Using Statistics to Automate Stochastic Optimization

Hunter Lang (Microsoft Research), Pengchuan Zhang (Microsoft Research AI), Lin Xiao (Microsoft Research AI)

Introduction & Motivation

Setup: Consider solving the stochastic optimization problem

$$\min_x F(x) = \min_x \mathbb{E}_\xi[f(x,\xi)]$$

with the stochastic heavy ball algorithm (SGD with momentum):

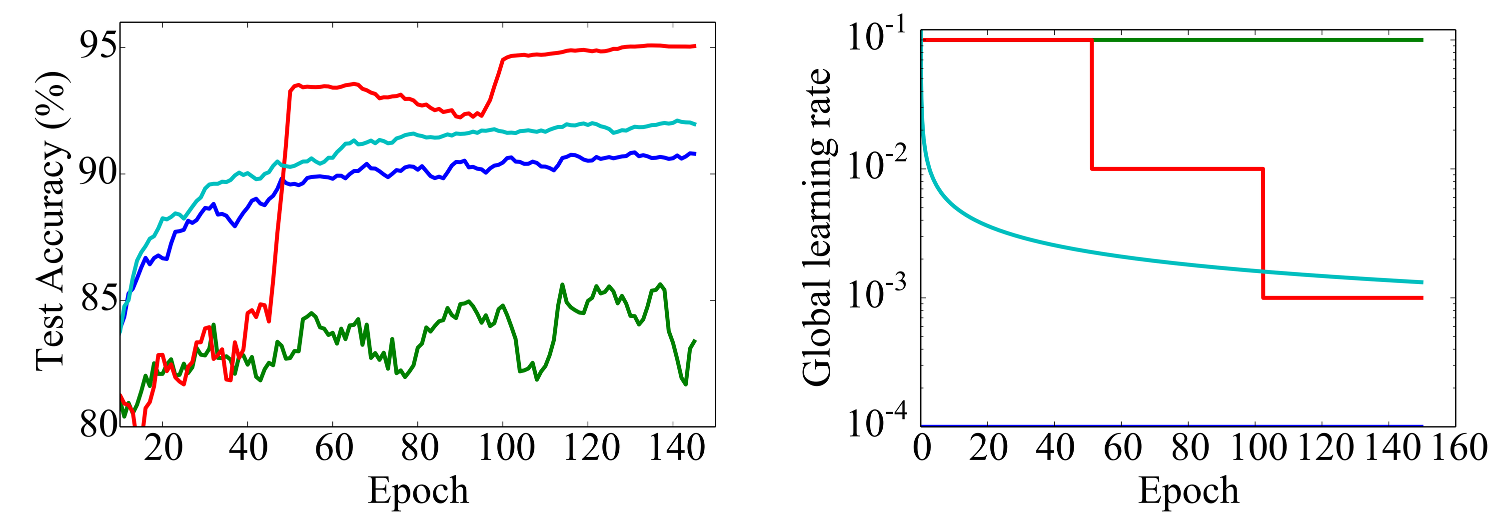
$$d^{k+1} = (1-\beta_k)g^k + \beta_k d^k$$
$$x^{k+1} = x^k - \alpha_k d^{k+1}.$$

Terms: g^k is a stochastic gradient of F at x^k, d^k is a "momentum vector," α_k is the learning rate, and β_k is the momentum parameter. *Deep learning setup*: $f(x,\xi)$ is the loss function for weight matrices x on a random training batch ξ, and $g^k = \nabla_x f(x^k, \xi_k)$.

Question: Need learning rate $\alpha_k \to 0$ for convergence. How to set the *learning rate schedule?*

1. *Polynomial decay*:
$$\alpha_k = \frac{a}{(k+b)^c}$$
2. *Adaptive optimizers*: (Adam, Adagrad, Adadelta, etc.)
$$x^{k+1} = x^k - \alpha G_k d^k,$$
where G_k is an adaptive diagonal matrix.

Problem: both tend to get **worse** generalization performance than hand-tuned *constant-and-cut* schedules:



Test accuracy and α_k schedule for different optimizers; model is a ResNet18 on CIFAR-10.

Constant-and-cut: Run SGD with constant α and β until "progress stops," then set $\alpha \leftarrow \alpha/Z$.

Refined Question: How can we automatically determine the "cut points" of constant-and-cut?

Rough algorithm: In each phase with constant α, β, collect statistics that measure progress of the algorithm. Once these statistics determine that we cannot make more progress, reset them and decrease α. Could use:

- training loss
- loss on a validation set (not always available)
- stationarity statistics (this work).

Algorithm Outline

Algorithm 1: SASA method outline

for $j \in \{0, 1, \ldots\}$ **do**
  for $k \in \{jM, \ldots, (j+1)M-1\}$ **do**
    $d^{k+1} = (1-\beta)g^k + \beta d^k$
    $x^{k+1} = x^k - \alpha d^{k+1}$
    // collect statistics
  end
  if *test(statistics)* **then**
    $\alpha \leftarrow \zeta\alpha$
    // reset statistics
  end
end

Stationarity conditions

What does it mean for "progress to stop?" If the iterates x^k reach a **stationary distribution** π, by definition we cannot make more progress with the current α and β. If we can determine *necessary conditions* for stationarity, we can test for them to determine if we should decrease α.

Yaida's condition: the following formula holds *exactly* at stationarity:

$$\mathbf{E}_\pi[\langle x, g\rangle] = \frac{\alpha}{2}\frac{1+\beta}{1-\beta}\mathbf{E}_\pi[\langle d, d\rangle] \qquad (1)$$

To approximate both sides during training, compute $\langle x^k, g^k\rangle$, $||d^k||^2$. Only two extra inner products per iteration.

Testing for stationarity

Keep track of the statistic:

$$\bar{z}_N = \frac{1}{N}\sum_{k=B}^{B+N}\left(\langle x^k, g^k\rangle - \frac{\alpha}{2}\frac{1+\beta}{1-\beta}\langle d^k, d^k\rangle\right)$$

After discarding B samples during a "burn-in" phase. Additionally keep track of:

$$\bar{v}_N = \frac{1}{N}\sum_{k=B}^{B+N}\frac{\alpha}{2}\frac{1+\beta}{1-\beta}\langle d^k, d^k\rangle$$

To test whether $\bar{z} \approx 0$, we check

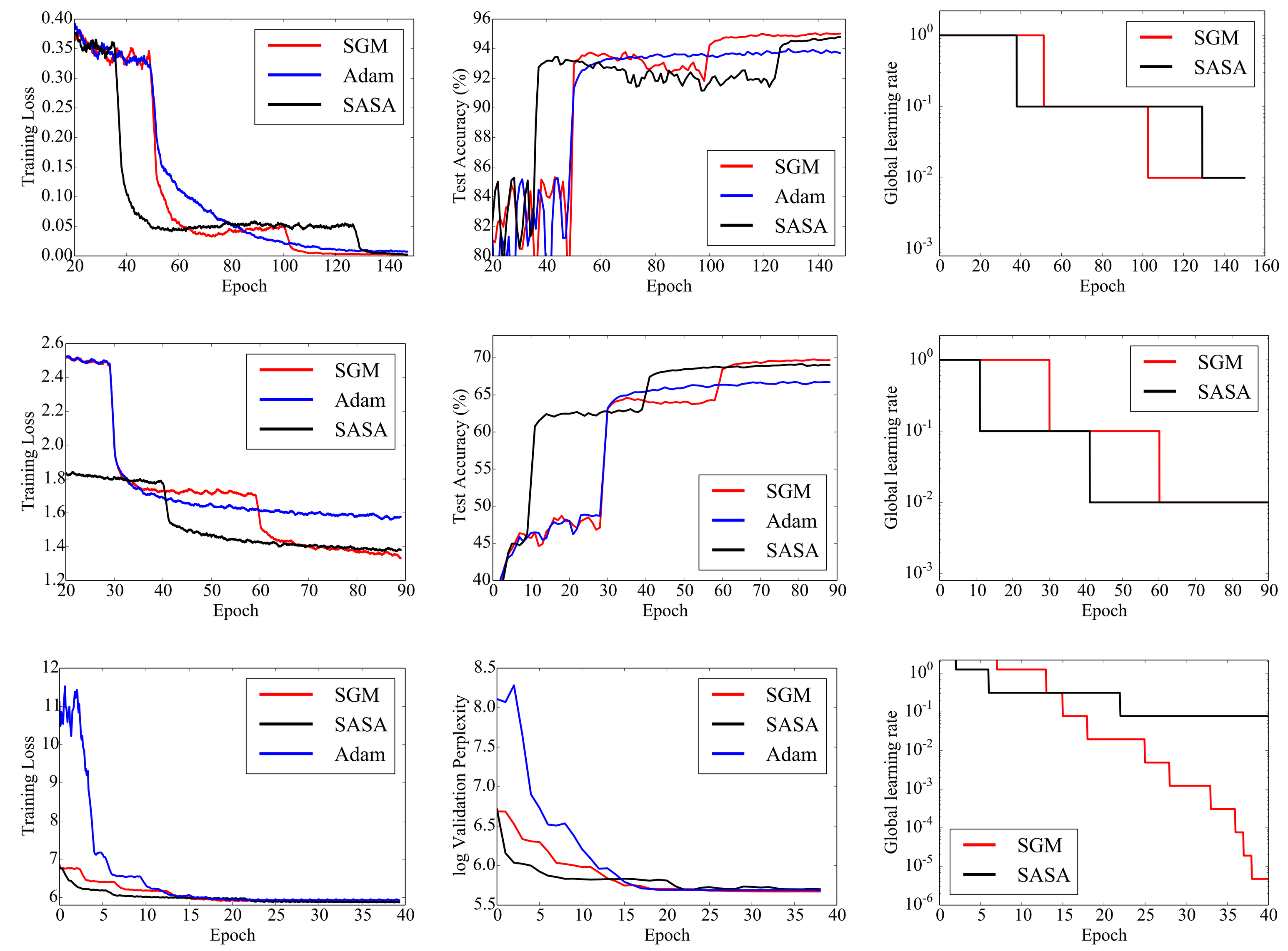
$$\left[\bar{z}_N - \frac{\hat{\sigma}_N}{\sqrt{N}}, \bar{z}_N + \frac{\hat{\sigma}_N}{\sqrt{N}}\right] \in (-\delta\bar{v}_N, \delta\bar{v}_N),$$

where δ is a hyperparameter and the variance estimator $\hat{\sigma}_N$ is computed using the *batch means* formula (2), *not* the sample variance!

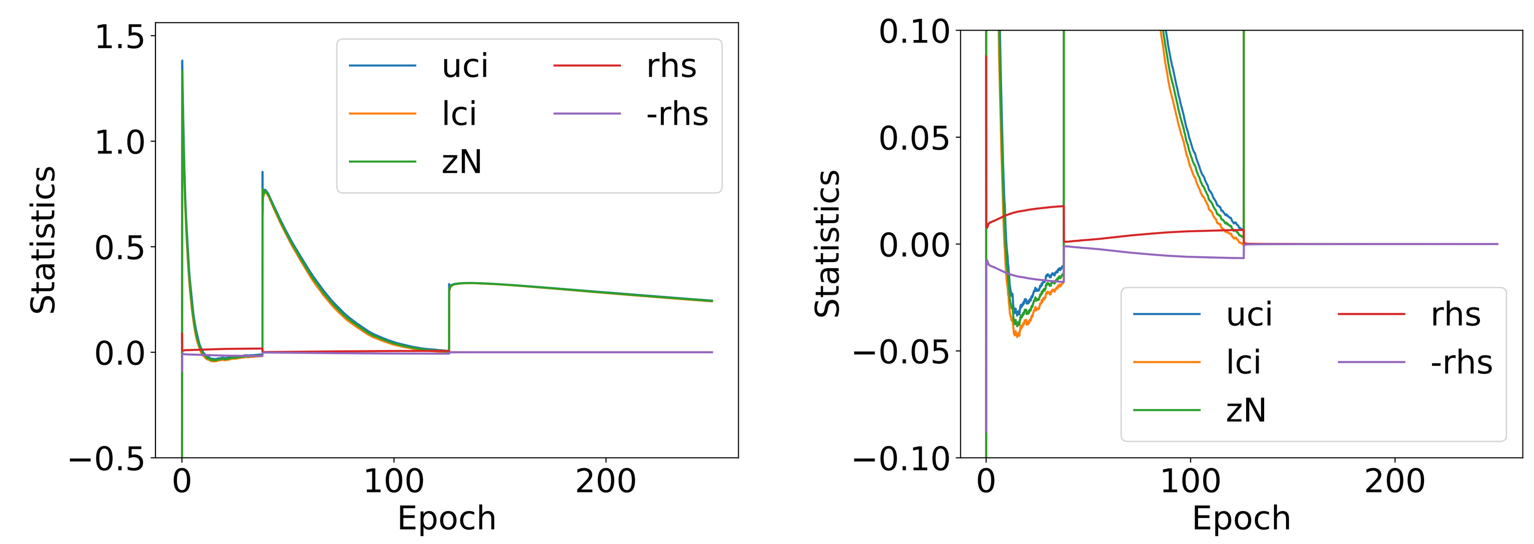
$$\bar{z}^j = \frac{1}{m}\sum_{i=jm}^{(j+1)m-1} z_i \qquad \hat{\sigma}_N^2 = \frac{m}{b-1}\sum_{j=0}^{b-1}(\bar{z}^j - \bar{z}_N)^2. \qquad (2)$$

For b, m large enough, this is a consistent estimator for the variance of $\bar{z}_N$. We take $b = m = \sqrt{N}$.

Experiments



Training loss, test accuracy, and learning rate schedule for SASA, SGM, and Adam on different datasets. Top: ResNet18 on CIFAR-10. Middle: ResNet18 on ImageNet. Bottom: RNN model on WikiText-2.



Evolution of SASA's statistics $\bar{z}_N$ and $\bar{v}_N$ (left) and a zoomed-in version (right) while training ResNet18 on CIFAR-10. Spikes correspond to drops in α.

How to use

Code: github.com/pzzhang/sasa

```
from optim import SASAYaida
optimizer = SASAYaida(model.parameters(), lr=1.0,
                      testfreq=steps_per_epoch)
```