Lecture 23: transformers

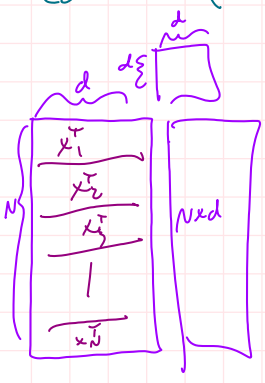
Announcement:

* project info up tonight.

Goal: network which can handle sequence data

3 answers: conv, RNN, attention

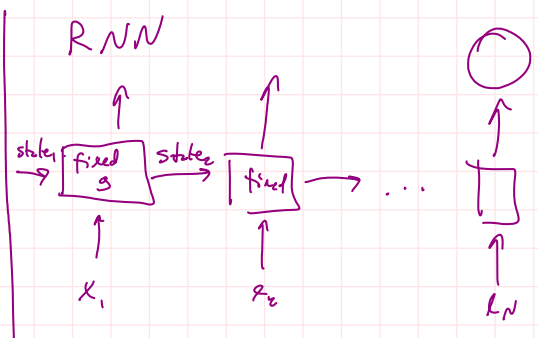
Convolutional (with pre-extracted patches)



Comparison against linear layer with Nd

* "less powerful" than linear
* few params (seem to suffice)
* can learn / gen to other sequence lengths
* computing a function with interacting patches requires multiple layers & overlapping patches (e.g., 14x14 go board, 3x3 conv filters).

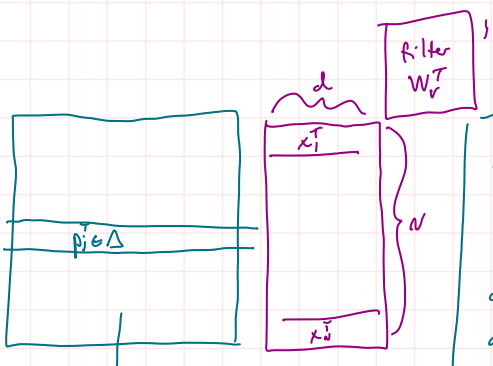
RNN



* if state = ϕ, similar to convolutional
* unlike conv, can model patch interaction, but state limits how far back interactions may be calculated. ("fixed-size memory")
* persevering state also makes training finicky. (e.g., vanishing/exploding gradients.)

Attention (Vaswani et al. '17)

convolution + fix to allow interacting patches



Remark: Suppose $p_j^T X$ $\frac{1}{2}(x^T + x'^T)$. Unclear a priori how to compute arbitrary $g(x, x')$ via $f(\frac{1}{2}(x+x'))$.

this matrix: $\text{softmax}_{\text{(row-wise)}}\left(X \underbrace{W_Q^T W_K}_{\text{often low rank}} X^T\right) \in \mathbb{R}^{N \times N}$ (can be conv?)

$W_K \in \mathbb{R}^{d' \times d}$

Remark (naming): W_{value}, W_{key}, W_{query} names are from NTM (neural turing machine).

Remark (computation). Naive implementation does dN^2 scalar multiplies (applied community fights over this.)

Bells & whistles:

* masking: elementwise product of softmax(—) with a fixed $\{0,1\}^{N \times N}$ ("causal")
* "positional encoding": note that attention is permutation invariant wrt sequence structure.

Example (Vaswani et al.)

$$\begin{bmatrix} x_{i1} \\ \vdots \\ x_{id} \end{bmatrix} \mapsto \begin{bmatrix} x_{i1} + \cos(i/10000^{\cdot}) \\ x_{i2} + \sin(i/10000^{\cdot}) \\ x_{i3} + \cos(2i/10000^{\cdot}) \\ \vdots \end{bmatrix}$$

AlphaFold paper $x_i \mapsto x_i + e_i$ assuming $d \geq N$

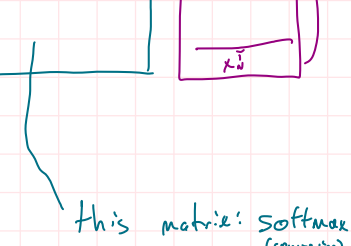
* self-attention vs attention
* "multihead": multiple parallel attention, concatenate outputs (get a higher-order tensor), apply ResNet network.

Theorems

* (Wei-Ma-et al.) Can simulate T-step Turing machine with depth T and $W_K, W_Q \in \mathbb{R}^{\log T \times \log T}$
* (Yao-Peng-Papadimitriou-Narasimhan)

Define $\text{Dyck}_{k,D}$ language: k types of parentheses, D nestings

e.g. $k=3$, $D=4$



(abstracts parse tree)

"Theorem" Can identify & generate $\text{Dyck}_{k,D}$ with depth D and all inner dimensions $\text{poly}(k)$.

Proof idea. Layer i identifies matching delimeters of depth D-i+1

IH: either higher depths are not in this language OR all higher depth delimiters are marked.

Inductive step: Scan right from each delimeter, skipping marked symbols, & identify matching bracket OR determine not in language

Negative result: Dyck_k is not possible

Large language models

Bert { predict occluded words; given two sentences, does one follow the other

GPT-3 = next word prediction

Question: why are these valuable "upstream tasks"