CSC413: Homework 4

Tianyu Du (1003801647)

2020/03/20 at 20:53:38

1 Architectural Choice v.s. Vanishing / Exploding Gradients

1.1 Warmup: A Single Neuron RNN

1.1.1 Effect of Activation - Sigmoid [1pt]

Solution. Obviously, $\left|\frac{\partial f(x)}{\partial x}\right| \geq 0$. Note the function can be written as

$$f(x) = \cdots \sigma(W_2 \sigma(W_1 x + b_1) + b_2) \cdots \tag{1.1}$$

Define the pre-activations to be

$$\begin{align}
a_1 &= W_1 x + b_1 \tag{1.2}\\
a_2 &= W_2 \sigma(a_1) + b_2 \tag{1.3}\\
&\vdots \tag{1.4}\\
a_n &= W_n \sigma(a_{n-1}) + b_n \tag{1.5}\\
f(x) &= W_{n+1} \sigma(a_n) + b_{n+1} \tag{1.6}
\end{align}$$

Because $\sigma'(x) = \sigma(x)(1 - \sigma(x))$,

$$\begin{align}
\frac{\partial \sigma(a_1(x))}{\partial x} &= \frac{\partial}{\partial x} \sigma(W_1 x + b) = W_1 \sigma(a_1)(1 - \sigma(a_1)) \tag{1.7}\\
\frac{\partial \sigma(a_n)}{\partial a_{n-1}} &= \frac{\partial}{\partial a_{n-1}} W_n \sigma(a_{n-1}) + b_n = W_n \sigma(a_{n-1})(1 - \sigma(a_{n-1})) \tag{1.8}
\end{align}$$

Therefore,

$$\begin{align}
\frac{\partial f(x)}{\partial x} &= \left(\prod_{i=1}^n W_i\right)\left(\prod_{i=1}^n \sigma'(a_i)\right) \tag{1.9}\\
&= \prod_{i=1}^n \sigma'(a_i) \text{ because } W_i = 1 \text{ for every } i \tag{1.10}\\
&= \prod_{i=1}^n \sigma(a_i)(1 - \sigma(a_i)) \tag{1.11}
\end{align}$$

Since $\sigma(x) \in (0,1)$, hence $0 \le \sigma(x)(1-\sigma(x)) \le \frac{1}{4}$, therefore,

$$0 \le \frac{\partial f(x)}{\partial x} = \prod_{i=1}^{n} \sigma(a_i)(1-\sigma(a_i)) \le \left(\frac{1}{4}\right)^n \tag{1.12}$$

Therefore,

$$0 \le \left|\frac{\partial f(x)}{\partial x}\right| \le \left(\frac{1}{4}\right)^n \tag{1.13}$$

The sigmoid activation solves the gradient exploding problem, but it is still for the model to suffer from gradient vanishing problem. ■

1.1.2 Effect of Activation - Tanh [1pt]

Solution. Note that $\tanh(x) \in [0,1]$ and $\tanh'(x) = 1 - \tanh^2(x)$. Therefore, $\tanh'(x) \in [0,1]$. Using the result from equation (1.10),

$$\frac{\partial f(x)}{\partial x} = \prod_{i=1}^{n} \tanh'(a_i) = \prod_{i=1}^{n} (1 - \tanh^2(a_i)) \tag{1.14}$$

Hence,

$$0 \le \frac{\partial f(x)}{\partial x} \le 1 \tag{1.15}$$

So that

$$0 \le \left|\frac{\partial f(x)}{\partial x}\right| \le 1 \tag{1.16}$$

The tanh activation solves the gradient exploding problem, but it is still for the model to suffer from gradient vanishing problem. ■

1.2 Matrices and RNN

1.2.1 Gradient through RNN [1pt]

Solution. Note that for any $x \in \mathbb{R}$, $\tanh'(x) = 1 - \tanh^2(x)$.

$$\begin{aligned} \frac{\partial x_{t+1}}{\partial x_t} &= \frac{\partial}{\partial x_t} \tanh(Wx_t) \qquad (1.17) \\ &= (1 - \tanh^2(Wx_t))W \qquad (1.18) \end{aligned}$$

$(1 - \tanh^2(Wx_t))$ is a diagonal matrix with diagonal entries in $[0,1]$. And all singular values of $(1 - \tanh^2(Wx_t))$ is therefore in $[0,1]$. Let $A = (1 - \tanh^2(Wx_t))$ and $B = W$, by the hint,

$$\begin{aligned} \sigma_{max}\left(\frac{\partial x_{t+1}}{\partial x_t}\right) &\le \sigma_{max}(1 - \tanh^2(Wx_t))\sigma_{max}(W) \qquad (1.19) \\ &= \sigma_{max}(1 - \tanh^2(Wx_t))\frac{1}{2} \qquad (1.20) \\ &\le \frac{1}{2} \qquad (1.21) \end{aligned}$$

By chain rule, the input-output Jacobian is

$$\frac{\partial x_n}{\partial x_1} = \prod_{t=1}^{n-1} \frac{\partial x_{t+1}}{\partial x_t} \tag{1.22}$$

Applying the hint inductively,

$$\sigma_{max}\left(\prod_{t=1}^{n-1} \frac{\partial x_{t+1}}{\partial x_t}\right) \leq \prod_{t=1}^{n-1} \sigma_{max}\left(\frac{\partial x_{t+1}}{\partial x_t}\right) \tag{1.23}$$

$$= \left(\frac{1}{2}\right)^{n-1} \tag{1.24}$$

Still have to show $0 \leq \sigma_{\max}\left(\frac{\partial x_n}{\partial x_1}\right)$

■

1.2.2 Gradient through Residual / GRU / LSTM Layers [0pt]

Solution. Note that

$$\frac{\partial z_{t+1}}{\partial z_t} = \frac{\partial z_t + f_t(z_t)}{\partial z_t} \tag{1.25}$$

$$= \mathbf{1}_n + \frac{\partial f_t(z_t)}{\partial z_t} \tag{1.26}$$

Note that all singular values of identity matrix are one. Let $A = \frac{\partial f_t(z_t)}{\partial z_t}$ and $B = \mathbf{1}_n$, then apply the lemma in hint, $\forall i \in \{1, \cdots, \frac{n}{2}\}$:

$$\sigma_i\left(\frac{\partial z_{t+1}}{\partial z_t}\right) \geq |\sigma_i(A) - \sigma_{max}(B)| \tag{1.27}$$

$$= |\sigma_i(A) - 1| \tag{1.28}$$

$$= 1 - \sigma_i(A) \text{ because } \sigma_i \ll 1 \tag{1.29}$$

For $i = 1$,

$$\sigma_1\left(\frac{\partial z_{t+1}}{\partial z_t}\right) = \sigma_{min}\left(\frac{\partial z_{t+1}}{\partial z_t}\right) \tag{1.30}$$

$$\geq 1 - \sigma_{min}\left(\frac{\partial f_t(z_t)}{\partial z_t}\right) \tag{1.31}$$

$$\geq 1 - \sigma_{small} \tag{1.32}$$

Similarly, $\forall i \in \{\frac{n}{2}, \cdots, 1\}$,

$$\sigma_i\left(\frac{\partial z_{t+1}}{\partial z_t}\right) \geq |\sigma_i(A) - \sigma_{max}(B)| \tag{1.33}$$

$$= |\sigma_i(A) - 1| \tag{1.34}$$

$$= \sigma_i(A) - 1 \text{ because } \sigma_i \gg 2 \tag{1.35}$$

In particular, take $i = n$,

$$\sigma_n\left(\frac{\partial z_{t+1}}{\partial z_t}\right) = \sigma_{max}\left(\frac{\partial z_{t+1}}{\partial z_t}\right) \tag{1.36}$$

$$\geq \sigma_{max}\left(\frac{\partial f_t(z_t)}{\partial z_t}\right) - 1 \tag{1.37}$$

$$\geq \sigma_{big} - 1 \tag{1.38}$$

Therefore,

$$\sigma_{min}\left(\frac{\partial z_{t+1}}{\partial z_t}\right) \gg 0 \tag{1.39}$$

$$\sigma_{max}\left(\frac{\partial z_{t+1}}{\partial z_t}\right) \gg 1 \tag{1.40}$$

■

1.2.3 Benefits of Residual Connections [1pt]

Solution. In the previous part, we've shown that the singular values of Jacobian matrix in each layer is bounded below and bounded away from zero. Hence, as the number of layers, n, grows the gradient does not vanish as fast as before, the gradient vanishing problem is solved. However, the residual connection does not help bound singular values from above, therefore, it is still possible for the gradient to explode as n grows large. The gradient exploding problem is $\underline{\text{not}}$ solved. ■

1.3 Batch Normalization

1.3.1 Gradients of Batch Norm [0pt]

Solution.

$$\frac{\partial y_i}{\partial x_i} = \gamma \frac{\partial \hat{x}_i}{\partial x_i} \tag{1.41}$$

$$= \gamma \frac{\partial}{\partial x_i} \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \tag{1.42}$$

$$= \frac{\gamma}{\sigma_{\mathcal{B}}^2 + \epsilon} \left[\frac{\partial x_i - \mu_{\mathcal{B}}}{\partial x_i} \sqrt{\sigma_{\mathcal{B}}^2 + \epsilon} - \frac{\partial \sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}{\partial x_i} (x_i - \mu_{\mathcal{B}}) \right] \tag{1.43}$$

$$= \frac{\gamma}{\sigma_{\mathcal{B}}^2 + \epsilon} \left[\left(1 - \frac{1}{m}\right) \sqrt{\sigma_{\mathcal{B}}^2 + \epsilon} - \frac{1}{2} \frac{(x_i - \mu_{\mathcal{B}})}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \frac{\partial \sigma_{\mathcal{B}}^2}{\partial x_i} \right] \tag{1.44}$$

$$= \frac{\gamma}{\sigma_{\mathcal{B}}^2 + \epsilon} \left[\frac{m-1}{m} \sqrt{\sigma_{\mathcal{B}}^2 + \epsilon} - \frac{1}{2} \frac{(x_i - \mu_{\mathcal{B}})}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \frac{1}{m} \sum_{j=1}^{m} 2(x_j - \mu_{\mathcal{B}}) \frac{\partial x_j - \mu_{\mathcal{B}}}{\partial x_i} \right] \tag{1.45}$$

$$= \frac{\gamma}{\sigma_{\mathcal{B}}^2 + \epsilon} \left[\frac{m-1}{m} \sqrt{\sigma_{\mathcal{B}}^2 + \epsilon} - \frac{1}{2} \frac{(x_i - \mu_{\mathcal{B}})}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \frac{1}{m} \left(\sum_{j=1}^{m} 2(x_j - \mu_{\mathcal{B}}) \left(-\frac{1}{m}\right) + 2(x_i - \mu_{\mathcal{B}}) \right) \right] \tag{1.46}$$

$$= \frac{\gamma}{\sigma_{\mathcal{B}}^2 + \epsilon} \left[\frac{m-1}{m} \sqrt{\sigma_{\mathcal{B}}^2 + \epsilon} - \frac{1}{2} \frac{(x_i - \mu_{\mathcal{B}})}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \left(2 \frac{(x_i - \mu_{\mathcal{B}})}{m} \right) \right] \tag{1.47}$$

$$= \frac{\gamma}{\sigma_{\mathcal{B}}^2 + \epsilon} \left[\frac{m-1}{m} \sqrt{\sigma_{\mathcal{B}}^2 + \epsilon} - \frac{1}{m} \frac{(x_i - \mu_{\mathcal{B}})^2}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \right] \tag{1.48}$$

$$= \frac{\gamma}{\sigma_{\mathcal{B}}^2 + \epsilon} \left[\frac{m-1}{m} \frac{\sigma_{\mathcal{B}}^2 + \epsilon}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} - \frac{1}{m} \frac{(x_i - \mu_{\mathcal{B}})^2}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \right] \tag{1.49}$$

■

Not Finished.

1.3.2 Batch Normalization and ResNet [1pt]

Unsure

Solution. The batch normalization layer aims to bound the magnitude of gradient flow $\frac{\partial y_i}{\partial x_i}$ to prevent exploding / vanishing gradient. In the first setup, if we add the residual connection after the batch normalization, the gradient is still possible to explode. Therefore, we should use the second setting. ■

2 Auto-regressive Models

2.1 WaveNet

2.1.1 Connections [0pt]

2.1.2 Parallelism [0pt]

2.1.3 Discussion [0pt]

2.2 PixelCNN

2.2.1 Connections [1pt]

Solution. Assuming zero padding, the output and input sizes of each layer in PixelCNN are the same. Between any two layers, there are 4 connections in a typical 3×3 convolution filter, the same filter is applied $H \times W$ times for each input and output channel. Therefore, the number of connections in a typical layer is $\mathcal{O}(4k^2 H \times W) = \mathcal{O}(k^2 H \times W)$. Given that there are d layers, the total number of connections is

$$\mathcal{O}(dk^2 H \times W) \tag{2.1}$$

■

2.2.2 Parallelism [1pt]

Solution. Operations of all k^2 in-out channel pairs can be run in parallel. Further, for each of these in-out channel pairs, the operations of computing convolution by applying filters on the $H \times W$ pixels can be executed in parallel. However, computing output of each layer requires the output from its previous layer, this cannot be run in parallel. Therefore, the minimum number of the sequential is

$$\mathcal{O}(d) \tag{2.2}$$

■

2.3 Multidimensional RNN

2.3.1 Connections [1pt]

Solution. Since $\mathbf{x}^{(i,j)}, \mathbf{h}^{(i,j)} \in \mathbb{R}^k$, therefore, for each single hidden neurone,

$$\mathbf{h}^{(i,j)} = \phi\left(\mathbf{W}_{\text{in}}^\top \mathbf{x}^{(i,j)} + \mathbf{W}_{\text{W}}^\top \mathbf{h}^{(i-1,j)} + \mathbf{W}_{\text{N}}^\top \mathbf{h}^{(i,j-1)}\right) \tag{2.3}$$

Each of these three matrix vector products involves k^2 connections. The total number of connections involved in the computation of $\mathbf{h}^{(i,j)}$ is $3k^2$. Between two layers, there are $H \times W$ such hidden neurones to be computed, therefore, the number of connections is $\mathcal{O}(3k^2 H \times W)$. Given there are d layers, the total number of connections is

$$\mathcal{O}(dk^2 H \times W) \tag{2.4}$$

■

2.3.2 Parallelism [0pt]

Solution. Within each layer, the computation of $\mathbf{h}^{(i,j)}$ are interdependent, and cannot be run in parallel. However, the matrix vector products can be executed in parallel. Therefore, for the entire model with d layers, the number of sequential operations requires is

$$\mathcal{O}(dH \times W) \tag{2.5}$$

■

2.3.3 Discussion [1pt]

Solution.
PixelCNN:

(i) Pro (1): Better parallelization potentials since all operations within one layer can be run in parallel.

(ii) Con (1): Pixel CNN has small context window, for $k = 3$, the model is only looking at 4 other pixels.

MDRNN:

(i) Pro (1): MDRNN has a larger context window, each of $\mathbf{h}^{(i,j)}$ takes information from both the current pixel and previous hidden neurones, $\mathbf{h}^{(i-1,j)}$ and $\mathbf{h}^{(i,j-1)}$. These two previous hidden neurones contain information from previous pixels as well. By induction, $\mathbf{h}^{(i,j)}$ is looking at information from all previous pixels. Effectively, the context window is infinitely large.

(ii) Pro (2): MDRNN has lower memory cost, after computing $\mathbf{h}^{(i,j)}$, the activation of previous hidden units, $\mathbf{h}^{(i-1,j)}$ and $\mathbf{h}^{(i,j-1)}$ can be discarded from memory, this leads to better memory efficiency.

(iii) Con (1): MDRNN has worse parallelization potential since operations within each layer cannot be fully run in parallel.

■