1 Architectural Choice v.s. Vanishing / Exploding Gradients

1.1 Warmup: A Single Neuron RNN

1.1.1 Effect of Activation - Sigmoid

For each layer $h^{(i)} = \sigma(w^{(i)}h^{(i-1)} + b^{(i)})$, we have

$$\frac{\partial h^{(i)}}{\partial h^{(i-1)}} = w^{(i)}\sigma'(w^{(i)}h^{(i-1)} + b^{(i)}) = \sigma'(w^{(i)}h^{(i-1)} + b^{(i)})$$

As $\sigma'(x) = \sigma(x)(1 - \sigma(x))$, when $\sigma(x) \leq 1$, $\sigma'(x) \leq \frac{1}{4}$.

Thus,

$$\frac{\partial f(x)}{\partial x} = \frac{\partial h^{(1)}}{\partial x}\frac{\partial h^{(2)}}{\partial h^{(1)}}\cdots\frac{\partial f(x)}{\partial h^{(n-1)}} = (\sigma')^n \leq (\frac{1}{4})^n$$

The gradients necessarily have to vanish as they are backpropagated.

1.1.2 Effect of Activation - Tanh

For each layer $h^{(i)} = tanh(w^{(i)}h^{(i-1)} + b^{(i)})$, we have

$$\frac{\partial h^{(i)}}{\partial h^{(i-1)}} = w^{(i)}tanh'(w^{(i)}h^{(i-1)} + b^{(i)}) = tanh'(w^{(i)}h^{(i-1)} + b^{(i)})$$

As $tanh'(x) = 1 - tanh^2(x)$, when $0 \leq tanh(x) \leq 1$, $0 \leq tanh'(x) \leq 1$.

$$\frac{\partial f(x)}{\partial x} = \frac{\partial h^{(1)}}{\partial x}\frac{\partial h^{(2)}}{\partial h^{(1)}}\cdots\frac{\partial f(x)}{\partial h^{(n-1)}} = (1 - tanh')^n$$

$$0 \leq (1 - tanh')^n \leq 1$$

Thus, the gradients do not necessarily have to vanish or explode this time.

1.2 Matrices and RNN

1.2.1 Gradient through RNN

For the input-output Jacobian, we have $\frac{\partial x_n}{\partial x_1} = \frac{\partial x_2}{\partial x_1}\frac{\partial x_3}{\partial x_2}\cdots\frac{\partial x_n}{\partial x_{n-1}}$

With the help of the given Hint,

$$\sigma_{max}(\frac{\partial x_n}{\partial x_1}) = \sigma_{max}(\frac{\partial x_2}{\partial x_1})\sigma_{max}(\frac{\partial x_3}{\partial x_2})...\sigma_{max}(\frac{\partial x_n}{\partial x_{n-1}})$$

As $x_{t+1} = tanh(Wx_t)$, $\frac{\partial x_{t+1}}{\partial x_t} = W tanh'(Wx_t)$.

$$\sigma_{max}(\frac{\partial x_{t+1}}{\partial x_t}) = \sigma_{max}(W)\sigma_{max}(tanh'(Wx_t)) = \frac{1}{2}\sigma_{max}(tanh'(Wx_t))$$

As $0 \leq tanh' \leq 1$, $0 \leq \sigma_{max}(\frac{\partial x_{t+1}}{\partial x_t}) \leq \frac{1}{2}$. Thus,

$$0 \leq \sigma_{max}(\frac{\partial x_n}{\partial x_1}) \leq (\frac{1}{2})^n$$

1.2.3 Benefits of Residual Connections

For the input-output Jacobian, we have $\frac{\partial z_n}{\partial z_1} = \frac{\partial z_2}{\partial z_1}\frac{\partial z_3}{\partial z_2}...\frac{\partial z_n}{\partial z_{n-1}}$

$z_{t+1} = z_t + f_t(z_t)$, $\frac{\partial z_{t+1}}{\partial z_t} = 1 + f_t'(z_t)$

As we showed in the last question,

$$\sigma_{max}(\frac{\partial z_{t+1}}{\partial z_t}) \geq \sigma_{big} - 1 \gg 2 - 1 = 1$$

Thus,

$$\sigma_{max}(\frac{\partial z_n}{\partial z_1}) = \sigma_{max}(\frac{\partial z_2}{\partial z_1})\sigma_{max}(\frac{\partial z_3}{\partial z_2})...\sigma_{max}(\frac{\partial z_n}{\partial z_{n-1}}) \geq (\sigma_{big} - 1)^n$$

Residual connections/GRU/LSTM layers can solve the vanishing gradients problem. However, they cannot also solve the exploding gradient problem.

1.3 Batch Normalization

1.3.2 Batch Normalization and ResNet

We denote the input as x, the function of the hidden layers i as $h^{(i)}$, $f(x) = h^{(n)}...h^{(1)}(x)$.

For the left form, the final result is $h^{(n)}...h^{(1)}(x) + x = f(x) + x$,

$$\frac{\partial(f(x)+x)}{\partial x} = 1 + \frac{\partial f(x)}{\partial x}$$

For the right form, final result is $h^{(6)}h^{(5)}(h^{(4)}...h^{(1)}(x) + x) = f(x) + h^{(6)}h^{(5)}(x)$,

$$\frac{\partial(f(x)+h^{(6)}h^{(5)}(x))}{\partial x} = \frac{\partial h^{(6)}h^{(5)}(x)}{\partial x} + \frac{\partial f(x)}{\partial x}$$

As these two architectures are two different forms of ResNet Blocks, we can consider that the ResNet is composed of a sequence of these blocks.

For the left form, if this block iterates n times, $\frac{\partial x_n}{\partial x_1} = (1 + \frac{\partial f(x)}{\partial x})^n$.

For the right form, if this block iterates n times, $\frac{\partial x_n}{\partial x_1} = (\frac{\partial h^{(6)}h^{(5)}(x)}{\partial x} + \frac{\partial f(x)}{\partial x})^n$.

The model on the left solves the vanishing gradient problem, but it can still get exploding gradients. The model on the right will automatically adjust weights to avoid vanishing and exploding through BatchNorm, but it does not necessarily avoid vanishing and exploding gradients.

2 Auto-regressive Models

2.2 PixelCNN

2.2.1 Connections

Consider a d layer PixelCNN model, the total number of connections is $O(dWHk^2)$.

2.2.2 Parallelism

The minimum number of sequential operations to compute the output is $O(d)$.

2.3 Multidimensional RNN

2.3.1 Connections

Consider a d layer MDRNN model, the total number of connections is $O(dWHk^2)$.

2.3.3 Discussion

Their computational complexities are the same, i.e., $O(dWHk^2)$.

Each layer of PixelCNN has k^2 different convolutional kernels, and each layer of MDRNN holds 3 different matrices, with dimension $k \times k$. Their memory complexities are also the same, i.e., $O(dk^2)$.

For PixelCNN, each unit is calculated independently at each layer; however, for MDRNN, the calculation of each x depends on the left and top values. Thus, PixelCNN has more parallelization potential than MDRNN.

PixelCNN only has the values that are in the kernel and not masked as the context window, whereas MDRNN has all the values before it as its context window. As a result, MDRNN has larger size of context windows.