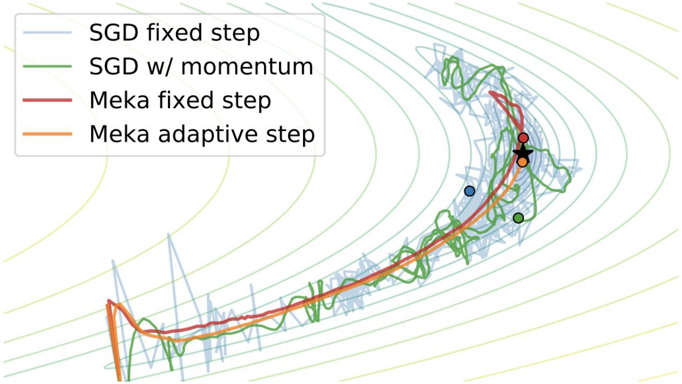
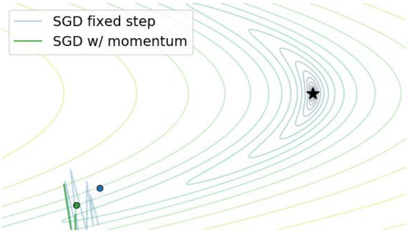
Self-Tuning Stochastic Optimization with Curvature-Aware **Gradient Filtering**

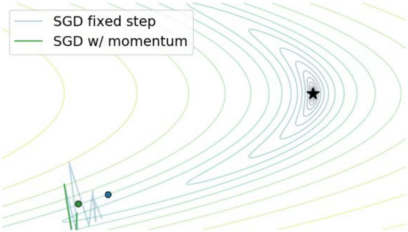
Ricky T. Q. Chen, Dami Choi, Lukas Balles
David Duvenaud, Philipp Hennig

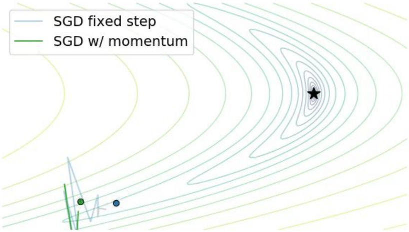
University of Toronto and Max Planck Institute



Gradient noise → Diffusion

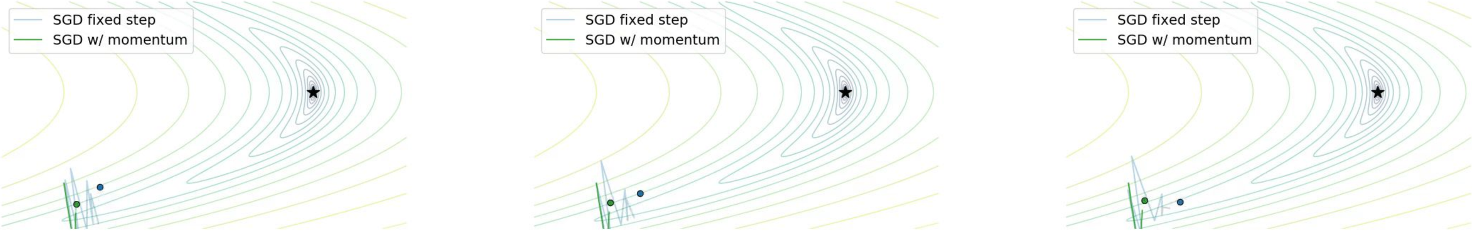






Larger gradient noise

Gradient noise → Diffusion

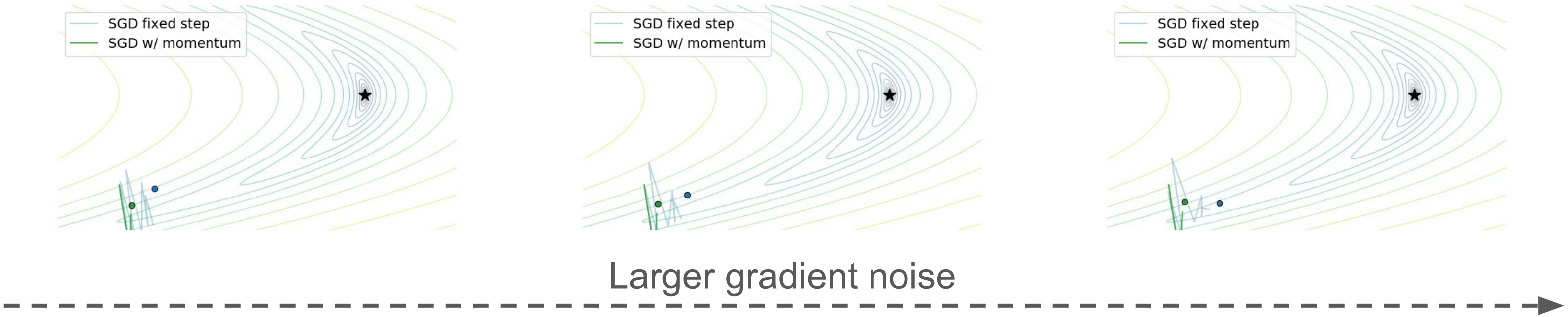


Larger gradient noise

Roger Grosse
@RogerGrosse

90% of all confusion about neural net training dynamics would vanish if everyone got used to thinking about and measuring neural net Jacobians, Hessians, Fisher information matrices, etc.

Gradient noise → Diffusion



Can we create a *self-controlled* / *self-tuning* optimizer?

- Autodiff for estimating *curvature* and *variance*.
- *Bayesian Inference* within a **gradient dynamics model**.
- Automatic step sizes based on *exploration vs exploitation*.

Gradient Estimation as Posterior Inference

What this work is about:

View gradient observations as a dynamical system.

Infer **full** gradient from history of stochastic observations.

$$\min_{\theta} \mathbb{E}_x \left[f(\theta, x) \right]$$

$$\theta_{t+1}$$

$$\nabla_\theta f(\theta_t, x) \qquad \nabla_\theta f(\theta_{t+1}, x)$$

$$\theta_{t+2}$$

$$\theta_t$$

$$\nabla_\theta \mathbb{E}_x \left[f(\theta_{t+1}, x) \right]$$

Track gradient?

Prediction?

$$\nabla_\theta \mathbb{E}_x \left[f(\theta_{t+2}, x) \right]$$

$$\nabla_\theta \mathbb{E}_x \left[f(\theta_t, x) \right]$$

Gradient Estimation as Posterior Inference

What this work is about:

View gradient observations as a dynamical system.

Infer **full** gradient from history of stochastic observations.

What this work is not about:

Inferring limiting distribution of SGD.

Bayesian neural networks.

$$\nabla_\theta \mathbb{E}_x \left[f(\theta_{t+1}, x) \right]$$

Track gradient?

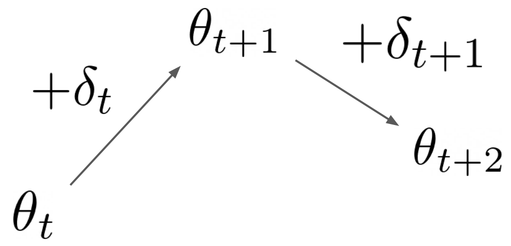
Prediction?

$$\nabla_\theta \mathbb{E}_x \left[f(\theta_t, x) \right]$$

$$\nabla_\theta \mathbb{E}_x \left[f(\theta_{t+2}, x) \right]$$

Constructing a Linear-Gaussian Dynamics Models

Notation:



Define: $f_t \triangleq \nabla_\theta \mathbb{E}_x \left[f(\theta_t, x) \right]$

Based on a Taylor expansion of the gradient:

$$\nabla f_t \approx \nabla f_{t-1} + H_t \delta_{t-1}$$

Constructing a Linear-Gaussian Dynamics Models

Notation:

$$\theta_t \xrightarrow{+\delta_t} \theta_{t+1} \xrightarrow{+\delta_{t+1}} \theta_{t+2}$$

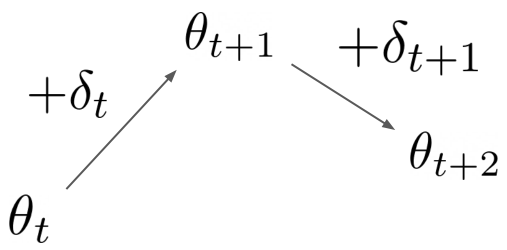
Define: $f_t \triangleq \nabla_\theta \mathbb{E}_x \left[f(\theta_t, x) \right]$

Model uncertainty from update:

$$\nabla f_t | \nabla f_{t-1} \sim \mathcal{N}(\nabla f_{t-1} + \underbrace{B_t \delta_{t-1}}_{\text{Minibatch Hessian-vector product}}, \underbrace{Q_t}_{\text{Variance of minibatch Hessian-vector product}})$$

Constructing a Linear-Gaussian Dynamics Models

Notation:



Define: $f_t \triangleq \nabla_\theta \mathbb{E}_x \left[f(\theta_t, x) \right]$

Model uncertainty from update:

$$\nabla f_t | \nabla f_{t-1} \sim \mathcal{N}(\nabla f_{t-1} + B_t \delta_{t-1}, Q_t)$$

Minibatch
Hessian-vector product

Variance of minibatch
Hessian-vector product

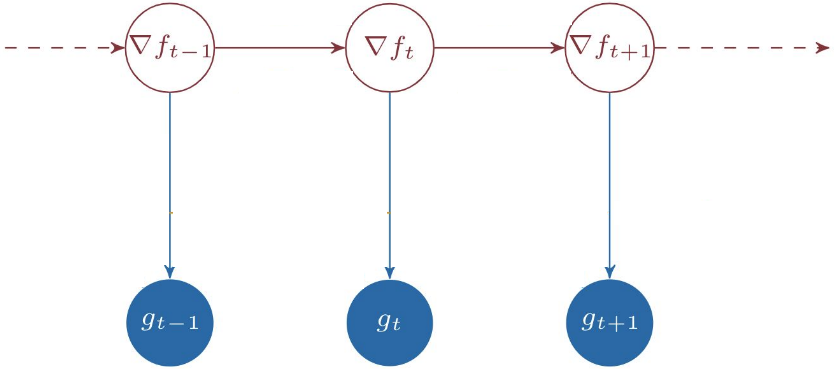
Model uncertainty from stochastic gradients:

$$g_t | \nabla f_t \sim \mathcal{N}(\nabla f_t, \Sigma_t)$$

Minibatch
gradient

Variance of minibatch
gradient

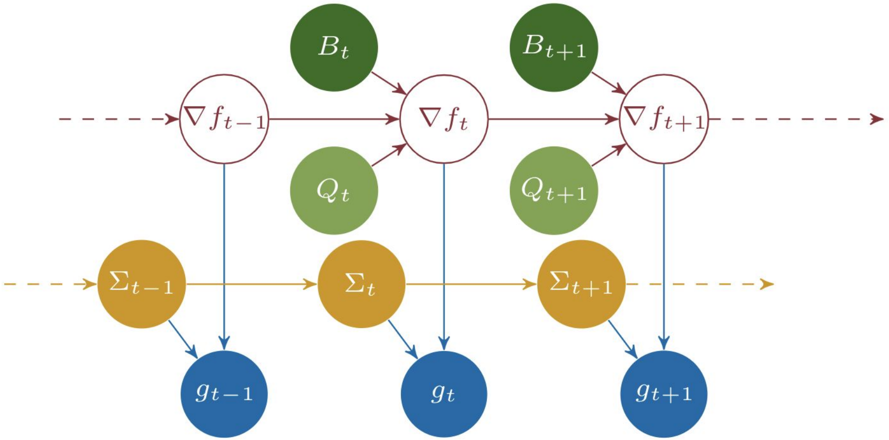
Gradient Filtering for Online Variance Reduction



$$\nabla f_t | \nabla f_{t-1} \sim \mathcal{N}(\nabla f_{t-1} + B_t \delta_{t-1}, Q_t)$$

$$g_t | \nabla f_t \sim \mathcal{N}(\nabla f_t, \Sigma_t)$$

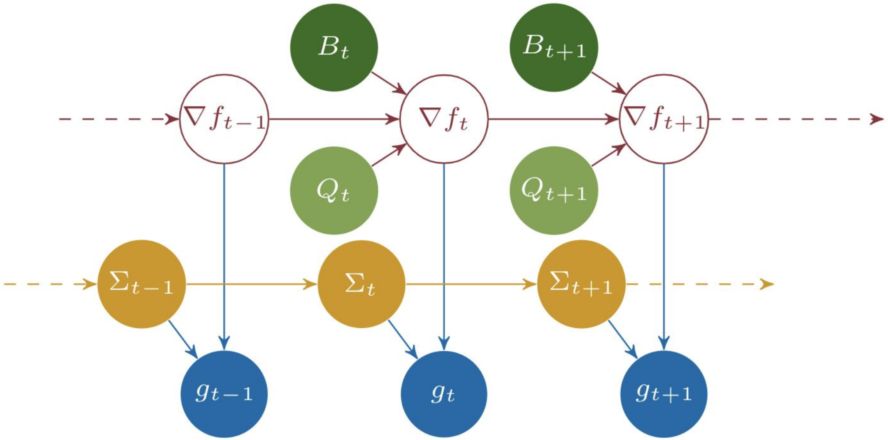
Gradient Filtering for Online Variance Reduction



$$\nabla f_t | \nabla f_{t-1} \sim \mathcal{N}(\nabla f_{t-1} + B_t \delta_{t-1}, Q_t)$$

$$g_t | \nabla f_t \sim \mathcal{N}(\nabla f_t, \Sigma_t)$$

Gradient Filtering for Online Variance Reduction



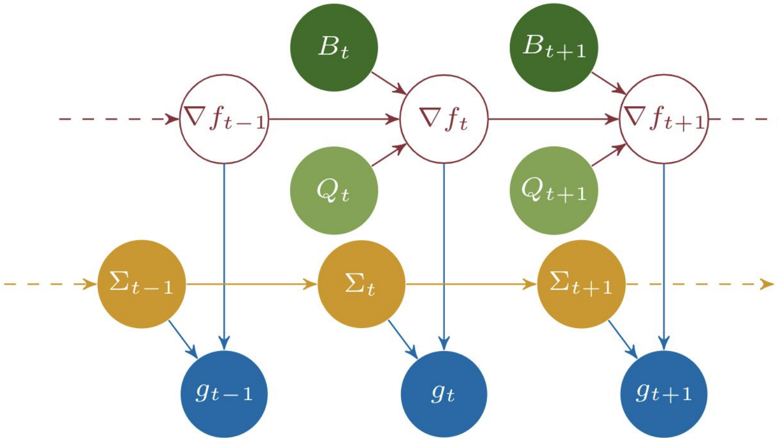
$$\nabla f_t | \nabla f_{t-1} \sim \mathcal{N}(\nabla f_{t-1} + B_t \delta_{t-1}, Q_t)$$

$$g_t | \nabla f_t \sim \mathcal{N}(\nabla f_t, \Sigma_t)$$

We can perform exact inference:

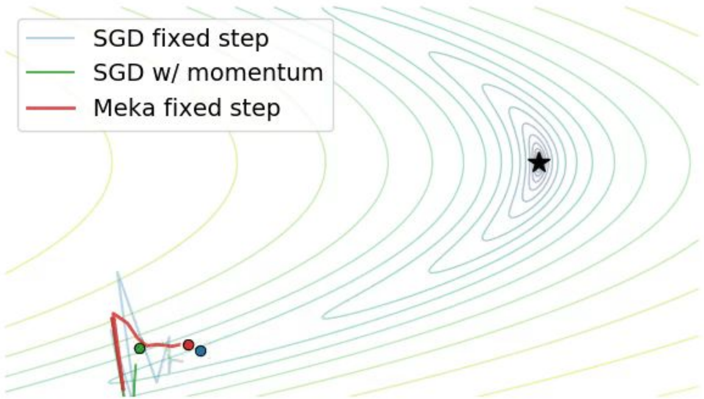
- Filtering $p(\nabla f_t | g_t, \ldots, g_0)$
    - (obtain low-variance gradients)

Gradient Filtering for Online Variance Reduction



$$\nabla f_t | \nabla f_{t-1} \sim \mathcal{N}(\nabla f_{t-1} + B_t \delta_{t-1}, Q_t)$$

$$g_t | \nabla f_t \sim \mathcal{N}(\nabla f_t, \Sigma_t)$$



We can perform exact inference:

- Filtering $p(\nabla f_t | g_t, \ldots, g_0)$
  - (obtain low-variance gradients)

Gradient Filtering for Online Variance Reduction

Computing $p(\nabla f_t | g_t, \ldots, g_0)$ amounts to Kalman Filtering.

$$
\begin{aligned}
m_t^- &= m_{t-1} + B_t \delta_{t-1}, & P_t^- &= P_{t-1} + Q_{t-1} \\
K_t &= P_t^- (P_t^- + \Sigma_t)^{-1} & & \\
m_t &= (I - K_t) m_t^- + K_t g_t, & P_t &= (I - K_t) P_t^- (I - K_t)^T + K_t \Sigma_t K_t^T
\end{aligned}
$$

where m_t and P_t are defined:

$$\nabla f_t \mid g_{1:t}, \delta_{1:t-1} \sim \mathcal{N}(m_t,\ P_t)$$

Gradient Filtering for Online Variance Reduction

Computing $p(\nabla f_t | g_t, \ldots, g_0)$ amounts to Kalman Filtering.

$$m_t^- = m_{t-1} + B_t \delta_{t-1}, \qquad P_t^- = P_{t-1} + Q_{t-1}$$

$$K_t = P_t^- (P_t^- + \Sigma_t)^{-1}$$

$$m_t = (I - K_t) m_t^- + K g_t, \qquad P_t = (I - K_t) P_t^- (I - K_t)^T + K_t \Sigma_t K_t^T$$

Momentum-like update

where m_t and P_t are defined:

$$\nabla f_t \mid g_{1:t}, \delta_{1:t-1} \sim \mathcal{N}(m_t, \ P_t)$$

Gradient Filtering for Online Variance Reduction

Computing $p(\nabla f_t | g_t, \ldots, g_0)$ amounts to Kalman Filtering.

$$m_t^- = m_{t-1} + B_t \delta_{t-1}, \qquad P_t^- = P_{t-1} + Q_{t-1}$$

$$K_t = P_t^- (P_t^- + \Sigma_t)^{-1}$$

$$m_t = (I - K_t) m_t^- + K g_t, \qquad P_t = (I - K_t) P_t^- (I - K_t)^T + K_t \Sigma_t K_t^T$$

Curvature-corrected

Momentum-like update

where m_t and P_t are defined:

$$\nabla f_t \mid g_{1:t}, \delta_{1:t-1} \sim \mathcal{N}(m_t,\ P_t)$$

Gradient Filtering for Online Variance Reduction

Computing $p(\nabla f_t | g_t, \ldots, g_0)$ amounts to Kalman Filtering.

$$m_t^- = m_{t-1} + B_t \delta_{t-1}, \qquad P_t^- = P_{t-1} + Q_{t-1}$$

$$K_t = P_t^- (P_t^- + \Sigma_t)^{-1}$$

$$m_t = (I - K_t) m_t^- + K g_t, \qquad P_t = (I - K_t) P_t^- (I - K_t)^T + K_t \Sigma_t K_t^T$$

Curvature-corrected

Momentum-like update

More weight on new gradient if its variance is relatively smaller

where m_t and P_t are defined:

$$\nabla f_t \mid g_{1:t}, \delta_{1:t-1} \sim \mathcal{N}(m_t,\ P_t)$$

Gradient Filtering for Online Variance Reduction

Computing $p(\nabla f_t | g_t, \ldots, g_0)$ amounts to Kalman Filtering.

$$m_t^- = m_{t-1} + B_t \delta_{t-1}, \qquad P_t^- = P_{t-1} + Q_{t-1}$$

$$K_t = P_t^- (P_t^- + \Sigma_t)^{-1}$$

$$m_t = (I - K_t) m_t^- + K g_t, \qquad P_t = (I - K_t) P_t^- (I - K_t)^T + K_t \Sigma_t K_t^T$$

Curvature-corrected

Momentum-like update

Check out "Implicit Gradient Transport" (Arnold et al.)

More weight on new gradient if its variance is relatively smaller

where m_t and P_t are defined:

$$\nabla f_t \mid g_{1:t}, \delta_{1:t-1} \sim \mathcal{N}(m_t,\ P_t)$$

Estimating Variance with AutoDiff

$$\nabla f_t | \nabla f_{t-1} \sim \mathcal{N}(\nabla f_{t-1} + B_t \delta_{t-1}, Q_t)$$
$$g_t | \nabla f_t \sim \mathcal{N}(\nabla f_t, \Sigma_t)$$

Quantities in blue are computed via *auto-vectorized* automatic differentiation.

Variances are estimated using a minibatch of gradients (or HVPs).

In various autodiff frameworks:

- JAX: jax.vmap
- Tensorflow: tf.vectorized_map
- PyTorch (incoming v1.8.0): torch.vmap

e.g. in JAX:

```
var(vmap(grad(loss_fn(params, batch))))
```

For simplicity, Q_t and Σ_t are set to **scalars**.

Adaptive Step Sizes through Acquisition Functions

$$\delta_t = -\alpha_t m_t$$

Q: Infer step size?

Adaptive Step Sizes through Acquisition Functions

Use estimated quantities like curvature, gradient, variances…

$$\delta_t = -\alpha_t m_t$$

Q: Infer step size?

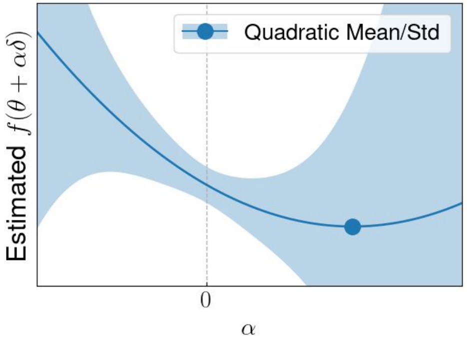
Adaptive Step Sizes through Acquisition Functions

Use estimated quantities like curvature, gradient, variances…

$$\delta_t = -\alpha_t m_t$$

Q: Infer step size?

Step 1: Construct a 1D Gaussian Process (in the descent direction).



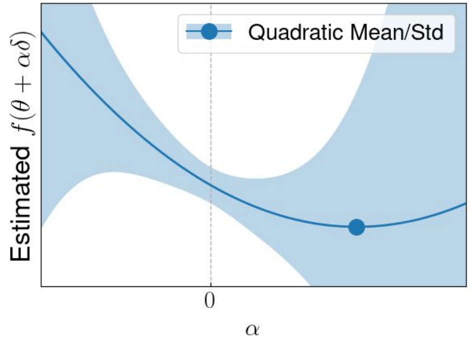
Adaptive Step Sizes through Acquisition Functions

Use estimated quantities like curvature, gradient, variances…

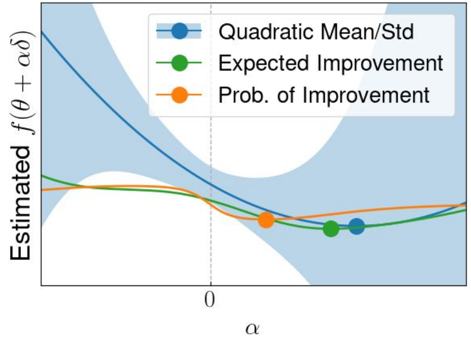
$$\delta_t = -\alpha_t m_t$$

Q: Infer step size?

Step 1: Construct a 1D Gaussian Process (in the descent direction).



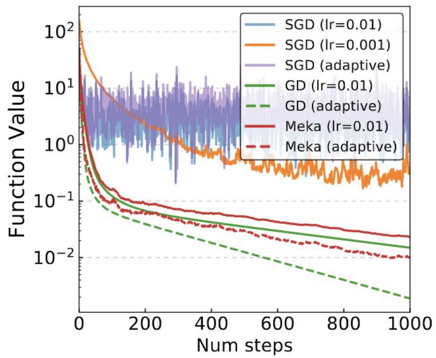
Step 2: Trade-off automatically between exploration and exploitation.



Does it work?

Preliminary testing:

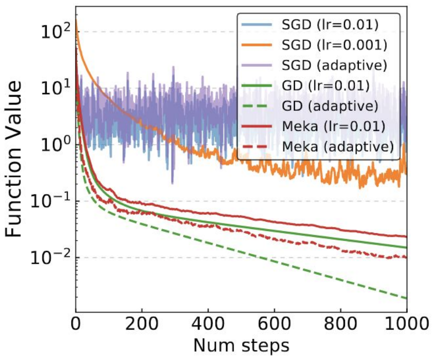
Noisy quadratic (toy):



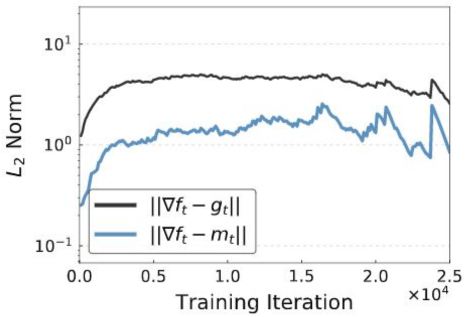
Does it work?

Preliminary testing:

Noisy quadratic (toy):



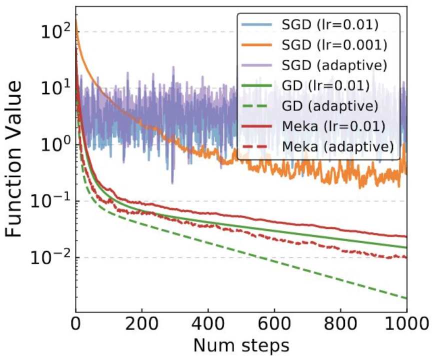
Gradient estimates are good:
(CIFAR10)



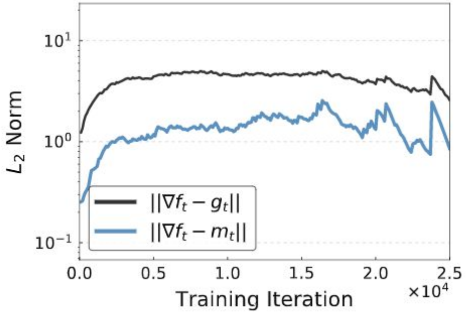
Does it work?

Preliminary testing:

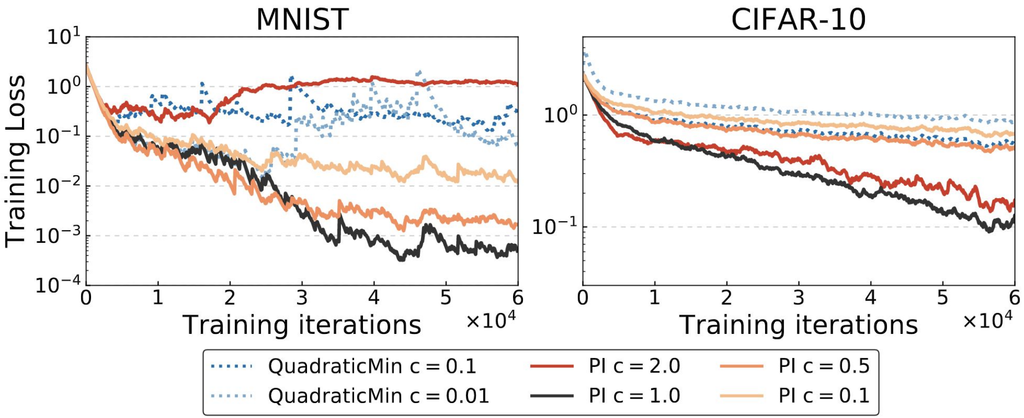
Noisy quadratic (toy):



Gradient estimates are good:
(CIFAR10)

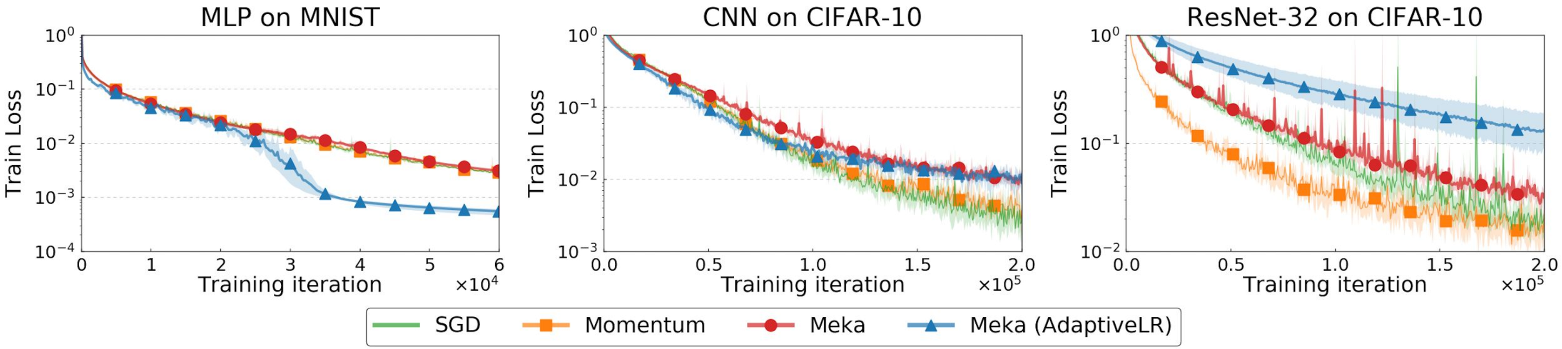


Uncertainty-based step sizes are good:



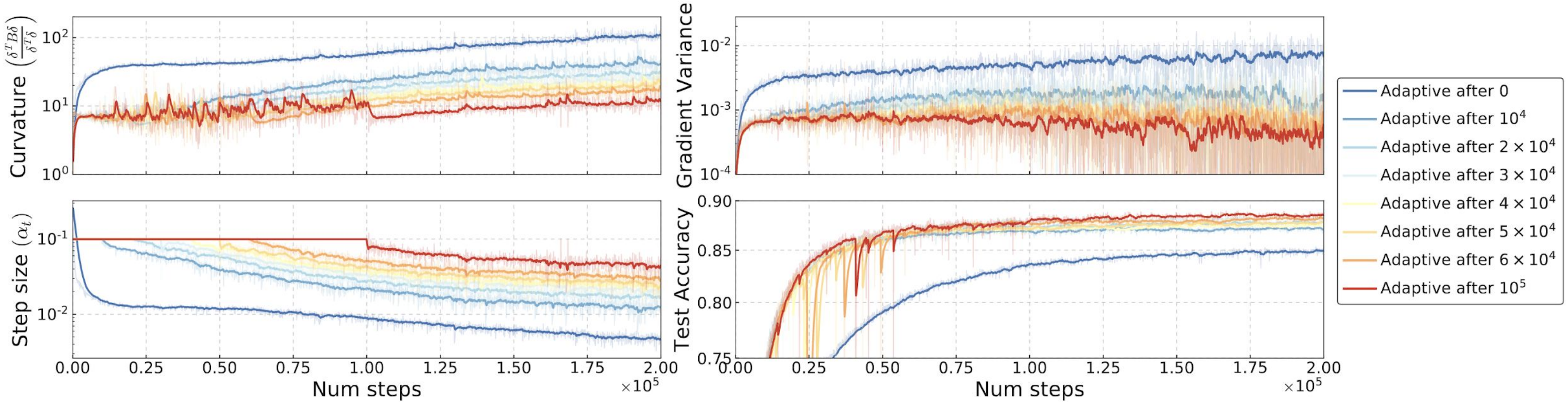
Does it work?

But on neural network training:



Why Not?

We can self-diagnose using quantities estimating during training:



- We dive into high-variance and high-curvature regions (because we can).
- Resulting in small step sizes and bad minima (because they exist).

Summary

- Build training dynamics model, with AD-estimated quantities.
- Perform inference, choose an acquisition function.
- Go!

Problems we saw:

- Model parameters are stochastic.
- Acquisition function has short-horizon bias.

Summary

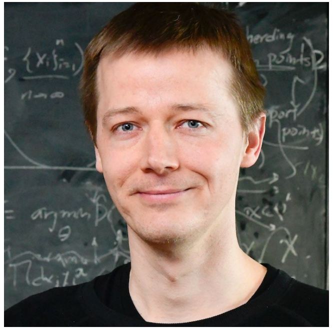
- Build training dynamics model, with AD-estimated quantities.
- Perform inference, choose an acquisition function.
- Go!

Amazing co-authors:

Dami Choi

Lukas Balles

David Duvenaud



Philipp Hennig