1.1 Convolutional Neural Networks

(a)

Given by [(W – F + 2P)/S] + 1; where W = original dimension; F = filter dimension; S = stride and P = padding

Thus, [17/4] + 1 **x** [7/4] + 1= 5 **x** 2

(b)

Given by F **x** (*floor*[(H-D*(K-1)+2P- 1)/S] + 1) **x** (*floor*[(W-D*(K-1)+2P - 1)/S] + 1)

(c) (i)

Given by $f_W(x)[r] = \sum(k=1 \text{ to } k=5) \sum (i=1 \text{ to } i=3) x[2(r-1)+ i] [k] W_{1,k,i}$

Dimension is R^3

(ii)

Dimension of $\partial f_W (x)/\partial W$ is $R^{(3)\times(1\times5\times3)}$

$(\partial f_W(x)/\partial W)[r, c, k, i] = x[2(r-1)+ i][k]$

(iii)

Dimension of $\partial f_W(x)/\partial x$ is $R^{(3)\times(5\times7)}$

$(\partial f_W(x)/\partial x)[r, k, i] = W_{1,k,i-2(r-1)}$, when $i - 2(r-1) \in [1, 3]$

= 0 , otherwise

(iv)

Dimension is $R^{1\times5\times3}$

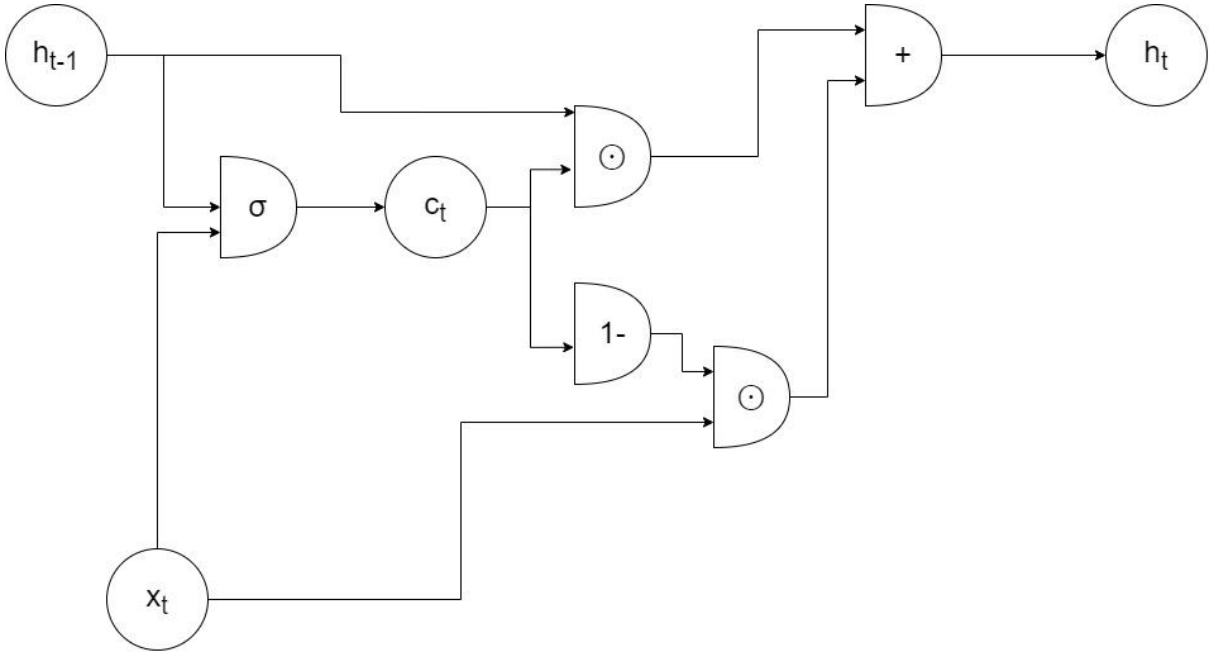
$(\partial l/\partial W)[1, k, i] = \sum (r = 1 \text{ to } r = 2) (\partial l/\partial f_W(x))[r]x[2(r-1) + i, k]$

The similarities are that both forward and backward pass involve a convolution. The differences are in the backward pass, because stride becomes dilation.

1.2 Recurrent Neural Networks

1.2.1 Part 1

(a)



(b)

The dimension of c[t] is the same as the dimension of $W_c x[t]$, which is R^m

(c)

$\partial l/\partial W_x = (\partial l/\partial h[t]) \cdot (\partial h[t]/\partial W_x)$ {Chain Rule}

Now, $\partial h[t]/\partial W_x = (1 - c[t]) \cdot x[t]$

Thus, $\partial l/\partial W_x = (\partial l/\partial h[t]) \cdot (1 - c[t]) \cdot x[t]$.

$\partial l/\partial W_x$ has the same dimension as W_x, which is $R^{(m \times n)}$

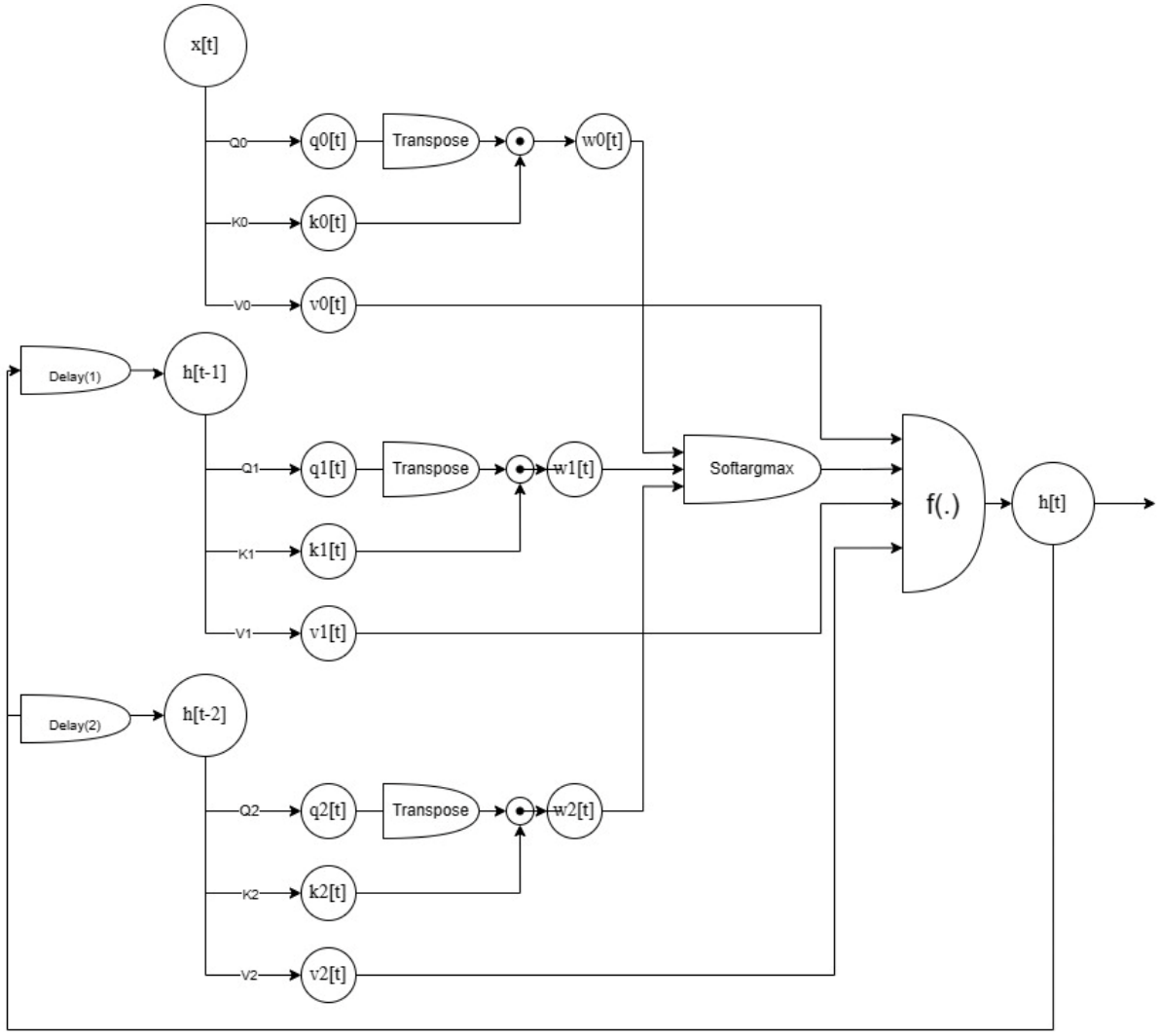
The similarity between the forward and backward pass is that the same computational graph is used but with the inputs and outputs swapped. In the forward pass, we use x[1], … x[t] to calculate h[1], h[2] … h[t] recurrently and in the backward pass we use $\partial l/\partial h[t]$ to calculate the input gradients $\partial l/\partial x[t]$ recurrently.

(d)

This RNN cannot be subject to exploding gradients because the state vector h_t is not multiplied by any matrix from one timestep to the next. It can be subject to vanishing gradients because on each iteration the state vector h_t is multiplied with c_t, which is a vector of values between 0 and 1.

1.2.2 Part 2

(a)



(b)

R^3

(c)

$q_0[t], q_1[t], q_2[t] \dots q_k[t] = Q_0x[t], Q_1h[t-1], Q_2h[t-2] \dots Q_kh[t-k]$

$k_0[t], k_1[t], k_2[t] \dots k_k[t] = K_0x[t], K_1h[t-1], K_2h[t-2] \dots K_kh[t-k]$

$v_0[t], v_1[t], v_2[t] \dots v_k[t] = V_0x[t], V_1h[t-1], V_2h[t-2] \dots V_kh[t-k]$

$w_i[t] = q_i[t]^T k_i[t]$

$a[t] = \text{softargmax}([w_0[t], w_1[t], w_2[t] \dots w_k[t]])$

$h[t] = \sum (i = 0 \text{ to } i = k)\ a_i[t]\ v_i[t]$

(d)

We can share weights for all the Q, K and V

Thus,

$q_0[t], q_1[t], q_2[t] \ldots q_{t-1}[t] = Q_0x[t], Qh[t-1], Qh[t-2] \ldots Qh[1]$

$k_0[t], k_1[t], k_2[t] \ldots k_{t-1}[t] = K_0x[t], Kh[t-1], Kh[t-2] \ldots Kh[1]$

$v_0[t], v_1[t], v_2[t] \ldots v_{t-1}[t] = V_0x[t], Vh[t-1], Vh[t-2] \ldots Vh[1]$

$w_i[t] = q_i[t]^T\ k_i[t]$

$a[t] = \text{softargmax}([w_0[t], w_1[t], w_2[t] \ldots w_{t-1}[t]])$

$h[t] = \sum (i = 0 \text{ to } i = t\text{-}1)\ a_i[t]\ v_i[t]$

(e)

$\partial h[t]/\partial h[t-1] = \partial\ (a_0[t]v_0[t] + a_1[t]v_1[t] + a_2[t]v_2[t])/\partial\ h[t-1]$ **[1]**

Where,

$a_0[t] = \exp(q_0[t]^T k_0[t])/\sum \exp(q_i[t]^T k_i[t])$

$a_1[t] = \exp(q_1[t]^T k_1[t])/\sum \exp(q_i[t]^T k_i[t])$

$a_2[t] = \exp(q_2[t]^T k_2[t])/\sum \exp(q_i[t]^T k_i[t])$

Now,

$\partial\ (a_0[t]v_0[t])/\partial\ h[t-1] = -v_0[t]\ (k_1Q_1 + q_1{}^TK_1)\ (\exp(q_0[t]^T k_0[t]))\ (\exp(q_1[t]^T k_1[t]))/\ (\sum \exp(q_i[t]^T k_i[t]))^2$ **[2]**

Similarly,

$\partial\ (a_2[t]v_2[t])/\partial\ h[t-1] = -v_2[t]\ (k_1Q_1 + q_1{}^TK_1)\ (\exp(q_2[t]^T k_2[t]))\ (\exp(q_1[t]^T k_1[t]))/\ (\sum \exp(q_i[t]^T k_i[t]))^2$ **[3]**

Finally,

$\partial\ (a_1[t]v_1[t])/\partial\ h[t-1] = v_1[t]\ \partial a_1[t]/\partial h[t-1] + a_1[t]\ \partial v_1[t]/\partial h[t-1]$ **[4]**

Now,

$\partial v_1[t]/\partial h[t-1] = V_1$ **[5]**

$\partial a_1[t]/\partial h[t-1] = \{(k_1Q_1 + q_1{}^TK_1)\ (\exp(q_1[t]^T k_1[t]))\}/\ (\sum \exp(q_i[t]^T k_i[t]) -$

$\{(\exp(q_1[t]^T k_1[t]))\ (k_1Q_1 + q_1{}^TK_1)\ (\exp(q_1[t]^T k_1[t]))\}/\ /\ (\sum \exp(q_i[t]^T k_i[t])^2$ **[6]**

We substitute [5] and [6] in [4]. And then [2], [3] and [4] in [1] to get the final expression.

(Not written here in one go for the sake of clarity)

(f)

$\sum_{i=1}^{i=k} \left(\frac{\partial h[i+T]}{\partial h[T]}\right) \left(\frac{\partial l}{\partial h[i+T]}\right)$

1.3 Debugging loss curves

(1)

As discussed in the lecture, this may be happening due to exploding gradients.

(2)

This is possible when the labels are being incorrectly assigned. Initially when the weights are randomly initialized the loss is given by ln(number of classes), however if say the labels are swapped completely (i.e in a binary classifier 1s are predicted as 0s and 0s are predicted as 1s) then the model loss and accuracy will be worse than a randomly initialized model. Such a situation may arise due to several reasons related to the training process and hyperparameters.

(3)

Some ways to fix them is using regularization techniques and gradient clipping.

(4)

Initial loss = ln(4) = 1.386

Initial accuracy = ¼ = 0.25