Neural CDEs for Long Time Series via the Log-ODE Method
James Morrill, Patrick Kidger, Cristopher Salvi, James Foster, and Terry Lyons
University of Oxford; Alan Turing Institute.

Summary

Neural CDEs can be viewed as a continuous-time RNN, much as Neural ODEs with ResNets. However, as with RNNs, training can begin to breakdown for long time series.

In this paper we show:

• How to convert the CDE equation into an equivalent ODE by using the log-ODE method, which is a standard method for solving CDEs in the field of rough analysis. Tools from Neural ODEs can then be directly applied.
• In particular this allow us to take integration steps larger than the discretisation of the data – whilst retaining solution accuracy.
• On long time series, it can result in speed-ups of up to 100x, significantly reduced memory requirements, and improvements in performance.

Background: Neural CDEs

Consider a time series $x$ as a collection of points $x_i \in \mathbb{R}^{n-1}$ with corresponding time-stamps $t_i \in \mathbb{R}$ such that $x = ((t_0, x_0), (t_1, x_1), ..., (t_n, x_n))$, and $t_0 < ... < t_n$ and let $X: [t_0, t_n] \rightarrow \mathbb{R}^n$ be some interpolation of the data such that $X_i = (t_i, x_i)$.

Let $\xi_0: \mathbb{R}^n \rightarrow \mathbb{R}^w$ and $f_0: \mathbb{R}^w \rightarrow \mathbb{R}^{w \times n}$ be neural networks and let $\xi: \mathbb{R}^w \rightarrow \mathbb{R}^9$ be linear.

We define $Z$ as the hidden state and $Y$ as the output of a neural controlled differential equation driven by $X$ if

$$ Z_{t_0} = \xi_0(t_0, x_0), \quad \text{with} \quad Z_t = Z_{t_0} + \int_{t_0}^{t} f_0(Z_s) \, dX_s \quad \text{and} \quad Y_t = \xi_0(Z_t) \quad \text{for} \quad t \in (t_0, t_n). \quad (1) $$

• If instead of $dX_t$ there was $ds$, then this would be a Neural ODE.
• The $dX_t$ is what allows the hidden state to be updated from incoming data. (Unlike Neural ODEs, for which the solution is determined by the initial condition.)
• For a differentiable control path $X$, the simple way to convert to an ODE via $dX_s = \frac{dX}{d\tau}(s) \, ds$.

Background: The Log-ODE Method

Consider the CDE integral equation for $Z_i$ defined above. The log-ODE method states:

$$ Z_{t_0} \approx \tilde{Z}_{t_0} \quad \text{where} \quad \tilde{Z}_t = \tilde{Z}_{t_0} + \int_{t_0}^{t} \text{LogSig}_N^X(\tilde{Z}_s) \, \frac{\text{LogSig}_N^X(X)}{b-a} \, ds, \quad \text{and} \quad \tilde{Z}_n = Z_n. \quad (2) $$

• That is, the solution to the CDE in (1) can be approximated by the solution to the ODE in (2).
• The $dX_t$ term has been replaced by $\frac{\text{LogSig}_N^X(X)}{b-a} \, ds$.
• $\text{LogSig}_N^X(X)$ is a particular map to a vector of values that summarise $X$. Intuitively, these are the terms that are most relevant to solving the CDE equation.
• This approximation becomes arbitrarily good as $N \rightarrow \infty$.

Updating the Neural CDE Hidden State Equation

Assume the time series is very long. Pick points $r_i$ such that $t_0 = r_0 < r_1 < ... < r_m = t_n$ with $m \ll n$. In model training we will simply split the integral from equation (1) over each $[r_i, r_{i+1}]$ and update via

$$ Z_{r_{i+1}} = Z_{r_i} + \int_{r_i}^{r_{i+1}} \frac{\text{LogSig}_N^X(\tilde{Z}_s)}{b-a} \, ds. \quad (3) $$

Experiments

• Tested on four problems with lengths up to 17,000.
• Observe significant training speed-ups, reduced memory requirements, and even improvements in performance when compared to the original NCDE model.

Find out more:

This paper: https://arxiv.org/abs/2009.08295
Code: https://github.com/patrick-kidger/torchcde
Library: https://github.com/patrick-kidger/torchcde
Email: morrill@maths.ox.ac.uk