Stochastic Cubic Regularization for Fast Nonconvex Optimization

Moscow, MIPT, 2020

Kyzyl-ool Kezhik,

Nursultan Kozhogulov

Ehson Kholzoda

Plan

- How does this method differ from classical gradient descent?
- How faster this method than other methods?
- Our experiments and confirmations / refutations

Gradient descent and Newton method

Classical gradient descent:

$$\mathbf{x}_{t+1}^{\mathrm{GD}} = \operatorname*{argmin}_{\mathbf{x}} \left[f(\mathbf{x}_t) + \nabla f(\mathbf{x}_t)^\top (\mathbf{x} - \mathbf{x}_t) + \frac{\ell}{2} \|\mathbf{x} - \mathbf{x}_t\|^2 \right],$$

Cubic regularized Newton method:

$$\mathbf{x}_{t+1}^{\mathrm{Cubic}} = \operatorname*{argmin}_{\mathbf{x}} \left[f(\mathbf{x}_t) + \nabla f(\mathbf{x}_t)^\top (\mathbf{x} - \mathbf{x}_t) + \frac{1}{2} (\mathbf{x} - \mathbf{x}_t)^\top \nabla^2 f(\mathbf{x}_t) (\mathbf{x} - \mathbf{x}_t) + \frac{\rho}{6} \|\mathbf{x} - \mathbf{x}_t\|^3 \right].$$

Comparing with other methods

Method	Runtime	Variance Reduction
Stochastic Gradient Descent [Ge et al., 2015]	$\mathcal{O}(\epsilon^{-4}\text{poly}(d))$	not needed
Natasha 2 [Allen-Zhu, 2017]	$\tilde{\mathcal{O}}(\epsilon^{-3.5})^2$	needed
Stochastic Cubic Regularization (this paper)	$\tilde{\mathcal{O}}(\epsilon^{-3.5})$	not needed

Meta-algorithm

Algorithm 1 Stochastic Cubic Regularization (Meta-algorithm)

Input: mini-batch sizes n_1, n_2, initialization $\mathbf{x_0}$, number of iterations T_{out}, and final tolerance ϵ.

for $t = 0, \ldots, T_{\text{out}}$ **do**
  Sample $S_1 \leftarrow \{\xi_i\}_{i=1}^{n_1}$, $S_2 \leftarrow \{\xi_i\}_{i=1}^{n_2}$.
  $\mathbf{g}_t \leftarrow \frac{1}{|S_1|} \sum_{\xi_i \in S_1} \nabla f(\mathbf{x}_t;\ \xi_i)$
  $\mathbf{B}_t[\cdot] \leftarrow \frac{1}{|S_2|} \sum_{\xi_i \in S_2} \nabla^2 f(\mathbf{x}_t, \xi_i)(\cdot)$
  $\mathbf{\Delta}, \Delta_m \leftarrow$ Cubic-Subsolver$(\mathbf{g}_t,\ \mathbf{B}_t[\cdot], \epsilon)$
  $\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t + \mathbf{\Delta}$
  if $\Delta_m \geq -\frac{1}{100}\sqrt{\frac{\epsilon^3}{\rho}}$ **then**
    $\mathbf{\Delta} \leftarrow$ Cubic-Finalsolver$(\mathbf{g}_t,\ \mathbf{B}_t[\cdot], \epsilon)$
    $\mathbf{x}^* \leftarrow \mathbf{x}_t + \mathbf{\Delta}$
    break
  end if
end for

Output: $\mathbf{x}^*$ if the early termination condition was reached, otherwise the final iterate $x_{T_{\text{out}}+1}$.

Cubic-Subsolver

Algorithm 2 Cubic-Subsolver via Gradient Descent

Input: $\mathbf{g}$, $\mathbf{B}[\cdot]$, tolerance ϵ.

if $\|\mathbf{g}\| \geq \frac{\ell^2}{\rho}$ **then**

  $R_c \leftarrow -\frac{\mathbf{g}^\top \mathbf{B}[\mathbf{g}]}{\rho\|\mathbf{g}\|^2} + \sqrt{\left(\frac{\mathbf{g}^\top \mathbf{B}[\mathbf{g}]}{\rho\|\mathbf{g}\|^2}\right)^2 + \frac{2\|\mathbf{g}\|}{\rho}}$

  $\boldsymbol{\Delta} \leftarrow -R_c \frac{\mathbf{g}}{\|\mathbf{g}\|}$

else

  $\boldsymbol{\Delta} \leftarrow 0, \sigma \leftarrow c' \frac{\sqrt{\epsilon\rho}}{\ell}, \eta \leftarrow \frac{1}{20\ell}$

  $\tilde{\mathbf{g}} \leftarrow \mathbf{g} + \sigma\zeta$ for $\zeta \sim \mathrm{Unif}(\mathbb{S}^{d-1})$

  for $t = 1, \ldots, \mathcal{T}(\epsilon)$ **do**

    $\boldsymbol{\Delta} \leftarrow \boldsymbol{\Delta} - \eta(\tilde{\mathbf{g}} + \mathbf{B}[\boldsymbol{\Delta}] + \frac{\rho}{2}\|\boldsymbol{\Delta}\|\boldsymbol{\Delta})$

  end for

end if

$\Delta_m \leftarrow \mathbf{g}^\top\boldsymbol{\Delta} + \frac{1}{2}\boldsymbol{\Delta}^\top\mathbf{B}[\boldsymbol{\Delta}] + \frac{\rho}{6}\|\boldsymbol{\Delta}\|^3$

Output: $\boldsymbol{\Delta}$, Δ_m

Cubic-Finalsolver

Algorithm 3 Cubic-Finalsolver via Gradient Descent

Input: $\mathbf{g}$, $\mathbf{B}[\cdot]$, tolerance ϵ.

$\boldsymbol{\Delta} \leftarrow 0$, $\mathbf{g}_m \leftarrow \mathbf{g}$, $\eta \leftarrow \frac{1}{20\ell}$

while $\|\mathbf{g}_m\| > \frac{\epsilon}{2}$ **do**

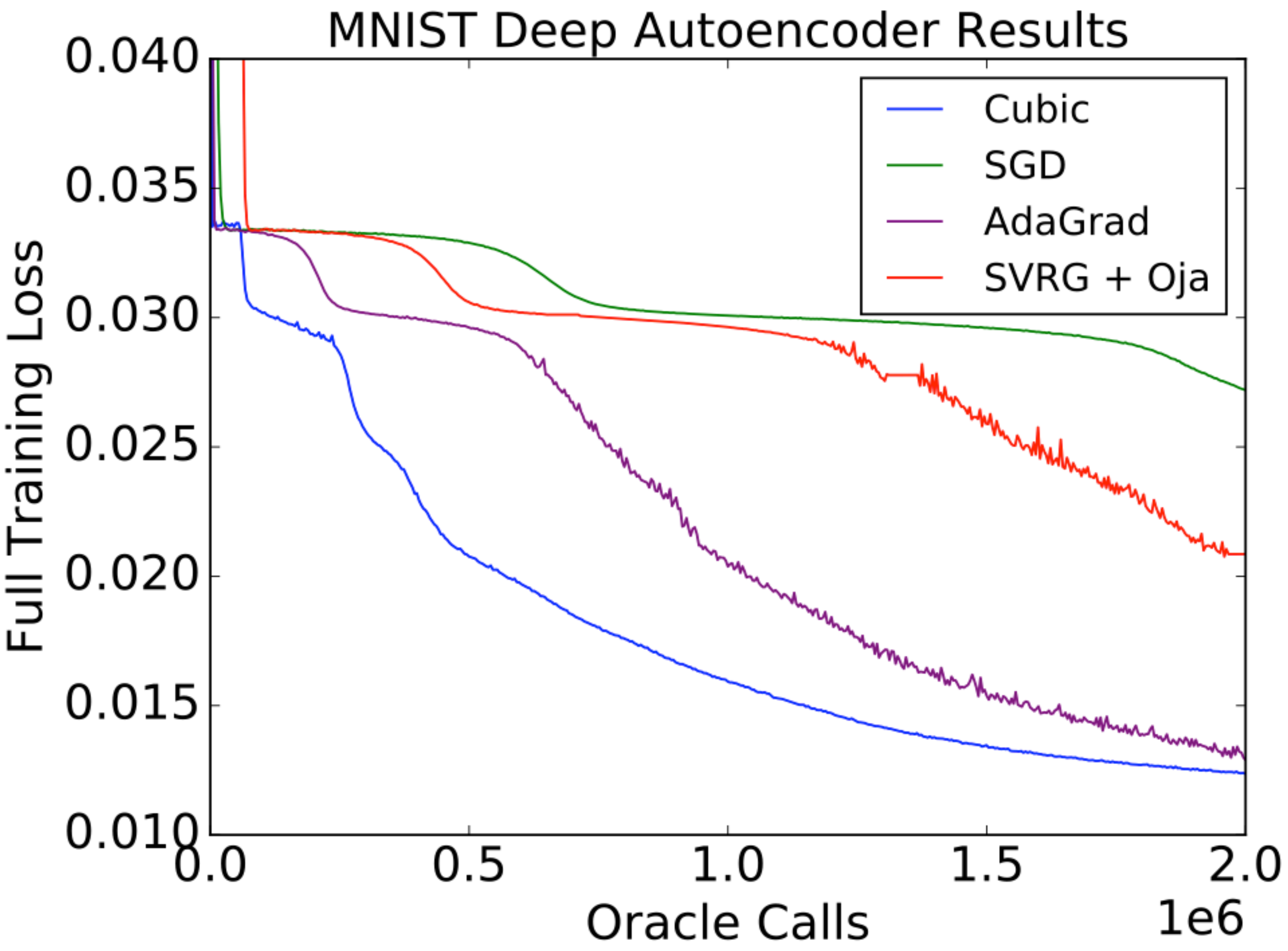
  $\boldsymbol{\Delta} \leftarrow \boldsymbol{\Delta} - \eta \mathbf{g}_m$

  $\mathbf{g}_m \leftarrow \mathbf{g} + \mathbf{B}[\boldsymbol{\Delta}] + \frac{\rho}{2}\|\boldsymbol{\Delta}\|\boldsymbol{\Delta}$

end while

Output: $\boldsymbol{\Delta}$

Method compared with other methods



Our experiments

- To be continued….