CSE 891 : Homework 3

Net-id : galibasa

(1) Binary addition

We can use the 3 hidden units to distinguish between the 3 cases:

(i) h_1 to check (input1 + input2 + carry) < 1 or not

(ii) h_2 to check (input1 + input2 + carry) < 2 or not

[if the sum >, 2, we have to carry 1 to the next time step, so this h_2 is crucial for forwarding carry to the next time step]

(iii) h_3 to check (input1 + input2 + ~~input~~ carry) = 3 or not

So,

$$W_{xh} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix}, \quad W_{hh} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \quad b_h = \begin{bmatrix} -0.1 \\ -1.1 \\ -2.1 \end{bmatrix}$$

↳ only considerin h_2 to forward carry (see (ii))

then, we have to ~~make sure~~ check, when $h_1 = 1$ (sum ≥ 1), whether h_2 and h_3 both are same (00 or 11) or not (01 or 10).

(00: h_2 h_3; 11: h_2 h_3) → then, $y = 1$

(01: h_2 h_3; 10: h_2 h_3) → then $y = 0$

So,

$$W_{hy} = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} \qquad b_y = [-0.1]$$

(Ans).

② LSTM Gradient:

②-1 $\overline{h^{(t)}} = \frac{dL}{di^{(t+1)}} \cdot \frac{di^{(t+1)}}{dh_t^{(t)}} + \frac{dL}{df^{(t+1)}} \cdot \frac{df^{(t+1)}}{dh_t^{(t)}} + \frac{dL}{do^{(t+1)}} \cdot \frac{do^{(t+1)}}{dh_t}$

$= i + \frac{dL}{dg^{(t+1)}} \cdot \frac{dg^{(t+1)}}{dh_t^{(t)}}$

$= \overline{i^{(t+1)}} \cdot \underbrace{i^{(t+1)}(1 - i^{(t+1)})}_{\sigma'(x) = \sigma(x)(1-\sigma(x))} \cdot w_{hi} + \overline{f^{(t+1)}} \cdot \underbrace{f^{(t+1)}(1 - f^{(t+1)}) \cdot w_{hf}}_{\sigma'(x) = \dots}$

$+ \overline{o^{(t+1)}} \cdot \underbrace{o^{(t+1)}(1 - o^{(t+1)})}_{\sigma'(x) = \dots} \cdot w_{ho}$

$+ \overline{g^{(t+1)}} \cdot \underbrace{\left(1 - \tanh^2\left(w_{xg} \cdot x^{(t+1)} + w_{hg} h^{(t)}\right)\right)}_{\tanh'(x) = 1 - \tanh^2(x)} w_{hg}$

$\left[+ \frac{dL}{do^{(t)}} \text{ if loss function has } h^{(t)} \text{ term} \right]$

$\overline{c^{(t)}} = \frac{dL}{dh^{(t)}} \cdot \frac{dh^{(t)}}{dc^{(t)}} + \frac{dL}{dc^{(t+1)}} \cdot \frac{dc^{(t+1)}}{dc^{(t)}}$

$= \overline{h^{(t)}} \cdot o^{(t)}(1 - \tanh^2(c^{(t)})) + \overline{c^{(t+1)}} \cdot f^{(t+1)}$

$\overline{g^{(t)}} = \overline{c^{(t)}} \cdot i^{(t)}$

$\overline{o^{(t)}} = \overline{h^{(t)}} \cdot \tanh(c^{(t)})$

$$\overline{f^{(t)}} = \overline{c^{(t)}} \cdot c^{(t-1)}$$

$$\overline{i^{(t)}} = \overline{c^{(t)}} \cdot g^{(t)}$$

(2-2)

$$\overline{w_{xi}} = \sum_{t=1}^{T} \frac{dL}{di^{(t)}} \cdot \frac{di^{(t)}}{dw_{xi}}$$

$$= \sum^{T} \overline{i^{(t)}} \cdot \underbrace{i^{(t)} \cdot (1 - i^{(t)})}_{\sigma'(x) = \sigma(x)\cdot(1-\sigma(x))} \cdot x^{(t)}$$

(2-3) If, $f^{(t)}$ is close to 1 and $i^{(t)}$ & $o^{(t)}$ are close to 0

then

$$\overline{c^{(t)}} = \overline{h^{(t)}}\, o^{(t)} \cdot (1 - \tanh^2(c^{(t)})) + \overline{c^{(t+1)}} \cdot f^{(t+1)}$$

$$\Rightarrow \boxed{\approx \overline{c^{(t+1)}}} \quad [\text{as } o^{(t)} \to 0 \text{ and } f^{(t+1)} \to 1]$$

So, gradient does not change in this case.

Also, $\overline{h^{(t)}} = \overline{i^{(t+1)}} \cdot o + \overline{f^{(t+1)}} \cdot o + \overline{o^{(t+1)}} \cdot o$

$$+ \overline{g^{(t+1)}} \cdot (1 - \tanh^2 \ldots..)$$

$\Rightarrow \boxed{\approx 0}$ [as $i^{(t+1)} \approx 0$, $o^{(t+1)} \approx 0$, $(1 - f^{(t+1)}) \approx 0$ and $\overline{g^{(t+1)}} = \overline{c^{(t+1)}} \cdot i^{(t+1)} = 0$]

[$+ dL/dh^{(t)}$] ← if loss has $h^{(t)}$ term

So, the gradient does not vanish or explode in that case.

③①

Conv3-64 : $64 \times 3 \times 3 \times 3 + 64 = 1728 + 64 = 1792$

Max-pool : 0 $(224 \rightarrow 112)$

Conv3-128 : $128 \times 64 \times 3 \times 3 + 128 = 73856$

Max-pool : 0 $(112 \rightarrow 56)$

Conv3-256 : $256 \times 128 \times 3 \times 3 + 256 = 295168$

Conv3-256 : $256 \times 256 \times 3 \times 3 + 256 = 590080$

Max-pool : 0 $(56 \rightarrow 28)$

FC-1024 : ~~$1024 \times 28 \times 28 \times 8 + 1024 =$~~

$1024 \times 28 \times 28 \times 256 + 1024 = 205520896 + 1024$

$= 205521920$

FC-1000 : $1000 \times 1024 + 1000 = 1025000$

Total : 207507816

③ ②

The left one is easier to learn in terms of exploding / vanishing gradient.

Because, the left one employs an identity mapping ($y = f(x) + x$) which is very easy to learn and can make the model robust from vanishing / exploding gradient problem.

Also, post-activation in the right can not gurantee that the output will not cause vanishing / exploding gradient.

(4) Auto-Regressive Model

(4-1) PixelCNN

(1-a) The $O(\cdot)$ of PixelCNN would be same as ~~convoluti~~ typical CNN. Because masking in PixelCNN can only affect complexity in constant factors.

So, the number of connections

$\Rightarrow O(dHWK^2)$

(1-b) As we can evaluate each pixel parallelly (in training) the number of sequential operation would be

$\rightarrow O(d)$.

But in case of testing, we can not get/evaluate ~~all~~ any pixels before getting all the prior pixels.

So, in that case parallel computation would not help. and the # of sequential operation would be :

$\rightarrow O(HWd)$

(4-2) MDRNN

(4-2-a)

The total number of connections:

$O(d + HWk^2)$ [as # of input channels = k
# of recurrent neurons = k
# of layer = d]

(4-2-b)

The number of sequential operations:

$O(\sqrt{HWd})$ [as here we can do it partially sequential order, like if the hidden nodes from north and west are evaluated already, we can evaluate the current node)

(4)
(3)

PixelCNN can be parrallelized while training. And it captures spatial representation more strongly. Also it ~~uses less~~ focuses less on the sequential pattern which may fail to capture long-term dependencies. (It has $O(d)$ seq. operations)

MDRNN can not be parrallelize like PixelCNN. It has $O(\sqrt{h+w}d)$ sequential operations.

But, MDRNN can capture sequential representation more strongly than PixelCNN. Yet, it may lacu in learning spatial representation with respect to PixelCNN.