Reducing the variance in online optimization by transporting past gradients

Sébastien M. R. Arnold — USC
Pierre-Antoine Manzagol — Google Brain
Reza Babanezhad — UBC
Ioannis Mitliagkas — Mila - U. Montreal
Nicolas Le Roux — Google Brain, Mila - McGill

Contributions

- We introduce a **simple method** to reduce the variance of gradient estimates, which can be **plugged into most existing algorithms**.
- For the quadratic case, we show that our estimator **converges at the rate of $\mathcal{O}(\frac{1}{t})$**.
- For non-quadratics, we use a **forgetting mechanism** to discard stale gradients.
- We empirically **benchmark our method** on many machine learning settings, and demonstrate its competitivity.
- We provide **open-source implementations** in PyTorch and TensorFlow.

Problem Formulation

We wish to solve the following minimization problem:

$$\theta^* = \arg\min_{\theta} \mathop{\mathbb{E}}_{x\sim p}\left[f(\theta, x)\right], \tag{1}$$

where

- x are data samples,
- θ are parameter iterates, and
- we have access to the derivative $g(\theta, x) = \frac{\partial f(\theta,x)}{\partial \theta}$ of $f(\theta, x)$ with respect to θ.

A popular method is to use stochastic gradient descent (SGD) or heavyball (HB):

$$w_t = \overbrace{\mu w_{t-1} \underbrace{- \alpha g(\theta_t, x_t)}_{\text{SGD}}}^{\text{HB}} \tag{2}$$

$$\theta_{t+1} = \theta_t + w_t \tag{3}$$

Problem SGD/HB with constant α are not convergent. Instead, they bounce around a ball of noise.

Goal We would like to build an estimate of $g(\theta) = \mathbb{E}_{x\sim p}[g(\theta, x)]$, which would ensure convergence.

Method

Overall Idea Instead of computing the stochastic gradient at θ_t, compute it at another point $\hat{\theta}_t$ such that the convex combination of the new gradient and the past estimate v_t reduces variance:

$$v_{t+1} = \gamma_t v_t + (1-\gamma_t) g(\hat{\theta}_t, x_t) \approx \mathop{\mathbb{E}}_{x\sim p}\left[g(\theta_t, x)\right]. \tag{4}$$

Solution On quadratics, this is achieved if we let:

$$\gamma_t = \frac{t}{t+1} \quad \text{and} \quad \hat{\theta}_t = \theta_t + \frac{\gamma_t}{1-\gamma_t}(\theta_t - \theta_{t-1}). \tag{5}$$

Solution 2 For non-quadratics we can *forget* stale gradients by using ATA [1]:

$$\gamma_t = \frac{c(t-1)}{1+c(t-1)}\left(1 - \frac{1}{c}\sqrt{\frac{1-c}{t(t-1)}}\right), \qquad c > 1. \tag{6}$$

Algorithm 1 Heavyball-IGT

procedure HEAVYBALL-IGT(Stepsize α, Momentum μ, Initial parameters θ_0)
  $v_0 \leftarrow g(\theta_0, x_0)$, $w_0 \leftarrow -\alpha v_0$, $\theta_1 \leftarrow \theta_0 + w_0$
  for $t = 1, \ldots, T-1$ **do**
    $\gamma_t \leftarrow \frac{t}{t+1}$ ▷ Or use Eq. 6 for ITA.
    $v_t \leftarrow \gamma_t v_{t-1} + (1-\gamma_t) g\left(\theta_t + \frac{\gamma_t}{1-\gamma_t}(\theta_t - \theta_{t-1}), x_t\right)$ ▷ Compute IGT gradient.
    $w_t \leftarrow \mu w_{t-1} - \alpha v_t$ ▷ Plug into heavyball.
    $\theta_{t+1} \leftarrow \theta_t + w_t$ ▷ Update iterates.
  return θ_T

Theory

Assumption

Let f be a quadratic function with positive definite Hessian H with largest eigenvalue L and condition number κ and if the stochastic gradients satisfy: $g(\theta, x) = g(\theta) + \epsilon$ with ϵ a random i.i.d. noise with covariance bounded by BI.

Theorem 1

With stepsize $\alpha = 1/L$, Eq. 5 leads to iterates θ_t satisfying

$$E[\|\theta_t - \theta^*\|^2] \leq \left(1 - \frac{1}{\kappa}\right)^{2t} \|\theta_0 - \theta^*\|^2 + \frac{d\alpha^2 B \bar{\nu}_0^2}{t},$$

with $\nu = (2 + 2\log\kappa)\kappa$ for every $t > 2\kappa$.

Theorem 2

When plugging v_{t+1} in HB (c.f. Heavyball-IGT), there exist constant $\alpha > 0$, $\mu > 0$ such that $\|E[\theta_t - \theta^*]\|^2$ converges to zero linearly, and the variance is $\tilde{O}(1/t)$.

Try it yourself !

PyTorch Implementation available at bit.ly/369wK18

```
opt = optim.SGD(model.parameters(), lr=0.01, momentum=0.9)
opt = IGTransporter(model.parameters(), opt)
# or
opt = ITA(model.parameters(), opt, interval=2.0)
```

TensorFlow Implementation available at bit.ly/2WgPNBY

```
optimizer = exp_igt_optimizer.ExpIgtOptimizer(
                learning_rate=0.01,
                tail_fraction=2.0,
                optimizer='mom'  # or 'gd', 'adam'
)
```

More information available at: seba1511.net/project/igt

References

1. Nicolas Le Roux. 2019. *"Anytime Tail Averaging."*
2. Jian Zhang and Ioannis Mitliagkas. 2017. *"YellowFin and the Art of Momentum Tuning."*
3. Prateek Jain, Sham M. Kakade, Rahul Kidambi, Praneeth Netrapalli, and Aaron Sidford. 2017. *"Accelerating Stochastic Gradient Descent."*

Experiments