ADAM, SARAH

Variance Reduction

Alex Salce

SIE 596 | 04/18/2024

Introduction

In general, we are solving an optimization problem of the form…

$$\min_{x \in \mathbb{R}^d} \left\{ P(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x) \right\}$$

We have covered techniques to reduce variance of the randomized gradient steps in a minibatch or stochastic gradient descent algorithm, but can we improve upon these (SVRG, SAG/SAGA, etc)?

ADAM (orig. 2015)

[7] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2017.

SARAH (2017)

[8] L. M. Nguyen, J. Liu, K. Scheinberg, and M. Takáč, *Sarah: A novel method for machine learning problems using stochastic recursive gradient*, 2017.

Each has a unique approach to address variance reduction. Both utilize recursive gradient information.

ADAM

ADAptive **M**oment estimation

Variant of minibatch GD, Momentum, RMSProp

SARAH

Stoch**A**stic **R**ecursive gr**A**dient algorit**H**m

Variant of SVRG

Why is it important?

We are highly motivated to improve gradient descent optimization methods. Why? Computational efficiency when optimizing model parameters for large datasets.

ADAM is a popular optimizer

➔ Very popular in training of neural networks (nonconvex objectives), NLP tasks, GANs, Reinforcement Learning

SARAH offers improved convergence over SVRG

➔ Can replace GD methods in practice for convex optimization like we have seen in this course

Applications for convex and nonconvex objectives (though we will only focus upon convex)

Assumptions (for reference)

- **[CONVEX]** f_i is convex

$$f_i(y) \geq f_i(x) + \nabla f_i(x)^T (y - x)$$

- **[SCONVEX]** Each f_i is μ-strongly convex, $\exists\ \mu > 0\ s.t.$

$$f_i(y) \geq f_i(x) + \nabla f_i(x)^T (y - x) + \frac{\mu}{2} ||y - x||^2, \forall x, y \in \mathbb{R}$$

⟹ **Note, a function is mu-strongly convex if $\boldsymbol{\lambda_{min}\left(\nabla^2 f(x)\right) \geq \mu, \mu > 0\ \forall x \in \mathbb{R}^d}$, where $\boldsymbol{\lambda_{min}(*)}$ operator returns the smallest eigenvalue of *.So, $\boldsymbol{\mu}$ should be less than or equal to the smallest eigenvalue of the Hessian of the objective function.**

Assumptions (for reference)

- **[LSMOOTH]** Each f_i is convex, L-smooth (Lipschitz continuous gradient), $\exists\, L > 0\ s.t.$

$$|f_i(x) - f_i(y)| \leq L\big||x - y|\big|, \forall x, y \in \mathbb{R}$$

- **[L0L1S]** (L_0, L_1) smoothness

Assumption 1 ((L_0, L_1) Smoothness)**.** *All of* $\{f_i\}_{i=0}^{n-1}$ *satisfy* (L_0, L_1) *smoothness, i.e., there exist positive constants* (L_0, L_1)*, such that,* $\forall \boldsymbol{w}_1, \boldsymbol{w}_2 \in \mathbb{R}^d$ *satisfying* $\|\boldsymbol{w}_1 - \boldsymbol{w}_2\| \leq \frac{1}{L_1}$,

$$\|\nabla f_i(\boldsymbol{w}_1) - \nabla f_i(\boldsymbol{w}_2)\| \leq (L_0 + L_1\|\nabla f_i(\boldsymbol{w}_1)\|)\|\boldsymbol{w}_1 - \boldsymbol{w}_2\|, \tag{3}$$

and $f_i(\boldsymbol{w})$ *is lower bounded,* $\forall i \in [0, n-1]$.

Eq. (3) generalizes the classical bounded smoothness condition (i.e., $L_1 = 0$ in Assumption 3), and allows for a wide range of simple and important functions such as the polynomials and even the exponential functions. Moreover, empirical observation [36, 34] suggests that Eq. (3) is a preciser characterization of the loss landscape of neural networks than the classical bounded smoothness condition in tasks where Adam outperforms SGD.

[9] B. Wang, Y. Zhang, H. Zhang, Q. Meng, Z.-M. Ma, T.-Y. Liu, and W. Chen, *Provable adaptivity in adam*, 2022.

ADAM – Summary

- Combines Momentum and RMSProp techniques for accelerating/reducing variance
- Constant learning rate hyperparameters, but adaptive steps (bias corrected)
- Recursive batch gradient information
- No gradient table
- Converges $\boldsymbol{O}\left(\frac{1}{\sqrt{T}}\right)$ for T iterations

Algorithm 1 ADAM

(0) **Require:** stochastic objective function $f_i(w)$

(1) **Require:** learning rate η, exponential decay rates $\beta_1, \beta_2 \in [0, 1)$, tolerance ϵ

(2) **Initialize:** initial parameter vector w_0, initial 1^{st} moment vector $m_0 \leftarrow 0$, initial 2^{nd} moment vector $v_0 \leftarrow 0$, initial timestep $t \leftarrow 0$

(3) **while** w_t is not converged **do**

$t \leftarrow t + 1$

$g_t \leftarrow \nabla_w f_t(w_{t-1})$ (*batch gradient at iteration t*)

$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (*udpate baised first moment estimate*)

$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (*udpate baised second raw moment estimate*)

$\hat{m}_t \leftarrow m_t/(1 - \beta_1^t)$ (*bias-corrected first moment estimate*)

$\hat{v}_t \leftarrow v_t/(1 - \beta_2^t)$ (*bias-corrected second raw moment estimate*)

$w_t \leftarrow w_{t-1} - \eta \cdot \hat{m}_t/(\sqrt{\hat{v}_t} + \epsilon)$ (*update parameters*)

end while

return w_t

[7] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2017.

ADAM Algorithm

Initialization
- Objective function $f \in \mathbb{R}^{n\times 1}$
- Constant learning rate $\boldsymbol{\eta}$
- Exponential decay rates $\boldsymbol{\beta_1}, \boldsymbol{\beta_2}$
- Epsilon $\boldsymbol{\varepsilon}$
- Vectors $w_0 \in \mathbb{R}^{1\times m}$, $m_0 \in \mathbb{R}^{1\times m}$, $v_0 \in \mathbb{R}^{1\times m}$

Loop:
- Stochastic (**batch**) Gradient step
- **Gradient 1st moment estimate (mean of past grads), moving average param & bias correction**
- **Gradient 2nd moment estimate (ssqares of past grads), moving average param & bias correction**
- Parameter update

Algorithm 1 ADAM

(0) **Require:** stochastic objective function $f_i(w)$

(1) **Require:** learning rate η, exponential decay rates $\beta_1, \beta_2 \in [0,1)$, tolerance ϵ

(2) **Initialize:** initial parameter vector w_0, initial 1^{st} moment vector $m_0 \leftarrow 0$, initial 2^{nd} moment vector $v_0 \leftarrow 0$, initial timestep $t \leftarrow 0$

(3) **while** w_t is not converged **do**

$t \leftarrow t+1$

$g_t \leftarrow \nabla_w f_t(w_{t-1})$ *(batch gradient at iteration t)*

$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1-\beta_1)\cdot g_t$ *(udpate baised first moment estimate)*

$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1-\beta_2)\cdot g_t^2$ *(udpate baised second raw moment estimate)*

$\hat{m}_t \leftarrow m_t/(1-\beta_1^t)$ *(bias-corrected first moment estimate)*

$\hat{v}_t \leftarrow v_t/(1-\beta_2^t)$ *(bias-corrected second raw moment estimate)*

$w_t \leftarrow w_{t-1} - \eta \cdot \hat{m}_t/(\sqrt{\hat{v}_t}+\epsilon)$ *(update parameters)*

end while

return w_t

Typical choice: $\eta = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 1e-8$

[7] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2017.

ADAM Algorithm – Momentum

- m_t – exponential moving average based on previous aggregate batch gradient information
- This step is analogous to **Momentum** (same general idea as Acceleration)
- Estimate is biased toward initialization (zero), so an additional **bias-correction step** is employed

Algorithm 1 ADAM

(0) **Require:** stochastic objective function $f_i(w)$

(1) **Require:** learning rate η, exponential decay rates $\beta_1, \beta_2 \in [0, 1)$, tolerance ϵ

(2) **Initialize:** initial parameter vector w_0, initial 1^{st} moment vector $m_0 \leftarrow 0$, initial 2^{nd} moment vector $v_0 \leftarrow 0$, initial timestep $t \leftarrow 0$

(3) **while** w_t is not converged **do**

$t \leftarrow t + 1$

$g_t \leftarrow \nabla_w f_t(w_{t-1})$ (*batch gradient at iteration* t)

$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (*udpate baised first moment estimate*)

$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (*udpate baised second raw moment estimate*)

$\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (*bias-corrected first moment estimate*)

$\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (*bias-corrected second raw moment estimate*)

$w_t \leftarrow w_{t-1} - \eta \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$ (*update parameters*)

end while

return w_t

[7] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2017.

ADAM Algorithm – RMSProp

- v_t – exponential moving average of sum of squares of past gradients
- This step is **RMSProp**
- Estimate is biased toward initialization (zero), so an additional **bias-correction step** is employed

Algorithm 1 ADAM

(0) **Require:** stochastic objective function $f_i(w)$
(1) **Require:** learning rate η, exponential decay rates $\beta_1, \beta_2 \in [0, 1)$, tolerance ϵ
(2) **Initialize:** initial parameter vector w_0, initial 1^{st} moment vector $m_0 \leftarrow 0$, initial 2^{nd} moment vector $v_0 \leftarrow 0$, initial timestep $t \leftarrow 0$
(3) **while** w_t is not converged **do**

$t \leftarrow t + 1$
$g_t \leftarrow \nabla_w f_t(w_{t-1})$ *(batch gradient at iteration t)*
$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ *(udpate baised first moment estimate)*
$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ *(udpate baised second raw moment estimate)*
$\hat{m}_t \leftarrow m_t/(1 - \beta_1^t)$ *(bias-corrected first moment estimate)*
$\hat{v}_t \leftarrow v_t/(1 - \beta_2^t)$ *(bias-corrected second raw moment estimate)*
$w_t \leftarrow w_{t-1} - \eta \cdot \hat{m}_t/(\sqrt{\hat{v}_t} + \epsilon)$ *(update parameters)*

end while
return w_t

[7] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2017.

ADAM Algorithm – ADAM update

- **Parameter update** like descent, subtracting learning rate η times bias-corrected Momentum $\hat{m}_t$ combined with bias-corrected RMSProp $1/(\sqrt{\hat{v}_t} + \varepsilon)$.
- Variance reduction -> RMSProp

Algorithm 1 ADAM

(0) **Require:** stochastic objective function $f_i(w)$

(1) **Require:** learning rate η, exponential decay rates $\beta_1, \beta_2 \in [0, 1)$, tolerance ϵ

(2) **Initialize:** initial parameter vector w_0, initial 1^{st} moment vector $m_0 \leftarrow 0$, initial 2^{nd} moment vector $v_0 \leftarrow 0$, initial timestep $t \leftarrow 0$

(3) **while** w_t is not converged **do**

$t \leftarrow t + 1$

$g_t \leftarrow \nabla_w f_t(w_{t-1})$ *(batch gradient at iteration t)*

$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ *(udpate baised first moment estimate)*

$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ *(udpate baised second raw moment estimate)*

$\hat{m}_t \leftarrow m_t/(1 - \beta_1^t)$ *(bias-corrected first moment estimate)*

$\hat{v}_t \leftarrow v_t/(1 - \beta_2^t)$ *(bias-corrected second raw moment estimate)*

$w_t \leftarrow w_{t-1} - \eta \cdot \hat{m}_t/(\sqrt{\hat{v}_t} + \epsilon)$ *(update parameters)*

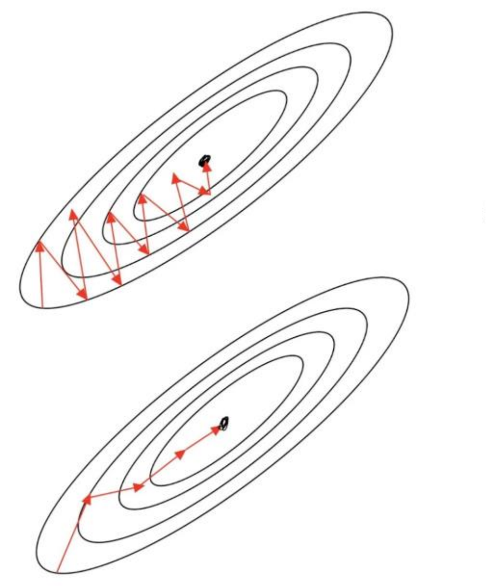
end while

return w_t

$\frac{\hat{m}_t}{\sqrt{\hat{v}_t}}$ - Signal to Noise Ratio (SNR)

[7] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2017.

ADAM



Momentum uses average past gradient information to reduce variance, RMSProp adaptively scales learning rate by magnitude of current and average of past gradients

https://wiki.cloudfactory.com/docs/mp-wiki/solvers-optimizers/rmsprop

ADAM Convergence Summary

For [CONVEX] and [L0L1S] objective

Under basic convexity assumptions for the objective,

ADAM is guaranteed convergence at rate $\boldsymbol{O}(\frac{1}{\sqrt{T}})$

NOTE: There are known flaws in the original proof of Kingma and Ba that are addressed with (L_0, L_1) smoothness condition

$$R(T) = \sum_{t=1}^{T}[f_t(\theta_t) - f_t(\theta^*)]$$

Theorem 4.1. *Assume that the function f_t has bounded gradients, $\|\nabla f_t(\theta)\|_2 \le G$, $\|\nabla f_t(\theta)\|_\infty \le G_\infty$ for all $\theta \in R^d$ and distance between any θ_t generated by Adam is bounded, $\|\theta_n - \theta_m\|_2 \le D$, $\|\theta_m - \theta_n\|_\infty \le D_\infty$ for any $m, n \in \{1, ..., T\}$, and $\beta_1, \beta_2 \in [0, 1)$ satisfy $\frac{\beta_1^2}{\sqrt{\beta_2}} < 1$. Let $\alpha_t = \frac{\alpha}{\sqrt{t}}$ and $\beta_{1,t} = \beta_1 \lambda^{t-1}, \lambda \in (0, 1)$. Adam achieves the following guarantee, for all $T \ge 1$.*

$$R(T) \le \frac{D^2}{2\alpha(1-\beta_1)} \sum_{i=1}^{d} \sqrt{T\widehat{v}_{T,i}} + \frac{\alpha(1+\beta_1)G_\infty}{(1-\beta_1)\sqrt{1-\beta_2}(1-\gamma)^2} \sum_{i=1}^{d} \|g_{1:T,i}\|_2 + \sum_{i=1}^{d} \frac{D_\infty^2 G_\infty \sqrt{1-\beta_2}}{2\alpha(1-\beta_1)(1-\lambda)^2}$$

Corollary 4.2. *Assume that the function f_t has bounded gradients, $\|\nabla f_t(\theta)\|_2 \le G$, $\|\nabla f_t(\theta)\|_\infty \le G_\infty$ for all $\theta \in R^d$ and distance between any θ_t generated by Adam is bounded, $\|\theta_n - \theta_m\|_2 \le D$, $\|\theta_m - \theta_n\|_\infty \le D_\infty$ for any $m, n \in \{1, ..., T\}$. Adam achieves the following guarantee, for all $T \ge 1$.*

$$\frac{R(T)}{T} = O(\frac{1}{\sqrt{T}})$$

⟹ [9] B. Wang, Y. Zhang, H. Zhang, Q. Meng, Z.-M. Ma, T.-Y. Liu, and W. Chen, *Provable adaptivity in adam*, 2022.

[7] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2017.

SARAH – Summary

- Very similar to SVRG (same hyperparameter choices)
- Modifies inner loop, uses recursive gradient info rather than only outer loop gradient
  - Biased inner loop computations, but total expectation is unbiased
- Constant learning rate
- No gradient table
- Similar performance to SVRG, some advantages in strong convex cases

Algorithm 1 SARAH

Parameters: the learning rate $\eta > 0$ and the inner loop size m.
Initialize: $\tilde{w}_0$
Iterate:
for $s = 1, 2, \dots$ **do**
  $w_0 = \tilde{w}_{s-1}$
  $v_0 = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(w_0)$
  $w_1 = w_0 - \eta v_0$
  Iterate:
  for $t = 1, \dots, m-1$ **do**
    Sample i_t uniformly at random from $[n]$
    $v_t = \nabla f_{i_t}(w_t) - \nabla f_{i_t}(w_{t-1}) + v_{t-1}$
    $w_{t+1} = w_t - \eta v_t$
  end for
  Set $\tilde{w}_s = w_t$ with t chosen uniformly at random from $\{0, 1, \dots, m\}$
end for

[8] L. M. Nguyen, J. Liu, K. Scheinberg, and M. Takáč, *Sarah: A novel method for machine learning problems using stochastic recursive gradient*, 2017.

SARAH Algorithm

Italics are same steps as SVRG

Initialization
- *Constant learning rate* $\boldsymbol{\eta}$
- *Objective function*
- *Inner loop steps* $\boldsymbol{m}$
- *Initial parameters* $\tilde{w}_0$

Outer Loop
- *Full gradient descent update ("snapshot point")*

Inner Loop (Variance Reduction)
- **Recursive Stochastic Gradient (<u>one</u> sample) estimate step ("SARAH" update)**
- *Parameter update*

Stochastic Re-initialization
- *Initialize random weight for outer loop* $\tilde{w}_{s-1}$

SARAH

$$\sigma_m \overset{def}{=} \frac{1}{\mu\eta(m+1)} + \frac{\eta L}{2-\eta L} < 1.$$

SVRG

$$\alpha_m = \frac{1}{\mu\eta(1-2L\eta)m} + \frac{2\eta L}{1-2\eta L} < 1.$$

Algorithm 1 SARAH

Parameters: the learning rate $\eta > 0$ and the inner loop size m.
Initialize: $\tilde{w}_0$
Iterate:
for $s = 1, 2, \dots$ **do**
  $w_0 = \tilde{w}_{s-1}$
  $v_0 = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(w_0)$
  $w_1 = w_0 - \eta v_0$
  Iterate:
  for $t = 1, \dots, m-1$ **do**
    Sample i_t uniformly at random from $[n]$
    $v_t = \nabla f_{i_t}(w_t) - \nabla f_{i_t}(w_{t-1}) + v_{t-1}$
    $w_{t+1} = w_t - \eta v_t$
  end for
  Set $\tilde{w}_s = w_t$ with t chosen uniformly at random from $\{0, 1, \dots, m\}$
end for

Outer Loop

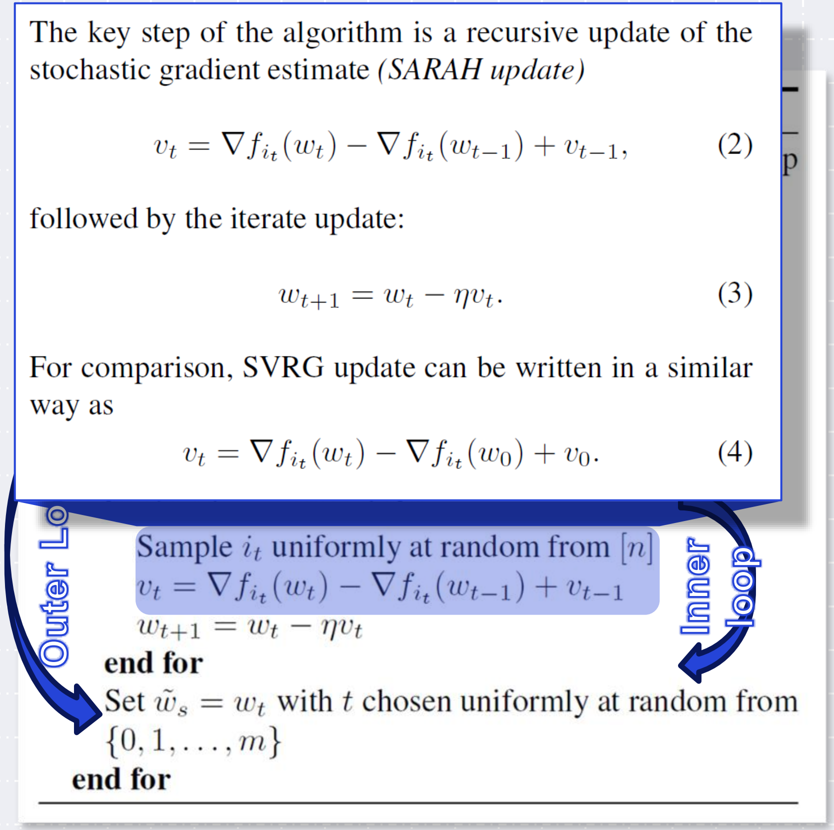
Inner loop

[8] L. M. Nguyen, J. Liu, K. Scheinberg, and M. Takáč, *Sarah: A novel method for machine learning problems using stochastic recursive gradient*, 2017.

SARAH Algorithm

Variance Reduction

The SARAH Algorithm first calculates a full gradient in the outer loop (like SVRG), then uses recursive stochastic gradient information v_t at each iteration of the inner loop, rather than stochastic updates relative to outer loop full gradient calculation.



The key step of the algorithm is a recursive update of the stochastic gradient estimate *(SARAH update)*

$$v_t = \nabla f_{i_t}(w_t) - \nabla f_{i_t}(w_{t-1}) + v_{t-1}, \quad (2)$$

followed by the iterate update:

$$w_{t+1} = w_t - \eta v_t. \quad (3)$$

For comparison, SVRG update can be written in a similar way as

$$v_t = \nabla f_{i_t}(w_t) - \nabla f_{i_t}(w_0) + v_0. \quad (4)$$

[8] L. M. Nguyen, J. Liu, K. Scheinberg, and M. Takáč, *Sarah: A novel method for machine learning problems using stochastic recursive gradient*, 2017.

SARAH Algorithm

Bias

For SVRG, v_t is an unbiased estimator for the gradient, $E_{it}[v_t] = \nabla P(\tilde{w}_t)$ *

Expectation of inner loop iterate is not equal to the full gradient "snapshot", but the total expectation of the full loop is.

$$E_{it}[v_t] \neq \nabla P(\tilde{w}_t) \quad *$$
$$E[v_t] = \nabla P(\tilde{w}_t)$$

The key step of the algorithm is a recursive update of the stochastic gradient estimate *(SARAH update)*

$$v_t = \nabla f_{i_t}(w_t) - \nabla f_{i_t}(w_{t-1}) + v_{t-1}, \qquad (2)$$ *

followed by the iterate update:

$$w_{t+1} = w_t - \eta v_t. \qquad (3)$$

For comparison, SVRG update can be written in a similar way as

$$v_t = \nabla f_{i_t}(w_t) - \nabla f_{i_t}(w_0) + v_0. \qquad (4)$$ *

Outer Lo

Sample i_t uniformly at random from $[n]$
$v_t = \nabla f_{i_t}(w_t) - \nabla f_{i_t}(w_{t-1}) + v_{t-1}$
$w_{t+1} = w_t - \eta v_t$
end for
Set $\tilde{w}_s = w_t$ with t chosen uniformly at random from $\{0, 1, \ldots, m\}$
end for

Inner loop

[8] L. M. Nguyen, J. Liu, K. Scheinberg, and M. Takáč, *Sarah: A novel method for machine learning problems using stochastic recursive gradient*, 2017.

SARAH Convergence Summary

$$\mathbb{E}[\|\nabla P(w_{\mathcal{T}})\|^2] \le \epsilon. \quad (7)$$

For [CONVEX], [LSMOOTH] objective

For functions satisfying [CONVEX] and [LSMOOTH], we can guarantee convergence $\boldsymbol{O}\left(\left(n + \frac{1}{\varepsilon}\right) \log\left(\frac{1}{\varepsilon}\right)\right)$

SARAH

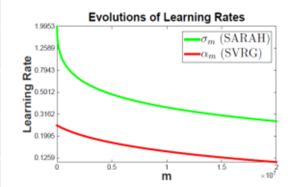
$$\sigma_m \overset{def}{=} \frac{1}{\mu\eta(m+1)} + \frac{\eta L}{2 - \eta L} < 1.$$

SVRG

$$\alpha_m = \frac{1}{\mu\eta(1-2L\eta)m} + \frac{2\eta L}{1-2\eta L} < 1.$$

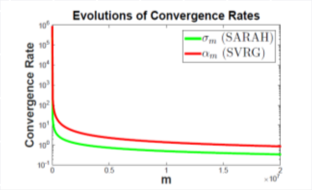
$$\min_{0<\eta<1/L} \sigma_m, \qquad \min_{0<\eta<1/4L} \alpha_m,$$

which can be interpreted as the best convergence rates for different values of m, for both SARAH and SVRG. After



For [CONVEX], [LSMOOTH], [SCONVEX] objective

If our objective has [SCONVEX], SARAH guarantees convergence $\boldsymbol{O}\left(\left(n + \frac{L}{\mu}\right) \log\left(\frac{1}{\varepsilon}\right)\right)$. Same order as SVRG, but uniformly better due to variance bound



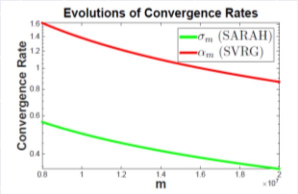


Figure 3: Theoretical comparisons of learning rates (left) and convergence rates (middle and right) with $n = 1,000,000$ for SVRG and SARAH in one inner loop.

[8] L. M. Nguyen, J. Liu, K. Scheinberg, and M. Takáč, *Sarah: A novel method for machine learning problems using stochastic recursive gradient*, 2017.

SARAH Convergence Summary

Table 1: Comparisons between different algorithms for strongly convex functions. $\kappa = L/\mu$ is the condition number.

Method	Complexity	Fixed Learning Rate	Low Storage Cost
GD	$\mathcal{O}(n\kappa \log(1/\epsilon))$	✓	✓
SGD	$\mathcal{O}(1/\epsilon)$	✗	✓
SVRG	$\mathcal{O}((n+\kappa)\log(1/\epsilon))$	✓	✓
SAG/SAGA	$\mathcal{O}((n+\kappa)\log(1/\epsilon))$	✓	✗
SARAH	$\mathcal{O}((n+\kappa)\log(1/\epsilon))$	✓	✓

Table 2: Comparisons between different algorithms for convex functions.

Method	Complexity
GD	$\mathcal{O}(n/\epsilon)$
SGD	$\mathcal{O}(1/\epsilon^2)$
SVRG	$\mathcal{O}(n+(\sqrt{n}/\epsilon))$
SAGA	$\mathcal{O}(n+(n/\epsilon))$
SARAH	$\mathcal{O}((n+(1/\epsilon))\log(1/\epsilon))$
SARAH (one outer loop)	$\mathcal{O}(n+(1/\epsilon^2))$

[5] L. M. Nguyen, J. Liu, K. Scheinberg, and M. Takáč, *Sarah: A novel method for machine learning problems using stochastic recursive gradient*, 2017.

Convergence comparisons

SARAH converges at comparable rates to SVRG/SAG/SAGA for convex functions, but has significant advantages for strong convexity of the objective function.

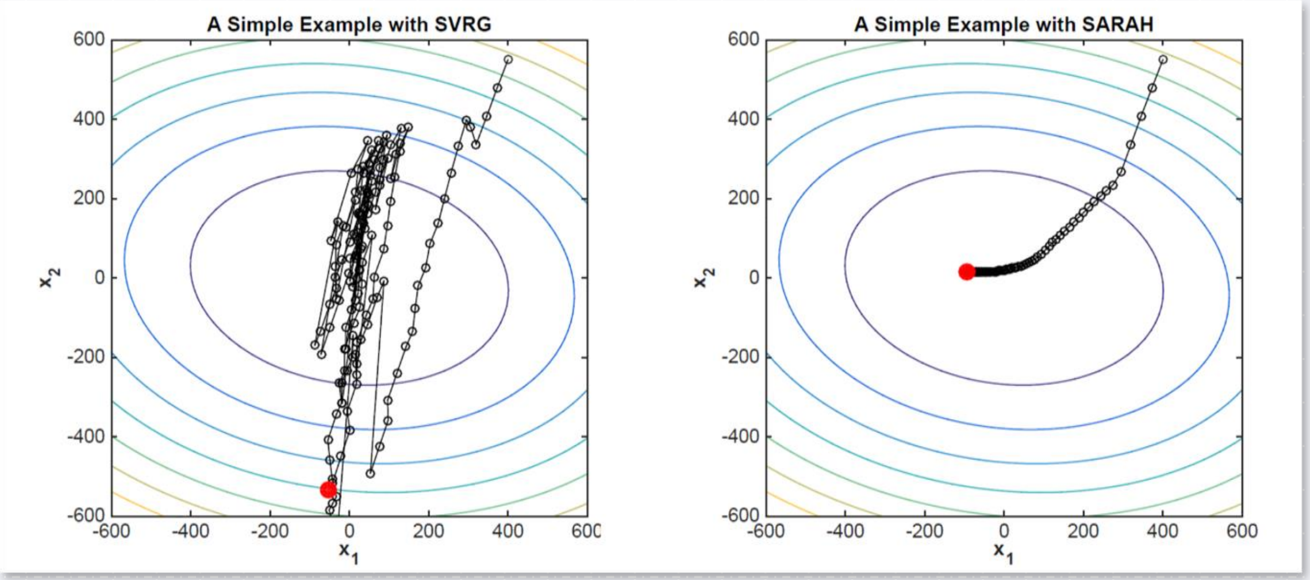
Computational advantages

Recursive update does not require storage of past information, less computationally expensive that similar methods like SAG/SAGA

[8] L. M. Nguyen, J. Liu, K. Scheinberg, and M. Takáč, *Sarah: A novel method for machine learning problems using stochastic recursive gradient*, 2017.

SARAH – Nguyen, Liu, Scheinberg, Takac

Variance of inner loops approaches zero as m increases for SARAH, does not for SVRG



[5] L. M. Nguyen, J. Liu, K. Scheinberg, and M. Takáč, *Sarah: A novel method for machine learning problems using stochastic recursive gradient*, 2017.

Conclusions

ADAM

- Used in practice for fast convergence
- Adaptive unlike SGD methods, locally smooth
    - Minimizes oscillations near optimal solution
- Batch descent can reduce noise

SARAH

- SARAH doesn't seem to be used as widely in practice, but has some nonconvex applications in use/research
- Ongoing research on modifications to algorithm, like random reshuffling, mini-batch, etc.

Comparisons

ADAM – Adaptive learning rate (with bias-correction)

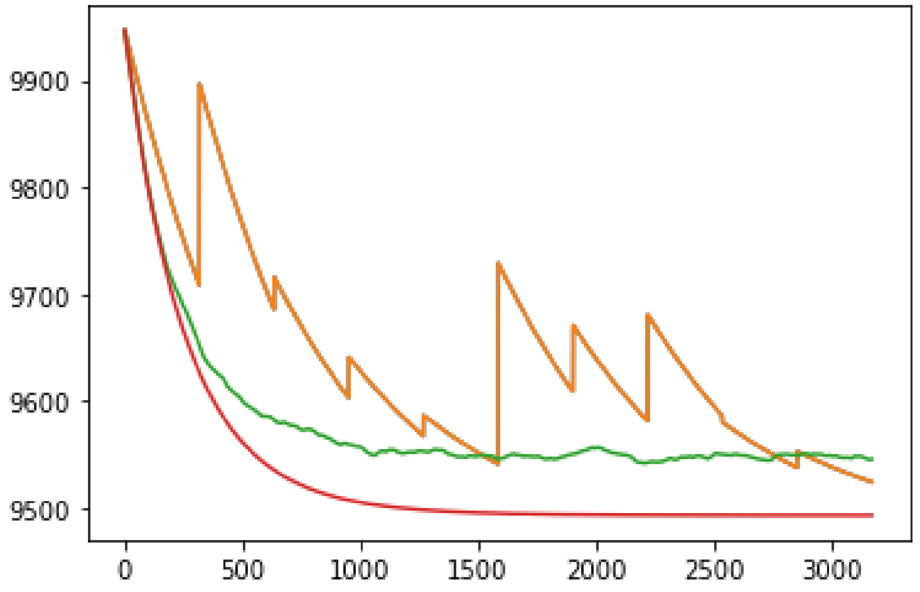
SARAH – SGD constant learning rate (generally a "step in the right direction" from SVRG)

Numerical Experiment – GD, SARAH ADAM

Data

- 500 features
- 10000 samples
- 3170 weight updates per algorithm

SARAH/SVRG – ~3.5s
ADAM – ~4.5s
GD – 95s



Norm of least squares loss vs. Iteraions

Extensions/Further Research

For those interested...

ADAM

- ➔ **AdaMax:** Variant of ADAM that utilizes infinity norm for update in lieu of RMSProp
- ➔ **Ada-class algorithms**: AdaMax, Adadelta, Nadam (Nesterov momentum)

SARAH

- ➔ **SARAH+:** Variant of ADAM that utilizes infinity norm for update in lieu of RMSProp
- ➔ **Random-Reshuffled SARAH:** Does not need full gradient computations [1]

References

[1] A. Beznosikov and M. Takáč, *Random-reshuffled sarah does not need full gradient computations*, Optimization Letters, 18 (2023), p. 727–749.

[2] S. Bock, J. Goppold, and M. Weiss, *An improvement of the convergence proof of the adam-optimizer*, 04 2018.

[3] V. K. Chauhan, A. Sharma, and K. Dahiya, *Saags: Biased stochastic variance reduction methods for large-scale learning*, Applied Intelligence, 49 (2019), p. 3331–3361.

[4] J. Chen, R. Zhang, and Y. Liu, *An adam-enhanced particle swarm optimizer for latent factor analysis*, 2023.

[5] A. Défossez, L. Bottou, F. Bach, and N. Usunier, *A simple convergence proof of adam and adagrad*, 2022.

[6] M. Gürbüzbalaban, A. Ozdaglar, and P. A. Parrilo, *Why random reshuffling beats stochastic gradient descent*, Mathematical Programming, 186 (2019), p. 49–84.

[7] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2017.

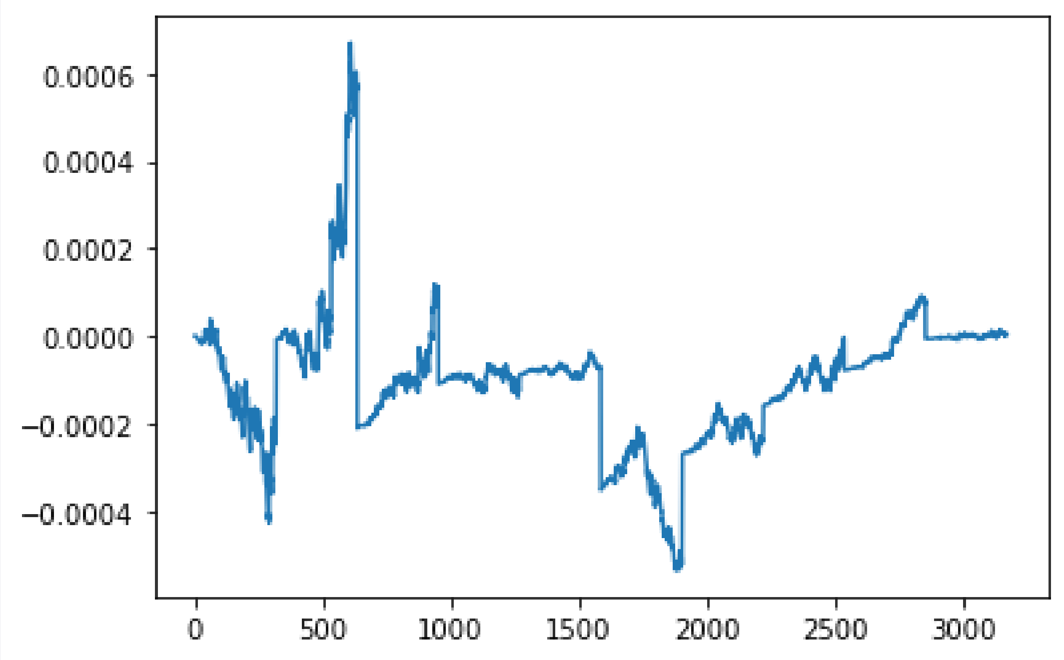
[8] L. M. Nguyen, J. Liu, K. Scheinberg, and M. Takáč, *Sarah: A novel method for machine learning problems using stochastic recursive gradient*, 2017.

[9] B. Wang, Y. Zhang, H. Zhang, Q. Meng, Z.-M. Ma, T.-Y. Liu, and W. Chen, *Provable adaptivity in adam*, 2022.

Additional Resources

- Momentum: https://distill.pub/2017/momentum/?ref=blog.paperspace.com
- https://medium.com/analytics-vidhya/a-complete-guide-to-adam-and-rmsprop-optimizer-75f4502d83be
- https://optimization.cbe.cornell.edu/index.php?title=Adam
- https://medium.com/geekculture/a-2021-guide-to-improving-cnns-optimizers-adam-vs-sgd-495848ac6008

SARAH SVRG delta from numerical experiment slide



SARAH+ Algorithm

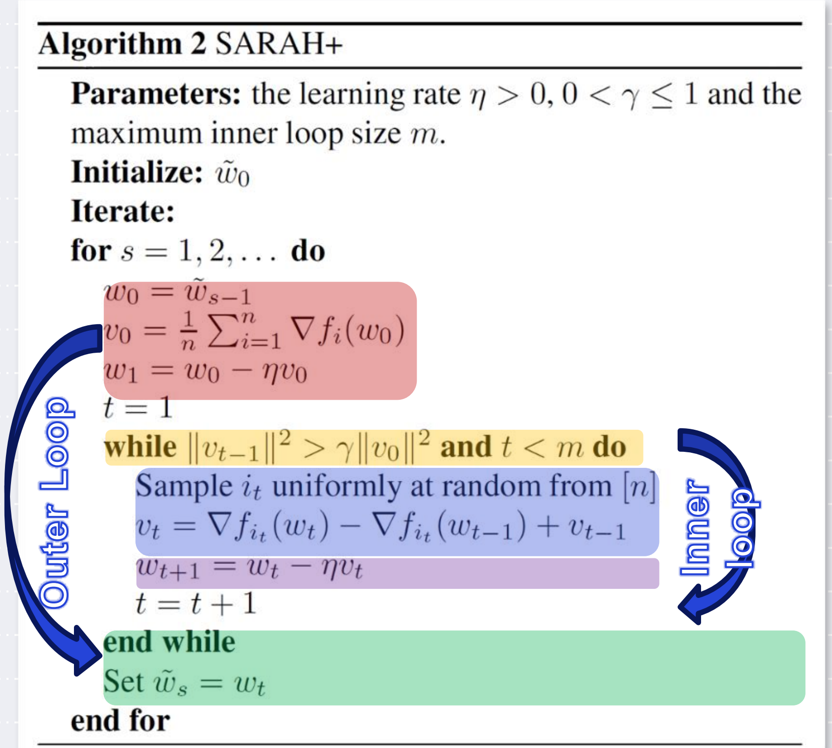
Initialization
- Constant learning rate*********
- Objective function
- Initial parameters $\widetilde{w}_0$

Outer Loop
- Full gradient descent update ("snapshot point")

Inner Loop (Variance Reduction)
- Recursive Stochastic Gradient (single sample) estimate step ("SARAH" update)
- Parameter update

Stochastic Re-initialization
- Initialize random weight for outer loop $\widetilde{w}_{s-1}$

Break inner loop when $||v_t||$ is small enough
- Specify hyperparameter $\boldsymbol{\gamma} \in (\mathbf{0}, \mathbf{1}]$


Algorithm 2 SARAH+

Parameters: the learning rate $\eta > 0$, $0 < \gamma \leq 1$ and the maximum inner loop size m.
Initialize: $\tilde{w}_0$
Iterate:
for $s = 1, 2, \ldots$ **do**
$w_0 = \tilde{w}_{s-1}$
$v_0 = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(w_0)$
$w_1 = w_0 - \eta v_0$
$t = 1$
while $\|v_{t-1}\|^2 > \gamma \|v_0\|^2$ **and** $t < m$ **do**
Sample i_t uniformly at random from $[n]$
$v_t = \nabla f_{i_t}(w_t) - \nabla f_{i_t}(w_{t-1}) + v_{t-1}$
$w_{t+1} = w_t - \eta v_t$
$t = t + 1$
end while
Set $\tilde{w}_s = w_t$
end for

[5] L. M. Nguyen, J. Liu, K. Scheinberg, and M. Takáč, *Sarah: A novel method for machine learning problems using stochastic recursive gradient*, 2017.