Invertible Residual Networks

Иван Провилков

20 декабря 2019 г.

1 Введение

Invertible Residual Network – архитектура, работающая хорошо одновременно как на дискриминативных, так и на генеративных задачах. Авторы концентрируются на нормализующих потоках и показывают, что заменив нормализацию в ResNet-ах, можно достичь их обратимости.

2 Нормализующие потоки

Нормализующий поток это алгоритм, который преобразует плотность начального распределение серией обратимых преобразований в плотность другого распределения.

Например, если мы генерируем x, то можем фактаризовать логарифм правдоподобия:

$$\log p_\theta(x) = \log \int p(x|z)p(z)dz = \log \int \frac{q_\phi(z|x)}{q_\phi(z|x)} p(x|z)p(z)dz \geq \mathbb{D}_{KL}[q_\phi(z|x)||p(z)] + \mathbb{E}_q[\log p_\theta(x|z)] = -\mathbb{F}(x),$$

где θ - параметры модели, z - скрытые переменные q_ϕ - аппроксиматор скрытых переменных. $\mathbb{F}$ (ELBO) – evidence lower bound, нижняя вариационная оценка. Во время обучения можно оптимизировать ELBO используя различные методы.

3 Обратимость

Авторы рассматривают ResNet как Эйлерову дискретизацию обыкновенного дифференциального уравнения:

$$x_{t+1} = x_t + g_{\theta_t}(x_t)$$
$$x_{t+1} = x_t + hf_{\theta_t}(x_t),$$

где g - residual block, h - step size.

Их интересует обратная динамика:

$$x_t = x_{t+1} - g_{\theta_t}(x_t)$$
$$x_t = x_{t+1} - hf_{\theta_t}(x_t),$$

решение такой динамики позволило бы Residual block-у работать в обратную сторону.

Следующая теорема накладывает достаточные условия для того, чтобы ResNet block был обратимым:

Theorem 1 *Пусть $F_\theta : \mathbb{R}^d \to \mathbb{R}^d$ и пусть $F_\theta = (F_\theta^1 \cdot ... \cdot F_\theta^T)$ обозначет ResNet с блоками $F_\theta^t = I + g_{\theta_t}$. Тогда, если $Lip(g_{\theta_t}) < 1, t = 1, ..., T$, тогда ResNet обратим. Где Lip(.) - означает константу Липшица.*

Так как аналитическую форму обратной функции найти сложно, а правая часть уравнения обратной динамики является сжимающей в нашем случае, то авторы используют метод простой итерации для обращения ResNet блока:

Algorithm 1. Inverse of i-ResNet layer via fixed-point iteration.

Input: output from residual layer y, contractive residual block g, number of fixed-point iterations n
Init: $x^0 := y$
for $i = 0, \dots, n$ **do**
  $x^{i+1} := y - g(x^i)$
end for

.

Согласно теореме Банаха такая итерация имеет экспоненциальную скорость сходимости.

Для того, чтобы ограничение на константу Липшица выполнялось, достаточно сделать спектральную норму весов сверток меньше 1: $g = W_3 f(W_2(f(W_1))$, $Lip(g) < 1$, if $||W_i||_2 < 1$. Авторы аппроксимируют спектральную норму с помощью метода степенной итерации, а затем нормализуют матрицу используя полученное значение $\sigma_i \leq ||W_i||_2$.

$W_i^{new} = \frac{cW_i}{\sigma_i} I(\frac{c}{\sigma_i} < 1) + W_i I(\frac{c}{\sigma_i} \geq 1)$, где c – гиперпараметр. Метод не дает полной гарантии того, что $||W_i||_2 \leq c$, однако авторы делали точный подсчет нормы, и это ограничение на Липшицевость выполнялось в экспериментах.

4 Генеративная модель

Чтобы сгенерировать x, вначале генерируется другое распределение $z \sim p_z(z)$, а затем применяется функция $F : x = F(z)$. Для любого x можно рассчитать правдоподобие с помощью формулы замены переменных:

$$\ln p_x(x) = \ln p_z(z) + \ln|\det J_{F^{-1}}(X)|,$$

где $J_{F^{-1}}$ – Якобиан обратной функции к F.

Так как iResNet обратим, то мы можем использовать его как параметризацию F^{-1}. Мы можем сэмплить $z \sim p(z)$, а затем считать $x = F(z)$.

Главной проблемой в этом процессе является подсчет $\ln|\det J_{F^{-1}}(X)|$, так как явное вычисление этой величины требует $O(d^3)$ времени, где d – размерность переменной.

С помощью нескольких лемм авторы показывают, что $\ln|det J_{F^{-1}}(X)| = tr(\ln J_{F^{-1}})$, в нашем случае $\ln p_x(x) = \ln p_z(z) + tr(\ln(I + J_g(x)))$.

Нужное нам выражение может быть записано в форме ряда:

$$tr(\ln(I + J_g(x))) = \sum_{k=1}^{\infty} (-1)^{k+1} \frac{tr(J_g^k)}{k},$$

который сходится при $||J_g||_2 < 1$. Это условие следует из Липшицевости.

Аппроксимируя первые члены этого ряда авторы и считают логарифм детерминанта.

Algorithm 2. Forward pass of an invertible ResNets with Lipschitz constraint and log-determinant approximation, SN denotes spectral normalization based on (2).

Input: data point x, network F, residual block g, number of power series terms n
for Each residual block **do**
  Lip constraint: $\hat{W}_j := \text{SN}(W_j, x)$ for linear Layer W_j.
  Draw v from $\mathcal{N}(0, I)$
  $w^T := v^T$
  $\ln\det := 0$
  for $k = 1$ **to** n **do**
    $w^T := w^T J_g$ (vector-Jacobian product)
    $\ln\det := \ln\det + (-1)^{k+1} w^T v / k$
  end for
end for

.

5 Результаты

Результаты на дискриминативной задаче:

Invertible Residual Networks

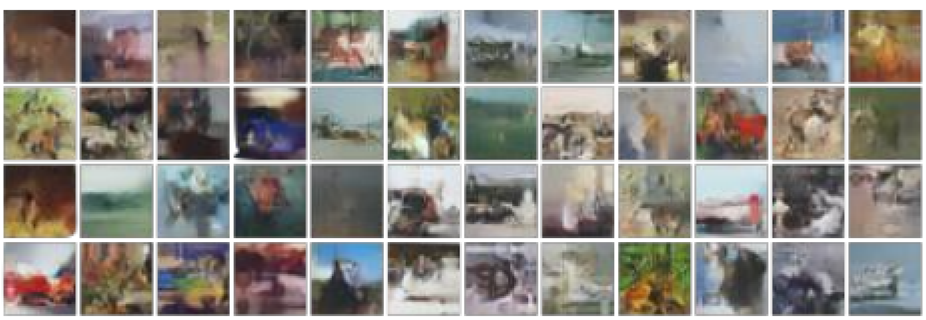
		ResNet-164	Vanilla	$c = 0.9$	$c = 0.8$	$c = 0.7$	$c = 0.6$	$c = 0.5$
Classification	MNIST	-	0.38	0.40	0.42	0.40	0.42	0.86
Error %	CIFAR10	5.50	6.69	6.78	6.86	6.93	7.72	8.71
	CIFAR100	24.30	23.97	24.58	24.99	25.99	27.30	29.45
Guaranteed Inverse		**No**	**No**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**

Результаты на генеративной задаче:

Method	MNIST	CIFAR10
NICE (Dinh et al., 2014)	4.36	4.48†
MADE (Germain et al., 2015)	2.04	5.67
MAF (Papamakarios et al., 2017)	1.89	4.31
Real NVP (Dinh et al., 2017)	1.06	3.49
Glow (Kingma & Dhariwal, 2018)	1.05	3.35
FFJORD (Grathwohl et al., 2019)	0.99	3.40
i-ResNet	1.06	3.45

Table 4. MNIST and CIFAR10 bits/dim results. † Uses ZCA pre-processing making results not directly comparable.

Примеры картинок, сгенерированных при обучении на CIFAR10:



6 Заключение

В этой статье авторы представили iResNet – архитектуру, основанную на нормализующем потоке и позволяющую при небольших ограничениях на слои делать интерпретируемую генеративную модель. Эта же модель хорошо показывает себя на дискриминативной задаче, не сильно уступая бейзлайну.