Deep equilibrium model

Лебедев Михаил Алексеевич

Основная идея Deep sequence model

На вход подается $\mathbf{x}_{1:T} = [x_1, \ldots, x_T] \in \mathbb{R}^{T \times p}$

$$\mathbf{z}_{1:T}^{[i+1]} = f_\theta^{[i]}\left(\mathbf{z}_{1:T}^{[i]}; \mathbf{x}_{1:T}\right) \quad \text{for } i = 0, 1, 2, \ldots, L-1$$

T – длина последовательности

Sequence model – модель $G(\mathbf{x}_{1:T}) = \mathbf{y}_{1:T} = \in \mathbb{R}^{T \times \tilde{q}}$

Выходные данные y зависят только от первых x_1,…,x_t и не зависят от последующих x_t+1, …,x_T.

Проблема

При обучении глубинной сети нам нужно много памяти для backpropagation. Но мы знаем, что сети сходятся. Нам проще найти точку эквилибриума, хранить в памяти только ее.

$$\lim_{i\to\infty} \mathbf{z}_{1:T}^{[i]} = \lim_{i\to\infty} f_\theta\left(\mathbf{z}_{1:T}^{[i]}; \mathbf{x}_{1:T}\right) \equiv f_\theta\left(\mathbf{z}_{1:T}^{\star}; \mathbf{x}_{1:T}\right) = \mathbf{z}_{1:T}^{\star}$$

Forward pass

В отличие от сверточных сетей, мы используем точку эквилибриума вместо значения активации. Для того чтобы нам ее найти, мы решаем следующее уравнение с помощью квазиньютоновских методов.

$$g_\theta(\mathbf{z}^\star_{1:T}; \mathbf{x}_{1:T}) = f_\theta(\mathbf{z}^\star_{1:T}; \mathbf{x}_{1:T}) - \mathbf{z}^\star_{1:T} \to 0.$$

$$\mathbf{z}^{[i+1]}_{1:T} = \mathbf{z}^{[i]}_{1:T} - \alpha B g_\theta(\mathbf{z}^{[i]}_{1:T}; \mathbf{x}_{1:T}) \quad \text{for } i = 0, 1, 2, \ldots$$

Backward pass

Здесь мы используем следующую теорему

Theorem 1. ***(Gradient of the Equilibrium Model)*** *Let $\mathbf{z}^\star_{1:T} \in \mathbb{R}^{T\times d}$ be an equilibrium hidden sequence with length T and dimensionality d, and $\mathbf{y}_{1:T} \in \mathbb{R}^{T\times q}$ the ground-truth (target) sequence. Let $h : \mathbb{R}^d \to \mathbb{R}^q$ be any differentiable function and let $\mathcal{L} : \mathbb{R}^q \times \mathbb{R}^q \to \mathbb{R}$ be a loss function (where $h, \mathcal{L}$ are applied in a vectorized manner) that computes*

$$\ell = \mathcal{L}(h(\mathbf{z}^\star_{1:T}), \mathbf{y}_{1:T}) = \mathcal{L}(h(\mathsf{RootFind}(g_\theta; \mathbf{x}_{1:T})), \mathbf{y}_{1:T}). \tag{7}$$

Then the loss gradient w.r.t. $(\cdot)$ (for instance, θ or $\mathbf{x}_{1:T}$) is

$$\frac{\partial \ell}{\partial (\cdot)} = -\frac{\partial \ell}{\partial \mathbf{z}^\star_{1:T}} \left(J^{-1}_{g_\theta}\big|_{\mathbf{z}^\star_{1:T}}\right) \frac{\partial f_\theta(\mathbf{z}^\star_{1:T}; \mathbf{x}_{1:T})}{\partial (\cdot)} = -\frac{\partial \ell}{\partial h}\frac{\partial h}{\partial \mathbf{z}^\star_{1:T}} \left(J^{-1}_{g_\theta}\big|_{\mathbf{z}^\star_{1:T}}\right) \frac{\partial f_\theta(\mathbf{z}^\star_{1:T}; \mathbf{x}_{1:T})}{\partial (\cdot)}, \tag{8}$$

where $J^{-1}_{g_\theta}\big|_{\mathbf{x}}$ is the inverse Jacobian of g_θ evaluated at $\mathbf{x}$.

The proof is provided in Appendix A. The insight provided by Theorem 1 is at the core of our method and its various benefits. Importantly, the backward gradient through the "infinite" stacking can be represented as one step of matrix multiplication that involves the Jacobian at equlibrium. For instance, an SGD update step on model parameters θ would be

$$\theta^+ = \theta - \alpha \cdot \frac{\partial \ell}{\partial \theta} = \theta + \alpha \frac{\partial \ell}{\partial \mathbf{z}^\star_{1:T}} \left(J^{-1}_{g_\theta}\big|_{\mathbf{z}^\star_{1:T}}\right) \frac{\partial f_\theta(\mathbf{z}^\star_{1:T}; \mathbf{x}_{1:T})}{\partial \theta}. \tag{9}$$

Далее мы решаем уравнение для нахождения произведения Якобиана на вектор(решаем с помощью метода Бройдена)

$$J^{-1}_{g_\theta}\Big|_{\mathbf{z}^{[i+1]}_{1:T}} \approx B^{[i+1]}_{g_\theta} = B^{[i]}_{g_\theta} + \frac{\Delta \mathbf{z}^{[i+1]} - B^{[i]}_{g_\theta} \Delta g^{[i+1]}_\theta}{\Delta \mathbf{z}^{[i+1]\top} B^{[i]}_{g_\theta} \Delta g^{[i+1]}_\theta} \Delta \mathbf{z}^{[i+1]\top} B^{[i]}_{g_\theta},$$

Свойства

Эффективное использование памяти – для deep equilibrium model используется константное количество памяти(не зависит от глубины сети)

Выбор функции для backforward – любая ограниченная и гладкая функция.

Trellis network

Trellis network – temporal convolutional network, которая сочетает в себе свойства реккурентных и сверточных нейронных сетей.

$$\tilde{\mathbf{x}}_{1:T} = \text{Input injection (i.e., linearly transformed inputs by Conv1D}(\mathbf{x}_{1:T}; W_x))$$
$$f_\theta(\mathbf{z}_{1:T}; \mathbf{x}_{1:T}) = \psi(\text{Conv1D}([\mathbf{u}_{-(k-1)s:}, \mathbf{z}_{1:T}]; W_z) + \tilde{\mathbf{x}}_{1:T})$$

Weight-tied transformer

Multi-head self-attention transformer со следующими свойствами

1) производится регуляризация данных для увеличения обобщающей способности

2) размер модели меньше

3) сеть можно развернуть на любую глубину

$$\tilde{\mathbf{x}}_{1:T} = \text{Input injection (i.e., linearly transformed inputs by } \mathbf{x}_{1:T}W_x)$$
$$f_\theta(\mathbf{z}_{1:T}; \mathbf{x}_{1:T}) = \mathsf{LN}(\phi(\mathsf{LN}(\mathsf{SelfAttention}(\mathbf{z}_{1:T}W_{QKV} + \tilde{\mathbf{x}}_{1:T}; \mathrm{PE}_{1:T}))))$$

Источники

https://arxiv.org/pdf/1909.01377.pdf