Deep learning as optimal control problems

Models and numerical methods

Martin Benning, Queen Mary University of London (QMUL)

This is joint work with Elena Celledoni, Matthias J. Ehrhardt, Brynjulf Owren and Carola-Bibiane Schönlieb

Regularisation for Inverse Problems and Machine Learning, Campus Jussieu, Sorbonne Université, Paris

19.11.2019

This is joint work with

MB, Elena Celledoni, Matthias J. Ehrhardt, Brynjulf Owren, and Carola-Bibiane Schönlieb. "Deep learning as optimal control problems: models and numerical methods." *arXiv preprint arXiv:1904.05657* (2019). To appear in Journal of Computational Dynamics in December 2019

Deep learning as optimal control problems

- Introduction
- Runge-Kutta (RK) networks
- Numerical results
- Regularisation properties of RK network

INTRODUCTION

Introduction

In this part we consider supervised machine learning problems, e.g. classification

Goal of classification is to find mapping $g : \mathbb{R}^n \to \{c^0, c^1, \ldots, c^{K-1}\}$ given input and output samples $\{(x_i, c_i)\}_{i=1}^m$.

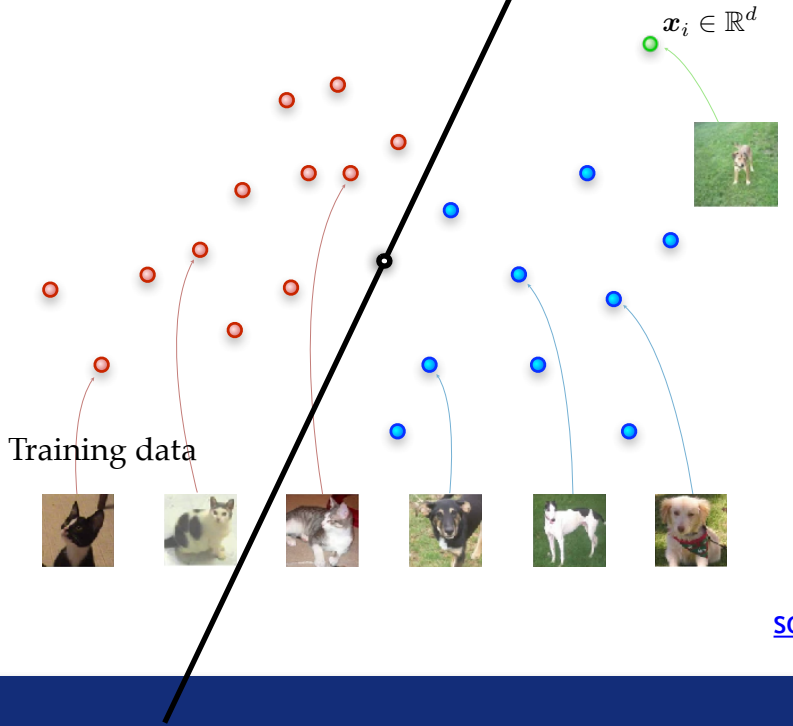
Here every *class label* $c_i \in \{c^0, c^1, \ldots, c^{K-1}\}$ for $i \in \{1, \ldots, m\}$ indicates that the input x_i belongs to the corresponding class, e.g.

$c_i = c^l$ for $l \in \{0, 1, \ldots, K-1\}$, then x_i belongs to class l

Introduction

Example:

image source



source

Introduction

In this part we consider supervised machine learning problems, e.g. classification

Goal of classification is to find mapping $g : \mathbb{R}^n \to \{c^0, c^1, \ldots, c^{K-1}\}$ given input and output samples $\{(x_i, c_i)\}_{i=1}^m$.

A potential model for such a classifier function g is

$$g(x) := \mathscr{C}\left(\hat{W} h(x, \hat{u}) + \hat{\mu}\right)$$

- $\mathscr{C}$ is the *hypothesis* function mapping from $\mathbb{R}$ to $\{c^0, c^1, \ldots, c^{K-1}\}$
- $\hat{W} \in \mathbb{R}^{1\times n}$ is a *weight* vector
- $\hat{\mu} \in \mathbb{R}$ is a *bias* factor
- $h : \mathbb{R}^n \times P \to \mathbb{R}^n$ is a model function parametrised by parameters $\hat{u} \in P$

Introduction

In this part we consider supervised machine learning problems, e.g. classification

Goal of classification is to find mapping $g : \mathbb{R}^n \to \{c^0, c^1, \ldots, c^{K-1}\}$ given input and output samples $\{(x_i, c_i)\}_{i=1}^m$.

A potential model for such a classifier function g is

$$g(x) := \mathscr{C}\left(\hat{W}h(x, \hat{u}) + \hat{\mu}\right)$$

We can train the model parameters by minimising a cost function of the form

$$\sum_{i=1}^{m} L\left(\mathscr{C}(W\,h(x_i, u) + \mu), c_i\right) + \mathscr{R}(u)$$

with respect to u, W and μ

Introduction

In this part we consider supervised machine learning problems, e.g. classification

Goal of classification is to find mapping $g : \mathbb{R}^n \to \{c^0, c^1, \ldots, c^{K-1}\}$ given input and output samples $\{(x_i, c_i)\}_{i=1}^m$.

A potential model for such a classifier function g is

$$g(x) := \mathscr{C}\left(\hat{W}h(x, \hat{u}) + \hat{\mu}\right)$$

We can train the model parameters by minimising a cost function of the form

$$\frac{1}{2}\sum_{i=1}^{m} \left| \mathscr{C}\left(Wh(x_i, u) + \mu\right) - c_i \right|^2 + \mathscr{R}(u)$$

with respect to u, W and μ

Introduction

In this part we consider supervised machine learning problems, e.g. classification

Goal of classification is to find mapping $g : \mathbb{R}^n \to \{c^0, c^1, \ldots, c^{K-1}\}$ given input and output samples $\{(x_i, c_i)\}_{i=1}^m$.

A potential model for such a classifier function g is

$$g(x) := \mathscr{C}\left(\hat{W}h(x, \hat{u}) + \hat{\mu}\right)$$

We can train the model parameters by minimising a cost function of the form

$$\sum_{i=1}^{m}\left[\log\left(1 + \exp(Wh(x_i, u) + \mu)\right) - c_i\left(Wh(x_i, \hat{u}) + \mu\right)\right] + \mathscr{R}(u)$$

with respect to u, W and μ

Introduction

In this part we consider supervised machine learning problems, e.g. classification

We can train the model parameters by minimising a cost function of the form

$$\frac{1}{2}\sum_{i=1}^{m}\left|\mathscr{C}\left(Wh(x_i,u)+\mu\right)-c_i\right|^2+\mathscr{R}(u)$$

with respect to u, W and μ

How do we choose the model function h?

For example as a deep neural network such as the *residual network* (ResNet), i.e.

$$h(x_i,u)=y_i^{[N]},\qquad u=\left(u^{[0]},\ldots,u^{[N-1]}\right),$$

$$y_i^{[j+1]}=y_i^{[j]}+f(y_i^{[j]},u^{[j]}),\qquad j=0,\ldots,N-1,\qquad y_i^{[0]}=x_i.$$

Introduction

How do we choose the model function h?

For example as a deep neural network such as the *residual network* (ResNet), i.e.

$$h(x_i, u) = y_i^{[N]}, \qquad u = \left(u^{[0]}, \ldots, u^{[N-1]}\right),$$

$$y_i^{[j+1]} = y_i^{[j]} + f(y_i^{[j]}, u^{[j]}), \qquad j = 0, \ldots, N-1, \qquad y_i^{[0]} = x_i .$$

Possible choices for $u^{[j]}$ and f:

$$u^{[j]} := \left(K^{[j]}, \beta^{[j]}\right), \qquad j = 0, \ldots, N-1, \qquad K^{[j]} \in \mathbb{R}^{n \times n} \qquad \beta^{[j]} \in \mathbb{R}^{n \times 1}$$

$$f(y_i^{[j]}, u^{[j]}) := \sigma(K^{[j]} y_i^{[j]} + \beta^{[j]}) \qquad \sigma(x) = \tanh(x)$$

Introduction

Training the ResNet with (mean) squared error cost function therefore reads as

$$\min_{y,u,W,\mu} \sum_{i=1}^{m} \left| \mathscr{C}\left(W y_i^{[N]} + \mu\right) - c_i \right|^2 + \mathscr{R}(u),$$

subject to

$$y_i^{[j+1]} = y_i^{[j]} + \quad f(y_i^{[j]}, u^{[j]}), \qquad j = 0, \ldots, N-1, \qquad y_i^{[0]} = x_i\,.$$

Introduction

Training the ResNet with (mean) squared error cost function therefore reads as

$$\min_{y,u,W,\mu} \sum_{i=1}^{m} \left| \mathscr{C}\left(W y_i^{[N]} + \mu\right) - c_i \right|^2 + \mathscr{R}(u),$$

subject to

$$y_i^{[j+1]} = y_i^{[j]} + \Delta t\, f(y_i^{[j]}, u^{[j]}), \qquad j = 0,\ldots,N-1, \qquad y_i^{[0]} = x_i\,.$$

By introducing a step-size parameter Δt we can view the constraint as a forward Euler discretisation of the differential equation

$$\dot{y}_i = f(y_i, u), \quad t \in [0,T], \quad y_i(0) = x_i\,.$$

Introduction

In the continuum we therefore obtain the following optimal control problem

$$\min_{y,u,W,\mu} \sum_{i=1}^{m} \left| \mathscr{C}\left(W y_i(T) + \mu\right) - c_i \right|^2 + \mathscr{R}(u),$$

subject to

$$\dot{y}_i = f(y_i, u), \quad t \in [0,T], \quad y_i(0) = x_i\,.$$

B. Chang, L. Meng, E. Haber, L. Ruthotto, D. Begert and E. Holtham, Reversible architectures for arbitrarily deep residual neural networks, in Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

A lot of research in this direction

- Yann LeCun. A theoretical framework for back-propagation. In Proceedings of the 1988 connectionist models summer school, volume 1, pages 21–28. CMU, Pittsburgh, Pa: Morgan Kaufmann, 1988.
- Weinan E, A Proposal on Machine Learning via Dynamical Systems, Comm. in Math. and Stat. 2017.
- B. Chang, L. Meng, E. Haber, L. Ruthotto, D. Begert and E. Holtham, Reversible Architectures for Arbitrarily Deep Residual Neural Networks, arXiv: 1709.03698v1, AAAI (National Conference on Artificial Intelligence).
- E. Haber, L. Ruthotto, Stable Architectures for Deep Neural Networks, arXiv: 1705.03341v2
- Qianxiao Li Shuji Hao, An Optimal Control Approach to Deep Learning and Applications to Discrete-Weight Neural Networks
- Qianxiao Li, Long Chen, Cheng Tai, Weinan E, Maximum Principle Based Algorithms for Deep Learning, Journal of Machine Learning Research 18 (2018).
- L. Ruthotto and E. Haber, Deep Neural Networks Motivated by Partial Differential Equations, arXiv:1804.04272
- Lu, Y., Zhong, A., Li, Q., Bin Dong. (2017, October 27). Beyond Finite Layer Neural Networks: Bridging Deep Architectures and Numerical Differential Equations. arXiv.org.
- Chen, T. Q., Rubanova, Y., Bettencourt, J., Duvenaud, D. (2018). Neural Ordinary Differential Equations. Presented at NeurIPS.
- Gholami, A., Keutzer, K., Biros, G. (2019, February 26). ANODE: Unconditionally Accurate Memory-Efficient Gradients for Neural ODEs.
- Zhang, T., Yao, Z., Gholami, A., Keutzer, K., Gonzalez, J., Biros, G., Mahoney, M. (2019, June 9). ANODEV2: A Coupled Neural ODE Evolution Framework.

A lot of research in this direction

- Yann LeCun. A theoretical framework for back-propagation. In Proceedings of the 1988 connectionist models summer school, volume 1, pages 21–28. CMU, Pittsburgh, Pa: Morgan Kaufmann, 1988.
- Weinan E, A Proposal on Machine Learning via Dynamical Systems, Comm. in Math. and Stat. 2017.
- B. Chang, L. Meng, E. Haber, L. Ruthotto, D. Begert and E. Holtham, Reversible Architectures for Arbitrarily Deep Residual Neural Networks, arXiv: 1709.03698v1, AAAI (National Conference on Artificial Intelligence).
- E. Haber, L. Ruthotto, Stable Architectures for Deep Neural Networks, arXiv: 1705.03341v2
- Qianxiao Li Shuji Hao, An Optimal Control Approach to Deep Learning and Applications to Discrete-Weight Neural Networks
- Qianxiao Li, Long Chen, Cheng Tai, Weinan E, Maximum Principle Based Algorithms for Deep Learning, Journal of Machine Learning Research 18 (2018).
- L. Ruthotto and E. Haber, Deep Neural Networks Motivated by Partial Differential Equations, arXiv:1804.04272
- Lu, Y., Zhong, A., Li, Q., Bin Dong. (2017, October 27). Beyond Finite Layer Neural Networks: Bridging Deep Architectures and Numerical Differential Equations. arXiv.org.
- Chen, T. Q., Rubanova, Y., Bettencourt, J., Duvenaud, D. (2018). Neural Ordinary Differential Equations. Presented at NeurIPS.
- Gholami, A., Keutzer, K., Biros, G. (2019, February 26). ANODE: Unconditionally Accurate Memory-Efficient Gradients for Neural ODEs.
- Zhang, T., Yao, Z., Gholami, A., Keutzer, K., Gonzalez, J., Biros, G., Mahoney, M. (2019, June 9). ANODEV2: A Coupled Neural ODE Evolution Framework.

DEEP LEARNING AS OPTIMAL CONTROL PROBLEMS

Deep learning as optimal control problems

Deep learning optimal control problem

$$\min_{y,u,W,\mu} \sum_{i=1}^{m} \left| \mathscr{C}\left(W y_i(T) + \mu\right) - c_i \right|^2 + \mathscr{R}(u),$$

subject to

$$\dot{y}_i = f(y_i, u), \quad t \in [0,T], \quad y_i(0) = x_i\,.$$

Deep learning as optimal control problems

Deep learning optimal control problem

$$\min_{y,u} \mathcal{J}(y(T))$$

subject to

$$\dot{y} = f(y,u), \quad t \in [0,T], \quad y(0) = x\,.$$

Variational calculus: consider variations

$$\tilde{y} := y + \varepsilon v\,, \qquad \tilde{u} := w + \varepsilon w\,.$$

How does the system $\dot{\tilde{y}} = f(\tilde{y},\tilde{u})$ behave for $\varepsilon \to 0$?

Deep learning as optimal control problems

Differential equation constraint

$$\dot{y} = f(y,u), \quad t \in [0,T], \quad y(0) = x\,.$$

Variational calculus: consider variations

$$\tilde{y} := y + \varepsilon v\,, \qquad \tilde{u} := w + \varepsilon w\,.$$

How does the system $\dot{\tilde{y}} = f(\tilde{y},\tilde{u})$ behave for $\varepsilon \to 0$?

$$\frac{d}{d\varepsilon}\left(\frac{d}{dt}\tilde{y}(t)\right)\Big|_{\varepsilon=0} = \frac{d}{dt}v(t)\,,$$

$$\frac{d}{d\varepsilon} f(\tilde{y}(t),\tilde{u}(t))\Big|_{\varepsilon=0} = \partial_y f(y(t),u(t))\,v(t) + \partial_u f(y(t),u(t))\,w(t)\,.$$

Deep learning as optimal control problems

Deep learning optimal control problem

$$\min_{y,u} \mathcal{J}(y(T))$$

subject to

$$\dot{y} = f(y,u), \quad t \in [0,T], \quad y(0) = x\,.$$

Variational equation

$$\frac{d}{dt}v(t) = \underbrace{\partial_y f(y(t),u(t))}_{\text{Jacobian of } f \text{ w.r.t. to } y}\, v(t) + \underbrace{\partial_u f(y(t),u(t))}_{\text{Jacobian of } f \text{ w.r.t. to } u}\, w(t)$$

Adjoint equation

The adjoint of the variational equation is a system of differential equations for a variable $p(t)$, obtained assuming

$$\langle p(t), v(t) \rangle = \langle p(0), v(0) \rangle, \qquad \forall t \in [0,T] .$$

This implies

$$\langle p(t), \dot{v}(t) \rangle = - \langle \dot{p}(t), v(t) \rangle,$$

which yields

$$\frac{d}{dt} p(t) = - \Big(\partial_y f(y(t), u(t)) \Big)^\top \, p(t) ,$$

with constraint

$$\big(\partial_u f(y(t), u(t)) \big)^\top \, p(t) = 0 .$$

First-order necessary conditions for optimality

Then, the boundary value problem system

$$\begin{aligned}
\dot{y} &= f(y,u), \qquad y(0) = x\,, \\
\dot{p} &= -\Big(\partial_y f(y,u)\Big)^{\top} p\,, \qquad p(T) = \partial_y \mathcal{J}(y)\Big|_{y=y(T)}\,, \\
0 &= \Big(\partial_u f(y,u)\Big)^{\top} p\,, \qquad t \in [0,T]\,,
\end{aligned}$$

expresses the first order necessary conditions for optimality of

$$\min_{y,\,u} \mathcal{J}(y(T))\,,$$

Sketch of proof: set up a Lagrange functional and compute the optimality conditions.

Associated Hamiltonian system

For this optimal control problem, there is an associated Hamiltonian system with Hamiltonian

$$\mathscr{H}(y,p,u) := p^\top f(y,u)$$

with

$$\dot{y} = \partial_p \mathscr{H}, \quad \dot{p} = -\partial_y \mathscr{H}, \quad \partial_u \mathscr{H} = 0.$$

The constraint $0 = \big(\partial_u f(y,u)\big)^\top p$ implies that this is a differential algebraic equation of index one

Provided the Hessian $\partial_{u,u}\mathscr{H}$ is invertible, we can regard this system as an ODE when applying numerical methods.

Numerical discretisation of the optimal control problem

How do we discretise the boundary value problem system

$$\dot{y} = f(y,u), \qquad y(0) = x\,,$$
$$\dot{p} = -\Big(\partial_y f(y,u)\Big)^{\top} p\,, \qquad p(T) = \partial_y \mathcal{J}(y)\Big|_{y=y(T)}\,,$$
$$0 = \Big(\partial_u f(y,u)\Big)^{\top} p\,, \qquad t \in [0,T]\,?$$

Symplectic Runge-Kutta methods are suited to this problem!

Numerical discretisation of the optimal control problem

Symplectic Runge-Kutta methods are suited to this problem!

Forward pass

$$y^{[j+1]} = y^{[j]} + \Delta t \sum_{i=1}^{s} b_i f_i^{[j]} \qquad y^{[0]} = x$$

$$f_i^{[j]} = f(y_i^{[j]}, u_i^{[j]}), \quad i = 1,\dots,s,$$

$$b_i = \tilde{b}_i$$

$$y_i^{[j]} = y^{[j]} + \Delta t \sum_{l=1}^{s} a_{i,l} f_l^{[j]}, \quad i = 1,\dots,s \qquad b_i\tilde{a}_{i,j} + \tilde{b}_j a_{i,j} - b_i\tilde{b}_j = 0$$

Backward pass

$$p^{[j+1]} = p^{[j]} + \Delta t \sum_{i=1}^{s} \tilde{b}_i \ell_i^{[j]} \qquad p^{[N]} := \partial \mathcal{J}(y^{[N]})$$

$$\ell_i^{[j]} = -\partial_y f(y_i^{[j]}, u_i^{[j]})^T p_i^{[j]}, \quad i = 1,\dots,s, \qquad \left(\partial_u f(y_i^{[j]}, u_i^{[j]})\right)^T p_i^{[j]} = 0$$

$$p_i^{[j]} = p^{[j]} + \Delta t \sum_{l=1}^{s} \tilde{a}_{i,l} \ell_l^{[j]}, \quad i = 1,\dots,s$$

This is the symplectic part!

Numerical discretisation of the optimal control problem

Symplectic Runge-Kutta methods are suited to this problem!

Special case: symplectic Euler

$$y^{[j+1]} = y^{[j]} + \Delta t\, f(y^{[j]}, u^{[j]}), \qquad y^{[0]} = x$$

$$p^{[j+1]} = p^{[j]} - \Delta t \left(\partial_y f(y^{[j]}, u^{[j]})\right)^T p^{[j+1]}, \qquad p^{[N]} := \partial_y \mathcal{J}(y^{[N]})$$

$$0 = \left(\partial_u f(y_i^{[j]}, u_i^{[j]})\right)^T p_i^{[j+1]}$$

Has the advantage that bilinear forms are preserved after discretisation, i.e. $\langle p(t), v(t) \rangle$ will be preserved after discretisation.

Numerical discretisation of the optimal control problem

Proposition: if the continuous optimal control system is discretised with a symplectic partitioned Runge-Kutta method with $b_i \neq 0$, then the first order optimality for the discrete control problem

$$\min_{\{u_i^{[j]}\}_{j=0}^{N-1},} \quad \mathcal{J}(y^{[N]}),$$

$$\{y^{[j]}\}_{j=1}^{N}, \{y_i^{[j]}\}_{j=1}^{N},$$

subject to

$$y^{[j+1]} = y^{[j]} + \Delta t \sum_{i=1}^{s} b_i f_i^{[j]}$$

$$f_i^{[j]} = f(y_i^{[j]}, u_i^{[j]}), \quad i = 1, \ldots, s,$$

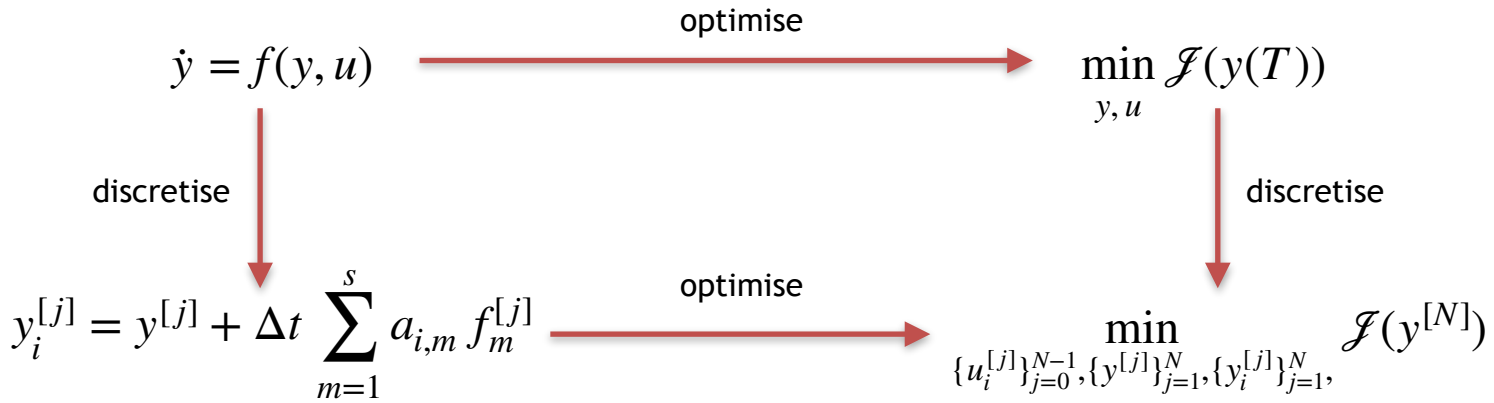
$$y_i^{[j]} = y^{[j]} + \Delta t \sum_{m=1}^{s} a_{i,m} f_m^{[j]}, \quad i = 1, \ldots, s,$$

is satisfied.

Numerical discretisation of the optimal control problem

What does this mean in practice?

If we use symplectic partitioned Runge-Kutta methods, then *first-optimise-then-discretise* and *first-discretise-then-optimise* lead to the same optimal control problems.



RUNGE-KUTTA NETWORKS

Runge-Kutta networks

The discretisation of the optimal control problem with symplectic partitioned Runge-Kutta methods inspires a range of different network architectures.

Activation function: $f(y,u) = \sigma(Ky + \beta)$ with parameters $u = (K, \beta)$

Runge-Kutta networks:

$$y^{[j+1]} = y^{[j]} + \Delta t \sum_{i=1}^{s} b_i f_i^{[j]}$$

$$f_i^{[j]} = \sigma(K_i^{[j]} y_i^{[j]} + \beta_i^{[j]}), \quad i = 1, \ldots, s,$$

$$y_i^{[j]} = y^{[j]} + \Delta t \sum_{l=1}^{s} a_{i,l} f_l^{[j]}, \quad i = 1, \ldots, s$$

Runge-Kutta networks

The discretisation of the optimal control problem with symplectic partitioned Runge-Kutta methods inspires a range of different network architectures.

Activation function: $f(y,u) = \sigma(Ky + \beta)$ with parameters $u = (K, \beta)$

Runge-Kutta networks: special case $s = 1$, the ResNet

Backpropagation:

$$y^{[j+1]} = y^{[j]} + \Delta t\, \sigma\left(K^{[j]} y^{[j]} + \beta^{[j]}\right)$$

$$\gamma^{[j]} = \sigma'\left(K^{[j]} y^{[j]} + \beta^{[j]}\right) \odot p^{[j+1]}$$

$$p^{[j+1]} = p^{[j]} - \Delta t\, (K^{[j]})^{\top} \gamma^{[j]}$$

Runge-Kutta networks

The discretisation of the optimal control problem with symplectic partitioned Runge-Kutta methods inspires a range of different network architectures.

Activation function: $f(y,u) = \sigma(Ky + \beta)$ with parameters $u = (K, \beta)$

Runge-Kutta networks: special case $s = 1$, the ResNet

$$y^{[j+1]} = y^{[j]} + \Delta t\, \sigma\left(K^{[j]} y^{[j]} + \beta^{[j]}\right)$$
$$\gamma^{[j]} = \sigma'\left(K^{[j]} y^{[j]} + \beta^{[j]}\right) \odot p^{[j+1]}$$
$$p^{[j+1]} = p^{[j]} - \Delta t\, (K^{[j]})^\top \gamma^{[j]}$$

Parameter gradient:

$$\partial_{K^{[j]}} \mathcal{J}(y^{[N]}) = \Delta t\, \gamma^{[j]} (y^{[j]})^\top$$
$$\partial_{\beta^{[j]}} \mathcal{J}(y^{[N]}) = \Delta t\, \gamma^{[j]}$$

ODENet

The discretisation of the optimal control problem with symplectic partitioned Runge-Kutta methods inspires a range of different network architectures.

Activation function: $f(y,u) = \alpha\,\sigma(Ky+\beta)$ with parameters $u = (K, \alpha, \beta)$

Example: ResNet with varying time steps, i.e.

$$y^{[j+1]} = y^{[j]} + \Delta t\,\alpha^{[j]}\,\sigma\left(K^{[j]}y^{[j]} + \beta^{[j]}\right)\ ;$$

we refer to this model as the *ODENet*.

Parameter gradient: $\partial_{\alpha^{[j]}}\mathcal{J}(y^{[N]}) = \Delta t\,\left\langle p^{[j+1]}, \sigma\left(K^{[j]}y^{[j]} + \beta^{[j]}\right)\right\rangle$

ODENet

The discretisation of the optimal control problem with symplectic partitioned Runge-Kutta methods inspires a range of different network architectures.

Activation function: $f(y,u) = \alpha\,\sigma(Ky+\beta)$ with parameters $u = (K, \alpha, \beta)$

Example: ResNet with varying time steps, i.e.

$$y^{[j+1]} = y^{[j]} + \Delta t\,\alpha^{[j]}\,\sigma\left(K^{[j]}y^{[j]} + \beta^{[j]}\right)\ ;$$

we refer to this model as the *ODENet*.

In some applications it can make sense to assume that the learned time steps have to lie in the simplex

$$S := \left\{ \alpha \in \mathbb{R}^N \;\middle|\; \alpha^{[j]} \geq 0,\ \sum_{j=1}^{N} \alpha^{[j]} = 1 \right\}\ .$$

NUMERICAL RESULTS

Setup

Deep learning optimal control problem

$$\min_{y,u,W,\mu} \sum_{i=1}^{m} \left| \mathscr{C}\left(W y_i(T) + \mu\right) - c_i \right|^2 + \mathscr{R}(u),$$

subject to

$$\dot{y}_i = f(y_i, u), \quad t \in [0,T], \quad y_i(0) = x_i\,.$$

Binary classification: $c_i \in \{0,1\}$

Hypothesis function: $\mathscr{C}(x) = \dfrac{1}{1+\exp(-x)}$

Activation function: $\sigma(x) = \tanh(x)$

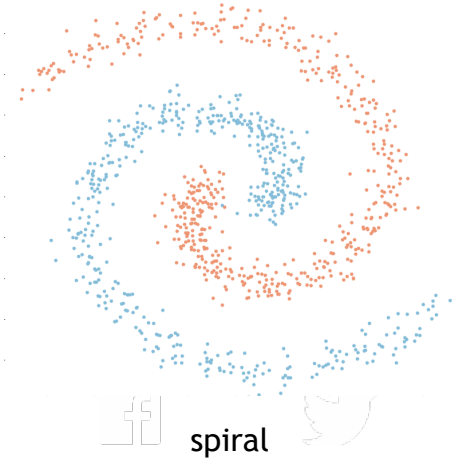
Varying number of layers after discretisation

Datasets

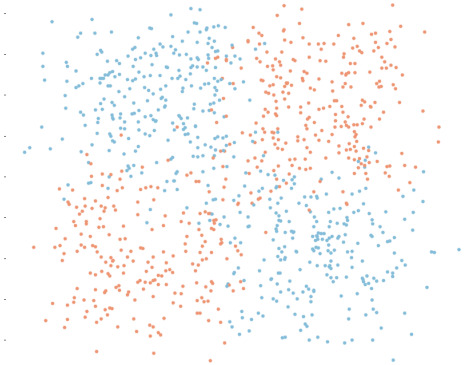
2D data sets

donut1d

donut2d

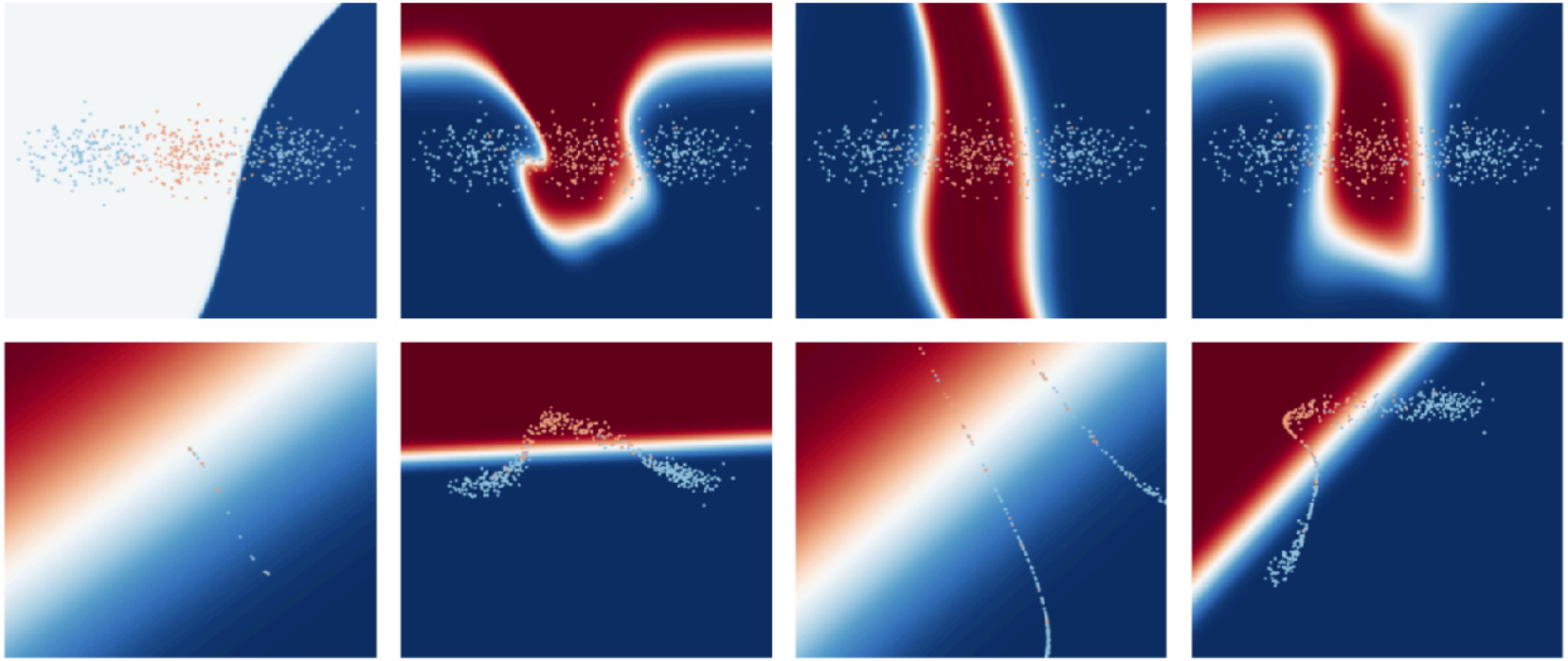


spiral



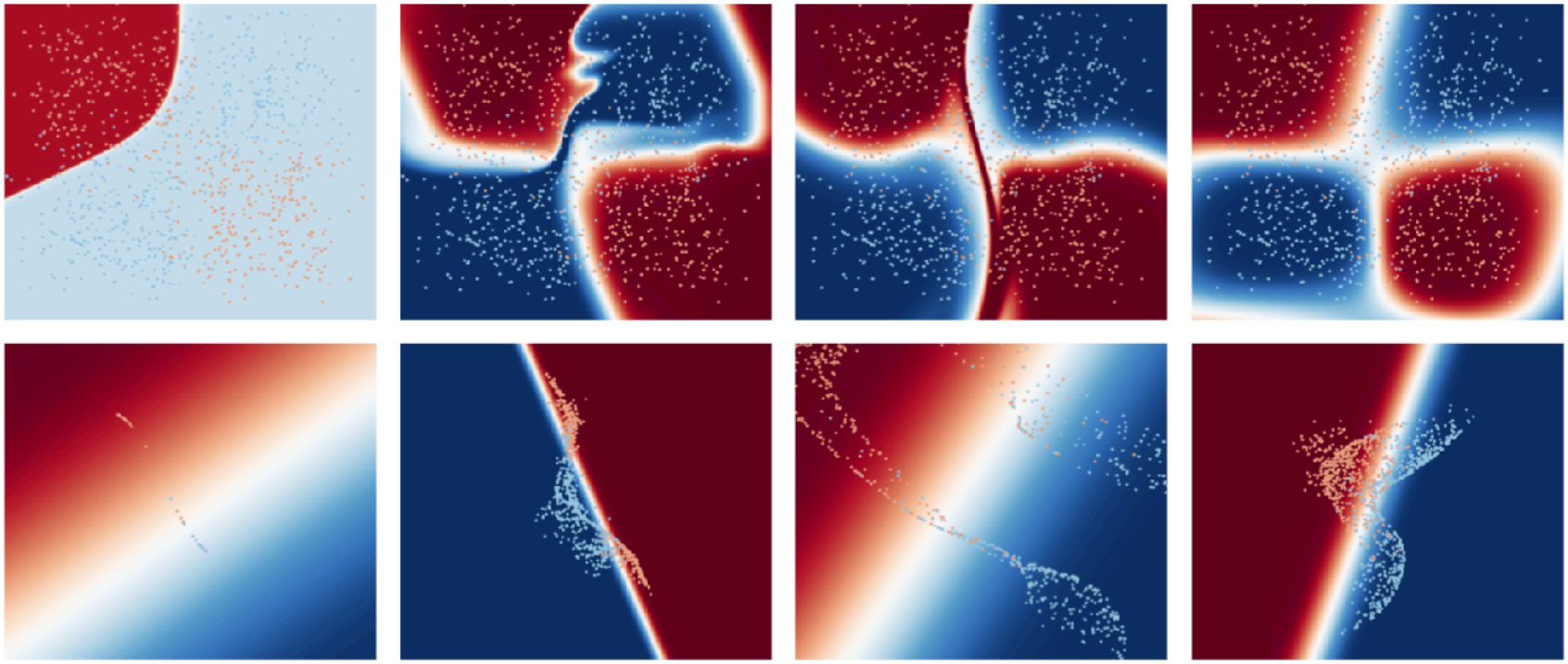
squares

Results for donut1d



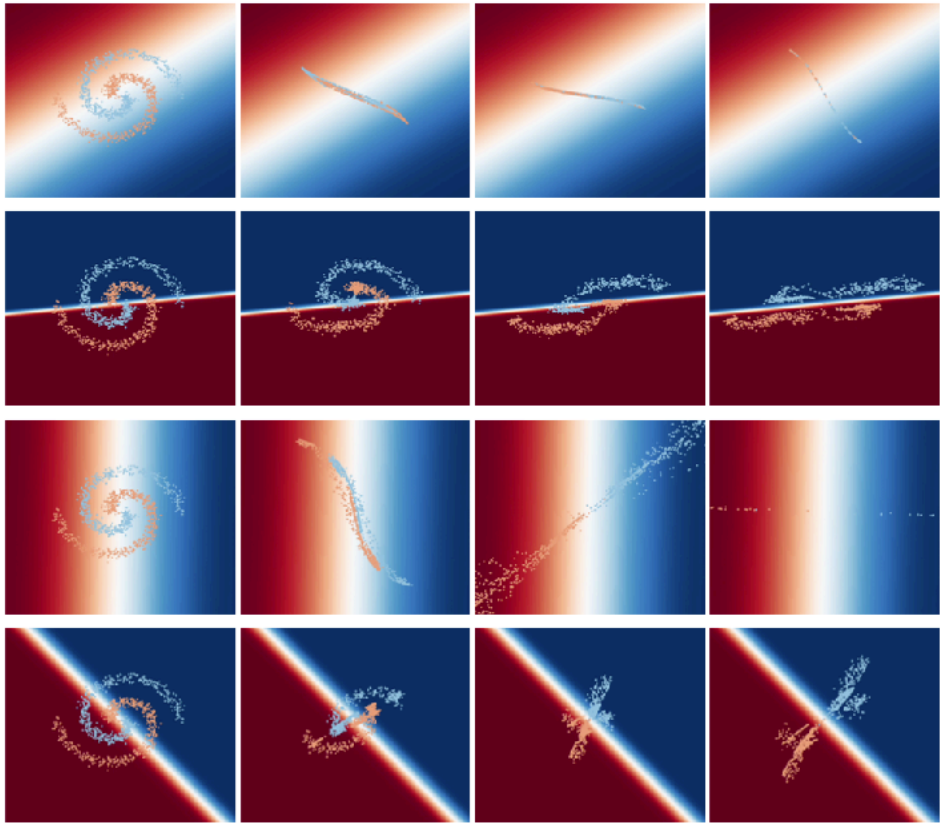
Prediction (top) and transformed data (`donut1d`) with linear classifier (bottom) and for Net, ResNet, ODENet and ODENet+simplex.

Results for squares



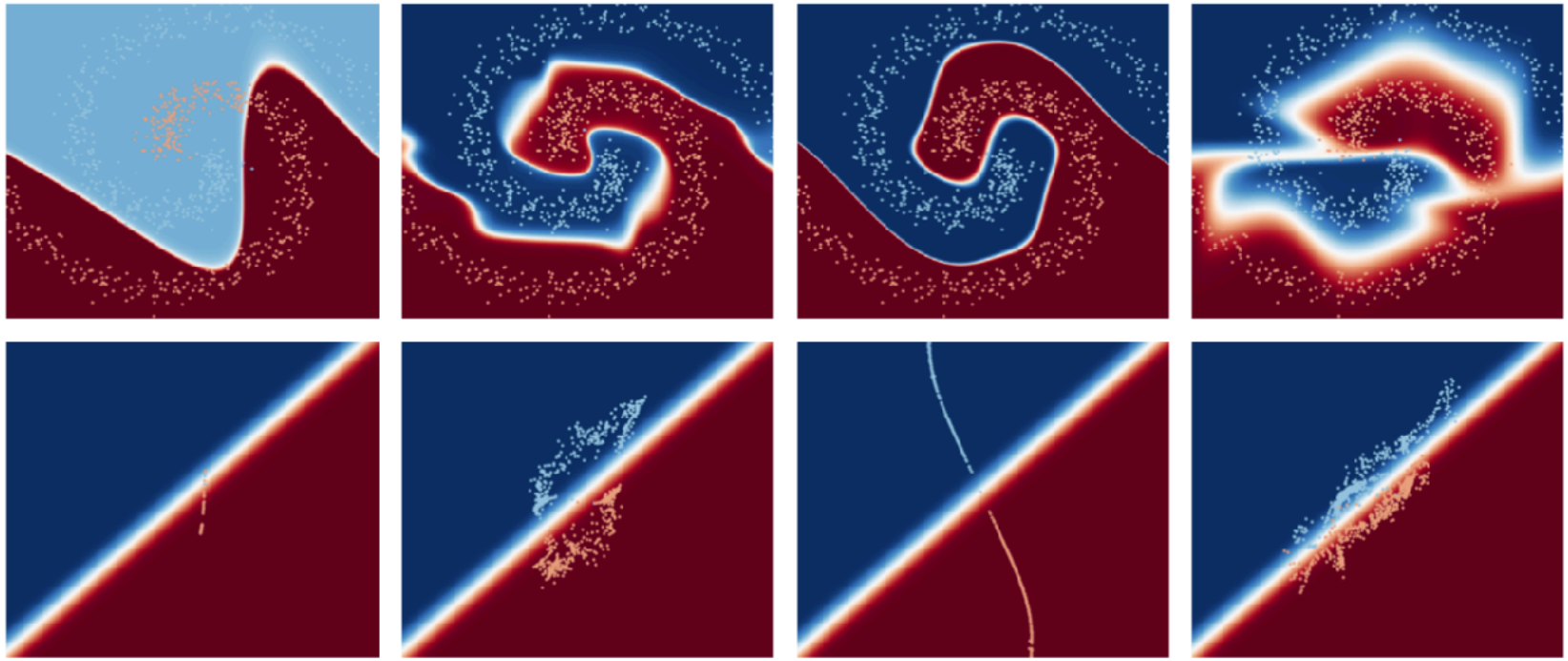
Prediction (top) and transformed data (`squares`) with linear classifier (bottom) and for Net, ResNet, ODENet and ODENet+simplex.

Transformation of features



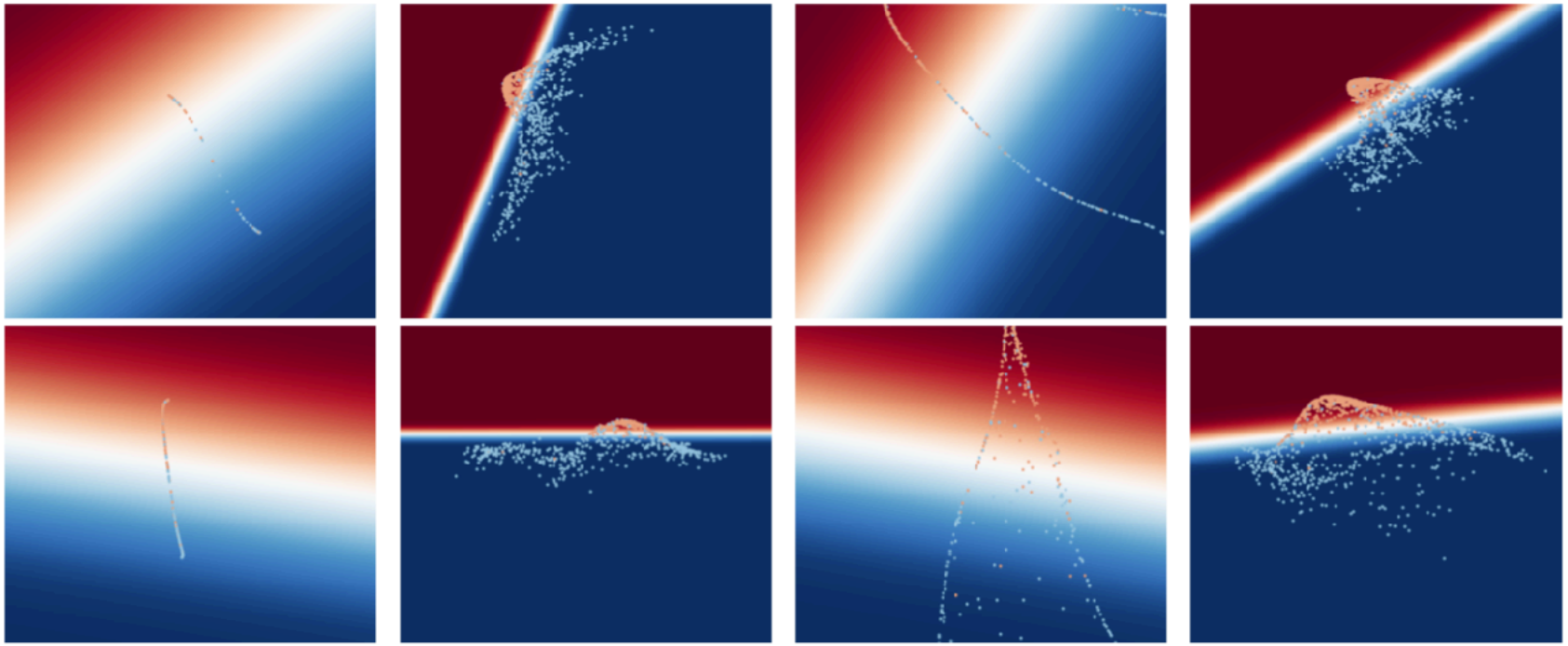
Transformation of the features for `spiral`. From top to bottom: Net, ResNet, ODENet, ODENet | simplex.

Transformations for fixed classifier



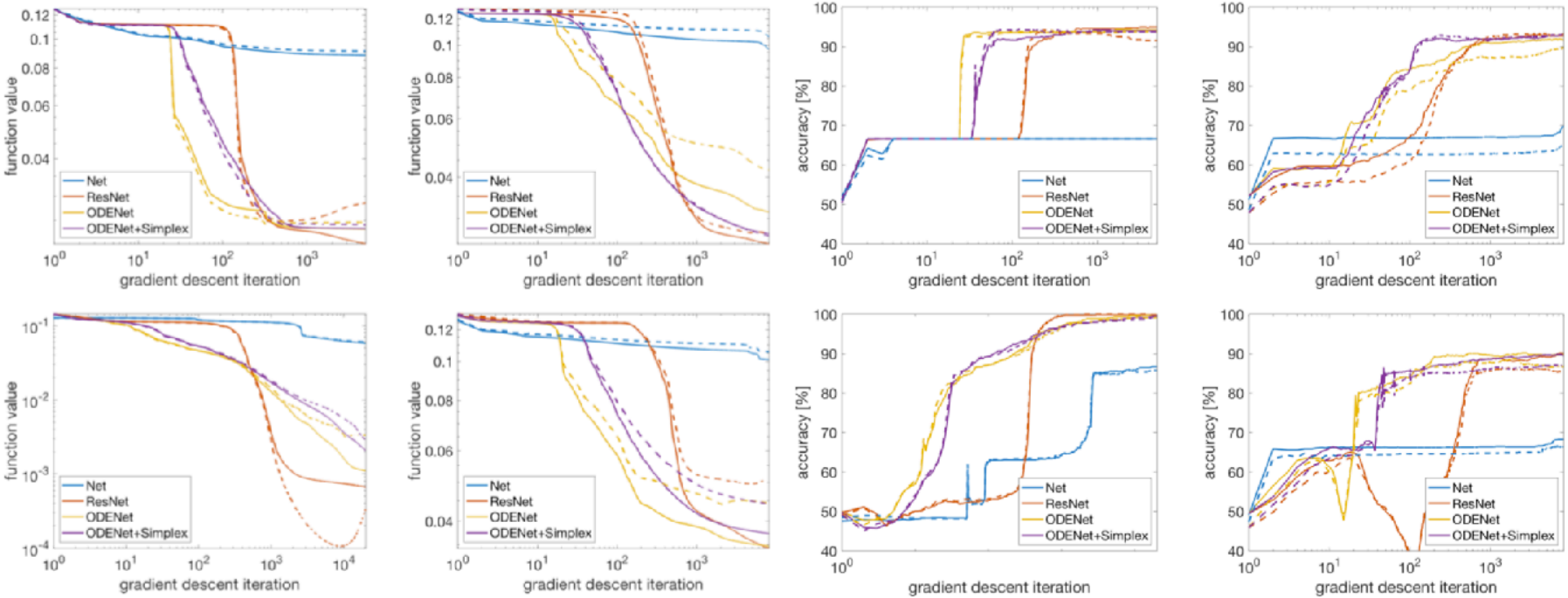
Learned transformation with fixed classifier. Prediction (top) and transformed data (`spiral`) with linear classifier (bottom) and for Net, ResNet, ODENet and ODENet+simplex.

Robustness to initialisations



Robustness on random initialisation. Transformed data `donut2d` with linear classifier for Net, ResNet, ODENet and ODENet-Simplex for two different initializations.

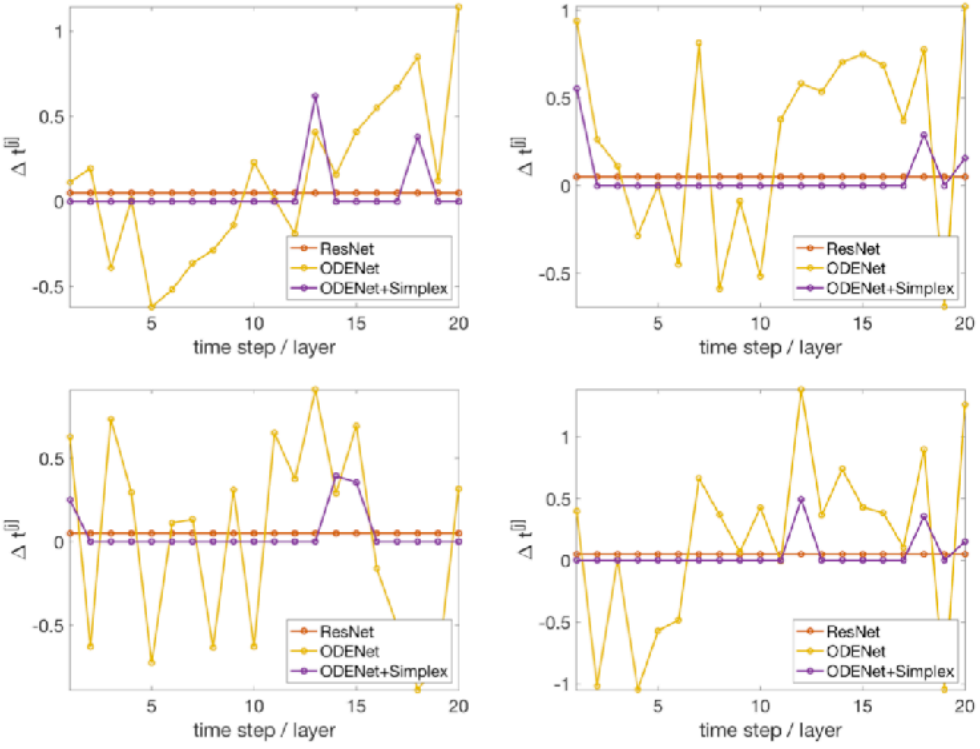
Optimisation



Function values over the course of the gradient descent iterations for data sets `donut1d`, `donut2d`, `spiral`, `squares`. The solid line represents training and the dashed line test data.

Classification accuracy over the course of the gradient descent iterations for data sets `donut1d`, `donut2d`, `spiral`, `squares`. The solid line represents training and the dashed line test data.

Adaptive time steps for ODENet



Estimated time steps by ResNet/Euler, ODENet and ODENet+simplex for for data sets `donut1d`, `donut2d`, `spiral`, `squares`. ODENet+simplex consistently picks two to three time steps and set the rest to zero.

Comparison of Runge-Kutta Networks

We now train different Runge-Kutta networks for the same classification tasks

Butcher tableau

$$\begin{array}{c|ccccc} 0 & & & & & \\ c_2 & a_{21} & & & & \\ c_3 & a_{21} & a_{32} & & & \\ \vdots & \vdots & & \ddots & & \\ c_s & a_{s1} & a_{s2} & \cdots & a_{s,s-1} & \\ \hline & b_1 & b_2 & \cdots & b_{s-1} & b_s \end{array}$$

ResNet/Euler

$$\begin{array}{c|cc} 0 & & \\ & & 1 \end{array}$$

Improved Euler

$$\begin{array}{c|cc} 0 & & \\ 1 & 1 & \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

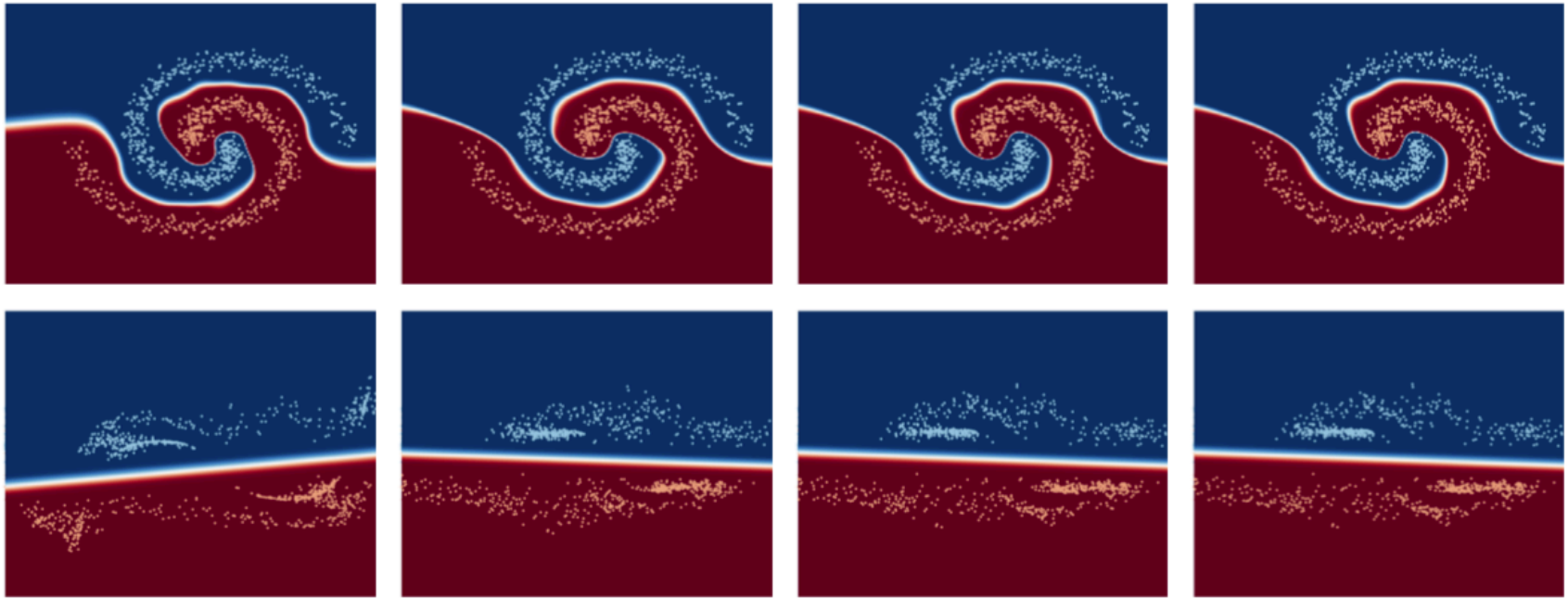
Kutta(3)

$$\begin{array}{c|ccc} 0 & & & \\ \frac{1}{2} & \frac{1}{2} & & \\ 1 & -1 & 2 & \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{array}$$

Kutta(4)

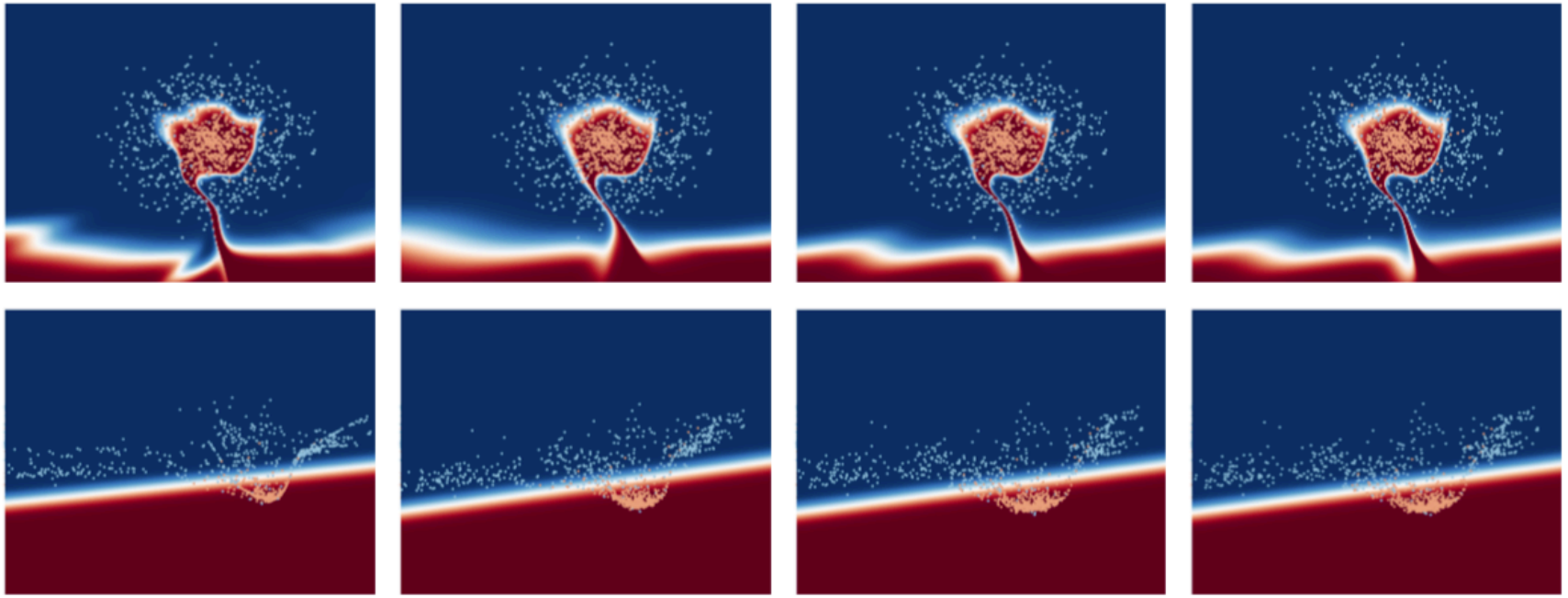
$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array}$$

Comparison of Runge-Kutta Networks



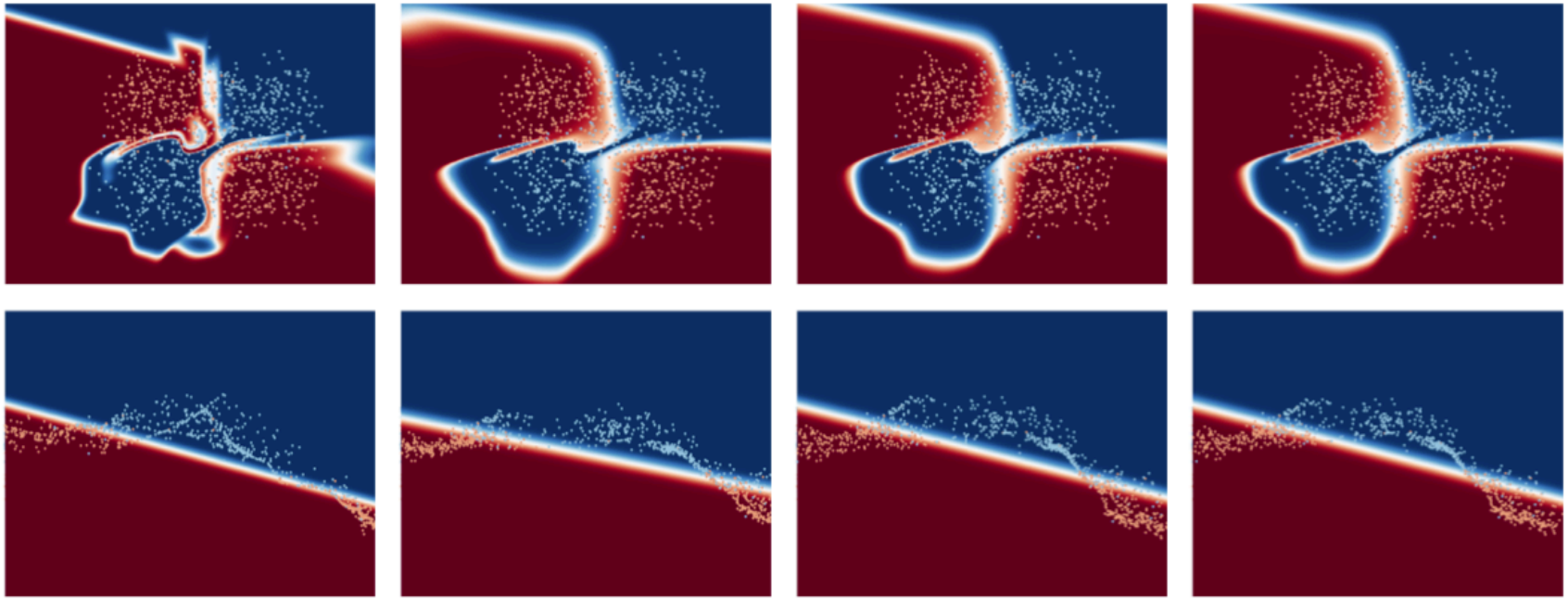
Prediction (top) and transformed points (bottom) for the Runge–Kutta methods (left to right) ResNet/Euler, Improved Euler, Kutta(3) and Kutta(4). The data set is `spiral` and we use 15 layers.

Comparison of Runge-Kutta Networks



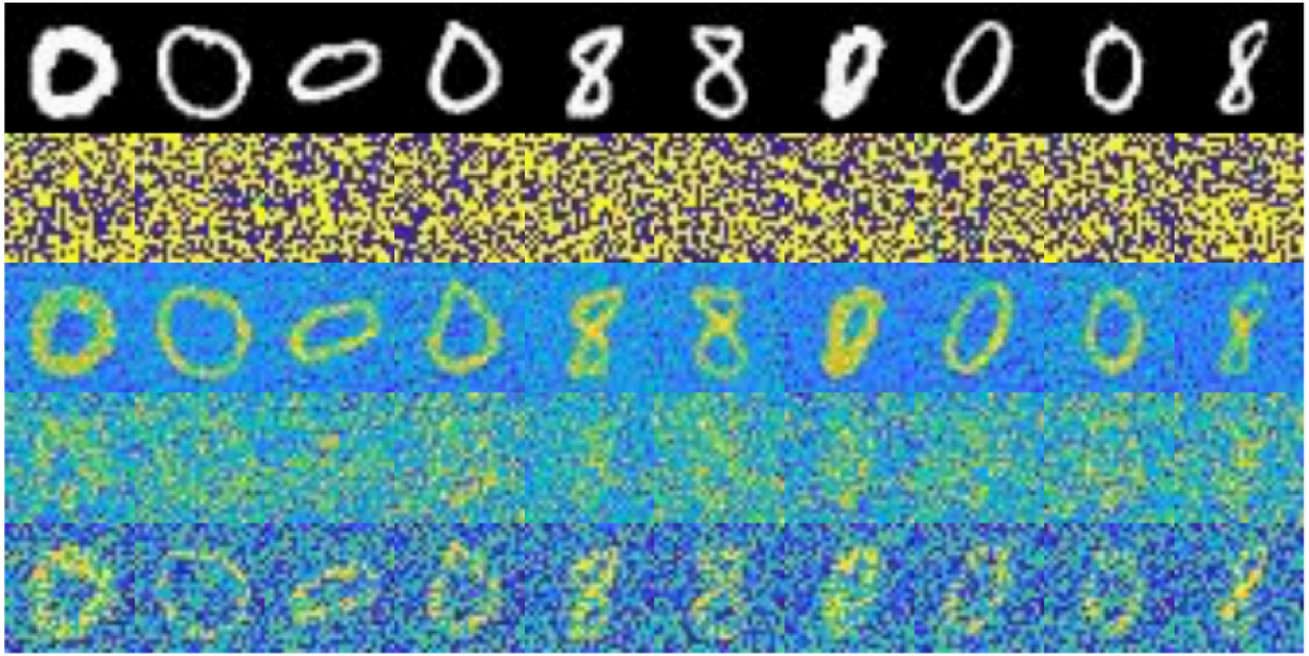
Prediction (top) and transformed points (bottom) for the Runge–Kutta methods (left to right) ResNet/Euler, Improved Euler, Kutta(3) and Kutta(4). The data set is `donut2d` and we use 15 layers.

Comparison of Runge-Kutta Networks



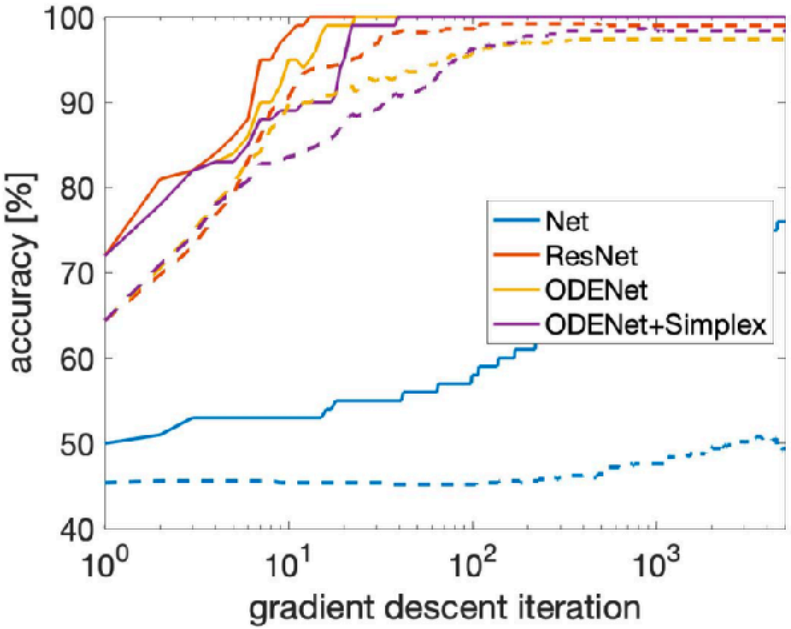
Prediction (top) and transformed points (bottom) for the Runge–Kutta methods (left to right) ResNet/Euler, Improved Euler, Kutta(3) and Kutta(4). The data set is `squares` and we use 15 layers.

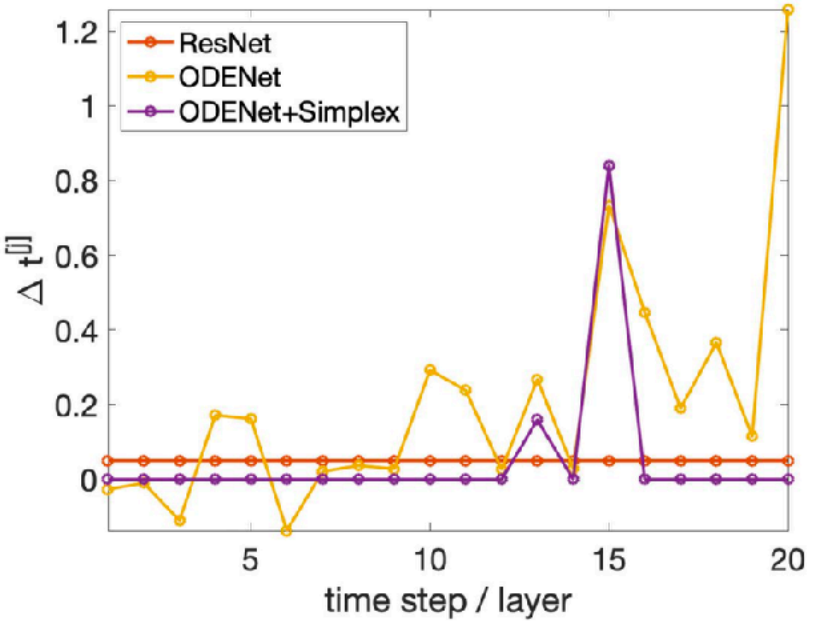
ODENet for MNIST



Features of testing examples from MNIST100 dataset [28] and transformed features by four networks under comparison: Net, ResNet, ODENet, ODENet+Simplex (from top to bottom). All networks have 20 layers.

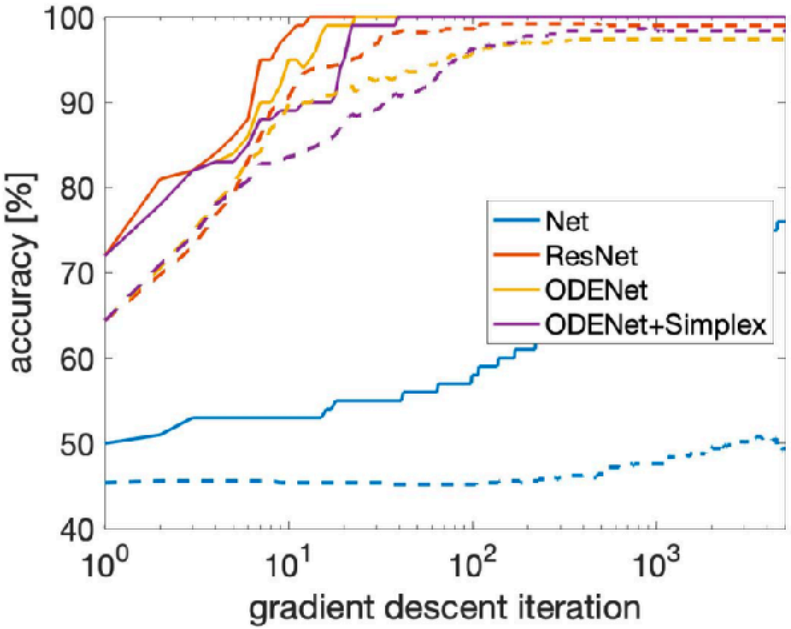
ODENet for MNIST

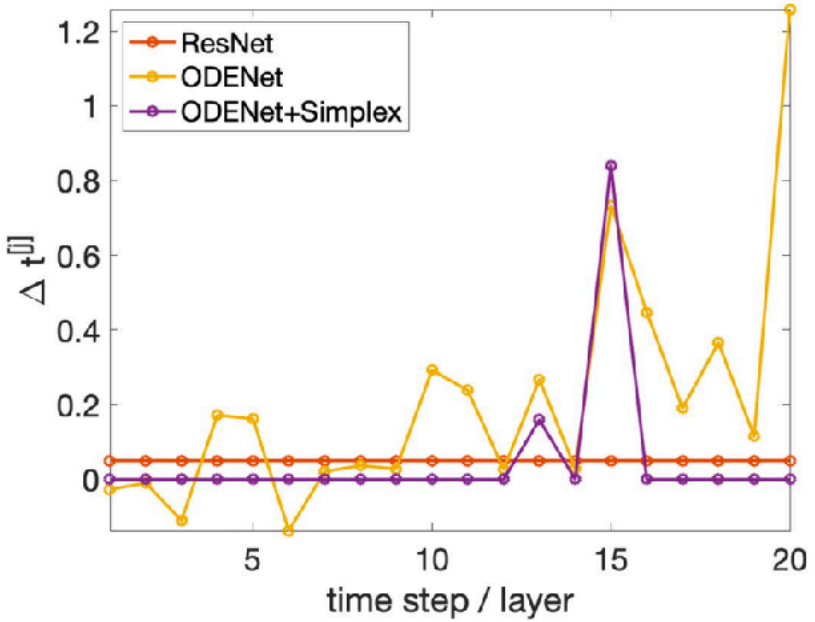




Accuracy (left) and time steps (right) for MNIST100 dataset [28].

ODENet for MNIST





Accuracy (left) and time steps (right) for MNIST100 dataset

REGULARISATION PROPERTIES

Regularisation properties

Let us consider the continuous formulation of the neural network constraint

$$\frac{d}{dt}y(t) = \sigma\left(K(t)y(t) + \beta(t)\right)$$

Regularisation properties

Let us consider the continuous formulation of the neural network constraint

$$\frac{d}{dt}y(t) = -K(t)^\top \sigma\left(K(t)y(t) + \beta(t)\right)$$

and modify it as suggested in *

Then we can verify the following stability estimate if σ is chosen to be monotone:

$$\|y(t) - y^\delta(t)\| \leq C \|y(0) - y^\delta(0)\|, \qquad \text{for } t \geq 0.$$

Here $C > 0$ is a constant and $y(t)$ and $y^\delta(t)$ denote solutions of the flow for initial values $y(0)$ and $y^\delta(0)$.

*Lars Ruthotto and Eldad Haber. "Deep neural networks motivated by partial differential equations." *arXiv preprint arXiv:1804.04272* (2018).

Regularisation properties

Then we can verify the following stability estimate if σ is chosen to be monotone:

$$\|y(t) - y^\delta(t)\| \leq C \|y(0) - y^\delta(0)\|, \qquad \text{for } t \geq 0.$$

Proof:

$$\frac{d}{dt}\left(\frac{1}{2}\|y(t) - y^\delta(t)\|\right)^2 = \left\langle \frac{d}{dt}y(t) - \frac{d}{dt}y^\delta(t), y(t) - y^\delta(t) \right\rangle$$

$$= -\left\langle K(t)^\top \Big(\sigma\left(K(t)y(t) + \beta(t)\right) - \sigma\left(K(t)y^\delta(t) + \beta(t)\right)\Big), y(t) - y^\delta(t) \right\rangle$$

$$= -\left\langle \sigma\left(K(t)y(t) + \beta(t)\right) - \sigma\left(K(t)y^\delta(t) + \beta(t)\right), K(t)y(t) + \beta(t) - \left(K(t)y^\delta(t) + \beta(t)\right) \right\rangle$$

Regularisation properties

Then we can verify the following stability estimate if σ is chosen to be monotone:

$$\|y(t) - y^\delta(t)\| \leq C \|y(0) - y^\delta(0)\|, \qquad \text{for } t \geq 0.$$

Proof:

$$\frac{d}{dt}\left(\frac{1}{2}\|y(t) - y^\delta(t)\|\right)^2 = \left\langle \frac{d}{dt}y(t) - \frac{d}{dt}y^\delta(t), y(t) - y^\delta(t) \right\rangle$$

$$= -\left\langle \sigma\left(K(t)y(t) + \beta(t)\right) - \sigma\left(K(t)y^\delta(t) + \beta(t)\right), K(t)y(t) + \beta(t) - \left(K(t)y^\delta(t) + \beta(t)\right) \right\rangle$$

$$= -\left\langle \sigma(z(t)) - \sigma(z^\delta(t)), z(t) - z^\delta(t) \right\rangle \leq 0,$$

for $z(t) := K(t)y(t) + \beta(t)$ and $z^\delta(t) := K(t)y^\delta(t) + \beta(t)$.

Regularisation properties

Then we can verify the following stability estimate if σ is chosen to be monotone:

$$\|y(t) - y^\delta(t)\| \leq C \|y(0) - y^\delta(0)\|, \qquad \text{for } t \geq 0.$$

Proof:

$$\frac{d}{dt}\left(\frac{1}{2}\|y(t) - y^\delta(t)\|\right)^2 \leq 0.$$

Integrating from 0 to t now yields

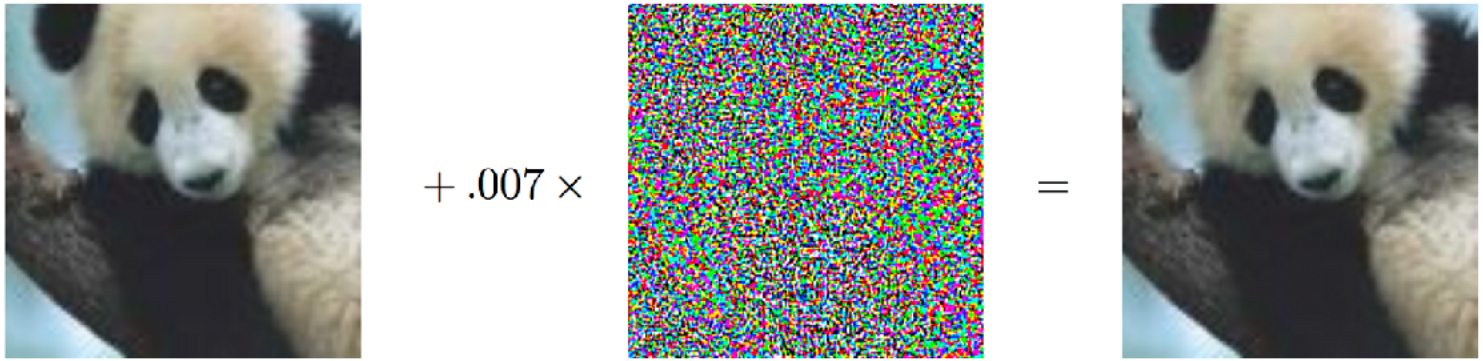
$$\frac{1}{2}\|y(t) - y^\delta(t)\|^2 - \frac{1}{2}\|y(0) - y^\delta(0)\|^2 \leq 0,$$

which concludes the proof. ∎

Regularisation properties

Is stability a desirable property in neural networks?

Adversarial examples in image classification:



Regularisation properties

Hardening of deep neural networks:

suppose we have inputs $y(0) = x$ and $y^\delta(0) = x^\delta$ with $\|x - x^\delta\| \leq \delta$, then our network produces outputs with

$$\|y(t) - y^\delta(t)\| \leq C\,\delta\,.$$

Hence, we observe

$$\lim_{\delta \to 0} \|y(t) - y^\delta(t)\| = 0\,.$$

This continuous dependency is also a feature of regularisation operators;

We should discretise or train such that this stability estimate is preserved.

Regularisation properties

Connection to inverse problems for example with additional reaction term:

$$\frac{d}{dt}y(t) = -K(t)^\top \sigma\left(K(t)y(t) + \beta(t)\right) - \lambda A^\top\left(Ay(t) - f\right)$$

Let $y(t)$ and $y^\delta(t)$ denote solutions of this flow for data f and f^δ with $\|f - f^\delta\| \leq \delta$ and identical initial values. For this model we can derive the following stability estimate:

$$\begin{aligned}\frac{d}{dt}\left(\frac{1}{2}\|y(t) - y^\delta(t)\|^2\right) &\leq -\lambda\left\langle A^\top(Ay(t) - f) - A^\top(Ay^\delta(t) - f^\delta), y(t) - y^\delta(t)\right\rangle \\ &= -\lambda\left\langle A(y(t) - y^\delta(t)) - (f - f^\delta), A(y(t) - y^\delta(t))\right\rangle\end{aligned}$$

Regularisation properties

Connection to inverse problems for example with additional reaction term:

$$\frac{d}{dt}y(t) = -K(t)^\top \sigma\left(K(t)y(t) + \beta(t)\right) - \lambda A^\top\left(Ay(t) - f\right)$$

Let $y(t)$ and $y^\delta(t)$ denote solutions of this flow for data f and f^δ with $\|f - f^\delta\| \leq \delta$ and identical initial values. For this model we can derive the following stability estimate:

$$\begin{aligned}\frac{d}{dt}\left(\frac{1}{2}\|y(t) - y^\delta(t)\|^2\right) &= -\lambda\left\langle A(y(t) - y^\delta(t)) - (f - f^\delta), A(y(t) - y^\delta(t))\right\rangle \\ &\leq \lambda\left\langle f - f^\delta, A(y(t) - y^\delta(t))\right\rangle \\ &\leq \lambda\|f - f^\delta\|\,\|A(y(t) - y^\delta(t))\|\end{aligned}$$

Regularisation properties

Connection to inverse problems for example with additional reaction term:

$$\frac{d}{dt}y(t) = -K(t)^\top \sigma\left(K(t)y(t) + \beta(t)\right) - \lambda A^\top\left(Ay(t) - f\right)$$

Let $y(t)$ and $y^\delta(t)$ denote solutions of this flow for data f and f^δ with $\|f - f^\delta\| \leq \delta$ and identical initial values. For this model we can derive the following stability estimate:

$$\frac{d}{dt}\left(\frac{1}{2}\|y(t) - y^\delta(t)\|^2\right) \leq \lambda\|f - f^\delta\|\,\|A(y(t) - y^\delta(t))\|$$

$$\leq \lambda\delta\|A\|\,\|y(t) - y^\delta(t)\|$$

Regularisation properties

Connection to inverse problems for example with additional reaction term:

$$\frac{d}{dt}y(t) = -K(t)^\top \sigma\left(K(t)y(t) + \beta(t)\right) - \lambda A^\top\left(Ay(t) - f\right)$$

Let $y(t)$ and $y^\delta(t)$ denote solutions of this flow for data f and f^δ with $\|f - f^\delta\| \leq \delta$ and identical initial values. For this model we can derive the following stability estimate:

$$\frac{d}{dt}\left(\frac{1}{2}\|y(t) - y^\delta(t)\|^2\right) \leq \lambda\delta\|A\|\|y(t) - y^\delta(t)\|$$

$$\Longrightarrow \quad \|y(t) - y^\delta(t)\| \leq \|y(0) - y^\delta(0)\| + \lambda\delta\|A\|\, t$$

$$= \lambda\|A\| t\, \delta$$

Regularisation properties

Connection to inverse problems for example with additional reaction term:

$$\frac{d}{dt}y(t) = -K(t)^\top \sigma\left(K(t)y(t) + \beta(t)\right) - \lambda A^\top\left(Ay(t) - f\right)$$

Let $y(t)$ and $y^\delta(t)$ denote solutions of this flow for data f and f^δ with $\|f - f^\delta\| \leq \delta$ and identical initial values. For this model we can derive the following stability estimate:

$$\|y(t) - y^\delta(t)\| \leq \lambda \|A\| t\, \delta\,.$$

Hence, we need to choose a stopping time t^* to guarantee convergence of this stability estimate, i.e. $\lim_{\delta \downarrow 0} \|y(t) - y^\delta(t)\| = 0.$

Regularisation properties

Connection to inverse problems for example with additional reaction term:

$$\frac{d}{dt}y(t) = -K(t)^\top \sigma\left(K(t)y(t) + \beta(t)\right) - \lambda A^\top\left(Ay(t) - f\right)$$

When do we converge to a solution $y^\dagger$ of the inverse problem, i.e. $Ay^\dagger = f$?

$$\begin{aligned}\frac{d}{dt}\left(\frac{1}{2}\|y^\delta(t) - y^\dagger\|^2\right) &= \left\langle \frac{d}{dt}y^\delta(t), y^\delta(t) - y^\dagger \right\rangle \\ &= -\left\langle K(t)^\top \sigma\left(K(t)y^\delta(t) + \beta(t)\right) + \lambda A^\top\left(Ay^\delta(t) - f^\delta\right), y(t) - y^\dagger \right\rangle \\ &= \underbrace{-\left\langle K(t)^\top \sigma\left(K(t)y^\delta(t) + \beta(t)\right), y(t) - y^\dagger \right\rangle}_{\text{first inner product}} \underbrace{-\lambda\left\langle A^\top\left(Ay^\delta(t) - f^\delta\right), y(t) - y^\dagger \right\rangle}_{\text{second inner product}}\end{aligned}$$

Regularisation properties

When do we converge to a solution $y^\dagger$ of the inverse problem, i.e. $Ay^\dagger = f$?

$$\underbrace{-\left\langle K(t)^\top \sigma\left(K(t)y^\delta(t) + \beta(t)\right), y(t) - y^\dagger \right\rangle}_{\text{first inner product}}$$

$$\begin{aligned} &= -\left\langle K(t)^\top \sigma\left(K(t)y^\delta(t) + \beta(t)\right) - K(t)^\top \sigma\left(K(t)y^\dagger + \beta(t)\right), y(t) - y^\dagger \right\rangle \\ &\quad + \left\langle K(t)^\top \sigma\left(K(t)y^\dagger + \beta(t)\right), y(t) - y^\dagger \right\rangle \\ &\leq \left\langle K(t)^\top \sigma\left(K(t)y^\dagger + \beta(t)\right), y(t) - y^\dagger \right\rangle \end{aligned}$$

↑ Monotonic increase of σ

Regularisation properties

When do we converge to a solution $y^\dagger$ of the inverse problem, i.e. $Ay^\dagger = f$?

$$\underbrace{-\left\langle K(t)^\top \sigma\left(K(t)y^\delta(t) + \beta(t)\right), y(t) - y^\dagger \right\rangle}_{\text{first inner product}}$$

$$\leq \quad \left\langle K(t)^\top \sigma\left(K(t)y^\dagger + \beta(t)\right), y(t) - y^\dagger \right\rangle$$

$$\leq \quad \left\| K(t)^\top \sigma\left(K(t)y^\dagger + \beta(t)\right) \right\| \quad \left\| y(t) - y^\dagger \right\|$$

Regularisation properties

Connection to inverse problems for example with additional reaction term:

$$\frac{d}{dt}y(t) = -K(t)^\top \sigma\left(K(t)y(t)+\beta(t)\right) - \lambda A^\top\left(Ay(t)-f\right)$$

When do we converge to a solution $y^\dagger$ of the inverse problem, i.e. $Ay^\dagger = f$?

$$\begin{aligned}\frac{d}{dt}\left(\frac{1}{2}\|y^\delta(t)-y^\dagger\|^2\right) &= \left\langle \frac{d}{dt}y^\delta(t), y^\delta(t)-y^\dagger\right\rangle \\ &= -\left\langle K(t)^\top\sigma\left(K(t)y^\delta(t)+\beta(t)\right) + \lambda A^\top\left(Ay^\delta(t)-f^\delta\right), y(t)-y^\dagger\right\rangle \\ \leq \left\| K(t)^\top\sigma\left(K(t)y^\dagger+\beta(t)\right)\right\| &\left\| y(t)-y^\dagger\right\| \underbrace{-\lambda\left\langle A^\top\left(Ay^\delta(t)-f^\delta\right), y(t)-y^\dagger\right\rangle}_{\text{second inner product}}\end{aligned}$$

Regularisation properties

When do we converge to a solution $y^\dagger$ of the inverse problem, i.e. $Ay^\dagger = f$?

$$\underbrace{-\lambda \left\langle A^\top \left(Ay^\delta(t) - f^\delta\right), y(t) - y^\dagger \right\rangle}_{\text{second inner product}}$$

$$\begin{aligned}
&= -\lambda \left\langle Ay^\delta(t) - f^\delta, Ay(t) - Ay^\dagger \right\rangle = -\lambda \left\langle Ay^\delta(t) - f^\delta, Ay(t) - f \right\rangle \\
&= -\lambda \left\langle Ay^\delta(t) - f + f - f^\delta, Ay(t) - f \right\rangle \\
&= -\lambda \left\| Ay^\delta(t) - f \right\|^2 - \lambda \left\langle f - f^\delta, Ay^\delta(t) - f \right\rangle \\
&\leq \lambda \left\| f - f^\delta \right\| \left\| Ay^\delta(t) - f \right\| \leq \lambda\delta\|A\| \left\| y^\delta(t) - y^\dagger \right\|
\end{aligned}$$

Regularisation properties

Connection to inverse problems for example with additional reaction term:

$$\frac{d}{dt}y(t) = -K(t)^\top \sigma\left(K(t)y(t)+\beta(t)\right) - \lambda A^\top\left(Ay(t)-f\right)$$

When do we converge to a solution $y^\dagger$ of the inverse problem, i.e. $Ay^\dagger = f$?

$$\begin{aligned}
\frac{d}{dt}\left(\frac{1}{2}\|y^\delta(t)-y^\dagger\|^2\right) &= \left\langle \frac{d}{dt}y^\delta(t), y^\delta(t)-y^\dagger \right\rangle \\
&= -\left\langle K(t)^\top \sigma\left(K(t)y^\delta(t)+\beta(t)\right) + \lambda A^\top\left(Ay^\delta(t)-f^\delta\right), y(t)-y^\dagger \right\rangle \\
&\leq \left\| K(t)^\top \sigma\left(K(t)y^\dagger+\beta(t)\right)\right\| \left\| y(t)-y^\dagger\right\| + \lambda\delta\|A\| \left\| y^\delta(t)-y^\dagger\right\| \\
&= \left(\left\| K(t)^\top \sigma\left(K(t)y^\dagger+\beta(t)\right)\right\| + \lambda\delta\|A\|\right) \left\| y^\delta(t)-y^\dagger\right\|
\end{aligned}$$

Regularisation properties

Connection to inverse problems for example with additional reaction term:

$$\frac{d}{dt}y(t) = -K(t)^\top \sigma\left(K(t)y(t) + \beta(t)\right) - \lambda A^\top\left(Ay(t) - f\right)$$

When do we converge to a solution $y^\dagger$ of the inverse problem, i.e. $Ay^\dagger = f$?

$$\frac{d}{dt}\left(\frac{1}{2}\|y^\delta(t) - y^\dagger\|^2\right) \leq \left(\left\| K(t)^\top \sigma\left(K(t)y^\dagger + \beta(t)\right)\right\| + \lambda\delta\|A\|\right)\left\| y^\delta(t) - y^\dagger\right\|$$

Assumption (source condition): $\|K(t)^\top \sigma\left(K(t)y^\dagger + \beta(t)\right)\|$ is bounded for all $t \geq 0$.

$$\implies \frac{d}{dt}\left(\frac{1}{2}\|y^\delta(t) - y^\dagger\|^2\right) \leq \left(C + \lambda\delta\|A\|\right)\left\| y^\delta(t) - y^\dagger\right\|$$

Regularisation properties

Connection to inverse problems for example with additional reaction term:

$$\frac{d}{dt}y(t) = -K(t)^\top \sigma\left(K(t)y(t) + \beta(t)\right) - \lambda A^\top\left(Ay(t) - f\right)$$

Assumption (source condition): $\|K(t)^\top \sigma\left(K(t)y^\dagger + \beta(t)\right)\|$ is bounded for all $t \geq 0$.

$$\frac{d}{dt}\left(\frac{1}{2}\|y^\delta(t) - y^\dagger\|^2\right) \leq \left(C + \lambda\delta\|A\|\right)\left\|y^\delta(t) - y^\dagger\right\|$$

$$\Longrightarrow \qquad \|y^\delta(t) - y^\dagger\| \leq \|y^\delta(0) - y^\dagger\| + \left(C + \lambda\delta\|A\|\right)t$$

Regularisation properties

Assumption (source condition): $\|K(t)^\top \sigma\left(K(t)y^\dagger + \beta(t)\right)\|$ is bounded for all $t \geq 0$.

$$\frac{d}{dt}\left(\frac{1}{2}\|y^\delta(t) - y^\dagger\|^2\right) \leq \left(C + \lambda\delta\|A\|\right) \left\| y^\delta(t) - y^\dagger \right\|$$

$$\implies \qquad \|y^\delta(t) - y^\dagger\| \leq \|y^\delta(0) - y^\dagger\| + \left(C + \lambda\delta\|A\|\right) t$$

Suppose $A = \mathrm{Id}$ and $y^\delta(0) = f^\delta$, then we could estimate

$$\|y^\delta(t) - y^\dagger\| \leq \|f^\delta - f\| + \left(C + \lambda\delta\|A\|\right) t$$

Regularisation properties

Assumption (source condition): $\|K(t)^\top \sigma\left(K(t)y^\dagger + \beta(t)\right)\|$ is bounded for all $t \geq 0$.

$$\frac{d}{dt}\left(\frac{1}{2}\|y^\delta(t) - y^\dagger\|^2\right) \leq \left(C + \lambda\delta\|A\|\right) \left\| y^\delta(t) - y^\dagger \right\|$$

$$\Longrightarrow \qquad \|y^\delta(t) - y^\dagger\| \leq \|y^\delta(0) - y^\dagger\| + \left(C + \lambda\delta\|A\|\right) t$$

Suppose $A = \mathrm{Id}$ and $y^\delta(0) = f^\delta$, then we could estimate

$$\|y^\delta(t) - y^\dagger\| \leq \quad \delta \quad + \left(C + \lambda\delta\|A\|\right) t$$

and conclude $\lim_{\delta \downarrow 0} \|y^\delta(t^*) - y^\dagger\| = 0$ for $t^* \sim \delta$.

Thank you for your attention!

Acknowledgements:

LEVERHULME TRUST

EPSRC
Engineering and Physical Sciences Research Council

NVIDIA

The Research Council of Norway

Horizon 2020
European Union Funding for Research & Innovation

CCIMI
CANTAB CAPITAL INSTITUTE FOR THE MATHEMATICS OF INFORMATION

The Alan Turing Institute