A The Full MEKA Algorithm with Adaptive Step Sizes

Algorithm 1 MEKA with adaptive step sizes based on maximizing the probability of improvement.

```
**Hyperparameters:** decay rates $\beta_r$=0.999, $\beta_\Sigma$=0.999, $\beta_\alpha$=0.999
$m_0, P_0 \leftarrow \vec{0}, 10^4$                ▷ large variance initialization ensures first Kalman gain is one
$u_0, s_0 \leftarrow 0, 10^4$
$\delta_0 = \vec{0}$
$t \leftarrow 0$
repeat
  $t \leftarrow t+1$
  $f_t^{(i)}, \nabla f_t^{(i)}, \nabla^2 f_t^{(i)} \delta_{t-1} \leftarrow$ VectorizedMap($f, \{x_i\}_{i=1}^M; \theta_{t-1}$)    ▷ compute per-example quantities
  $y_t, r_t \leftarrow$ MeanVarEMA($\{f_t^{(i)}\}; \beta_r$)    ▷ exponential moving average (EMA) on the variances
  $g_t, \Sigma_t \leftarrow$ MeanVarEMA($\{\nabla f_t^{(i)}\}; \beta_\Sigma$)
  $b_t, Q_t \leftarrow$ MeanVar($\{\nabla^2 f_t^{(i)} \delta_{t-1}\}$)
  $u_t, s_t \leftarrow$ FilterUpdate($u_{t-1}, s_{t-1}; m_{t-1}, P_{t-1}, y_t, r_t, b_t, Q_t$)    ▷ filter update equations
  $m_t, P_t \leftarrow$ FilterUpdate($m_{t-1}, P_{t-1}; g_t, \Sigma_t, b_t, Q_t$)    ▷ filter update equations
  $\alpha_t \leftarrow \arg\min_\alpha$ (13)    ▷ with an EMA (decay rate $\beta_\alpha$) on the coefficients
  $\delta_t \leftarrow \alpha_t m_t$
  $\theta_t \leftarrow \theta_{t-1} - \delta_t$
until convergence
```

B The Function Value Dynamics Model

We discuss inferring the function value f_t, taking into account uncertainty due to changes in function value and observation noise during optimization. The gradient dynamics (5) imply the following dynamics model for the function value itself:

$$\begin{aligned} f_t \mid f_{t-1} &\sim \mathcal{N}(f_{t-1} + m_{t-1}^T \delta_{t-1} + \frac{1}{2}\delta_{t-1}^T B_t \delta_{t-1}, \\ &\qquad \lambda_t + \delta_{t-1}^T P_{t-1} \delta_{t-1} + \frac{1}{4}\delta_{t-1}^T Q_t \delta_{t-1}) \\ y_t \mid f_t &\sim \mathcal{N}(f_t, r_t) \end{aligned} \tag{16}$$

where we again use a quadratic approximation using Taylor expansion. Instead of the intractable ∇f and $\nabla^2 f$, we use the estimates from Section 2. The observations y_t and r_t are the empirical mean and variance of f_t from a minibatch. The variance terms in the dynamics model are due to the uncertainty associated with m_{t-1} and $B_t \delta_{t-1}$.

Here we have included a scalar term λ_t, which acts as a correction to the local quadratic model. This acts similar to a damping component, except we can automatically infer an optimal λ_t by maximizing the likelihood of $p(y_t \mid y_{1:t-1})$, with a closed form solution (see Appendix B.1). While damping terms are usually difficult to set empirically (Choi et al., 2019), we note that including our λ_t term is essentially free, and it automatically decays after optimization stabilizes.

B.1 Adaptively Correcting the Dynamics Model

We construct a dynamics model of the function value as follows (repeated for convenience):

$$\begin{aligned} f_t \mid f_{t-1} &\sim \mathcal{N}(f_{t-1} + m_{t-1}^T \delta_{t-1} + \frac{1}{2}\delta_{t-1}^T B_t \delta_{t-1},\ \lambda_t + \delta_{t-1}^T P_{t-1} \delta_{t-1} + \frac{1}{4}\delta_{t-1}^T Q_t \delta_{t-1}) \\ y_t \mid f_t &\sim \mathcal{N}(f_t, r_t) \end{aligned} \tag{17}$$

We include a scalar parameter λ_t in case the local quadratic approximation is inaccurate, ie. when y_t is significantly different from the predicted value. If this occurs, a high value of λ_t causes the Kalman gain to become large, throwing away the stale estimate and putting more weight on the new observed function value.

We pick a value for λ_t by maximizing the likelihood of $p(y_t|y_{1:t-1})$. Marginalizing over f_t, we get

$$p(y_t|y_{1:t-1}) = \mathcal{N}\left(\underbrace{u_{t-1} + m_{t-1}^T\delta_{t-1} + \frac{1}{2}\delta_{t-1}^T B_t \delta_{t-1}}_{u_t^-},\ \lambda_t + \underbrace{s_{t-1} + \delta_{t-1}^T P_{t-1}\delta_{t-1} + \frac{1}{4}\delta_{t-1}^T Q_t \delta_{t-1} + r_t}_{c_t}\right) \tag{18}$$

Taking the log and writing it out, we get

$$\log p(y_t|y_{1:t-1}) \propto -\frac{1}{2}\left[\frac{(y_t - u_t^-)^2}{\lambda_t + c_t} + \log(\lambda_t + c_t)\right] \tag{19}$$

and its derivative is

$$-\frac{1}{2}\left[\frac{-(y_t - u_t^-)^2}{(\lambda_t + c_t)^2} + \frac{1}{\lambda_t + c_t}\right] = -\frac{1}{2}\left[\frac{-(y_t - u_t^-)^2 + \lambda_t + c_t}{(\lambda_t + c_t)^2}\right] \tag{20}$$

Setting this to zero, we get

$$\lambda_t = (y_t - u_t^-)^2 - c_t \tag{21}$$

Since the role of λ_t is to ensure we are not overconfident in our predictions, and we don't want to deal with negative variance values, we set

$$\lambda_t^* = \max\{(y_t - u_t^-)^2 - c_t,\ 0\} \tag{22}$$

As can be seen from Figure 8, this λ_t term goes to zero when it is not needed, ie. when the dynamics model is correct, which occurs on MNIST after convergence. It is also a quantity that shows us just how incorrect our dynamics model is, and for the problems we tested, we find that it is significantly smaller than the posterior variance s_t. This suggests that it has minimal impact if removed, but we keep it in the algorithm for cases when a quadratic approximation is not sufficient.

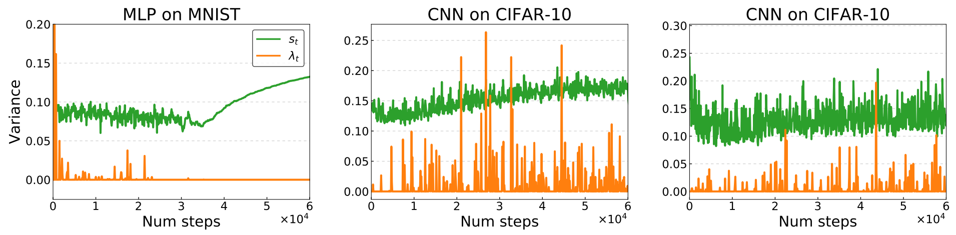
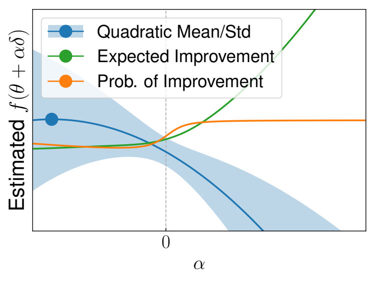


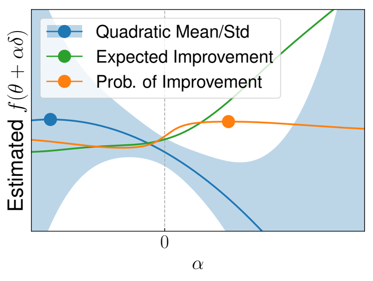
Figure 8: The quantity $2s_t + \lambda_t$ shows up as a constant variance term during step size adaptation. We find that though λ_t has an effect only during a few iterations, it is usually small enough to be ignored. This suggests that the quadratic approximation assumption is okay most of the time. Nevertheless, a self-correcting term that is essentially compute-free is a desirable component.

B.2 Dealing with negative curvature in step size adaptation

When the gradient estimating is pointing in a direction of negative curvature, there is a chance that the optimal step size is infinity (Figure 9a).



(a) Negative curvature.



(b) Negative curvature with $\lambda_t||\delta||^6$.

Figure 9: Negative curvature can result in infinite step sizes. An extra correction term to the variance ensures the optimal step size is finite.

This occurs when the variance of $f_{t+1} - f_t$ rises slower than the expectation. To handle this situation, we can add a third order correction term, which appears in the variance as a term that scales with $||\delta||^6$. Using the same procedure as inferring a constant λ_t, we can instead add the term $\lambda_t||\delta||^6$ to the variance. We then choose λ_t as

$$\lambda_t^* = \max\left\{\frac{1}{||\delta||^6}\left((y_t - u_t^-)^2 - c_t\right),\ 0\right\} \tag{23}$$

This extra term (if $\lambda_t > 0$) in the variance ensures that variance increases faster than the expectation. Adaptive step sizes based on the probability of improvement will then have an optimal step size that is finite in value (Figure 9b). An additional damping effect may be added by lower bounding λ_t. We did not fully test this approach as the exponential moving averaged curvature used in practice was always positive for our test problems. Incidentally, Figures 9a and 9b show that the expected improvement is not a good heuristic as it is extremely large even with this extra term. Moreover, the quadratic approximation will result in negative step sizes.

C Using the Current Hessian vs the Previous Hessian

For the dynamics model, we make the choice to use the Hessian at the updated location B_t, which is an unbiased estimate of $\nabla^2 f(\theta_t)$, instead of B_{t-1}, the Hessian at θ_{t-1}. Firstly, we made this choice for computational reasons: it is easier to compute the Hessian at θ_t since we are already computing the gradient g_t evaluated at θ_t. Secondly, we found that using the current Hessian results in better performance and more stability. Figure 10 shows this on MNIST. For CIFAR-10, we found that using the previous Hessian B_{t-1} resulted in immediate divergence and NaNs, so we do not show those plots.

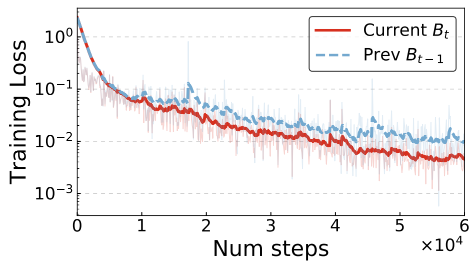


Figure 10: The choice of using current vs previous Hessian on MNIST.

D Comparison of Adaptive Step Size Schemes

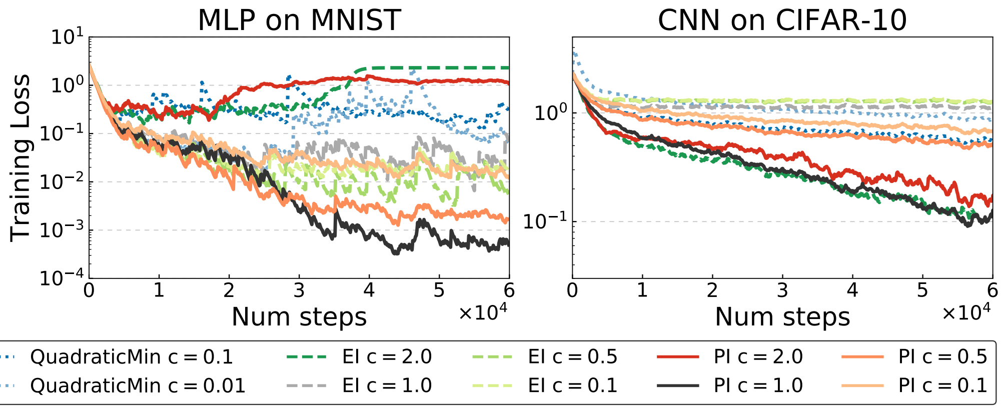


Figure 11: Comparing step sizes with a scaling factor c such that the update is $\theta_t = \theta_{t-1} + c\alpha_t\delta_t$. This comparison includes expected improvement (EI). We note that it performs poorly on MNIST and requires a scaling of 2.0 to match PI on CIFAR-10. Using a smaller scaling factor results in worse performance.

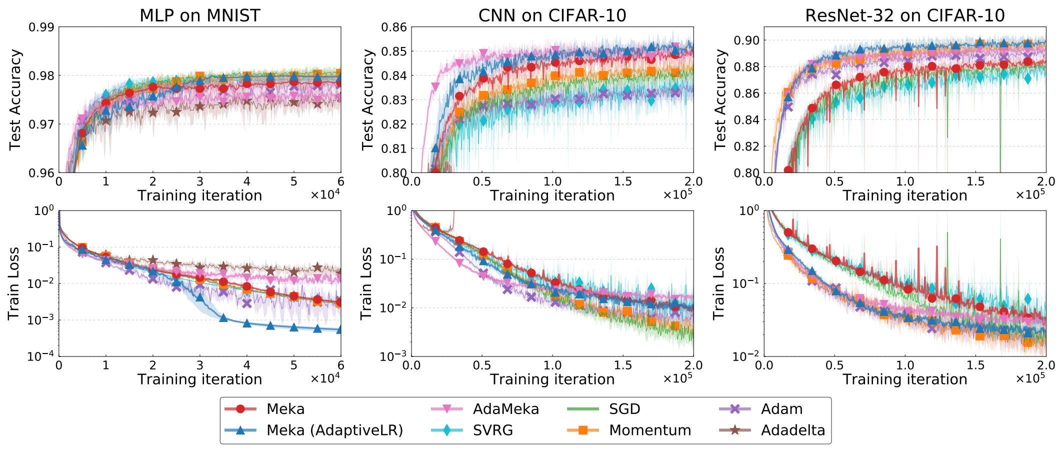


Figure 12: MEKA is competitive with optimizers that have additional tunable hyperparameters. Results are averaged over 5 random seeds; shaded regions are 5th and 95th percentiles.

D.1 Comparison with Tuned Optimizers

We measure the performance of our method against a variety of other approaches. We compare with fixed step size versions of SGD, SGD with momentum (Polyak, 1964), and ADAM (Kingma and Ba, 2014). For these, we tune the step size using grid search. We also compare against ADADELTA (Zeiler, 2012), which is a competing learning rate-free algorithm, and SVRG (Johnson and Zhang, 2013), an optimization algorithm focused around estimating the full batch gradient, using the same step size as the tuned SGD. For comparison, we implemented MEKA using the same constant learning rate as the tuned SGD, and ADAMEKA with the same learning rate as the tuned ADAM. We applied fully adaptive step sizes with MEKA update directions; on ResNet-32, we used ADAMEKA update directions to mitigate poor conditioning.

Figure 12 shows the resulting loss and accuracy curves. As this includes optimizers across a wide range of motivations, we highlight some specific comparisons. MEKA generally performs better than SGD in terms of test accuracy, showing that too much stochasticity can hurt generalization. Though both designed with gradient estimation in mind, MEKA seems to compare favorable against SVRG in terms of performance. We note that the per-iteration costs of MEKA were also cheaper as SVRG requires two gradient evaluations. With the default learning rate of 1.0, ADADELTA performs decently on MNIST but ends up diverging on CIFAR-10. In comparison, our adaptive step sizes converge well and generally outperform fixed step sized MEKA (or ADAMEKA) in both trainng loss and test accuracy.

E Experiment Details

E.1 Dataset Description

We used the official train and test split for MNIST and CIFAR-10. We did not do any data augmentation for MNIST. For CIFAR-10, we normalized the images by subtracting every pixel with the global mean and standard deviation across the training set. In addition to this, unless specified otherwise, we pre-processed the images following He et al. (2016a) by padding the images by 4 pixels on each side and applying random cropping and horizontal flips.

E.2 Architecture Description

All models use the the ReLU (Nair and Hinton, 2010) activation function.

MLP We used an MLP with 1 hidden layer of 100 hidden units.

CNN We used the same architecture as the "3c3d" architecture in Schneider et al. (2019), which consists of 3 convolutional layers with max pooling, followed by 3 fully connected layers. The first convolutional layer has a kernel size of 5×5 with stride 1, "valid" padding, and 64 filters. The second convolutional layer has a kernel size of 3×3 with stride 1, "valid" padding, and 96 filters. The third convolutional layer has a kernel size of 3×3 with stride 1, "same" padding, and 128 filters. The max pooling layers have a window size of 3×3 with stride 2. The 2 fully connected layers have 512 and 256 units respectively.

ResNet-32 Our ResNet-32 (He et al., 2016a) model uses residual blocks based on He et al. (2016b). We replaced the batch normalization layers with group normalization (Wu and He, 2018) as batch-dependent transformations conflict with our assumption that the gradient samples are independent, and hinder our method to estimate gradient variance.

E.3 Optimizer Comparisons Description

We tuned the step size of SGD in a grid of $\{0.001, 0.01, 0.1, 1.0\}$. We tuned the step size of SGD with momentum, and ADAM in a grid of $\{0.0001, 0.001, 0.01, 0.1\}$. We chose the best step size of SGD for the variant of MEKA with a constant learning rate, and for SVRG.

The chosen step size for SGD was 0.1 for MNIST, and 0.1 for CIFAR-10. For SGD with momentum, the chosen step size was 0.01 for MNIST, 0.1 for ResNet-32 on CIFAR-10, and 0.001 for CNN on CIFAR-10. The best step size for ADAM was 0.001 for MNIST and ResNet-32 on CIFAR-10, and 0.0001 for CNN on CIFAR-10.

For SGD with momentum, the momentum coefficient γ was fixed to 0.9. For ADAM, β_1, β_2, ε were fixed to 0.9, 0.999, and 10^{-8} respectively. For ADADELTA, ρ and ε were fixed to 0.95 and 10^{-6} respectively.

F Sensitivity of MEKA's Hyperparameters

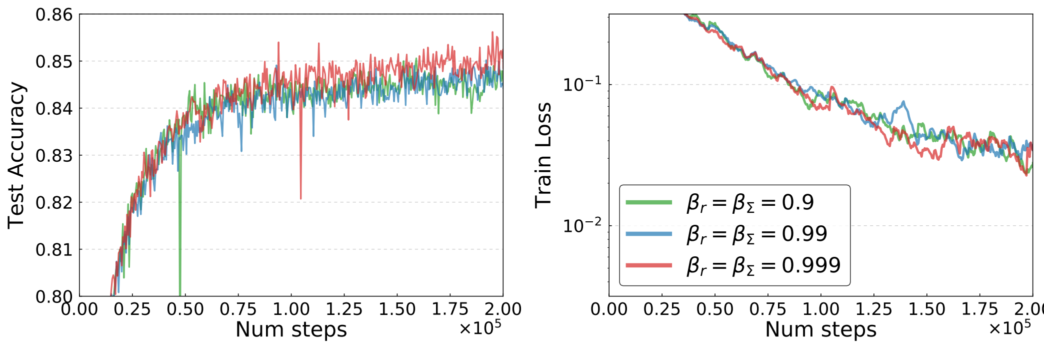


Figure 13: The performance of MEKA with constant learning rate for CNN on CIFAR-10 is not sensitive to the choice of the exponential moving average decay rates β_r and β_Σ.

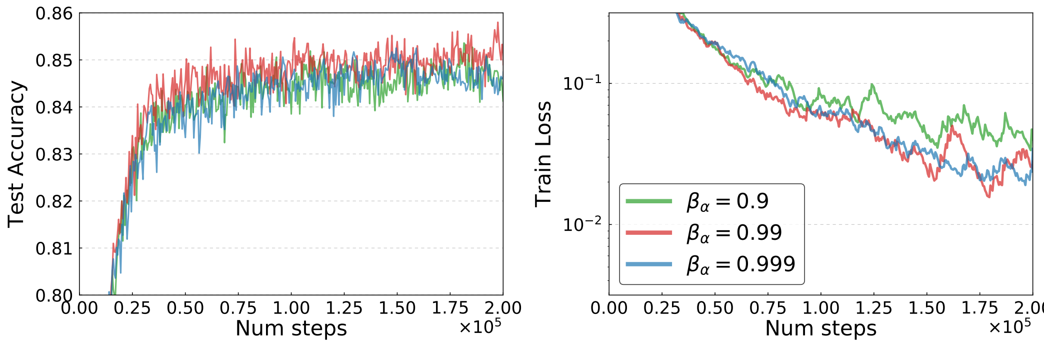


Figure 14: The performance of MEKA with the PI adaptive scheme for CNN on CIFAR-10 is not sensitive to the choice of the exponential moving average decay rate β_α.

G Additional Cost of MEKA

Table 1: The memory cost of using a vectorized map to obtain individual gradients is greater than taking the gradient of a sum over the minibatch, whereas the asymptotic compute cost is the same. B is the batch size, $|\theta|$ is the number of parameters, and D is the number of activations in the model.

	$\nabla \sum_{i=1}^{m} f_t^{(i)}$	$\text{VMap}(\nabla, f_t^{(i)})$
Memory	$\mathcal{O}(\lvert\theta\rvert + BD)$	$\mathcal{O}(B(\lvert\theta\rvert + D))$
Compute	$\mathcal{O}(BD\lvert\theta\rvert)$	$\mathcal{O}(BD\lvert\theta\rvert)$

Table 2: The ratio of the time it takes to complete one iteration for MEKA versus SGD. Note that in addition to the vector mapped gradients, we also compute an additional Hessian-vector product. The runtimes are after just-in-time compilation of JAX has settled. Runtimes are tested on the NVIDIA TITAN Xp GPU.

Dataset	Architecture	SGD	Meka (fixed lr)	Meka (PI adaptive)
MNIST	MLP	1.00	1.10	1.00
CIFAR-10	CNN	1.00	0.83	1.34
CIFAR-10	ResNet-32	1.00	1.68	2.97

H Proofs

Proof of Proposition 1. A standard Lipschitz bound yields

$$\begin{aligned}\mathbb{E}[f_{t+1}] &\leq \mathbb{E}[f_t] - \alpha\mathbb{E}[\nabla f_t^T m_t] + \frac{L\alpha^2}{2}\mathbb{E}[\|m_t\|^2] \\ &\leq \mathbb{E}[f_t] - \frac{\alpha}{2}\left(\mathbb{E}[2\nabla f_t^T m_t - \|m_t\|^2]\right) \\ &= \mathbb{E}[f_t] - \frac{\alpha}{2}\left(\mathbb{E}[\|\nabla f_t\|^2] - \mathbb{E}[\|m_t - \nabla f_t\|^2]\right).\end{aligned} \tag{24}$$

Using strong convexity ($\|\nabla f_t\|^2 \geq 2\mu(f_t - f_*)$) and subtracting f_* from both sides results in

$$\mathbb{E}[f_{t+1} - f_*] \leq (1 - \alpha\mu)\mathbb{E}[f_t - f_*] + \frac{\alpha}{2}\mathbb{E}[\|m_t - \nabla f_t\|^2] \tag{25}$$

So in each step, we get a multiplicative decrease in the expected function value (left term) but we add a term that depends on the variance of our filtered gradient estimate m_t. So, in essence, to establish convergence, we have to show that $\mathbb{E}[\|m_t - \nabla f_t\|^2]$ decreases to zero sufficiently fast.

Since all assumptions of the Kalman filter are satisfied, we know that $\mathbb{E}[m_t] = \mathbb{E}[\nabla f_t]$ and $\mathbb{E}[(m_t - \nabla f_t)(m_t - \nabla f_t)^T] = P_t$. Hence, $\mathbb{E}[\|m_t - \nabla f_t\|^2] = \text{tr}(P_t)$. We now show inductively that $P_t = \frac{1}{t+1}\Sigma$. This holds for $t = 0$ by construction. Assume it holds for arbitrary but fixed $t - 1$. Then

$$K_t = P_{t-1}(P_{t-1} + \Sigma)^{-1} = \frac{1}{t}\Sigma\left(\frac{1}{t}\Sigma + \Sigma\right)^{-1} = \frac{1}{t}\Sigma\left(\frac{t+1}{t}\Sigma\right)^{-1} = \frac{1}{t+1}I \tag{26}$$

and, thus,

$$P_t = (I - K_t)P_{t-1} = \left(I - \frac{1}{t+1}I\right)\frac{1}{t}\Sigma = \frac{1}{t+1}\Sigma \tag{27}$$

Plugging $\mathbb{E}[\|m_t - \nabla f_t\|^2] = \text{tr}(P_t) = \frac{1}{t+1}\text{tr}(\Sigma)$ back into Eq. (25) and introducing the shorthands $e_t = \mathbb{E}[f_t - f_*]$ and $\sigma^2 := \text{tr}(\Sigma)$ reads

$$e_t \leq (1 - \alpha\mu)e_{t-1} + \frac{\alpha\sigma^2}{2}\frac{1}{t}. \tag{28}$$

Iterating backwards results in

$$e_t \leq (1-\alpha\mu)^t e_0 + \frac{\alpha\sigma^2}{2}\sum_{s=0}^{t-1}\frac{(1-\alpha\mu)^{t-1-s}}{s+1} = (1-\alpha\mu)^t e_0 + \frac{\alpha\sigma^2}{2}\sum_{s=1}^{t}\frac{(1-\alpha\mu)^{t-s}}{s}. \tag{29}$$

Lemma 1 shows that the sum term is $O(1/t)$. The first (exponential) term is trivially $O(1/t)$, which concludes the proof. □

Lemma 1. *Let* $0 < c < 1$ *and define the sequence (for* $t \geq 1$*)*

$$a_t = \sum_{s=1}^{t}\frac{c^{t-s}}{s}.$$

Then $a_t \in O(\frac{1}{t})$.

Proof. Let T be the smallest index such that $c\frac{T+1}{T} < 1$, i.e., $T = \lceil c/(1-c)\rceil$. Define

$$M = \max\left(Ta_T, \left(1 - c\frac{T+1}{T}\right)^{-1}\right) \tag{30}$$

This ensures that $a_T \leq \frac{M}{T}$ and

$$\frac{1}{M} + c\frac{t+1}{t} \leq 1 \tag{31}$$

for all $t \geq T$. We now show inductively that $a_t \leq \frac{M}{t}$ for all $t \geq T$. It holds for $t = T$ by construction of M. Assume it holds for some $t \geq T$. Then

$$\begin{aligned} a_{t+1} &= \sum_{s=1}^{t+1}\frac{c^{t+1-s}}{s} = \frac{1}{t+1} + c\underbrace{\sum_{s=1}^{t}\frac{c^{t-s}}{s}}_{=a_t \leq M/t} \\ &\leq \frac{1}{t+1} + c\frac{M}{t} = \frac{M}{t+1}\underbrace{\left(\frac{1}{M} + c\frac{t+1}{t}\right)}_{\leq 1 \text{ by Eq. (31)}} \leq \frac{M}{t+1}. \end{aligned} \tag{32}$$

□