SpiderBoost and Proximal SpiderBoost Algorithms

Final Project based on Wang et al.,2018

Aaron Cockley Jess Cooley Hari Dahal Molly Noel

Machine Learning and Optimization
December 2, 2021

Outline

About the Paper

Title:

SpiderBoost and Momentum: Faster Stochastic Variance Reduction Algorithms

Authors:

Zhe Wang, The Ohio State University
Kaiyi Ji, The Ohio State University
Yi Zhou, The University of Utah
Yingbin Laing, The Ohio State University
Vahid Tarokh, Duke University

Published:

Curran Associates, Inc., 2019 presented at NeurIPS 2019

Hyperlink:

https://proceedings.neurips.cc/paper/2019/file/512c5cad6c37edb98ae91c8a76c3a291-Paper.pdf

Problem Statement

Problem (P): $\min_{x \in \mathbb{R}^d} \Psi(x) := f(x)$, where $f(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x)$

1. f_i is the loss for sample i
2. f is the average loss on the training samples (generally nonconvex)

Problem (Q): $\min_{x \in X} \Psi(x) := f(x) + h(x)$, where $f(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x)$

1. f is possibly nonconvex
2. h is a simple convex but possibly nonsmooth regularizer
3. X is a convex set

Bottom Line Up Front

Highlights of SpiderBoost

1. Allows for a larger stepsize than existing methods, $O(1/L)$, while still guaranteeing convergence
2. Can be generalized to solve composite optimization problems (Prox-SpiderBoost)
3. Can use momentum to accelerate Prox-SpiderBoost (Prox-SpiderBoost-M)

Motivation

Motivation for SpiderBoost:
SpiderBoost is able to maintain the level of complexity of the SPIDER algorithm, while also allowing for a larger step size that allows convergence to be achieved faster.

Need for stochastic type methods:
In problem statement (P), the value of sample size n can be very big for a large-scale machine learning problems, this is why the full-batch gradient descent algorithm has a very high computational complexity and stochastic gradient descent algorithms are more efficient and practical.

Defining Gradient Estimator and Proximal Mapping

Equation 1: Gradient Estimator

$$v_0 = \nabla f(x_0)$$

$$v_k = \frac{1}{|S|} \sum_{i \in S} [\nabla f_i(x_k) - \nabla f_i(x_{k-1}) + v_{k-1}]$$

Equation 2: Proximal Mapping

$$\text{for } \eta > 0, \qquad \text{prox}_{\eta h}(x) := \arg\min_{u \in X} (h(u) + \frac{1}{2\eta} \|u - x\|^2)$$

Equation 3: Generalized Gradient

$$G_\eta(x) := \frac{1}{\eta}(x - prox_{\eta h}(x - \eta \nabla f(x)))$$

SpiderBoost and Prox-SpiderBoost

Algorithm 1 SpiderBoost

Input: $\eta = \frac{1}{2L}, q, K, |S| \in \mathbb{N}$.
for $k = 0, 1, \ldots, K-1$ **do**
  if $mod(k, q) = 0$ **then**
    Compute $v_k = \nabla f(x_k)$,
  else
    Draw $|S|$ samples with replacement.
    Compute v_k according to eq. (1).
  end
  $x_{k+1} = x_k - \eta v_k$.
end
Output: x_ξ, where $\xi \overset{\text{Unif}}{\sim} \{0, \ldots, K-1\}$.

Algorithm 2 Prox-SpiderBoost

Input: $\eta = \frac{1}{2L}, q, K, |S| \in \mathbb{N}$.
for $k = 0, 1, \ldots, K-1$ **do**
  if $mod(k, q) = 0$ **then**
    Compute $v_k = \nabla f(x_k)$,
  else
    Draw $|S|$ samples with replacement.
    Compute v_k according to eq. (1).
  end
  $x_{k+1} = \text{prox}_{\eta h}(x_k - \eta v_k)$.
end
Output: x_ξ, where $\xi \overset{\text{Unif}}{\sim} \{0, \ldots, K-1\}$.

Figure: SpiderBoost and Prox-SpiderBoost Algorithms (source: Wang et al.,2018)

where q is the number of points in the mini-batch, K is the number of mini-batches

Experiment Description

- Tested accuracy and sparsity for the SpiderBoost and Prox-SpiderBoost on the Fashion MNIST Dataset for the following hyperparameters:
  - stepsize η
  - batch size B
  - regularization parameter λ
- LeNet5 Architecture
- $\|\cdot\|_1$ regularizer used for Prox-SpiderBoost

Empirical Evaluation-SpiderBoost Accuracy

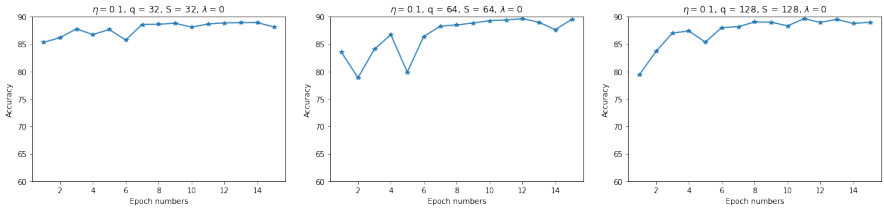


Figure: Testing change in Accuracy with varying Batch sizes (S)

- Accuracy is similar for the three batch sizes, but smaller batch sizes take longer to run:
- 1233 seconds for S=32, 1009 seconds for S=64, 629 seconds for S=128

Empirical Evaluation-SpiderBoost Accuracy

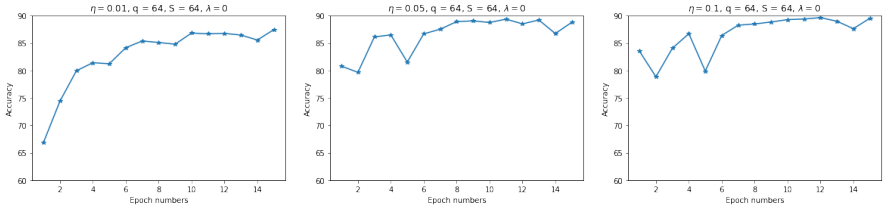


Figure: Testing change in Accuracy with varying step sizes (η)

- By tuning the algorithm using different step sizes, we found that the algorithm was giving optimal results around $\eta = 0.1$
- As expected, smaller step size($\eta = 0.01$) resulted in a slower convergence.
- As we increased our step size towards the optimal step size, the accuracy of the algorithm increased faster

Empirical Evaluation-Prox-SpiderBoost Accuracy

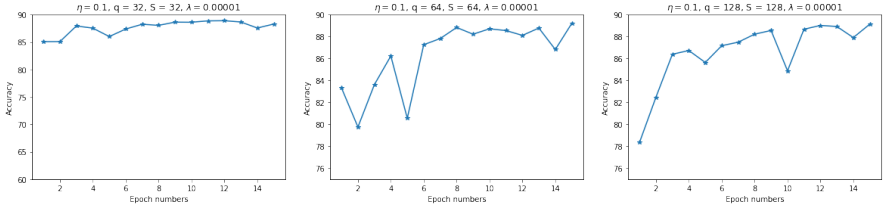


Figure: Testing change in Accuracy with varying Batch sizes (S)

- Using smaller batch sizes gives us good accuracy but takes more time to run: *517 seconds for $S = 64$* vs. *733 seconds for $S = 32$*

Empirical Evaluation-Prox-SpiderBoost Accuracy

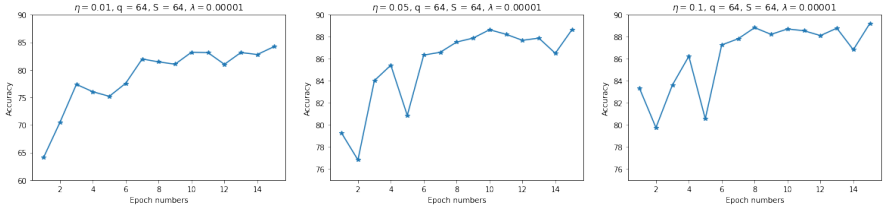


Figure: Testing change in Accuracy with varying step sizes (η)

- As we increased our step size up to an optimal step size our model achieved higher accuracy faster in terms of number of epochs
- The accuracy plot is not monotone as expected of a stochastic model for a non-convex problem

Empirical Evaluation-SpiderBoost Sparsity

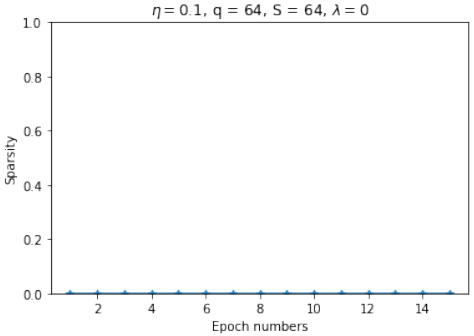


Figure: Sparsity

- Does not use a regularizer, so does not result in a sparse solution

Empirical Evaluation-Prox-SpiderBoost Sparsity

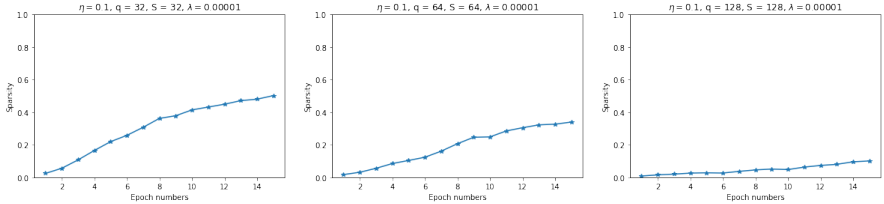


Figure: Testing change in Sparsity with varying Batch sizes (S)

- Prox-SpiderBoost decreases sparsity as the batch sizes increase

Empirical Evaluation-Prox-SpiderBoost Sparsity

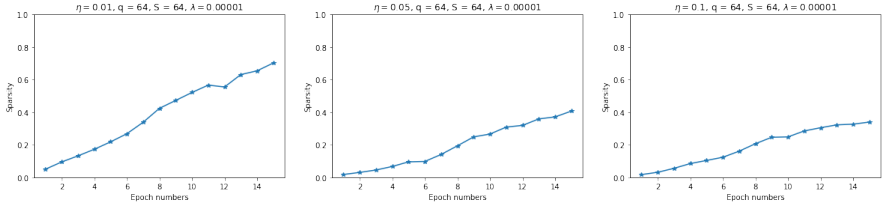


Figure: Testing change in Sparsity with varying step sizes (η)

- Prox-SpiderBoost decreases sparsity as the step sizes increase

Convergence Analysis of SpiderBoost

Assumption: (source: Wang et al.,2018)

The objective function in the problem (P) satisfies:

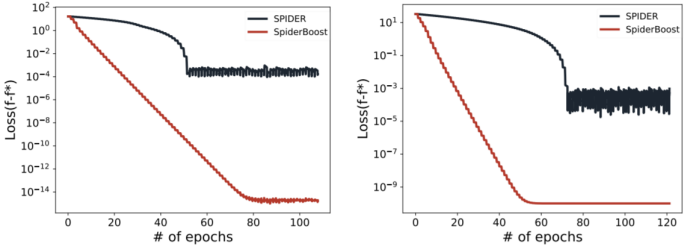
1. The object function ψ is bounded below, i.e., $\psi^* := \inf_{x \in \mathbb{R}^d} \psi(x) > -\infty$,
2. Each gradient $\nabla f_i, i = 1, \ldots, n$ is L-Lipschitz continuous, i.e., $\forall x, y \in \mathbb{R}^d, \|\nabla f_i(x) - \nabla f_i(y)\| \leq L\|x - y\|$.

Theorem: (source: Wang et al.,2018)

Let the assumption hold and apply Algorithm 1 and Algorithm 2 to solve problem (P) and problem (Q), respectively, with parameters $q = |S| = \sqrt{n}$ and stepsize $\eta = \frac{1}{2L}$. Then, the corresponding output x_ξ satisfies $\mathbb{E}\|\nabla f(x_\xi)\| \leq \epsilon$ for (P) and $\mathbb{E}\|G_\eta(x_\xi)\| \leq \epsilon$ for (Q) provided the number K of iterations satisfies

$$K \geq \mathcal{O}\left(\frac{L(\Psi(x_0) - \Psi^*)}{\epsilon^2}\right)$$

Comparison to SPIDER



(c) Dataset: a9a (d) Dataset: w8a

Figure: Comparison to SPIDER with l_2 regularizer (source: Wang et al.,2018)

SPIDER requires a stepsize that depends on the accuracy, $\eta = O(\epsilon/L)$
SpiderBoost allows for a larger stepsize, $\eta = O(1/L)$, which leads to faster convergence

Conclusion

- The SpiderBoost and Prox-SpiderBoost algorithms achieve an optimal convergence with a similar complexity to SPIDER faster than SPIDER due to allowing a larger step size

- According to our experiments:
  - SpiderBoost Algorithm gave optimal results around $\eta = 0.1$
  - Smaller step sizes resulted in a slower convergence
  - Smaller batch sizes have better accuracy but at the cost of having slightly longer run times
  - SpiderBoost does not result in a sparse solution and hence takes a longer time to get a similar accuracy as Prox-SpiderBoost
  - Prox-SpiderBoost increases sparsity as batch sizes decrease
  - Prox-SpiderBoost increases sparsity as step sizes decrease

Thank you!

Backup Slides

Complexity Analysis

Table 1: Comparison of SFO complexity and PO complexity for composite optimization.

Algorithms		Stepsize η	Finite-Sum		Finite-Sum/Online[1]	
			SFO	PO	SFO	PO
ProxGD	[11]	$\mathcal{O}(L^{-1})$	$\mathcal{O}(n\epsilon^{-2})$	$\mathcal{O}(\epsilon^{-2})$	N/A	N/A
ProxSGD	[11]	$\mathcal{O}(L^{-1})$	N/A	N/A	$\mathcal{O}(\epsilon^{-4})$	$\mathcal{O}(\epsilon^{-2})$
ProxSVRG/SAGA	[30]	$\mathcal{O}(L^{-1})$	$\mathcal{O}(n+n^{2/3}\epsilon^{-2})$	$\mathcal{O}(\epsilon^{-2})$	N/A	N/A
Natasha1.5	[3]	$\mathcal{O}(\epsilon^{2/3}L^{-2/3})$	N/A	N/A	$\mathcal{O}(\epsilon^{-3}+\epsilon^{-10/3})$	$\mathcal{O}(\epsilon^{-10/3})$
ProxSVRG$^+$	[22]	$\mathcal{O}(L^{-1})$	$\mathcal{O}(n+n^{2/3}\epsilon^{-2})$	$\mathcal{O}(\epsilon^{-2})$	$\mathcal{O}(\epsilon^{-10/3})$	$\mathcal{O}(\epsilon^{-2})$
Prox-SpiderBoost	(This Work)	$\mathcal{O}(L^{-1})$	$\mathcal{O}(n+n^{1/2}\epsilon^{-2})$	$\mathcal{O}(\epsilon^{-2})$	$\mathcal{O}(\epsilon^{-2}+\epsilon^{-3})$	$\mathcal{O}(\epsilon^{-2})$

[1] The online setting refers to the case where the objective function takes the form of the expected value of the loss function over the data distribution. Such a method can also be applied to solve the finite-sum problem, and hence the SFO complexity in the last column is applicable to both the finite-sum and online problems. Thus, for algorithms that have SFO bounds available in both of the last two columns, the minimum between the two bounds provides the best bound for the finite-sum problem.

Figure: Complexity Comparison (source: Wang et al.,2018)