Neural ODEs and Control

March 14, 2023

References

Kidger, Patrick. "On neural differential equations." arXiv preprint arXiv:2202.02435 (2022).

Ruiz-Balet, Domenec, and Enrique Zuazua. "Neural ODE control for classification, approximation and transport." arXiv preprint arXiv:2104.05278 (2021).

$$y(0) = y_0 \qquad \frac{dy}{dt}(t) = f_\theta(t, y(t))$$

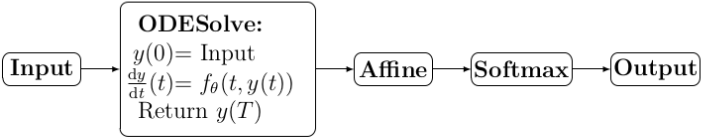
- $d = d_1 \times \cdots \times d_k$ and $f_\theta \colon \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}^d$ is (for example) a feedforward neural net
- θ learnable parameters
- existence/uniqueness of solutions?
- solutions are functions $y \colon [0, T] \to \mathbb{R}^d$

Simple example of a concrete architecture:

- inputs are small images with $d = 3 \times 32 \times 32$
- binary classification via learnable affine map $A_\theta \colon \mathbb{R}^d \to \mathbb{R}^2$ with softmax loss

Simple example of a concrete architecture:

- inputs are small images with $d = 3 \times 32 \times 32$
- binary classification via learnable affine map $A_\theta \colon \mathbb{R}^d \to \mathbb{R}^2$ with softmax loss



$$\operatorname{softmax}\left(A_\theta\left(y(0) + \int_0^T f_\theta(t, y(t))\, dt\right)\right)$$

More interesting example combining the modeling power of DEs with the high blackbox capacity of NNs $\rightarrow$ inductive bias:

Consider the *Lotka-Volterra equations* modeling prey-predator relations

$$\dot{x} = \alpha x - \beta xy \qquad \dot{y} = -\gamma x + \delta xy$$

More interesting example combining the modeling power of DEs with the high blackbox capacity of NNs $\rightarrow$ inductive bias:

Consider the *Lotka-Volterra equations* modeling prey-predator relations

$$\dot{x} = \alpha x - \beta xy \qquad \dot{y} = -\gamma x + \delta xy$$

Universal Differential Equation (UDE)

$$\dot{x} = \alpha x - \beta xy + f_\theta(x, y) \qquad \dot{y} = -\gamma x + \delta xy + g_\theta(x, y)$$

- f_θ and g_θ are ffnns (generally small)

More interesting example combining the modeling power of DEs with the high blackbox capacity of NNs $\rightarrow$ inductive bias:

Consider the *Lotka-Volterra equations* modeling prey-predator relations

$$\dot{x} = \alpha x - \beta xy \qquad \dot{y} = -\gamma x + \delta xy$$

Universal Differential Equation (UDE)

$$\dot{x} = \alpha x - \beta xy + f_\theta(x, y) \qquad \dot{y} = -\gamma x + \delta xy + g_\theta(x, y)$$

- f_θ and g_θ are ffnns (generally small)
- admit that there are effects that the model does not capture

More interesting example combining the modeling power of DEs with the high blackbox capacity of NNs $\rightarrow$ inductive bias:

Consider the *Lotka-Volterra equations* modeling prey-predator relations

$$\dot{x} = \alpha x - \beta xy \qquad \dot{y} = -\gamma x + \delta xy$$

Universal Differential Equation (UDE)

$$\dot{x} = \alpha x - \beta xy + f_\theta(x, y) \qquad \dot{y} = -\gamma x + \delta xy + g_\theta(x, y)$$

- f_θ and g_θ are ffnns (generally small)
- admit that there are effects that the model does not capture
- favorable to maintain interpretability during training

Interpretation as *continuous-depth* neural nets.

Start with a neural ODE of the form

$$\frac{dy}{dt}(t) = f_\theta(t, y(t))$$

Interpretation as *continuous-depth* neural nets.

Start with a neural ODE of the form

$$\frac{dy}{dt}(t) = f_\theta(t, y(t))$$

Discretize: explicit Euler at times t_j with uniform separation distance Δt

$$\frac{y(t_{j+1}) - y(t_j)}{\Delta t} \approx \frac{dy}{dt}(t_j) = f_\theta(t_j, y(t_j))$$

Interpretation as *continuous-depth* neural nets.

Start with a neural ODE of the form

$$\frac{dy}{dt}(t) = f_\theta(t, y(t))$$

Discretize: explicit Euler at times t_j with uniform separation distance Δt

$$\frac{y(t_{j+1}) - y(t_j)}{\Delta t} \approx \frac{dy}{dt}(t_j) = f_\theta(t_j, y(t_j))$$

i.e.

$$y(t_{j+1}) = y(t_j) + \Delta t \cdot f_\theta(t_j, y(t_j)) = y(t_j) + \widetilde{f}_\theta(t_j, y(t_j))$$

Interpretation as *continuous-depth* neural nets.

Start with a neural ODE of the form

$$\frac{dy}{dt}(t) = f_\theta(t, y(t))$$

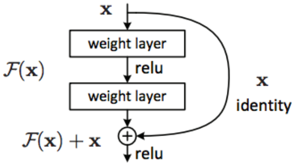
Discretize: explicit Euler at times t_j with uniform separation distance Δt

$$\frac{y(t_{j+1}) - y(t_j)}{\Delta t} \approx \frac{dy}{dt}(t_j) = f_\theta(t_j, y(t_j))$$

i.e.

$$y(t_{j+1}) = y(t_j) + \Delta t \cdot f_\theta(t_j, y(t_j)) = y(t_j) + \widetilde{f_\theta}(t_j, y(t_j))$$

corresponds to the forward equations of *ResNet* blocks



Allows for an interpretation as optimal control problem. Specify the neural ResNet:

$$\theta = \{W(t), A(t), b(t)\}$$

$$y(0) = y_0 \qquad \frac{dy}{dt}(t) = W(t)\ \mathrm{ReLU}(A(t)y(t) + b(t))$$

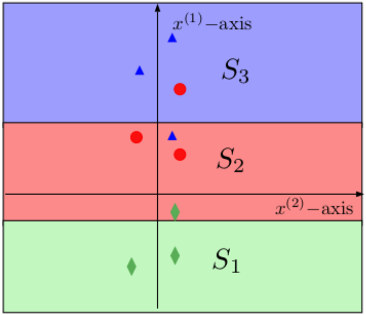
Allows for an interpretation as optimal control problem. Specify the neural ResNet:

$$\theta = \{W(t), A(t), b(t)\}$$

$$y(0) = y_0 \qquad \frac{dy}{dt}(t) = W(t)\ \mathrm{ReLU}(A(t)y(t) + b(t))$$

- parameters act as controls $W(t), A(t) \in L^\infty((0,T); R^{d\times d})$, and $b(t) \in L^\infty((0,T); R^d)$
- reformulate classification problem

- split R^d into horizontal strips corresponding to classes $S_1, \ldots, S_m$



- training goal: find controls that steer the training examples into the right strip

Theorem (Zuazua and Ruiz-Balet, 2021)

Let $\{y_i, z_i\}_{i=1}^N \subset \mathbb{R}^d \times \mathbb{R}^d$ *be the dataset to be classified with* $y_i \neq y_j$ *and* $z_i \in S_{m(i)}$.
Then for every $T > 0$ *there exist piecewise constant controls with* $O(N)$ *switches such that*

$$\forall i : \quad y(0) = y_i \implies y(T) \in S_{m(i)}.$$

Sketch of proof

Fully constructive.

1. Preparation: apply simple flows to separate the data w.r.t. at least one cartesian axis: $y_i^{(1)} \neq y_j^{(1)}$

Sketch of proof

Fully constructive.

1. Preparation: apply simple flows to separate the data w.r.t. at least one cartesian axis: $y_i^{(1)} \neq y_j^{(1)}$
2. Classification: apply atomic moves (next slide) to steer into the right strip

Sketch of proof

Fully constructive.

1. Preparation: apply simple flows to separate the data w.r.t. at least one cartesian axis: $y_i^{(1)} \neq y_j^{(1)}$
2. Classification: apply atomic moves (next slide) to steer into the right strip
3. Time rescaling: above construction has finite-time $\widetilde{T}$... simple rescaling argument to reach required T

Sketch of proof

Fully constructive.

1. Preparation: apply simple flows to separate the data w.r.t. at least one cartesian axis: $y_i^{(1)} \neq y_j^{(1)}$
2. Classification: apply atomic moves (next slide) to steer into the right strip
3. Time rescaling: above construction has finite-time $\widetilde{T}$... simple rescaling argument to reach required T

Preparation requires at most N switches, classification at most $3N$.

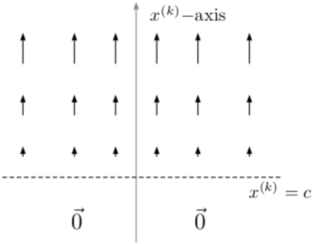
Atomic moves

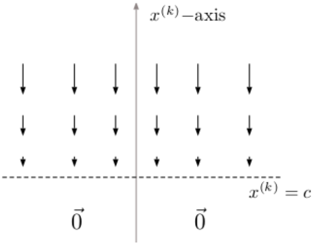
Place a hyperplane $\{y^{(k)} - c = 0\}$ decide which of the two half-spaces will be frozen (left side of ReLU) and which will be active (right side of ReLU).

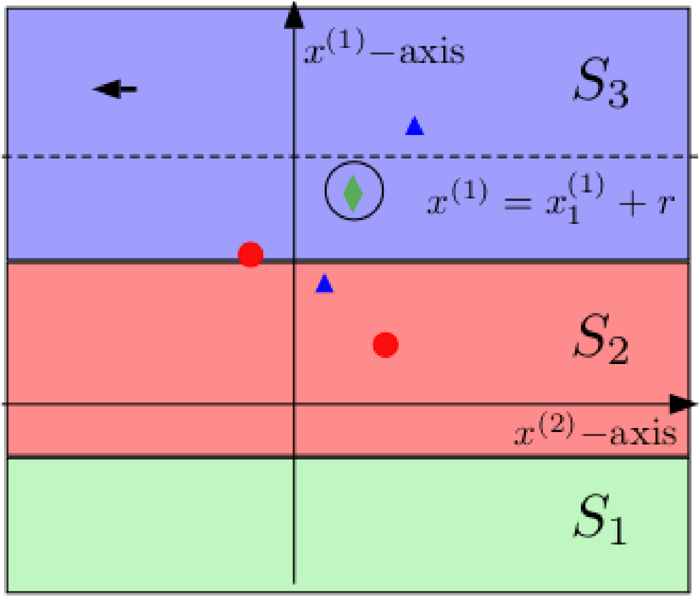
Atomic moves

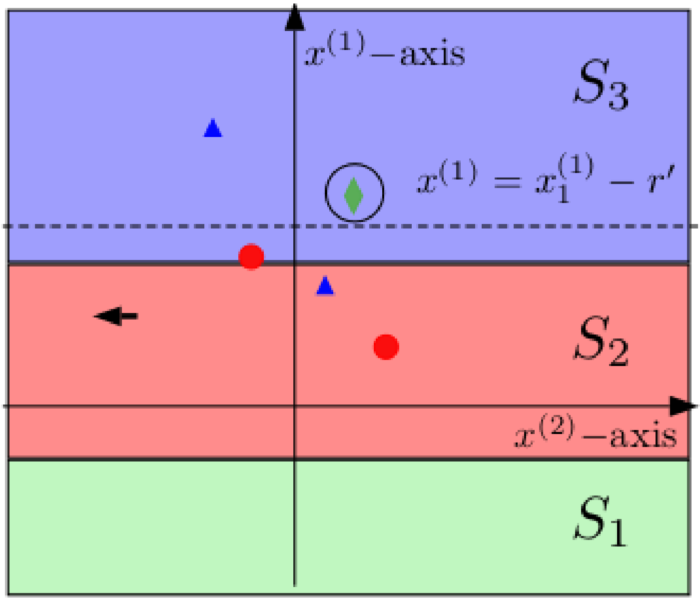
Place a hyperplane $\{y^{(k)} - c = 0\}$ decide which of the two half-spaces will be frozen (left side of ReLU) and which will be active (right side of ReLU).

- **expand/contract**: see pictures. decide whether the hyperplane will be repulsive or attractive
- **translate**: flow parallel to the hyperplane

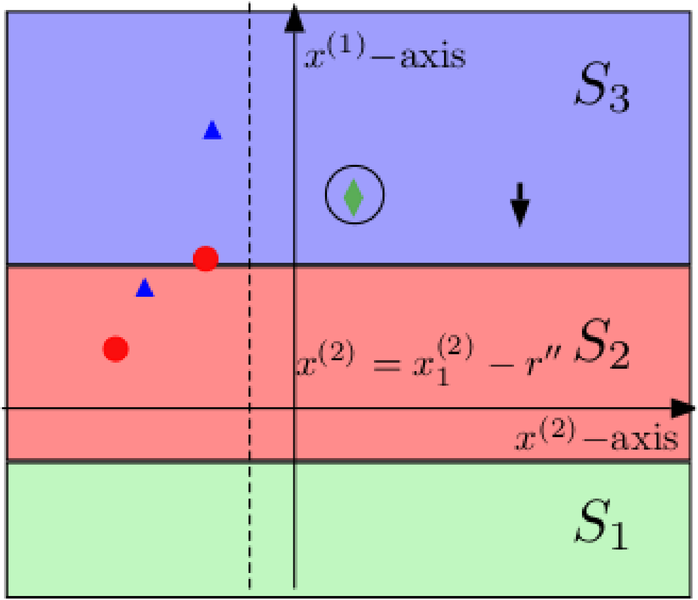




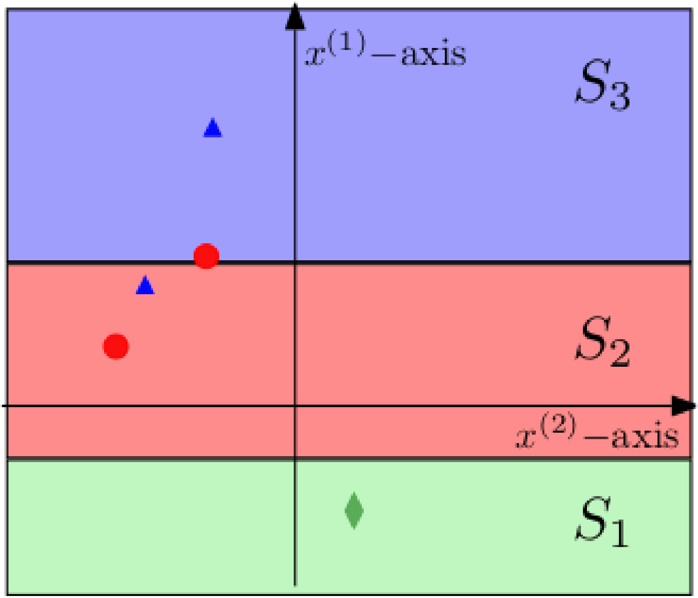

$x^{(1)}-$axis
S_3
$x^{(1)}=x_1^{(1)}+r$
S_2
$x^{(2)}-$axis
S_1



$x^{(1)}$−axis
S_3
$x^{(1)} = x_1^{(1)} - r'$
S_2
$x^{(2)}$−axis
S_1



$x^{(1)}$−axis
S_3
$x^{(2)} = x_1^{(2)} - r''$
S_2
$x^{(2)}$−axis
S_1


$x^{(1)}$−axis
S_3
S_2
$x^{(2)}$−axis
S_1

Some remarks:

- construction needs a half-vanishing activation

Some remarks:

- construction needs a half-vanishing activation
- an instance of simultaneous control: one control to steer all examples into their strips
    - easy extension: steer the initial points y_i (approximately) to their exact targets z_i (instead of their strips)

Some remarks:

- construction needs a half-vanishing activation
- an instance of simultaneous control: one control to steer all examples into their strips
  - easy extension: steer the initial points y_i (approximately) to their exact targets z_i (instead of their strips)
- in the continuous-depth interpretation: dynamically adding layers during training

Some remarks:

- construction needs a half-vanishing activation
- an instance of simultaneous control: one control to steer all examples into their strips
  - easy extension: steer the initial points y_i (approximately) to their exact targets z_i (instead of their strips)
- in the continuous-depth interpretation: dynamically adding layers during training
- regularization = fewer moves with lower amplitude? generalization bounds?