HW2 - Theory

hhp9256 - Hoang Pham

1 Theory

1.1 Convolutional Neural Networks

(a) The output size is 9×13.

(b) F filters will yield F channels in the output. Therefore, the output tensor size is

$$F \times \left[\frac{(H+2P)-(H_K+D(H_K-1)}{S}+1\right] \times \left[\frac{(W+2P)-(W_K+D(W_K-1)}{S}+1\right]$$

(c) (i) Note that the convention used in PyTorch for ConvNet weight shape is always (out_channels, in_channels, kernel size), so $f_{\mathbf{W}}(\mathbf{x}) \in \mathbb{R}^5$, where $5 = \frac{11-3}{2}+1$, and $\mathbf{W} \in \mathbb{R}^{1\times 7\times 3}$ with

$$f_{\mathbf{W}}(\mathbf{x})[n] = \sum_{k=1}^{7}\sum_{i=1}^{3} \mathbf{x}[2(n-1)+i][k]\mathbf{W}[1,k,i]$$

(ii) Since $f_{\mathbf{W}}(\mathbf{x}) \in \mathbb{R}^5$, and $\mathbf{W} \in \mathbb{R}^{1\times 7\times 3}$, we have $\frac{\partial f_{\mathbf{W}}(\mathbf{x})}{\partial \mathbf{W}} \in \mathbb{R}^{5\times(1\times 7\times 3)}$, and

$$\frac{\partial f_{\mathbf{W}}(\mathbf{x})}{\partial \mathbf{W}}[n,1,k,i] = \mathbf{x}[2(n-1)+i][k].$$

(iii) Since $\mathbf{x} \in \mathbb{R}^{7\times 11}$, we have $\frac{\partial f_{\mathbf{W}}(\mathbf{x})}{\partial \mathbf{x}} \in \mathbb{R}^{5\times 7\times 11}$, and, using 1-based indexing:

$$\frac{\partial f_{\mathbf{W}(\mathbf{x})}}{\partial \mathbf{x}}[n,k,i] = \begin{cases} \mathbf{W}[1,k,i-2(n-1)] & \text{when } 2(n-1) < i \leq 3 \\ 0 & \text{otherwise} \end{cases}$$

(iv) Since $\mathbf{W} \in \mathbb{R}^{1\times 7\times 3}$, we have $\frac{\partial \ell}{\partial \mathbf{W}} \in \mathbb{R}^{1\times 7\times 3}$, since ℓ is a scalar. Additionally, by chain rule $\frac{\partial \ell}{\partial \mathbf{W}} = \frac{\partial \ell}{\partial f_{\mathbf{W}}(x)}\frac{\partial f_{\mathbf{W}}(x)}{\partial \mathbf{W}}$, we have:

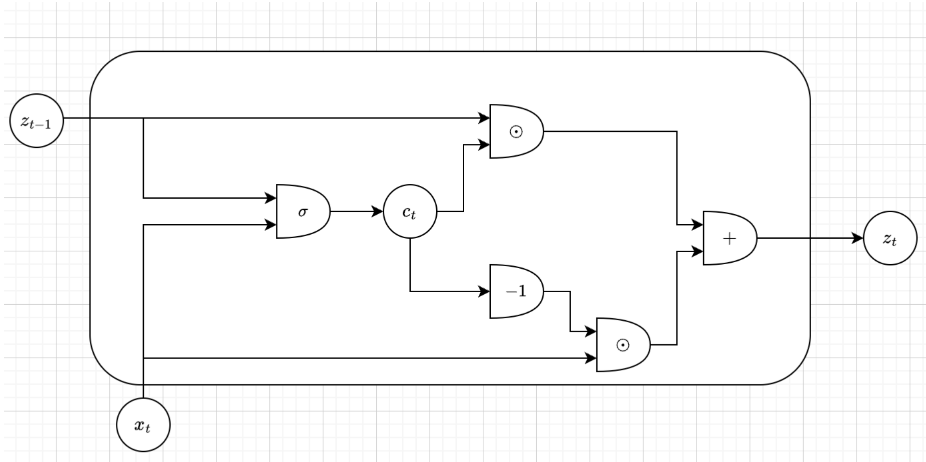
$$\frac{\partial \ell}{\partial \mathbf{W}}[1,k,i] = \sum_{n=1}^{2}\left(\frac{\partial \ell}{\partial f_{\mathbf{W}}(\mathbf{x})}\right)[n].\mathbf{x}[2(n-1)+i][k]$$

This formula is calculating gradient for a backward pass, while the first part (i) is a forward pass. The initial forward pass performs a stride (size 2), which makes the backward pass perform a dilation operation.

1.2 Recurrent Neural Networks

1.2.1 Part 1

(a) Attached is the graph of the RNN.



(b) The dimension of c_t is $\mathbb{R}^m$.

(c) For two vectors x, y, $x \odot y = D(y)x = D(x)y$, where $D(c)$ denotes diagonal matrix created from c. Therefore

$$h_t = D(c_t)h_{t-1} + D(\mathbf{W_x x}_t)(1 - c_t)$$

and

$$\begin{aligned}\frac{\partial h_t}{\partial h_{t-1}} &= D(c_t) + D(h_{t-1})\frac{\partial c_t}{\partial h_{t-1}} + D(\mathbf{W_x x}_t)\frac{\partial(1-c_t)}{\partial h_{t-1}} \\ &= D(c_t) + D(h_{t-1})\frac{\partial c_t}{\partial h_{t-1}} - D(\mathbf{W_x x}_t)\frac{\partial c_t}{\partial h_{t-1}}\end{aligned}$$

and with $\sigma'(x) = \sigma(x)(1-\sigma(x))$, we have:

$$\begin{aligned}\frac{\partial c_t}{\partial h_{t-1}} &= \frac{\partial \sigma(\mathbf{W}_c \mathbf{x}_t + \mathbf{W}_h h_{t-1})}{\partial h_{t-1}} \\ &= D\left(\sigma(\mathbf{W}_c \mathbf{x}_t + \mathbf{W}_h h_{t-1}) \odot (1 - \sigma(\mathbf{W}_c \mathbf{x}_t + \mathbf{W}_h h_{t-1}))\right) \mathbf{W}_h^\top\end{aligned}$$

(d) Since ℓ is a scalar and $\mathbf{W_x} \in \mathbb{R}^{m \times n}$, $\frac{\partial \ell}{\partial \mathbf{W_x}} \in \mathbb{R}^{1 \times m \times n}$ (by numerator layout), and

$$\frac{\partial \ell}{\partial \mathbf{W_x}} \in \mathbb{R}^{1 \times m \times n} = \sum_{t=1}^{K} \frac{\partial \ell}{\partial h_t} \left(\frac{\partial \tilde{h}_t}{\partial \mathbf{W_x}} + \sum_{i=1}^{t-1} \left(\prod_{j=1}^{t-1} \frac{\partial h_{i+1}}{\partial h_i} \right) \frac{\partial \tilde{h}_i}{\mathbf{W_x}} \right),$$

where

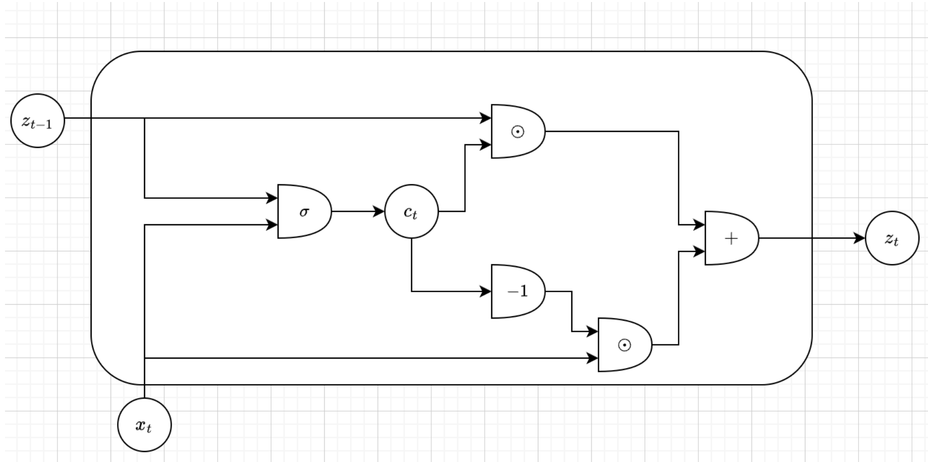
$$\frac{\partial \tilde{h}_t}{\partial \mathbf{W_x}} = (1 - c_t) \odot \tilde{\mathbf{W}},$$

and $\tilde{\mathbf{W}} \in \mathbb{R}^{m \times n \times m}$ such that $\tilde{\mathbf{W}}_i = \begin{bmatrix} 0 \dots 0 & x_t & 0 \dots 0 \end{bmatrix}$ with the first zeros are of length i.

(e)
- The network cannot have exploding gradient since the state vector does not get multiplied with a $c > 1$ constant during each time step.
- The network can have a vanishing gradient, since σ sigmoid function can produce a value close to 0, therefore c_t is a vector of values between 0, 1. This c_t then gets Hadamard multiplied during the time step, therefore the gradient can get arbitrarily small.

1.2.2 Part 2

(a) (Note: the recurrent gated unit graph is the same as the graph above, given the intermediate state z_t)



(b) Replace 2 as k in the original system of equations, we have

$$\text{AttentionRNN}(k) = \begin{cases} q_0[t], q_1[t], \dots, q_k[t] & = Q_0 x[t], Q_1 h[t-1], \dots, Q_k h[t-k] \\ k_0[t], k_1[t], \dots, k_k[t] & = K_0 x[t], K_1 h[t-1], \dots, K_k h[t-k] \\ v_0[t], v_1[t], \dots, v_k[t] & = V_0 x[t], V_1 h[t-1], \dots, V_k h[t-k] \\ w_i[t] & = q_i[t]^\top k_i[t] \\ a[t] & = \text{softargmax}(\{w_0[t], \dots, w_k[t]\}) \\ h[t] & = \sum_{i=1}^{k} a_i[t] v_i[t] \end{cases}$$

(c) In a similar fashion, AttentionRNN(∞) can be defined as AttentionRNN($t-1$), recurrent up to $t-1$ previous time steps at time t:

$$\text{AttentionRNN}(\infty) = \begin{cases} q_0[t], q_1[t], \dots, q_{t-1}[t] & = Q_0 x[t], Q_1 h[t-1], \dots, Q_{t-1} h[1] \\ k_0[t], k_1[t], \dots, k_{t-1}[t] & = K_0 x[t], K_1 h[t-1], \dots, K_{t-1} h[1] \\ v_0[t], v_1[t], \dots, v_{t-1}[t] & = V_0 x[t], V_1 h[t-1], \dots, V_k h[1] \\ w_i[t] & = q_i[t]^\top k_i[t] \\ a[t] & = \text{softargmax}(\{w_0[t], \dots, w_{t-1}[t]\}) \\ h[t] & = \sum_{i=1}^{t-1} a_i[t] v_i[t] \end{cases}$$

(d) Since this NN has different indexing scheme compare to regular CNN, I'll use subscript notation $h[t]$ instead of underscore for h_t in part (d) of CNN question above. For $k = 2$ in AttentionRNN(2), we have

$$\frac{\partial h[t]}{\partial h[t-1]} = \sum_{i=0}^{2} \frac{\partial (a_i[t] v_i[t])}{\partial h[t-1]},$$

where $a_i[t] = \frac{\exp\left(q_i[t]^\top k_0[t]\right)}{\sum_{j=0}^{2} \exp(q_j[t]^\top k_0[t])}$ by the definition of softargmax. Both $a_i[t]$ and $v_i[t]$ depends on $h[t-1]$, so we can write the above expression as

$$\frac{\partial h[t]}{\partial h[t-1]} = \sum_{i=0}^{2} \frac{\partial (a_i[t] v_i[t])}{\partial h[t-1]} = \sum_{i=0}^{2} \left(a_i[t] \frac{\partial v_i[t]}{\partial h[t-1]} + v_i[t] \frac{\partial a_i[t]}{\partial h[t-1]} \right)$$

Each $i = 0, 1, 2$ has different results due to dependency of a_i with respect to $h[t-1]$. By calculation, with $Z = \sum_{j=0}^{2} \exp\left(q_j[t]^T k_0[t]\right)$, we have

- $$\frac{\partial (a_0[t] v_0[t]}{\partial h[t-1]} = -v_0[t] \frac{\partial Z}{Z^2} = -v_0[t](k_1 Q_q + q_1^\top K_1) \frac{\exp\left(q_1[t]^\top k_1[t]\right)}{Z^2}$$
- Similarly,
$$\frac{\partial (a_2[t] v_2[t]}{\partial h[t-1]} = -v_2[t](k_1 Q_q + q_1^\top K_1) \frac{\exp\left(q_1[t]^\top k_1[t]\right)}{Z^2}$$

- Finally, with index 1 we can use product rule as stated above:

$$\frac{\partial(a_1[t]v_1[t]}{\partial h[t-1]} = \frac{k_1 Q_1 + q_t^\top K_q \exp\left(q_1[t]^\top k_1[t]\right)}{Z} - \exp\left(q_1[t]^\top k_1[t]\right)\left(k_1 Q_q + q_1^\top K_1\right)\frac{\exp\left(q_1[t]^\top k_1[t]\right)}{Z^2}$$

(e) For AttentionRNN(k) during loss backpass:

$$\frac{\partial \ell}{\partial h[T]} = \sum_{t>T} \frac{\partial h[t]}{\partial h[T]} \frac{\partial \ell}{\partial h[t]}.$$

1.3 Debugging loss curve

(a) Spike on the left due to exploding gradients.

(b) Exploding gradients can create unstable optimization environment, e.g., let the gradient step onto the different local minima, making the loss spiking point larger than the initial loss.

(c) We can use gradient clipping or regularization to avoid the spikes.

(d) The model is randomly initialized so the accuracy is roughly 1/4 initially.