The Search-direction Correction makes first-order methods faster

Yifei Wang

Peking University

August 23

Joint work with Zeyu Jia and Prof. Zaiwen Wen

Outline

The problem setting

- We are interested in the following unconstrained optimization problems

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}) = \psi(\mathbf{x}) + h(\mathbf{x}), \tag{1}$$

  where ψ is smooth and h is a possibly non-smooth convex function.
- In machine learning, ψ often has the form

$$\psi(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N}\psi_i(\mathbf{x}), \tag{2}$$

  where ψ_i is the prediction error to the i-th sample.
- The dimension of the variable x and the number of samples N are often extremely **HUGE**.

First-order and/or stochastic methods

- The dimension of the variable x and the number of samples N are often extremely **HUGE**.
- The high dimension of the variable x makes first-order methods popular. Many of these methods incorporate a momentum term to accelerate the convergence.
- Stochastic algorithms deal with the large sample number N.
  - The vanilla SGD may suffer from the large variance of stochastic gradients.
  - SVRG , SAG and SAGA tackle this problem and achieve acceleration compared to SGD.

First-order methods with momentum

- Nesterov accelerated method

$$\begin{cases} \mathbf{x}_k = \mathbf{y}_{k-1} - s\nabla f(\mathbf{y}_{k-1}), \\ \mathbf{y}_k = \mathbf{x}_k + \dfrac{k-1}{k+2}(\mathbf{x}_k - \mathbf{x}_{k-1}). \end{cases}$$

- The heavy-ball method

$$\begin{cases} \mathbf{u}_{k+1} = \beta^{(\mathsf{HB})}\mathbf{u}_k - \nabla f(\mathbf{x}_k), \\ \mathbf{x}_{k+1} = \mathbf{x}_k - s\mathbf{u}_{k+1}. \end{cases}$$

- The non-linear conjugate gradient method

$$\begin{cases} \mathbf{u}_{k+1} = \beta_k^{(\mathsf{CG})}\mathbf{u}_k - \nabla f(\mathbf{x}_k), \\ \mathbf{x}_{k+1} = \mathbf{x}_k - s_k\mathbf{u}_{k+1}. \end{cases}$$

FIRE from molecular dynamics

- Recently, an optimization algorithm called fast inertial relaxation engine (FIRE[1]) is proposed for finding the atomic structures with the minimum potential energy.
- Involve an extra term of the velocity correction along the gradient direction with the same magnitude of the current velocity and adopt a carefully designed restarting criterion.

$$\mathbf{u}_{k+1} = (1-\beta_k)\mathbf{u}_k - \beta_k \frac{\|\mathbf{u}_k\|}{\|\nabla f(\mathbf{x}_k)\|}\nabla f(\mathbf{x}_k) - \nabla f(\mathbf{x}_k).$$

- FIRE can outperform the conjugate gradient method. It is even competitive to the limited-memory BFGS in several test cases.
- Nevertheless, the analysis on the convergence rate is absent.

[1] Erik Bitzek et al. "Structural Relaxation Made Simple". In: Physical Review Letters 97.17 (2006).

Our contributions

- Adapt FIRE in molecular dynamics to solve general smooth and nonsmooth optimization problems.
  - Introduce a family of first-order methods with the search direction correction (SDC) and propose the fast inertial search direction correction (FISC) algorithm.
  - SDC is extended to composite optimization and stochastic optimization problems.
- Interpret methods with SDC by second-order ODEs.
  - FISC's ODE has the convergence rate of $O(1/t^2)$ on smooth convex optimization problems.
  - On composite optimization problems, FISC is proven to have the $\mathcal{O}(1/k^2)$ convergence rate.
- Numerical experiments on sparse optimization, logistic regression and deep learning indicate the strength of SDC.

A family of first-order methods with SDC

- We first focus on solving the smooth problem (1) with $h = 0$.
- SDC involves two sequences of parameters $\{\beta_k\}_{k=1}$ and $\{\gamma_k\}_{k=1}$ and introduces $\mathbf{u}$ as a search direction to update $\mathbf{x}$.
- In the beginning of the $(k+1)$-th iteration, consider the restarting criterion

$$\varphi_k = \langle -\nabla f(\mathbf{x}_k), \mathbf{u}_k \rangle \geq 0. \tag{3}$$

- If this criterion holds, we update

$$\mathbf{u}_{k+1} = (1-\beta_k)\mathbf{u}_k - \gamma_k \frac{\|\mathbf{u}_k\|}{\|\nabla f(\mathbf{x}_k)\|} \nabla f(\mathbf{x}_k) - \nabla f(\mathbf{x}_k). \tag{4}$$

and update β_k, γ_k.

A family of first-order methods with SDC

- In **FIRE**, β_k and γ_k are updated by

$$\gamma_{k+1} = \beta_{k+1} = d_\beta \beta_k, \quad 0 < d_\beta < 1.$$

- In **FISC**, β_k and γ_k are parameterized with l_k, i.e.,

$$\beta_k = \frac{r}{l_k - 1 + r}, \quad \gamma_k = \frac{r-3}{l_k - 1 + r}, \tag{5}$$

  where $r \geq 3$. l_{k+1} is updated by $l_{k+1} = l_k + 1$.

- If the criterion (3) is not met, reset $\mathbf{u}_{k+1}, \beta_{k+1}$ and γ_{k+1} as

$$\begin{aligned} \mathbf{u}_{k+1} &= -\nabla f(\mathbf{x}_k), \\ \beta_{k+1} &= \beta_1, \gamma_{k+1} = \gamma_1. \end{aligned} \tag{6}$$

A family of first-order methods with SDC

- Then, calculate the step size s_k. Either of the following choices of s_k is acceptable:
  - (i) Fix the step size $s_k = s_0$.
  - (ii) Perform a backtracking line search to find a step size s_k that satisfies the Armijo conditions.
  - (iii) Perform a nonmonotone line search[1] to find a step size s_k that satisfies nonmonotone Armijo conditions.
- Update

$$\mathbf{x}_{k+1} = \mathbf{x}_k + s_k \mathbf{u}_{k+1}. \tag{7}$$

[1] Hongchao Zhang and William W. Hager. "A nonmonotone line search technique and its application to unconstrained optimization. In: SIAM Journal on Optimization 14.4 (2004), pp. 10431056.

Algorithm 1 A family of first-order methods with SDC

Require: initial guess $\mathbf{x}_0$, initial value $\mathbf{u}_0 = 0$, other required parameters.

set $k = 0$, fix step size s_0 or calculate it by the line search.
while *Convergence criteria are not met* or $k < N_{max}$ **do**
  Calculate φ_k by (3).
  if $\varphi_k \geq 0$ **then**
    Compute $\mathbf{u}_{k+1}$ and update $\beta_{k+1}, \gamma_{k+1}$.

$$\mathbf{u}_{k+1} = (1 - \beta_k)\mathbf{u}_k - \gamma_k \frac{\|\mathbf{u}_k\|}{\|\nabla f(\mathbf{x}_k)\|} \nabla f(\mathbf{x}_k) - \nabla f(\mathbf{x}_k).$$

  else
    Set $\mathbf{u}_{k+1} = -\nabla f(\mathbf{x}_k)$ and reset $\beta_{k+1}, \gamma_{k+1}$.
  end if
  Fix step size s_k or calculate it using line search techniques.
  Update $\mathbf{x}_{k+1} = \mathbf{x}_k + s_k \mathbf{u}_{k+1}$, $k \to k + 1$.
end while

The importance of restarting

- Momentum methods without/with restarting behaves like a heavy ball/a skilled skier.
- Nesterov accelerated method without restarting has an $\mathcal{O}(1/k^2)$ convergence rate.
- On quadratic cases $f(x) = \frac{1}{2}x^T Ax + b^T x$ where A is positive definite, Brendan O'Donoghue and Emmanuel Candés[1] showed that Nesterov accelerated method with restarting asymptoticly exhibits $\mathcal{O}(\exp(1-\sqrt{\kappa})^k)$ convergence rate, even without knowing μ. Here $\kappa = L/\mu$ is the condition number of f.

[1] Brendan O'Donoghue and Emmanuel Candés, "Adaptive restart for accelerated gradient schemes", In: Foundations of computational mathematics (2015).

The importance of restarting

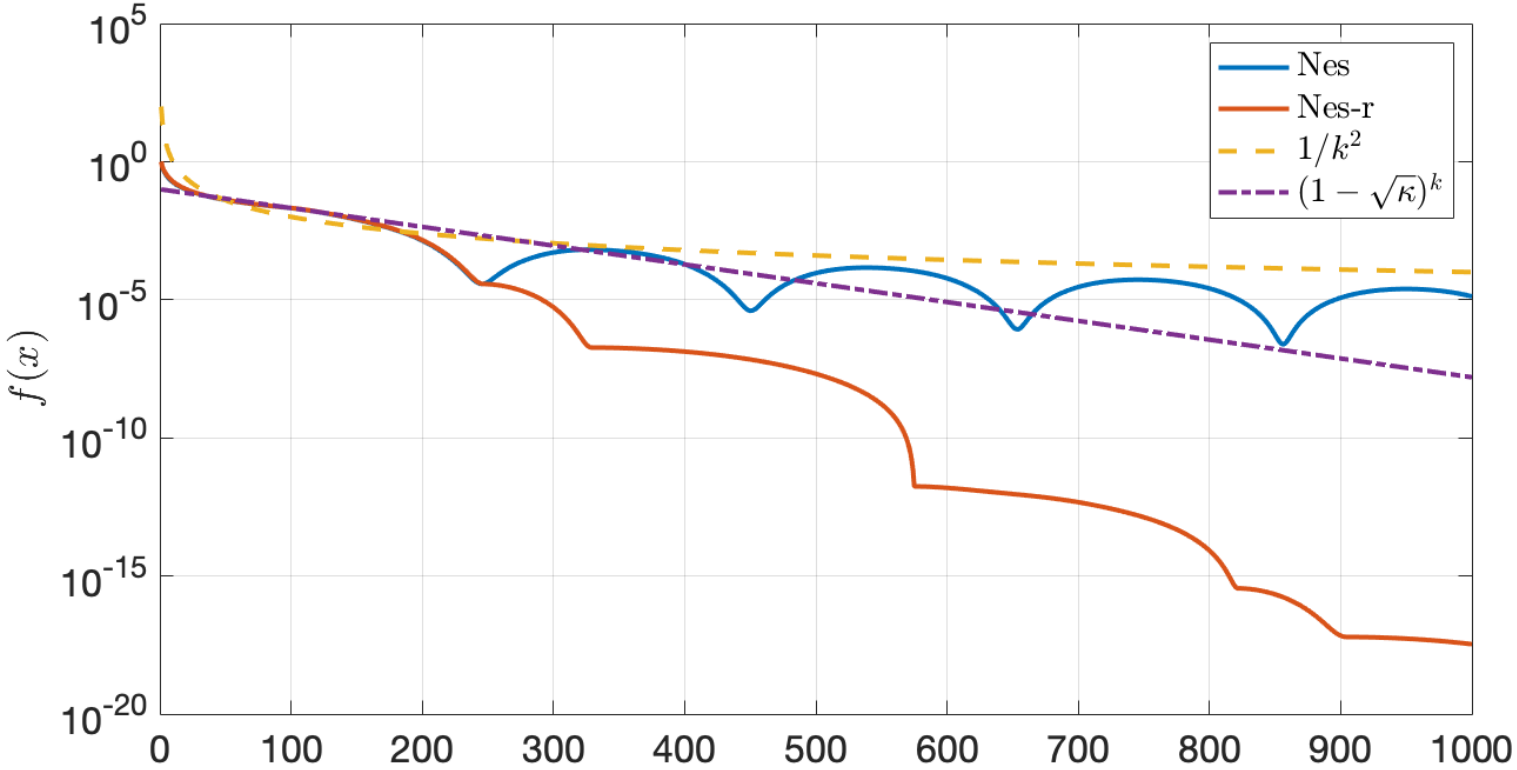


Figure: $f(x) = \frac{1}{2}x^T Ax$, the condition number of A is $\kappa = 4 \times 10^3$.

SDC for composite optimization problems

- Consider the composite problem (1) with ψ is smooth and convex.
- Given the convex function h and the step size $s > 0$, the proximal mapping of h is defined as

$$\mathrm{prox}_h^s(\mathbf{x}) = \arg\min_{\mathbf{z}} \left(\frac{1}{2s}\|\mathbf{z} - \mathbf{x}\|^2 + h(\mathbf{z}) \right).$$

- The proximal gradient is defined by

$$G_s(\mathbf{x}) = \frac{\mathbf{x} - \mathrm{prox}_h^s(\mathbf{x} - s\nabla\psi(\mathbf{x}))}{s}.$$

- Here we present two ways to modify SDC for composite optimization problems.

The proximal gradient

- The first way is to use the proximal gradient.
- Simply replace $\nabla f(\mathbf{x})$ in (4) by the proximal gradient $G_s(\mathbf{x})$.
- The restarting criterion uses the quantity

$$\varphi_k = \langle \mathbf{u}_k, -G_{s_k}(\mathbf{x}_k) \rangle .$$

- If $\varphi_k \geq 0$, then update $\beta_{k+1}, \gamma_{k+1}$ and

$$\mathbf{u}_{k+1} = (1-\beta_k)\mathbf{u}_k - \gamma_k \frac{\|\mathbf{u}_k\|}{\|G_{s_k}(\mathbf{x}_k)\|} G_{s_k}(\mathbf{x}_k) - G_{s_k}(\mathbf{x}_k).$$

- Otherwise, reset $\beta_{k+1}, \gamma_{k+1}$ and

$$\mathbf{u}_{k+1} = -G_{s_k}(\mathbf{x}_k).$$

- Update $\mathbf{x}_{k+1} = \mathbf{x}_k + s_k \mathbf{u}_{k+1}$.

The proximal mapping

- The second way is to use the proximal mapping.
- Introduce an auxiliary variable $\mathbf{y} \in \mathbb{R}^n$ and start with $\mathbf{x}_0 = \mathbf{x}_{-1}$.
- The restarting criterion uses the quantity:

$$\varphi_k = \langle \mathbf{x}_k - \mathbf{x}_{k-1}, -G_{s_k}(\mathbf{x}_k) \rangle .$$

- If $\varphi_k \geq 0$, compute

$$\mathbf{y}_k = \mathbf{x}_k + (1-\beta_k)(\mathbf{x}_k - \mathbf{x}_{k-1}) - \gamma_k \frac{\|\mathbf{x}_k - \mathbf{x}_{k-1}\|}{\|G_{s_k}(\mathbf{x}_k)\|} G_{s_k}(\mathbf{x}_k).$$

  Update $\beta_{k+1}, \gamma_{k+1}$ and

$$\mathbf{x}_{k+1} = \mathbf{y}_k - \bar{s}_k G_{\bar{s}_k}(\mathbf{y}_k).$$

- Otherwise, reset $\beta_{k+1}, \gamma_{k+1}$ and update

$$\mathbf{x}_{k+1} = \mathbf{x}_k - s_k G_{s_k}(\mathbf{x}_k).$$

SDC for stochastic composite optimization problems

- Consider the stochastic composite optimization problem (1), where ψ has the form (2) and each ϕ_i is convex.
- The proximal stochastic gradient is calculated by

$$G_{s_k}(\mathbf{x}) = \frac{\mathbf{x} - \mathrm{prox}_h^{s_k}(\mathbf{x} - s\nabla\psi^{(k)}(\mathbf{x}))}{s_k}.$$

  where $\psi^{(k)}(\mathbf{x})$ is the stochastic approximation of the gradient.
- In each iteration, $\psi^{(k)}(\mathbf{x})$ can be generated via selecting sub-samples $\mathcal{T}_k \subset [N]$ uniformly at random.

$$\nabla\psi^{(k)}(\mathbf{x}) = \frac{1}{|\mathcal{T}_k|}\sum_{i\in\mathcal{T}_k}\nabla\psi_i(\mathbf{x}). \tag{8}$$

SDC for stochastic composite optimization problems

- The variance reduced version of stochastic gradient is also adopted. With $m \in \mathbb{N}$, the stochastic oracle $\psi^{(k)}(\mathbf{x})$ can be computed by:

$$\begin{cases} \text{If } \ k \bmod m = 0 \quad \text{then} \quad \text{set } \tilde{\mathbf{x}} = \mathbf{x}_k \text{ and calculate } \nabla\psi(\tilde{\mathbf{x}}). \\ \text{Compute } \nabla\psi^{(k)}(\mathbf{x}_k) = \dfrac{1}{|\mathcal{T}_k|} \displaystyle\sum_{i \in \mathcal{T}_k} (\nabla\psi_i(\mathbf{x}) - \nabla\psi_i(\tilde{\mathbf{x}})) + \nabla\psi(\tilde{\mathbf{x}}). \end{cases} \tag{9}$$

- m is the number of iterations after which the full gradient $\nabla\psi$ is evaluated at the auxiliary variable $\tilde{\mathbf{x}}$.
- The additional noise-free information $\nabla\psi(\tilde{\mathbf{x}})$ is stored and utilized in the computation of the stochastic oracles in the following iterations.

SDC in deep learning

- Calculate the "momentum and gradient update" on $\mathbf{u}_k$.

$$\tilde{\mathbf{u}}_k = \alpha \mathbf{u}_k - \mathbf{g}_k,$$

  where $\mathbf{g}_k$ is the stochastic gradient of f evaluated at $\mathbf{x}_k$.
- The restarting criterion uses

$$\varphi_k = \langle \tilde{\mathbf{u}}_k, -\mathbf{g}_k \rangle .$$

- If $\varphi_k \geq 0$, calculate $\mathbf{u}_{k+1}$ by correcting $\tilde{\mathbf{u}}_k$ to

$$\mathbf{u}_{k+1} = (1-\beta_k)\tilde{\mathbf{u}}_k - \gamma_k \frac{\|\tilde{\mathbf{u}}_k\|}{\|\mathbf{g}_k\|}\mathbf{g}_k. \tag{10}$$

- Otherwise, set

$$\mathbf{u}_{k+1} = -\mathbf{g}_k.$$

- Update $\mathbf{x}_{k+1} = \mathbf{x}_k + s_k \mathbf{u}_{k+1}$.

SDC in continuous time

- By rescaling $\mathbf{v}_k = \sqrt{s}\mathbf{u}_k$ and taking the fixed step size $s \to 0$,

$$\begin{cases} \dot{\mathbf{v}} = -\nabla f(\mathbf{x}) - \beta(t)\mathbf{v} + \gamma(t)\dfrac{||\mathbf{v}||}{||\nabla f(\mathbf{x})||}\nabla f(\mathbf{x}), \\ \dot{\mathbf{x}} = \mathbf{v}. \end{cases} \tag{11}$$

- (11) is equivalent to a second-order ODE

$$\ddot{\mathbf{x}} + \nabla f(\mathbf{x}) + \beta(t)\dot{\mathbf{x}} + \gamma(t)\frac{||\dot{\mathbf{x}}||}{||\nabla f(\mathbf{x})||}\nabla f(\mathbf{x}) = 0.$$

- Using the symplectic Euler scheme, the discretization of (11) recovers the update rule of methods with SDC.

ODE of FISC

- With $r \geq 3$, the ODE of FISC reads

$$\ddot{\mathbf{x}} + \frac{r}{t}\dot{\mathbf{x}} + \nabla f(\mathbf{x}) + \frac{r-3}{t}\frac{\|\dot{\mathbf{x}}\|}{\|\nabla f(\mathbf{x})\|}\nabla f(\mathbf{x}) = 0. \qquad \text{(FISC-ODE)}$$

- The Lyapunov function for (FISC-ODE) is given by

$$\mathcal{E}(t) = \frac{(r-3)t^2}{4(r-1)^2}\|\dot{\mathbf{x}}\|^2 + \frac{1}{2}\left\|\mathbf{x} - \mathbf{x}^* + \frac{t}{r-1}\dot{\mathbf{x}}\right\|^2 + \frac{t^2}{2(r-1)}(f(\mathbf{x}) - f(\mathbf{x}^*)).$$

Convergence rate of FISC-ODE

Lemma

If f is convex and smooth, then $\mathcal{E}(t)$ satisfies $\dot{\mathcal{E}}(t) \leq 0$.

Theorem

For any $r \geq 3$, let $\mathbf{x}(t)$ be the solution to (FISC-ODE) with initial conditions $\mathbf{x}(0) = \mathbf{x}_0$ and $\dot{\mathbf{x}}(0) = 0$. Then, for $t > 0$, we have

$$f(\mathbf{x}(t)) - f(\mathbf{x}^*) \leq \frac{(r-1)\|\mathbf{x}_0 - \mathbf{x}^*\|^2}{t^2}.$$

ODE of Nesterov accelerated method

- With $r \geq 3$, the ODE of Nesterov accelerated method[1] reads

$$\ddot{\mathbf{x}} + \frac{r}{t}\dot{\mathbf{x}} + \nabla f(\mathbf{x}) = 0. \tag{12}$$

- The Lyapunov function for this ODE is given by

$$\mathcal{E}(t) = \frac{r-1}{4}\left\|\mathbf{x} - \mathbf{x}^* + \frac{t}{r-1}\dot{\mathbf{x}}\right\|^2 + \frac{t^2}{2(r-1)}(f(\mathbf{x}) - f(\mathbf{x}^*)).$$

[1] Su et al. "A Differential Equation for Modeling Nesterovs Accelerated Gradient Method: Theory and Insights". In: Advances in Neural Information Processing Systems (2014).

The global convergence of methods with SDC

- We consider the case where the objective function is smooth, i.e., $h = 0$ in (1).

Lemma

$\mathbf{u}_{k+1}$ is updated by (4) or (9) depending on the restarting criterion using φ_k, and $\mathbf{x}_{k+1}$ is updated by (7). For any integer $k \geq 0$, we have

$$\langle \mathbf{u}_{k+1}, -\nabla f(\mathbf{x}_k) \rangle \geq \|\nabla f(\mathbf{x}_k)\|^2. \tag{13}$$

The global convergence of methods with SDC

- We add two restarting criteria:

$$d_f \|\nabla f(\mathbf{x}_{k+1})\| \geq \|\nabla f(\mathbf{x}_k)\|, \tag{14}$$

$$n_k \leq K, \tag{15}$$

where $K \in \mathbb{N}$, $d_f > 1$ and n_k is the number of iterations since the last restart.

Lemma

$K' < K$ is an integer and both $\varphi_k = \langle -\nabla f(\mathbf{x}_k), \mathbf{u}_k \rangle \geq 0$ and (14) hold for $0 \leq k \leq K'$. Then, $\|\mathbf{u}_{k+1}\|/\|\nabla f(\mathbf{x}_k)\|$ is upper bounded for all $0 \leq k \leq K'$.

The global convergence of methods with SDC

- The direction assumption[1] holds. Namely, there exists positive constants c_1 and c_2 such that

$$\langle \mathbf{u}_{k+1}, \nabla f(\mathbf{x}_k) \rangle \leq -c_1 \|\nabla f(\mathbf{x}_k)\|^2, \quad \|\mathbf{u}_{k+1}\| \leq c_2 \|\nabla f(\mathbf{x}_k)\|. \quad (16)$$

- The step sizes s_k are obtained by the nonmonotone line search.
- According to Theorem 2.2[1] , we obtain

$$\lim_{k\to\infty} \inf \|\nabla f(\mathbf{x}_k)\| = 0.$$

- Moreover, if $\eta_{max} < 1$ (η_{max} is a parameter for the nonmonotone line search), then we have

$$\lim_{k\to\infty} \|\nabla f(\mathbf{x}_k)\| = 0.$$

[1] Hongchao Zhang and William W. Hager. "A nonmonotone line search technique and its application to unconstrained optimization. In: SIAM Journal on Optimization 14.4 (2004), pp. 10431056.

The $\mathcal{O}(1/k^2)$ convergence rate of FISC-PM

- The smooth part ψ is convex and L-smooth.
- The step size is fixed to be $0 < s \leq 1/L$ and no restarts are used.
- Consider the discrete Lyapunov function

$$\begin{aligned}\mathcal{E}(k) =& 2\left\|\mathbf{x}_k - \mathbf{x}^* + \frac{k-1}{r-1}(\mathbf{x}_k - \mathbf{x}_{k-1})\right\|^2 \\ &+ \frac{2(k+r-2)^2 s}{r-1}(f(\mathbf{x}_k) - f(\mathbf{x}^*)) \\ &+ \frac{(r-3)(k-1)^2}{(r-1)^2}\|\mathbf{x}_k - \mathbf{x}_{k-1}\|^2.\end{aligned} \tag{17}$$

The $\mathcal{O}(1/k^2)$ convergence rate of FISC-PM

Lemma

The discrete Lyapunov function $\mathcal{E}(k)$ given by (17) satisfies

$$\begin{aligned}&\mathcal{E}(k)-\mathcal{E}(k-1)\\ \leq&\alpha(\phi_{k-1}-2)\|\Delta\mathbf{x}_{k-1}\|^2-\alpha\phi_k\|\Delta\mathbf{x}_k\|^2-\frac{2s}{r-1}(f(\mathbf{x}_{k-1})-f(\mathbf{x}^*)),\end{aligned}\tag{18}$$

where

$$\alpha=\frac{r-3}{r-1},\ \phi_k=2k+r-3,\ \Delta\mathbf{x}_k=\mathbf{x}_k-\mathbf{x}_{k-1}.\tag{19}$$

Lemma (Discrete Lyapunov analysis of FISC-PM)

The Lyapunov function $\mathcal{E}(k)$ *defined in* (17) *satisfies*

$$\mathcal{E}(k) \leq \mathcal{E}(0) - \frac{2s}{r-1}(f(\mathbf{x}_0) - f(\mathbf{x}^*)). \tag{20}$$

Theorem (The $\mathcal{O}(k^{-2})$ convergence rate of FISC-PM)

$$f(\mathbf{x}_k) - f(\mathbf{x}^*) \leq \frac{(r-1)C_0}{2(k+r-2)^2 s} = \mathcal{O}(k^{-2}),$$

where

$$C_0 = \mathcal{E}(0) - \frac{2s}{r-1}(f(\mathbf{x}_0) - f(\mathbf{x}^*)) = 2\|\mathbf{x}_0 - \mathbf{x}^*\|^2 + (r-3)s\left(f(\mathbf{x}_0) - f(\mathbf{x}^*)\right).$$

Sparse optimization

- We compare FIRE, FISC and other optimization solvers on the following problem:

$$\min_{\mathbf{x}\in\mathbb{R}^n} \frac{1}{2}\|A\mathbf{x}-b\|^2+\lambda\|\mathbf{x}\|_1,$$

  where $A\in\mathbb{R}^{m\times n}, b\in\mathbb{R}^m, \lambda>0$.
- $b=A\bar{\mathbf{x}}+w$, where w is a Gaussian noise with a standard deviation $\bar{\sigma}=0.1$. $\bar{\mathbf{x}}\in\mathbb{R}^n$ has k nonzero entries $\bar{\mathbf{x}}_i=\pm 10^{dc(i)/20}$, where $c(i)$ is uniformly distributed in $[0,1]$.
- $A\bar{\mathbf{x}}=(\mathrm{dct}(\bar{\mathbf{x}}))_J$ are m random cosine measurements. $\lambda\approx 8\times 10^{-3}$, $n=262144, m=32768$, $k=5553$ and $d=40,80$.
- Terminate all methods by the relative criterion $f(\mathbf{x}^k)\leq f(\mathbf{x}^*)$. where $\mathbf{x}^*$ is generated by ASSN satisfying $\|sG_s(\mathbf{x}^*)\|\leq\epsilon$.

Method	$\epsilon : 10^0$		$\epsilon : 10^{-1}$		$\epsilon : 10^{-2}$		$\epsilon : 10^{-4}$		$\epsilon : 10^{-6}$	
	Time	N_A	Time	N_A	Time	N_A	Time	N_A	Time	N_A
SNF	2.06	158.2	5.01	380.8	6.19	483.2	6.69	525.0	7.16	566.8
SNF(aCG)	2.08	158.2	4.97	380.8	6.16	483.2	7.07	553.6	7.30	580.0
ASSN	2.28	182.2	3.53	285.4	4.10	338.6	4.97	**407.0**	**5.56**	**459.2**
FPC-AS	2.12	158.0	5.34	399.2	7.72	578.4	9.62	720.2	10.41	774.8
SpaRSA	5.05	523.4	5.07	530.0	5.56	588.2	6.38	671.6	7.28	755.8
F-PG	4.28	378.0	5.76	522.4	7.28	642.8	9.28	813.6	11.05	990.0
FS-PG(3)	1.71	153.6	3.05	276.4	3.94	354.6	4.89	439.6	6.37	567.2
FS-PG(5)	**1.62**	**143.6**	**2.72**	**245.6**	**3.46**	**317.6**	**4.41**	415.6	5.68	518.0
F-PM	2.02	171.2	2.68	244.0	3.94	347.4	5.34	480.8	7.09	626.2
FS-PM(3)	2.11	184.2	3.14	279.8	4.68	424.0	7.29	648.2	10.11	903.4
FS-PM(5)	2.17	191.2	3.50	308.8	4.54	401.4	6.07	537.0	8.25	716.8

Table: Total number of A-calls and A^T-calls N_A and CPU time (in seconds) averaged over 10 independent runs with dynamic range $40dB$.

Method	$\epsilon : 10^0$		$\epsilon : 10^{-1}$		$\epsilon : 10^{-2}$		$\epsilon : 10^{-4}$		$\epsilon : 10^{-6}$	
	Time	N_A	Time	N_A	Time	N_A	Time	N_A	Time	N_A
SNF	7.65	591.0	10.87	841.6	12.49	978.6	13.08	1024.8	15.89	1227.6
SNF(aCG)	7.58	591.0	10.78	841.6	12.44	978.6	13.30	1042.2	13.99	1105.8
ASSN	5.96	482.8	7.47	601.0	8.39	690.6	9.52	780.6	10.32	852.6
FPC-AS	**4.28**	**321.4**	8.28	611.0	10.61	788.0	11.85	883.2	12.13	902.0
SpaRSA	5.18	543.2	6.26	665.4	7.35	763.0	8.26	871.8	8.98	942.0
F-PG	7.18	642.8	8.90	792.8	10.35	951.0	12.47	1134.8	13.50	1231.6
FS-PG(3)	4.85	444.8	6.09	555.4	7.01	649.2	7.76	727.0	8.65	789.2
FS-PG(5)	4.30	407.2	5.72	521.6	6.77	625.8	**7.64**	**702.0**	**8.15**	**753.2**
F-PM	4.17	388.8	**5.26**	**463.2**	**6.55**	583.2	8.14	729.2	9.06	814.6
FS-PM(3)	6.00	533.4	6.87	635.4	8.41	748.4	13.08	1162.8	15.04	1348.4
FS-PM(5)	4.99	436.4	5.75	525.0	7.08	639.8	9.51	860.0	10.93	987.0

Table: Total number of A-calls and A^T-calls N_A and CPU time (in seconds) averaged over 10 independent runs with dynamic range $80dB$.

Sparse Logistic regression

- We consider the ℓ_1-logistic regression problem

$$\min_{\mathbf{x}=(\hat{\mathbf{x}},y)\in\mathbb{R}^{n+1}} \frac{1}{N}\sum_{i=1}^{N}\log(1+\exp(-b_i(\langle \mathbf{a}_i,\hat{\mathbf{x}}\rangle+y)))+\lambda\|\mathbf{x}\|_1,$$

  where data pairs $(\mathbf{a}_i,b_i)\in\mathbb{R}^n\times\{-1,1\}$, correspond to a given data set. λ is set to be 0.001.

- For all solvers, the sample size is fixed to be $|\mathcal{S}_k|=\lfloor 0.01N\rfloor$. In SVRG, we set $m=200$ in (9); in sFVR-PG/sFSVR-PG, we set $m=20$ in (9). Here we intentionally set a larger m in SVRG because it generates a higher precision solution.

Data sets

- The tested data sets obtained from libsvm in our numerical comparison are summarized in the following table.
- Except for the large data set tfidf, we linearly scale the entry of the data-matrix $\mathbf{A} = (\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_N)$ to $[0, 1]$. Then, we add a row of ones into the data-matrix $\mathbf{A}$ as coefficients for the bias term in our linear classifier.

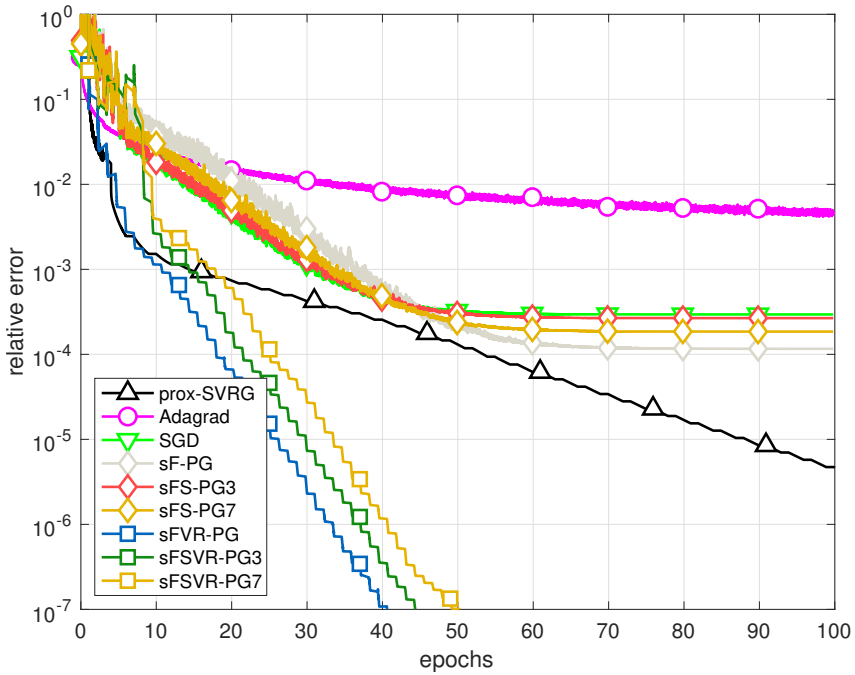
Data Set	Data Points N	Variables n	Density
rcv1	$20,242$	$47,236$	0.16%
MNIST	$60,000$	784	19.12%
mushroom	8,124	112	18.75%
tfidf	16,087	150,360	0.83%

Table: Information of the data sets in ℓ_1-logistic regression

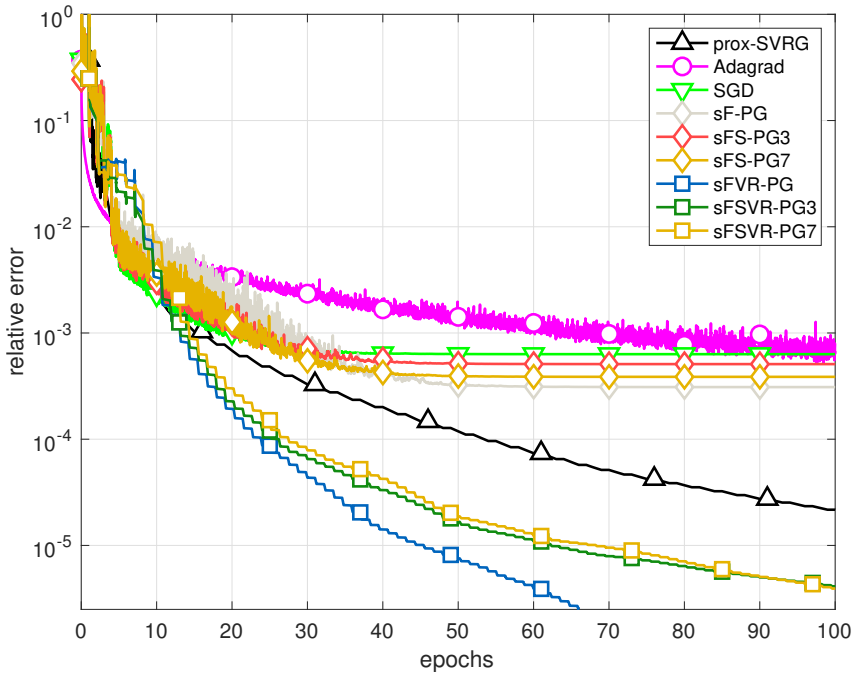
Implementation details

- The initial step sizes varies for different tested data sets and it determines the performance of solvers. Hence, we chose the initial step size from set $\{2^i | i \in \{-7, -6, \ldots, 7\}\}$.
- For SGD, sF(S)-PG and sF(S)VR-PG, we use an exponentially decaying step size. Namely, we decrease the step size by multiplying 0.85 in each epoch.
- For all methods, we choose $\mathbf{x}_0 = 0$ as the initial point.
- The change of the *relative error* $(f(\mathbf{x}) - f(\mathbf{x}^*))/(\max\{1, |f(\mathbf{x}^*)|\})$ is reported with respect to *epochs*. Here x^* is a reference solution generated by S2N-D[1] with a stopping criterion $\|\mathbf{x}_k - \mathbf{x}_{k-1}\| < 10^{-12}$.

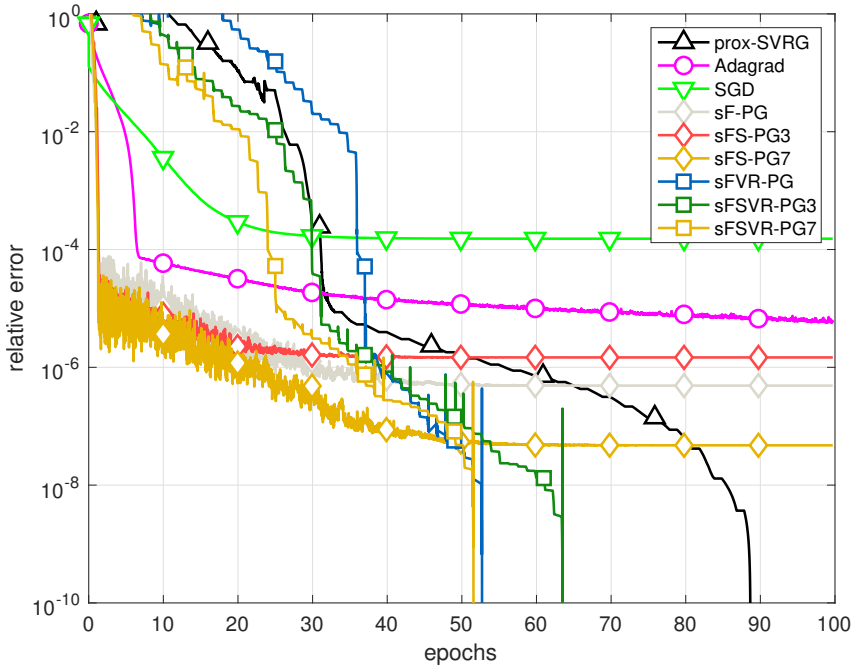
[1] Andre Milzarek et al."A Stochastic Semismooth Newton Method for Nonsmooth Nonconvex Optimizatio". In: arXiv:1803.03466 (2018).


relative error
epochs
prox-SVRG
Adagrad
SGD
sF-PG
sFS-PG3
sFS-PG7
sFVR-PG
sFSVR-PG3
sFSVR-PG7

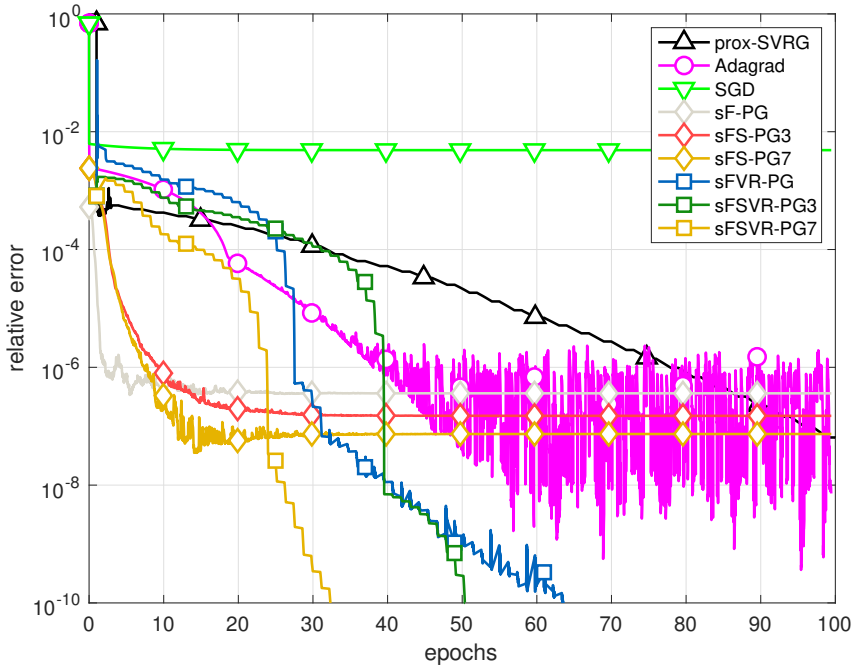
rcv1


relative error
epochs
prox-SVRG
Adagrad
SGD
sF-PG
sFS-PG3
sFS-PG7
sFVR-PG
sFSVR-PG3
sFSVR-PG7

MNIST



mushroom



tfidf

Deep learning

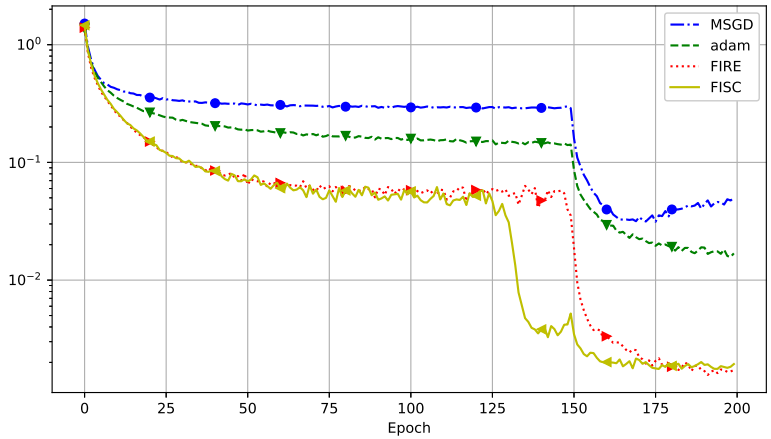
- The optimization problem in deep learning is

$$\min_{\mathbf{x}\in\mathbb{R}^n} \frac{1}{N}\sum_{i=1}^{N} l\left(f\left(\mathbf{a}^{(i)},\mathbf{x}\right),b^{(i)}\right)+\lambda\|\mathbf{x}\|_2^2,$$

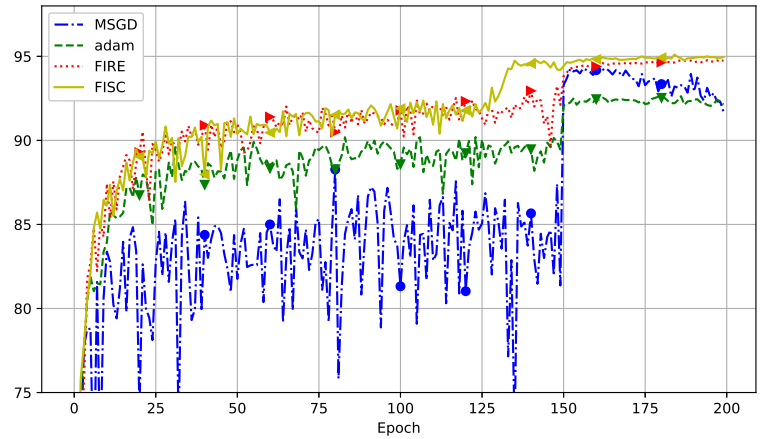
  where $\mathbf{x}$ denotes the parameters for training, data pairs $\{(\mathbf{a}^{(i)},b^{(i)})\}$ correspond to a given data set, $f(\cdot,\mathbf{x})$ represents the function determined by the neural network architecture, $l(\cdot,\cdot)$ denotes the loss function and λ is the coefficient of weight decay (ℓ_2-regularization).
- Data sets: CIFAR-10 and CIFAR-100.
- Neural network architectures: DenseNet121 and ResNet34.

Implementation details

- On CIFAR-10, we train the network using a batch size 128 for 200 epochs. The coefficient λ is 5×10^{-4}. The learning rate is decreased 10 times at epoch 150.
- On CIFAR-100, we train the network using a batch size 64 for 300 epochs and λ is 1×10^{-4}. The learning rate is multiplied by 0.1 at epoch 150 and epoch 225.
- For both data sets, the momentum factor is 0.9 in MSGD, FIRE and FISC; (β_1, β_2) in Adam are $(0.9, 0.999)$ on DenseNet and $(0.99, 0.999)$ on ResNet; ϵ in Adam is 10^{-8}.

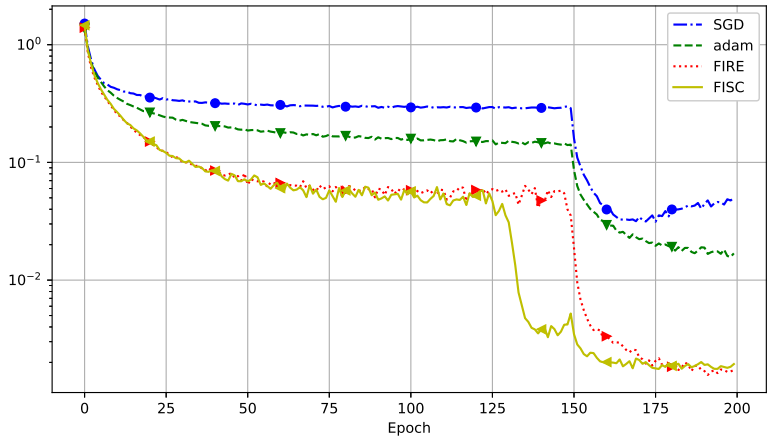


Training Loss

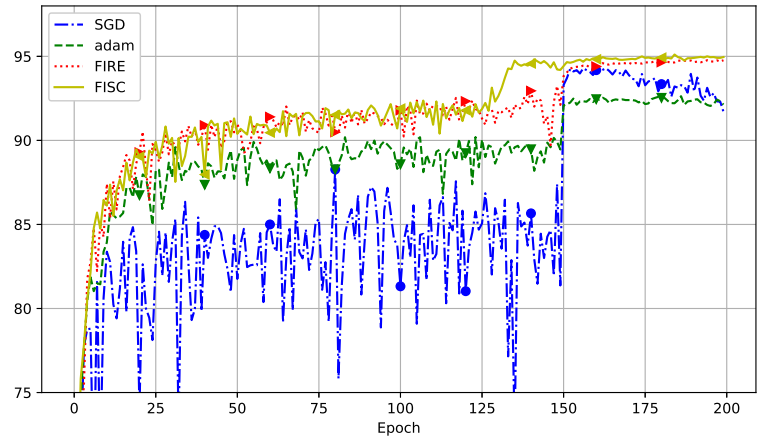


Test Accuracy

Figure: CIFAR-10, DenseNet121.

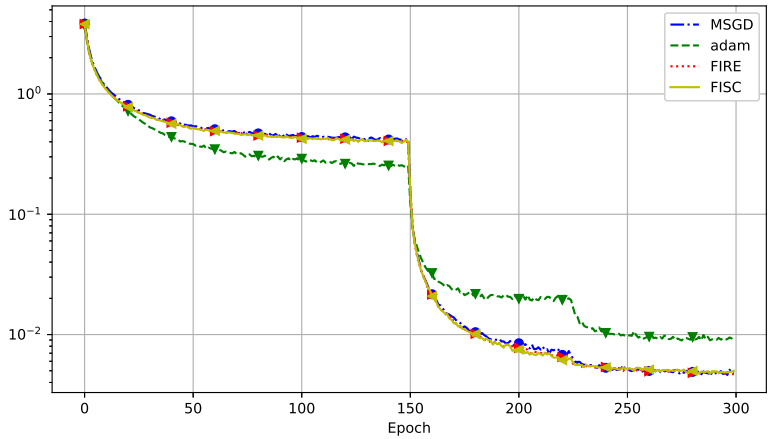


Training Loss

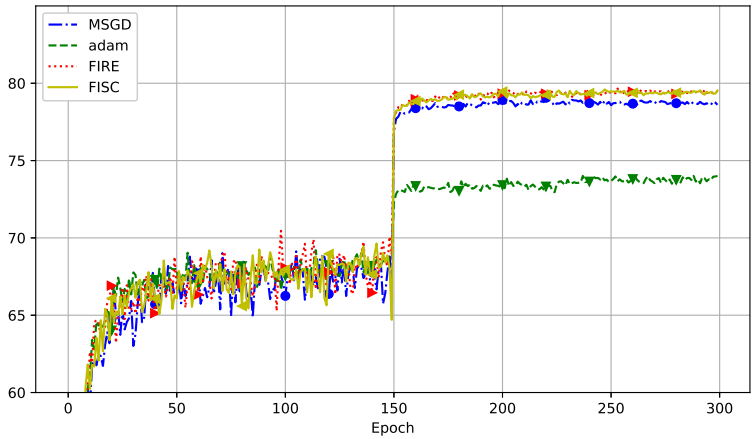


Test Accuracy

Figure: CIFAR-10, ResNet34.

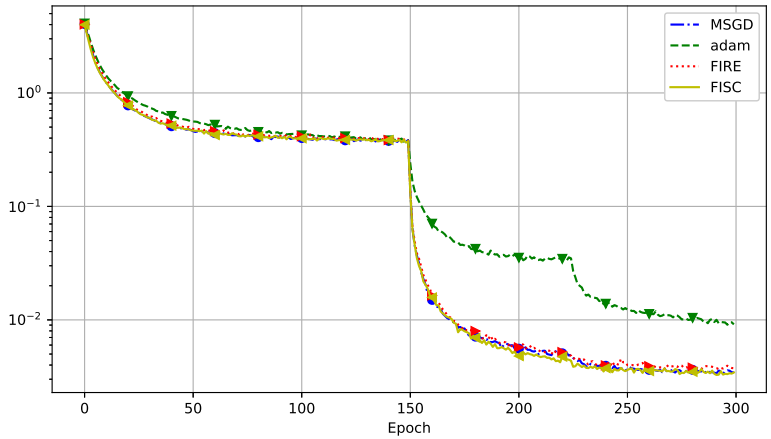


Training Loss

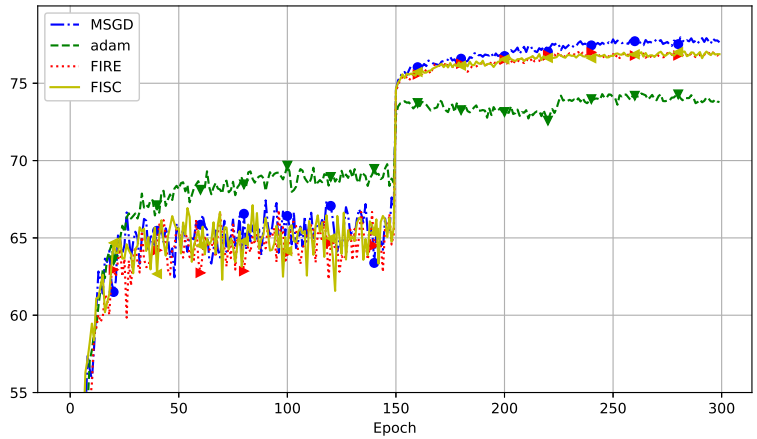


Test Accuracy

Figure: CIFAR-100, DenseNet121.



Training Loss



Test Accuracy

Figure: CIFAR-100, ResNet34.

Conclusion

- We propose a family of first-order methods with SDC.
- The restarting criterion is the foundation for the global converge of methods with SDC and leads to better numerical results.
- From an ODE perspective, we construct the FISC-ODE with an $\mathcal{O}(t^{-2})$ convergence rate. In the discrete case, FISC-PM has a provable $\mathcal{O}(k^{-2})$ convergence rate.
- Numerical experiments indicate that our algorithmic framework with SDC is competitive and promising.