Linearly-constrained nonsmooth optimization for training autoencoders

Wei Liu
liuwei175@lsec.cc.ac.cn

State Key Laboratory of Scientific and Engineering Computing
Institute of Computational Mathematics and Scientific/Engineering Computing
Academy of Mathematics and Systems Science
Chinese Academy of Sciences, China

Joint work with **Xin Liu** (AMSS, CAS), and **Xiaojun Chen** (PolyU)

September 17, 2021

1. The optimization problem for training a deep neural network

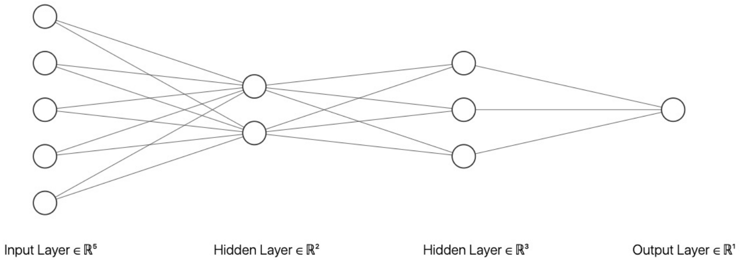
Given an input data $\{(x_n, y_n)\}_{n=1}^{N}$, where $x_n \in \mathbb{R}^{N_0}$, $y_n \in \mathbb{R}^{N_L}$:

♠ **Neural Networks (NN)– deep learning:**

$$(1) \quad \min_{\substack{W_\ell, b_\ell, \\ \ell=1,2,\ldots,L}} \frac{1}{N} \sum_{n=1}^{N} \|\sigma_L(W_L \sigma_{L-1}(\cdots \sigma_1(W_1 x_n + b_1) + b_2 \cdots) + b_L) - y_n\|^2.$$

- N: the number of input data.
- L: the number of layers.
- N_ℓ: the number of neurons at the ℓ-th layer.
- σ_ℓ: the activation function. Here we use ReLU, i.e., $\sigma_\ell(z) = \max(z, 0)$.
- variables to be determined:
  - $W_\ell \in \mathbb{R}^{N_\ell \times N_{\ell-1}}$;
  - $b_\ell \in \mathbb{R}^{N_\ell}$.

An Example



Why we use ReLU?

- ReLU reduces the vanishing gradient problem by making the problem more sparse;
- ReLU becomes the most popular activation functions since 2012 ([Sun 2019]).

An Autoencoder is a special network with two layers.

Existing Approaches: SGD-based Methods

Properties

- calculating the gradient via the chain rule;
- in cases where nonsmooth activation functions are used, a subgradient in a neighborhood is often used.

Some SGD-based Methods

- Vanilla SGD ([Cramir 1946])
- Adagrad ([Duchi-Hazan-Singer 2011])
- AdagradDecay ([Duchi-Hazan-Singer 2011])
- Adadelata ([Zeiler 2012])
- Adam ([Kingma-Ba-Adam 2014])
- Adamax ([Kingma-Ba-Adam 2014])

Existing Approaches: SGD-based Methods (Cont'd)

Limitations

- neglecting the exactness in calculating the subgradient of the objective function;
- for solving a nonsmooth nonconvex problem, whether the SGD-based methods produce a convergent sequences is unknown;
- in practice, SGD-based approaches suffer from vanishing gradient phenomenon, poor conditioning, and saddle points that affect convergence.

An equivalent Model

[Carreira Perpiñán-Wang 2014]

$$
(2) \qquad \begin{aligned} \min_{\substack{W_\ell, u_{n,\ell}, b_\ell, \\ \ell=1,2,\ldots,L,\ n=1,2,\ldots,N}} \quad & \frac{1}{N}\sum_{n=1}^{N} \|u_{n,L} - y_n\|_F^2 \\ \text{s. t.} \quad & u_{n,\ell} = \sigma_\ell(W_\ell u_{n,\ell-1} + b_\ell), \\ & n = 1,\ldots,N,\ \ell = 1,\ldots,L. \end{aligned}
$$

Here $u_{n,0} = x_n$, $u_{n,\ell}$: the output of the ℓ-th layer with respect to the n-th input data.

Solving (2) instead of (1) is able to alleviate the problems that SGD-based methods suffer.

Existing Approaches for Solving the Model (2)

- ℓ_2 penalty method: no exact penalty result.
  - Carreira-Perpinan-Wang 2014
  - Lau-Zeng-Wu-Yao 2018
  - Zeng-Lau-Lin-Yao 2019
- multi-block 'ADMM': hard to deal with the quadratic terms.
  - Taylor et. al. 2016
  - Zhang-Chen-Saligrama 2016
  - Evens-Latafat-Themelis 2020

The above methods have no convergence results!

ℓ_1 penalty method: present the exact penalty result, and establish the subsequence convergence result ([Cui-He-Pang 2020]).

- limitation in theory: restrictive assumptions, which exclude ReLU;
- limitation in practice: cannot solve large-scale problems.

A New Penalty Approach

- using l_1 penalty terms;
- adding additional inequality constraints $u_{n,\ell} \geq 0,\ u_{n,\ell} \geq W_\ell u_{n,\ell-1} + b_\ell$.

$$\begin{aligned}
\min_{\substack{W_\ell, u_{n,\ell}, b_\ell, \\ \ell=1,2,\ldots,L,\ n=1,2,\ldots,N}} \quad & \frac{1}{N}\sum_{n=1}^{N} \|(W_L u_{n,L} + b_L)_+ - y_n\|_2^2 \\
& \qquad + \beta \sum_{n=1}^{N}\sum_{\ell=1}^{L-1} e_{N_\ell}^{\mathrm{T}}(u_{n,\ell} - (W_\ell u_{n,\ell-1} + b_\ell)_+) \\
\text{s. t.} \quad & u_{n,\ell} \geq 0,\ u_{n,\ell} \geq W_\ell u_{n,\ell-1} + b_\ell, \quad n = 1,\ldots,N,\ \ell = 1,\ldots,L.
\end{aligned}$$

where $\beta > 0$, $e_{N_\ell} \in \mathbb{R}^{N_\ell}$ represent a vector whose elements are all 1.

- the subdifferential of the objective function enjoys an explicit expression;
- we first present the exact penalty results, regarding global solutions, local minimizers, Clarke stationary point and directional stationary point.

2. Autoencoders

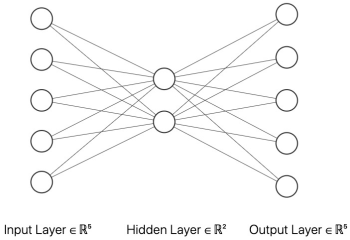
Autoencoders: Unsupervised Learning

Given data matrix $X = (x_1, x_2, \ldots, x_N)$ with $x_n \in \mathbb{R}^{N_0}$, $W \in \mathbb{R}^{N_1 \times N_0}$, $b = ((b_1)^T, (b_2)^T)^T$ with $b_1 \in \mathbb{R}^{N_1}$, $b_2 \in \mathbb{R}^{N_0}$.

♠ **Autoencoder**: A special two-layer network

$$\min_{W,b} \frac{1}{N} \sum_{n=1}^{N} \|\sigma(W^T \sigma(W x_n + b_1) + b_2) - x_n\|_2^2.$$

- Encoder (representation):
  $\varphi_n(W, b) = \sigma(W x_n + b_1)$.
- Decoder (prediction function):
  $\psi_n(W, b) = \sigma(W^T \varphi_n(W, b) + b_2)$.



Features and Notations

Features:

- tied weight: $W = W_1 = W_2^{\mathrm{T}}$.
- $x_n = y_n$.

Notations:

- activation function $\sigma : \mathbb{R}^m \mapsto \mathbb{R}^m$ being ReLU, i.e., $\sigma(y) = \max\{y, 0\} = y_+$.
- variables to be determined: W, b.

Objective: to learn an encoder and an decoder for input X.

Limitation: training the basic model for autoencoders may lead to poor performance.

Limitations of The Existing Methods

SGD-based methods:

- neglecting the exactness in calculating the subgradient of the objective function;
- for solving a nonsmooth nonconvex problem, whether the SGD-based methods produce a convergent sequences is unknown;
- in practice, SGD-based approaches suffer from vanishing gradient phenomenon, poor conditioning, and saddle points that affect convergence.

methods for solving (2):

- due to the special structure $W_1 = W_2^{\mathrm{T}}$, this kind of methods cannot be used

Sparse Autoencoder

♠ **Sparse autoencoder**:

$$\min_{W,b} \frac{1}{N}\sum_{n=1}^{N} \|(W^T(Wx_n+b_1)_+ + b_2)_+ - x_n\|_2^2 + \lambda_1 \sum_{n=1}^{N} e_{N_1}^T (Wx_n+b_1)_+ + \lambda_2\|W\|_F^2.$$

- $\lambda_1, \lambda_2 > 0$, $\lambda_1 \sum_{n=1}^N e^T v_n + \lambda_2\|W\|_F^2$: regularization term.
- widely used in feature learning ([Goodfellow 2016]).

Regularization terms:

- for pursuing sparsity: $\sum_{n=1}^N \|\sigma(Wx_n + b_1)\|_1$ ([Ng et. al. 2011]);
- for avoiding overfitting phenomenon: $\mathcal{R}(W, b) = \|W\|_F^2$, called weight decay ([Krogh-Hertz 1992]).

Equivalent Model with Additional Constraints

An equivalent model with auxiliary variables:

$$(R)\quad \begin{aligned} &\min_{z}\ \frac{1}{N}\sum_{n=1}^{N}\left\|(W^T v_n + b_2)_+ - x_n\right\|_2^2 + \lambda_1 \sum_{n=1}^{N} e^T v_n + \lambda_2\|W\|_F^2 \\ &\text{s. t. } v_n = (W x_n + b_1)_+, \\ &\qquad \text{for } n = 1, 2, \ldots, N. \end{aligned}$$

Notations:

- $e = (1, 1, \ldots, 1)^T \in \mathbb{R}^{N_1}$.
- $V = (v_1, v_2, \ldots, v_N) \in \mathbb{R}^{N_1 \times N}$.
- $z = (W_{\cdot,1}^T, \ldots, W_{\cdot,N_0}^T, b^T, v_1^T, \ldots, v_N^T)^T$.
- reularization term $\mathcal{R}(z) = \lambda_1 \sum_{n=1}^{N} e^T v_n + \lambda_2\|W\|_F^2$.
- fidelity term $\mathcal{F}(z) := \frac{1}{N}\sum_{n=1}^{N}\left\|(W^T v_n + b_2)_+ - x_n\right\|_2^2$.

A New Penalty Model

Let $\beta > 0$.

$$
(RP) \quad \begin{aligned} & \min_z \ \mathcal{O}(z) := \mathcal{F}(z) + \mathcal{R}(z) + \beta \sum_{n=1}^{N} e^T \left(v_n - (Wx_n + b_1)_+\right) \\ & \text{s. t. } v_n \geq (Wx_n + b_1)_+, \\ & \qquad \text{for } n = 1, 2, \ldots, N. \end{aligned}
$$

Notes:

- $\sum_{n=1}^{N} \|v_n - (Wx_n + b_1)_+\|_1$ equals $\sum_{n=1}^{N} e^T \left(v_n - (Wx_n + b_1)_+\right)$.
- penalty term $\mathcal{P}(z) := \beta \sum_{n=1}^{N} e^T \left(v_n - (Wx_n + b_1)_+\right)$.
- Ω_2: the feasible set of (RP).

Linear Constrained Regularized Autoencoder

Define

$$\Omega_3 := \{z : \|b\|_\infty \leq \alpha\},$$

where $\alpha = \frac{\|X\|_F^2}{\lambda_1 N} + \sqrt{\frac{N_1 N_0}{\lambda_2 N}} \|X\|_F \|X\|_1$.
We then reformulate (2) over Ω_3 as

$$(LRP) \qquad \min_{z \in \mathcal{Z}} \mathcal{O}(z),$$

where $\mathcal{Z} := \Omega_2 \cap \Omega_3 = \{z : Az \leq c\}$.
Here, $\nu = 2(NN_1 + N_0 + N_1)$,

$$A = \begin{bmatrix} X^{\mathrm{T}} \otimes I_{N_1} & e_{N_1} \otimes I_{N_1} & 0 & -I_{N_1 N} \\ 0 & 0 & 0 & -I_{N_1 N} \\ 0 & I_{N_1+N_0} & & 0 \\ 0 & -I_{N_1+N_0} & & 0 \end{bmatrix} \in \mathbb{R}^{\nu \times N_2}, \quad c = \begin{bmatrix} 0 \\ 0 \\ \alpha e_{N_1+N_0} \\ \alpha e_{N_1+N_0} \end{bmatrix} \in \mathbb{R}^{\nu},$$

where $\otimes$ represents the Kronecker product.

3. Theoretical Analysis

Preliminaries

- The Clarke subdifferential [Clarke 1990] of a locally Lipschitz continuous function $f : \mathbb{R}^m \to \mathbb{R}$ at y^* is defined by $\partial f(y^*) = \mathrm{co}\left\{\lim_{y \to y^*} \nabla f(y) : f \text{ is smooth at } y\right\}$.
- $f'(y; d)$: the directional derivative of a directional differentiable function f at y along the direction d, i.e.,

$$f'(y; d) = \lim_{t \downarrow 0} \frac{f(y + td) - f(y)}{t}.$$

- $f^\circ(\bar{y}; d) := \limsup_{\substack{y \to \bar{y} \\ t \downarrow 0}} \frac{f(y+td)-f(y)}{t}$ is the Clarke generalized directional derivative at $\bar{y}$ along the direction d.
- A function f is said to be regular [Clarke 1990] at $\bar{y} \in \mathbb{R}^m$ provided that if for all d, the directional derivative $f'(\bar{y}; d)$ exists, and $f'(\bar{y}; d) = f^\circ(\bar{y}; d)$.
- The objective functions of (R), (RP) and (LRP) are then semismooth functions and directional differentiable [Mifflin 1977].

Preliminaries (Cont'd)

We call $\bar{z} \in \Omega_1$, $\bar{z} \in \Omega_2$, $\bar{z} \in \mathcal{Z}$ a d(irectional)-stationary point [Cui et al. 2020] of (R), (RP) and (LRP) respectively, if

$$\mathcal{F}'(\bar{z}; d) + \nabla \mathcal{R}(\bar{z})^{\mathrm{T}} d \geq 0, \quad \forall d \in \mathcal{T}_{\Omega_1}(\bar{z}),$$

$$\mathcal{O}'(\bar{z}; d) \geq 0, \quad \forall d \in \mathcal{T}_{\Omega_2}(\bar{z}),$$

$$\mathcal{O}'(\bar{z}; d) \geq 0, \quad \forall d \in \mathcal{T}_{\mathcal{Z}}(\bar{z}).$$

We call $\bar{z} \in \Omega_1$, $\bar{z} \in \Omega_2$, $\bar{z} \in \mathcal{Z}$ a generalized d(irectional)-stationary point of (R), (RP) and (LRP), respectively, if the above three inequalities hold with $\mathcal{F}^{\circ}(\bar{z}; d)$ and $\mathcal{O}^{\circ}(\bar{z}; d)$ instead of $\mathcal{F}'(\bar{z}; d)$ and $\mathcal{O}'(\bar{z}; d)$.

Main Contributions in theory

- The solution set of problem (LRP) is nonempty and bounded.

R :	global minimizer	local minimizer	d-stationary point	generalized d-stationary point
	$\Downarrow\Uparrow$	$\Downarrow$	$\Uparrow$	$\Uparrow$ $\mathcal{P}$ is regular at $\bar{z}$
RP :	global minimizer	local minimizer	d-stationary point	generalized d-stationary point
	$\bar{z} \in \Omega_3$ $\Downarrow\Uparrow$	$\bar{z} \in \Omega_3$ $\Downarrow\Uparrow$	$\Uparrow$ $\mathcal{O}(\bar{z}) < \theta$	$\Uparrow$ $\bar{z} \in \mathrm{int}(\Omega_3), \mathcal{O}(\bar{z}) < \theta$
LRP :	global minimizer	local minimizer	d-stationary point	generalized d-stationary point

4. A Smoothing Proximal Gradient Algorithm (SPG)

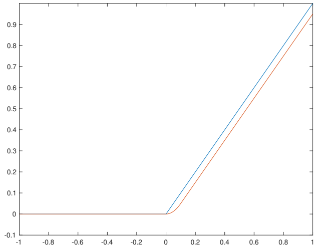
A Smoothing Function for ReLU

Definition 1

Let $\zeta : \mathbb{R}^m \mapsto R$ be continuous. We call $\widetilde{\zeta} : \mathbb{R}^m \times \mathbb{R}_+ \mapsto R$ a smoothing function of ζ, if for all fixed $\mu > 0$, $\widetilde{\zeta}(\cdot, \mu)$ is continuously differentiable, and $\lim_{y \to \bar{y}, \mu \downarrow 0} \widetilde{\zeta}(y, \mu) = \zeta(\bar{y})$.

μ-smoothing function $\widetilde{\sigma}(y, \mu) : \mathbb{R}^m \times \mathbb{R}_+ \mapsto \mathbb{R}^m$ for $\sigma(y) = y_+ : \mathbb{R}^m \mapsto \mathbb{R}^m$

$$\widetilde{\sigma}_i(y, \mu) = \begin{cases} 0 & \text{if } y_i < 0 \\ \dfrac{y_i^2}{2\mu} & \text{if } 0 \le y_i \le \mu \\ y_i - \dfrac{\mu}{2} & \text{if } y_i > \mu \end{cases}$$



$\widetilde{\sigma}(y, \mu_1) > \widetilde{\sigma}(y, \mu_2)$ with $\mu_1 < \mu_2$.

A Smoothing Function for $\mathcal{O}(z)$

A smoothing function for $\mathcal{O}(z)$ with $\mu > 0$:

$$\widetilde{\mathcal{O}}(z,\mu) := \widetilde{\mathcal{H}}(z,\mu) + \mathcal{R}(z),$$

$\widetilde{\mathcal{H}}(z,\mu) := \widetilde{\mathcal{F}}(z,\mu) + \widetilde{\mathcal{P}}(z,\mu)$, where

$\widetilde{\mathcal{F}}(z,\mu) = \frac{1}{N}\sum_{n=1}^{N} \|(W^T v_n + b_2)_+\|_2^2 + \frac{1}{N}\|X\|_F^2 - \frac{2}{N}\sum_{n=1}^{N} x_n^T \widetilde{\sigma}(W^T v_n + b_2, \mu)$,

$\widetilde{\mathcal{P}}(z,\mu) = \beta \sum_{n=1}^{N} e^T (v_n - \widetilde{\sigma}(W x_n + b_1, \mu))$

are a smoothing function for $\mathcal{F}(z)$ and $\mathcal{P}(z)$, respectively.

- $\widetilde{\mathcal{O}}(z,\mu_1) < \widetilde{\mathcal{O}}(z,\mu_2)$ with $\mu_1 < \mu_2$.
- for $\mu > 0$, $z \in \mathcal{Z}$, $0 \le \mathcal{O}(z) \le \widetilde{\mathcal{O}}(z,\mu) \le \mathcal{O}(z) + (\|X\|_1 + N_1 N\beta)\,\mu$.

SPG

Step 1: Initialization: choose $z^{(0)} \in \mathcal{Z}$, $0 < \mu^{(0)} < 1$, $0 < \tau_1 < 1$, $\tau_2 > 0$, $\tau_3 \geq 1$, and $L^{(0)} \geq 1$. Set $k := 0$.

Step 2: Set $z^{(k+1)}$ be the unique minimizer of the strongly convex quadratic program

$$\min_{z \in \mathcal{Z}} \left\langle \nabla_z \widetilde{\mathcal{H}}(z^{(k)}, \mu^{(k)}), z - z^{(k)} \right\rangle + \mathcal{R}(z) + \frac{L^{(k)}}{2} \|z - z^{(k)}\|_2^2.$$

Step 3: Update the smoothing and proximal parameters $\mu^{(k+1)}$ and $L^{(k+1)}$ by

$$\begin{cases} (\mu^{(k+1)}, L^{(k+1)}) := (\mu^{(k)}, L^{(k)}), & \text{if } \widetilde{O}(z^{(k+1)}, \mu^{(k)}) - \widetilde{O}(z^{(k)}, \mu^{(k)}) < -\tau_2 \dfrac{\mu^{(k)}}{L^{(k)}}, \\ (\mu^{(k+1)}, L^{(k+1)}) := (\tau_1 \mu^{(k)}, \tau_3 L^{(k)}), & \text{otherwise.} \end{cases}$$

Step 4: Increment k by one, return to step 2.

5. Numerical Performances of SPG

Implementation Details

- $\lambda_2 = 0.1$, $\lambda_1 = 0.0001$, $\beta = \frac{1}{N}$.
- $\tau_1 = 0.5$, $\tau_2 = 0.001$, $\mu^{(0)} = 0.001$.
- $\tau_3 = 1.1$, $L^{(0)} = L_* := \max\{1, \sqrt{N_0 N_1/N}, \beta, N_0/30\}$ ($L^{(0)}$ should not be too small).
- max iteration: 4000.
- initialization: $W^{(0)} = \mathrm{randn}(N_1, N_0)/N$, $b^{(0)} = 0$ and $v_n^{(0)} = (W^{(0)} x_n)_+$ for all $n = 1, 2, \ldots, N$.
- stop criterion: $\mu^{(k)} \leq 10^{-7}$.

Implementation Details (Cont'd)

- data type 1: under Gaussian distribution with some noises: we generate the data matrix $X_{\text{all}} = (x_1, x_2, \ldots, x_{N+N_{\text{test}}})$ by setting

$$x_i \sim \mathcal{N}(\vartheta, \Sigma_0^{\mathrm{T}}\Sigma) + \epsilon_0 \mathcal{N}(0, 1)$$

  for all $i = 1, 2, \ldots, N + N_{\text{test}}$, where $\vartheta = 0.5 + \text{randn}(N_0, 1)$ and $\Sigma_0 = \text{randn}(N_0, 1)$. We then set all negative elements of X_{all} to be zero.
- data type 2: distributed in $[0, 1]$ with some noises: we generate the data matrix X_{all} by

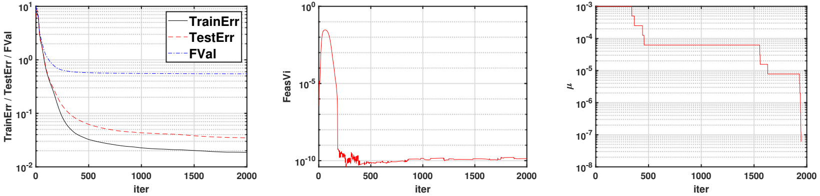
$$X_{\text{all}} = \text{rand}(N + N_{\text{test}}, N_0) + \epsilon_0 \text{randn}(N + N_{\text{test}}, N_0).$$

  We then set all negative elements of X_{all} to be zero.
- the first N and the last N_{test} columns of X_{all} are selected to be the training and test sets, respectively.
- noise: ϵ_0 (we set $\epsilon_0 = 0.05$).

Measurements

- FVal: the function value of (LRP), denoted by $\mathcal{O}(z)$.
- FeasVi: the average feasibility violation, denoted by $\frac{1}{NN_1}\sum_{n=1}^{N}\|v_n-(Wx_n+b_1)_+\|_1$.
- TrainErr: denoted by $\frac{1}{N}\sum_{n=1}^{N}\|(W^T(Wx_n+b_1)_+ + b_2)_+ - x_n\|_2^2$.
- TestErr: denoted by $\frac{1}{N_{\text{test}}}\sum_{n=N_{\text{test}}+1}^{N+N_{\text{test}}}\left\|(W^{\mathrm{T}}v_n+b_2)_+ - x_n\right\|_2^2$.
- Time: CPU time (s).

Algorithm performance of SPG

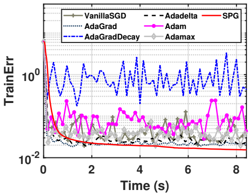


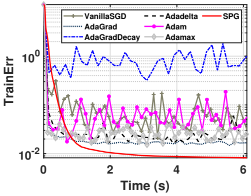
(a) TainErr/TestErr/FVal (b) FeasVi (c) μ

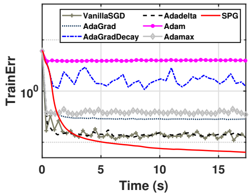
FIG 1: Algorithm performance of SPG

Here $N = 100$, $N_1 = 10$, $N_0 = 5$.

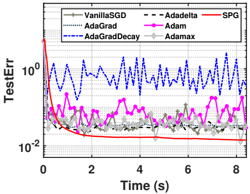
Comparisons on Random Data sets with data type 1

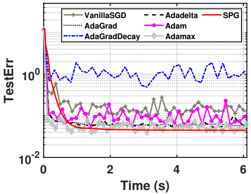


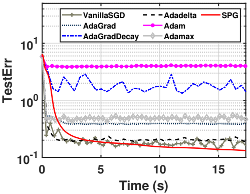




(a) $N = 75, N_1 = 10, N_0 = 5$ (b) $N = 100, N_1 = 10, N_0 = 5$ (c) $N = 150, N_1 = 20, N_0 = 10$

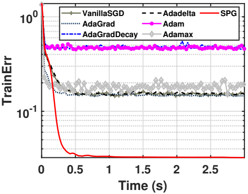




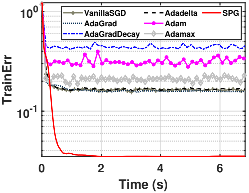


(d) $N = 75, N_1 = 10, N_0 = 5$ (e) $N = 100, N_1 = 10, N_0 = 5$ (f) $N = 150, N_1 = 20, N_0 = 10$

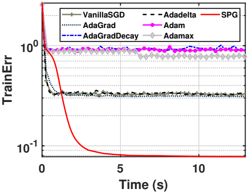
Comparisons on Random Data sets with data type 2



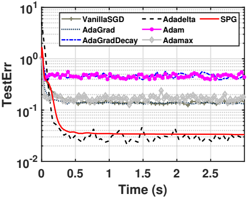
(a) $N = 75, N_1 = 10, N_0 = 5$



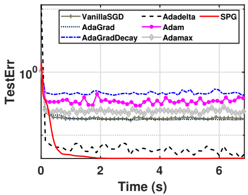
(b) $N = 100, N_1 = 10, N_0 = 5$



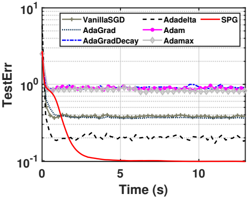
(c) $N = 150, N_1 = 20, N_0 = 10$



(d) $N = 75, N_1 = 10, N_0 = 5$



(e) $N = 100, N_1 = 10, N_0 = 5$



(f) $N = 150, N_1 = 20, N_0 = 10$

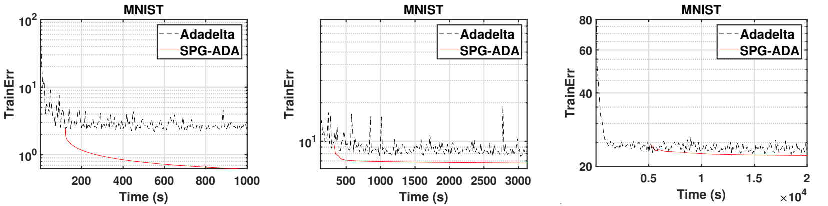
Comparisons on Random Data sets with $N = 1000$

We then consider to use Adadelta as a pre-process to accelerate SPG. More specifically, we first run Adadelta for 1000 epochs and then switch to SPG. We call the consequent hybrid algorithm SPG-ADA.

TAB 1: Comparisons between SPG-ADA and Adadelta with $N = 1000$.

		SPG-ADA				Adadelta		
N_0	N_1	TrainErr	TestErr	FeaErr	Time	TrainErr	TestErr	Time
5	20	3.297e-02	3.636e-02	1.234e-11	8.758	5.518e-02	5.847e-02	3.044
5	30	2.974e-02	3.103e-02	5.599e-12	10.592	5.470e-02	5.566e-02	3.623
5	40	2.960e-02	3.200e-02	7.238e-12	15.206	5.474e-02	5.632e-02	3.786
10	40	6.708e-02	7.727e-02	1.140e-11	19.184	1.257e-01	1.341e-01	5.590
10	60	6.867e-02	7.863e-02	5.138e-11	22.599	1.348e-01	1.436e-01	6.149
10	80	8.105e-02	9.057e-02	8.814e-11	25.701	1.364e-01	1.441e-01	7.169
20	80	1.824e-01	2.200e-01	3.020e-12	33.962	3.766e-01	4.265e-01	8.992
20	120	1.135e-01	2.611e-01	3.634e-12	38.191	4.051e-01	4.566e-01	12.275
20	160	1.946e-01	2.380e-01	2.181e-12	72.942	3.746e-01	4.240e-01	20.268

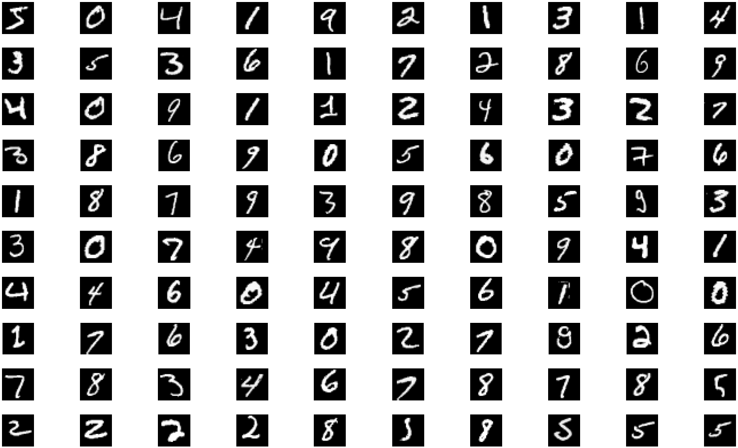
Results on Real datasets: Reconstruction on MNIST



(a) $N = 100, N_1 = 500$ (b) $N = 1000, N_1 = 1000$ (c) $N = 10000, N_1 = 2000$

FIG 2: Comparisons between SPG-ADA and Adadelta on MNIST.

Results on Real datasets: Reconstruction on MNIST



(a) SPG

Results on Real datasets: Reconstruction on MNIST

(a) Adam

Conclusions

- the solution set is nonempty and bounded.
- first present the exact penalty results regarding global solutions, local minimizers, Clarke stationary point and directional stationary point.
- present an effective algorithm: global convergence✓ to a Clarke stationary point✓, which is stronger than a critical point.
- SPG can be used to solve large-scale problems, and a better TrainErr/TestErr is obtained (compared with that obtained by SGD-based algorithms).
- the Clarke stationary point always performs better than the critical point in the sense of TrainErr/TestErr.
- Wei Liu, Xin Liu, Xiaojun Chen. Linearly-constrained nonsmooth optimization for training autoencoders. Preprint, arXiv:2103.16232, 2021.

Thanks for your listening!

Email: liuwei175@lsec.cc.ac.cn