A Continuous Model for Developing Fast Algorithms

Zhonglin xie

Peking University

December 8, 2022

How Mathematicians Model Fast Algorithms: PEP

$\mathcal{F}_{\mu,L}$: μ-strongly convex L-smooth functions ($\mu \geq 0$)

Performance Estimate Problem (PEP): given N and x_0

$$\min_{\{h_{i,j}\}} \max_{f \in \mathcal{F}_{\mu,L}} \quad \frac{\|\nabla f(x_N)\|}{\|\nabla f(x_0)\|}$$

$$\text{s.t.} \quad x_N \text{ obtained from}$$

$$x_{i+1} = x_i - \sum_{j=0}^{i} h_{i,j} \nabla f(x_j) \text{ and } x_0$$

Systematically generate the fast algorithms with worst-case guarantees. Often tractable for first order methods.

- Convex: Optimized Gradient Method (OGM)
- Strongly convex: Information-Theoretic Exact Method
- Composite, Operator Splitting, Primal Dual, ...
- Simple proofs for first-order methods

PEP Summary

- PROS

  Convex interpolation: from infinite to finite problems

  Mathematical-orientation: accelerate with guarantee

  Very general: can be used to analyze any interpolable function class

- CONS

  Ill-posed SDP: hard to scale when N is large

  Not automatic: first get a numerical solution, then manually approximate it with a symbolic formula

  Pessimistic and conservative: try to minimize the worst-case performance

How Computer Scientists Model Fast Algorithms: L2O

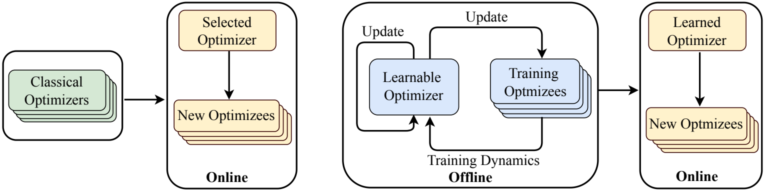
Learning to Optimize: given x_0 and N

$$\min_{\{\theta_i\}} \quad \mathbb{E}_f\left[f\left(x_N\right)\right]$$
$$\text{s.t.} \quad x_{i+1} = x_i - \mathrm{NN}(\{x_j, \nabla f(x_j)\}_{j=0}^{i}, \theta_i), \quad t = 1, \ldots, N-1$$

where $f \sim \mathcal{T}$, a probability measure defined in functional space

PROS: significant improvement; easy to implement

CONS: no theoretical guarantee; not explainable; finite iterate; sometimes needs the ground truth $x_\star$



(a) Classic Optimizer (b) Learning to Optimize

An Equavilent Form of PEP

Given x_0, minimizing $\|\nabla f(x_N)\|$ with a fixed N

$$\begin{aligned}\min_{\{h_{i,j}\}} \max_{f \in \mathcal{F}_{\mu,L}} \quad & \frac{\|\nabla f(x_N)\|}{\|\nabla f(x_0)\|} \\ \text{s.t.} \quad & x_N \text{ obtained from} \\ & x_{i+1} = x_i - \sum_{j=0}^{i} h_{i,j} \nabla f(x_j) \text{ and } x_0\end{aligned}$$

Given x_0, minimizing N with a fixed optimality $\|\nabla f(x_N)\| \leq \varepsilon$

$$\begin{aligned}\min_{\{h_{i,j}\}} \max_{f \in \mathcal{F}_{\mu,L}} \quad & N \\ \text{s.t.} \quad & x_N \text{ obtained from} \\ & x_{i+1} = x_i - \sum_{j=0}^{i} h_{i,j} \nabla f(x_j) \text{ and } x_0 \\ & N = \min\{n \colon \|\nabla f(x_n)\| \leq \varepsilon\}\end{aligned}$$

Optimisitic and Computation Tractable Reformulation

Consider a function $F(x;\theta)$ with variable x and parameter θ. Given a probability measure $\mathcal{T}$ of the parameter θ, we say $\mathcal{T}$ is the probability measure of functions generated by $f(\cdot) = F(\cdot;\theta), \theta \sim \mu$.

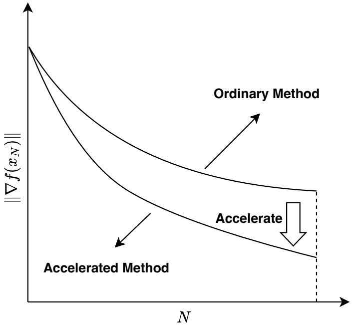
Given a task distribution $f \sim \mathcal{T}$, a tolerance ε and x_0

$$\begin{aligned} \min_{\{h_{i,j}\}} \quad & \mathbb{E}_f[N] \\ \text{s.t.} \quad & x_N \text{ obtained from} \\ & x_{i+1} = x_i - \sum_{j=0}^{i} h_{i,j} \nabla f(x_j) \text{ and } x_0 \\ & N = \min\{n \colon \|\nabla f(x_n)\| \le \varepsilon\} \end{aligned}$$

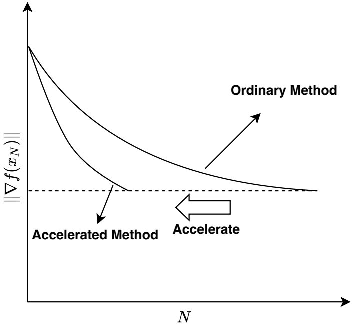
N is not differentiable with respect to $\{h_{i,j}\}$

We will solve this in a continuous time model!

A comparison of two approaches for acceleration



(a) Performance measure based



(b) Complexity based

Optimization Methods: Discrete and Continuous

Gradient Flow

$$\frac{\mathrm{d}x}{\mathrm{d}t}(t) = -\nabla f(x(t))$$

Euler applied to gradient flow with $t_k = t_0 + kh, x_k \approx x(t_k)$

$$\frac{x_{k+1} - x_k}{h} = -\nabla f(x_k) \Leftrightarrow x_{k+1} = x_k - h\nabla f(x_k)$$

Model Nesterov Accelerated Gradient using an ODE

$$\ddot{x}(t) + \frac{3}{t}\dot{x}(t) + \nabla f(x(t)) = 0 \Leftrightarrow \begin{cases} x_k = y_{k-1} - s\nabla f\left(y_{k-1}\right) \\ y_k = x_k + \dfrac{k-1}{k+2}\left(x_k - x_{k-1}\right) \end{cases}$$

Derivation of Su-Boyd-Candès ODE

$$\frac{x_{k+1}-x_k}{\sqrt{s}} = \left(1-\frac{3}{k+2}\right)\frac{x_k - x_{k-1}}{\sqrt{s}} - \sqrt{s}\nabla f\left(y_k\right).$$

Introduce the Ansatz $x_k \approx x(k\sqrt{s})$ for $t \geq 0$. Put $k = t/\sqrt{s}$.

Then as the step size s goes to zero, $x(t) \approx x_{t/\sqrt{s}} = x_k$

$$\left(x_{k+1}-x_k\right)/\sqrt{s} = \dot{x}(t) + \frac{1}{2}\ddot{x}(t)\sqrt{s} + o(\sqrt{s}),$$
$$\left(x_k - x_{k-1}\right)/\sqrt{s} = \dot{x}(t) - \frac{1}{2}\ddot{x}(t)\sqrt{s} + o(\sqrt{s})$$

and $\sqrt{s}\nabla f\left(y_k\right) = \sqrt{s}\nabla f(x(t)) + o(\sqrt{s})$. Omit $o(\sqrt{s})$ term.

$$\dot{x}(t) + \frac{1}{2}\ddot{x}(t)\sqrt{s} = (1-\frac{3\sqrt{s}}{t})(\dot{x}(t) - \frac{1}{2}\ddot{x}(t)\sqrt{s}) - \sqrt{s}\nabla f(x(t))$$

By comparing the coefficients of $\sqrt{s}$, we obtain

$$\ddot{x} + \frac{3}{t}\dot{x} + \nabla f(x) = 0$$

Convergence in Continuous Time

Define $\mathcal{E}(t) = t^2\left(f(x(t)) - f^\star\right) + 2\|x + t\dot{x}/2 - x^\star\|^2$

$$\dot{\mathcal{E}} = 2t\left(f(x) - f^\star\right) + t^2\langle \nabla f, \dot{x}\rangle + 4\left\langle x + \frac{t}{2}\dot{x} - x^\star, \frac{3}{2}\dot{x} + \frac{t}{2}\ddot{x}\right\rangle$$

Substituting $3\dot{x}/2 + t\ddot{x}/2$ with $-t\nabla f(x)/2$ gives

$$\begin{aligned}\dot{\mathcal{E}} &= 2t\left(f(x) - f^\star\right) + 4\left\langle x - x^\star, -t\nabla f(x)/2\right\rangle \\ &= 2t\left(f(x) - f^\star\right) - 2t\left\langle x - x^\star, \nabla f(x)\right\rangle \\ &\leq 0\end{aligned}$$

Lyapunov argument gives $\mathcal{O}(1/t^2)$ rate

An ODE with Unprecedented Level of Generality

$$\ddot{x}(t) + \frac{a}{t}\dot{x}(t) + \beta(t)\nabla^2 f(x(t))\dot{x}(t) + \gamma(t)\nabla f(x(t)) = 0$$

Denote $w(t) = \gamma(t) - \dot{\beta}(t) - \beta(t)/t$. Provided the conditions

$$\gamma(t) > \dot{\beta}(t) + \frac{\beta(t)}{t}, \quad t\dot{w}(t) \leqslant (a-3)w(t), \quad \text{for all } t \geqslant t_0,$$

the solution trajectory $x(t)$ of above ODE satisfies

$$f(x(t)) - f_\star = \mathcal{O}\left(\frac{1}{t^2 w(t)}\right) \text{ as } t \to +\infty$$

$$\int_{t_0}^{+\infty} t^2 \beta(t) w(t) \|\nabla f(x(t))\|^2 \, \mathrm{d}t < +\infty$$

This ODE can be written as a first order system

$$\begin{aligned} \dot{x}(t) &= v(t) - x(t) - \beta(t)\nabla f(x(t)), \\ \dot{v}(t) &= (1 - a/t)\dot{x}(t) + (\dot{\beta}(t) - \gamma(t))\nabla f(x(t)). \end{aligned}$$

The Effects of the Hessian-driven Damping Term

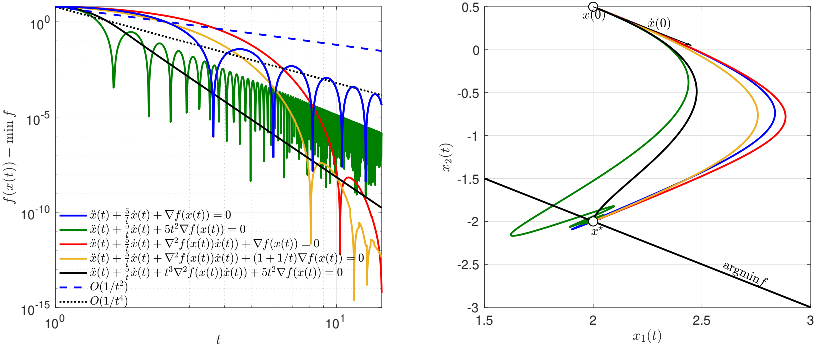


Figure: $f(x) = (x_1 + x_2)^2$ with different a, β, γ

The Hessian-driven damping term $\nabla^2 f(x(t))\dot{x}(t)$ is inspired from Newton's flow $\nabla^2 f(x(t))\dot{x}(t) + \nabla f(x(t)) = 0$

Reduce the oscillation; Accelerate converge; Stabilize discretization

From the Stopping Index N to the Stopping Time T

$$T = \inf\{t \mid \|\nabla f(x(t))\| \le \varepsilon, t \ge t_0\}.$$

We further write $T = T(f, a, \beta, \gamma)$ to emphasize the dependence on the variables f, a, β, γ and take $x_0, v_0, t_0, \varepsilon$ as exogenous variables.

T is the infimum of the set $\{t \colon \|\nabla f(x(t))\|^2 = \varepsilon^2\}$.

Suppose θ is one of the variables a, β and γ, the variation of both sides with respect to θ satisfies

$$2\nabla f(x(T))^\top \nabla^2 f(x(T)) \left(\dot{x}(T) \frac{\delta T}{\delta \theta} + \frac{\delta x_T}{\delta \theta} \right) = 0,$$

where x_T represents the value of x at fixed time T.

Stopping Time Continued

Stopping time is a standard concept in random process. We first introduce it to model fast algorithms, which does not need ground truth solutions.

- PROS

  Pretty natural and general

  Differentiable with respect to variables a, β and γ

- CONS

  Hard to generalize in the discrete time case

  Definition of the stopping time of a function value-based optimality condition involves $f_\star$

A Continuous Model for Fast Algorithms

Let $w(t) = \gamma(t) - \dot{\beta}(t) - \beta(t)/t$. We want to find a stable solution trajectory that converges fast on a task distribution:

$$
\begin{aligned}
\min_{a,\beta,\gamma} \quad & \mathbb{E}_{\mathcal{T}}[T(f, a, \beta, \gamma)], \\
\text{s.t.} \quad & T(f, a, \beta, \gamma) = \inf\{t \mid \|\nabla f(x(t))\| \leq \varepsilon, t \geq t_0\}, \text{ where} \\
& \ddot{x}(t) + \frac{a}{t}\dot{x}(t) + \beta(t)\nabla^2 f(x(t))\dot{x}(t) + \gamma(t)\nabla f(x(t)) = 0, \\
& \gamma(t) > \dot{\beta}(t) + \frac{\beta(t)}{t}, \quad t\dot{w}(t) \leqslant (a-3)w(t), \quad \forall t \geqslant t_0, \\
& (s\gamma(t) - \sqrt{s}\beta(t))\nabla^2 f(x(t)) \preceq \sqrt{s}a/tI, \forall t \geqslant t_0, \\
& (2\sqrt{s}\beta(t) - s\gamma(t))\nabla^2 f(x(t)) \preceq (4 - 2\sqrt{s}a/t)I, \forall t \geqslant t_0.
\end{aligned}
$$

The last two constraints come from the linear stability in discretization, where $\sqrt{s}$ denotes the step size of forward Euler scheme.

Arbitrary Fast Convergence?

Recall the convergence rate

$$f(x(t)) - f_\star = \mathcal{O}\left(\frac{1}{t^2 w(t)}\right) \text{ as } t \to +\infty$$

For any $p \in \mathbb{N}$, simply setting $\beta(t) \equiv 0$ and $a = p+1, \gamma(t) = t^{p-2}$

$$\ddot{x}(t) + \frac{p+1}{t}\dot{x}(t) + t^{p-2}\nabla f\left(x(t)\right) = 0$$
$$\Rightarrow w(t) = \gamma(t) - \dot{\beta}(t) - \beta(t)/t = t^{p-2}$$

The convergence rate is $f\left(x(t)\right) - f_\star \leqslant \mathcal{O}\left(1/t^p\right)$.

We get arbitrary fast convergence rate with convex differentiable functions in continuous time case.

What is wrong here?

Direct Runge-Kutta Discretization is Unstable

Consider the logistic regression problem

$$\min_{x \in \mathbb{R}^n} f(x) = \frac{1}{N} \sum_{i=1}^{N} \log(1 + \exp(-b_i \langle a_i, w \rangle)),$$

where the data pairs $\{a_, b_i\} \in \mathbb{R}^n \times \{0, 1\}, i \in [N]$

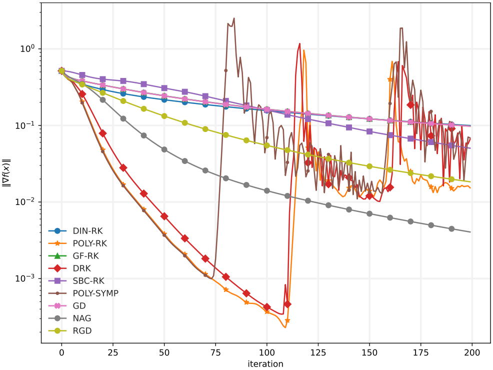


Figure: Directly applying 4-th Runge-Kutta with $p = 5$ diverges.

How to discretize: Methodologies

- It is hard to obtain stable discretization!
- "Empirically, we find that the algorithm is unstable. Even for the simple case in which f is a quadratic function in two dimensions, ... eventually the oscillation increases and the iterates shoot off to infinity"—[Wibisono et al., 2016]
- Geometric numerical integrator: the existence of $t^{p-2}\nabla f(x(t))$ makes the stepsize decrease to 0
- Tradeoff between higher order convergence and large step stability
- Momentum restarting: hard to analysis; not stable enough; a short-term solution
- Why not selecting an ODE to fit an integrator?
- In natural science, ODEs can not be changed
- In optimization, ODEs can vary in a large range

Optimization is Not Numerical Solution

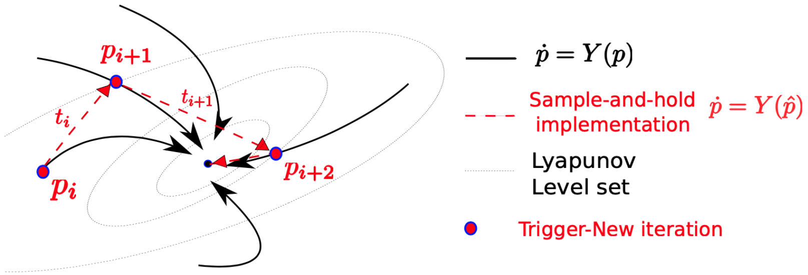


Figure: Discretization in Optimization

In optimization, we do not need an exact numerical solution

Provided the discretization has linear stability, we will approach the minimizer finally!

Stability Implies Convergence

Linear Stability is a leading indicator!

Linear stability excesses 1 before the function value diverges

A new regularization condition: Linear Stability

This is a long-term solution

Stability Analysis for AVD-DIN System

Consider the forward Euler method with step length $\sqrt{s}$

$$
\begin{aligned}
&\frac{x(t+2\sqrt{s}) - 2x(t+\sqrt{s}) + x(t)}{s} \\
&+ \Big(\alpha/t + \beta(t)\nabla^2 f(x(t))\Big) \cdot \frac{x(t+\sqrt{s}) - x(t)}{\sqrt{s}} \\
&+ \gamma(t)\nabla f(x(t)) = 0.
\end{aligned}
$$

The characteristic polynomial is

$$|\lambda^2 \boldsymbol{I} - b(t,\sqrt{s})\lambda \boldsymbol{I} + (1-\sqrt{s}\alpha/t)\boldsymbol{I} + (s\gamma(t) - \sqrt{s}\beta(t))\nabla^2 f(x(t))| = 0$$

where $b(t,\sqrt{s}) = 2 - \sqrt{s}(\alpha/t + \beta(t)\nabla^2 f(x(t)))$.

The necessary and sufficient condition

The necessary and sufficient condition for the roots (may be complex) of $r^2 + \mu r + \nu = 0, \mu, \nu \in \mathbb{R}$ lie in the unit cycle is

$$\nu \leq 1, \quad \nu \geq \mu - 1, \quad \nu \geq -\mu - 1.$$

We get a necessary and sufficient condition for our discretization to be stable:

$$(s\gamma(t) - \sqrt{s}\beta(t))\nabla^2 f(x(t)) \preceq \sqrt{s}\alpha/t\boldsymbol{I},$$
$$(2\sqrt{s}\beta(t) - s\gamma(t))\nabla^2 f(x(t)) \preceq (4 - 2\sqrt{s}\alpha/t)\boldsymbol{I},$$
$$s\gamma(t)\nabla^2 f(x(t)) \succeq 0.$$

Training a Polynomial Surrogate Model

Given a degree $k \in \mathbb{N}$, and a step size $s > 0$, we choose

$$a = k + 3, \quad \beta(t) = \sum_{i=0}^{k} p_i t^i, \quad \gamma(t) = \beta(t)/\sqrt{s},$$

with $p_i \geq 0$. When t_0 is sufficiently large, this choice automatically satisfies

$$\gamma(t) > \dot{\beta}(t) + \frac{\beta(t)}{t}, \quad t\dot{w}(t) \leqslant (a-3)w(t), \quad \forall t \geqslant t_0;$$
$$(s\gamma(t) - \sqrt{s}\beta(t))\nabla^2 f(x(t)) \preceq \sqrt{s}a/tI, \quad \forall t \geqslant t_0.$$

Dropping above constraints gives:

$$\begin{aligned} \min_p \quad & \mathbb{E}_{\mathcal{T}}[T(f,p)], \\ \text{s.t.} \quad & T(f,p) = \inf\{t \mid \|\nabla f(x(t))\| \leq \varepsilon, t \geq t_0\}, \text{ where} \\ & \ddot{x}(t) + \frac{a}{t}\dot{x}(t) + \beta(t)\nabla^2 f(x(t))\dot{x}(t) + \gamma(t)\nabla f(x(t)) = 0, \\ & \sqrt{s}\beta(t)\lambda_{\max}(\nabla^2 f(x(t))) \leq 4 - 2\sqrt{s}a/t, \forall t \geqslant t_0. \end{aligned}$$

Training Algorithm

REQUIRE: Training set $\mathcal{D}$, which sampled from the probability measure $\mathcal{T}$ of f. Degree k of the polynomials. Initial value x_0. Step size $\sqrt{s}$ of forward Euler scheme. Events $\{\varepsilon_0, \varepsilon_1, \ldots, \varepsilon_M\}$. Initial value of the coefficient p^0. Training epoch N_{epoch}.

ENSURE: A differential equation that adapts to the probability measure $\mathcal{T}$ of f, converges fast and possesses stability under forward Euler discretization with step size $\sqrt{s}$.

Algorithm 1 Stochastic Projected Gradient Descent for Polynomial Surrogate Problem

1: **for** $n_{\text{epoch}} = 1, 2, \ldots, N_{\text{epoch}}$ **do**

2:     **for** $n_{\text{sample}} = 1, 2, \ldots, |\mathcal{D}|$ **do**

3:         Randomly draw one sample f from $\mathcal{D}$.

4:         Simulate the solution trajectory $x(t)$ of

$$\ddot{x}(t) + \frac{a}{t}\dot{x}(t) + \beta(t)\nabla^2 f(x(t))\dot{x}(t) + \gamma(t)\nabla f(x(t)) = 0.$$

5:         Record the event times $T_m = \inf\{t \mid \|\nabla f(x(t))\| \leq \varepsilon_m, t \geq t_0\}$ for $m = 0, 1, \ldots, M$

6:         Compute the derivative of T_M with respect to p and Perform a step of the gradient descent: $p \leftarrow p - \frac{\partial T_M}{\partial p}$.

7:         Denote the polyhedra

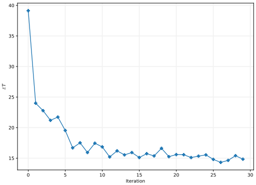
$$\{p \mid \sqrt{s}\beta(T_m)\lambda_{\max}(\nabla^2 f(x(T_m))) \leq 4 - 2\sqrt{s}a/T_m, \forall m\}$$

as $\mathcal{P}$. Project p with respect to it: $p \leftarrow \mathrm{Proj}_{\mathcal{P}}(p)$.

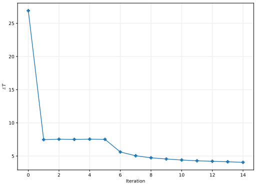
8:     **end for**

9: **end for**

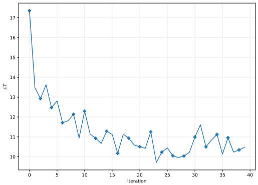
Training Result in Different Datasets



(a) `a5a`



(b) `mushrooms`



(c) `phishing`

Figure: Comparison of the training process in different datasets

Discrete time testing

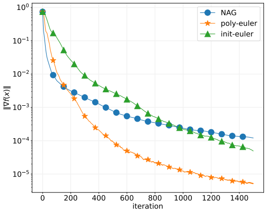
$$
\begin{aligned}
\frac{x_{k+1}-x_k}{\sqrt{s}} &= v_k - x_k - \beta_k \nabla f(x_k), \\
\frac{v_{k+1}-v_k}{\sqrt{s}} &= (1-\alpha/t_{k+1})(v_k - x_{k+1} - \beta_{k+1}\nabla f(x_{k+1})) \\
&\qquad + (\dot{\beta}_{k+1} - \gamma_{k+1})\nabla f(x_{k+1}).
\end{aligned}
$$

PROS:

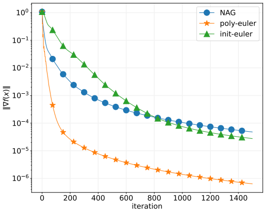
can be extended to infinite iterates;

fully explainable; has convergence guarantee;

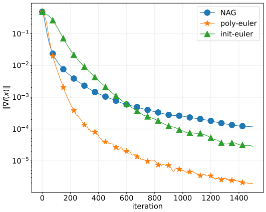
Testing Results



(a) `a5a`



(b) `mushrooms`



(c) `phishing`

Figure: Comparison of the forward Euler discretization applied to ODE trained in different datasets

Our Motivation&Methodology

A learning to optimize framework with theoretical guarantee

First give a condition that guarantees convergence

Then search parameters under this guarantee

Learning and adaptivity are equivalent

Illustrate this using gradient descent and adaptive methods

Examples

- Nonsmooth: [Banert et al., 2020]
- Inexact gradient: [Banert et al., 2021]

Future Directions

- Precondition (dimension dependent)
- Investigate discrete scheme directly (We solve the continuous time model numerically)
- New adaptive methods (In this work, the ODE remembers the local curvature first, then using this information to discretize. Another way is estimates these quantities adaptively.)
- Apply this paradigm to other problems (Composite, Monotone inclusion, ADMM, Primal-dual, ...)
- Closed-loop control? (Continuous adaptivity)
- Direct solve the equivalent form of PEP!?

Epilogue

A general viewpoint of optimization and learning?

Parameterization gives a general way for producing optimization methods

Best papers preferring analytical solutions, e.g. Analytic LISTA, Analytic DPM (PnP, White-Box Net)

Learning researchers are willing to fire themselves

The most important thing is the meaning of each parameter

References I

Banert, S., Ringh, A., Adler, J., Karlsson, J., and Öktem, O. (2020).

Data-Driven Nonsmooth Optimization.

SIAM Journal on Optimization, 30(1):102–131.

Banert, S., Rudzusika, J., Öktem, O., and Adler, J. (2021).

Accelerated Forward-Backward Optimization using Deep Learning.

arXiv:2105.05210 [math].

References II

Wibisono, A., Wilson, A. C., and Jordan, M. I. (2016).

A variational perspective on accelerated methods in optimization.

Proceedings of the National Academy of Sciences, 113(47):E7351–E7358.