Mathematical Forecasting Methods

Лекция 8

МФТИ

Осень, 2023

RNN и метод Эйлера

Рассмотрим рекуррентные модели:

$$\mathbf{h}_{t+1} = \mathbf{h}_t + f(\mathbf{h}_t, \boldsymbol{\theta}),$$

где $t \in \{1, \ldots, T\}$.
Можно заметить что данная формула эквивалентна дискретизации обыкновенных ДУ методом Эйлера при $\Delta t = 1$

$$\mathbf{h}(t+1) = \mathbf{h}(t) + f(\mathbf{h}(t), t, \boldsymbol{\theta}) \Rightarrow f(\mathbf{h}(t), t, \boldsymbol{\theta}) = \frac{\mathbf{h}(t+1) - \mathbf{h}(t)}{\Delta t}$$

Т.о. если мы в пределе возьмем больше слоев с меньшим шагом, мы получим непрерывную динамики праметризированную обычной нейронной сетью

$$\frac{d\mathbf{h}(t)}{dt} = f(\mathbf{h}(t), t, \boldsymbol{\theta}).$$

Обобщение

Таким образом RNN аппроксимирует функцию динамики с помощью метода Эйлера. Однако мы не ограничены только этим методом. На самом деле мы можем взять любой (нарпимер метод Рунге-Кутты).

Пусть $\mathbf{x} = \mathbf{h}(t_0)$ - начальное значение и $\mathbf{y} = \mathbf{h}(t_1)$ - конечное значение, тогда

$$\mathbf{y} = \int_{t_0}^{t_1} f(\mathbf{h}(t), t, \boldsymbol{\theta})dt + \mathbf{x} = ODESolve(\mathbf{x}, f, t_0, t_1, \boldsymbol{\theta})$$

Теперь чтобы обновить параметры $\boldsymbol{\theta}$ необходимо взять градиент скалярной функции потерь $\mathcal{L}(\mathbf{y})$ по параметрам $\frac{\partial \mathcal{L}(\mathbf{y})}{\partial \boldsymbol{\theta}}$

Проблема

Если воспользоваться бэкпропом то потребуется больших затрат памяти, т.к. будем проходить по операциям солвера.

Neural ODE

Для этого введем сопряженное (adjoint) состояние, которое показывает как изменяется функция потерь от скрытого состояния $\mathbf{h}(t)$ в некоторый момент t:

$$\mathbf{a_h}(t) = \frac{\partial \mathcal{L}(\mathbf{y})}{\partial \mathbf{h}(t)}$$

и динамика которого задается следующим ДУ:

Теорема Понтрягина (часть 1)

$$\frac{d\mathbf{a_h}(t)}{dt} = -\mathbf{a_h}(t)\frac{\partial f(\mathbf{h}(t), t, \boldsymbol{\theta})}{\partial \mathbf{h}(t)} = g(\mathbf{h}(t), t)$$

Соответственно чтобы получить сопряженное состояние в момент t_0 мы таже можем запустить солвер из t_1 в t_0:

$$\mathbf{a_h}(t_0) = -\int_{t_1}^{t_0} \mathbf{a_h}(t)\frac{\partial f(\mathbf{h}(t), t, \boldsymbol{\theta})}{\partial \mathbf{h}(t)} dt + \frac{\partial \mathcal{L}(\mathbf{y})}{\partial \mathbf{y}} =$$
$$= ODESolve(\mathbf{y}, g, t_1, t_0, \boldsymbol{\theta})$$

Chen R. T. Q. et al. Neural Ordinary Differential Equations, 2018

Neural ODE

Теперь, когда мы можем посчитать сопряженное состояние для любого момента t, получим градиент по параметрам:

$$\mathbf{a}_{\boldsymbol{\theta}}(t_0) = \frac{\partial \mathcal{L}(\mathbf{y})}{\partial \boldsymbol{\theta}},$$

динамика которого задается следующим ДУ:

Теорема Понтрягина (часть 2)

$$\frac{d\mathbf{a}_{\boldsymbol{\theta}}(t)}{dt} = -\mathbf{a}_{\mathbf{h}}(t)\frac{\partial f(\mathbf{h}(t), t, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = g'(\mathbf{a}_{\mathbf{h}}(t), \mathbf{h}(t), t)$$

Соответственно чтобы получить сопряженное состояние в момент t_0 мы таже можем запустить солвер из t_1 в t_0:

$$\mathbf{a}_{\boldsymbol{\theta}}(t_0) = -\int_{t_1}^{t_0} \mathbf{a}_{\mathbf{h}}(t)\frac{\partial f(\mathbf{h}(t), t, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} dt + 0 =$$
$$= ODESolve(\mathbf{y}, g', t_1, t_0, \boldsymbol{\theta})$$

Chen R. T. Q. et al. Neural Ordinary Differential Equations, 2018

Neural ODE

По полученным выражениям заметно, что для того чтобы посчитать сопряженное состояние $\mathbf{a_h}(t)$ требуется знать скрытое состояние $\mathbf{h}(t)$, а для градиета по параметрам $\mathbf{a}_{\boldsymbol{\theta}}(t)$ необходимо знать оба вышеперечисленных значения. Таким образом обучение Neural ODE следует по следующей схеме

Forward pass

$$\mathbf{y} = \mathit{ODESolve}(\mathbf{x}, f, t_0, t_1, \boldsymbol{\theta})$$

Backward pass

$$\begin{cases} \mathbf{a}_{\boldsymbol{\theta}}(t_0) = \mathit{ODESolve}(\mathbf{y}, g', t_1, t_0, \boldsymbol{\theta}) \\ \mathbf{a_h}(t_0) = \mathit{ODESolve}(\mathbf{y}, g, t_1, t_0, \boldsymbol{\theta}) \\ \mathbf{x} = \mathit{ODESolve}(\mathbf{y}, f, t_1, t_0, \boldsymbol{\theta}) \end{cases}$$

Chen R. T. Q. et al. Neural Ordinary Differential Equations, 2018

Neural ODE

Для того чтобы не вызывать несколько раз солвер на этапе *backward pass* можно конкатенировать все три вектора $\mathbf{a}_\theta(t)$, $\mathbf{a}_\mathbf{h}(t)$ и $\mathbf{h}(t)$ и вызвать по ним *только один* солвер

Algorithm 1 Reverse-mode derivative of an ODE initial value problem

---

Input: dynamics parameters θ, start time t_0, stop time t_1, final state $\mathbf{z}(t_1)$, loss gradient $\partial L / \partial \mathbf{z}(t_1)$

$s_0 = [\mathbf{z}(t_1), \frac{\partial L}{\partial \mathbf{z}(t_1)}, \mathbf{0}_{|\theta|}]$ ▷ Define initial augmented state

def aug_dynamics($[\mathbf{z}(t), \mathbf{a}(t), \cdot], t, \theta$): ▷ Define dynamics on augmented state

  return $[f(\mathbf{z}(t), t, \theta), -\mathbf{a}(t)^\mathsf{T} \frac{\partial f}{\partial \mathbf{z}}, -\mathbf{a}(t)^\mathsf{T} \frac{\partial f}{\partial \theta}]$ ▷ Compute vector-Jacobian products

$[\mathbf{z}(t_0), \frac{\partial L}{\partial \mathbf{z}(t_0)}, \frac{\partial L}{\partial \theta}] = \text{ODESolve}(s_0, \text{aug\_dynamics}, t_1, t_0, \theta)$ ▷ Solve reverse-time ODE

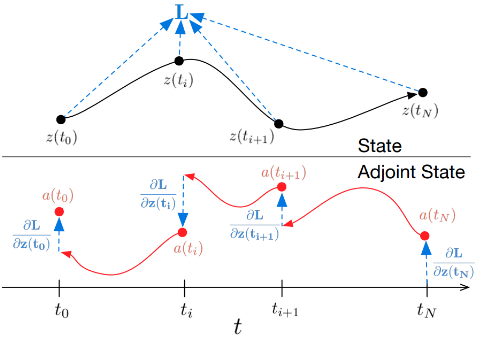
return $\frac{\partial L}{\partial \mathbf{z}(t_0)}, \frac{\partial L}{\partial \theta}$ ▷ Return gradients

---

Chen R. T. Q. et al. Neural Ordinary Differential Equations, 2018

Neural ODE

Теперь если нам необходимо знать градиенты для разных моментов t_i $\forall i \in \{1, \dots, N\}$ в forward pass мы получаем значение $h(t_i)$. Этап backward pass мы разбиваем на пары (t_i, t_{i+1}) и применяем для каждой солвер. Далее каждое сопряженное состояние $\mathbf{a_h}(t_i)$ мы смещаем на соответсвенный градиент $\frac{\partial \mathcal{L}(\mathbf{h}(t_{i+1}))}{\partial \mathbf{h}(t_i)}$

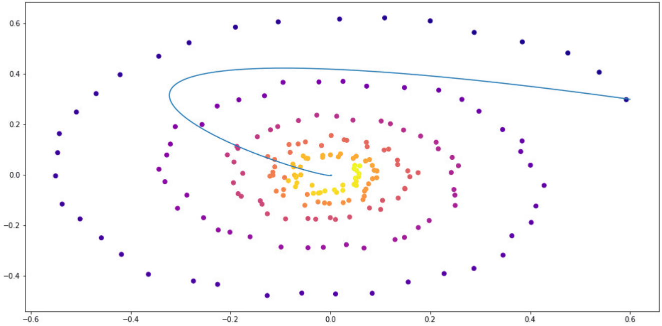


Chen R. T. Q. et al. Neural Ordinary Differential Equations, 2018

Выводы

- **Модели непрерывных временных рядов**: В отличие от рекуррентных нейронных сетей, которые требуют дискретизации интервалов наблюдения, Neural ODE позволяет работать с данными, полученными с *произвольными* временными интервалами
- **Непрерывные нормализующие потоки**: Неожиданным побочным преимуществом непрерывных преобразований является то, что формула замены переменных становится легче вычислять

Пример



Михаил Сурцуков, Знакомство с Neural ODE, 2019