Linear Transformer

Author	Angelos Katharopoulos
Created time	@2021년 9월 19일 오전 12:35
Date	
Link	
Organization	EPFL
Presented @	ICML 2020
Property	Transformers are RNNs:Fast Autoregressive Transformers with Linear Attention
Published	Aug 2020
Remarks	O(N.D^2)
Tags	Conference Journal
Tags 1	
Tags 1 1	
Type	

Transformers are RNNs:Fast Autoregressive Transformers with Linear Attention

Linear Transformer

code: https://linear-transformers.com/

github: https://github.com/idiap/fast-transformers

They say...

> ... linear transformer is more than 3× faster per epoch compares to the softmax transformer. ...

Let's check out if it's true!

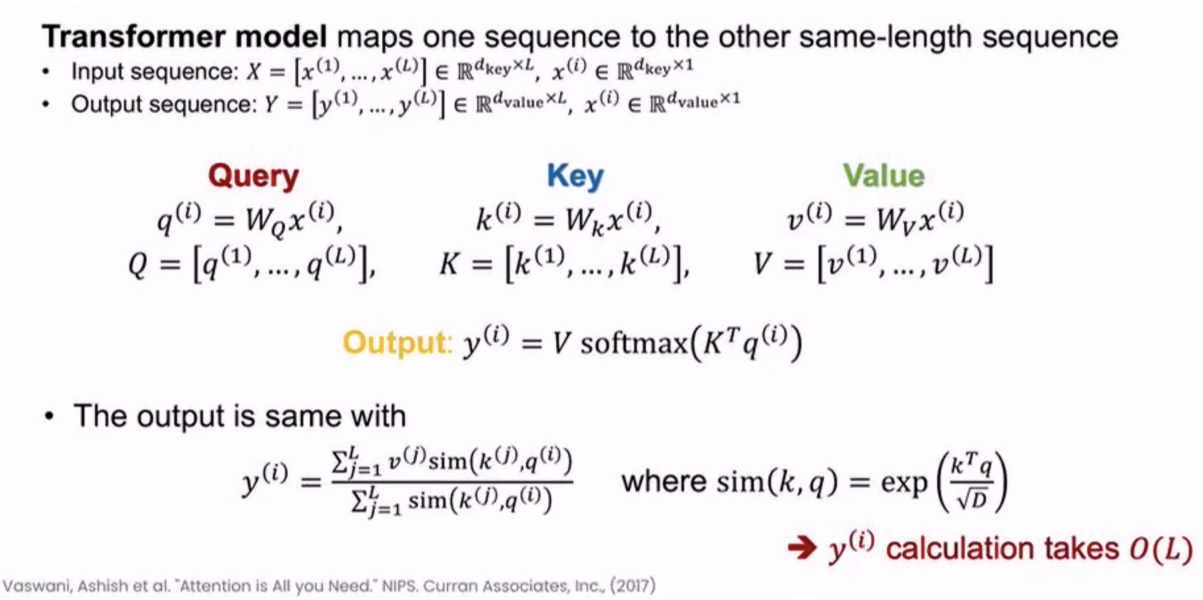
What's related to this 'Linear Transformers'?

- 기존 Transformer의 self attention (multi-head attention)의 quadratic한 time and memory complexity $O(N^2)$ 를 줄이고자 하는 연구들 중, related to this model.
  - 2.1. Efficient Transformers
    - Generating long sequences with sparse transformers.
      - sparse factorizations of the attention matrix
      - $O(N\sqrt{N})$
      - still too big. ❌
    - Reformer: The efficient transformer.
      - using locality-sensitive hashing (LSH) to perform fewer dot products.
      - reduces complexity to $O(N\ logN)$
      - cannot be used for decoding tasks where the keys need to be different from the queries. ❌ 키 쿼리 서로 달라야함.
  - 2.2. Understanding Self-Attention
    - a kernel-based formulation of attention
    - Transformer Dissection : https://arxiv.org/pdf/1908.11775.pdf
    - On the relationship between self-attention and convolutional layers.
      - a multi-head self-attention with sufficient number of heads can express any convolutional layer. head많으면 CNN과 유사.
  - 2.3. Linearized softmax

        - softmax —> bottleneck for classification models with a large categories.
        - Adaptive sampled softmax with kernel based sampling.
          && Sampled softmax with random fourier features.
            - approximated softmax with a linear dot product of feature maps.
    - 'The However parts'
        - Reformer랑 다르게 쿼리와 키에 제약을 가하지 않고 시퀀스 길이에 대해 선형으로 확장함. → they can be used to perform inference in autoregressive tasks.
        - Unlike CNN과의 비교, Autoregressive하게 학습된 self-attention은 RNN처럼 볼 수 있고 이에 대한 유의미한 시간 단축을 보일꺼다.
        - 우리도 linearize the softmax attention in transformers 할꺼다!

(태석님 linear transformers)

Notations

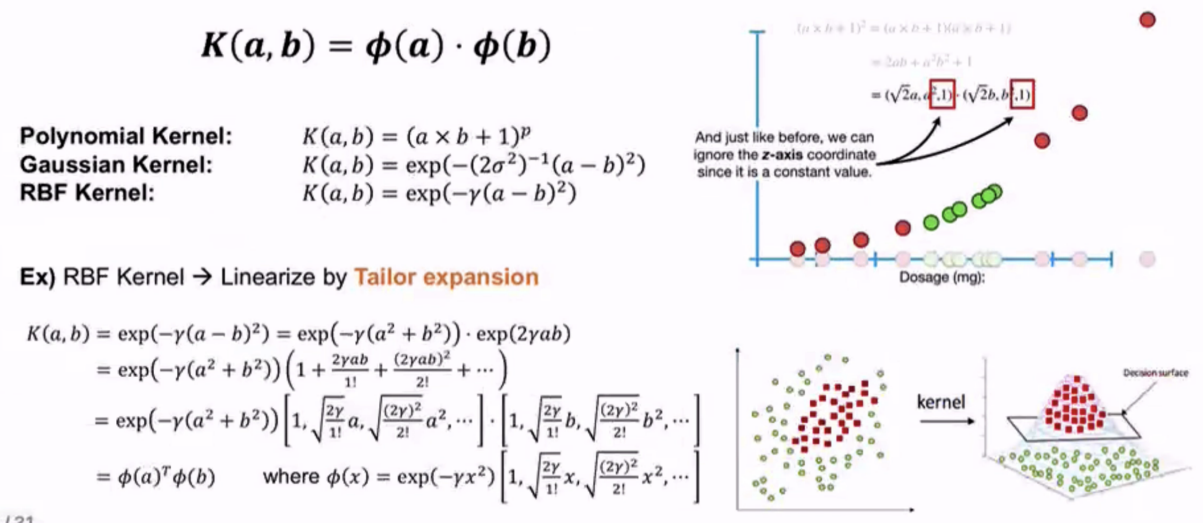


- Input seq와 output seq는 서로 같은 seq len을 가지지만 dim은 다를 수 있음.
- 각각의 열백터 끼리의 연산.
- Computation cost $O(N^2)$이 걸리는 곳은 $K^T q^{(i)}$ 연산의 shape 자체가 N^2.
- softmax를 similarity function으로 표현. scaled dot-product연산에 exp취한 연산이 k와 q의 similarity를 구한 것과 같다고 보는 것. 그래서 $y^{(i)}$를 위와 같이 표현.

- normalization를 위한 합으로 weighted sum을 나누어주는 형식.
- token 하나를 계산 하기 위해서는 L을 모두 돌아야하기 때문에 결국 $O(L)$만큼만 연산하면 된다.

- 이 similiarity로 표현한 식에서 Kernel Trick을 생각하게 됨.

Kernel and Softmax approximation



- ϕ(비선형)$->$ 선형. how to find $\phi()$??? → kernel의 아이디어에서는 비선형 문제를 transform 시키지 않은채 해결하는 방법을 찾는데서 부터 시작된다.
- Generalised inner product X 공간에 있는 임의의 주 점 x, x'를 Z공간으로 Transform 시킨 점을 a와 b라고 할때 주어진 데이터 만으로 이 a와 b의 내적을 구하고 싶은 것. (w/o transform just find!!!)
  - 이 inner product를 $K(a,b)$로 표현하고 Kernel이라고 부르자.
  - $x = (x_1, x_2)$가 주어지고 ϕ라는 transform function이 2차 다항식을 때, transform결과는 $z = (1, x_1, x_2, x_1^2, x_2^2, x_1, x_2)$가 되고 이 경우의 kernel $K(x, x') = z^T z'$는
    $= 1 + x_1x_1' + x_2x_2' + x_1^2x_1'^2 + x_2^2x_2'^2 + +x_1x_1'x_2x_2'$와 같다.
  - 그럼 transform안하고 kernel trick을 사용해서 $K(x, x') = z^T z'$를 구하면 위와 같이 다항식 커널, 가우지안 커널, 테일러 급수를 사용한 RBF커널등을 사용해 구할 수 있다.
    - x라는 데이터를 더 높은 차원으로 맵핑해줘서 classifier학습 시 비선형 데이터를 분류 할 수 있게 해줌.

-
  - 우리가 위에서 본 similiarity function자체를 일종의 커널이라고 생각해보면,
    - $sim(k,q) = exp\left(\frac{k^T q}{\sqrt{D}}\right)$
    - $\phi(a) = exp(-\gamma x^2) = [1, \frac{\sqrt{2\gamma}}{1!}x, \frac{\sqrt{2\gamma}^2}{2!}x^2, ...]$

Linearized dot product:

$$\begin{aligned}\text{sim}(k,q) = \exp\left(\frac{k^T q}{\sqrt{D}}\right) = \exp(x^T y) &= \exp\left(\frac{\|x\|^2}{2}\right)\exp\left(-\frac{\|x-y\|^2}{2}\right)\exp\left(\frac{\|y\|^2}{2}\right)\\ &= \exp\left(\frac{\|x\|^2}{2}\right)K_{Gauss}(x,y)\exp\left(\frac{\|y\|^2}{2}\right)\\ &= \exp\left(\frac{\|x\|^2}{2}\right)\phi(x)^T\phi(y)\exp\left(\frac{\|y\|^2}{2}\right)\\ &= \phi'(k)^T\phi'(q)\end{aligned}$$

- With the linearized kernel ➔

$$y^{(i)} = \frac{\sum_{j=1}^{L} v^{(j)}\text{sim}(k^{(j)},q^{(i)})}{\sum_{j=1}^{L}\text{sim}(k^{(j)},q^{(i)})} = \frac{\sum_{j=1}^{L} v^{(j)}\phi(k^{(j)})^T\phi(q^{(i)})}{\sum_{j=1}^{L}\phi(k^{(j)})^T\phi(q^{(i)})} = \frac{\left(\sum_{j=1}^{L} v^{(j)}\otimes\phi(k^{(j)})\right)\phi(q^{(i)})^T}{\left(\sum_{j=1}^{L}\phi(k^{(j)})\right)\cdot\phi(q^{(i)})}$$

행렬의 곱이 결합법칙(associativity)를 만족한다

$$y^{(i)} = \frac{\left(\sum_{j=1}^{L} v^{(j)}\otimes\phi(k^{(j)})\right)\phi(q^{(i)})^T}{\left(\sum_{j=1}^{L}\phi(k^{(j)})\right)\cdot\phi(q^{(i)})}$$

Encoder:

- Pre-calculation: $\boldsymbol{W} = \sum_{j=1}^{L} v^{(j)}\otimes\phi(k^{(j)})$, $z = \sum_{j=1}^{L}\phi(k^{(j)})$ ➔ $O(L)$
- Output: $y^{(i)} = \frac{\boldsymbol{W}\phi(q^{(i)})}{z\cdot\phi(q^{(i)})}$ ➔ $y^{(i)}$ takes $O(1)$, Y takes $O(L)$

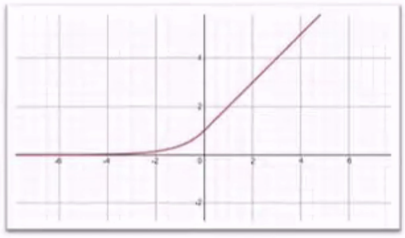
Decoder (Auto-regressive mask):

- Output:
  - $\boldsymbol{W}^{(i)} = \sum_{j=1}^{L} v^{(j)}\otimes\phi(k^{(j)}) = \boldsymbol{W}^{(i-1)} + v^{(i)}\otimes\phi(k^{(i)})$ ➔ $O(1)$
  - $z^{(i)} = \sum_{j=1}^{i}\phi(k^{(j)}) = z^{(i-1)} + \phi(k^{(i)})$ ➔ $O(1)$
  - $y^{(i)} = \frac{\boldsymbol{W}^{(i)}\phi(q^{(i)})}{z^{(i)}\cdot\phi(q^{(i)})}$ ➔ $y^{(i)}$ takes $O(1)$, Y takes $O(L)$

$\phi(x)$ has infinite dimension ☹

Alternatively, the first linear transformer paper use $\phi(x) = \text{elu}(x) + 1$

- Just simply verify their idea
- Make $\text{sim}(q, k) = \phi(q)^T \phi(k)$ positive
  - Guarantee non-negative attention score
  - Prevent unstable and abnormal behaviors



Auto-regressive 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×)**

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×)**

$$\phi(x) = \text{elu}(x) + 1 = \begin{cases} x + 1 & x > 0 \\ e^x & x \leq 0 \end{cases}$$

Then, can we approximate

$\text{sim}(x, y) = \exp(x^T y) \simeq \phi(x)^T \phi(y)$ with the finite-dimension ϕ?

Simple idea: Use Tailor expansion with degree n:

$K_{\text{RBF}}(a, b) \simeq \phi(a)^T \phi(b)$ where $\phi(x) = \exp(-\gamma x^2) \left[1, \sqrt{\frac{2\gamma}{1!}} x, \sqrt{\frac{(2\gamma)^2}{2!}} x^2, \cdots, \sqrt{\frac{(2\gamma)^n}{n!}} x^n\right]$

Problem

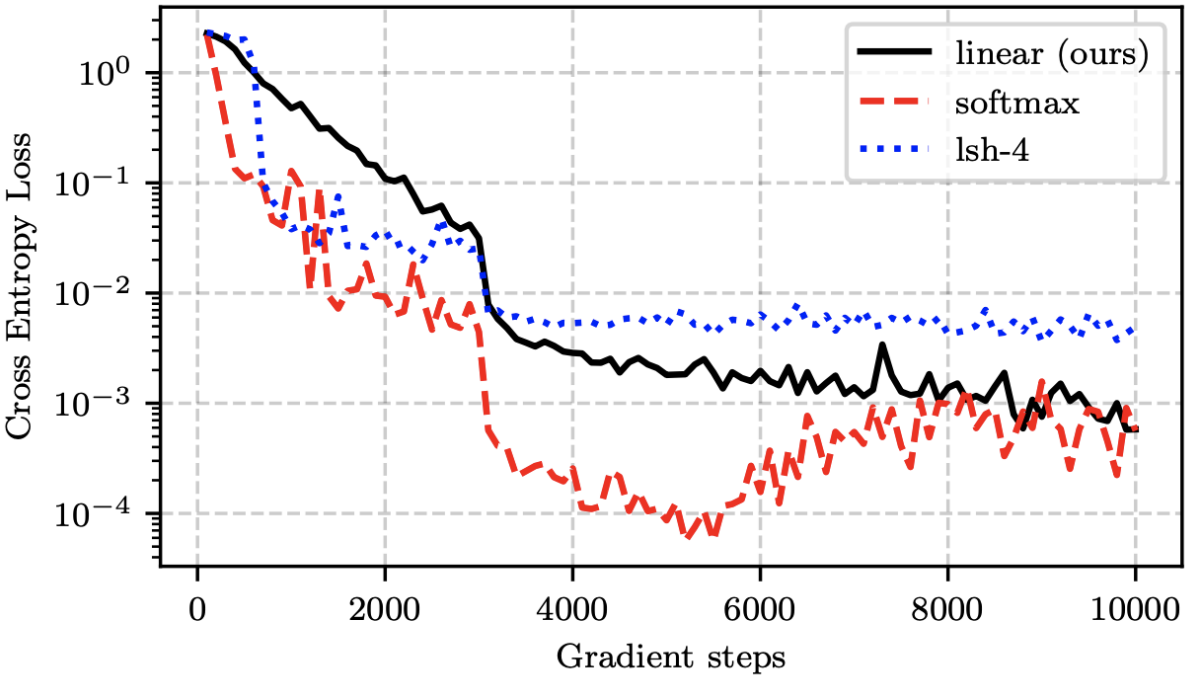
- Inequal contribution on features
- Gradient vanishing on the high degree feature

/ 31

Experiments

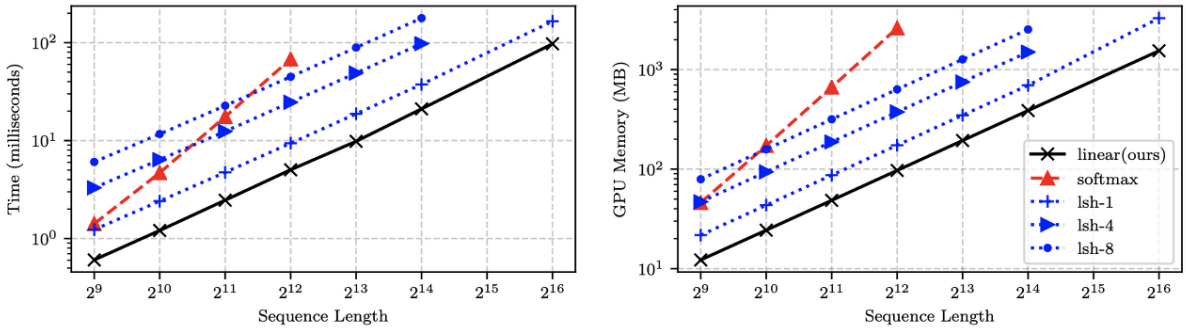
pixel-wise image generation과 automatic speech recognition

baseline은 vanilla transformer와 reformer



reformer에 비해서 좀 더 smooth하게 수렴하고, vanilla transformer와 비슷한 loss까지 수렴

긴 sequence들을 만들어서 training/inference time과 memory를 비교.



vanilla attention은 sequence의 길이에 대해서 quadratic하게 증가하는 데에 비해서 reformer와 linearized attention은 linear하게 증가