Обзор нейронных дифференциальных уравнений

1 Введение. Residual архитектуры.

Среди нейросетевых архитектур очень популярными оказались архитектуры, использующие механизм residual связей. Для анализа изображений примером такой архитектуры является ResNet [1]. В задаче анализа текстов примером такой архитектуры служит Transformer [2].

Опишем теперь данный механизм residual связи. Пусть стандартный слой на вход принимает результат работы предыдущих t слоёв сети — представление h_t. Тогда выход сети можно задать в следующем виде:

$$h_{t+1} = f(h_t, \theta_t),$$

где $f(h_t, \theta_t)$ — это какое-то нелинейное преобразование, задающееся на практике с помощью полносвязанных, свёрточных слоёв, их комбинаций и нелинейности.

Пусть теперь $f(h_t, \theta_t)$ совпадает по размерности с h_t. Тогда слоем с residual связью называют следующее преобразование:

$$h_{t+1} = f(h_t, \theta_t) + h_t, \tag{1}$$

где суммирование происходит поэлементно.

Как было замечено авторами архитектуры ResNet, описанный слой позволяет сети легко «сохранять» информацию с предыдущих слоёв. Благодаря этому удалось существенно увеличить глубину сетей, не теряя при этом качества [1].

2 Нейронные обыкновенные дифференциальные уравнения.

Рассмотрим уравнение (1) и возьмём представление h_t как функцию от t. Тогда, перенеся h_t в левую часть и добавив в функцию $f(\cdot)$ зависимость от t, получим:

$$h(t+1) - h(t) = f\big(h(t), t, \theta\big)$$
$$\frac{h(t+1) - h(t)}{(t+1) - t} = f\big(h(t), t, \theta\big),$$

Если рассматривать функцию $h(t)$ как функцию от непрерывного аргумента на некотором отрезке $[t_0, t_1]$, то уравнение можно переписать в следующем виде:

$$\frac{h(t+\Delta t) - h(t)}{\Delta t} = f\big(h(t), t, \theta\big),$$

и, устремив число «слоёв» к бесконечности, получим:

$$\frac{\partial h(t)}{\partial t} = f\big(h(t), t, \theta\big), \tag{2}$$

Уравнение (2) является обыкновенным дифференциальным уравнением (ОДУ). ОДУ с начальным условием $h(t_0) = h_{t_0}$ называется задачей Коши и по теореме о существовании и единственности решения задачи Коши $h(t)$ представимо в виде:

$$h(t) = h(t_0) + \int_{t_0}^{t} f\big(h(t), t, \theta\big)dt$$

При этом значение $h(t)$ ищется приблизительно, с заданной точностью, с помощью численного решения дифференциального уравнения (методами Adams или Рунге-Кутта):

$$h(t) = \text{Solver}(h(t_0), f(\cdot), t_0, t_1, \theta).$$

Рассмотрим теперь как оптимизировать выучиваемые параметры θ функции $f(\cdot)$ при наличии функции потерь $\mathcal{L}\big(z(t_1)\big)$. Стандартно, оптимизация ведётся с помощью метода градиентного спуска. Для этого необходимо знать градиент $\frac{\partial \mathcal{L}}{\partial \theta}$.

Для подсчёта градиента вводится вспомогательное понятие сопряжённого (adjoint): $a(t) = \frac{\partial \mathcal{L}}{\partial z(t)}$, которое показывает как зависит функция потерь от скрытого состояния. Сопряжённое $a(t)$, в свою очередь, задаётся другим ОДУ, которое имеет следующий вид:

$$\frac{da(t)}{dt} = -a(t)^T \frac{\partial f\big(z(t), t, \theta\big)}{\partial z}.$$

Доказательство этого факта можно найти в Appendix B.1 оригинальной статьи [3].

Для данного уравнения известно начальное состояние в точке t_1. Соответственно, чтобы найти $a(t)$ в любой точке необходимо решить задачу Коши. Решение вычисляется в обратную сторону, от t_1 к t_0, что соответствует обратному распространению ошибки при обучении нейросетей.

Градиент по параметрам вычисляется следующим образом:

$$\frac{\partial \mathcal{L}}{\partial \theta} = -\int_{t_1}^{t_0} a(t) \frac{\partial f\big(z(t), t, \theta\big)}{\partial \theta} dt. \tag{3}$$

Доказательство этого равенства можно найти в Appendix B.2 оригинальной статьи [3]. Зная $a(t)$ в нужных точках, значение этого интеграла считается численными методами.

Результат. Рассмотрим теперь основные достоинства предложенного подхода по сравнению со стандартными нейросетями:

- Использование численных методов при решении задачи Коши позволяет выбирать баланс между точностью решения и временем вычисления;
- Задание нейросетевого ОДУ требует меньшего количество параметров, так как residual сеть должна учить каждый слой по отдельности, а здесь все «слои» связаны одним дифференциальным уравнением, вид которого оптимизируется;

- Решение задачи Коши методом Adams занимают константную память, в отличии от residual сетей, линейно зависящий от числа слоёв. Память является существенным ограничением на размер сети при вычислениях на современных устройствах.

Данные достоинства анализировались на примере данных MNIST. Авторам удалось показать, что при значительно меньшем количестве параметров удаётся достичь качества архитектуры ResNet.

3 Непрерывные нормализующие потоки

Теперь рассмотрим применение данного подхода к задаче построения нормализующих потоков.

3.1 Нормализующие потоки

Пусть дана выборка из сложного распределения, требуется научиться генерировать объекты из того же распределения, из которого была сгенерирована данная выборка.

Обычно для решения этой задачи генерируется выборка объектов z_0 из какого-то простого распределения, например, многомерного нормального. Далее рассматривается биективное преобразование, переводящее сгенерированные объекты в объекты из требуемого распределения. Вся сложность заключается в построении этого биективного преобразования, называемого нормализующим потоком (NF).

Запишем подробнее, пусть дан объект z_0 из распределения $p(z_0)$. Пусть функция $f(\cdot)$ биективно переводит z_0 в z_1: $f(z_0) = z_1$. Тогда плотности величин z_0 и z_1 связаны следующим соотношением:

$$\log p(z_1) = \log p(z_0) - \log \left| \det \frac{\partial f}{\partial z_0} \right|.$$

Доказательство этого факта можно найти в исходной работе, описывающей нормализующие потоки [4].

Можно заметить, что вычисление определителя якобиана $\det \frac{\partial f}{\partial z_0}$, в общем случае имеет кубическую сложность от размерности пространства. Решением этой проблемы является представление функции $f(\cdot)$ в виде цепочки биективных преобразований $f_t : z_t \longrightarrow z_{t+1}$ более простого вида. Например, было предложено преобразование следующего вида:

$$z(t+1) = z(t) + uh\big(w^T z(t) + b\big); \quad \log p(z(t+1)) = \log p(z(t)) - \log \left| 1 + u^T \frac{\partial h}{\partial z} \right|. \tag{4}$$

Данная процедура вычислительно более эффективна, однако в прикладных задачах может понадобиться довольно большая глубина сети (большая цепочка преобразований) для хорошего качества выучивания сложных распределений.

3.2 Предложенный подход

В уравнениях (4), так же как это было сделано для уравнения (1) можно перейти в непрерывной форме:

$$\frac{dz(t)}{dt} = uh\big(w^T z(t) + b\big); \quad \frac{\partial \log p(z(t+1))}{\partial t} = -u^T \frac{\partial h}{\partial z(t)}. \tag{5}$$

Зная начальное условие $p(z(0))$, мы снова решаем задачу Коши для системы уравнений (5) и находим $p(z(t_1))$.

При этом верность перехода от (4) к (5) гарантируется благодаря выполнению теоремы о том, что при задании $z(t)$ с помощью ОДУ вида:

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

и при выполнении некоторых ограничений на функцию f, изменение плотности распределения величины $z(t)$ удовлетворяет следующему уравнению:

$$\frac{\partial \log p(z(t))}{\partial t} = -\mathrm{tr}\left(\frac{df}{dz(t)}\right).$$

Доказательство этого утверждения можно найти в [3] в Appendix A. Данное представление потока называется непрерывным нормализующим потоком (CNF).

Если «глубокие» нормализующие потоки задаются в виде последовательных преобразований, то в непрерывном случае предлагается использовать «широкие» потоки. В этом случае функция задаётся в виде суммы:

$$f(z(t)) = \sum_{i=1}^{M} f_i(z(t)).$$

Вычисление плотности в силу линейности оператора «следа» tr, записывается в виде:

$$\frac{\partial \log p(z(t))}{\partial t} = -\sum_{i=1}^{M} \mathrm{tr}\left(\frac{df_i}{dz(t)}\right).$$

Таким образом, удаётся существенно понизить вычислительную сложность нормализующего потока.

Результат. В качестве демонстрации эффективности, авторы предлагают посмотреть на результаты обучения модели для модельных распределений рисунок 1. Видно, что результаты достигаются похожие по качеству, но при этом непрерывные нормализующие потоки сходятся в 50 раз быстрее по количеству итераций.

4 Непрерывная генеративная модель временного ряда

Обычно временные ряды обрабатываются рекуррентными нейросетевыми архитектурами вида encoder-decoder, которые по обозреваемой части временного ряда считают скрытое представление (кодировка), а на закрытой части предсказывают значения на основе полученного скрытого состояния (декодировка).

В данной работе предлагается способ построения модели со скрытыми переменными, описывающую генерацию временного ряда на основе дифференциальных уравнений.

Итак, пусть t_i — это значения времени, в которых известны значения временного ряда x_{t_i}. Пусть z_{t_i} — это скрытое представление временного ряда, изменяющееся непрерывно по закону:

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

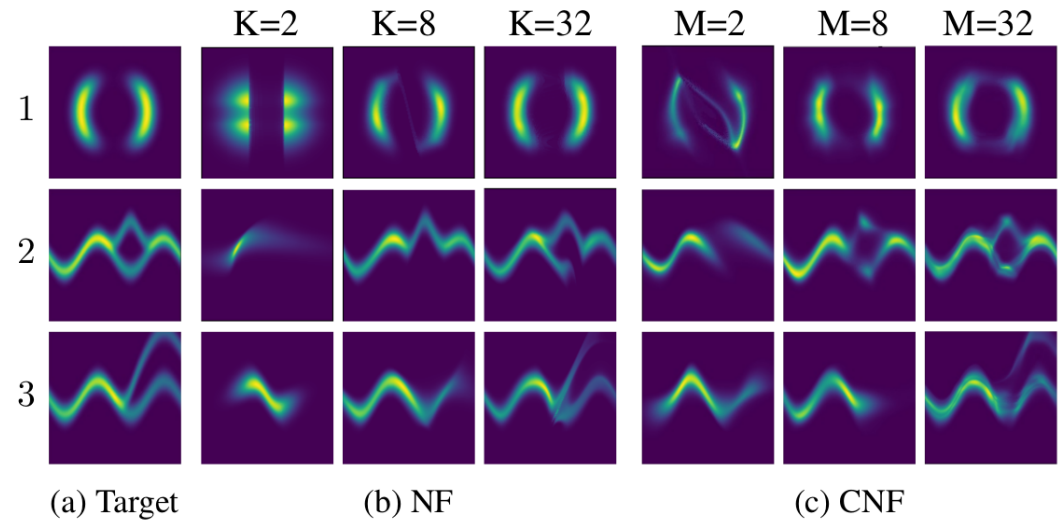

Рис. 1: Пример работы дискретных и непрерывных нормализующих потоков.
K — количество «глубоких» слоёв дискретного норм. потока (NF), M — «широких» слоёв непрерывного норм. потока (CNF).

тогда генеративную модель предлагается записывать так:

$$\begin{aligned} z_{t_0} &\sim p(z_{t_0}); \\ z_{t_1}, ..., z_{t_N} &= Solve(z_{t_0}, f, \theta_f, t_1, ..., t_N); \\ x_{t_i} &= p(x|z_{t_i}, \theta_x). \end{aligned}$$

Причём θ_x — задают параметры декодера.

В данной модели начальное скрытое состояние z_{t_0} генерируется из априорного распределения $p(z_{t_0})$. Для каждой точки t_i скрытое состояние z_{t_i} считается решением задачи Коши с начальным состоянием z_{t_0} и описанных выше дифференциальным уравнением. Далее с помощью декодера, параметризованного θ_x, из скрытого состояние z_{t_i} получают значение временного ряда x_{t_i}.

Понятно, что для генеративной модели необходимо знать, θ_f и θ_x, но как решать обратную задачу — оптимизировать параметры модели, зная из данных истинные значения ряда x_{t_i}? Для этого используются стандартные методы вариационного вывода (Variational Inference).

В вариационным выводе максимизируется нижняя оценка на достоверность модели (evidence lower bound или ELBo [5]), которая, в общем случае, для скрытых состояний Z, параметров φ и θ и известных величин X записывается следующим образом:

$$\int_Z q_\varphi(Z|X) \log p_\theta(X|Z) dZ - KL(q_\varphi(Z|X)||p(Z)) \longrightarrow \min_{\varphi,\theta}, \tag{6}$$

Интеграл первого слагаемого задаёт правдоподобие модели (возможность восстановления X из скрытого Z) и оценивается случайным сэмплированием одного скрытого предсталвения z. Таким образом по объекту x с помощью кодировщика строится распределение

(обычно $q(\cdot)$ задаётся гауссианой), из нее сэмплируется z, по которому снова восстанавливается x с помощью декодировщика.

Второе слагаемое соответствует регуляризатору, накладывающему ограничение априорного распределения на z.

Таким образом, в общем случае, вариационная модель напоминает обычный автокодировщик, однако она называется вариационным автокодировщиком (VAE), так как значение z считается сэмплированием из некоторого распределения, а не точечно.

В нашем случае вариационное распределение $q(z_{t_0}|\{x_{t_i}, t_i\})$ ищется в виде нормального с центром $\mu_{z_{t_0}}$ и дисперсией $\sigma_{z_{t_0}}$.

1. Таким образом, кодировщик по объектам x_{t_i} выдаёт распределение $q(\cdot)$ в виде двух величин: центра и дисперсии. Зная их, можно сэмплировать z_{t_0}. В качестве кодировщика при этом используется рекуррентная нейронная сеть RNN.

2. Далее, зная начальное условие z_{t_0}, вычисляются z_{t_i} решением задачи Коши с уравнением $\frac{dz}{dt} = f(z_0, \theta_f)$.

3. Далее вычисляется ELBo как было описано в (6) с использованием стандартного нормального распределения $\mathcal{N}(0, 1)$ в качестве априорного распределения $p(z_{t_0})$:

$$ELBo(\varphi, \theta_f, \theta_x) = \sum_i \log p(x_i|z_{t_i}, \theta_x) + \log p(z_{t_0}) - \log q(z_{t_0}|\{x_{t_i}, t_i\}_i, \varphi).$$

   При этом максимизация ELBo производится градиентным спуском с вычислением градиента в задаче Коши полностью аналогично описанному ранее способу для residual сетей.

Результат. Таким образом, получена модель, архитектурно представляющую из себя рекуррентную сеть и модуль по решению задачи Коши и при этом непрерывно моделирующую дискретный временной ряд.

Авторами было показано, что качество предсказаний такой модели значительно превосходит предсказания обычной рекуррентной сети, выученной на тех же данных.

5 Выводы

Таким образом, основными результатами статьи является общих подход, позволяющий от дискретных скрытых состояний переходить к непрерывным путём представления скрытого состояния как решения задачи Коши для обыкновенного дифференциального уравнения.

Также показаны примеры применения данного подхода к различным стандартным задачам машинного обучения: residual архитектурам, нормализующим потокам и моделям временных рядов с латентными представлениями. Во всех подходах удалось за счёт перехода к непрерывным представлениям либо сократить число параметров, либо увеличить скорость обучения, либо повысить качество решения задачи.

Список литературы

[1] *Kaiming He and Xiangyu Zhang and Shaoqing Ren and Jian Sun.* (2015) Deep Residual Learning for Image Recognition. CoRR

[2] *Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.* (2017). Attention Is All You Need. NIPS.

[3] *Chen, Tian Qi and Rubanova, Yulia and Bettencourt, Jesse and Duvenaud, David K* (2018) Neural Ordinary Differential Equations. Advances in Neural Information Processing Systems 31, 6571–6583 NIPS2018

[4] *Danilo Jimenez Rezende and Shakir Mohamed.* (2015) Variational inference with normalizing flows. Volume 37 (ICML'15), Francis Bach and David Blei (Eds.), Vol. 37. JMLR.org 1530-1538.

[5] *Xitong Yang* (2017) Understanding the Variational Lower Bound.