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 Bayesian filtering problem. Inference in the Bayesian 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 amoungst 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.

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, this broad range of methods 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 returning to 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). In particular, they start with an independent Gaussian prior over the weights, and incrementally condition on each additional minibatch of data to form a series of increasingly narrow posterior distributions. In contrast, most current work on Bayesian neural networks does not have this incremental nature, and instead approximates the full posterior, conditioned on all data, in a single step (Watanabe & Tzafestas, 1990; Neal, 1992; MacKay, 1992; Blundell et al., 2015). The incremental nature of Bayesian filtering has the potential to be extremely powerful because it allows us to reason not just about the optimal final distribution, but also about the optimal updates formed by taking the posterior at the previous iteration, and conditioning on a single additional minibatch of data.

However, the typical approach to Bayesian filtering, where we infer a distribution over the optimal setting, $\boldsymbol{w}$, for all the parameters jointly, forces us to use extremely strong, factorised approximations, and it is legitimate to worry that these strong approximations might meaningfully disrupt the ability of Bayesian filtering to give close-to-optimal updates. As such, we instead consider an alternative problem that incorporates factorisation into the problem setting, and therefore requires fewer approximations downstream. In particular, instead of formulating one big Bayesian inference problem over all N parameters, $\boldsymbol{w}$, jointly, we instead formulate N different inference problems: one over each parameter, w_i, separately. Formally, the goal is to set each parameter estimate, μ_i, as close as possible to the unknown optimal value, $w_i(t)$, conditioned on the current estimate of all the other parameters, $\mu_{-i}(t)$,

$$w_i(t) = w_i(\mu_{-i}(t)) = \arg\max_{w_i'} \mathcal{L}(w_i'; \mu_{-i}(t)),$$

where $\mathcal{L}(w_i'; \mu_{-i}(t))$ is the objective function for the full problem, using w_i' for the ith parameter, and using $\mu_{-i}(t)$ for the other parameters. Critically, the optimal setting for the ith parameter, $w_i(t)$, given the other parameters, $\mu_{-i}(t)$, must depend on those other parameters (Fig. 1A). If these other parameters do not vary, then $w_i(t)$ does not vary, and this is a straightforward one-parameter

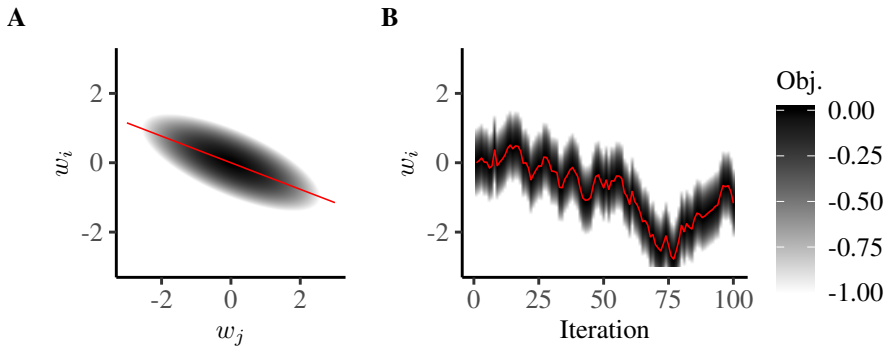

Figure 1: **A.** Grayscale: the objective plotted as a function of w_i and w_j, with all the other weights fixed. Red line: the ith optimal weight (i.e. the setting of w_i that maximizes the objective, conditioned on all the other weights), therefore changes with the current setting of w_j. **B.** Grayscale: As learning progresses, all the other weights change, inducing changes in the objective, when viewed as a function of w_i. Red line: Thus, the optimal weight (i.e. the setting of w_i that maximizes the objective, conditioned on all the other weights), changes in an effectively stochastic fashion over time.

Bayesian optimization problem. However, the other parameters, $\mu_{-i}(t)$, do change over time, as they are also being optimized, inducing changes over time in $w_i(t)$ (Fig. 1B). In essence, optimizing $\mu_{-i}(t)$ "moves the goalposts" for the problem of optimizing the ith parameter, and optimal Bayesian weight updates must take account of these moving goalposts. As such, when we break up the big inference problem over all N parameters jointly into N small inference problems over each parameter, w_i, separately, the dependencies between parameters that arise originally as posterior correlations instead get converted into changes over time, and accounting for these changes over time is necessary to recover effective optimization methods such as RMSprop and Adam.

1 Related work

It has been noted that under specific circumstances, natural gradient is approximate Bayesian filtering (Ollivier, 2017), allowing us to link Bayesian filtering to the rich literature on natural gradients (Amari, 1998; Martens & Grosse, 2015; Grosse & Martens, 2016; Smith et al., 2018). However, this only occurs when the dynamical prior in the Bayesian filtering problem has a specific form: the parameters being fixed over time (i.e. arguably an online data, rather than a true Bayesian filtering setting). While there have been attempts to use natural gradients to recover the Adam or RMSprop root-mean-square form for the gradient normalizer, in practice a different sum-square form emerges (Khan & Lin, 2017; Khan et al., 2018). In contrast, we show that to recover the Adam or RMSprop form for the gradient normalizer, it is necessary to use a more general dynamical prior under which the parameters change over time. Further, our approach is also able to recover additional variants such as Nesterov acceleration (Nesterov, 1983), and AdamW (Loshchilov & Hutter, 2017).

2 Results

We first 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), amoungst others.

2.1 Bayesian adaptive SGD

We perform inference over the latent variable $\boldsymbol{z}$. For RMSprop, $\boldsymbol{z}$ will have one element representing a single parameter, and for Adam it will have two elements representing a parameter and the

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

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

where $\boldsymbol{Q}$ is the covariance of Gaussian perturbations and $\boldsymbol{A}$ is weight decay, which is required to ensure that the weights remain bounded as $t \to \infty$. We approximate the likelihoods for each minibatch of data, $d(t)$, using a second-order Taylor expansion,

$$\log \mathrm{P}\left(d(t)|\boldsymbol{z}\right) \approx -\tfrac{1}{2}\left(\boldsymbol{z}-\boldsymbol{\mu}_{\text{like}}(t)\right)^T \boldsymbol{\Lambda}_{\text{like}}(t)\left(\boldsymbol{z}-\boldsymbol{\mu}_{\text{like}}(t)\right),$$

where $\boldsymbol{\mu}_{\text{like}}(t)$ is the mode (which we will identify implicitly using the gradient), and $\boldsymbol{\Lambda}_{\text{like}}(t)$ is a Fisher Information based estimate of the Hessian for that minibatch. 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, $\mathrm{P}\left(\boldsymbol{z}(t-1)|\mathcal{D}(t-1)\right)$, with mean $\boldsymbol{\mu}_{\text{post}}(t-1)$ and covariance $\boldsymbol{\Sigma}_{\text{post}}(t-1)$, forward in time to obtain a prior at time t,

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

where,

$$\boldsymbol{\mu}_{\text{prior}}(t) = \left(\boldsymbol{I}-\boldsymbol{A}\right)\boldsymbol{\mu}_{\text{post}}(t-1), \tag{2a}$$

$$\boldsymbol{\Sigma}_{\text{prior}}(t) = \left(\boldsymbol{I}-\boldsymbol{A}\right)\boldsymbol{\Sigma}_{\text{post}}(t-1)\left(\boldsymbol{I}-\boldsymbol{A}\right)^T + \boldsymbol{Q}, \tag{2b}$$

and where $\mathcal{D}(t-1) = \{d(1), d(2)...d(t-1)\}$ is all data up to time $t-1$. Second we use Bayes theorem to incorporate new data,

$$\begin{aligned}\mathrm{P}\left(\boldsymbol{z}(t)|\mathcal{D}(t)\right) &= \mathrm{P}\left(d(t)|\boldsymbol{z}(t)\right)\mathrm{P}\left(\boldsymbol{z}(t)|\mathcal{D}(t-1)\right)/\,\mathrm{P}\left(d(t)|\mathcal{D}(t-1)\right)\\ &= \mathcal{N}\left(\boldsymbol{z}(t); \boldsymbol{\mu}_{\text{post}}(t), \boldsymbol{\Sigma}_{\text{post}}(t)\right).\end{aligned}$$

where,

$$\boldsymbol{\Sigma}_{\text{post}} = \left(\boldsymbol{\Sigma}_{\text{prior}}^{-1} + \boldsymbol{\Lambda}_{\text{like}}\right)^{-1}, \tag{3a}$$

$$\boldsymbol{\mu}_{\text{post}} = \boldsymbol{\mu}_{\text{prior}} + \boldsymbol{\Sigma}_{\text{post}}\boldsymbol{\Lambda}_{\text{like}}\left(\boldsymbol{\mu}_{\text{like}} - \boldsymbol{\mu}_{\text{prior}}\right). \tag{3b}$$

Thus far, we have simply repeated standard Kalman filtering results. To relate Kalman filtering to gradient ascent, the key additional step is to identify the gradient,

$$\mathbf{g} = \left.\frac{\partial \log \mathrm{P}\left(d|\boldsymbol{z}\right)}{\partial \boldsymbol{z}}\right|_{\boldsymbol{z}=\boldsymbol{\mu}_{\text{prior}}} = \boldsymbol{\Lambda}_{\text{like}}\left(\boldsymbol{\mu}_{\text{like}} - \boldsymbol{\mu}_{\text{prior}}\right), \tag{4}$$

which gives a particularly simple form for the mean update,

$$\boldsymbol{\mu}_{\text{post}} = \boldsymbol{\mu}_{\text{prior}} + \boldsymbol{\Sigma}_{\text{post}}\mathbf{g}. \tag{5}$$

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.

The covariance updates are more problematic, as we need to estimate $\boldsymbol{\Lambda}_{\text{like}}(t)$. 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-Information based estimate,

$$\boldsymbol{\Lambda}_{\text{like}} \approx \mathrm{E}_{\mathrm{P}\left(d_{\text{model}}|\boldsymbol{z}\right)}\left[\frac{\partial^2}{\partial \boldsymbol{z}^2}\log \mathrm{P}\left(d_{\text{model}}|\boldsymbol{z}\right)\right] = \mathrm{E}_{\mathrm{P}\left(d_{\text{model}}|\boldsymbol{z}\right)}\left[\boldsymbol{e}\boldsymbol{e}^T\right], \tag{6}$$

where

$$\boldsymbol{e} = \left.\frac{\partial \log \mathrm{P}\left(d_{\text{model}}|\boldsymbol{z}\right)}{\partial \boldsymbol{z}}\right|_{\boldsymbol{z}=\boldsymbol{\mu}_{\text{prior}}}. \tag{7}$$

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. (6) to hold. As such, to use Eq. (6) 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,

$$\boldsymbol{e}(t) \approx \boldsymbol{g}(t) - \hat{\boldsymbol{g}}(t)$$

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

Using these Fisher-Information based estimates, we usually have a low-rank estimate of the precision of the likelihood. For simplicity, we consider a rank-1 estimate, in which case Eq. (3a) can be rewritten using the Sherman Morrison formula (Hager, 1989),

$$\boldsymbol{\Sigma}_{\text{post}} = \boldsymbol{\Sigma}_{\text{prior}} - \frac{\boldsymbol{\Sigma}_{\text{prior}} \boldsymbol{e}\boldsymbol{e}^T \boldsymbol{\Sigma}_{\text{prior}}}{1 + \boldsymbol{e}^T \boldsymbol{\Sigma}_{\text{prior}} \boldsymbol{e}}, \tag{8}$$

and we can use the Woodbury matrix identity if we wish to consider higher-rank updates.

We now apply these generic derivations to recover RMSprop and Adam, and variants such as AdamW and Nesterov acceleration.

2.2 Bayesian RMSprop (BRMSprop)

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

$$\mathrm{P}\left(z(t+1)|z(t)\right) = \mathcal{N}\left(\left(1 - \tfrac{\eta^2}{2\sigma^2}\right) z(t), \eta^2\right), \tag{9}$$

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. 2) and the posterior (Eq. 5 and Eq. 8),

$$\mu_{\text{prior}}(t) = \left(1 - \tfrac{\eta^2}{2\sigma^2}\right) \mu_{\text{post}}(t-1) \qquad \mu_{\text{post}} = \mu_{\text{prior}} + \sigma^2_{\text{post}} g, \tag{10}$$

$$\sigma^2_{\text{prior}}(t) = \left(1 - \tfrac{\eta^2}{2\sigma^2}\right)^2 \sigma^2_{\text{post}}(t-1) + \eta^2 \qquad \sigma^2_{\text{post}} = \sigma^2_{\text{prior}} \left(1 - \frac{\sigma^2_{\text{prior}} e^2}{1 + \sigma^2_{\text{prior}} e^2}\right) \tag{11}$$

where, as described above, e is either a centered gradient, or a gradient based on labels sampled from the prior. For an efficient implementation of the full algorithm, see SI Alg. 1.

2.2.1 Recovering RMSprop from BRMSprop

Now we show that with suitable choices for the parameters, 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 ignore weight decay by setting $\sigma^2 = \infty$, and use the simplified covariance updates derived in SI Sec. 7.1,

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

Now, we solve for the steady state posterior variance (and hence learning rate) by setting $\sigma^2_{\text{post}} = \sigma^2_{\text{post}}(t) = \sigma^2_{\text{post}}(t+1)$,

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

Substituting this form into Eq. (10) recovers the root-mean-square form for the gradient normalization used in RMSProp.

2.3 BAYESIAN ADAM (BADAM)

We now develop a Bayesian variant of Adam (Graves, 2013; Kingma & Ba, 2015), which we christen BAdam. To introduce momentum into our Bayesian updates, we introduce an auxilliary momentum variable, $p(t)$, corresponding to each parameter, $w(t)$,

$$\boldsymbol{z}(t) = \begin{pmatrix} w(t) \\ p(t) \end{pmatrix},$$

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

$$p(t+1) = (1-\eta_{\text{p}})\, p(t) + \eta_{\text{p}}\, \xi_{\text{p}}(t), \tag{12}$$

where $\xi_{\text{p}}(t)$ 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. 7.2). The weight obeys similar dynamics, with the addition of a momentum-dependent term which in practice causes changes in $w(t)$ to be correlated across time (e.g. multiple consecutive increases or decreases in $w(t)$),

$$w(t+1) = \left(1 - \tfrac{\eta^2+\eta_{\text{w}}^2}{2\sigma^2}\right) w(t) + \eta p(t) + \eta_{\text{w}}\, \xi_{\text{w}}(t), \tag{13}$$

where $\xi_{\text{w}}(t)$ is again standard Gaussian noise, η is the strength of the momentum coupling, $\frac{\eta^2+\eta_{\text{w}}^2}{2\sigma^2}$ 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. 1), by using,

$$\boldsymbol{A} = \begin{pmatrix} \frac{\eta_{\text{w}}^2+\eta^2}{2\sigma^2} & -\eta \\ 0 & \eta_{\text{p}} \end{pmatrix} \qquad \boldsymbol{Q} = \begin{pmatrix} \eta_{\text{w}}^2 & 0 \\ 0 & \eta_{\text{p}}^2 \end{pmatrix}, \tag{14}$$

And as the gradient of the objective with respect to the momentum is zero, we have,

$$\boldsymbol{e}(t) = \begin{pmatrix} e(t) \\ 0 \end{pmatrix}, \tag{15}$$

and these settings fully determine the updates, according to Eq. (2) and Eq. (3). For an efficient implementation of the full algorithm, see SI Alg. 2.

2.3.1 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 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 $\eta_{\text{w}} = 0$,

$$\mu_{\text{w}}(t+1) = \mu_{\text{w}}(t) + \eta\mu_{\text{p}}(t) + \Sigma_{\text{ww}} g(t), \tag{16a}$$

$$\mu_{\text{p}}(t+1) = (1-\eta_{\text{p}})\, \mu_{\text{p}}(t) + \Sigma_{\text{wp}} g(t). \tag{16b}$$

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. 7.1,

$$\boldsymbol{Q} \approx \boldsymbol{A}\boldsymbol{\Sigma} + \boldsymbol{\Sigma}\boldsymbol{A}^T + \boldsymbol{\Sigma}\langle \boldsymbol{e}\boldsymbol{e}^T \rangle \boldsymbol{\Sigma}. \tag{17}$$

Substituting Eq. (14) and Eq. (15) into Eq. (17), and again using $\sigma^2 = \infty$ and $\eta_{\text{w}}^2 = 0$, we obtain,

$$\begin{pmatrix} 0 & 0 \\ 0 & \eta_{\text{p}}^2 \end{pmatrix} \approx \eta_{\text{p}} \begin{pmatrix} 0 & \Sigma_{\text{wp}} \\ \Sigma_{\text{wp}} & 2\Sigma_{\text{pp}} \end{pmatrix} - \eta \begin{pmatrix} 2\Sigma_{\text{wp}} & \Sigma_{\text{pp}} \\ \Sigma_{\text{pp}} & 0 \end{pmatrix} + \langle e^2 \rangle \begin{pmatrix} \Sigma_{\text{ww}}^2 & \Sigma_{\text{ww}}\Sigma_{\text{wp}} \\ \Sigma_{\text{ww}}\Sigma_{\text{wp}} & \Sigma_{\text{wp}}^2 \end{pmatrix}. \tag{18}$$

We now assume that the data is strong enough to reduce the uncertainty in the momentum below its levels under the prior (i.e. $\Sigma_{\text{pp}} \ll \eta_{\text{p}}/2$; see SI Sec. 7.2), allowing us to solve for Σ_{wp}, using the ppth (i.e. lower right) element of the above expression,

$$\Sigma_{\text{wp}} \approx \frac{\eta_{\text{p}}}{\sqrt{\langle e^2 \rangle}}. \tag{19}$$

Using the value for Σ_{wp}, we can solve for Σ_{ww} using the wwth (i.e. upper left) element of Eq. (18),

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

However, in order to recover Adam, we need to drop this term (i.e. to set $\Sigma_{\text{ww}} \leftarrow 0$), but as it is not possible to justify this analytically, this constitutes a major difference between BAdam and Adam.

Thus, the updates become,

$$\mu_{\text{w}}(t+1) \approx \mu_{\text{w}}(t) + \eta\mu_{\text{p}}(t), \tag{20a}$$

$$\mu_{\text{p}}(t+1) \approx (1-\eta_{\text{p}})\,\mu_{\text{p}}(t) + \eta_{\text{p}}\frac{g(t)}{\sqrt{\langle e^2 \rangle}}. \tag{20b}$$

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,

$$\tilde{\mu}_{\text{p}}(t) = \mu_{\text{p}}(t)\sqrt{\langle e^2 \rangle} \tag{21}$$

giving,

$$\mu_{\text{w}}(t+1) \approx \mu_{\text{w}}(t) + \eta\frac{\tilde{\mu}_{\text{p}}(t)}{\sqrt{\langle e^2 \rangle}}, \tag{22a}$$

$$\tilde{\mu}_{\text{p}}(t+1) \approx (1-\eta_{\text{p}})\,\tilde{\mu}_{\text{p}}(t) + \eta_{\text{p}}g(t), \tag{22b}$$

which recovers Adam.

3 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 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).

4 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.

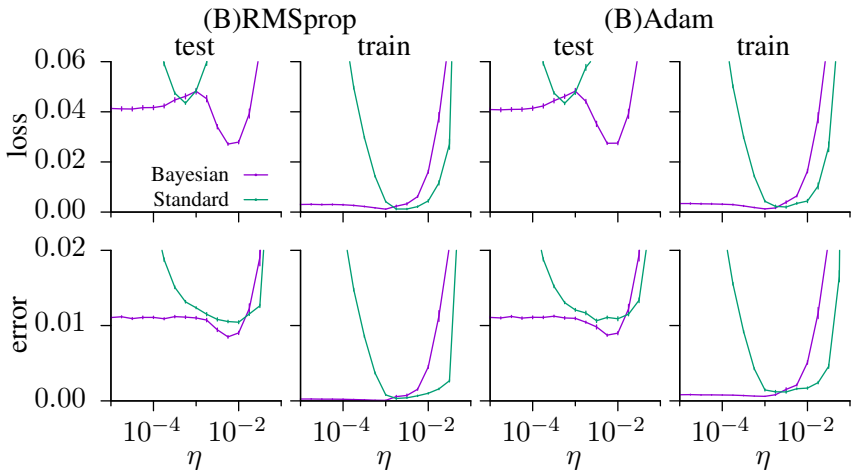

Figure 2: Comparing RMSprop and BRMSprop on MNIST. The vertical bars represent one standard error and are based on 30 independent runs. Adam and BAdam both use $\eta_{\mathrm{w}} = 0.1$.

4.1 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. 10), providing a potential explanation for its improved performance.

4.2 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. 4) is evaluated at $\boldsymbol{\mu}_{\mathrm{prior}}$ (Eq. 2a), which is precisely a prediction formed by combining the current setting of the parameters, $\boldsymbol{\mu}_{\mathrm{post}}(t)$, with momentum and decay, embodied in $\boldsymbol{A}$ (Eq. 14) to form a prediction, $\boldsymbol{\mu}_{\mathrm{prior}}(t+1)$. 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 christen "Nesterov accelerated weight decay" (NAWD).

4.3 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 $\eta_{\mathrm{w}} \to 0$ and $\eta_{\mathrm{p}} \to 0$ 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.

4.4 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. (5) is the posterior, rather than the prior, covariance (i.e. updated with the current gradient).

4.5 CENTERED GRADIENTS

BRMSprop and BAdam suggest that just as centered 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).

4.6 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.

4.7 AUTOMATICALLY SETTING THE WEIGHT DECAY

A final advantage of the Bayesian method is that it may 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.

5 FUTURE WORK

There are three natural directions for future work. First, the current BRMSprop and BAdam updates assume that each training datapoint is only presented once. While this is valid during the first epoch, it is not valid in subsequent epochs. As such, it is important to account for these effects to ensure that data is not "double-counted", which is non-trivial in this dynamical setting. Second, stochastic regularization (including dropout Srivastava et al., 2014) 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. Third, 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).

6 CONCLUSIONS

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.

REFERENCES

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.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. In *International Conference on Machine Learning*, pp. 1613–1622, 2015.

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

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.

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.

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 and Wu Lin. Conjugate-computation variational inference: Converting variational inference in non-conjugate models to inferences in conjugate models. *arXiv preprint arXiv:1703.04265*, 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.

David JC MacKay. A practical Bayesian framework for backpropagation networks. *Neural Computation*, 4:448–472, 1992.

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

Radford M Neal. Bayesian training of backpropagation networks by the hybrid Monte Carlo method. Technical report, Technical Report CRG-TR-92-1, Dept. of Computer Science, University of Toronto, 1992.

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.

Keigo Watanabe and Spyros G Tzafestas. Learning algorithms for neural networks with the Kalman filters. *Journal of Intelligent & Robotic Systems*, 3(4):305–319, 1990.

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.

7 Supplementary Information

7.1 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. 2b and Eq. 8) into a single update, which we simplify by neglecting various third-order terms. If we naively substitute Eq. 2b into Eq. 8, 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,

$$\boldsymbol{A} \in \mathcal{O}(\eta) \qquad\qquad \boldsymbol{Q} \in \mathcal{O}(\eta^2).$$

where η is the learning rate, which is assumed to be small. To obtain a self-consistent solution, we need,

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

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.

7.2 SETTING THE MOMEMTUM DECAY

In the main text, we briefly note that we couple the injected noise and decay in Eq. (12) 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. (13) 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 $\boldsymbol{A}$,

$$\boldsymbol{Q} \approx \boldsymbol{A}\boldsymbol{\Psi} + (\boldsymbol{A}\boldsymbol{\Psi})^T, \tag{23}$$

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. (23) becomes,

$$\begin{pmatrix} \eta_{\text{w}}^2 & 0 \\ 0 & \eta_{\text{p}}^2 \end{pmatrix} \approx \begin{pmatrix} 2\left(\frac{\eta^2+\eta_{\text{w}}^2}{2\sigma^2}\Psi_{\text{ww}} - \eta\Psi_{\text{wp}}\right) & \frac{\eta^2+\eta_{\text{w}}^2}{2\sigma^2}\Psi_{\text{wp}} - \eta\Psi_{\text{pp}} + \eta_{\text{p}}\Psi_{\text{wp}} \\ \frac{\eta^2+\eta_{\text{w}}^2}{2\sigma^2}\Psi_{\text{wp}} - \eta\Psi_{\text{pp}} + \eta_{\text{p}}\Psi_{\text{wp}} & 2\eta_{\text{p}}\Psi_{\text{pp}} \end{pmatrix} \tag{24}$$

Now, we can solve for Ψ_{pp} using the ppth element of Eq. (24),

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

and we can solve for Ψ_{wp} using the wpth element of Eq. (24), 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. (24),

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

as required.

7.3 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. (11),

$$\sigma^2_{\text{post}} = \sigma^2_{\text{prior}}\left(1 - \frac{\sigma^2_{\text{prior}}e^2}{1 + \sigma^2_{\text{prior}}e^2}\right) = \frac{\sigma^2_{\text{prior}}}{1 + \sigma^2_{\text{prior}}e^2}. \tag{25}$$

7.4 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,

$$\boldsymbol{T} = \boldsymbol{I} - \boldsymbol{A}. \tag{26}$$

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

7.5 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.

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 $(1/(2N_{\text{inputs}}))$
$T_{\mathrm{ww}} = 1 - (\eta^2 + \eta_{\mathrm{w}}^2)/(2\sigma^2)$ ▷ Set dynamics matrix
$T_{\mathrm{wp}} = \eta$ ▷ Set dynamics matrix
$T_{\mathrm{pp}} = 1 - \eta_{\mathrm{p}}$ ▷ Set dynamics matrix

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