Содержание

1 Введение

В машинном обучении широко используются рекуррентные нейронные сети (RNN), остаточные нейронные сети (ResNet), а также нормализованные потоки. Все эти алгоритмы имеют общую особенность. Вычисление в них делится на шаги, и зависимость следующего шага от предыдущего описывается формулой:

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

Так например, для ResNet: $f(h_t, \theta_t)$ — t-ый слой нейронной сети, h_t — его вход, h_{t+1} —выход. Для предотвращения затухания или взрыва градиента к f прибавляется h.

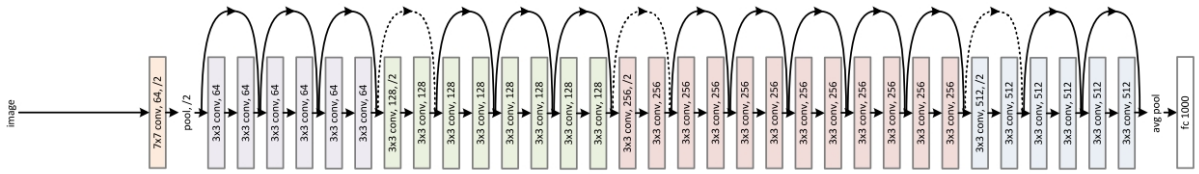

Рис. 1: Аритектура остаточной нейронной сети

Если вспомнить численные методы, то в формуле (1) можно увидеть хорошо известный численный способ решения задачи Коши — метод Эйлера.

Задача Коши выглядит так:

$$\begin{cases} \frac{du}{dt} = f(t, u(t)), \qquad t > 0 \\ u(0) = u_0 \end{cases}$$

Для использования метода Эйлера необходимо разбить прямую t с помощью сетки $w_\tau = \{t_n = n\tau | n = 0, 1, 2, ...\}$:

$$\begin{cases} \frac{y_{n+1} - y_n}{\tau} = f(t_n, y_n), \qquad t_n \in w_\tau \\ y_0 = u_0 \end{cases}$$

Таким образом получается что ResNet — метод эйлера с шагом 1. Попробуем в задаче Коши заменить функцию f нейронной сетью. θ — это её параметры. Причем эта функция зависит не только от входа, но и от t. Задаём $h(0)$ — вход новой сети, $h(T)$ — её выход. Как её обучить?

2 Обучение

2.1 Сопряженное уравнение

Пусть дано уравнение:

$$\frac{dz(t)}{dt} = f(z(t), t, \theta), \qquad z(t_0) = z_0 \tag{2}$$

Как найти производную функции потерь, зависящую от выхода $z(T)$, по параметрам f? Если смотреть на f как на непрерывную функцию, то для оптимизации функции потерь L можно ввести сопряженную переменную $a(t)$. По определению она равна:

$$a(t) = \frac{\partial L}{\partial z(t)} \tag{3}$$

Попробуем найти уравнение, задающее изменение $a(t)$ от времени t.

$$\begin{aligned}
\frac{\partial L}{\partial z(t)} &= \frac{\partial L}{\partial z(t+\epsilon)} \frac{z(t+\epsilon)}{\partial z(t)} = a(t+\epsilon)\frac{z(t+\epsilon)}{\partial z(t)} \\
\frac{dz(t+\epsilon)}{dz(t)} &\approx \frac{d}{dz(t)}\big(z(t) + \epsilon f(z(t), t, \theta)\big) = 1 + \epsilon \frac{\partial}{\partial z(t)} f(z(t), t, \theta) \\
a(t) &\approx a(t+\epsilon)\left(1 + \epsilon \frac{\partial}{\partial z(t)} f(z(t), t, \theta)\right) \\
\frac{da(t)}{dt} &= \lim_{\epsilon \to 0} \frac{a(t+\epsilon) - a(t)}{\epsilon} = -\lim_{\epsilon \to 0} \frac{\epsilon a(t+\epsilon)\frac{\partial}{\partial z(t)} f(z(t), t, \theta)}{\epsilon} \\
&= -a(t)\frac{\partial}{\partial z(t)} f(z(t), t, \theta)
\end{aligned}$$

В итоге имеем сопряженное диффиренциальное уравнение. Решая его численными методами, можно находить значения $a(t)$ в различных точках:

$$\frac{da(t)}{dt} = -a(t)\frac{\partial}{\partial z(t)} f(z(t), t, \theta), \qquad a(T) = \frac{\partial L}{\partial z(T)} \tag{4}$$

2.2 Смысл сопряженного уравнения

Смысл тесно связан с методом обратного распространения ошибки для дискретного случая. Рассмотрим дискретную модель ResNet, $z(t+1)$ — выход слоя t:

$$z(t+1) - z(t) = f_t(z(t), \theta_t) \tag{5}$$

Для неё вводим такую же сопряженную переменную, которая имеет смысл производной ошибки по выходу слоя $t-1$:

$$a(t) = \frac{\partial L}{\partial z(t)}$$

Обычный chain-rule, используемый в методе обратного распространения ошибки:

$$a(t) = a(t+1)\frac{dz(t+1)}{dz(t)}$$

Отсюда, учитывая (5), приращение a(t) описывается так:

$$a(t+1) - a(t) = -a(t+1)\frac{\partial f_t(z(t), \theta_t)}{\partial z(t)} \tag{6}$$

А так, из формулы (4) для непрерывного $a(t)$:

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

Таким образом выражение (4) является обобщением для выражения (6) для сети с континуумом слоёв, с условием, что каждый слой не сильно меняет сигнал.

2.3 Оптимизация функции потерь

Чтобы обновлять параметры нейронной сети f, следует найти производную $\frac{\partial L}{\partial \theta}$. Для этого следует ввести дополнительные сопряженные переменные и векторную функцию f_{aug}:

$$\frac{\partial \theta(t)}{\partial t} = \mathbf{0}, \qquad \frac{\partial t(t)}{\partial t} = 1$$

$$\frac{d}{dt}\begin{bmatrix} z \\ \theta \\ t \end{bmatrix}(t) = f_{aug}\left([z, \theta, t]\right) = \begin{bmatrix} f(z, \theta, t) \\ \mathbf{0} \\ 1 \end{bmatrix} \tag{7}$$

$$a_{aug} = \begin{bmatrix} a \\ a_\theta \\ a_t \end{bmatrix}, \qquad a_\theta(t) = \frac{dL}{d\theta(t)}, \qquad a_t(t) = \frac{dL}{dt(t)} \tag{8}$$

Якобиан для векторной функции f_{aug} выглядит так:

$$\frac{\partial f_{aug}}{\partial\left[z, \theta, t\right]} = \begin{bmatrix} \frac{\partial f}{\partial z} & \frac{\partial f}{\partial \theta} & \frac{\partial f}{\partial t} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}(t)$$

Тогда для уравнения (7), и его сопряжения (8), как было показано выше, верно выражение (4):

$$\frac{da_{aug}(t)}{dt} = -\begin{bmatrix} a(t) & a_\theta & a_t \end{bmatrix} \cdot \frac{\partial f_{aug}}{\partial\left[z, \theta, t\right]}(t) = -\begin{bmatrix} a\frac{\partial f}{\partial z} & a\frac{\partial f}{\partial \theta} & a\frac{\partial f}{\partial t} \end{bmatrix}(t) \tag{9}$$

Первая компонента в получившемся векторном уравнении, является выражением (4), а из второй компоненты можно найти искомую производную $\frac{\partial L}{\partial \theta}$. Для этого выражение достаточно проинтегрировать по t на участке $[t_0, T]$, учитывая, что $a_\theta(T) = 0$:

$$\frac{\partial L}{\partial \theta} = \int_{t_0}^{T} a(t)\frac{\partial f(z(t), t, \theta)}{\partial \theta} dt \tag{10}$$

Для получения конкретного значения интеграла можно использовать численные методы.

Есть и еще один способ обучать модель. Можно не вводить сопряженные переменные, а просто пробрасывать градиент через численные методы, устройство которых позволяет делать обычное обратное распространение ошибки. Правда при таком способе не будет одного из преимуществ ODE-Net — эффективности по памяти.

3 Преимущества описанного метода

Эффективность по памяти. Введение сопряженных переменных позволяет не использовать стандартный метод обратного распространения ошибки. Таким образом для вычисления производной, требуется O(1) памяти, если рассматривать объём памяти как функцию от глубины архитектуры.

Адаптивность модели. Метод Эйлера является, пожалуй, самым простым методом решения задачи Коши. Но есть и более эффективные и точные численные методы (метод Рунге-Кутта).

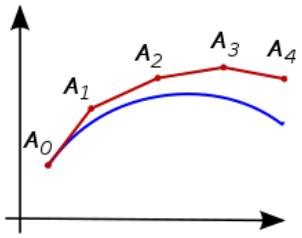

Рис. 2: Ломаная Эйлера (красная линия) — приближённое решение в пяти узлах задачи Коши и точное решение этой задачи (выделено синим цветом).

Более того сами численные методы также являются адаптивными. У них можно менять шаг по сетке. Чем меньше шаг сетки, тем ближе численное решение к точному, но тем больше время вычисления.

Возникает возможность, которой не было в классических остаточных сетях: можно варьировать масштаб сетки, получая некоторое соотношение между точностью и скоростью работы. Более того можно использовать разный размер шага сетки при обучении и при тестировании.

Таким образом, если необходимо запустить описанные вычисления при ограниченной мощности, например, на мобильном устройстве, нужно лишь увеличить шаг сетки. Это преимущество также неободимо для систем реального времени, где вычисления ограничены по времени.

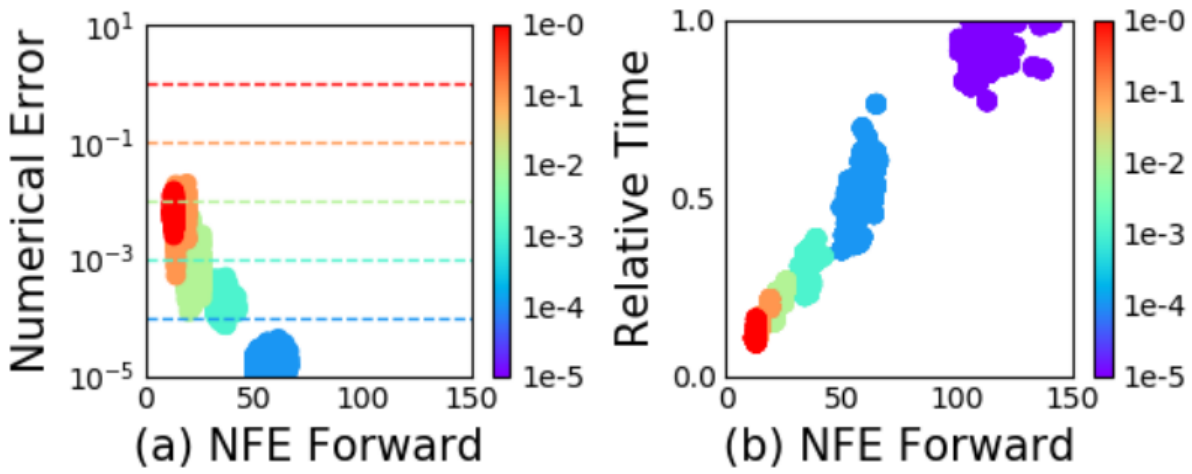

Рис. 3: Графики из исходной статьи. NFE(number of function evaluations) — показывает объем необходимых вычислений с разными сетками: а) зависимость ошибки от NFE, б) зависимость времени вычисления от NFE.

Меньшее число параметров. В эксперименте было показано, что модель ODE-Net, достигает той же точности, что и ResNet, при этом число параметров почти в 3 раза меньше.

	Test Error	# Params	Memory	Time
1-Layer MLP†	1.60%	0.24 M	-	-
ResNet	0.41%	0.60 M	$\mathcal{O}(L)$	$\mathcal{O}(L)$
RK-Net	0.47%	0.22 M	$\mathcal{O}(\tilde{L})$	$\mathcal{O}(\tilde{L})$
ODE-Net	0.42%	0.22 M	$\boldsymbol{\mathcal{O}(1)}$	$\mathcal{O}(\tilde{L})$

Рис. 4: Таблица эксперимента из исходной статьи. L показывает число слоёв сети. ResNet состоит из 6 стандартных Residual блоков. RK-Net — сеть использующая в качестве численного метода метод Рунге-Кутта без введения сопряженных переменных. ODE-Net — сеть с аналогичной RK-Net архитектурой, но оптимизируемая с помощью введения сопряженных переменных.

Масштабируемые и обратимые нормализующие потоки. В современных генеративных моделях остро стоит проблема mode collapsing. Генератор выдает ограниченное количество разных образцов, не покрывая даже обучающую выборку. Это связано с тем, что GAN реализует неявную вероятностную модель, то есть доступа к плотности нет. Это сделано для возможности приближения сложных распределений, при этом распределение моделируется нейронной сетью. Но такая нейронная сеть может лишь сэмплировать точки, не выдавая их плотности.

Чтобы строить сложные распределения и иметь доступ к плотности, можно брать случайную величину с простым распределением и преобразовывать её с помощью непрерывных и обратимых преобразований. Тогда для вычисления в точке плотности нового распределения необходимо знать только плотность в этой точке старого распределения и якобиан преобразования. Вычисление якобиана имеет сложность $O(D^3)$, где D — размерность пространства. Имеются преобразования с сильными ограничениями, вычисление якобиана которых имеет меньшую сложность. Но из-за ограничений этих преобразований получение сложных распределений из простых становится трудоёмким.

Модель, представленная в статье, позволяет находить плотность в точке нового распределения за линейное время. Более того, нет ограничений на преобразования. Единственное ограничение: шаги по t должны быть достаточно маленькими.

4 Эксперимент

В файле Notebook.ipynb приведён эксперимент, аналогичный эксперименту из статьи на выборке MNIST. Изменяя параметры в ячейке с параметрами и запуская эксперимент заново, можно исследовать поведение ODE-Net.