## Introduction & Motivation

**Setup:** Consider solving the stochastic optimization problem

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

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

$$d^{k+1} = (1 - \beta_k) d^k + \beta_k \frac{\partial F(x)}{\partial x}$$

$$x^{k+1} = x^k - \alpha_k d^{k+1}$$

**Terms:** $\frac{\partial F(x)}{\partial x}$ is a stochastic gradient of $F$ at $x$, $d$ is a “momentum vector,” $\alpha_k$ is the learning rate, and $\beta_k$ is the momentum parameter. **Deep learning setup**: $f(x, \xi)$ is the loss function for weight matrices $x$ on a random training batch $\xi$, and $\nabla_x f(x, \xi)$.  

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

1. **Polynomial decay:**
   $$\alpha_k = \frac{a}{(k+b)^p}$$

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 $\alpha_k$ schedule for different optimizers; model is a ResNet18 on CIFAR-10.

**Constant-and-cut:** Run SGD with constant $\alpha$ and $\beta$ until “progress stops,” then set $\alpha \leftarrow \alpha/2$.

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

**Rough algorithm:** In each phase with constant $\alpha$, $\beta$, collect statistics that measure progress of the algorithm. Once these statistics determine that we cannot make more progress, reset them and decrease $\alpha$. Could use:
- training loss
- loss on a validation set (not always available)
- stationarity statistics (this work).

## Algorithm Outline

**Algorithm 1:** SASA method outline

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

## Stationarity conditions

What does it mean for “progress to stop?” If the iterates $x^k$ reach a stationary distribution $\mu$, by definition we cannot make more progress with the current $\alpha$ and $\beta$. If we can determine necessary conditions for stationarity, we can test for them to determine if we should decrease $\alpha$.

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

$$\mathbb{E}_\mu [\langle x, g \rangle] = \frac{\alpha_1}{2} + \frac{\beta}{2} \mathbb{E}_\mu [\langle d, d \rangle]$$

(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:

$$\tilde{z}_N = \frac{1}{N} \sum_{k=0}^{N-1} \langle x^k, g^k \rangle - \frac{\alpha_1}{2} + \frac{\beta}{2} \mathbb{E}_\mu [\langle d, d \rangle]$$

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

$$\tilde{v}_N = \frac{1}{N} \sum_{k=0}^{N-1} \langle d^k, d^k \rangle$$

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

$$\tilde{z}_N - \frac{\delta \sqrt{N}}{\sqrt{\tilde{v}_N}} \tilde{z}_N = \frac{\delta \sqrt{N}}{\sqrt{\tilde{v}_N}} \in (-\tilde{z}_N, \tilde{z}_N),$$

where $\delta$ is a hyperparameter and the variance estimator $\tilde{v}_N$ is computed using the batch means formula (2), not the sample variance!

$$\tilde{v}_B^k = \frac{1}{m} \sum_{j=p}^{(p+1)m-1} z_i \tilde{z}_N^2 = \frac{m}{b-1} \sum_{j=0}^{b-1} (\tilde{z}_N^2).$$

(2)

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

## Experiments


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

**How to use**

Code: [github.com/pzzhang/sasa](https://github.com/pzzhang/sasa)

from optim import SASAYaida
optimizer = SASAYaida(model_parameters(), lr=1.0, testfreq=steps_per_epoch)