in the proximity of $\mathbf{w}_*$. When the neighborhood contains information about the fragility of $\mathbf{w}_*$, we can expect the Bayesian solution $q_*(\mathbf{w})$ to be more robust than the optimization solution $\mathbf{w}_*$.

Fig. 3.2 shows two such cases. In Fig. 1(a), $\mathbf{w}_*$ lies next to a *wall* with extremely high loss. Such scenario occurs in classication settings, e.g., logistic regression, where a small perturbation in the data could suddenly lead to a high drop in accuracy. The Bayesian solution in this case pushes the solution to the away from the wall and reduce the chance of such loss. The push to the right in this case is due to the 1^{st}-order optimimality condition: $\mathbb{E}_{q_*(\mathbf{w})}[\nabla_{\boldsymbol{w}} \bar{\ell}(\mathbf{w})] = 0$, which cancels the large negative gradients on the left side of $\mathbf{w}_*$ by putting higher probability-mass to the right side of $\mathbf{w}_*$ under $q_*(\mathbf{w})$. The Bayesian solutions attempts to improve the robustness by exploiting the neighborhood information.

Another situation is shown in Fig. 3.2 where the neighborhood of one solution is sharp while the other is relatively shallower. A small local perturbation in the sharper solution will usually incur a much larger loss than in the shallower solution. Such situations are hypothesized to exist in deep learning problems (Hochreiter and Schmidhuber, 1997, 1995; Keskar et al., 2016), where a good performance of SGD is attributed to its ability to find shallow minima. The Bayesian solution in such cases is expected to favor the shallow solution since such neighborhoods are expected to be more stable under $q_*(\mathbf{w})$. For example, when initialized from left hand side of the sharp minima, the optimization solution will most likely converge to the sharp minima, e.g., when taking small step sizes. The Bayesian solution with similar step sizes are more likely to *tunnel* through the solution when $\mathbf{S}_t$ is reasonably high (see an



(a)



(b)

Figure 1: Two examples to illustrate the robustness of the Bayesian solution over the optimization solution, as dictated by the optimality conditions given in Table 1. (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. (b) Given a sharp minima vs a flat minima, the Bayesian solution often prefers the flatter minima.

illustration in Fig. 1 in Khan et al. (2017)). It is known that, for many such problems, there exist a lower bound σ^2 over $\mathbf{S}_t \succ \sigma^2\mathbf{I}$ such that the $\mathbb{E}_{q(\mathbf{w})}[\bar{\ell}(\mathbf{w})]$ is convex (Mobahi and Fisher III, 2015), and the iterations then will most likely converge to the shallower solution. Similar strategies exist in optimization literature where a probability distribution is used to *smooth* the loss function, e.g., 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 strategiesHuning (1976); Wierstra et al. (2014). The benefit of the Bayesian approach is that it *learns* the smoothing distribution, greatly simplifying the procedure.

The Bayesian solution is not always guaranteed to be more accurate when measured in terms of the loss $\bar{\ell}(\mathbf{w})$. For example, as the *wall* in Fig. 3.2 becomes more and more vertical, the mean of the $q_*(\mathbf{w})$ is pushed further away, which could lead to worse performance. This is because the Gaussian distribution is required to put smaller and smaller mass at $\boldsymbol{\theta}$ around the wall (this is related to the *zero-avoiding* property Bishop (2006)). In high-dimensional settings, this depends on the spectrum of the Hessian at/around $\mathbf{w}_*$. Such problems may arise in deep learning since the neural network models are non-identifiable and the landscape around $\mathbf{w}_*$ might have a complicated spectrum. However, such problems also require robustness considerations since they are prone to failure when training and testing conditions are perturbed. The Bayesian approach pays attention to both accuracy and robustness. In general, the robustness of the Bayesian solution is expected to be better than the optimization solution, even when there might be a slight loss in accuracy.

Another benefit of the Bayesian solution is that it generalizes beyond the second-order conditions. The fixed-point of the Bayesian learning rule (5) is obtained by setting $\boldsymbol{\lambda}_{t+1} = \boldsymbol{\lambda}_t = \boldsymbol{\lambda}_*$, to get,

$$\boldsymbol{\lambda}_* = \nabla_{\boldsymbol{\mu}_*} \mathbb{E}_{q_*(\mathbf{w})}[\bar{\ell}(\mathbf{w})]. \tag{52}$$

In other words, *the optimal natural parameter is equal to the natural-gradient of the loss* evaluated at the optimal expectation parameter. This condition is first discussed in this form in Khan and Nielsen (2018)

(a slightly different version is discussed by Salimans et al. (2013) but connection to natural gradients is not discussed). The second-order optimality conditions of Gaussian approximations are simply an instance of (52), and can be derived by using (43)-(45), as shown below:

$$\mathbf{S}_* = 2\nabla_{\boldsymbol{\mu}^{(2)}}\mathbb{E}_{q_*(\mathbf{w})}\left[\bar{\ell}(\mathbf{w})\right] = 2\nabla_{\mathbf{S}^{-1}}\mathbb{E}_{q_*(\mathbf{w})}\left[\bar{\ell}(\mathbf{w})\right] \quad \implies \mathbb{E}_{q_*(\mathbf{w})}[\nabla^2_{\boldsymbol{ww}}\bar{\ell}(\mathbf{w})] \succ 0, \tag{53}$$

$$\mathbf{S}_*\mathbf{m}_* = \nabla_{\boldsymbol{\mu}^{(1)}}\mathbb{E}_{q_*(\mathbf{w})}\left[\bar{\ell}(\mathbf{w})\right] = \mathbb{E}_{q_*(\mathbf{w})}\left[\nabla_{\mathbf{w}}\bar{\ell}(\mathbf{w})\right] - \mathbf{S}_*\mathbf{m}_* \quad \implies \mathbb{E}_{q_*(\mathbf{w})}\left[\nabla_{\mathbf{w}}\bar{\ell}(\mathbf{w})\right] = 0. \tag{54}$$

Remarkably, the 1st-order condition is obtained from the first natural/expectation parameter, while the 2nd-order condition is obtained from the second set. Therefore, multivariate-Gaussians with free mean and covariance parameters are sufficient to derive the second-order optimality condition in optimization. When the covariance is fixed, we recover the first-order condition, which corresponds to methods such as gradient-descent. To go beyond the second-order optimization, we simply need to pick a more expressive sufficient statistics, i.e., a more occurate posterior approximation. Higher-order optimization methods therefore are justified by Bayesian principles, since they result in a better posterior approximation.

The example discussed in this section should convince the reader of the importance of natural-gradients in deriving learning algorithms from Bayesian principles. *The natural-gradients simply are a way to compute the gradient of the loss through the distribution* $q(\mathbf{w})$. Similar ideas are used in stochastic relaxation methods and black-box optimization such as evolution strategies (Huning, 1976). When using a Gaussian $\mathcal{N}(\mathbf{w}|\mathbf{m}, \mathbf{I})$, the natural-gradient can be obtained using just the first-order information, i.e., the gradients $\nabla_{\mathbf{w}}\bar{\ell}(\mathbf{w})$. When using a more complex distribution $\mathcal{N}(\mathbf{w}|\mathbf{m}, \mathbf{S}^{-1})$, the natural-gradients are obtained using second-order information, i.e., both the gradient $\nabla_{\mathbf{w}}\bar{\ell}(\mathbf{w})$ and the Hessian $\nabla^2_{\mathbf{ww}}\bar{\ell}(\mathbf{w})$. This is essentially the result of using a more complex sufficient statistics, which gives rise to two expectation parameters $\boldsymbol{\mu}^{(1)}$ and $\boldsymbol{\mu}^{(2)}$ for natural-gradient computation. In general, by employing more complex sufficient statistics, we can design higher-order optimization algorithm.

The bottleneck in a more accurate method therefore boilds down to the computation of natural-gradient, which remains challenging for more complex distribution than Gaussians. Now, we will discuss an alternate way to go beyond second-order method by using a mixture of exponential family based on (Lin et al., 2019).

3.3 Going Beyond Newton's Method

(Still writing this :( )

4 Deep-Learning Algorithms from the Bayesian Learning Rule

4.1 Stochastic Gradient Descent

Stochastic gradient descent (SGD), one of the most popular deep-learning algorithm, is straightforwad to derive from Bayesian learning rule, since it is simply a variant of gradient descent with stochastic gradients. The most common stochastic approximation is the one based on *minibatches* of data,

$$\nabla_{w_j}\bar{\ell}(\mathbf{w}) \approx \widehat{\nabla}_{w_j}\bar{\ell}(\mathbf{w}) := \underbrace{\frac{N}{M}\sum_{i\in\mathcal{M}}\nabla_{w_j}\ell(y_i, \mathbf{f}_{\mathbf{w}}(\mathbf{x}_i))}_{\text{Minibatch gradient}} + w_j\delta, \tag{55}$$

where $\mathcal{M}$ is a set of M examples sampled uniformly from the training dataset and we have assumed an L_2 regularizer $r(\mathbf{w}) := \frac{1}{2}\delta\mathbf{w}^\top\mathbf{w}$ for simplicity with $\delta > 0$ as the regularization parameter. Therefore, SGD is obtained as a special case when Bayesian learning rule is used to estimate the mean $\mathbf{m}$ of a

Gaussian approximation $q(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{m}, \mathbf{I})$, while employing the gradient approximation shown in (21) along with the stochastic gradients (55).

The choice of covariance matrix in $q(\mathbf{w})$ can be arbitrary. For example, if we set $q(\mathbf{w}) := \mathcal{N}(\mathbf{w}|\mathbf{m}, \boldsymbol{\Sigma})$ where $\boldsymbol{\Sigma}$ is an arbitrary positive-definite matrix, and estimate $\mathbf{m}$ using the Bayesian learning rule, we still get a gradient descent algorithm:

$$\mathbf{m}_{t+1} \leftarrow \mathbf{m}_t - \rho_t \boldsymbol{\Sigma} \, [\widehat{\nabla}_{\mathbf{w}} \bar{\ell}(\mathbf{m}_t)], \tag{56}$$

but now the gradients are scaled by $\boldsymbol{\Sigma}$. The choice of $\boldsymbol{\Sigma}$ scales the learning-rate and the noise covariance of the gradients, and could improve the performance of the algorithm if set appropriately.

An open question in the deep-learning community is whether the SGD iterates can be used to estimate the covariance. Recently, such attempts have been made (Mandt et al., 2017; Maddox et al., 2019). Our derivation here suggests that plain SGD itself is not enough to estimate the covariance since the covariance can be set arbitrarily *without* affecting the solution of the SGD iterates (given that it converges). One possible idea is to use a preconditioner in (56) to match the noise in the steps $\mathbf{m}_{t+1} - \mathbf{m}_t$ and use the preconditioner to estimate $\boldsymbol{\Sigma}$, but this can only be done after convergence and requires us to estimate the statistics about the gradient noise. To this end, Mandt et al. (2017) derive an optimal choice of the preconditioner that minimizes the Bayesian objective. They do this by relating the SGD iterates to an Ornstein-Uhlenbeck (OU) process. As expected, the optimal choice of the preconditioner $\boldsymbol{\Sigma}$ is related to the source of gradient noise: learning rate, minibatch size, and momentum coefficient. The optimal settings of these parameters are difficult to estimate in practice, and require estimation of the Hessian or the noise covariance of the gradients. In addition, the assumptions under which these results are valid rarely hold true, and only close to convergence. Another work by Chaudhari and Soatto (2018) shows further negative results: SGD does optimize the Bayesian objective (2), but not on the actual loss rather a variant of it. The loss that is minimized by SGD is equivalent to the actual loss only when the gradient noise is *isotropic* which is rarely the case in practice. Again, using the preconditioner might fix this issue, but the preconditioner needs to be learnt separately.

The choice of the preconditioner makes this a chicken-and-egg problem: the covariance requires an estimate of gradient noise which requires the correct setting of the preconditioner but that itself depends on the covariance. We can of course estimate the gradient noise from the iterates, e.g., Mandt et al. (2017) use an exponential smoothing similar to (23)), but this additional effort is unavoidable and resorting such heuristic estimation methods might not be a good idea. The Bayesian learning rule offers a simple fix to this issue: estimate both the mean $\mathbf{m}$ and the preconditioner $\boldsymbol{\Sigma}$, which can be done through Newton-like update shown in (23). These updates are in fact very similar to the exponential-smoothing scheme used by Mandt et al. (2017), which fits well in our Bayesian narrative. As we shall see, the adaptive learning-rate algorithms discussed in the next section provide a cheaper alternative and we can design better variants to estimate the covariance using them.

4.2 Adaptive Learning-Rate Algorithms

A common practice in deep-learning is to employ variants of SGD where the learning rate is adapted using a scaling vector. Many such variants are based on Newton's method and employed several approximations to reduce the computation cost. For example, instead of using the full Hessian, a diagonal Hessian approximation is used Barto and Sutton (1981); Becker et al. (1988) which can be efficiently obtained using the backpropagation algorithm. A minibatch estimate of the Hessian further reduces the cost, but since the scaling vector can be very noisy, it is necessary to stabilize the Hessian estimate, and reduce numerical issues and costly line-search procedures. A popular heuristic is to employ *exponential smoothing* of the Hessian, e.g., LeCun et al. (1998) recommend the following update with a scaling vector

$\mathbf{s}_t$ obtained using an exponential smoothing of the diagonal Hessian (see Section 9.1 in their paper):

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \alpha \frac{1}{\mathbf{s}_{t+1}} \circ \left[\widehat{\nabla}_{\mathbf{w}} \bar{\ell}(\mathbf{w}_t)\right], \text{ where } \qquad \mathbf{s}_{t+1} \leftarrow (1-\beta)\mathbf{s}_t + \beta \text{ Diag}\left[\widehat{\nabla}^2_{\mathbf{ww}} \bar{\ell}(\mathbf{w}_t)\right]. \tag{57}$$

with $\mathbf{a} \circ \mathbf{b}$ and $\mathbf{a}/\mathbf{b}$ denoting element-wise multiplication and division respectively, and $\text{Diag}[\widehat{\nabla}^2_{\mathbf{ww}} \bar{\ell}(\mathbf{w})]$ denotes a diagonal matrix with the j'th diagonal element approximated using a minibatch Hessian estimate: $\frac{N}{M} \sum_{i \in \mathcal{M}_t} |\nabla^2_{w_j w_j} \ell(y_i, \mathbf{f}_{\mathbf{w}}(\mathbf{x}_i))| + \delta$. The absolute value of the Hessian is used to ensure that the scaling vector does not become negative, while the regularization parameter δ ensures that the vector is nonzero. Exponential smoothing used in the above update is a common feature even in modern adaptive learning-rate algorithms. For example, the natural Newton method of Le Roux and Fitzgibbon (2010) and the vSGD method by Schaul et al. (2013) both use exponential-smoothing of the second-order information, while modern variants, such as RMSprop (Tieleman and Hinton, 2012), AdaDelta (Zeiler, 2012), and Adam (Kingma and Ba, 2015), use an exponential-smoothing of the square of the gradient, which can be interpreted as a crude but much cheaper approximation of second-order information (Bottou et al., 2016). Exponential smoothing is an essential trick-of-the-trade which works extremely well in practice. Its application is motivated from the time-series literature (Brown, 1959; Holt et al., 1960; Gardner Jr, 1985) where it is used to estimate a nonstationary signal, but here we show that the Bayesian learning rule gives rise to an update similar to (57).

We can derive the exponential-smoothing update (57) from Bayesian principles by using the Bayesian Learning rule to estimate a factorized Gaussian approximation $q(\mathbf{w}) := \mathcal{N}(\mathbf{w}|\mathbf{m}, \text{diag}(\mathbf{s})^{-1})$ where $\text{diag}(\mathbf{a})$ denotes a diagonal matrix whose diagonal is equal to $\mathbf{a}$. A factorized Gaussian is a minimal exponential-family distribution, since each entry w_j is a minimal univariate Gaussian. The update directly follows from the Newton-like update (22) derived in the previous section, with a difference that the precision matrix is replaced by a vector:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \rho_t \frac{1}{\mathbf{s}_{t+1}} \circ \left[\nabla_{\mathbf{w}} \bar{\ell}(\mathbf{w}_t)\right], \qquad \text{where } \mathbf{s}_{t+1} \leftarrow (1-\rho_t)\mathbf{s}_t + \rho_t \text{ Diag}\left[\nabla^2_{\mathbf{ww}} \bar{\ell}(\mathbf{w}_t)\right]. \tag{58}$$

If we replace the gradient $\nabla_{\mathbf{w}} \bar{\ell}(\mathbf{w}_t)$ and Hessian $\nabla^2_{\mathbf{ww}} \bar{\ell}(\mathbf{w}_t)$ by their stochastic approximation and use different learning rates for $\mathbf{w}_t$ and $\mathbf{s}_t$, we recover the update (57). The similarity is uncanny: Bayesian updating somehow naturally employs exponential smoothing of the second-order information.

This might seem surprising at first, but it is not a coincidence. A more careful look at the Bayesian learning rule (5) reveals that this is a direct result of use of the entropy term $\mathbb{E}_{q(\mathbf{w})}[\log q(\mathbf{w})]$ (in the KL term) of the Bayesian objective (2). The natural-gradient of this term is equal to the natural parameter $\boldsymbol{\lambda}$ as shown in (32), which when plugged in the natural-gradient update (31) gives rise to the exponential-smoothing in the Bayesian learning rule. For Gaussians, the second natural parameter $\text{diag}(\mathbf{s})$ is the precision matrix which turns out to contain an expoentially-smoothed estimate of the (diagonal of the) Hessian. The entropy term in the Bayesian objective is the reason why we are able to derive the exponential-smoothing used in deep learning from Bayesian principles.

Such exponential smoothing is an attractive feature of Bayesian principles to deal with stochastic and non-stationary settings. It recommends the use of exponential smoothing to combine the new evidence with the old ones. Modern deep-learning optimizers, such as RMSprop Tieleman and Hinton (2012), AdaDelta (Zeiler, 2012), and Adam (Kingma and Ba, 2015), all employ such exponential smoothing for the scaling vector, even though they approximate the second-order information by square of the gradient to reduce the computation cost of computing the Hessian. For example, RMSprop uses the following update:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \alpha \frac{1}{\sqrt{\mathbf{s}_{t+1}} + c\mathbf{1}} \circ \left[\widehat{\nabla}_{\mathbf{w}} \bar{\ell}(\mathbf{w}_t)\right], \text{ where } \mathbf{s}_{t+1} \leftarrow (1-\beta)\mathbf{s}_t + \beta \left[\widehat{\nabla}_{\mathbf{w}} \bar{\ell}(\mathbf{w}_t) \circ \widehat{\nabla}_{\mathbf{w}} \bar{\ell}(\mathbf{w}_t)\right]. \tag{59}$$

which is very similar to (58) but now replaces the Hessian by the square of the gradient and uses a square-root of the scale vector whille updating $\mathbf{w}_t$ (another minor difference is a small constant c to ensure that the scale vector is nonzero). The square-root is a heuristic to get the units right. The square of a minibatch gradient over M examples involves M^2 terms as opposed to M terms in the minibatch hessian. Square root helps to keep the order of magnitude roughly same as a Newton-like method. The overall form of the update used in such deep-learning optimizers remains very similar to the one obtained by using the Bayesian learning rule.

Due to the stochastic approximation using minibatches, the new gradient and Hessian are noisy and could be unreliable. The Bayesian principle recommends to not rely too much on the new information, essentially implying that the learning rates should be set small. This fits well with the small learning-rates typically used in practice, e.g., the original recommendation for Adam in Kingma and Ba (2015) is to use $\alpha = \beta = 0.001$ in the update (59) (see Algorithm 1 in their paper where $\beta_2 = 1 - \beta = 0.999$ and $\alpha = 0.001$). In reality, the optimal setting of these learning rate do require some fine tuning, but the general practical recommendations seem to fit the Bayesian narrative.

Aitchison (2018) derive deep-learning optimizers using a Bayesian filtering approach, where the updates are very similar to the ones we presented. They further discuss modifications to obtain the square-root in Adam and RMSprop. Another similar approach by Ollivier et al. (2018) exploits the connection of Kalman filtering and natural-gradient method to derive results of similar nature. In contrast to these works, our proposal is more general, enabling us to derive all sorts of learning algorithms.

4.3 Momentum

Momentum is yet another technique, which is extremely useful in improving the performance of SGD (Sutskever et al., 2013). Modern adaptive-learning algorithms, such as RMSprop and Adam, employ variants of the classical momentum where local geometry of the data is taken into account. We will now show that such variants as well as classical momentum techniques can all be derived from Bayesian principles, where the geometry is automatically chosen by the posterior approximation $q(\mathbf{w})$.

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

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \alpha \widehat{\nabla}_{\mathbf{w}} \bar{\ell}(\mathbf{w}_t) + \underbrace{\gamma(\mathbf{w}_t - \mathbf{w}_{t-1})}_{\text{Momentum}} \tag{60}$$

where $\gamma > 0$ is the *mometum coefficient*. This simple modification of SGD, when combined with initialization tricks, leads to a significant improvement in their performance (Sutskever et al., 2013).

Adpative-learning rate algorithms, such as RMSprop and Adam, employ a variant of the classical momentum method where the local geometry of the data is taken into account. The variant is implemented by using a exponential-smoothing the gradient: $\mathbf{u}_{t+1} \leftarrow \gamma \mathbf{u}_t + (1-\gamma)\widehat{\nabla}_{\mathbf{w}} \bar{\ell}(\mathbf{w}_t)$, and modifying the update to use $\mathbf{u}_{t+1}$ instead of the stochastic gradient (Graves, 2013; Kingma and Ba, 2015). This is illustrated below for the RMSprop update (59):

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \alpha \frac{1}{\sqrt{\mathbf{s}_{t+1}} + c\mathbf{1}} \circ \mathbf{u}_{t+1}, \text{ where } \mathbf{s}_{t+1} \leftarrow (1-\beta)\mathbf{s}_t + \beta \left[\widehat{\nabla}_{\mathbf{w}} \bar{\ell}(\mathbf{w}_t) \circ \widehat{\nabla}_{\mathbf{w}} \bar{\ell}(\mathbf{w}_t)\right], \tag{61}$$

The Adam optimizer uses these updates along with additional bias-corrections for $\mathbf{u}_t$ and $\mathbf{s}_t$ (Kingma and Ba, 2015). By simply plugging the expression for $\mathbf{u}_{t+1}$ in (61) and expressing $\mathbf{u}_t$ in terms of $\mathbf{w}_t$ and $\mathbf{w}_{t-1}$, we can write the update in a similar form as (60):

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \alpha(1-\gamma) \frac{1}{\sqrt{\mathbf{s}_{t+1}} + c\mathbf{1}} \circ \left[\widehat{\nabla}_{\mathbf{w}} \bar{\ell}(\mathbf{w}_t)\right] + \gamma \frac{\sqrt{\mathbf{s}_t} + c\mathbf{1}}{\sqrt{\mathbf{s}_{t+1}} + c\mathbf{1}} \circ (\mathbf{w}_t - \mathbf{w}_{t-1}), \tag{62}$$

We see that the updates rescale the mometum term using the scaling vectors $\mathbf{s}_{t+1}$ and $\mathbf{s}_t$.

The rescaling used in (62), although intuitive, appears to be an arbitrary choice. When switching from SGD to an adaptive-learning rate algorithm, why should one resort to the particular rescaling used in (62)? We will now show that Bayesian principles provide an elegant justification for the mometum variants used in the adaptive-learning rate algorithms.

We start with an alternate formulation of the mometum. The momentum term can be interpreted as a *push* away from the previous iterate $\mathbf{w}_{t-1}$. This interpreation arises by rewriting (60) as the following optimization problem where a quadratic penalty term is added to penalize the proximity to $\mathbf{w}_{t-1}$:

$$\mathbf{w}_{t+1} \leftarrow \arg\min_{\boldsymbol{w}} \langle \nabla_{\boldsymbol{w}} \bar{\ell}(\mathbf{w}_t), \mathbf{w} \rangle + \frac{1+\gamma}{2\alpha} \|\mathbf{w} - \mathbf{w}_t\|_2^2 - \underbrace{\frac{\gamma}{2\alpha} \|\mathbf{w} - \mathbf{w}_{t-1}\|_2^2}_{\text{Momentum}}. \tag{63}$$

This formulation can be generalized by using Bayesian principles where L_2 distance used above the KL divergence between the posterior approximations $q(\mathbf{w})$ and $q_t(\mathbf{w})$. When coupled with the Bayesian learning rule, this results in a momentum update where the rescaling is adjusted according to the geometry of the posterior approximation. We will now show this.

We modify the Bayesian learning rule to add a momentum term. This can be done via the mirror descent update (38) discussed in Section 2 to obtain a generalization of (63):

$$\boldsymbol{\mu}_{t+1} \leftarrow \arg\min_{\boldsymbol{\mu} \in \mathcal{M}} \langle \nabla_{\boldsymbol{\mu}} \mathcal{L}(q_t), \boldsymbol{\mu} \rangle + \frac{1+\gamma_t}{\rho_t} \mathbb{D}_{A^*}(\boldsymbol{\mu} \| \boldsymbol{\mu}_t) - \frac{\gamma_t}{\rho_t} \underbrace{\mathbb{D}_{A^*}(\boldsymbol{\mu} \| \boldsymbol{\mu}_{t-1})}_{\text{Natural Momentum}}, \tag{64}$$

where $\boldsymbol{\mu}$ is the expectation parameter, $A*(\boldsymbol{\mu})$ is the convex-conjugate of the log-partition function, and γ_t, ρ_t are step-sizes. Similarly to (63), the last term penalizes the proximity to $\boldsymbol{\mu}_{t-1}$ where the proximity is measured according to the Bregman divergence (equivalenty a KL divergence). Following Khan et al. (2018), we call this term the *natural momentum* term. The addition of this term gives us the following Bayesian learning rule with momentum for exponential-family $q(\mathbf{w})$ with a constant base-measure:

$$\boldsymbol{\lambda}_{t+1} \leftarrow (1-\rho_t)\boldsymbol{\lambda}_t - \rho_t \nabla_{\boldsymbol{\mu}} \mathbb{E}_{q_t(\mathbf{w})} \left[\bar{\ell}(\mathbf{w})\right] + \gamma_t(\boldsymbol{\lambda}_t - \boldsymbol{\lambda}_{t-1}), \tag{65}$$

We will now show that by choosing an appropriate family of $q(\mathbf{w})$, we can recover (60) and (62).

The update (60) can be recovered by simply choosing $q(\mathbf{w}|\mathbf{m}, \mathbf{I})$ (the choice of covariance is arbitrary). The proof follows the procedure discussed in Section 1.5. As discussed there, the natural parameter $\boldsymbol{\lambda} = \mathbf{m}$, therefore the term $\boldsymbol{\lambda}_t - \boldsymbol{\lambda}_{t-1}$ reduces to $\mathbf{m} - \mathbf{m}_t$. Following the derivation in Section 1.5 where we use the zeroth-order delta approximation (21) and denote $\mathbf{m}$ by $\mathbf{w}$, we can show that (65) reduces to (60) (the step-sizes need to be chosen appropriately).

A strategy similar to (62) can be obtained by choosing $q(\mathbf{w}|\mathbf{m}, \text{diag}(\mathbf{s})^{-1})$ and following the derivation of the Newton's method in Section 3. The resulting update is a modification of (58) where the modification is due to the term $\boldsymbol{\lambda}_t - \boldsymbol{\lambda}_{t-1}$. This is shown below with the additional terms highlighted in red:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \rho_t \frac{1}{\mathbf{s}_{t+1}} \circ \left[\nabla_{\mathbf{w}} \bar{\ell}(\mathbf{w}_t)\right] + \frac{1}{\mathbf{s}_{t+1}} \circ (\mathbf{s}_t \mathbf{w}_t - \mathbf{s}_{t-1}\mathbf{w}_{t-1}), \tag{66}$$

$$\mathbf{s}_{t+1} \leftarrow (1-\rho_t)\mathbf{s}_t + \rho_t \, \text{Diag}\left[\nabla^2_{\mathbf{ww}} \bar{\ell}(\mathbf{w}_t)\right] + \gamma_t(\mathbf{s}_t - \mathbf{s}_{t-1}). \tag{67}$$

If we assume $\mathbf{s}_t \approx \mathbf{s}_{t-1}$, which holds when ρ_t close to 1, then the update take a very similar form as (62):

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \rho_t \frac{1}{\mathbf{s}_{t+1}} \circ \left[\nabla_{\mathbf{w}} \bar{\ell}(\mathbf{w}_t)\right] + \frac{\mathbf{s}_t}{\mathbf{s}_{t+1}} \circ (\mathbf{w}_t - \mathbf{w}_{t-1}), \tag{68}$$

$$\mathbf{s}_{t+1} \leftarrow (1-\rho_t)\mathbf{s}_t + \rho_t \, \text{Diag}\left[\nabla^2_{\mathbf{ww}} \bar{\ell}(\mathbf{w}_t)\right]. \tag{69}$$

By applying approximations similar to the ones we used to derive RMSprop, we can obtain (62) (replace the Hessian by square of gradients and use square-root to update $\mathbf{s}_t$). The Bayesian learning rule, not only justifies the use of scaling vectors for momentum but also to the use of exponential-smoothing to obtain the scaling vectors.

5 Bayesian Inference and Its Approximations from the Bayesian Learning Rule

[Writing in progress...]

References

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

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

Amari, S.-I. (2008). Information geometry and its applications: Convex function and dually flat manifold. In *LIX Fall Colloquium on Emerging Trends in Visual Computing*, pages 75–102. Springer.

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

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

Becker, S., Le Cun, Y., et al. (1988). Improving the convergence of back-propagation learning with second order methods. In *Proceedings of the 1988 connectionist models summer school*, pages 29–37.

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

Bissiri, P. G., Holmes, C. C., and Walker, S. G. (2016). A general framework for updating belief distributions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 78(5):1103–1130.

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

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

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

Chaudhari, P. and Soatto, S. (2018). 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.

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

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

Geman, S. and Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence*, (6):721–741.

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

Hazan, E., Agarwal, A., and Kale, S. (2007). Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2-3):169–192.

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

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

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

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

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

Jiang, W. and Tanner, M. A. (2008). Gibbs posterior for variable selection in high-dimensional classification and data mining. *The Annals of Statistics*, pages 2207–2231.

Johansen, S. (1979). Introduction to the Theory of Regular Exponential Famelies.

Kalai, A. and Vempala, S. (2005). Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71(3):291–307.

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

Khan, M., Nielsen, D., Tangkaratt, V., Lin, W., Gal, Y., and Srivastava, A. (2018). Fast and scalable Bayesian deep learning by weight-perturbation in Adam. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2611–2620, Stockholmsmässan, Stockholm Sweden. PMLR.

Khan, M. E. and Lin, W. (2017). 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.

Khan, M. E., Lin, W., Tangkaratt, V., Liu, Z., and Nielsen, D. (2017). Variational Adaptive-Newton Method for Explorative Learning. *ArXiv e-prints*.

Khan, M. E. and Nielsen, D. (2018). 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.

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

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

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

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

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

Lin, W., Khan, M. E., and Schmidt, M. (2019). Fast and simple natural-gradient variational inference with mixture of exponential-family approximations.

Maddox, W. J., Izmailov, P., Garipov, T., Vetrov, D. P., and Wilson, A. G. (2019). A simple baseline for bayesian uncertainty in deep learning. In *Advances in Neural Information Processing Systems*, pages 13153–13164.

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

Martens, J. (2014). New insights and perspectives on the natural gradient method. *arXiv preprint arXiv:1412.1193*.

McMahan, H. B. (2011). Follow-the-regularized-leader and mirror descent: Equivalence theorems and l1 regularization. In *International conference on Artificial Intelligence and Statistics*.

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

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

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

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

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

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

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

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

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

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

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

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

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

Vovk, V. G. (1990). Aggregating strategies. *Proc. of Computational Learning Theory, 1990*.

Wainwright, M. J. and Jordan, M. I. (2008). Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1–2:1–305.

Warmuth, M. K. (2010). The blessing and the curse of the multiplicative updates. In *Discovery Science*, page 382.

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

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

Zellner, A. (1988). Optimal information processing and Bayes's theorem. *The American Statistician*, 42(4):278–280.

A Bayesian Inference from Bayesian Learning Problem

We show that Bayesian inference can be obtained as a special case of the Bayesian learning problem. This happens when the loss corresponds to the log of a probability distribution $\ell(y, f_{\mathbf{w}}(\mathbf{x})) := -\log p(y|f_{\mathbf{w}}(\mathbf{x}))$. The corresponding Bayesian model has a likelihood $p(y|f_{\mathbf{w}}(\mathbf{x}))$ with prior $p(\mathbf{w})$, and the posterior distribution takes the following form:

$$p(\mathbf{w}|\mathcal{D}) := \frac{1}{\mathcal{Z}(\mathcal{D})} \prod_{i=1}^{N} p(\mathbf{y}_i|f_w(\mathbf{x}_i))p(\mathbf{w}) \tag{70}$$

where $\mathcal{Z}(\mathcal{D})$ is the normalizing constant of the posterior. The minimizer of the Bayesian learning problem recovers this posterior distribution.

We can show this by reorganizing the Bayesian objective (2):

$$\mathcal{L}(q) := -\mathbb{E}_{q(\mathbf{w})}\left[\sum_{i=1}^{N} \log p(y_i|f_{\mathbf{w}}(\mathbf{x}_i))\right] + \mathbb{D}_{KL}[q(\mathbf{w}) \,\|\, p(\mathbf{w})] \tag{71}$$

$$= \mathbb{E}_{q(\mathbf{w})}\left[\log \frac{q(\mathbf{w})}{\prod_{i=1}^{N} p(y_i|f_{\mathbf{w}}(\mathbf{x}_i))p(\mathbf{w})}\right] \tag{72}$$

$$= \mathbb{E}_{q(\mathbf{w})}\left[\log \frac{q(\mathbf{w})}{\frac{1}{\mathcal{Z}(\mathcal{D})}\prod_{i=1}^{N} p(y_i|f_{\mathbf{w}}(\mathbf{x}_i))p(\mathbf{w})}\right] + \log \mathcal{Z}(\mathcal{D}) \tag{73}$$

$$= \mathbb{D}_{KL}[q(\mathbf{w}) \,\|\, p(\mathbf{w}|\mathcal{D})] + \log \mathcal{Z}(\mathcal{D}) \tag{74}$$

When $\mathcal{Q} \equiv \mathcal{P}$, we can choose any distribution $q(\mathbf{w})$ and clearly setting $q(\mathbf{w}) \equiv p(\mathbf{w}|\mathcal{D})$ minimizes the above equation (since $\mathcal{Z}(\mathcal{D})$ is a constant, and minimum value of KL divergence is zero).