Akash Tadwai
Indian Institute of Technology Hyderabad
Deep Learning for Vision
ES18BTECH11019

भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

Assignment-IV

Akash Tadwai - ES18BTECH11019

March 31, 2021

1 BackProp through Time

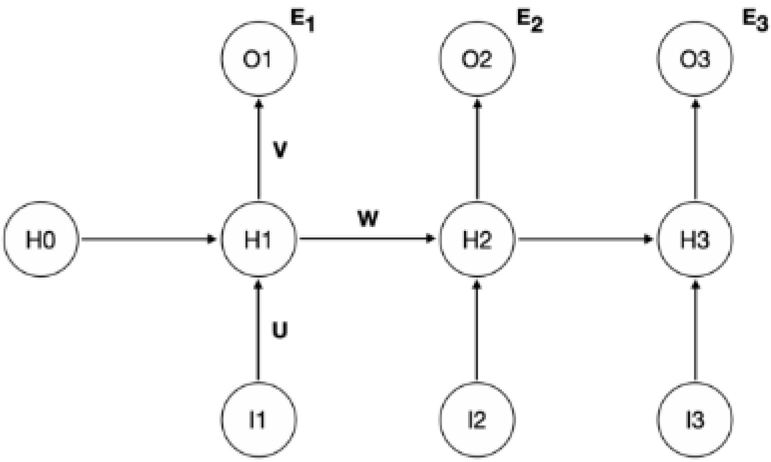

Figure 1: BPTT

- The formulations for an RNN are as follows:

$$h_t = \tanh[UI_t + Wh_{t-1}]$$
$$\hat{y}_t = softmax(Vh_t)$$

- Assuming that *xent* loss is used, we can write the derivative of E_t with respect to h_t as follows:

$$\begin{aligned}\frac{\partial E_t}{\partial h_t} &= \frac{\partial E_t}{\partial \hat{y}_t} * \frac{\partial \hat{y}_t}{\partial z_t} * \frac{\partial z_t}{\partial h_t} \\ &= (\hat{y}_t - y_t)V\end{aligned} \tag{1}$$

Akash Tadwai
Indian Institute of Technology Hyderabad
Deep Learning for Vision
ES18BTECH11019

- We also need derivatives of h_t through time, which we can write as follows:

$$\begin{aligned}\frac{\partial h_t}{\partial h_{t-1}} &= \frac{\partial}{\partial h_{t-1}} \tanh[UI_t + Wh_{t-1}] \\ &= \{1 - \tanh^2[UI_t + Wh_{t-1}]\}W \\ &= [1 - h_t^2]W\end{aligned} \tag{2}$$

- Now we calculate $\frac{\partial E_i}{\partial W}, \frac{\partial E_i}{\partial U}, \frac{\partial E_i}{\partial V}$ in increasing order of i and finally sum them up to get $\frac{\partial E}{\partial W}, \frac{\partial E}{\partial U}, \frac{\partial E}{\partial V}$.
- Calculating errors with respect to E_1.

$$\begin{aligned}\frac{\partial E_1}{\partial W} &= \frac{\partial E_1}{\partial h_1} * \frac{\partial h_1}{\partial W} \\ &= (\hat{y_1} - y_1) * V * (1 - h_1^2) * h_0\end{aligned}$$

$$\begin{aligned}\frac{\partial E_1}{\partial U} &= \frac{\partial E_1}{\partial h_1} * \frac{\partial h_1}{\partial U} \\ &= (\hat{y_1} - y_1) * V * (1 - h_1^2) * I_1\end{aligned}$$

$$\begin{aligned}\frac{\partial E_1}{\partial V} &= \frac{\partial E_1}{\partial z_1} * \frac{\partial z_1}{\partial V} \\ &= (\hat{y_1} - y_1) * h_1\end{aligned}$$

1.1 (b)

- $$\begin{aligned}\frac{\partial E_2}{\partial W} &= \frac{\partial E_2}{\partial h_2} \sum_{k=1}^{2} \left[\frac{\partial h_2}{\partial h_k} * \frac{\partial h_k}{\partial W}\right] \\ &= (\hat{y_2} - y_2) * V * \{\frac{\partial h_2}{\partial W} + \left[\frac{\partial h_2}{\partial h_1} * \frac{\partial h_1}{\partial W}\right]\} \\ &= (\hat{y_2} - y_2) * V * \left[(1 - h_2^2)h_1\right] + \left[(1 - h_2^2)W(1 - h_1^2)h_0\right] \\ &= (\hat{y_2} - y_2) * V * (1 - h_2^2) * \{h_1 + W(1 - h_1^2)h_0\}\end{aligned}$$

$$\begin{aligned}\frac{\partial E_2}{\partial U} &= \frac{\partial E_2}{\partial h_2} \sum_{k=1}^{2} [\frac{\partial h_2}{\partial h_k} * \frac{\partial h_k}{\partial U}] \\ &= (\hat{y_2} - y_2) * V * \{\frac{\partial h_2}{\partial U} + [\frac{\partial h_2}{\partial h_1} * \frac{\partial h_1}{\partial U}]\} \\ &= (\hat{y_2} - y_2) * V * \{[(1 - h_2^2)I_2] + [(1 - h_2^2)W(1 - h_1^2)I_1]\} \\ &= (\hat{y_2} - y_2) * V * (1 - h_2^2) * \{I_2 + W(1 - h_1^2)I_1\}\end{aligned}$$

$$\begin{aligned}\frac{\partial E_2}{\partial V} &= \frac{\partial E_2}{\partial z_2} * \frac{\partial z_2}{\partial V} \\ &= (\hat{y_2} - y_2) * h_2\end{aligned}$$

1.2 (c)

- Using the equations for E_1, E_2 and general results obtained, writing the error terms for E_3 as follows:

$$\begin{aligned}\frac{\partial E_3}{\partial W} &= \frac{\partial E_3}{\partial h_3} \sum_{k=1}^{3} [\frac{\partial h_3}{\partial h_k} * \frac{\partial h_k}{\partial W}] \\ &= (\hat{y_3} - y_3) * V * \{\frac{\partial h_3}{\partial W} + [\frac{\partial h_3}{\partial h_2} * \frac{\partial h_2}{\partial W}] + [\frac{\partial h_3}{\partial h_2} * \frac{\partial h_2}{\partial h_1} * \frac{\partial h_1}{\partial W}]\} \\ &= (\hat{y_3} - y_3) * V * \{[(1 - h_3^2)h_2] + [(1 - h_3^2)W(1 - h_2^2)h_1] + [(1 - h_3^2)W(1 - h_2^2)W(1 - h_1^2)h_0]\} \\ &= (\hat{y_3} - y_3) * V * (1 - h_3^2) * [h_2 + W(1 - h_2^2)\{h_1 + W(1 - h_1^2)h_0\}]\end{aligned} \tag{3}$$

$$\begin{aligned}\frac{\partial E_3}{\partial U} &= \frac{\partial E_3}{\partial h_3} \sum_{k=1}^{3} [\frac{\partial h_3}{\partial h_k} * \frac{\partial h_k}{\partial U}] \\ &= (\hat{y_3} - y_3) * V * \{\frac{\partial h_3}{\partial U} + [\frac{\partial h_3}{\partial h_2} * \frac{\partial h_2}{\partial U}] + [\frac{\partial h_3}{\partial h_2} * \frac{\partial h_2}{\partial h_1} * \frac{\partial h_1}{\partial U}]\} \\ &= (\hat{y_3} - y_3) * V * \{[(1 - h_3^2)I_3] + [(1 - h_3^2)W(1 - h_2^2)I_2] + [(1 - h_3^2)W(1 - h_2^2)W(1 - h_1^2)I_1]\} \\ &= (\hat{y_3} - y_3) * V * (1 - h_3^2) * [I_3 + W(1 - h_2^2)\{I_2 + W(1 - h_1^2)I_1\}]\end{aligned} \tag{4}$$

$$\begin{aligned}\frac{\partial E_3}{\partial V} &= \frac{\partial E_3}{\partial z_3} * \frac{\partial z_3}{\partial V} \\ &= (\hat{y_3} - y_3) * h_3\end{aligned} \tag{5}$$

1.3 (a)

- We can find the total derivative of E with respect to W, U, V as follows:

$$\begin{aligned}\frac{\partial E}{\partial W} &= \sum_{k=1}^{3} \frac{\partial E_k}{\partial W} \\ &= (\hat{y_1} - y_1) * V * (1 - h_1^2) * h_0 + (\hat{y_2} - y_2) * V * (1 - h_2^2) * \{h_1 + W(1 - h_1^2)h_0\} \\ &\quad + (\hat{y_3} - y_3) * V * (1 - h_3^2) * [h_2 + W(1 - h_2^2)\{h_1 + W(1 - h_1^2)h_0\}]\end{aligned} \tag{6}$$

Akash Tadwai
Indian Institute of Technology Hyderabad
Deep Learning for Vision
ES18BTECH11019

$$\begin{aligned}\frac{\partial E}{\partial U} &= \sum_{k=1}^{3} \frac{\partial E_k}{\partial U} \\ &= (\hat{y_1} - y_1) * V * (1 - h_1^2) * I_1 + (\hat{y_2} - y_2) * V * (1 - h_2^2) * \{I_2 + W(1 - h_1^2)I_1\} \\ &\quad + (\hat{y_3} - y_3) * V * (1 - h_3^2) * [I_3 + W(1 - h_2^2)\{I_2 + W(1 - h_1^2)I_1\}]\end{aligned} \tag{7}$$

$$\begin{aligned}\frac{\partial E}{\partial V} &= \sum_{k=1}^{3} \frac{\partial E_k}{\partial V} \\ &= (\hat{y_1} - y_1) * h_1 + (\hat{y_2} - y_2) * h_2 + (\hat{y_3} - y_3) * h_3\end{aligned} \tag{8}$$

2 Vanishing and Exploding Gradients

2.1 Reason for Vanishing Gradients

- From the previous question, the derivative of the error term E_3 wrt W is given by,

$$\frac{\partial E_3}{\partial W} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial h_3} \left(\prod_{j=k+1}^{3} \frac{\partial h_j}{h_{j-1}} \right) \frac{\partial h_k}{\partial W}$$

- Each of these values is upper bounded by 1, which implies that a **cascaded multiplication** of these terms as shown in the above loss function will be **close to 0**. Hence they suffer from the Vanishing Gradients Problem.

We can solve this vanishing gradient problem in RNN by

- By changing Activation to ReLU.
- Using only short time sequences.
- Using architectures such as LSTMs and GRUs

2.2 Text Series Prediction

2.2.1 Problems with Dataset

- The dataset has repetitive words. In the presence of repetitive words, the network has to capture the long term dependencies to predict the sequence accurately.
- To capture long dependencies, it has to back propagate through time over a long interval, which leads to vanishing gradient problem as discussed in the previous answer.

2.2.2 Modifications

- Same as before, We can reduce the effect of vanishing gradients by introducing ReLU activation instead of tanh.
- We can instead use an LSTM, since it avoids vanishing gradients by having ”highway” connections for the cell states.

3 Novel Object Detector

- Given that there are 5 samples belonging to roses class. From the given data we calculate some quantities as follows (exception for Rank=0):
  - **CTP** - Cumulative True Positives
  - **CFP** - Cumulative False Positives
  - **Precision** - $\frac{CTP}{CTP+CFP}$
  - **Recall** - $\frac{CTP}{CTP+CFN}$ which is same as $\frac{CTP}{Total\ Bounding\ Boxes}$

Rank	CTP	CFP	Precision	Recall
0	0	0	0	0
1	1	0	1	1/5
2	2	0	1	2/5
3	2	1	2/3	2/5
4	2	2	1/2	2/5
5	2	3	2/5	2/5
6	3	3	1/2	3/5
7	4	3	4/7	4/5
8	4	4	1/2	4/5
9	4	5	4/9	4/5
10	5	5	1/2	1

- From the above tables we can evaluate interpolated precision using the formula:

$$AP = \frac{1}{11} \sum_{r \in \{0, 0.1, \ldots, 1\}} p_{interp}(r) \tag{9}$$

where,

$$p_{interp}(r) = \max_{\tilde{r}: \tilde{r} \geq r} p(\tilde{r}) \tag{10}$$

- Using the formulae we can tabulate p_{interp} as follows:

Akash Tadwai
Indian Institute of Technology Hyderabad
Deep Learning for Vision
ES18BTECH11019

Recall Level	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
$p_{interp}(r)$	1	1	1	1	1	4/7	4/7	4/7	4/7	1/2	1/2

- From the above table, we can calculate AP as follows:

$$\begin{aligned} AP &= \frac{1}{11}[1+1+1+1+1+\frac{4}{7}+\frac{4}{7}+\frac{4}{7}+\frac{4}{7}+\frac{1}{2}+\frac{1}{2}] \\ &= \frac{58}{77} \\ &= 0.753 \end{aligned} \tag{11}$$

4 Focal Loss

- When $\gamma = 0$, the focal loss is same as cross entropy loss.

$$\begin{aligned} FL &= -(1-p_t)^\gamma \log(p_t) \\ &= -\log(p_t) \\ &= CE \end{aligned} \tag{12}$$

- When an example is misclassified, $p_t \to 0$ and hence the loss will have similar effects as CE Loss. Whereas when $p_t \to 1$, the factor goes to 0 and the loss for well-classified examples is downweighted.
- γ smoothly adjusts the rate at which easy examples are downweighted.

5 L_2 Norm and IoU

We can have same L_2 norm with different IoU values as follows,

- Let us assume that the bounding box is denoted by the 4-tuple i.e., (x_1, y_1, x_2, y_2).
- Now let's fix the distance between one of the two corners, say l, and consider a circle of radius, say r, around the other corner of the ground truth bounding box.
- For any predicted bounding box with corresponding opposite corner lying on this circle will have same L_2 norm of $\sqrt{l^2+r^2}$. But these different BBoxes have different IoUs even though they have same L_2 norm.
- This happens because L_2 norm is just one kind of distance b/w BB whereas IoU measures the extent of Overlap b/w them.

Akash Tadwai
Indian Institute of Technology Hyderabad
Deep Learning for Vision
ES18BTECH11019

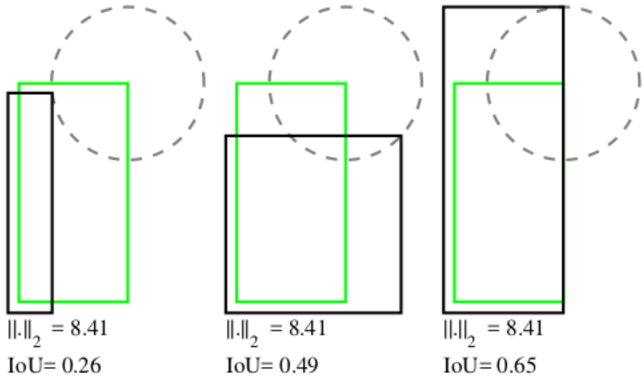

Figure 2: Bounding Boxes with diff IoUs but same L_2 norm
credit: Generalised IoU

6 Transpose Convolution

6.1 OutputSize of Transposed Conv

- The output shape of transpose convolution is given by $out\_size = in\_size * stride - stride + kernel\_size - 2 * padding$.
- As the input size is 3×3 and filter shape is 7×7, the output size will become $3 * 1 - 1 + 7 - 0 = 9$. Hence the output size is 9×9

6.2 2D Transpose Conv in Matrix form

I have followed the idea from the below illustration,

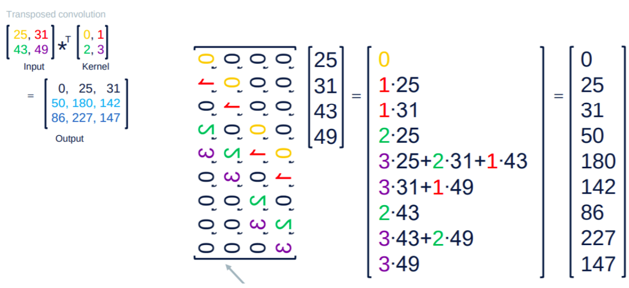

Figure 3: BPTT

- To implement 2D transpose convolution as matrix multiplication, we should first modify the filter into a 4×9 filter.
- The following code snippet achieves the matrix transposition:

```python
def matr_transposition(K):
    k, W = torch.zeros(5), torch.zeros((4, 9))
    k[:2], k[3:5] = K[0, :], K[1, :]
    W[0, :5], W[1, 1:6], W[2, 3:8], W[3, 4:] = k, k, k, k
    return W.T

```

- Using the matrix obtained we can perform tranpose convolution as follows:

```python
def trans_conv(x,K):
    W = matr_transposition(K)
    y = torch.mv(W,x.reshape(-1)).reshape((3,3))
    return y

```

For the LaTeXtyped version of this Report visit : https://www.overleaf.com/read/tngzhnsrrnjf

******THE END******