Faster Stochastic Alternating Direction Method of Multipliers for Nonconvex Optimization

By Anik Saha, Brian Wu, Kevin Kim, Gary Wang

Information about Paper

- Published in ICML 2019
- Authors:
  - **Feihu Huang:** Postdoctoral Researcher at University of Pittsburgh
  - **Songcan Chen**: Professor, College of Computer Science & Technology, Nanjing University of Aeronautics & Astronautics, China.
  - **Heng Huang**: John A. Jurenko Endowed Professor in Computer Engineering at the University of Pittsburgh.

Main Results of Paper

- Proposes a faster stochastic alternating direction method of multipliers (ADMM) for nonconvex optimization by using a new stochastic path-integrated differential estimator (SPIDER), called as SPIDER-ADMM.
- SPIDER-ADMM is proved to achieve record-breaking IFO complexity of:

$$O(n + n^{\frac{1}{2}} n\epsilon^{-1})$$

- Provides a new theoretical analysis framework for nonconvex stochastic ADMM methods with providing the unsolved optimal incremental first-order oracle (IFO) complexity for complexity SVRG-ADMM and SAGA-ADMM methods

The Method of Lagrange Multipliers

- The method of lagrange multipliers is utilized to find local maxima/minima of constrained optimization problem of the form:

$$\min_x f(x)$$

$$s.t. g(x) = 0$$

- We construct the lagrangian function and find the stationary points to find the optimality point:

$$L(x, \lambda) = f(x) - \lambda g(x)$$

What is ADMM?

- The alternating direction method of multipliers (ADMM) is an algorithm used to solve optimization problems of the form:

$$\text{minimize } f(x) + g(y)$$
$$\text{subject to } Ax + By = c$$

Here f and g both are convex functions.
And the augmented Lagrangian function

$$\mathcal{L}_\rho(x, y, z) = f(x) + g(y) - \langle z^T, Ax + By - c\rangle + \rho/2 \, || \, Ax + By - c \, ||^2$$

- ADMM iterations consist of these 3 steps

$$\begin{aligned} x^{k+1} &= \text{argmin}_x \, \mathcal{L}_\rho\left(x, \, y^k, \, z^k\right) \\ y^{k+1} &= \text{argmin}_y \, \mathcal{L}_\rho\left(x^{k+1}, \, y, \, z^k\right) \\ z^{k+1} &= z^k - \rho\left(Ax^{k+1} + By^{k+1} - c\right) \end{aligned}$$

SPIDER-ADMM

- Solves the following nonconvex nonsmooth problem

$$f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x) + \sum_{j=1}^{m} g_j(y_j)$$

$$\text{subject to } Ax + \sum_{j=1}^{m} B_j y_j = c$$

where f is a nonconvex and smooth function
and g_j is a convex and possibly nonsmooth function for all $j \in [m]$

SPIDER-ADMM

- The SPIDER-ADMM utilizes the the Augmented Lagrangian Method which adds a penalty term to the lagrangian

$$\mathcal{L}_\rho(x, y_{[m]}, z) = f(x) + \sum_{j=1}^{m} g_j(y_j) - \langle z, Ax + \sum_{j=1}^{m} B_j y_j - c \rangle$$
$$+ \frac{\rho}{2} ||Ax + \sum_{j=1}^{m} B_j y_j - c||^2$$

- In addition, the authors linearize this function over x

$$\hat{\mathcal{L}}_\rho(x, y_{[m]}^{k+1}, z_k, v_k) = f(x_k) + v_k^T (x - x_k) + \frac{1}{2\eta} ||x - x_k||_G^2 + \sum_{j=1}^{m} g_j(y_j^{k+1})$$
$$- z_k^T (Ax + \sum_{j=1}^{m} B_j y_j^{k+1} - c) + \frac{\rho}{2} ||Ax + \sum_{j=1}^{m} B_j y_j^{k+1} - c||^2$$

Describe the result or algorithm and motivate it intuitively.

- SPIDER-ADMM constructs an unbiased estimate of the gradient over a minibatch at every iteration; every q iterations, the full gradient is used.
- This estimate of the gradient is utilized for solving the optimization problem outlined in lines 9, 10, 11.

Algorithm 1 SPIDER-ADMM Algorithm

1: **Input:** b, q, K, $\eta > 0$ and $\rho > 0$;
2: **Initialize:** $x_0 \in \mathbb{R}^d$, $y_j^0 \in \mathbb{R}^p$, $j \in [m]$ and $z_0 \in \mathbb{R}^l$;
3: **for** $k = 0, 1, \cdots, K-1$ **do**
4: **if** $\mathrm{mod}(k, q) = 0$ **then**
5: Compute $v_k = \nabla f(x_k)$;
6: **else**
7: Uniformly randomly pick a mini-batch $\mathcal{I}_k$ (with replacement) from $\{1, 2, \cdots, n\}$ with $|\mathcal{I}_k| = b$, and compute

$$v_k = \nabla f_{\mathcal{I}_k}(x_k) - \nabla f_{\mathcal{I}_k}(x_{k-1}) + v_{k-1};$$

8: **end if**
9: $y_j^{k+1} = \arg\min_{y_j} \{\mathcal{L}_\rho(x_k, y_{[j-1]}^{k+1}, y_j, y_{[j+1:m]}^k, z_k) + \frac{1}{2}\|y_j - y_j^k\|_{H_j}^2\}$ for all $j \in [m]$;
10: $x_{k+1} = \arg\min_x \hat{\mathcal{L}}_\rho(x, y_{[m]}^{k+1}, z_k, v_k)$;
11: $z_{k+1} = z_k - \rho(Ax_{k+1} + \sum_{j=1}^m B_j y_j^{k+1} - c)$;
12: **end for**
13: **Output:** $\{x, y_{[m]}, z\}$ chosen uniformly random from $\{x_k, y_{[m]}^k, z_k\}_{k=1}^K$.

Complexity Comparison with non-convex ADMM Methods

Table 1. IFO complexity comparison of the non-convex ADMM methods for finding an ϵ-approximate solution of the problem (1), i.e., $\mathbb{E}\|\nabla\mathcal{L}(x, y_{[m]}, z)\|^2 \leq \epsilon$. n denotes the sample size.

Problem	**Algorithm**	**Reference**	**IFO**
Finite-sum	ADMM	Jiang et al. (2019)	$\mathcal{O}(n\epsilon^{-1})$
	SVRG-ADMM	Huang et al. (2016); Zheng & Kwok (2016b)	$\mathcal{O}(n + n^{\frac{2}{3}}\epsilon^{-1})$
	SAGA-ADMM	Huang et al. (2016)	$\mathcal{O}(n + n^{\frac{2}{3}}\epsilon^{-1})$
	SPIDER-ADMM	Ours	$\mathcal{O}(n + n^{\frac{1}{2}}\epsilon^{-1})$

- Incremental first-order oracle (IFO) complexity measures the number of calls made to the stochastic gradient oracle function

$$\nabla f_i(x)$$

Empirical Evaluations

We were primarily interested analyzing SPIDER-ADMM against other baselines:

- How does SPIDER-ADMM compare to other existing non-convex ADMM methods?
  - Baselines: SVRG-ADMM and SAGA-ADMM
- How does the selection of the penalty parameter ρ affect convergence?
  - Baselines: fixed ρ and varying ρ

Convergence Experiment Description

- Binary Classification task with nonconvex sigmoid loss

$$\min_{\mathbf{x}\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x}) + \lambda||\mathbf{A}\mathbf{x}||_1$$

where $f_i(x) = \frac{1}{1+\exp(b_i a_i^T \mathbf{x})}$

- The graph guided fused lasso is used as the nonsmooth regularizer with A obtained by the sparse inverse covariance estimation method
- Here the auxiliary variable is $y = Ax$
  This is the constraint for our problem

Convergence Experiment Setup

- First we obtain an expression for the gradient of f using the gradient of the sigmoid function
- The update equations for x and y requires minimizing the augmented Lagrangian of the loss function and its linear approximation

$$y_{k+1} = \operatorname{argmin}_y \left\{ \mathcal{L}_p(x_k, y, z_k) + ||y - y_k||_H^2 \right\}$$

$$x_{k+1} = \operatorname{argmin}_x \left\{ \hat{\mathcal{L}}_p(x, y_{k+1}, z_k, v_k) \right\}$$

- As both of these minimization problems are convex, we obtain an expression for x_{k+1} and y_{k+1} by solving these equations

Convergence Experiment Setup

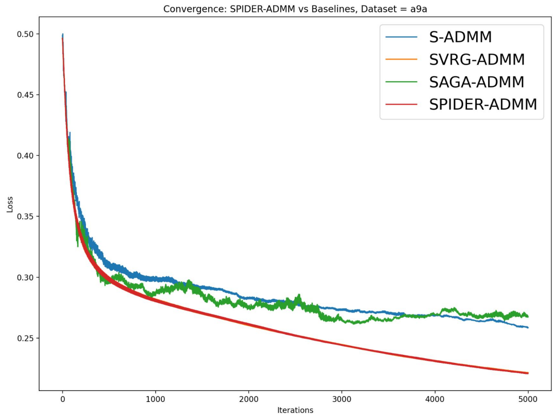
- We use the regularization parameter $\lambda = 10^{-5}$ following the paper
- The batch size b is set to 32 and update interval q is 10
- We tuned the learning rate η by looking at the convergence of a small subset of the training set
- Similarly we set the value of penalty parameter ρ
- Here is a list of the tuned hyperparameters

lambda	b	q	eta	rho
1e-5	32	10	0.05	0.5

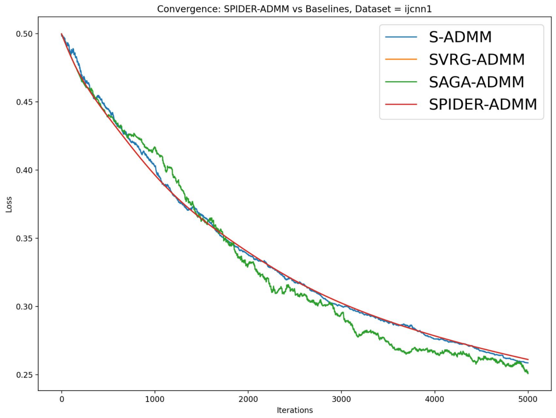
- We implemented S-ADMM, SAGA-ADMM and SVRG-ADMM as our baseline comparisons.

Convergence Comparison Plots

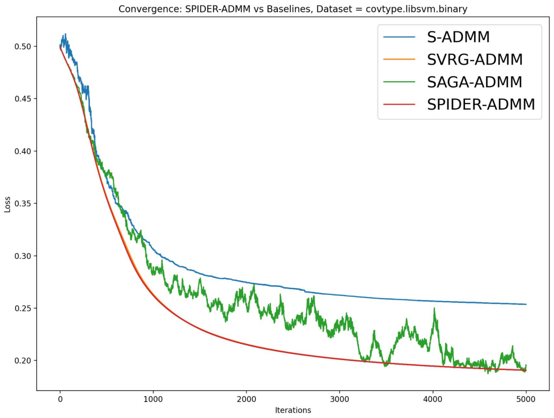
a9a



ijcnn1



covtype.libsvm.binary



Rho Selection Experiments

- We compare choosing a fixed ρ with a varying ρ selection technique:
  - Handpicking fixed ρ=0.5
  - Residual Balancing: Update ρ every iteration in an attempt to improve convergence use the recommended hyperparameters settings: $\mu = 10, \tau\_incr = 2, \tau\_decr = 2$
    - Vary starting ρ
    - This reduces the reliance on picking a good starting ρ, and instead updates ρ to the optimal value

Residual Balancing Explained

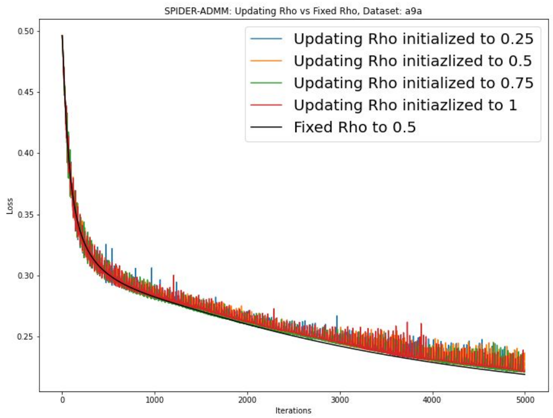
- One standard extension to the classic ADMM algorithm is to vary the penalty parameters at each iteration. Here we applied it to the SPIDER-ADMM algorithm.

$$\rho^{k+1} := \begin{cases} \tau^{incr}\rho^k & \text{if } ||r^k||_2 > \mu||s^k||_2 \\ \rho^k/\tau^{decr} & \text{if } ||s^k||_2 > \mu||r^k||_2 \\ \rho^k & \text{otherwise,} \end{cases}$$

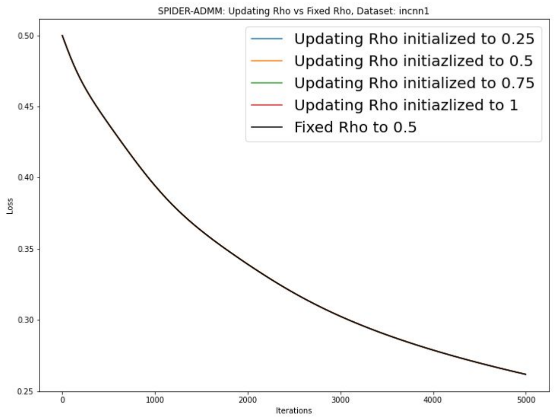
- We then have to choose three hyperparameters: μ, τ^{incr}, τ^{decr}.
  - Although we now have more hyperparameters, these three matter less when compared to the starting value of ϱ.
  - Makes sure that the primal and dual residual are within a factor of μ, by increasing or decreasing by τ^{incr} or τ^{decr} respectively

Performance Plots for This Experiment

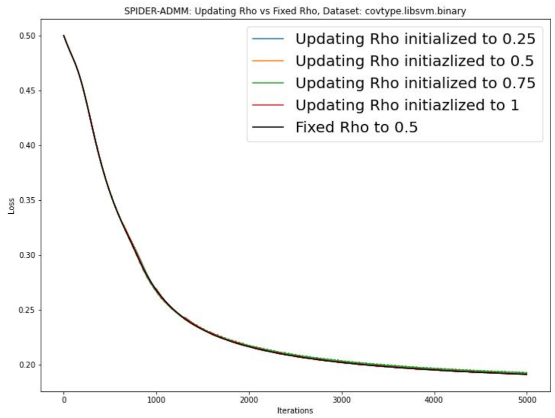
a9a



ijcnn1



covtype.libsvm.binary



Questions?

Resources

- https://arxiv.org/pdf/2008.01296.pdf
- https://arxiv.org/pdf/1610.02758.pdf
- https://stanford.edu/~boyd/papers/pdf/admm_distr_stats.pdf