Assignment - 05

K. Surya Prakash
EE18BTECH11026

Q1) t : seq. length
l : no. of layers
@ n : no. of neurons

For training :

Arch.	Time Complexity	Space Complexity
RNN	$O(t \cdot n^2 \cdot l)$	$O(t \cdot l \cdot n)$
Transformer	$O(t^2 \cdot n \cdot l)$	$O(t \cdot l \cdot n)$

For testing.

Arch	Time Complexity	Space Complexity
RNN	$t \cdot n^2 \cdot l$	$l \cdot n$
Transformer	$t^2 \cdot n \cdot l$	$t \cdot n \cdot l$

1b)

If $n < t$

⇒ Complexity for RNN: $O(tn^2 l)$

Complexity for transformer: $O(t^2 n l)$

⇒ In this case:

RNN performs better than a transformer

This is because:

Since $t > n$ ⇒ $O(tn^2 l) < O(t^2 n l)$

⇒ More computation will be taken in self attention layer itself in Transformer.

1c) Yes.

→ self attention layer considers all the input sequence tokens & ~~produces a~~ learns the relation b/w each token.

⇒ Although this might rule-out the sequential modeling. But, the current hidden layer depends on the past hidden layer of every token, which makes it expensive in decoder.

→ Because, decorder uses masked-self attention. Thus computation in self-attention increases as sequence size increases

⇒ Hence, this is a bottleneck for parallelism. But at the same time, this layer helps us do parallel computation in the later layers. since all the attention has been learnt earlier.

(d) No. Feed forward & layer norm. are ~~only~~ computed ~~only~~ based on the o/p of the Self-attention layers (which look through tokens).

⇒ Thus these blocks become independent across time (token), and can be computed in parallel. ⇒ Parallelism

2Q)

a) $x = v_j \Rightarrow \alpha_i = 1$ if $i = j$
$\qquad\qquad\qquad\quad 0$ else

$\Rightarrow \alpha_j \approx 1$; $\cancel{\alpha_y}$

$$\Rightarrow \quad \alpha_j = 1 = \frac{\exp(k_j^T q)}{\sum_{i=1}^{m} \exp(k_i^T q)}$$

$$\Rightarrow \quad \exp(k_j^T q) \simeq \sum_{i=1}^{m} \exp(k_i^T q)$$

ie; $\exp(k_j^T q) >> \exp(\underbrace{k_x^T q}_{x \in \{1, \dots m\} - \{j\}})$

This tells us that:

query vector 'q' is alligned with

$k_j \Rightarrow \quad k_j^T q >> k_x^T q \quad ; \quad \begin{matrix} x \in \{1, \dots m\} \\ x \neq j \end{matrix}$

$\Rightarrow \quad \boxed{q \parallel k_j}$

(b) Given: $\{k_1, \dots k_m\}$ are orthonormal ~~orthogonal~~

Sol.

$\Rightarrow$ $k_i^T k_j = 1$: if $i=j$
$= 0$ otherwise

$\Rightarrow$ Given: $z = \frac{1}{2}(v_a + v_b)$

$$z = \sum_{i=1}^{m} \alpha_i v_i = \frac{1}{2}(v_a + v_b)$$

$\Rightarrow$ $\boxed{\alpha_a = \alpha_b = 1/2}$; $\alpha_i = 0 \ \forall$
$i \in \{1, \dots m\} - \{a, b\}$

This can happen if:

Let $q = \sum_{i=1}^{m} c_i k_i$

$$\Rightarrow \quad k_i^T q = \underbrace{c_i(k_i^T k_i)}_{c_i(1)} + \underbrace{\sum_{\substack{j=1 \\ j \neq i}}^{m} c_j (k_i^T k_j)}_{0}$$

$$\therefore \quad \alpha_a = \frac{\exp(k_a^T q)}{\sum_{i=1}^{m} \exp(k_i^T q)} = \frac{\exp(c_a)}{\sum_{i=1}^{m} \exp(c_i)} = 1/2$$

Similarly $\alpha_b = \dfrac{\exp(c_b)}{\sum \exp(c_i)} = 1/2$

Hence: $\because \alpha_a = \alpha_b = 1/2$

This can happen if

$$\boxed{\begin{array}{c} C_a = C_b \\ (C_a, C_b) \gg C_i \end{array}} \quad \forall\, i \in \{1, \dots m\} - \{a, b\}$$

* A simple case is: if

$$\boxed{q = c(k_a + k_b)} \Rightarrow \begin{array}{l} C_a = C_b = C \\ C_i = 0 \,;\ \forall\, i \notin \{a, b\} \end{array}$$

Q3)

$$\mathcal{L}(q) = \int q(z/x) \log\left(\frac{P(x, z)}{q(z/x)}\right) dz$$

$$= \int q(z/x) \cdot \log\left(\frac{P(x/z) \cdot P(z)}{q(z/x)}\right) dz$$

$$= \int q(z/x) \log(P(x/z))\, dz - \int q(z/x) \log\left(\frac{q(z/x)}{P(z)}\right) dz$$

$$= \underset{z \sim q(z/x)}{E}\left(\log(P(x/z))\right) - KL\left(q(z/x) \,||\, P(z)\right)$$

$\because \; KL(p\|q) = \underset{x\sim p}{E}\left(\log(p/q)\right) = \int p_{(x)} \log\left(p/q_{(x)}\right) dx$

$$\therefore \mathcal{L}(q) = \underbrace{\underset{z\sim q(z/x)}{E}\left(\log\left(P(x/z)\right)\right)}_{\text{Reconstruction}} - \underbrace{KL\left(q(z/x)\,\|\,p(z)\right)}_{\text{Regularisation}}$$

* Reconstruction: (Main loss term)

$\underset{z\sim q(z/x)}{E}\left(\log\left(P(x|z)\right)\right)$ accounts for MLE

⇒ This is analogous with mse loss b/w actual x (img) & reconstructed img ($\tilde{x}$)

* Regularisation:

→ This pushes $q(z/x)$ to be as close as possible to $p(z)$ distribution which is a assumed to be gaussian with var: 1

Q4) $f(p,q) = pq$

Obj := $\min_p \max_q f$

(a) Update steps (given p_t, q_t) ⇒ $f = p_t q_t$

(i) Obj maximises w.r.t q

⇒ Require gradient ascent

⇒ $q_{t+1} = q_t + \frac{\partial f}{\partial q_t} = q_t + p_t$

$\boxed{q_{t+1} = q_t + p_t}$ ⇒ $\boxed{f = p_t q_{t+1}}$

(ii) Obj minimises w.r.t p

⇒ Requires grad. descent

$p_{t+1} = p_t - \frac{\partial f}{\partial p_t} = p_t - q_{t+1}$

$\boxed{p_{t+1} = p_t - q_{t+1}}$

$\boxed{f = p_{t+1} \cdot q_{t+1}}$

Table:

q_0	q_1	q_2	q_3	q_4	q_5	q_6
1	2	1	-1	-2	-1	1
p_0	p_1	p_2	p_3	p_4	p_5	p_6
1	-1	-2	-1	1	2	1

(b) No. In this setting, it's impossible to find a optimal value for f.

Reason:

From the table above, we can find that p, q values oscillate periodically, which do not lead to convergence

(c) Equilibrium point:

$$f_{t+1} = f_t$$

$$p_{t+1} q_{t+1} = p_t q_t$$

$$\Rightarrow (p_t - q_{t+1})(q_{t+1}) = p_t q_t$$

$$\Rightarrow (p_t - q_t - p_t)\overset{(q_t+p_t)}{(\cancel{q_{t+1}})} = p_t q_t$$

$$= -q_t(q_t+p_t) = p_t q_t$$

$$\Rightarrow q_t + p_t = -p_t \text{ (or) } q_t = 0$$

$$\Rightarrow \boxed{q_t = -2p_t}\ ; \boxed{q_t = 0}$$

$\Rightarrow$ Hence if $q_t = 0$ (which does not occur any time in this case)

$$q_t = 0 \Rightarrow p_t = 0 \left(\because p_t = -\frac{q_t}{2}\right) \Rightarrow \begin{array}{l} q_{t+1} = 0+0 = 0 \\ \rightarrow p_{t+1} = 0-0 = 0 \end{array}$$

$\therefore$ Therefore, if one of them reachs '0'

$\Rightarrow$ The fun. reaches an eq. @ 0

$$\boxed{f = 0}$$

x