Stability of neural ordinary differential equations

Arturo De Marinis[1], Nicola Guglielmi[1], Anton Savostianov[1], Stefano Sicilia[1], Francesco Tudisco[1,2]

[1]Gran Sasso Science Institute (GSSI), L'Aquila, Italy

[2]University of Edinburgh, Edinburgh, United Kingdom

25 April 2024

Introduction

- The simplest neural ODE is

$$\dot{x}(t) = \sigma\left(Ax(t) + b\right), \qquad t \in [0, T],$$

where $A \in \mathbb{R}^{n,n}$, $b \in \mathbb{R}^n$ and $\sigma : \mathbb{R} \to \mathbb{R}$ is a smooth activation function that acts entry-wise such that $\sigma'(\mathbb{R}) \subset [m, 1]$, $m > 0$.

Introduction

- The simplest neural ODE is

$$\dot{x}(t) = \sigma\left(Ax(t) + b\right), \qquad t \in [0, T],$$

where $A \in \mathbb{R}^{n,n}$, $b \in \mathbb{R}^n$ and $\sigma : \mathbb{R} \to \mathbb{R}$ is a smooth activation function that acts entry-wise such that $\sigma'(\mathbb{R}) \subset [m, 1]$, $m > 0$.

- Integrating with the Euler method with constant step size h yields

$$x_{k+1} = x_k + h\sigma\left(Ax_k + b\right), \quad k = 0, 1, \ldots$$

Introduction

- The simplest neural ODE is

$$\dot{x}(t) = \sigma\left(Ax(t) + b\right), \qquad t \in [0, T],$$

where $A \in \mathbb{R}^{n,n}$, $b \in \mathbb{R}^n$ and $\sigma : \mathbb{R} \to \mathbb{R}$ is a smooth activation function that acts entry-wise such that $\sigma'(\mathbb{R}) \subset [m, 1]$, $m > 0$.

- Integrating with the Euler method with constant step size h yields

$$x_{k+1} = x_k + h\sigma\left(Ax_k + b\right), \quad k = 0, 1, \dots$$

- Using suitable numerical methods, we can build a neural network that preserves some properties of the differential equation, e.g. stability.

Table of contents

1. Stability of neural ODEs
2. Methodology
3. Numerical experiments

Table of contents

1. Stability of neural ODEs
2. Methodology
3. Numerical experiments

Preliminary definitions

- The symmetric part of a matrix A is

$$\mathrm{Sym}(A) = \frac{A + A^\top}{2}.$$

Preliminary definitions

- The symmetric part of a matrix A is

$$\mathrm{Sym}(A) = \frac{A + A^\top}{2}.$$

- The logarithmic 2-norm of a matrix A is

$$\mu_2(A) = \lambda_{\mathsf{max}}\left(\mathrm{Sym}(A)\right).$$

Preliminary definitions

- The symmetric part of a matrix A is

$$\mathrm{Sym}(A) = \frac{A + A^\top}{2}.$$

- The logarithmic 2-norm of a matrix A is

$$\mu_2(A) = \lambda_{\mathsf{max}} \left(\mathrm{Sym}(A)\right).$$

- The positive part of a real-valued function f is defined as

$$(f(x))_+ = \max(0, f(x)).$$

Preliminary definitions

- The symmetric part of a matrix A is

$$\mathrm{Sym}(A) = \frac{A + A^\top}{2}.$$

- The logarithmic 2-norm of a matrix A is

$$\mu_2(A) = \lambda_{\max}\left(\mathrm{Sym}(A)\right).$$

- The positive part of a real-valued function f is defined as

$$(f(x))_+ = \max(0, f(x)).$$

- Given $0 < m \leq 1$, define

$$\Omega_m = \{D \in \mathbb{D}^{n,n} \ : \ m \leq D_{ii} \leq 1 \quad \forall i = 1, \dots, n\}.$$

Stability (1/2)

- We wish, for any two solutions $x_1(t)$ and $x_2(t)$, that the bound

$$\|x_1(t) - x_2(t)\|_2 \leq C\|x_1(0) - x_2(0)\|_2, \quad \forall t \in [0, T],$$

is satisfied for a moderately sized constant $C > 0$.

Stability (1/2)

- We wish, for any two solutions $x_1(t)$ and $x_2(t)$, that the bound

$$\|x_1(t) - x_2(t)\|_2 \leq C\|x_1(0) - x_2(0)\|_2, \quad \forall t \in [0, T],$$

is satisfied for a moderately sized constant $C > 0$.

- Let $f(t, x) = \sigma(Ax + b)$. The bound above is satisfied if there exists $\mu \in \mathbb{R}$ such that

$$\langle f(t, x) - f(t, y), x - y \rangle_2 \leq \mu\|x - y\|_2^2$$

for all $x, y \in \mathbb{R}^n$ and $t \in [0, T]$. Then $C = \mathrm{e}^{\mu T}$.

Stability (2/2)

- The neural ODE satisfies the one-sided Lipschitz condition with

$$\mu = \max_{D \in \Omega_m} \mu_2(DA),$$

(see Guglielmi et al., 2024).

Stability (2/2)

- The neural ODE satisfies the one-sided Lipschitz condition with

$$\mu = \max_{D \in \Omega_m} \mu_2(DA),$$

(see Guglielmi et al., 2024).

- Then

$$\|x_1(t) - x_2(t)\|_2 \leq e^{\mu t} \|x_1(0) - x_2(0)\|_2, \quad \forall t \in [0, T],$$

is satisfied, therefore the neural ODE (2)

1. might be slightly unstable if $\mu \leq c$, with $c > 0$ small;
2. is non-expansive if $\mu \leq 0$;
3. is contractive if $\mu < 0$.

Stability (2/2)

- The neural ODE satisfies the one-sided Lipschitz condition with

$$\mu = \max_{D \in \Omega_m} \mu_2 \left(DA \right),$$

  (see Guglielmi et al., 2024).

- Then

$$\|x_1(t) - x_2(t)\|_2 \leq \mathrm{e}^{\mu t} \|x_1(0) - x_2(0)\|_2, \quad \forall t \in [0, T],$$

  is satisfied, therefore the neural ODE (2)
  1. might be slightly unstable if $\mu \leq c$, with $c > 0$ small;
  2. is non-expansive if $\mu \leq 0$;
  3. is contractive if $\mu < 0$.

- We are interested in cases 2. and 3.

Table of contents

1 Stability of neural ODEs

2 Methodology

3 Numerical experiments

Optimal stabilization

Our goal is to compute a matrix $B \in \mathbb{R}^{n,n}$ such that

$$\max_{D \in \Omega_m} \mu_2(DB) = \delta \leq 0 \quad \text{and} \quad B = \operatorname*{argmin}_{M \in \mathbb{R}^{n,n}} \|M - A\|_F.$$

Optimal stabilization

Our goal is to compute a matrix $B \in \mathbb{R}^{n,n}$ such that

$$\max_{D \in \Omega_m} \mu_2(DB) = \delta \leq 0 \quad \text{and} \quad B = \operatorname*{argmin}_{M \in \mathbb{R}^{n,n}} \|M - A\|_F.$$

We write $B = A + \varepsilon E$, where $E \in \mathbb{R}^{n,n}$, $\|E\|_F = 1$, and $\varepsilon > 0$.

Optimal stabilization

Our goal is to compute a matrix $B \in \mathbb{R}^{n,n}$ such that

$$\max_{D \in \Omega_m} \mu_2(DB) = \delta \leq 0 \quad \text{and} \quad B = \operatorname*{argmin}_{M \in \mathbb{R}^{n,n}} \|M - A\|_F.$$

We write $B = A + \varepsilon E$, where $E \in \mathbb{R}^{n,n}$, $\|E\|_F = 1$, and $\varepsilon > 0$.

Bilevel optimization

Optimal stabilization

Our goal is to compute a matrix $B \in \mathbb{R}^{n,n}$ such that

$$\max_{D \in \Omega_m} \mu_2(DB) = \delta \leq 0 \quad \text{and} \quad B = \operatorname*{argmin}_{M \in \mathbb{R}^{n,n}} \|M - A\|_F.$$

We write $B = A + \varepsilon E$, where $E \in \mathbb{R}^{n,n}$, $\|E\|_F = 1$, and $\varepsilon > 0$.

Bilevel optimization

- **Inner level.** Fixed $\varepsilon > 0$, we aim to minimize

$$F_\varepsilon(E) = \frac{1}{2} \sum_{i=1}^{n} (\lambda_i(\mathrm{Sym}(D_\star(A + \varepsilon E))) - \delta)_+^2,$$

  over $E \in \mathbb{R}^{n,n}$, $\|E\|_F = 1$, with $D_\star = \operatorname{argmax}_{D \in \Omega_m} \mu_2(D(A + \varepsilon E))$ and $\delta \in \mathbb{R}$. We denote a minimum as $E[\varepsilon]$.

Optimal stabilization

Our goal is to compute a matrix $B \in \mathbb{R}^{n,n}$ such that

$$\max_{D \in \Omega_m} \mu_2(DB) = \delta \leq 0 \quad \text{and} \quad B = \operatorname*{argmin}_{M \in \mathbb{R}^{n,n}} \|M - A\|_F.$$

We write $B = A + \varepsilon E$, where $E \in \mathbb{R}^{n,n}$, $\|E\|_F = 1$, and $\varepsilon > 0$.

Bilevel optimization

- **Inner level.** Fixed $\varepsilon > 0$, we aim to minimize

$$F_\varepsilon(E) = \frac{1}{2} \sum_{i=1}^{n} (\lambda_i(\mathrm{Sym}(D_\star(A + \varepsilon E))) - \delta)_+^2,$$

  over $E \in \mathbb{R}^{n,n}$, $\|E\|_F = 1$, with $D_\star = \operatorname{argmax}_{D \in \Omega_m} \mu_2(D(A + \varepsilon E))$ and $\delta \in \mathbb{R}$. We denote a minimum as $E[\varepsilon]$.
- **Outer level.** We look for the smallest zero $\varepsilon^\star$ of

$$f(\varepsilon) = F_\varepsilon(E[\varepsilon]).$$

Inner level

- We consider E a smooth matrix valued function of t, $E = E(t)$, and we use the following lemma to compute $\frac{d}{dt} F_\varepsilon(E(t))$.

Inner level

- We consider E a smooth matrix valued function of t, $E = E(t)$, and we use the following lemma to compute $\frac{d}{dt} F_\varepsilon(E(t))$.

Lemma

Consider a symmetric continuously differentiable $C(t) : \mathbb{R} \to \mathbb{R}^{n,n}$. Let $\lambda(t)$ be a simple eigenvalue of $C(t)$ for all t and let $x(t)$ with $\|x(t)\|_2 = 1$ be the associated eigenvector. Then $\lambda(t)$ is differentiable with

$$\dot{\lambda}(t) = x(t)^\top \dot{C}(t) x(t) = \langle x(t)x(t)^\top, \dot{C}(t) \rangle_F.$$

Inner level

- We consider E a smooth matrix valued function of t, $E = E(t)$, and we use the following lemma to compute $\frac{d}{dt} F_\varepsilon(E(t))$.

Lemma

Consider a symmetric continuously differentiable $C(t) : \mathbb{R} \to \mathbb{R}^{n,n}$. Let $\lambda(t)$ be a simple eigenvalue of $C(t)$ for all t and let $x(t)$ with $\|x(t)\|_2 = 1$ be the associated eigenvector. Then $\lambda(t)$ is differentiable with

$$\dot{\lambda}(t) = x(t)^\top \dot{C}(t) x(t) = \langle x(t)x(t)^\top, \dot{C}(t) \rangle_F.$$

- Setting $C(t) = A + \varepsilon E(t)$, we obtain the norm-preserving gradient system for the functional $F_\varepsilon(E(t))$:

$$\dot{E} = -G(E) + \langle G(E), E \rangle_F E,$$

with $G(E) = \sum_{i=1}^n \gamma_i z_i x_i^\top$, x_i eigenvector to $\lambda_i(\mathrm{Sym}(D_\star(A + \varepsilon E)))$, $z_i = D_\star x_i$ and $\gamma_i = (\lambda_i(\mathrm{Sym}(D_\star(A + \varepsilon E))) - \delta)_+$.

Outer level

- We seek for the zero of $f(\varepsilon) = F_\varepsilon(E[\varepsilon])$.

Outer level

- We seek for the zero of $f(\varepsilon) = F_\varepsilon(E[\varepsilon])$.
- Let $\lambda_i[\varepsilon]$ and $x_i[\varepsilon]$ be the eigenvalues and the eigenvectors of $\mathrm{Sym}(D_\star(A + \varepsilon E[\varepsilon]))$.

Outer level

- We seek for the zero of $f(\varepsilon) = F_\varepsilon(E[\varepsilon])$.
- Let $\lambda_i[\varepsilon]$ and $x_i[\varepsilon]$ be the eigenvalues and the eigenvectors of $\mathrm{Sym}(D_\star(A + \varepsilon E[\varepsilon]))$.

Assumption

For ε close to $\varepsilon^\star$ and $\varepsilon < \varepsilon^\star$, we assume that the eigenvalues $\lambda_i[\varepsilon]$ of $\mathrm{Sym}(D_\star(A + \varepsilon E[\varepsilon]))$ are simple eigenvalues. Moreover, $E[\varepsilon]$, $\lambda_i[\varepsilon]$ and $x_i[\varepsilon]$ are assumed to be smooth functions of ε.

Outer level

- We seek for the zero of $f(\varepsilon) = F_\varepsilon(E[\varepsilon])$.
- Let $\lambda_i[\varepsilon]$ and $x_i[\varepsilon]$ be the eigenvalues and the eigenvectors of $\mathrm{Sym}(D_\star(A + \varepsilon E[\varepsilon]))$.

Assumption

For ε close to $\varepsilon^\star$ and $\varepsilon < \varepsilon^\star$, we assume that the eigenvalues $\lambda_i[\varepsilon]$ of $\mathrm{Sym}(D_\star(A + \varepsilon E[\varepsilon]))$ are simple eigenvalues. Moreover, $E[\varepsilon]$, $\lambda_i[\varepsilon]$ and $x_i[\varepsilon]$ are assumed to be smooth functions of ε.

- Newton method

$$\varepsilon_{k+1} = \varepsilon_k - \frac{f(\varepsilon_k)}{f'(\varepsilon_k)}, \quad k = 0, 1, \ldots$$

Outer level

- We seek for the zero of $f(\varepsilon) = F_\varepsilon(E[\varepsilon])$.
- Let $\lambda_i[\varepsilon]$ and $x_i[\varepsilon]$ be the eigenvalues and the eigenvectors of $\mathrm{Sym}(D_\star(A + \varepsilon E[\varepsilon]))$.

Assumption

For ε close to $\varepsilon^\star$ and $\varepsilon < \varepsilon^\star$, we assume that the eigenvalues $\lambda_i[\varepsilon]$ of $\mathrm{Sym}(D_\star(A + \varepsilon E[\varepsilon]))$ are simple eigenvalues. Moreover, $E[\varepsilon]$, $\lambda_i[\varepsilon]$ and $x_i[\varepsilon]$ are assumed to be smooth functions of ε.

- Newton method

$$\varepsilon_{k+1} = \varepsilon_k - \frac{f(\varepsilon_k)}{f'(\varepsilon_k)}, \quad k = 0, 1, \dots$$

- An inexpensive formula: under the given Assumption, it holds

$$f'(\varepsilon) = -\|G[\varepsilon]\|_F.$$

Summary

Recall that

$$B = A + \varepsilon E,$$

with $E \in \mathbb{R}^{n,n}$, $\|E\|_F = 1$, $\varepsilon > 0$, and $f(\varepsilon) = F_\varepsilon(E[\varepsilon])$.

Summary

Recall that

$$B = A + \varepsilon E,$$

with $E \in \mathbb{R}^{n,n}$, $\|E\|_F = 1$, $\varepsilon > 0$, and $f(\varepsilon) = F_\varepsilon(E[\varepsilon])$.

1. Fix $\varepsilon_0 > 0$ and $k = 0$.

Summary

Recall that

$$B = A + \varepsilon E,$$

with $E \in \mathbb{R}^{n,n}$, $\|E\|_F = 1$, $\varepsilon > 0$, and $f(\varepsilon) = F_\varepsilon(E[\varepsilon])$.

1. Fix $\varepsilon_0 > 0$ and $k = 0$.
2. Integrate the constrained gradient system for E at level ε_0 to get $E[\varepsilon_0]$.

Summary

Recall that

$$B = A + \varepsilon E,$$

with $E \in \mathbb{R}^{n,n}$, $\|E\|_F = 1$, $\varepsilon > 0$, and $f(\varepsilon) = F_\varepsilon(E[\varepsilon])$.

1. Fix $\varepsilon_0 > 0$ and $k = 0$.
2. Integrate the constrained gradient system for E at level ε_0 to get $E[\varepsilon_0]$.
3. While $f(\varepsilon_k) > 0$:

Summary

Recall that

$$B = A + \varepsilon E,$$

with $E \in \mathbb{R}^{n,n}$, $\|E\|_F = 1$, $\varepsilon > 0$, and $f(\varepsilon) = F_\varepsilon(E[\varepsilon])$.

1. Fix $\varepsilon_0 > 0$ and $k = 0$.
2. Integrate the constrained gradient system for E at level ε_0 to get $E[\varepsilon_0]$.
3. While $f(\varepsilon_k) > 0$:

   3.1. update

$$\varepsilon_{k+1} = \varepsilon_k - \frac{f(\varepsilon_k)}{f'(\varepsilon_k)};$$

Summary

Recall that

$$B = A + \varepsilon E,$$

with $E \in \mathbb{R}^{n,n}$, $\|E\|_F = 1$, $\varepsilon > 0$, and $f(\varepsilon) = F_\varepsilon(E[\varepsilon])$.

1. Fix $\varepsilon_0 > 0$ and $k = 0$.
2. Integrate the constrained gradient system for E at level ε_0 to get $E[\varepsilon_0]$.
3. While $f(\varepsilon_k) > 0$:

   3.1. update

   $$\varepsilon_{k+1} = \varepsilon_k - \frac{f(\varepsilon_k)}{f'(\varepsilon_k)};$$

   3.2. integrate the constrained gradient system for E at level ε_{k+1} to get $E[\varepsilon_{k+1}]$;

Summary

Recall that

$$B = A + \varepsilon E,$$

with $E \in \mathbb{R}^{n,n}$, $\|E\|_F = 1$, $\varepsilon > 0$, and $f(\varepsilon) = F_\varepsilon(E[\varepsilon])$.

1. Fix $\varepsilon_0 > 0$ and $k = 0$.
2. Integrate the constrained gradient system for E at level ε_0 to get $E[\varepsilon_0]$.
3. While $f(\varepsilon_k) > 0$:
   3.1. update
   $$\varepsilon_{k+1} = \varepsilon_k - \frac{f(\varepsilon_k)}{f'(\varepsilon_k)};$$
   3.2. integrate the constrained gradient system for E at level ε_{k+1} to get $E[\varepsilon_{k+1}]$;
   3.3. $k = k + 1$.

Table of contents

1 Stability of neural ODEs

2 Methodology

3 Numerical experiments

What do we do?

- Training the neural ODE means updating A and b until a scalar function $\mathcal{L}(A, b)$ is minimized. Starting from an initial guess of the parameters A_0 and b_0,

$$A_{k+\frac{1}{2}} = A_k - h\nabla_A \mathcal{L}(A_k, b_k),$$
$$b_{k+1} = b_k - h\nabla_b \mathcal{L}(A_k, b_k),$$

where h is a sufficiently small step size.

What do we do?

- Training the neural ODE means updating A and b until a scalar function $\mathcal{L}(A, b)$ is minimized. Starting from an initial guess of the parameters A_0 and b_0,

$$A_{k+\frac{1}{2}} = A_k - h\nabla_A \mathcal{L}(A_k, b_k),$$
$$b_{k+1} = b_k - h\nabla_b \mathcal{L}(A_k, b_k),$$

  where h is a sufficiently small step size.

- We set

$$A_{k+1} = A_{k+\frac{1}{2}} + \varepsilon^\star E[\varepsilon^\star],$$

  with $\varepsilon^\star$ and $E[\varepsilon^\star]$ computed according to the above-mentioned numerical procedure to get

$$\mu_2(D_\star A_{k+1}) = \delta \leq 0.$$

MNIST dataset

- Full perturbation E

ε	0	0.01	0.02	0.03	0.04	0.05	0.06
Reference	97.29%	94.49%	89.07%	77.73%	61.95%	46.61%	34.14%
$\delta = 0$	93.59%	89.21%	82.09%	69.84%	52.19%	35.15%	22.33%
$\delta = -0.1$	96.97%	94.39%	90.19%	83.48%	72.83%	58.71%	42.48%
$\delta = -0.2$	97.25%	95.48%	91.98%	86.35%	78.44%	66.26%	52.29%
$\delta = -0.3$	96.2%	93.74%	89.5%	83.57%	74.35%	62.49%	49.61%

- Diagonal perturbation E

ε	0	0.01	0.02	0.03	0.04	0.05	0.06
Reference	97.29%	94.49%	89.07%	77.73%	61.95%	46.61%	34.14%
$\delta = 0$	97.25%	95.07%	91.63%	85.87%	75.31%	60.81%	46.21%
$\delta = -0.1$	97.36%	95.36%	92.23%	87.48%	78.78%	66.21%	51.81%
$\delta = -0.2$	96.95%	94.63%	91.54%	85.59%	77.17%	64.65%	50.15%
$\delta = -0.3$	95.9%	93.27%	89.48%	83.04%	73.1%	60.31%	46.49%

FashionMNIST dataset

- Full perturbation E

ε	0	0.01	0.02	0.03	0.04	0.05	0.06
Reference	88.07%	75.17%	57.2%	38.83%	23.37%	12.59%	6.35%
$\delta = 0$	85.33%	73.31%	57.24%	42.13%	28.33%	18.23%	11.38%
$\delta = -0.1$	86.98%	74.6%	58.19%	43.36%	29.83%	19.11%	11.78%
$\delta = -0.2$	86.93%	75%	59.88%	44.05%	30.38%	19.62%	12.85%
$\delta = -0.3$	86.77%	75.4%	60.11%	44.71%	31.01%	20.44%	13.48%

- Diagonal perturbation E

ε	0	0.01	0.02	0.03	0.04	0.05	0.06
Reference	88.07%	75.17%	57.2%	38.83%	23.37%	12.59%	6.35%
$\delta = 0$	87.05%	75.92%	60.75%	45.69%	31.62%	20.41%	13.15%
$\delta = -0.1$	87.41%	76.12%	61.33%	46.2%	32.53%	21.53%	14.24%
$\delta = -0.2$	87.23%	76.15%	61.55%	46.44%	33.11%	22.14%	14.63%
$\delta = -0.3$	87.18%	75.43%	60.47%	45.49%	31.8%	20.88%	13.22%

Take home message

- Our goal is to make a neural ODE contractive.

Take home message

- Our goal is to make a neural ODE contractive.
- We have encountered an eigenvalue optimization problem.

Take home message

- Our goal is to make a neural ODE contractive.
- We have encountered an eigenvalue optimization problem.
- We have embedded the developed algorithm in the state-of-the-art training strategy of a neural ODE.

Take home message

- Our goal is to make a neural ODE contractive.
- We have encountered an eigenvalue optimization problem.
- We have embedded the developed algorithm in the state-of-the-art training strategy of a neural ODE.
- Numerical experiments show a significant improvement in robustness.

Some references

- N. Guglielmi, A. De Marinis, A. Savostianov and F. Tudisco, *Contractivity of neural ODEs: an eigenvalue optimization problem*, arXiv preprint arXiv:2402.13092, 2024.
- R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, *Neural ordinary differential equations*, Advances in Neural Information Processing Systems, 2018.

Some references

- N. Guglielmi, A. De Marinis, A. Savostianov and F. Tudisco, *Contractivity of neural ODEs: an eigenvalue optimization problem*, arXiv preprint arXiv:2402.13092, 2024.
- R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, *Neural ordinary differential equations*, Advances in Neural Information Processing Systems, 2018.

Thanks for your attention!