Higher School of Economics

Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention

Колесников Георгий
2021

Введение

Трансформеры очень эффективны на ряде задач, однако имеют квадратичную сложность относительно длины вводимых данных. Поэтому они очень медленные на больших данных.

В этом докладе рассмотрен подход, который снижает сложность с $O(n^2)$ до $O(n)$.

Итеративная имплементация позволяет получить значительный прирост производительности и показывает сходство трансформеров с рекуррентными нейронными сетями.

Существующие ускорения трансформеров

$O(n\sqrt{n})$ - (Child et al., 2019) - sparse factorization.

$O(n \log(n))$ - (Kitaev et al., 2020) - LSH с последующим уменьшением необходимых умножений матриц. Однако данный метод не может быть использован в случае различных размерностей ключей и запросов. Предложенный линейный трансформер не имеет таких ограничений.

Transformer

Имеем $x \in \mathbb{R}^{N \times F}$, то есть N объектов размера F.

Трансформер $T : \mathbb{R}^{N \times F} \to \mathbb{R}^{N \times F}$, то есть композицию L слоев $T_1(\cdot), ..., T_L(\cdot)$.

$$T_l(x) = f_l(A_l(x) + x).$$

$f_l(\cdot)$ - преобразует признаки независимо друг от друга, обычно двухслойная нейронная сеть с прямой связью.

$A_l(\cdot)$ - преобразует признаки зависимо друг от друга. Входная последовательность x проецируется тремя матрицами $W_q \in \mathbb{R}^{F \times D}$, $W_k \in \mathbb{R}^{F \times D}$ и $W_v \in \mathbb{R}^{F \times M}$.

$$\begin{aligned} Q &= xW_Q, \\ K &= xW_K, \\ V &= xW_V, \end{aligned}$$

$$A_l(x) = V' = \text{softmax}\left(\frac{QK^T}{\sqrt{D}}\right)V.$$

Transformer

Введем функцию $sim(q,k) = exp(\frac{q^T k}{\sqrt{D}})$, тогда:

$$V_i' = \frac{\sum_{j=1}^{N} \text{sim}\left(Q_i, K_j\right) V_j}{\sum_{j=1}^{N} \text{sim}\left(Q_i, K_j\right)}.$$

Заметим, что единственное ограничение, которое мы должны наложить на $sim(\cdot)$, чтобы данное выражение было attention function - ее неотрицательность. Также как и во всех ядрах $k(x,y) : \mathbb{R}^{2\times F} \to \mathbb{R}_+$.

Перепишем выражение с помощью ядра $\phi(x)$:

$$V_i' = \frac{\sum_{j=1}^{N} \phi\left(Q_i\right)^T \phi\left(K_j\right) V_j}{\sum_{j=1}^{N} \phi\left(Q_i\right)^T \phi\left(K_j\right)},$$

Преобразование

Используем ассоциативность матриц:

$$V_i' = \frac{\phi\left(Q_i\right)^T \sum_{j=1}^{N} \phi\left(K_j\right) V_j^T}{\phi\left(Q_i\right)^T \sum_{j=1}^{N} \phi\left(K_j\right)}.$$

Видно, что исходное выражение имеет сложность $O(N^2)$, а выведенное за счет однократного подсчета $\sum_{j=1}^{N} \phi(K_j)V_j^T$ и $\sum_{j=1}^{N} \phi(K_j)$ лишь $O(N)$.

Общая сложность

Общая сложность **softmax** подхода составляет $O(N^2 \max(D, M))$

D - размерность запросов и ключей

M - размерность значений

Общая сложность **linear attention** подхода составляет $O(NCM))$

C - подсчет feature maps

M - размерность значений

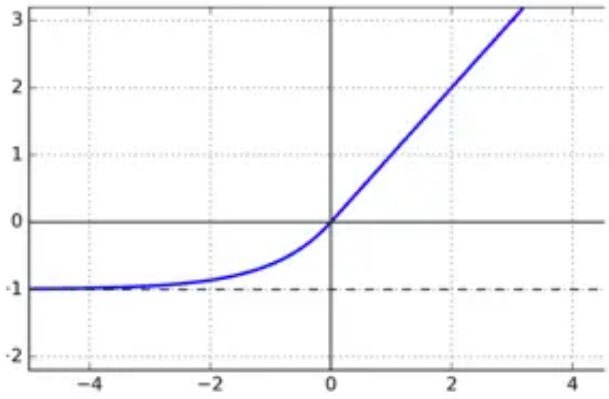
Например, для **полиномиального ядра степени 2** потребуется $O(ND^2M)$ операций сложения и умножения. Что хорошо на выборка, где $N > D^2$, и это зачастую выполняется, когда мы используем десятки тысяч объектов.

Общая сложность

Для работы с выборками поменьше будем использовать $elu(\cdot)$:

$$\phi(x) = \mathbf{elu}(x) + 1,$$

С этой функцией общая сложность составляет $O(NDM)$.



ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

Causal masking

Перестроим теперь модель так, чтобы на каждый объект влияли только предыдущие, а не все:

$$V_i' = \frac{\sum_{j=1}^{i} \text{sim}(Q_i, K_j) V_j}{\sum_{j=1}^{i} \text{sim}(Q_i, K_j)}.$$

$$V_i' = \frac{\phi(Q_i)^T \sum_{j=1}^{i} \phi(K_j) V_j^T}{\phi(Q_i)^T \sum_{j=1}^{i} \phi(K_j)}.$$

$$S_i = \sum_{j=1}^{i} \phi(K_j) V_j^T,$$

$$Z_i = \sum_{j=1}^{i} \phi(K_j),$$

$$V_i' = \frac{\phi(Q_i)^T S_i}{\phi(Q_i)^T Z_i}.$$

Заметим, что S_i и Z_i могут быть вычеслены за константное время из S_{i-1}, Z_{i-1}

Общая сложность вычислительная - $O(NCM)$, затраты памяти - $O(N \max(C, M)$.

Обучение и предсказание

Преимущество трансформеров (чего не может RNN): возможность распараллеливания вычислений при обучении.

Недостаток трансформеров на последовательных данных (если предсказание является следующим входным объектом), приходится считать всю модель целиком. В RNN такого недостатка нет.

Linear Transformer позволяет получить преимущества обоих подходов, потеряв недостатки.

Transformers are RNNs

Трансформер с causal masking может быть написан как RNN.

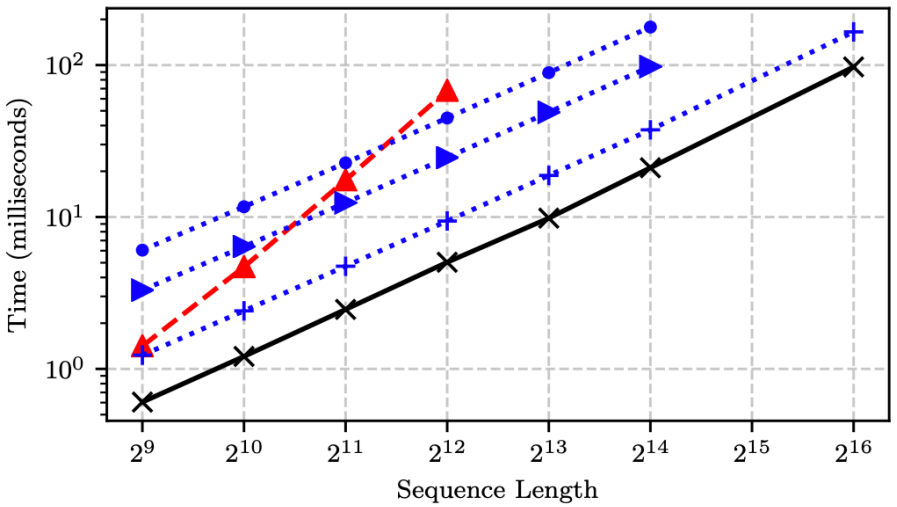
s - attention memory

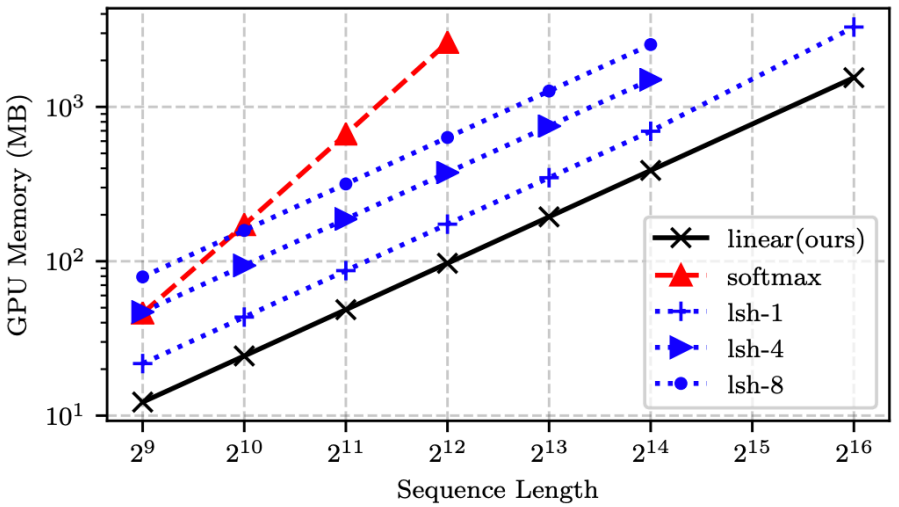
z - normalizer memory

$$
\begin{aligned}
s_0 &= 0, \\
z_0 &= 0, \\
s_i &= s_{i-1} + \phi\left(x_i W_K\right)\left(x_i W_V\right)^T, \\
z_i &= z_{i-1} + \phi\left(x_i W_K\right), \\
y_i &= f_l\left(\frac{\phi\left(x_i W_Q\right)^T s_i}{\phi\left(x_i W_Q\right)^T z_i} + x_i\right).
\end{aligned}
$$

Сравнение скорости и затрат памяти

softmax - standart PyTorch implementation, lsh-X - PyTorch reformer





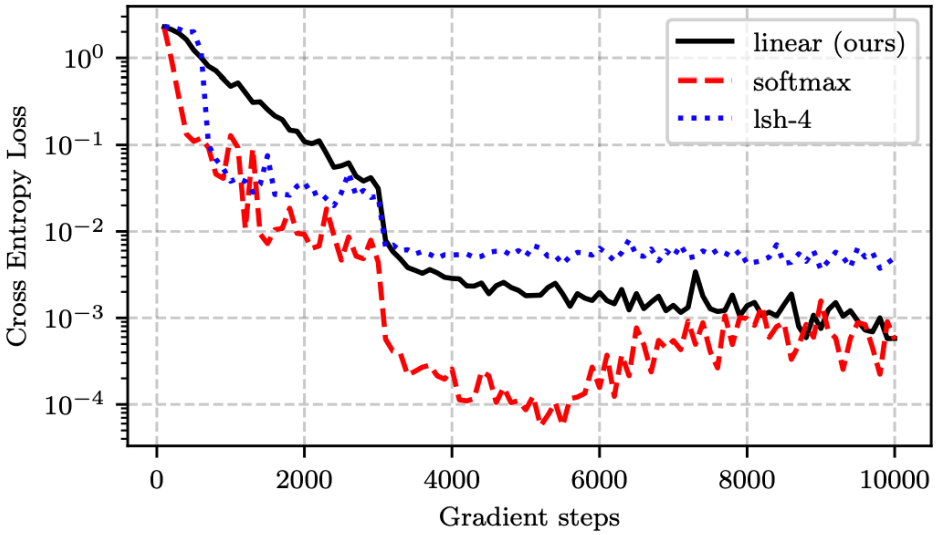


Image generation

MNIST

Method	Bits/dim	Images/sec	
Softmax	0.621	0.45	(1×)
LSH-1	0.745	0.68	(1.5×)
LSH-4	0.676	0.27	(0.6×)
Linear (ours)	0.644	**142.8**	**(317×)**

Unconditional samples

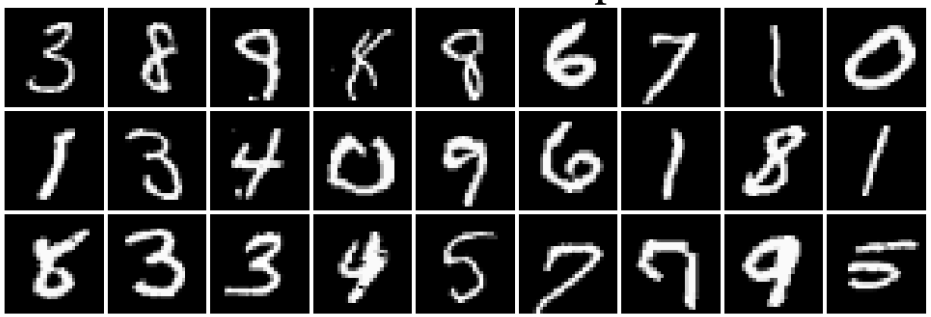
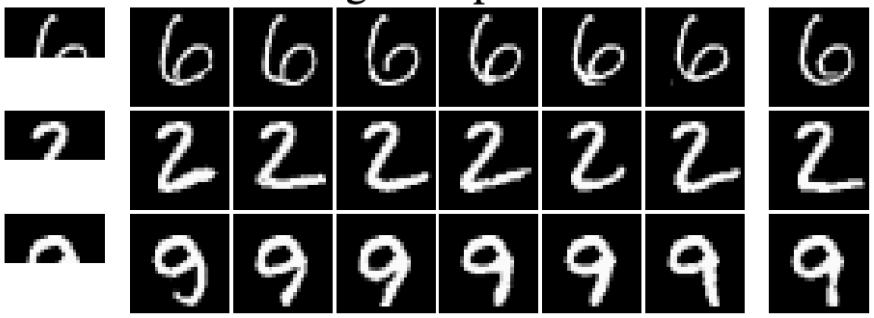


Image completion



(a) (b) (c)

CIFAR-10

Method	Bits/dim	Images/sec	
Softmax	3.47	0.004	(1×)
LSH-1	3.39	0.015	(3.75×)
LSH-4	3.51	0.005	(1.25×)
Linear (ours)	3.40	**17.85**	**(4,462×)**

Unconditional samples

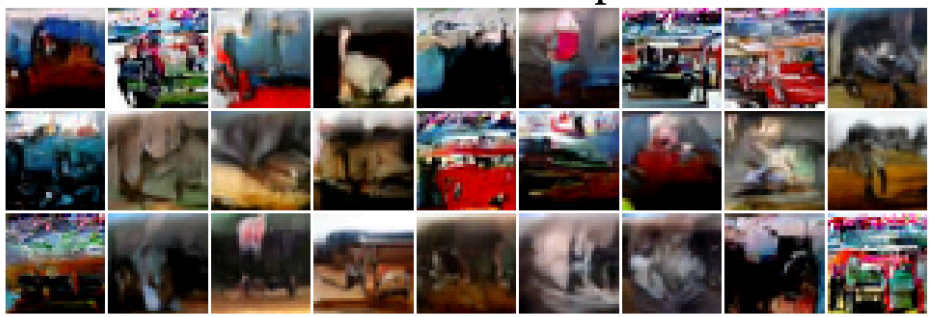
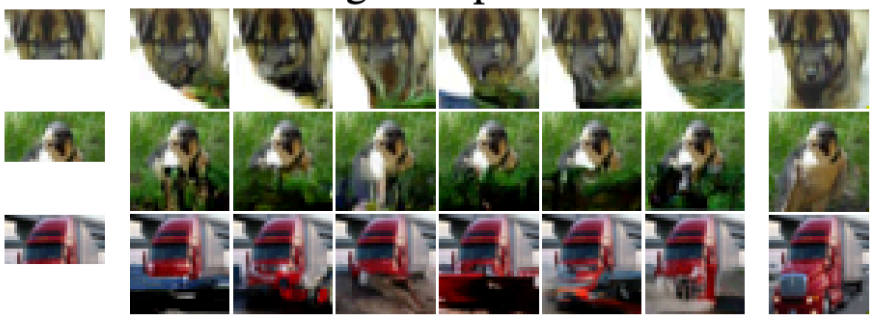


Image completion



(a) (b) (c)

Speech recognition

Method	Validation PER	Time/epoch (s)
Bi-LSTM	10.94	1047
Softmax	5.12	2711
LSH-4	9.33	2250
Linear (ours)	8.08	**824**

PER - Phoneme Error Rate

Итог

Линейный трансформер позволяет существенно снизить затраты на вычисление и памяти, сохраняя приближенную к softmax точность.

Линейный трансформер универсаьнее, точнее и быстрее, чем LSH метод оптимизации трансформеров.

Используя ассоциативность матриц нам удается добится линейной зависимости сложности от длины входа, в том числе с causal masking.

Также если настроить линейный трансформер как RNN, удется ускорить выполнение некоторых задач в тысячи раз.