Neural Controlled Differential Equations

Mathematical Institute

Patrick Kidger

Mathematical Institute
University of Oxford

Unlocking data streams, March 2021

Oxford Mathematics

Overview

Di erential equations have arguably been the dominant modelling paradigm for centuries.

Overview

Di erential equations have arguably been the dominant modelling paradigm for centuries.

Neural networks have more recently joined them as another dominant modelling paradigm.

Overview

Di erential equations have arguably been the dominant modelling paradigm for centuries.

Neural networks have more recently joined them as another dominant modelling paradigm.

Neural di erential equations:

Overview

Di erential equations have arguably been the dominant modelling paradigm for centuries.

Neural networks have more recently joined them as another dominant modelling paradigm.

Neural di erential equations: best of both worlds. Memory e ciency,

Overview

Di erential equations have arguably been the dominant modelling paradigm for centuries.

Neural networks have more recently joined them as another dominant modelling paradigm.

Neural di erential equations: best of both worlds. Memory e ciency, strong priors on model space,

Overview

Di erential equations have arguably been the dominant modelling paradigm for centuries.

Neural networks have more recently joined them as another dominant modelling paradigm.

Neural di erential equations: best of both worlds. Memory e ciency, strong priors on model space, high capacity function approximation,

Overview

Di erential equations have arguably been the dominant modelling paradigm for centuries.

Neural networks have more recently joined them as another dominant modelling paradigm.

Neural di erential equations: best of both worlds. Memory e ciency, strong priors on model space, high capacity function approximation, draw on a deep well of theory on both sides.

Overview

Di erential equations have arguably been the dominant modelling paradigm for centuries.

Neural networks have more recently joined them as another dominant modelling paradigm.

Neural di erential equations: best of both worlds. Memory e ciency, strong priors on model space, high capacity function approximation, draw on a deep well of theory on both sides.

Applications to traditional mathematical modelling (nance, physics, engineering, . . .),

Overview

Di erential equations have arguably been the dominant modelling paradigm for centuries.

Neural networks have more recently joined them as another dominant modelling paradigm.

Neural di erential equations: best of both worlds. Memory e ciency, strong priors on model space, high capacity function approximation, draw on a deep well of theory on both sides.

Applications to traditional mathematical modelling (nance, physics, engineering, ...), and to modern machine learning (time series, generative modelling).

Neural di erential equations

Programme

Neural di erential equations

Intuition

Programme

Neural di erential equations

- Intuition
- Applications

Programme

- Neural differential equations
  - Intuition
  - Applications
- Neural controlled differential equations

Programme

- Neural differential equations
  - Intuition
  - Applications
- Neural controlled differential equations
  - Continuous time recurrent neural networks

Programme

- Neural di erential equations
  - Intuition
  - Applications
- Neural controlled di erential equations
  - Continuous time recurrent neural networks
  - Irregular time series

Programme

- Neural di erential equations
  - Intuition
  - Applications
- Neural controlled di erential equations
  - Continuous time recurrent neural networks
  - Irregular time series
- Neural stochastic di erential equations

Programme

- Neural di erential equations
  - Intuition
  - Applications
- Neural controlled di erential equations
  - Continuous time recurrent neural networks
  - Irregular time series
- Neural stochastic di erential equations
- Software

What is a neural di erential equation anyway?

(And why you might already be using them.)

These are di erential equations for which the vector eld is parameterised as a neural network.

What is a neural di erential equation anyway?

(And why you might already be using them.)

These are di erential equations for which the vector eld is parameterised as a neural network.

Canonical example - neural ordinary di erential equation (Chen et al. 2018):

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

What is a neural di erential equation anyway?

(And why you might already be using them.)

These are di erential equations for which the vector eld is parameterised as a neural network.

Canonical example - neural ordinary di erential equation (Chen et al. 2018):

$$y(0) = y_0 \qquad \frac{dy}{dt}(t) = f(t; y(t))$$

f is any standard network - for our purposes today, often a feedforward network:

What is a neural di erential equation anyway?

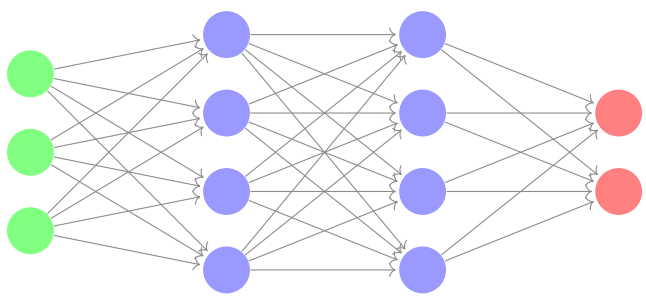
(And why you might already be using them.)

These are di erential equations for which the vector eld is parameterised as a neural network.

Canonical example - neural ordinary di erential equation (Chen et al. 2018):

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

f_θ is any standard network - for our purposes today, often a feedforward network:



What is a neural di erential equation anyway?

(And why you might already be using them.)

Classical example of a `neural' di erential equation: the SIR model.

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} s(t) \\ i(t) \\ r(t) \end{pmatrix} = \begin{pmatrix} -\beta s(t) i(t) \\ \beta s(t) i(t) - \kappa i(t) \\ \kappa i(t) \end{pmatrix}$$

What is a neural di erential equation anyway?

(And why you might already be using them.)

Classical example of a `neural' di erential equation: the SIR model.

$$\frac{d}{dt}\begin{pmatrix} s(t) \\ i(t) \\ r(t) \end{pmatrix} = \begin{pmatrix} b\,s(t)\,i(t) \\ b\,s(t)\,i(t) \quad k\,i(t) \\ k\,i(t) \end{pmatrix}$$

b and k are parameters, controlling infectivity and removal rates respectively.

What is a neural di erential equation anyway?

(And why you might already be using them.)

Classical example of a `neural' di erential equation: the SIR model.

$$\frac{d}{dt}\begin{pmatrix} s(t) \\ i(t) \\ r(t) \end{pmatrix} = \begin{pmatrix} b\,s(t)\,i(t) \\ b\,s(t)\,i(t) \quad k\,i(t) \\ k\,i(t) \end{pmatrix}$$

b and k are parameters, controlling infectivity and removal rates respectively.

Crucially for our purposes: they are parameters learnt from data.

What is a neural di erential equation anyway?

(And why you might already be using them.)

Classical example of a `neural' di erential equation: the SIR model.

$$\frac{d}{dt}\begin{pmatrix} s(t) \\ i(t) \\ r(t) \end{pmatrix} = \begin{pmatrix} b\,s(t)\,i(t) \\ b\,s(t)\,i(t) \quad k\,i(t) \\ k\,i(t) \end{pmatrix}$$

b and k are parameters, controlling infectivity and removal rates respectively.

Crucially for our purposes: they are parameters learnt from data.

What is a neural di erential equation anyway?

(And why you might already be using them.)

Another example: the Heston model.

$$dS_t = \quad S_t\,dt + \sqrt{\quad_t}S_t\,dW_t^1$$
$$d\ _t = (\qquad\ _t)\,dt + \sqrt{\quad_t}\,dW_t^2$$

What is a neural di erential equation anyway?

(And why you might already be using them.)

Another example: the Heston model.

$$dS_t = \quad S_t\,dt + \sqrt{\quad_t}S_t\,dW_t^1$$
$$d\ _t = (\qquad\ _t)\,dt + \sqrt{\quad_t}\,dW_t^2$$

What is a neural di erential equation anyway?

(And why you might already be using them.)

Consider solving

$$\frac{dy}{dt}(t) = f(t; y(t))$$

with the explicit Euler method.

What is a neural di erential equation anyway?

(And why you might already be using them.)

Consider solving

$$\frac{dy}{dt}(t) = f(t; y(t))$$

with the explicit Euler method. Approximate

$$\frac{y_{t_{n+1}} \quad y_{t_n}}{t} \qquad \frac{dy}{dt}$$

What is a neural di erential equation anyway?

(And why you might already be using them.)

Consider solving

$$\frac{dy}{dt}(t) = f(t; y(t))$$

with the explicit Euler method. Approximate

$$\frac{y_{t_{n+1}} \quad y_{t_n}}{t} \quad \frac{dy}{dt}$$

which rearranges to

$$y_{t_{n+1}} = y_{t_n} + \quad t\, f(t; y_{t_n}):$$

This is now the same as a residual network, one of the main neural network architectures.

Bene ts

Bene ts

From traditional mathematical modelling:

Bene ts

From traditional mathematical modelling: neural networks are very high capacity models

Bene ts

From traditional mathematical modelling: neural networks are very high capacity models, easy to train

Bene ts

From traditional mathematical modelling: neural networks are very high capacity models, easy to train, the NDE formalism means they easily augment traditional techniques.

Bene ts

From traditional mathematical modelling: neural networks are very high capacity models, easy to train, the NDE formalism means they easily augment traditional techniques.

From machine learning:

Bene ts

From traditional mathematical modelling: neural networks are very high capacity models, easy to train; the NDE formalism means they easily augment traditional techniques.

From machine learning: di erential equations are good priors on model space

Bene ts

From traditional mathematical modelling: neural networks are very high capacity models, easy to train; the NDE formalism means they easily augment traditional techniques.

From machine learning: di erential equations are good priors on model space, able to handle irregular data

Bene ts

From traditional mathematical modelling: neural networks are very high capacity models, easy to train; the NDE formalism means they easily augment traditional techniques.

From machine learning: di erential equations are good priors on model space, able to handle irregular data, memory e cient

Bene ts

From traditional mathematical modelling: neural networks are very high capacity models, easy to train; the NDE formalism means they easily augment traditional techniques.

From machine learning: di erential equations are good priors on model space, able to handle irregular data, memory e cient, give theoretical grounding to many neural networks.

Applications

Neural di erential equations have (to my knowledge) three main uses:

Applications

Neural di erential equations have (to my knowledge) three main uses:

- Physical (nancial, . . .) modelling, for which a di erential equation is explicitly desired.

Applications

Neural di erential equations have (to my knowledge) three main uses:

- Physical (nancial, . . .) modelling, for which a di erential equation is explicitly desired.
- Generative modelling: continuous normalising ows; neural SDEs.

Applications

Neural di erential equations have (to my knowledge) three main uses:

- Physical (nancial, . . .) modelling, for which a di erential equation is explicitly desired.
- Generative modelling: continuous normalising ows; neural SDEs.
- Time series applications: data may arrive irregularly sampled, partially observed and so on.

Neural controlled di erential equations

Neural controlled di erential equations

Consider the (neural) ODE

$$y(t) = y(0) + \int_0^t f(y(s))\, ds \quad \text{for } t \in [0; T]:$$

Neural controlled di erential equations

Consider the (neural) ODE

$$y(t) = y(0) + \int_0^t f(y(s))\, ds \quad \text{for } t \in [0; T]:$$

The integral over $s \in [0; T]$ is introduced and then integrated over, and is just an internal detail of the model.

Neural controlled di erential equations

Consider the (neural) ODE

$$y(t) = y(0) + \int_0^t f(y(s))\,ds \quad \text{for } t \in [0, T]:$$

The integral over $s \in [0, T]$ is introduced and then integrated over, and is just an internal detail of the model.

But given some ordered data $x_0, \ldots, x_n$, we would like to align $s \in [0, T]$ with the ordering of the data.

Neural controlled di erential equations

Consider the (neural) ODE

$$y(t) = y(0) + \int_0^t f(y(s))\,ds \quad \text{for } t \in [0, T]:$$

The integral over $s \in [0, T]$ is introduced and then integrated over, and is just an internal detail of the model.

But given some ordered data $x_0, \ldots, x_n$, we would like to align $s \in [0, T]$ with the ordering of the data.

Problem: it's an ODE. The solution is determined by its initial condition.

Neural controlled di erential equations

Consider the (neural) ODE

$$y(t) = y(0) + \int_0^t f(y(s))\,ds \quad \text{for } t \in [0, T]:$$

The integral over $s \in [0, T]$ is introduced and then integrated over, and is just an internal detail of the model.

But given some ordered data $x_0, \ldots, x_n$, we would like to align $s \in [0, T]$ with the ordering of the data.

Problem: it's an ODE. The solution is determined by its initial condition.

Want a way to incorporate incoming information.

Neural controlled di erential equations

Neural controlled di erential equations

(Lyons 1998)

Neural controlled di erential equations

(Lyons 1998)

Have the local dynamics of the system depend upon some time-varying $X : [0; T] ! \quad R^{v}$:

Neural controlled di erential equations

(Lyons 1998)

Have the local dynamics of the system depend upon some time-varying $X : [0; T] ! \quad R^v$:

$$y(t) = y(0) + \int_0^t f(y(s))\, dX(s) \quad \text{for } t \ 2 \ [0; T]:$$

Neural controlled di erential equations

(Lyons 1998)

Have the local dynamics of the system depend upon some time-varying $X : [0; T] ! \quad R^v$:

$$y(t) = y(0) + \int_0^t f(y(s))\, dX(s) \quad \text{for } t \ 2 \ [0; T]:$$

which is a Riemann{Stieltjes integral.

Neural controlled di erential equations

(Lyons 1998)

Have the local dynamics of the system depend upon some time-varying $X : [0; T] ! \quad R^v$:

$$y(t) = y(0) + \int_0^t f(y(s))\, dX(s) \quad \text{for } t \ 2 \ [0; T]:$$

which is a Riemann{Stieltjes integral. Intuitively similar to

$$y(t) = y(0) + \int_0^t f(y(s); X(s))\, ds \quad \text{for } t \ 2 \ [0; T]:$$

Neural controlled di erential equations

(Lyons 1998)

Have the local dynamics of the system depend upon some time-varying $X : [0; T] \to \mathbb{R}^v$:

$$y(t) = y(0) + \int_0^t f(y(s))\, dX(s) \quad \text{for } t \in [0; T]:$$

which is a Riemann{Stieltjes integral. Intuitively similar to

$$y(t) = y(0) + \int_0^t f(y(s); X(s))\, ds \quad \text{for } t \in [0; T]:$$

(The rst equation ends up being slightly more exible, and a bit mathematically neater.)

Neural controlled di erential equations

(Lyons 1998)

Have the local dynamics of the system depend upon some time-varying $X : [0; T] \to R^v$:

$$y(t) = y(0) + \int_0^t f(y(s))\, dX(s) \quad \text{for } t \in [0; T]:$$

which is a Riemann{Stieltjes integral. Intuitively similar to

$$y(t) = y(0) + \int_0^t f(y(s); X(s))\, ds \quad \text{for } t \in [0; T]:$$

(The rst equation ends up being slightly more exible, and a bit mathematically neater.)

Changes in X provoke changes in y.

Neural controlled di erential equations

Irregular time series

Neural controlled di erential equations

Irregular time series

Observe $x = ((t_0; x_0); : : : ; (t_n; x_n))$ with $t_i \ 2 \ R$ and $x_i \ 2 \ R^v$.

Neural controlled di erential equations

Irregular time series

Observe $x = ((t_0; x_0); : : : ; (t_n; x_n))$ with $t_i \ 2 \ R$ and $x_i \ 2 \ R^v$.
Let $X : [t_0; t_n] \ ! \ R^{v+1}$ interpolate this data, so $X(t_i) = (t_i; x_i)$.

Neural controlled di erential equations

Irregular time series

Observe $x = ((t_0; x_0); : : : ; (t_n; x_n))$ with $t_i \ 2 \ R$ and $x_i \ 2 \ R^v$.
Let $X : [t_0; t_n] \ ! \ R^{v+1}$ interpolate this data, so $X(t_i) = (t_i ; x_i)$.

Learn functions , f and a linear map` such that

Neural controlled di erential equations

Irregular time series

Observe $x = ((t_0; x_0); : : : ; (t_n; x_n))$ with $t_i \ 2 \ R$ and $x_i \ 2 \ R^v$.
Let $X : [t_0; t_n] \ ! \ R^{v+1}$ interpolate this data, so $X(t_i) = (t_i ; x_i)$.

Learn functions , f and a linear map` such that

$$y(t_0) = \quad (t_0; x_0);$$

Neural controlled di erential equations

Irregular time series

Observe $x = ((t_0; x_0); : : : ; (t_n; x_n))$ with $t_i \ 2 \ R$ and $x_i \ 2 \ R^v$.
Let $X : [t_0; t_n] \ ! \ R^{v+1}$ interpolate this data, so $X(t_i) = (t_i ; x_i)$.

Learn functions $, f$ and a linear map $`$ such that

$$y(t_0) = \quad (t_0; x_0); \quad y(t) = y(t_0) + \int_{t_0}^{t} f\,(y(s))\, dX(s);$$

Neural controlled di erential equations

Irregular time series

Observe $x = ((t_0; x_0); : : : ; (t_n; x_n))$ with $t_i \ 2 \ R$ and $x_i \ 2 \ R^v$.
Let $X : [t_0; t_n] \ ! \ R^{v+1}$ interpolate this data, so $X(t_i) = (t_i ; x_i)$.

Learn functions $, f$ and a linear map $`$ such that

$$y(t_0) = \quad (t_0; x_0); \quad y(t) = y(t_0) + \int_{t_0}^{t} f\,(y(s))\, dX(s);$$

and the output is either $`\,(y(t_n))$ or $`\,(y(t))$.

Neural controlled di erential equations

Irregular time series

Observe $x = ((t_0, x_0), \ldots, (t_n, x_n))$ with $t_i \in \mathbb{R}$ and $x_i \in \mathbb{R}^v$.
Let $X : [t_0, t_n] \to \mathbb{R}^{v+1}$ interpolate this data, so $X(t_i) = (t_i, x_i)$.

Learn functions ζ, f and a linear map ℓ such that

$$y(t_0) = \zeta(t_0, x_0), \quad y(t) = y(t_0) + \int_{t_0}^{t} f(y(s))\, dX(s),$$

and the output is either $\ell(y(t_n))$ or $\ell(y(t))$.

ζ and f are arbitrary neural networks (feed-forward, ...), and y is hidden state.

Neural controlled di erential equations

Irregular time series

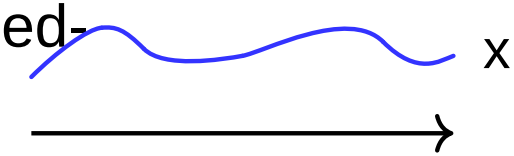
Observe $x = ((t_0, x_0), \ldots, (t_n, x_n))$ with $t_i \in \mathbb{R}$ and $x_i \in \mathbb{R}^v$.
Let $X : [t_0, t_n] \to \mathbb{R}^{v+1}$ interpolate this data, so $X(t_i) = (t_i, x_i)$.

Learn functions ζ, f and a linear map ℓ such that

$$y(t_0) = \zeta(t_0, x_0); \quad y(t) = y(t_0) + \int_{t_0}^{t} f(y(s))\, dX(s);$$

and the output is either $\ell(y(t_n))$ or $\ell(y(t))$.

ζ and f are arbitrary neural networks (feed-forward, ...), and y is hidden state.



Neural controlled di erential equations

Irregular time series

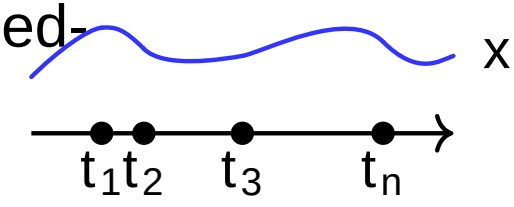
Observe $x = ((t_0; x_0); : : : ; (t_n; x_n))$ with $t_i \in \mathbb{R}$ and $x_i \in \mathbb{R}^v$.
Let $X : [t_0; t_n] \to \mathbb{R}^{v+1}$ interpolate this data, so $X(t_i) = (t_i; x_i)$.

Learn functions ζ, f and a linear map ℓ such that

$$y(t_0) = \zeta(t_0; x_0); \quad y(t) = y(t_0) + \int_{t_0}^{t} f(y(s))\, dX(s);$$

and the output is either $\ell(y(t_n))$ or $\ell(y(t))$.

ζ and f are arbitrary neural networks (feed-forward, ...), and y is hidden state.



Neural controlled di erential equations

Irregular time series

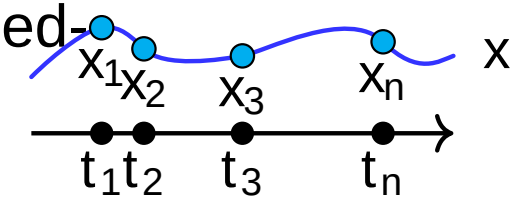
Observe $x = ((t_0; x_0); : : : ; (t_n; x_n))$ with $t_i \in \mathbb{R}$ and $x_i \in \mathbb{R}^v$.
Let $X : [t_0; t_n] \to \mathbb{R}^{v+1}$ interpolate this data, so $X(t_i) = (t_i; x_i)$.

Learn functions ζ, f and a linear map ℓ such that

$$y(t_0) = \zeta(t_0; x_0); \quad y(t) = y(t_0) + \int_{t_0}^{t} f(y(s))\, dX(s);$$

and the output is either $\ell(y(t_n))$ or $\ell(y(t))$.

ζ and f are arbitrary neural networks (feed-forward, ...), and y is hidden state.



Neural controlled di erential equations

Irregular time series

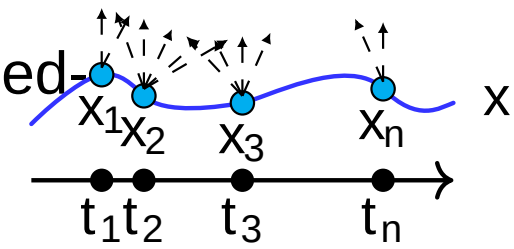
Observe $x = ((t_0; x_0); : : : ; (t_n; x_n))$ with $t_i \in \mathbb{R}$ and $x_i \in \mathbb{R}^v$.
Let $X : [t_0; t_n] \to \mathbb{R}^{v+1}$ interpolate this data, so $X(t_i) = (t_i; x_i)$.

Learn functions ζ, f and a linear map ℓ such that

$$y(t_0) = \zeta(t_0; x_0); \quad y(t) = y(t_0) + \int_{t_0}^{t} f(y(s))\, dX(s);$$

and the output is either $\ell(y(t_n))$ or $\ell(y(t))$.

ζ and f are arbitrary neural networks (feed-forward, ...), and y is hidden state.



Neural controlled di erential equations

Irregular time series

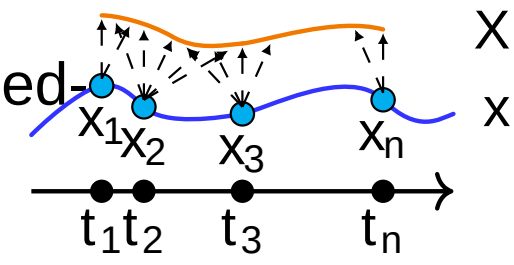
Observe $x = ((t_0; x_0); : : : ; (t_n; x_n))$ with $t_i \in \mathbb{R}$ and $x_i \in \mathbb{R}^v$.
Let $X : [t_0; t_n] \to \mathbb{R}^{v+1}$ interpolate this data, so $X(t_i) = (t_i; x_i)$.

Learn functions ζ, f and a linear map ℓ such that

$$y(t_0) = \zeta(t_0; x_0); \quad y(t) = y(t_0) + \int_{t_0}^{t} f(y(s))\, dX(s);$$

and the output is either $\ell(y(t_n))$ or $\ell(y(t))$.

ζ and f are arbitrary neural networks (feed-forward, ...), and y is hidden state.



Neural controlled di erential equations

Irregular time series

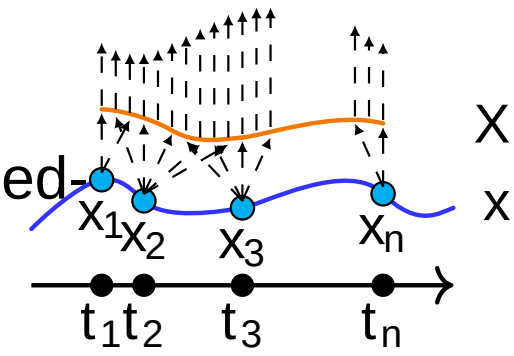
Observe $x = ((t_0; x_0); : : : ; (t_n; x_n))$ with $t_i \in \mathbb{R}$ and $x_i \in \mathbb{R}^v$.
Let $X : [t_0; t_n] \to \mathbb{R}^{v+1}$ interpolate this data, so $X(t_i) = (t_i; x_i)$.

Learn functions ζ, f and a linear map ℓ such that

$$y(t_0) = \zeta(t_0; x_0); \quad y(t) = y(t_0) + \int_{t_0}^{t} f(y(s))\, dX(s);$$

and the output is either $\ell(y(t_n))$ or $\ell(y(t))$.

ζ and f are arbitrary neural networks (feed-forward, ...), and y is hidden state.



Neural controlled di erential equations

Irregular time series

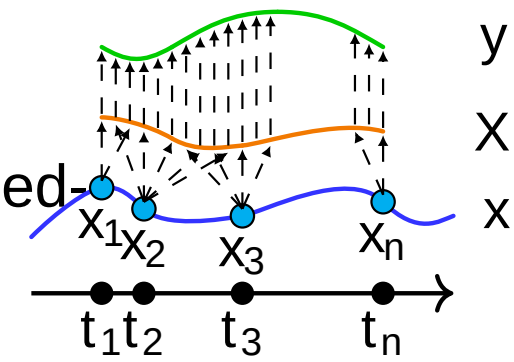
Observe $x = ((t_0; x_0); : : : ; (t_n; x_n))$ with $t_i \in R$ and $x_i \in R^v$.
Let $X : [t_0; t_n] \to R^{v+1}$ interpolate this data, so $X(t_i) = (t_i; x_i)$.

Learn functions ξ, f and a linear map ℓ such that

$$y(t_0) = \xi(t_0; x_0); \quad y(t) = y(t_0) + \int_{t_0}^{t} f(y(s))\, dX(s);$$

and the output is either $\ell(y(t_n))$ or $\ell(y(t))$.

ξ and f are arbitrary neural networks (feed-forward, ...), and y is hidden state.



Neural controlled di erential equations

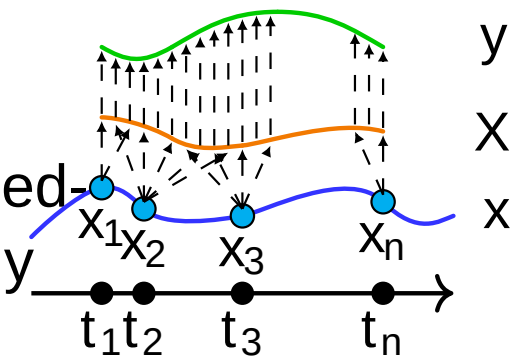
Irregular time series

Observe $x = ((t_0; x_0); : : : ; (t_n; x_n))$ with $t_i \in R$ and $x_i \in R^v$.
Let $X : [t_0; t_n] \to R^{v+1}$ interpolate this data, so $X(t_i) = (t_i; x_i)$.

Learn functions $, f$ and a linear map $`$ such that

$$y(t_0) = (t_0; x_0); \quad y(t) = y(t_0) + \int_{t_0}^{t} f(y(s))\, dX(s);$$

and the output is either $`(y(t_n))$ or $`(y(t))$.



and f are arbitrary neural networks (feed-forward, ...), and y is hidden state. Directly analogous to an RNN "$y_{n+1} = f(y_n; x_n)$".

We call this aneural controlled di erential equation.

Neural controlled di erential equations

Irregular time series

We call this aneural controlled di erential equation.
Examples: the time series $x = t((0; x_0); : : : ; (t_n; x_n))$ can correspond to:

Neural controlled di erential equations

Irregular time series

We call this aneural controlled di erential equation.

Examples: the time series $x = t((t_0; x_0); : : : ; (t_n; x_n))$ can correspond to:

The $x; y$-position as a pen is moved over paper, to draw a character.

Neural controlled di erential equations

Irregular time series

We call this aneural controlled di erential equation.

Examples: the time series $x = ((t_0, x_0), \ldots, (t_n, x_n))$ can correspond to:

- The x, y-position as a pen is moved over paper, to draw a character.
- Or patient hospital records: heart rate, lab results, and so on. A classic example of messy irregular data.

Neural controlled di erential equations

Irregular time series

We call this aneural controlled di erential equation.

Examples: the time series $x = ((t_0, x_0), \ldots, (t_n, x_n))$ can correspond to:

- The x, y-position as a pen is moved over paper, to draw a character.
- Or patient hospital records: heart rate, lab results, and so on. A classic example of messy irregular data.
- Spoken audio.

Neural controlled di erential equations

Irregular time series

We call this a neural controlled di erential equation.

Examples: the time series $x = ((t_0, x_0), \ldots, (t_n, x_n))$ can correspond to:

- The x, y-position as a pen is moved over paper, to draw a character.
- Or patient hospital records: heart rate, lab results, and so on. A classic example of messy irregular data.
- Spoken audio.
- Weather data, e.g. temperature and pressure as they change over time.

Neural controlled di erential equations

Irregular time series

We call this aneural controlled di erential equation.

Examples: the time series $x = f(t_0, x_0), \ldots, (t_n, x_n))$ can correspond to:

- The x, y-position as a pen is moved over paper, to draw a character.
- Or patient hospital records: heart rate, lab results, and so on. A classic example of messy irregular data.
- Spoken audio.
- Weather data, e.g. temperature and pressure as they change over time.
- Physics: the movement of a double pendulum.

Advantages:

Using a continuous-time theory pushes the problem of messy data into the interpolation, which handles it easily.

Advantages:

Using a continuous-time theory pushes the problem of messy data into the interpolation, which handles it easily.

The equation $y(t) = y(0) + \int_0^t f(y(s))\, dX(s)$ may (with a bit of mathematical rearranging) be solved as a (neural) ODE:

Advantages:

Using a continuous-time theory pushes the problem of messy data into the interpolation, which handles it easily.

The equation $y(t) = y(0) + \int_0^t f(y(s))\, dX(s)$ may (with a bit of mathematical rearranging) be solved as a (neural) ODE:

So we can solve it using existing software.

Advantages:

Using a continuous-time theory pushes the problem of messy data into the interpolation, which handles it easily.

The equation $y(t) = y(0) + \int_0^t f(y(s))\, dX(s)$ may (with a bit of mathematical rearranging) be solved as a (neural) ODE:

So we can solve it using existing software.

So we can utilise the memory-e cient adjoint method.

Advantages:

Using a continuous-time theory pushes the problem of messy data into the interpolation, which handles it easily.

The equation $y(t) = y(0) + \int_0^t f(y(s))\,dX(s)$ may (with a bit of mathematical rearranging) be solved as a (neural) ODE:

So we can solve it using existing software.

So we can utilise the memory-e cient adjoint method.

Drawing on the existing theory of CDEs gives strong theoretical guarantees. For example, neural CDEs are universal approximators.

Advantages:

Using a continuous-time theory pushes the problem of messy data into the interpolation, which handles it easily.

The equation $y(t) = y(0) + \int_0^t f(y(s))\, dX(s)$ may (with a bit of mathematical rearranging) be solved as a (neural) ODE:

So we can solve it using existing software.

So we can utilise the memory-e cient adjoint method.

Drawing on the existing theory of CDEs gives strong theoretical guarantees. For example, neural CDEs are universal approximators.

Neural CDEs demonstrate state-of-the-art performance.

Neural controlled di erential equations

Irregular time series

Model	Test Accuracy (mean std, computed across ve runs)						Memory usage
	30% dropped		50% dropped		70% dropped		
GRU-ODE	92.6%	1.6%	86.7%	3.9%	89.9%	3.7%	1.5
GRU- t	93.6%	2.0%	91.3%	2.1%	90.4%	0.8%	15.8
GRU-D	94.2%	2.1%	90.2%	4.8%	91.9%	1.7%	17.0
ODE-RNN	95.4%	0.6%	96.0%	0.3%	95.3%	0.6%	14.8
Neural CDE (ours)	98.7%	0.8%	98.8%	0.2%	98.6%	0.4%	1.3

Comparison to control theory problems (speaking very broadly):

Comparison to control theory problems (speaking very broadly):

- Control theory: System f is fixed; try to find optimal X producing a desired response y

Neural controlled di erential equations

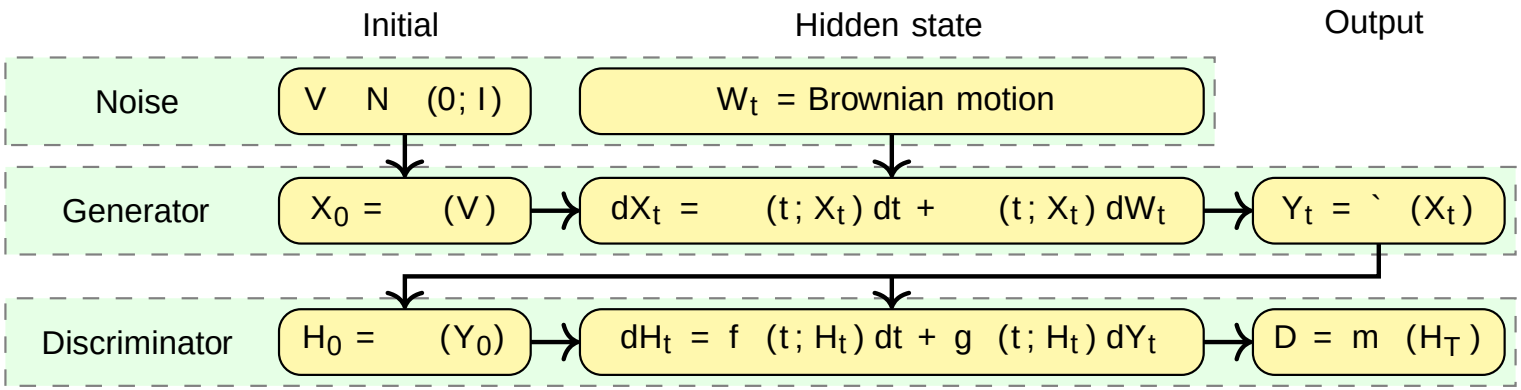
Irregular time series

Comparison to control theory problems (speaking very broadly):

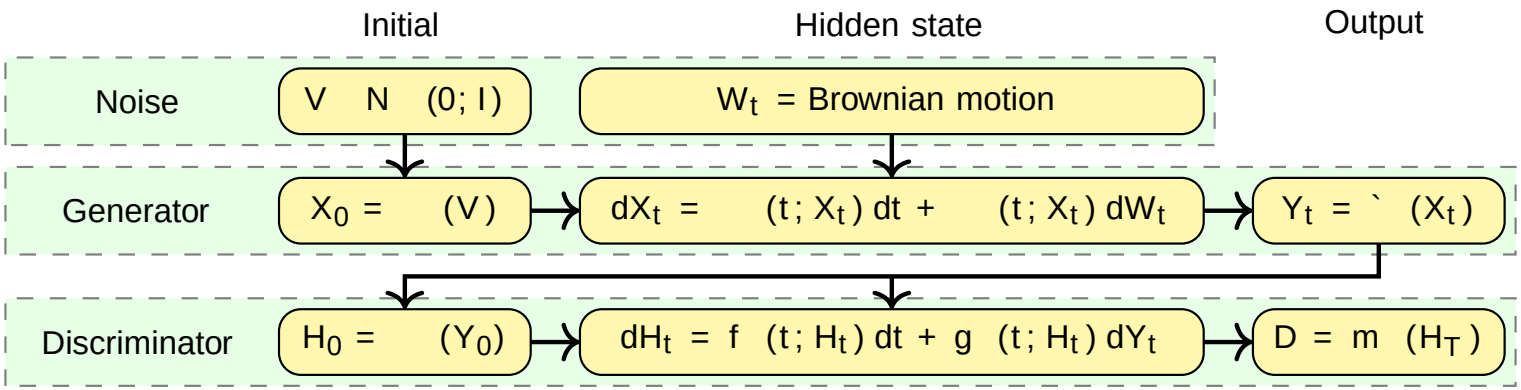
- Control theory: System f is fixed; try to find optimal X producing a desired response y.
- (Neural) CDEs: Input X is fixed; try to find optimal f producing a desired response y.

Neural stochastic di erential equations

Neural stochastic di erential equations

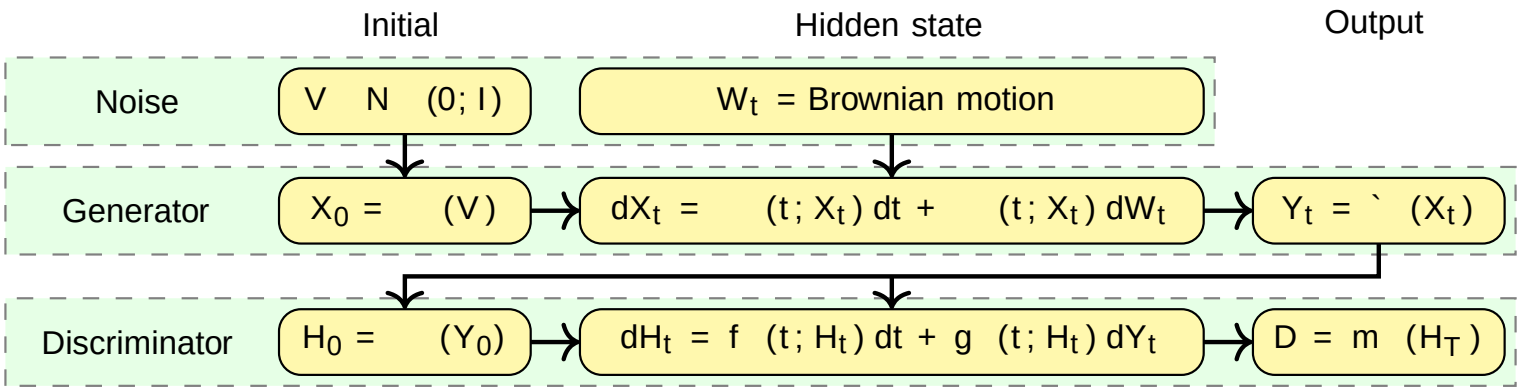


Neural stochastic di erential equations



Neural SDE / CDE form a generator/discriminator pair.

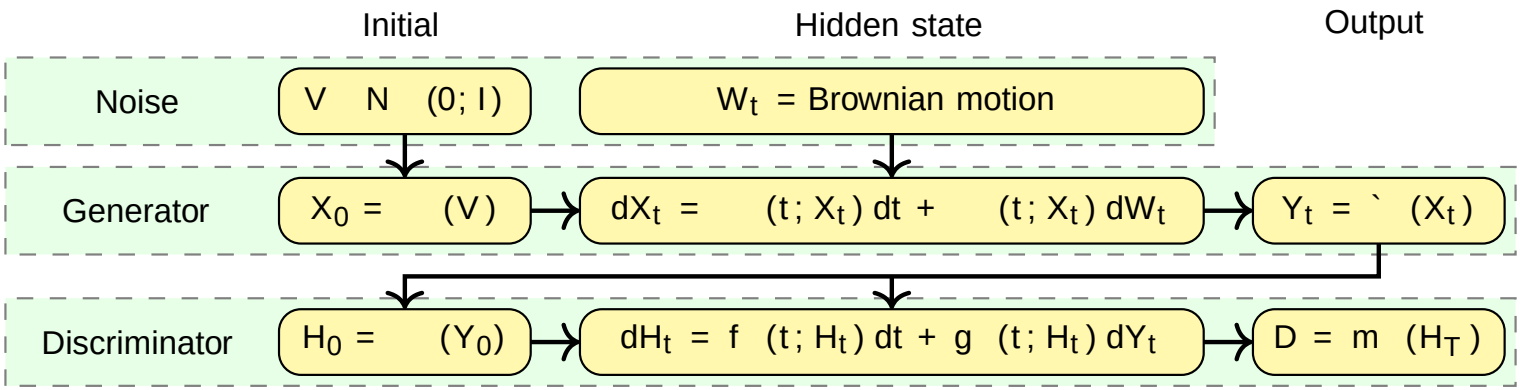
Neural stochastic di erential equations



Neural SDE / CDE form a generator/discriminator pair.

Arbitrary drift and di usions are admissible.

Neural stochastic di erential equations

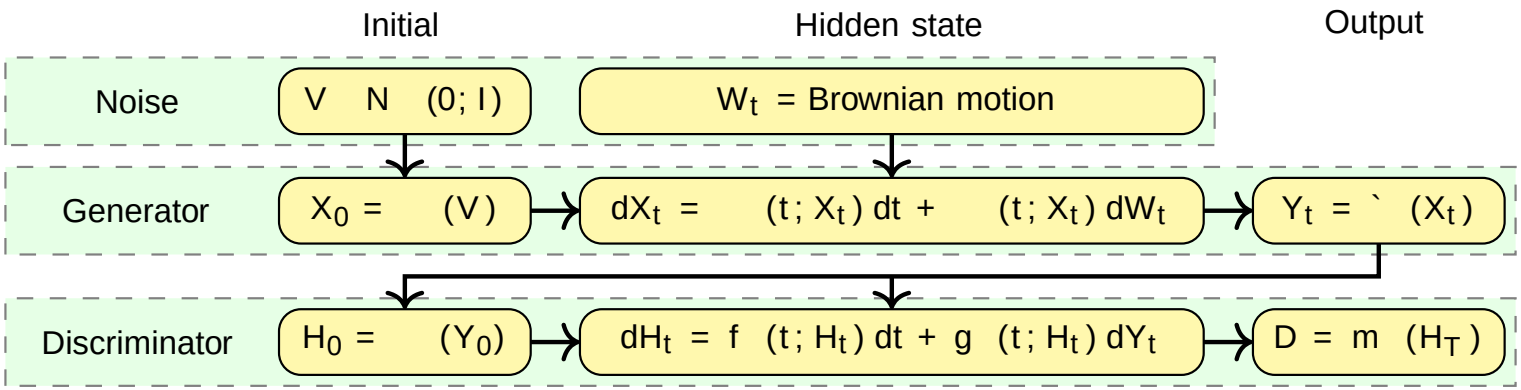


Neural SDE / CDE form a generator/discriminator pair.

Arbitrary drift and di usions are admissible.

In the in nite data limit any SDE may be learnt.

Neural stochastic di erential equations



Neural SDE / CDE form a generator/discriminator pair.

Arbitrary drift and di usions are admissible.

In the in nite data limit any SDE may be learnt.

We've come a long way from models like the Heston model...

$$dS_t = \quad S_t\,dt + \sqrt{\ }_t S_t\,dW_t^1$$
$$d\ _t = \quad (\qquad _t)\,dt + \sqrt{\ }_t\,dW_t^2$$

Neural stochastic di erential equations

Google/Alphabet Stocks:

Metric	Neural SDE		CTFP		Latent ODE	
Classi cation	0.357	0.045	0.165	0.087	0.000239	0.000086
Prediction	0.144	0.045	0.725	0.233	46.2	12.3
MMD	1.92	0.09	2.70	0.47	60.4	35.8

Beijing Air Quality:

Metric	Neural SDE		CTFP		Latent ODE	
Classi cation	0.589	0.051	0.764	0.064	0.392	0.011
Prediction	0.395	0.056	0.810	0.083	0.456	0.095
MMD	0.000160	0.000029	0.00198	0.00001	0.000242	0.000002

SGD dynamics:

Metric	Neural SDE		CTFP		Latent ODE	
Classi cation	0.507	0.019	0.676	0.014	0.0112	0.0025
Prediction	0.00843	0.00759	0.0808	0.0514	0.127	0.152
MMD	5.28	1.27	12.0	0.5	23.2	11.8

Software

We've got standardised software for solving neural di erential equations.

Software

We've got standardised software for solving neural di erential equations.

PyTorch:

Software

We've got standardised software for solving neural di erential equations.

PyTorch:

torchdiffeq (ODEs)

Software

We've got standardised software for solving neural di erential equations.

PyTorch:

torchdiffeq (ODEs)
torchcde (CDEs)

Software

We've got standardised software for solving neural di erential equations.

PyTorch:

torchdiffeq (ODEs)
torchcde (CDEs)
torchsde (SDEs)

Software

We've got standardised software for solving neural differential equations.

- PyTorch:
  - torchdiffeq (ODEs)
  - torchcde (CDEs)
  - torchsde (SDEs)
- Julia:

Software

We've got standardised software for solving neural di erential equations.

- PyTorch:
  - torchdiffeq (ODEs)
  - torchcde (CDEs)
  - torchsde (SDEs)
- Julia:
  - Di erentialEquations.jl

Software

We've got standardised software for solving neural di erential equations.

- PyTorch:
  - torchdiffeq (ODEs)
  - torchcde (CDEs)
  - torchsde (SDEs)
- Julia:
  - Di erentialEquations.jl

Software

We've got standardised software for solving neural di erential equations.

- PyTorch:
  - torchdiffeq (ODEs)
  - torchcde (CDEs)
  - torchsde (SDEs)
- Julia:
  - Di erentialEquations.jl

Autodi erentiable, GPU-capable, composable.

Summary

Summary

Parameterised di erential equation are already ubiquitous throughout mathematical modelling.

Summary

Parameterised di erential equation are already ubiquitous throughout mathematical modelling.

Neural di erential equations are a powerful extension to this paradigm.

Summary

Parameterised di erential equation are already ubiquitous throughout mathematical modelling.

Neural di erential equations are a powerful extension to this paradigm.

Widespread applications to improve traditional mathematical modelling (nance, physics, engineering, :), and to modern machine learning problems (time series, generative modelling).

Summary

Parameterised di erential equation are already ubiquitous throughout mathematical modelling.

Neural di erential equations are a powerful extension to this paradigm.

Widespread applications to improve traditional mathematical modelling (nance, physics, engineering, :), and to modern machine learning problems (time series, generative modelling).

Convince the rest of the world that they should be using neural di erential equations too.

References

R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, \Neural Ordinary Di erential Equations", in Advances in Neural Information Processing Systems 31, pp. 6571{6583, Curran Associates, Inc., 2018

T. J. Lyons, \Di erential equations driven by rough signals", Revista Matemática Iberoamericana, vol. 14, no. 2, pp. 215{310, 1998

P. Kidger, J. Morrill, J. Foster, and T. Lyons, \Neural Controlled Di erential Equations for Irregular Time Series", Advances in Neural Information Processing Systems 33, 2020a

P. Kidger, J. Foster, X. Li, H. Oberhauser, and T. Lyons, \Neural SDEs as In nite-Dimensional GANs", Machine Learning and the Physical Sciences, NeurIPS, 2020b

References

(Software)

torchdiffeq : https://github.com/rtqichen/torchdiffeq
torchsde : https://github.com/google-research/torchsde
torchcde : https://github.com/patrick-kidger/torchcde

Di erentialEquations.jl:
https://github.com/SciML/DifferentialEquations.jl

Final notes

(A version of) these slides are available on my website.
(`https://kidger.site`)

Final notes

(A version of) these slides are available on my website. (`https://kidger.site`)

Feel free to send me an email / contact me on Twitter (@PatrickKidger) if you have any questions later.

Final notes

(A version of) these slides are available on my website. (`https://kidger.site`)

Feel free to send me an email / contact me on Twitter (@PatrickKidger) if you have any questions later.

I'm a developer for most of the major PyTorch libraries so you can get in touch about those too.

Final notes

(A version of) these slides are available on my website. (`https://kidger.site`)

Feel free to send me an email / contact me on Twitter (@PatrickKidger) if you have any questions later.

I'm a developer for most of the major PyTorch libraries so you can get in touch about those too.

Any questions now?