A UNIFIED THEORY OF ADAPTIVE STOCHASTIC GRADIENT DESCENT AS BAYESIAN FILTERING

Anonymous authors
Paper under double-blind review

ABSTRACT

We formulate stochastic gradient descent (SGD) as a novel factorised Bayesian filtering problem, in which each parameter is inferred separately, conditioned on the corresponding backpropagated gradient. Inference in this setting naturally gives rise to BRMSprop and BAdam: Bayesian variants of RMSprop and Adam. Remarkably, the Bayesian approach recovers many features of state-of-the-art adaptive SGD methods, including amongst others root-mean-square normalization, Nesterov acceleration and AdamW. As such, the Bayesian approach provides one explanation for the empirical effectiveness of state-of-the-art adaptive SGD algorithms. Empirically comparing BRMSprop and BAdam with naive RMSprop and Adam on MNIST, we find that Bayesian methods have the potential to considerably reduce test loss and classification error.

1 INTRODUCTION

Deep neural networks have recently shown huge success at a range of tasks including machine translation (Wu et al., 2016), dialogue systems (Serban et al., 2016), handwriting generation (Graves, 2013) and image generation (Radford et al., 2015). These successes have been facilitated by the development of a broad range of adaptive SGD methods, including ADAGrad (Duchi et al., 2011), RMSprop (Hinton et al., 2012), Adam (Kingma & Ba, 2015), and variants thereof, including Nesterov acceleratation (Nesterov, 1983; Bengio et al., 2013; Dozat, 2016) and AdamW (Loshchilov & Hutter, 2017). However, such a broad range of approaches raises the question of whether it is possible to obtain a unified theoretical understanding of adaptive SGD methods. Here we provide such a theory by reconciling state-of-the-art adaptive SGD algorithms with very early work that used Bayesian (Kalman) filtering to optimize the parameters of neural networks (Puskorius & Feldkamp, 1991; Sha et al., 1992; Puskorius & Feldkamp, 1994; 2001; Feldkamp et al., 2003; Ollivier, 2017).

There have recently been attempts to connect adaptive SGD algorithms to natural gradient variational inference (VI) (Zhang et al., 2017; Khan et al., 2017; 2018). These approaches give a momentum-free algorithm with a mean-square normalizer, in contrast to perhaps the most popular adaptive method, Adam (Kingma & Ba, 2015), which combines momentum with a root-mean-square normalizer. To achieve a closer match to Adam, they modified their natural gradient VI updates, without a principled justification based on approximate inference, to incorporate momentum (Zhang et al., 2017; Khan et al., 2018), and the root-mean-square normalizer (Khan et al., 2017; 2018). As such, there appears to be only a loose connection between successful adaptive SGD algorithms such as Adam, and natural gradient VI.

There is a formal correspondence between natural gradient VI (Zhang et al., 2017; Khan et al., 2017; 2018) and Bayesian filtering (Ollivier, 2017). While Ollivier (2017) did not examine the relationship between their filtering updates and RMSprop/Adam, the equivalence of this particular filtering approach and natural gradient VI indicates that they would encounter the issues described above, and thus be unable to obtain momentum or the root-mean-square normalizer (Zhang et al., 2017; Khan et al., 2017; 2018). More problematically, Ollivier (2017) introduces dynamics into the Kalman filter, but these dynamics correspond to the "addition of an artificial process noise Q_t proportional to [the posterior covariance] P_{t-1}". Thus, their generative model depends on inferences made under that model: a highly unnatural assumption that most likely does not correspond to any "real" generative process.

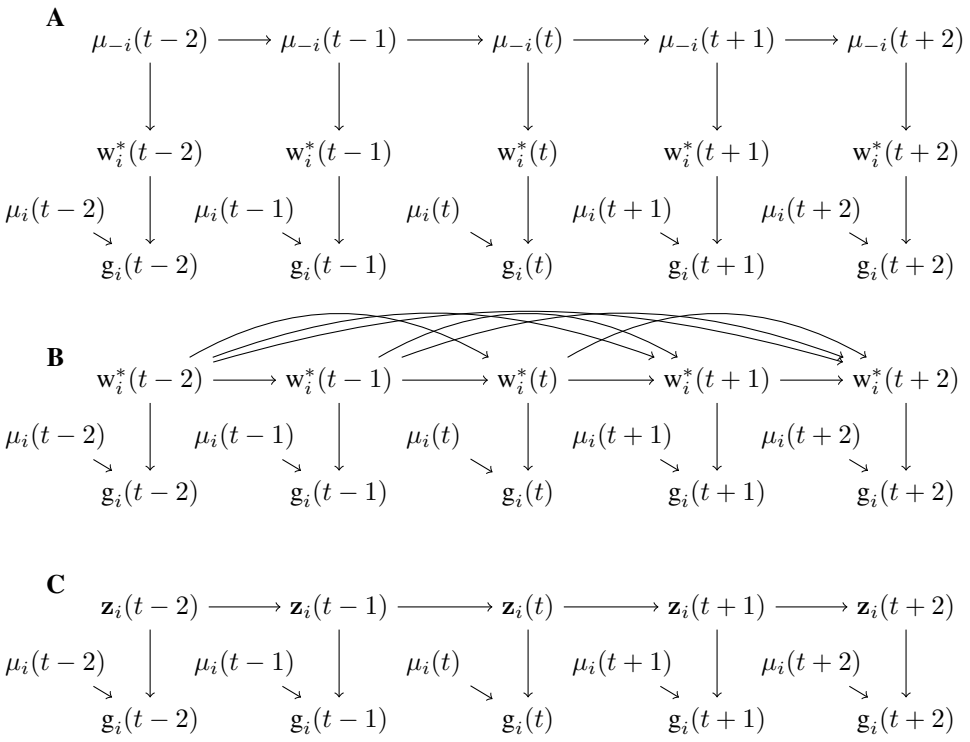

Figure 1: The heirarchy of generative models underlying our updates. **A** Full model for the gradients for a single parameter. The current estimate for all the other parameters, $\mu_{-i}(t)$ vary slowly over time, and give rise to the current optimal value for the ith parameter, w_i^*. The gradient then arises from the current estimate of the ith parameter, $\mu_i(t)$ (which is treated as an input here), and the optimal value, ith parameter, w_i^*. **B** The graphical model obtained by integrating over trajectories for the other parameter estimates, $\mu_{-i}(t)$. In practice, we use a simplified model as reasoning about all possible trajectories of $\mu_{-i}(t)$ is intractable. **C** To convert the model in **B** into a tractable hidden Markov model (HMM), we define a new variable, $\mathbf{z}_i(t)$, which incorporates w_i^* along with other information about the dynamics.

How might we obtain a principled Bayesian filtering approach that recovers the two key features of state-of-the-art adaptive SGD algorithms: momentum and the root-mean-square normalizer? Here, we note that past approaches including natural gradient VI take a complex generative model over all N parameters jointly, and use a very strong approximation: factorisation. Given that we know that the true posterior is a highly complicated, correlated distribution, it is legitimate to worry that these strong approximations might meaningfully disrupt the ability of Bayesian filtering to give close-to-optimal updates. Here we take an alternative approach, baking factorisation into our generative model, so that we can use Bayesian inference to reason about (Bayes) optimal updates under the constraints imposed by factorisation. In particular, we split up the single large inference problem over all N parameters, $\mathbf{w}$, into N small inference problems over a single parameter. Remarkably, by incorporating factorisation into the problem setting, we convert intractable, high-dimensional correlations in the original posterior into tractable low-dimensional dynamics in the factorised model. This dynamical prior has a "natural" form, at least compared with Ollivier (2017), in that it does not depend on the posterior. Next, we give a generic derivation showing that Bayesian SGD is an adaptive SGD method, where the uncertainty is used to precondition the gradient. We then adapt the generic derivation to the two cases of interest: RMSprop (Hinton et al., 2012) and Adam (Kingma & Ba, 2015). Finally, we discuss the general features of Bayesian adaptive SGD methods, including AdamW (Loshchilov & Hutter, 2017) and Nesterov acceleration (Nesterov, 1983; Dozat, 2016), amongst others.

2 FACTORISATION IMPLIES A RICH DYNAMICAL PRIOR

The most obvious factorised approach is to compute the true posterior, $\mathrm{P}\left(\mathrm{w}_i|\mathcal{D}\right)$, over a single weight, w_i, conditioned on all the data, $\mathcal{D}$. However, this approach immediately fails, because symmetries involving permuting the hidden units (Sussmann, 1992) imply that marginal posteriors such as $\mathrm{P}\left(\mathrm{w}_i|\mathcal{D}\right)$ are always so broad as to be useless.

To obtain an informative, narrow posterior, one approach is to eliminate these symmetries by conditioning on our current estimate of the other parameters. In particular, we define a random variable, w_i^*, representing the optimal value for the ith weight, conditioned on the other weights, w_{-i}, being equal to our current estimate of those parameters, μ_{-i},

$$\mathrm{w}_i^* = \arg\max_{w_i} \mathbb{E}\left[\log \mathrm{P}\left(\mathrm{d}|\mathrm{w}_i = w_i, \mathrm{w}_{-i} = \mu_{-i}\right)\right], \tag{1}$$

where w_i^* is a random variable because it depends on our current estimate of the other weights, μ_{-i}, which is treated as a random variable, and w_i is a "dummy" variable giving the location at which we evaluate the likelihood. The random variable describing our current estimate of the other weights, μ_{-i}, is generated from a complex distribution controlled by our overall optimization scheme. However, we cannot now use the usual Bayesian approach of inferring w_i^* based on the data, because the data depend on the full weight vector, w, not on w_i^*. Instead, note that the updates for almost all neural network optimization algorithms depend only on the history of backpropagated gradients. This suggests a potential approach: writing down a generative model for the backpropagated gradients that depends on w_i^*, then inverting that model to infer w_i^* from those gradients.

Following standard approximations (e.g. Zhang et al., 2017; Khan et al., 2017; 2018) as closely as possible, we approximate the likelihood as a second-order Taylor series expansion. In particular, we consider the log-likelihood for a single minibatch, d, where the minibatch is treated as a random variable drawn from the underlying true data distribution, and we expand in the ith direction, keeping the other parameters, w_{-i}, fixed to their current estimates, μ_{-i},

$$\log \mathrm{P}\left(\mathrm{d}|\mathrm{w}_i = w_i, \mathrm{w}_{-i} = \mu_{-i}\right) \approx -\tfrac{1}{2}\Lambda_i \left(w_i - \mu_{\text{like},i}\right)^2 + \text{const}, \tag{2}$$

where,

$$\mu_{\text{like},i} = \arg\max_{w_i} \ \log \mathrm{P}\left(\mathrm{d}|\mathrm{w}_i = w_i, \mathrm{w}_{-i} = \mu_{-i}\right), \tag{3}$$

$$\Lambda_i = -\left.\frac{\partial^2 \log \mathrm{P}\left(\mathrm{d}|\mathrm{w}_i = w_i, \mathrm{w}_{-i} = \mu_{-i}\right)}{\partial^2 w_i}\right|_{w_i = \mu_{\text{like},i}}. \tag{4}$$

Here, both the mode, $\mu_{\text{like},i}$, and the negative Hessian, Λ_i, are random variables as they are deterministic functions of the minibatch, d, and the current estimate of all the other parameters, μ_{-i}, which are both random variables. As such, the gradient of the log-likelihood, which depends on Λ_i and μ_{-i}, is also random variable,

$$\mathrm{g}_i = \left.\frac{\partial \log \mathrm{P}\left(\mathrm{d}|\mathrm{w}_i = w_i, \mathrm{w}_{-i} = \mu_{-i}\right)}{\partial w_i}\right|_{w_i = \mu_i} = \Lambda_i \left(\mu_{\text{like},i} - \mu_i\right). \tag{5}$$

Our goal is to understand the distribution of the gradients, conditioned on the random variable Λ_i being equal to some specific value, λ_i. In practice, we will set Λ_i, using the standard approximations in the natural gradient literature (Zhang et al., 2017; Khan et al., 2017; 2018). To compute the expected gradient, we assume that the expected value of $\mu_{\text{like},i}$ is independent of Λ_i, and note that the expected gradient must be zero at $\mu_i = \mathrm{w}_i^*$, which gives,

$$\mathbb{E}\left[\mu_{\text{like},i}|\Lambda_i = \lambda_i, \mathrm{w}_i^* = w_i^*\right] = \mathbb{E}\left[\mu_{\text{like},i}|\mathrm{w}_i^* = w_i^*\right] = w_i^*, \tag{6}$$

substituting this into Eq .(5), we obtain,

$$\mathbb{E}\left[\mathrm{g}_i|\Lambda_i = \lambda_i, \mathrm{w}_i^* = w_i^*\right] = \lambda_i \left(w_i^* - \mu_i\right). \tag{7}$$

However, obtaining an analytic form for the full distribution of the gradient is more difficult. Given that all work in this domain makes fairly strong assumptions (e.g. Zhang et al., 2017; Khan et al., 2017; 2018), an alternative approach opens up: choosing a model for the gradients to match as

closely as possible any given approximation of the original likelihood in Eq. (2). In particular, if we take,

$$\mathrm{P}\left(\mathrm{g}_i = g_i | \Lambda_i = \lambda_i, \mathrm{w}_i^* = w_i^*\right) = \mathcal{N}\left(g_i; \lambda_i\left(w_i^* - \mu_i\right), \lambda_i\right). \tag{8}$$

then for any given approximation to the original log-likelihood (Eq. 2), the log-likelihood induced by conditioning on the gradient (Eq. 8) has the same form,

$$\log \mathrm{P}\left(\mathrm{g}_i = g_i | \Lambda_i = \lambda_i, \mathrm{w}_i^* = w_i^*\right) + \text{const} \approx -\frac{\left(g_i - \lambda_i\left(w_i^* - \mu_i\right)\right)^2}{2\lambda_i} = -\tfrac{1}{2}\lambda_i\left(w_i^* - \mu_{\text{like},i}\right). \tag{9}$$

The only material difference between the right-hand-side of this expression and the original likelihood (Eq. 2) is that here the parameter of interest is the optimal weight, w_i^*, as opposed to the weight in the underlying generative model, w_i. Also note that here we have specified the values for all the random variables: $\mathrm{g}_i = g_i$ and $\Lambda_i = \lambda_i$ which fix the value of $\mu_{\text{like},i}$ to $\mu_{\text{like}} = \frac{g_i}{\lambda_i} + \mu_i$.

The key difference between the methods is that in the original problem, where we infer a single distribution over all weights jointly, the underlying weights, $\mathbf{w}$, were fixed, and as such, Bayesian filtering reduces to recursively applying Bayes theorem, which does not give rise to interesting dynamics. In contrast, when we consider N separate inference problems, in which w_i^* is inferred from the gradient, then we are forced to introduce dynamics, because w_i^* varies over time, as it depends on all the other parameters in the network, μ_{-i}, which are also being optimized (Fig. 1A). However, the full generative process in Fig. 1A is very difficult to reason about, as it requires us to integrate over all possible trajectories for the other parameters, μ_{-i}. Instead, we write down simplified approximate dynamics over w_i^* directly (Fig. 1B), and evaluate the quality of these simplified dynamics empirically.

3 Bayesian (Kalman) filtering as adaptive SGD

Note that in Fig. 1B, w_i^* has arbitrary length temporal dependencies, which are difficult to specify or to perform inference over. Instead, we define a new random variable, $\mathbf{z}_i$, incorporating w_i^*, whose generative process is Markovian (Fig. 1C). For Adam, $\boldsymbol{z}_i$ will have two elements representing a parameter, w_i^* and the associated momentum, p_i, whereas for RMSprop it will have only one element representing just the parameter,

$$\boldsymbol{z}_{\text{RMSprop},i}(t) = \left(w_i^*(t)\right) \qquad\qquad \boldsymbol{z}_{\text{Adam},i}(t) = \begin{pmatrix} w_i^*(t) \\ p_i(t) \end{pmatrix} \tag{10}$$

As $\boldsymbol{z}_i$ represents the optimal setting for a parameter it will change over time, and we assume a simple Gaussian form for these changes,

$$\mathrm{P}\left(\boldsymbol{z}_i(t) | \boldsymbol{z}_i(t-1)\right) = \mathcal{N}\left(\boldsymbol{z}_i(t); \left(\boldsymbol{I} - \boldsymbol{A}\right)\boldsymbol{z}_i(t-1), \boldsymbol{Q}\right), \tag{11}$$

where $\boldsymbol{Q}$ is the covariance of Gaussian perturbations and $\boldsymbol{A}$ is the dynamics matrix incorporating weight decay and momentum. We can write the second-order Taylor expansion of the likelihood as a function of the expanded latent, $\boldsymbol{z}_i$ (Eq. 10),

$$\log \mathrm{P}\left(g_i | w_i^*\right) = \log \mathrm{P}\left(g_i | \boldsymbol{z}_i\right) \approx -\tfrac{1}{2}\left(\boldsymbol{z}_i - \boldsymbol{\mu}_{\text{like},i}\right)^T \boldsymbol{\Lambda}_i \left(\boldsymbol{z}_i - \boldsymbol{\mu}_{\text{like},i}\right), \tag{12}$$

where have omitted time indices for clarity,

$$g_i = g_i(t) \qquad w_i^* = w_i^*(t) \qquad \boldsymbol{z}_i = \boldsymbol{z}_i(t) \qquad \boldsymbol{\mu}_{\text{like},i} = \boldsymbol{\mu}_{\text{like},i}(t) \qquad \boldsymbol{\Lambda}_i = \boldsymbol{\Lambda}_i(t).$$

This likelihood is equivalent to our original likelihood (Eq. 9) because the gradients depend only on $\mathrm{w}_i^*(t)$. As such, the negative Hessians must take the form,

$$\boldsymbol{\Lambda}_{\text{RMSprop, i}} = \left(\Lambda_i\right) \qquad\qquad \boldsymbol{\Lambda}_{\text{Adam, i}} = \begin{pmatrix} \Lambda_i & 0 \\ 0 & 0 \end{pmatrix}. \tag{13}$$

The Gaussian prior and approximate likelihood allows us to use standard two-step Kalman filter updates. First, we propagate the previous time-step's posterior, $\mathrm{P}\left(\boldsymbol{z}_i(t-1)|\mathcal{D}(t-1)\right)$, with mean $\boldsymbol{\mu}_{\text{post},i}(t-1)$ and covariance $\boldsymbol{\Sigma}_{\text{post},i}(t-1)$, forward in time to obtain a prior at time t,

$$\begin{aligned}\mathrm{P}\left(\boldsymbol{z}_i(t)|\mathcal{G}(t-1)\right) &= \int \boldsymbol{d}\boldsymbol{z}_i(t-1)\, \mathrm{P}\left(\boldsymbol{z}_i(t)|\boldsymbol{z}_i(t-1)\right) \mathrm{P}\left(\boldsymbol{z}_i(t-1)|\mathcal{G}_i(t-1)\right) \\ &= \mathcal{N}\left(\boldsymbol{z}_i(t); \boldsymbol{\mu}_{\text{prior},i}(t), \boldsymbol{\Sigma}_{\text{prior},i}(t)\right),\end{aligned}$$

where,

$$\boldsymbol{\mu}_{\text{prior},i} = \boldsymbol{\mu}_{\text{prior},i}(t) = (\boldsymbol{I} - \boldsymbol{A})\, \boldsymbol{\mu}_{\text{post},i}(t-1), \tag{14a}$$

$$\boldsymbol{\Sigma}_{\text{prior},i} = \boldsymbol{\Sigma}_{\text{prior},i}(t) = (\boldsymbol{I} - \boldsymbol{A})\, \boldsymbol{\Sigma}_{\text{post},i}(t-1)\, (\boldsymbol{I} - \boldsymbol{A})^T + \boldsymbol{Q}, \tag{14b}$$

and where $\mathcal{G}_i(t-1) = \{g_i(1), g_i(2), \ldots, g_i(t-1)\}$ is all gradients up to time $t-1$. Note that we have also defined abbreviations for $\boldsymbol{\mu}_{\text{prior},i}(t)$ and $\boldsymbol{\Sigma}_{\text{prior},i}(t)$, omitting the temporal index, t. Second we use Bayes theorem to incorporate new data,

$$\mathrm{P}\left(\boldsymbol{z}_i(t)|\mathcal{D}(t)\right) = \frac{\mathrm{P}\left(g_i(t)|\boldsymbol{z}_i(t)\right)\mathrm{P}\left(\boldsymbol{z}_i(t)|\mathcal{G}(t-1)\right)}{\mathrm{P}\left(g_i(t)|\mathcal{G}(t-1)\right)} = \mathcal{N}\left(\boldsymbol{z}_i(t); \boldsymbol{\mu}_{\text{post},i}(t), \boldsymbol{\Sigma}_{\text{post},i}(t)\right).$$

where,

$$\boldsymbol{\Sigma}_{\text{post},i} = \boldsymbol{\Sigma}_{\text{post},i}(t) = \left(\boldsymbol{\Sigma}_{\text{prior},i}^{-1} + \boldsymbol{\Lambda}_i\right)^{-1}, \tag{15a}$$

$$\boldsymbol{\mu}_{\text{post},i} = \boldsymbol{\mu}_{\text{post},i}(t) = \boldsymbol{\mu}_{\text{prior},i} + \boldsymbol{\Sigma}_{\text{post},i}\boldsymbol{\Lambda}_i\left(\boldsymbol{\mu}_{\text{like},i} - \boldsymbol{\mu}_{\text{prior},i}\right). \tag{15b}$$

Thus far, we have simply repeated standard Kalman filtering results. To relate Kalman filtering to gradient ascent, we compute the gradient of Eq. (12) at $\boldsymbol{z}_i = \boldsymbol{\mu}_{\text{prior},i}$,

$$\boldsymbol{g}_i = \left.\frac{\partial \log \mathrm{P}\left(g_i|\boldsymbol{z}\right)}{\partial \boldsymbol{z}_i}\right|_{\boldsymbol{z}_i = \boldsymbol{\mu}_{\text{prior},i}} = \boldsymbol{\Lambda}\left(\boldsymbol{\mu}_{\text{like},i} - \boldsymbol{\mu}_{\text{prior},i}\right). \tag{16}$$

Note that as the log-likelihood depends only on w_i^*, we have,

$$\boldsymbol{g}_{\text{RMSprop},i} = (g_i) \qquad\qquad \boldsymbol{g}_{\text{Adam},i} = \begin{pmatrix} g_i \\ 0 \end{pmatrix}. \tag{17}$$

Now, we identify this gradient (Eq. 16) in the mean updates (Eq. 15b),

$$\boldsymbol{\mu}_{\text{post},i} = \boldsymbol{\mu}_{\text{prior},i} + \boldsymbol{\Sigma}_{\text{post},i}\, \boldsymbol{g}_i. \tag{18}$$

This form is extremely intuitive: it states that the uncertainty should be used to precondition gradient updates, such that the updates are larger when there is more uncertainty, and smaller when past data gives high confidence in the current estimates.

As the precision is always rank-1 (Eq. 13), we can always write it as,

$$\boldsymbol{\Lambda}_i = \boldsymbol{e}_i\boldsymbol{e}_i^T, \qquad \text{where} \qquad \boldsymbol{e}_{\text{RMSprop},i} = (e_i) \qquad \boldsymbol{e}_{\text{Adam},i} = \begin{pmatrix} e_i \\ 0 \end{pmatrix} \tag{19}$$

As such, the updates for the posterior covariance (Eq. 15a) can be rewritten using the Sherman Morrison formula (Hager, 1989),

$$\boldsymbol{\Sigma}_{\text{post},i} = \boldsymbol{\Sigma}_{\text{prior},i} - \frac{\boldsymbol{\Sigma}_{\text{prior},i}\boldsymbol{e}_i\boldsymbol{e}_i^T\boldsymbol{\Sigma}_{\text{prior},i}}{1 + \boldsymbol{e}_i^T\boldsymbol{\Sigma}_{\text{prior},i}\boldsymbol{e}_i}. \tag{20}$$

To estimate $\boldsymbol{e}_i$, we use the Fisher information (see SI Sec. 9.7). In particular, we could use the gradient itself, but could also (under weaker approximations) use the centred gradient (Graves, 2013), or the gradient for data sampled under the prior.

4 Bayesian RMSprop (BRMSprop)

Here, we develop a Bayesian variant of RMSprop, which we call BRMSprop. We consider each parameter to be inferred by considering a separate Bayesian inference problem, so the latent variable, $z = w^*$, is a single scalar, representing a single parameter (we omit the index i on z for brevity). We use $A = \eta^2/(2\sigma^2)$ and $Q = \eta^2$ giving a dynamical prior,

$$\mathrm{P}\left(z(t+1)|z(t)\right) = \mathcal{N}\left(\left(1 - \tfrac{\eta^2}{2\sigma^2}\right) z(t), \eta^2\right), \tag{21}$$

such that the stationary distribution over z has standard-deviation σ. To obtain a complete description of our updates, we substitute these choices into the updates for the prior (Eq. 14) and the posterior (Eq. 18 and Eq. 20),

$$\mu_{\text{prior}} = \left(1 - \tfrac{\eta^2}{2\sigma^2}\right)\mu_{\text{post}}(t-1) \qquad\qquad \mu_{\text{post}} = \mu_{\text{prior}} + \sigma_{\text{post}}^2 g, \tag{22}$$

$$\sigma_{\text{prior}}^2 = \left(1 - \tfrac{\eta^2}{2\sigma^2}\right)^2\sigma_{\text{post}}^2(t-1) + \eta^2 \qquad\qquad \sigma_{\text{post}}^2 = \sigma_{\text{prior}}^2\left(1 - \frac{\sigma_{\text{prior}}^2 e^2}{1 + \sigma_{\text{prior}}^2 e^2}\right) \tag{23}$$

For an efficient implementation of the full algorithm, see SI Alg. 1.

4.1 RECOVERING RMSPROP FROM BRMSPROP

Now we show that with in steady state, BRMSprop closely approximates RMSprop. Making this comparison is non-trivial because the "additional" variables in RMSprop and BRMSprop (i.e. the average squared gradient and the uncertainty respectively) are not directly comparable. However, the implied learning rate *is* directly comparable. We therefore look at the learning rate when the average squared gradient and the uncertainty have reached steady-state. In particular we use the simplified covariance updates derived in SI Sec. 9.1,

$$\sigma^2_{\text{post}}(t+1) \approx \sigma^2_{\text{post}}(t)\left(1 - \sigma^2_{\text{post}}(t)e^2(t)\right) + \eta^2.$$

As $t \to \infty$, we expect σ^2_{post} to reach steady-state, at which point, $\sigma^2_{\text{post}} = \sigma^2_{\text{post}}(t) = \sigma^2_{\text{post}}(t+1)$. Solving for σ^2_{post},

$$\sigma^2_{\text{post}} \approx \frac{\eta}{\sqrt{\langle e^2 \rangle}}.$$

Substituting this form into Eq. (22) recovers the root-mean-square normalization used in RMSprop.

5 BAYESIAN ADAM (BADAM)

We now develop a Bayesian variant of Adam (Graves, 2013; Kingma & Ba, 2015), which we call BAdam. To introduce momentum into our Bayesian updates, we introduce an auxiliary momentum variable, $\mathrm{p}(t)$, corresponding to each parameter, $\mathrm{w}^*(t)$,

$$\mathbf{z}(t) = \begin{pmatrix} \mathrm{w}^*(t) \\ \mathrm{p}(t) \end{pmatrix},$$

then we infer both the parameter and momentum jointly. Under the prior, the momentum, $\mathrm{p}(t)$, evolves through time independently of $\mathrm{w}^*(t)$, obeying an AR(1) (or equivalently a discretised Ornstein-Uhlenbeck) process, with decay η_{p} and injected noise variance η_{p}^2,

$$\mathrm{p}(t+1) = (1 - \eta_{\mathrm{p}})\,\mathrm{p}(t) + \eta_{\mathrm{p}}\,\xi_{\mathrm{p}}(t), \tag{24}$$

where $\xi_{\mathrm{p}}(t)$ is standard Gaussian noise. This particular coupling of the injected noise variance and the decay rate ensures that the influence of the momentum on the weight is analogous to unit-variance Gaussian noise (see SI Sec. 9.2). The weight obeys similar dynamics, with the addition of a momentum-dependent term which in practice causes changes in $\mathrm{w}_i^*(t)$ to be correlated across time (i.e. multiple consecutive increases or decreases in $\mathrm{w}_i^*(t)$),

$$\mathrm{w}^*(t+1) = \left(1 - \tfrac{\eta^2 + \eta_{\mathrm{w}}^2}{2\sigma^2}\right)\mathrm{w}^*(t) + \eta \mathrm{p}(t) + \eta_{\mathrm{w}}\,\xi_{\mathrm{w}}(t), \tag{25}$$

where $\xi_{\mathrm{w}}(t)$ is again standard Gaussian noise, η is the strength of the momentum coupling, $\frac{\eta^2+\eta_{\mathrm{w}}^2}{2\sigma^2}$ is the strength of the weight decay, and η_{w}^2 is the variance of the noise injected to the weight. It is possible to write these dynamics in the generic form given above (Eq. 11), by using,

$$\boldsymbol{A} = \begin{pmatrix} \frac{\eta_{\mathrm{w}}^2+\eta^2}{2\sigma^2} & -\eta \\ 0 & \eta_{\mathrm{p}} \end{pmatrix} \qquad \boldsymbol{Q} = \begin{pmatrix} \eta_{\mathrm{w}}^2 & 0 \\ 0 & \eta_{\mathrm{p}}^2 \end{pmatrix}, \tag{26}$$

and these settings fully determine the updates, according to Eq. (14) and Eq. (15). For an efficient implementation of the full algorithm, see SI Alg. 2.

5.1 RECOVERING ADAM FROM BADAM

Now we show that with suitable choices for the parameters, BAdam closely approximates Adam. In particular, we compare the updates for the (posterior) parameter estimate, μ_{w}, and the momentum, μ_{p}, when we eliminate weight decay by setting $\sigma^2 = \infty$, and eliminate noise injected directly into the weight by setting $\eta_{\mathrm{w}} = 0$,

$$\mu_{\mathrm{w}}(t+1) = \mu_{\text{post,w}}(t+1) = \mu_{\mathrm{w}}(t) + \eta\mu_{\mathrm{p}}(t) + \Sigma_{\mathrm{ww}}g(t), \tag{27a}$$

$$\mu_{\mathrm{p}}(t+1) = \mu_{\text{post,p}}(t+1) = (1 - \eta_{\mathrm{p}})\,\mu_{\mathrm{p}}(t) + \Sigma_{\mathrm{wp}}g(t). \tag{27b}$$

These updates depend on two quantities, Σ_{ww} and Σ_{wp}, which are related to $\langle e^2 \rangle$, but have no direct analogue in standard Adam. As such, to make a direct comparison, we use the same approach as we used previously for RMSprop: we compare the updates for the parameter and momentum, when Σ_{ww}, Σ_{wp} in BAdam and $\langle e^2 \rangle$ in Adam have reached their steady-state values. To find the steady-states for Σ_{ww} and Σ_{wp}, we again use the simplified covariance updates derived in SI Sec. 9.1,

$$\boldsymbol{Q} \approx \boldsymbol{A}\boldsymbol{\Sigma} + \boldsymbol{\Sigma}\boldsymbol{A}^T + \boldsymbol{\Sigma}\langle \boldsymbol{e}\boldsymbol{e}^T \rangle \boldsymbol{\Sigma}. \tag{28}$$

Substituting Eq. (26) and Eq. (19) into Eq. (28), and again using $\sigma^2 = \infty$ and $\eta_{\text{w}}^2 = 0$, we obtain,

$$\begin{pmatrix} 0 & 0 \\ 0 & \eta_{\text{p}}^2 \end{pmatrix} \approx \eta_{\text{p}} \begin{pmatrix} 0 & \Sigma_{\text{wp}} \\ \Sigma_{\text{wp}} & 2\Sigma_{\text{pp}} \end{pmatrix} - \eta \begin{pmatrix} 2\Sigma_{\text{wp}} & \Sigma_{\text{pp}} \\ \Sigma_{\text{pp}} & 0 \end{pmatrix} + \langle e^2 \rangle \begin{pmatrix} \Sigma_{\text{ww}}^2 & \Sigma_{\text{ww}}\Sigma_{\text{wp}} \\ \Sigma_{\text{ww}}\Sigma_{\text{wp}} & \Sigma_{\text{wp}}^2 \end{pmatrix}. \tag{29}$$

We now assume that the data is informative, in the sense that it is strong enough to give a narrow posterior relative to the prior (without which any neural network training algorithm is unlikely to be able to obtain good performance). This implies $\Sigma_{\text{pp}} \ll \eta_{\text{p}}/2$ (see SI Sec. 9.2), allowing us to solve for Σ_{wp}, using the lower-right element of the above expression (i.e. $\eta_{\text{p}}^2 \approx 2\eta_{\text{p}}\Sigma_{\text{pp}} + \langle e^2 \rangle \Sigma_{\text{wp}}^2$),

$$\Sigma_{\text{wp}} \approx \frac{\eta_{\text{p}}}{\sqrt{\langle e^2 \rangle}}. \tag{30}$$

To recover a very close approximation to Adam, we need a specific relationship between learning rates and evidence strength, such that,

$$\Sigma_{\text{ww}} \approx \frac{\eta\eta_{\text{p}}}{\sqrt{\langle e^2 \rangle}}.$$

While this may seem restrictive, it is only necessary to achieve the closest possible match between Bayesian filtering and plain Adam. However, our goal is not to match Adam exactly, given that Adam does not even converge (Reddi et al., 2018). Instead, our goal is to capture the essential dynamical insights of Adam in a Bayesian method. Indeed, we hope that the resulting Bayesian method constitutes an improvement over Adam, which implies that it must exhibit some differences. Nonetheless, focusing on the regime where filtering and Adam are most similar, the filtering updates become,

$$\mu_{\text{w}}(t+1) \approx \mu_{\text{w}}(t) + \eta\mu_{\text{p}}(t) + \eta\eta_{\text{p}} \frac{g(t)}{\sqrt{\langle e^2 \rangle}} \approx \mu_{\text{w}}(t) + \eta\mu_{\text{p}}(t+1) \tag{31a}$$

$$\mu_{\text{p}}(t+1) \approx (1-\eta_{\text{p}})\,\mu_{\text{p}}(t) + \eta_{\text{p}} \frac{g(t)}{\sqrt{\langle e^2 \rangle}}, \tag{31b}$$

where we have substituted for $\mu_{\text{p}}(t+1)$ into Eq. 31a, which assumes that $\eta_{\text{p}} \ll 1$. This is very close to the Adam updates, except that the root-mean-square normalization is in the momentum updates, rather than the parameter updates. To move the location of the root-mean-square normalization, we rewrite the updates in terms of a rescaled momentum,

$$\tilde{\mu}_{\text{p}}(t+1) = \mu_{\text{p}}(t+1)\sqrt{\langle e^2 \rangle} \tag{32}$$

giving,

$$\mu_{\text{w}}(t+1) \approx \mu_{\text{w}}(t) + \eta \frac{\tilde{\mu}_{\text{p}}(t+1)}{\sqrt{\langle e^2 \rangle}}, \tag{33a}$$

$$\tilde{\mu}_{\text{p}}(t+1) \approx (1-\eta_{\text{p}})\,\tilde{\mu}_{\text{p}}(t) + \eta_{\text{p}} g(t), \tag{33b}$$

which recovers Adam. See SI Sec. 9.6 for a similar discussion for NAG/NAdam

6 EXPERIMENTS

We compared Bayesian and standard methods on MNIST. In particular, we trained a CNN with relu nonlinearities and maxpooling for 50 epochs. The network had two convolutional layers with 10 channels in the first layer and 20 in the second, with 5×5 convolutional kernels, followed by a single fully connected layer with 50 units, and was initialized with draws from a Gaussian with

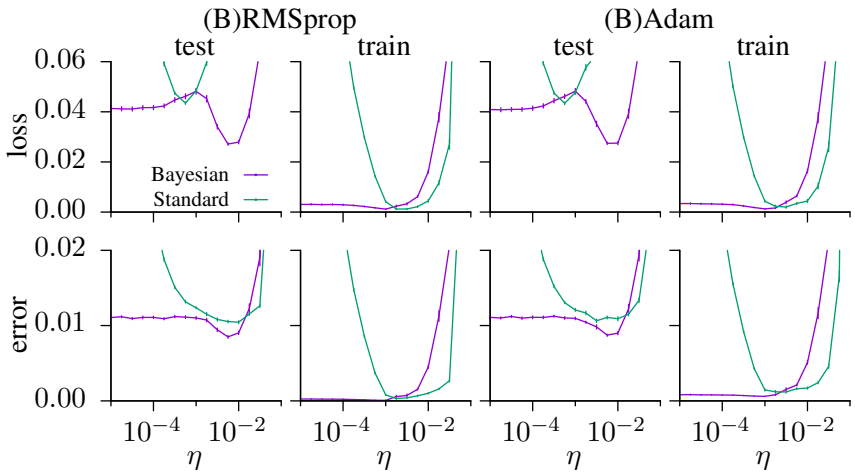

Figure 2: Comparing RMSprop and BRMSprop on MNIST. The vertical bars represent one standard error and are based on 30 independent runs. Adam and BAdam both use $\eta_{\text{w}} = 0.1$.

variance $2/N_{\text{inputs}}$. The network did not use dropout (or any other form of stochastic regularisation), which would increase the variance of the gradients, artificially inflating $\langle e^2 \rangle$.

The key Bayesian-specific parameters are the variance of the stationary distribution and the initial uncertainty; both of which were set to $1/(2N_{\text{inputs}})$. In principle, they should be similar to the initialization variance, but in practice we found that using a somewhat lower value gave better performance, though this needs further investigation. For the RMSprop and Adam specific parameters, we used the PyTorch defaults.

Comparing the Bayesian and non-Bayesian methods, we do not expect to see very large discrepancies, because they are approximately equivalent in steady-state. Nonetheless, the Bayesian methods show considerably lower test loss, somewhat lower classification error, and similar training loss and training error (Fig. 2). The local maximum in test-loss may arise because we condition on each datapoint multiple times, which is not theoretically justified (correcting this is non-trivial in the dynamical setting, so we leave it for future work).

7 Features of Bayesian stochastic gradient descent

Here we summarise the features of Bayesian stochastic gradient descent schemes, pointing out where we recover previously known best practice and where our approach suggests new ideas.

7.1 Weight decay, L2 regularization and Bayesian priors

Loshchilov & Hutter (2017) recently examined different forms of weight decay in adaptive SGD methods such as Adam. In particular, they asked whether or not the root-mean-square normalizer should be applied to the weight decay term. In common practice, we take weight decay as arising from the gradient of an L2 regularizer, in which case it is natural to normalize the gradient of the objective and the gradient of the regularizer in the same way. However, there is an alternative: to normalize only the gradient of the objective, but to keep the weight decay constant (in which case, weight decay cannot be interpreted as arising from an L2 normalizer). Loshchilov & Hutter (2017) show that this second method, which they call AdamW, gives better test accuracy than the standard approach. Remarkably, AdamW arises naturally in BRMSprop and BAdam (e.g. Eq. 22), providing a potential explanation for its improved performance.

7.2 NESTEROV ACCELERATED GRADIENTS/WEIGHT DECAY

In Nesterov accelerated gradients (NAG), we compute the gradient at a "predicted" location formed by applying a momentum update to the current parameters (Nesterov, 1983). These updates arise naturally in our Bayesian scheme, as the required gradient term (Eq. 16) is evaluated at $\boldsymbol{\mu}_{\text{prior}}$ (Eq. 14a), which is precisely a prediction formed by combining the current setting of the parameters, $\boldsymbol{\mu}_{\text{post}}(t)$, with momentum and decay, embodied in $\boldsymbol{A}$ (Eq. 26) to form a prediction, $\boldsymbol{\mu}_{\text{prior}}(t+1)$. Interestingly, as we also implement weight decay through the dynamics matrix, $\boldsymbol{A}$, we should also apply the updates from weight decay before computing the gradient, giving, to our knowledge, a novel method that we denote "Nesterov accelerated weight decay" (NAWD).

7.3 CONVERGENCE

A series of recent papers have discussed the convergence properties of RMSprop and Adam, noting that they may fail to converge (Wilson et al., 2017; Reddi et al., 2018) if the exponentially decaying average over the squared gradients is computed with a too-small timescale (Reddi et al., 2018).

Our method circumvents these issues by coupling the learning rate to the timescale over which the implicit exponential moving average for the mean-square gradient is performed. As such, in the limit as the learning rates go to zero (i.e. $\eta \to 0$ for BRMSprop and $\eta_{\text{w}} \to 0$ and $\eta_{\text{p}} \to 0$ for BAdam), our method becomes SGD with adaptive learning rates that scale as $1/t$ and is therefore likely to be convergent for convex functions (Robbins & Monro, 1951; Bottou, 1998), though we leave a rigorous proof to future work.

7.4 UPDATING THE PRECONDITIONER BEFORE APPLYING THE UPDATE

ADAM (Kingma & Ba, 2015) first updates the root-mean-square gradient normalizer before computing the parameter update. This is important, because it ensures that updates are bounded in the pathological case that gradients are initially all very small, such that the root-mean-square normalizer is also small, and then there is a single large gradient. Bayesian filtering naturally recovers this choice as the gradient preconditioner in Eq. (18) is the posterior, rather than the prior, covariance (i.e. updated with the current gradient).

7.5 CENTERED GRADIENTS

BRMSprop and BAdam suggest that just as centred gradients (or gradients based on labels sampled from the current model) are used in work on natural gradients, they should be used in RMSprop and Adam (and indeed, this is sometimes done, see Graves, 2013).

7.6 COMBINE MOMENTUM WITH DIRECT GRADIENT

To recover Adam from BAdam we needed to set $\Sigma_{\text{ww}} \leftarrow 0$. As such, BAdam differs from Adam in that it resembles a linear combination of RMSprop and Adam, suggesting that such linear combinations might provide fruitful directions for future investigation.

7.7 AUTOMATICALLY SETTING THE WEIGHT DECAY

A final advantage of BRMSprop and BAdam is that they eliminate the need to set a separate weight decay parameter: the weight decay is specified by the stationary variance, σ^2, which is very close to that used by standard initialization schemes, and the learning rate, η.

8 DISCUSSION

Here, we developed BRMSprop and BAdam, Bayesian versions of RMSprop and Adam respectively. These methods naturally recover many features of state-of-the-art adaptive SGD methods, including the root-mean-square normalizer, Neseterov acceleration and AdamW. As such, BRMSprop and BAdam provide a possible explanation for the empirical success of RMSprop, Adam, Nesterov acceleration and AdamW. Experimentally, we find, BRMSprop and BAdam are superior

to the standard methods in the tasks that we examined. Finally, the novel interpretation of adaptive SGD as filtering in a dynamical model opens several avenues for future research, including new stochastic regularisation methods and combinations with Kronecker factorisation.

We believe that it should be possible to combine the temporal changes induced by the factorised approximations with other types of inference, including probabilistic backpropagation (Hernández-Lobato & Adams, 2015) or assumed density filtering (Ghosh et al., 2016). At present, our method bears closest relation to natural gradient variational inference methods (Zhang et al., 2017; Khan et al., 2018), as they also use the Fisher Information to approximate the likelihood. Indeed, our method becomes equivalent to these approaches in the limit as we send the learning rate, η to zero. The key difference is that because they do not formulate a factorised generative model, they are unable to provide a strong justification for the introduction of rich dynamics, and they are unable to reason about optimal inference under these dynamics.

Bayesian filtering presents a novel approach to neural network optimization, and as such, there are variety of directions for future work. First, Bayesian filtering converts the problem of neural network optimization into the statistical problem of understanding the dynamics of changes in the optimal weight induced by optimization in the other parameters. In particular, we can perform an empirical investigation in large scale systems, or attempt to find closed-form expressions for the dynamics in simplified domains such as linear regression. Second, here we wrote down a statistical model for the gradient. However, there are many circumstances where the gradient is not available. Perhaps a low precision or noisy gradient is available due to noise in the parameters (e.g. due to dropout Srivastava et al., 2014), or perhaps we wish to consider a biological setting, where the gradient is not present at all (Aitchison et al., 2014). The Bayesian approach presented here gives a straightforward recipe for developing (Bayes) optimal algorithms for such problems. Third, stochastic regularization has been shown to be extremely effective at reducing generalization error in neural networks. This Bayesian interpretation of adaptive SGD methods presents opportunities for new stochastic regularization schemes. Fourth, it should be possible to develop filtering methods that represent the covariance of a full weight matrix by exploiting Kronecker factorisation (Martens & Grosse, 2015; Grosse & Martens, 2016; Zhang et al., 2017).

REFERENCES

Laurence Aitchison, Alex Pouget, and Peter E Latham. Probabilistic synapses. *arXiv preprint arXiv:1410.1029*, 2014.

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

Yoshua Bengio, Nicolas Boulanger-Lewandowski, and Razvan Pascanu. Advances in optimizing recurrent networks. In *Acoustics, Speech and Signal Processing*, pp. 8624–8628. IEEE, 2013.

Léon Bottou. Online learning and stochastic approximations. *On-line learning in neural networks*, 17(9):142, 1998.

Michael B Cohen, Jelena Diakonikolas, and Lorenzo Orecchia. On acceleration with noise-corrupted gradients. *arXiv preprint arXiv:1805.12591*, 2018.

Timothy Dozat. Incorporating nesterov momentum into ADAM. In *International Conference on Learning Representations Workshop*, 2016.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

Lee A Feldkamp, Danil V Prokhorov, and Timothy M Feldkamp. Simple and conditioned adaptive behavior from kalman filter trained recurrent networks. *Neural Networks*, 16(5-6):683–689, 2003.

Soumya Ghosh, Francesco Maria Delle Fave, and Jonathan S Yedidia. Assumed density filtering methods for learning bayesian neural networks. In *AAAI*, pp. 1589–1595, 2016.

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

Roger Grosse and James Martens. A kronecker-factored approximate fisher matrix for convolution layers. In *International Conference on Machine Learning*, pp. 573–582, 2016.

William W Hager. Updating the inverse of a matrix. *SIAM review*, 31(2):221–239, 1989.

José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning*, pp. 1861–1869, 2015.

Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Overview of mini-batch gradient descent. *COURSERA: Neural Networks for Machine Learning: Lecture 6a*, 2012.

Mohammad Emtiyaz Khan, Zuozhu Liu, Voot Tangkaratt, and Yarin Gal. Vprop: Variational inference using rmsprop. *arXiv preprint arXiv:1712.01038*, 2017.

Mohammad Emtiyaz Khan, Didrik Nielsen, Voot Tangkaratt, Wu Lin, Yarin Gal, and Akash Srivastava. Fast and scalable bayesian deep learning by weight-perturbation in adam. *arXiv preprint arXiv:1806.04854*, 2018.

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

Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. *arXiv preprint arXiv:1711.05101*, 2017.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pp. 2408–2417, 2015.

Yurii Nesterov. A method for unconstrained convex minimization problem with the rate of convergence $O(1/k^2)$. In *Doklady AN USSR*, volume 269, pp. 543–547, 1983.

Yann Ollivier. Online natural gradient as a kalman filter. *arXiv preprint arXiv:1703.00209*, 2017.

Gintaras V Puskorius and Lee A Feldkamp. Decoupled extended kalman filter training of feedforward layered networks. In *Neural Networks, 1991., IJCNN-91-Seattle International Joint Conference on*, volume 1, pp. 771–777. IEEE, 1991.

Gintaras V Puskorius and Lee A Feldkamp. Neurocontrol of nonlinear dynamical systems with kalman filter trained recurrent networks. *IEEE Transactions on neural networks*, 5(2):279–297, 1994.

Gintaras V Puskorius and Lee A Feldkamp. Parameter-based kalman filter training: theory and implementation. In *Kalman filtering and neural networks*. 2001.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. *ICLR*, 2018.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, pp. 400–407, 1951.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*, volume 16, pp. 3776–3784, 2016.

S Sha, F Palmieri, and M Datum. Optimal filtering algorithms for fast learning in feedforward neual networks. *Neural Networks*, 1992.

Samuel L Smith, Daniel Duckworth, Quoc V Le, and Jascha Sohl-Dickstein. Stochastic natural gradient descent draws posterior samples in function space. *arXiv preprint arXiv:1806.09597*, 2018.

Yang Song and Stefano Ermon. Accelerating natural gradient with higher-order invariance. *arXiv preprint arXiv:1803.01273*, 2018.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Héctor J Sussmann. Uniqueness of the weights for minimal feedforward nets with a given input-output map. *Neural networks*, 5(4):589–593, 1992.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pp. 1139–1147, 2013.

Ashia C Wilson, Rebecca Roelofs, Mitchell Stern, Nati Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning. In *Advances in Neural Information Processing Systems*, pp. 4148–4158, 2017.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

Guodong Zhang, Shengyang Sun, David Duvenaud, and Roger Grosse. Noisy natural gradient as variational inference. *arXiv preprint arXiv:1712.02390*, 2017.

9 Supplementary Information

9.1 Approximate covariance updates

To recover previously known update rules (e.g. RMSprop and Adam), it is usually useful to collapse the two covariance updates (Eq. 14b and Eq. 20) into a single update, which we simplify by neglecting various third-order terms. If we naively substitute Eq. 14b into Eq. 20, we obtain,

$$\begin{aligned}\boldsymbol{\Sigma}_{\text{post}}(t+1) = &\left((\boldsymbol{I}-\boldsymbol{A})\,\boldsymbol{\Sigma}_{\text{post}}\,(\boldsymbol{I}-\boldsymbol{A})^T + \boldsymbol{Q}\right) \\ &- \frac{\left((\boldsymbol{I}-\boldsymbol{A})\,\boldsymbol{\Sigma}_{\text{post}}\,(\boldsymbol{I}-\boldsymbol{A})^T + \boldsymbol{Q}\right)\boldsymbol{e}\boldsymbol{e}^T\left((\boldsymbol{I}-\boldsymbol{A})\,\boldsymbol{\Sigma}_{\text{post}}\,(\boldsymbol{I}-\boldsymbol{A})^T + \boldsymbol{Q}\right)^T}{1+\boldsymbol{e}^T\left((\boldsymbol{I}-\boldsymbol{A})\,\boldsymbol{\Sigma}_{\text{post}}\,(\boldsymbol{I}-\boldsymbol{A})^T + \boldsymbol{Q}\right)\boldsymbol{e}}.\end{aligned}$$

We have,

$$\boldsymbol{A} \sim \eta \qquad\qquad \boldsymbol{Q} \sim \eta^2.$$

where η is the learning rate, which is assumed to be small. Note that for Adam we have $\boldsymbol{A} \sim \eta$ but for RMSprop we have, $\boldsymbol{A} \sim \eta^2$. As we are taking the limit of small learning rates, we use the pessimistic case of $\boldsymbol{A} \sim \eta$ for the following derivations, and we can subsequently introduce a smaller scaling for $\boldsymbol{A}$. To obtain a self-consistent solution, we need,

$$\boldsymbol{\Sigma}_{\text{post}} \sim \mathcal{O}(\eta).$$

As $\boldsymbol{\Sigma}_{\text{post}}$ is $\mathcal{O}(\eta)$, and updates to $\boldsymbol{\Sigma}_{\text{post}}$ are $\mathcal{O}(\eta^2)$, we can neglect $\mathcal{O}(\eta^3)$ terms,

$$\boldsymbol{\Sigma}_{\text{post}}(t+1) = \boldsymbol{\Sigma}_{\text{post}} - \boldsymbol{A}\boldsymbol{\Sigma}_{\text{post}} - \boldsymbol{\Sigma}_{\text{post}}\boldsymbol{A}^T + \boldsymbol{Q} - \boldsymbol{\Sigma}_{\text{post}}\boldsymbol{e}\boldsymbol{e}^T\boldsymbol{\Sigma}_{\text{post}}.$$

which gives the required simplified form required for Adam. Further, note that for RMSprop, $\boldsymbol{A}$ is actually $\mathcal{O}(\eta^2)$, and therefore we are able to obtain an even simpler form,

$$\boldsymbol{\Sigma}_{\text{post}}(t+1) = \boldsymbol{\Sigma}_{\text{post}} + \boldsymbol{Q} - \boldsymbol{\Sigma}_{\text{post}}\boldsymbol{e}\boldsymbol{e}^T\boldsymbol{\Sigma}_{\text{post}}.$$

9.2 SETTING THE MOMENTUM DECAY

In the main text, we briefly note that we couple the injected noise and decay in Eq. (24) such that the effect of $p(t)$ on $w(t)$ is analogous to that of unit-variance Gaussian noise. In particular, we require that the stationary variance of $w(t)$ is again σ^2, despite the momentum term in Eq. (25) being treated as if it was unit-variance Gaussian noise. To show that the stationary variance of $w(t)$ is indeed σ^2, we solve the Lyapunov equation for the stationary covariance, $\boldsymbol{\Psi}$,

$$\boldsymbol{\Psi} = (\boldsymbol{I} - \boldsymbol{A})\,\boldsymbol{\Psi}\,(\boldsymbol{I} - \boldsymbol{A})^T + \boldsymbol{Q},$$

where we do not condition on data. Neglecting the second-order term in $\boldsymbol{A}$,

$$\boldsymbol{Q} \approx \boldsymbol{A}\boldsymbol{\Psi} + (\boldsymbol{A}\boldsymbol{\Psi})^T, \tag{34}$$

and as,

$$\boldsymbol{A}\boldsymbol{\Psi} = \begin{pmatrix} \frac{\eta^2+\eta_\text{w}^2}{2\sigma^2} & -\eta \\ 0 & \eta_\text{p} \end{pmatrix} \begin{pmatrix} \Psi_\text{ww} & \Psi_\text{wp} \\ \Psi_\text{wp} & \Psi_\text{pp} \end{pmatrix} = \begin{pmatrix} \frac{\eta^2+\eta_\text{w}^2}{2\sigma^2}\Psi_\text{ww} - \eta\Psi_\text{wp} & \frac{\eta^2+\eta_\text{w}^2}{2\sigma^2}\Psi_\text{wp} - \eta\Psi_\text{pp} \\ \eta_\text{p}\Psi_\text{wp} & \eta_\text{p}\Psi_\text{pp} \end{pmatrix}.$$

Eq. (34) becomes,

$$\begin{pmatrix} \eta_\text{w}^2 & 0 \\ 0 & \eta_\text{p}^2 \end{pmatrix} \approx \begin{pmatrix} 2\left(\frac{\eta^2+\eta_\text{w}^2}{2\sigma^2}\Psi_\text{ww} - \eta\Psi_\text{wp}\right) & \frac{\eta^2+\eta_\text{w}^2}{2\sigma^2}\Psi_\text{wp} - \eta\Psi_\text{pp} + \eta_\text{p}\Psi_\text{wp} \\ \frac{\eta^2+\eta_\text{w}^2}{2\sigma^2}\Psi_\text{wp} - \eta\Psi_\text{pp} + \eta_\text{p}\Psi_\text{wp} & 2\eta_\text{p}\Psi_\text{pp} \end{pmatrix} \tag{35}$$

Now, we can solve for Ψ_pp using the ppth element of Eq. (35),

$$\Psi_\text{pp} = \frac{\eta_\text{p}}{2},$$

and we can solve for Ψ_wp using the wpth element of Eq. (35), in combination with our solution to Ψ_pp,

$$\Psi_\text{wp} = \frac{\eta\eta_\text{p}/2}{\eta_\text{p} + \frac{\eta^2+\eta_\text{w}^2}{2\sigma^2}} \approx \eta/2.$$

This approximation becomes exact in the limit we consider to recover Adam (i.e. $\sigma^2 = \infty$), but it also holds when σ^2 is finite, because η, η_w and η_p are all smaller than 1, and η_p — the "learning rate" for the momentum — is usually larger than η and η_w (in fact, η_p can be as high as 0.1). Finally, we can solve for Ψ_ww using the wwth element of Eq. (35),

$$\Psi_\text{ww} \approx \sigma^2,$$

as required.

9.3 EFFICIENT IMPLEMENTATION OF BRMSPROP

See Alg. 1 for a complete specification of our approach. There are two important points to note. First, the method is as memory efficient as RMSprop, as the updates for Σ and μ can be done in-place, so there is no need to maintain separate μ_prior and μ_post and σ^2_prior and σ^2_post in memory. Second, we simplified the covariance updates in Eq. (23),

$$\sigma^2_\text{post} = \sigma^2_\text{prior}\left(1 - \frac{\sigma^2_\text{prior}e^2}{1 + \sigma^2_\text{prior}e^2}\right) = \frac{\sigma^2_\text{prior}}{1 + \sigma^2_\text{prior}e^2}. \tag{36}$$

9.4 EFFICIENT IMPLEMENTATION OF BADAM

See Alg. 2 for a complete specification of our approach. There are two important points to note. First, we have carefully ordered the updates so that they can all be done in-place, improving memory efficiency. Second, we have simplified the derivation, by defining,

$$\boldsymbol{T} = \boldsymbol{I} - \boldsymbol{A}. \tag{37}$$

Algorithm 1 BRMSprop

Require: η ▷ Learning rate
Require: η_{g} ▷ Learning rate for average gradient (0.01)
Require: σ^2 ▷ Prior variance ($1/(2N_{\text{inputs}})$)
$\Sigma \leftarrow \sigma^2$ ▷ Initialize uncertainties
$\mu \sim \mathcal{N}(0, 2/N_{\text{inputs}})$ ▷ Initialize parameters
$\hat{g} \leftarrow 0$ ▷ Initialize average gradient
while not converged **do** ▷ $\Sigma = \sigma^2_{\text{prior}}(t)$, $\mu = \mu_{\text{prior}}(t)$
  $g \leftarrow \nabla \mathcal{L}_t(\mu)$ ▷ Gradients of minibatch t
  $\Sigma \leftarrow \Sigma/(1 + (g - \hat{g})^2 \Sigma)$ ▷ $\Sigma = \sigma^2_{\text{post}}(t)$
  $\mu \leftarrow \mu + \Sigma g$ ▷ $\mu = \mu_{\text{post}}(t)$
  $\hat{g} \leftarrow (1 - \eta_{\text{g}})\, \hat{g} + \eta_{\text{g}} g$ ▷ Update average gradient
  $\Sigma \leftarrow \left(1 - \eta^2/(2\sigma^2)\right)^2 \Sigma + \eta^2$ ▷ $\Sigma = \sigma^2_{\text{prior}}(t+1)$
  $\mu \leftarrow \left(1 - \eta^2/(2\sigma^2)\right) \mu$ ▷ $\mu = \mu_{\text{prior}}(t+1)$
end while
return μ

9.5 Trick to allow learning rates to vary across parameters

We require that the initialization for Σ varies across parameters, according to the number of inputs (as in typical neural network initialization schemes). While this is possible in automatic differentiation libraries, including PyTorch, it is extremely awkward. As such, we reparameterise the network, such that all parameters are initialized with a draw from a standard Gaussian, and to ensure that the outputs have the same scale, we explicitly scale the output.

9.6 Nesterov Accelerated Gradient

To begin, we note that accelerated stochastic gradient descent remains an open research problem, in which the optimality of plain NAG is unclear (Cohen et al., 2018). As such, our goal is again, not to match NAG exactly, but to capture its essential insights within a Bayesian framework, so that we can suggest improved methods. To highlight the link between our approach and Nesterov accelerated gradient, we rewrite Eq. (31) in terms of $v(t+1) = \eta \mu_{\text{p}}(t+1)$,

$$\mu_{\text{w}}(t) \approx \mu_{\text{w}}(t-1) + v(t) \tag{38a}$$

$$v(t) \approx (1 - \eta_{\text{p}})\, v(t-1) + \eta \eta_{\text{p}} \frac{g(t)}{\sqrt{\langle e^2 \rangle}}, \tag{38b}$$

which results in updates with a very similar form to standard momentum and Nesterov accelerated gradient (e.g. Sutskever et al., 2013), with the addition of root-mean-square normalization for the gradient. The key difference between momentum and Nesterov accelerated gradient is where we evaluate the gradient: for momentum, we evaluate the gradient at $\mu_{\text{w}}(t-1)$ (i.e. at $\mu_{\text{post}}(t-1)$), and for Nesterov accelerated gradient, we evaluate the gradient at a "predicted" location (i.e. at $\mu_{\text{prior}}(t)$),

$$g_{\text{Mom}}(t) = g\left(\mu_{\text{w}}(t-1)\right) = g\left(\mu_{\text{post, w}}(t-1)\right) \tag{39a}$$

$$g_{\text{NAG}}(t) = g\left(\mu_{\text{w}}(t-1) + (1 - \eta_{\text{p}}) v(t-1)\right) \approx g\left(\mu_{\text{w}}(t-1) + v(t-1)\right) = g\left(\mu_{\text{prior, w}}(t)\right) \tag{39b}$$

As noted in Eq. (16), Bayesian filtering requires us to evaluate the gradient at μ_{prior}, implying that we use updates based on NAG (Eq. 39b), rather than updates based on standard momentum (Eq. 39a).

9.7 Estimating the negative Hessian

One approach that works well in practice in natural-gradient methods (Amari, 1998; Martens & Grosse, 2015; Grosse & Martens, 2016; Song & Ermon, 2018; Smith et al., 2018) is to use a Fisher-

Algorithm 2 BAdam

Require: η ▷ Learning rate
Require: η_{g} ▷ Learning rate for average gradient (0.01)
Require: η_{p} ▷ Learning rate for momentum (0.1)
Require: η_{w} ▷ Learning rate for weight alone (0)
Require: σ^2 ▷ Prior variance $(1/(2N_{\text{inputs}}))$
$T_{\text{ww}} = 1 - (\eta^2 + \eta_{\text{w}}^2)/(2\sigma^2)$ ▷ Set dynamics matrix
$T_{\text{wp}} = \eta$ ▷ Set dynamics matrix
$T_{\text{pp}} = 1 - \eta_{\text{p}}$ ▷ Set dynamics matrix

$\Sigma_{\text{ww}} \leftarrow \sigma^2$ ▷ Initialize uncertainties
$\Sigma_{\text{wp}} \leftarrow 0$ ▷ Initialize uncertainties
$\Sigma_{\text{pp}} \leftarrow 0$ ▷ Initialize uncertainties
$\mu_{\text{w}} \sim \mathcal{N}(0, 2/N_{\text{inputs}})$ ▷ Initialize parameter
$\mu_{\text{p}} \leftarrow 0$ ▷ Initialize momentum
$\hat{g} \leftarrow 0$ ▷ Initialize average gradient
while not converged **do**
  $g \leftarrow \nabla \mathcal{L}_t(\mu_{\text{w}})$ ▷ Gradients of minibatch t
  $e^2 \leftarrow \nabla(g - \hat{g})^2$
  $\Sigma_{\text{ww}} \leftarrow \Sigma_{\text{ww}} - \Sigma_{\text{ww}}^2 e^2/(1 + \Sigma_{\text{ww}} e^2)$
  $\Sigma_{\text{wp}} \leftarrow \Sigma_{\text{wp}} - \Sigma_{\text{ww}}\Sigma_{\text{wp}} e^2/(1 + \Sigma_{\text{ww}} e^2)$ ▷ $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}_{\text{post}}(t)$
  $\Sigma_{\text{pp}} \leftarrow \Sigma_{\text{pp}} - \Sigma_{\text{wp}}^2 e^2/(1 + \Sigma_{\text{ww}} e^2)$
  $\mu_{\text{w}} \leftarrow \mu_{\text{w}} + \Sigma_{\text{ww}} g$ ▷ $\boldsymbol{\mu} = \boldsymbol{\mu}_{\text{post}}(t)$
  $\mu_{\text{p}} \leftarrow \mu_{\text{p}} + \Sigma_{\text{wp}} g$
  $\hat{g} \leftarrow (1 - \eta_{\text{g}})\,\hat{g} + \eta_{\text{g}} g$ ▷ Update average gradient
  $\Sigma_{\text{ww}} \leftarrow T_{\text{ww}}^2 \Sigma_{\text{ww}} + 2T_{\text{ww}}T_{\text{wp}}\Sigma_{\text{wp}} + T_{\text{wp}}^2\Sigma_{\text{pp}} + \eta_{\text{w}}^2$
  $\Sigma_{\text{wp}} \leftarrow T_{\text{ww}}T_{\text{pp}}\Sigma_{\text{wp}} + T_{\text{wp}}T_{\text{pp}}\Sigma_{\text{pp}}$ ▷ $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}_{\text{prior}}(t+1)$
  $\Sigma_{\text{pp}} \leftarrow T_{\text{pp}}^2\Sigma_{\text{pp}} + \eta_{\text{p}}^2$
  $\mu_{\text{w}} \leftarrow T_{\text{ww}}\mu_{\text{w}} + T_{\text{wp}}\mu_{\text{p}}$ ▷ $\boldsymbol{\mu} = \boldsymbol{\mu}_{\text{prior}}(t+1)$
  $\mu_{\text{p}} \leftarrow T_{\text{pp}}\mu_{\text{p}}$
end while
return μ

Information based estimate, of the negative Hessian,

$$\boldsymbol{\Lambda} \approx \mathrm{E}_{\mathrm{P}(d_{\text{model}}|\boldsymbol{z})}\left[-\frac{\partial^2}{\partial \boldsymbol{z}^2}\log \mathrm{P}\left(d_{\text{model}}|\boldsymbol{z}\right)\right] = \mathrm{E}_{\mathrm{P}(d_{\text{model}}|\boldsymbol{z})}\left[\boldsymbol{e}\boldsymbol{e}^T\right], \tag{40}$$

where

$$\boldsymbol{e} = \left.\frac{\partial \log \mathrm{P}\left(d_{\text{model}}|\boldsymbol{z}\right)}{\partial \boldsymbol{z}}\right|_{\boldsymbol{z}=\boldsymbol{\mu}_{\text{prior}}}. \tag{41}$$

Importantly, this Fisher-Information based estimate does not depend on actual data, d. Instead, it uses expectations (and gradients) taken over data sampled from the model, d_{model}, and this is fundamentally necessary for the the second equality in Eq. (40) to hold. As such, to use Eq. (40) we need to sample surrogate data from the model, and compute gradients with respect to that sampled data. Unfortunately, sampling surrogate data increases complexity and the required computational cost. As such, a simple low-cost alternative is to note that the key difference between $\boldsymbol{e}$ and $\boldsymbol{g}$ is that $\boldsymbol{e}$ has zero-mean, whereas $\boldsymbol{g}$ has non-zero mean, suggesting that we should correct the gradient by subtracting its mean,

$$\boldsymbol{e} \approx \boldsymbol{g} - \hat{\boldsymbol{g}}$$

where $\hat{\boldsymbol{g}}$ is an empirical estimate of the mean gradient.