Нейронные обыкновенные дифференциальные уравнения

Елена Щербакова

1 июня 2020

Прежде чем перейдем к нейронным обыкновенным дифференциальным уравнениям, рассмотрим сперва их предшественника - residual network. В простой нейронной сети преобразование скрытого состояния через сеть - $h(t+1) = f(h(t), \Theta(t))$, где f - сеть, $h(t)$ - скрытое состояние слоя t и $\Theta(t)$ веса на слое t. Преобразование скрытого состояния в residual network похоже и может быть формализовано как $h(t+1) = h(t) + f(h(t), \Theta(t))$.

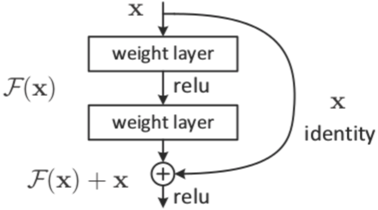


Рис.: ResNets Architecture.

Residual network и метод Эйлера

Связь между ResNets и ОДУ лучше всего демонстрирует уравнение $h(t+1) = h(t) + f(h(t), \Theta(t))$. Поскольку отношение рекурсивное, то его можно расписать в следующую последовательность (i - вход):

$$
\begin{aligned}
h_1 &= i + f(i, \Theta_1) \\
h_2 &= h_1 + f(h_1, \Theta_2) \\
h_3 &= h_2 + f(h_2, \Theta_3) \\
&\vdots \\
h_{t+1} &= h_t + f(h_t, \Theta_t)
\end{aligned}
$$

Рис.: The ResNet relation unraveled.

Проводится аналогия с методом Эйлера с шагом 1. Вывод: нейронные сети можно представить как дифференциальные уравнения.

ODENet

Дифференциальные уравнения не делают той же дискретизации, что и нейронная сеть, поэтому требуется ее модифицировать, чтобы получить ODENet.

ResNet:

```
def f(z, t, Θ):
    return nnet(z, Θ_t)

def resnet(z, Θ):
    for t in range(1, T):
        z = z + f(z, t, Θ)
    return z
```

ODENet:

```
def f(z, t, Θ):
    return nnet([z,t], Θ)

def ODENet(z, Θ):
    for t in range(1, T):
        z = z + f(z, t, Θ)
    return z
```

У ODENet Θ общие для всех слоев. В итоге у ODENet меньше параметров, чем у ResNet. ResNet для получения 0.4 test error на MNIST использует 0.6 million parameters, а ODENet для той же точности - 0.2 million параметров!

Перспективы

- Эффективность в работе с памятью. В оригинальной статье авторы демонстрируют, как вычислить градиенты скалярной функции потерь по входам ODE solver без backpropagating. Не приходится хранить значения при forward pass, что позволяет тренировать с константными затратами памяти.
- Адаптивные вычисления. Можно использовать более продвинутые ODE solver.
- Эффективность по параметрам.
- Применимость к моделированию “иррегулярных” временных рядов.

Проверка Neural ODE архитектуры на MNIST Fashion

Neural ODE могут быть использованными в виде компонентов в более привычных архитектурах. Для этого заменяют остаточные (residual) блоки на Neural ODE в классификаторе MNIST/MNIST Fashion.

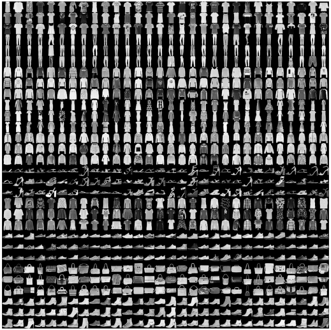
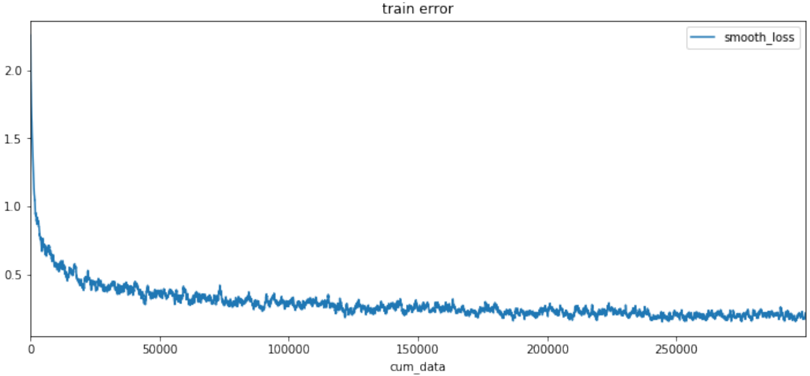


Рис.: MNIST Fashion.

Проверка Neural ODE архитектуры на MNIST Fashion

После очень грубой тренировки в течение всего 5 эпох модель уже достигла Test Accuracy: 91.670 %. Можно сказать, что Нейронные ОДУ хорошо интегрируются в виде компонента в более традиционные сети.



Скрытая генеративная функция для моделирования временного ряда

Генеративная модель через процедуру сэмплирования:

$$z_{t_0} \sim \mathcal{N}(0, I)$$

$$z_{t_1}, z_{t_2}, \ldots, z_{t_M} = \text{ODESolve}(z_{t_0}, f, \theta_f, t_0, \ldots, t_M)$$

$$x_{t_i} \sim p(x \mid z_{t_i}; \theta_x)$$

, которая может быть обучена, используя подход вариационных автокодировщиков.

1. Пройтись рекуррентным энкодером через временную последовательность назад во времени, чтобы получить параметры $\mu_{z_{t_0}}$, $\sigma_{z_{t_0}}$ вариационного апестериорного распределения, а потом сэмплировать из него:

$$z_{t_0} \sim q\left(z_{t_0} \mid x_{t_0}, \ldots, x_{t_M}; t_0, \ldots, t_M; \theta_q\right) = \mathcal{N}\left(z_{t_0} \mid \mu_{z_{t_0}} \sigma_{z_{t_0}}\right)$$

Скрытая генеративная функция для моделирования временного ряда

2. Получить скрытую траекторию:

$$z_{t_1}, z_{t_2}, \ldots, z_{t_N} = \text{ODESolve}(z_{t_0}, f, \theta_f, t_0, \ldots, t_N), \text{ где } \frac{dz}{dt} = f(z, t; \theta_f)$$

3. Отобразить скрытую траекторию в траекторию в данных, используя другую нейросеть: $\hat{x_{t_i}}(z_{t_i}, t_i; \theta_x)$.

Скрытая генеративная функция для моделирования временного ряда

4. Максимизировать оценку нижней границы обоснованности (ELBO) для сэмплированной траектории:

$$\text{ELBO} \approx N\Big(\sum_{i=0}^{M} \log p(x_{t_i} \mid z_{t_i}(z_{t_0}; \theta_f); \theta_x) +$$

$$+ KL\left(q(z_{t_0} \mid x_{t_0}, \dots, x_{t_M}; t_0, \dots, t_M; \theta_q) \parallel \mathcal{N}(0, I)\right)\Big)$$

И в случае Гауссовского апостериорного распределения $p(x \mid z_{t_i}; \theta_x)$ и известного уровня шума σ_x:

$$\text{ELBO} \approx -N\Big(\sum_{i=1}^{M} \frac{(x_i - \hat{x}_i)^2}{\sigma_x^2} - \log \sigma_{z_{t_0}}^2 + \mu_{z_{t_0}}^2 + \sigma_{z_{t_0}}^2\Big) + C$$

Скрытая генеративная функция для моделирования временного ряда

Граф вычислений скрытой ОДУ модели можно изобразить следующим образом:

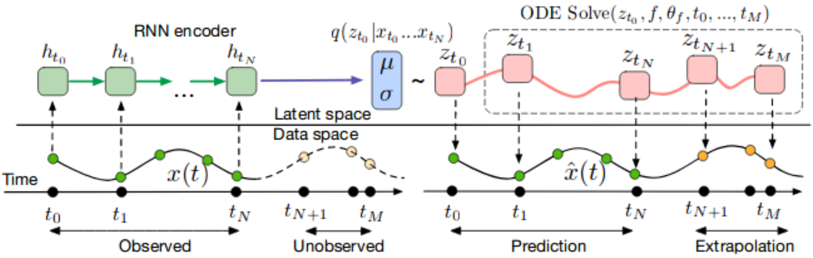
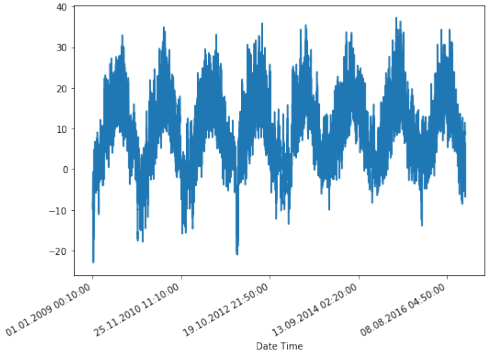


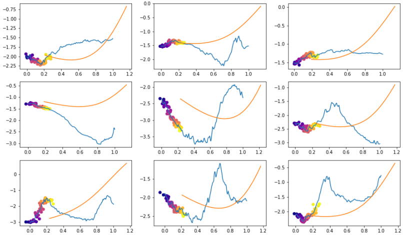
Иллюстрация взята из оригинальной статьи.

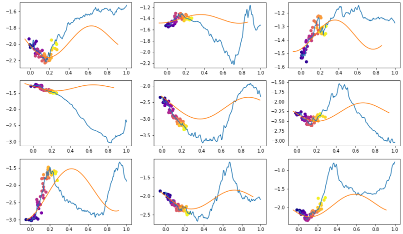
Данные для временного ряда

Использовался weather time series dataset, собранный Max Planck Institute for Biogeochemistry.
Данные содержат 14 разных признаков, таких как температура воздуха, давление и влажность. Периодичность - каждые 10 минут, начиная с 2003. Использовались данные с 2009 по 2016.
В качестве временного ряда были взяты температуры.

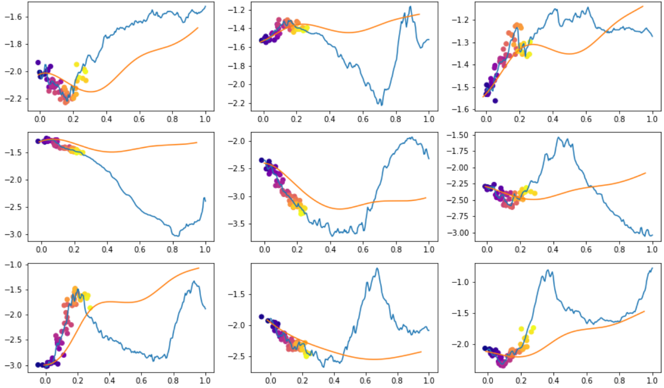


Результаты эксперимента





Результаты эксперимента



Точки — это зашумленные наблюдения оригинальной траектории (синий), желтая — это реконструированная и интерполированная траектория, используя точки как входы.

Список литературы I

- Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, David Duvenaud. "Neural Ordinary Differential Equations."Advances in Neural Processing Information Systems. 2018.
- Yulia Rubanova, Ricky Chen, David Duvenaud. "Latent ODEs for Irregularly-Sampled Time Series"(2019)
- https://jontysinai.github.io/jekyll/update/2019/01/18/understanding-neural-odes.html
- https://towardsdatascience.com/neural-odes-breakdown-of-another-deep-learning-breakthrough-3e78c7213795
- https://github.com/rtqichen/torchdiffeq

Список литературы II

- https://github.com/kmkolasinski/deep-learning-notes/tree/master/seminars/2019-03-Neural-Ordinary-Differential-Equations
- https://github.com/msurtsukov/neural-ode
- https://github.com/Zymrael/awesome-neural-ode
- https://github.com/SciML/DiffEqFlux.jl
- https://github.com/saparina/neural-ode
- X. Xie, A. K. Parlikad and R. S. Puri, "A Neural Ordinary Differential Equations Based Approach for Demand Forecasting within Power Grid Digital Twins,"2019 IEEE (SmartGridComm), Beijing, China, 2019, pp. 1-6.
- Augmented Neural ODEs, Emilien Dupont, Arnaud Doucet, Yee Whye Teh. https://arxiv.org/abs/1904.01681

Список литературы III

Спасибо за внимание!

Постановка задачи

Пусть есть процесс, который подчиняется некоторому неизвестному ОДУ, и несколько (зашумленных) наблюдений вдоль траектории процесса:

$$\frac{dz}{dt} = f(z(t), t)$$

$$\{(z_0, t_0), (z_1, t_1), \ldots, (z_M, t_M)\} - \text{наблюдения}$$

Как найти аппроксимацию $\widehat{f}(z, t, \theta)$ функции динамики $f(z, t)$?

Постановка задачи

Сначала рассмотрим более простую задачу: есть только 2 наблюдения, в начале и в конце траектории, $(z_0, t_0), (z_1, t_1)$. Эволюция системы запускается из состояния z_0, t_0 на время $t_1 - t_0$ с какой-то параметризованной функцией динамики, используя любой метод эволюции систем ОДУ. После того, как система оказывается в новом состоянии $\hat{z}_1, t_1$, оно сравнивается с состоянием z_1 и разница между ними минимизируется варьированием параметров θ функции динамики.
Рассмотрим минимизацию функции потерь $L(\hat{z}_1)$:

$$L(z(t_1)) = L\Big(\int_{t_0}^{t_1} f(z(t), t, \theta) dt \Big) = L\big(\text{ODESolve}(z(t_0), f, t_0, t_1, \theta)\big)$$