Recurrent Neural Network

Recurrent Neural Networks are another member of the deep neural networks' architecture family. Unlike a Convolutional structure or a Feedforward structure, RNNs have an internal state vector which allows it to capture temporal (or spatial) dependencies. Throughout the course we will get more familiar with the architecture.

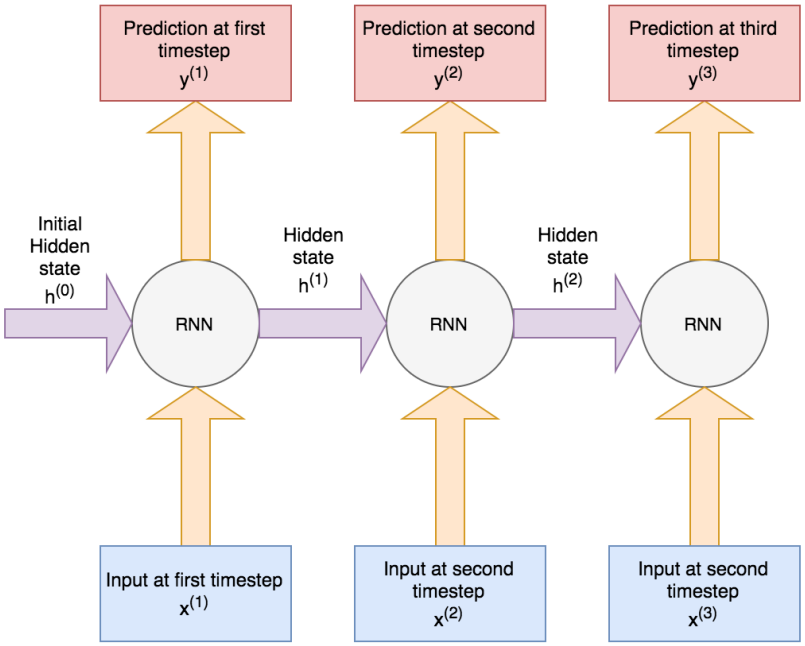

Figure 1: RNNs

In Figure 2 you can see the unfolded network for one timestep.

For an input dictionary size of $|V|$ (e.g. $|V| = 4^k$ for a dictionary of k-mers) and a hidden state with a dimension of D_h, we have:

$$
\begin{aligned}
& x^{(t)} \in \mathbb{R}^{1\times|V|} \\
& e^{(t)} = x^{(t)} L_x^{(t)} \\
& z^{(t)} = h^{(t-1)} H + e^{(t)} I + b_1 \\
& h^{(t)} = \sigma(z^{(t)}) \\
& q^{(t)} = h^{(t)} U + b_2 \\
& \hat{y}^{(t)} = softmax(q^{(t)}) \\
\text{loss: } & J^{(t)} = Cross - Entropy(y^{(t)}, \hat{y}^{(t)})
\end{aligned}
$$

where:

$L_x^{(t)} \in \mathbb{R}^{|V|\times d}$ $H \in \mathbb{R}^{D_h\times D_h}$ $I \in \mathbb{R}^{d\times D_h}$ $b_1 \in \mathbb{R}^{1\times D_h}$ $U \in \mathbb{R}^{D_h\times V}$ $b_2 \in \mathbb{R}^{1\times|V|}$

Here, we will go through the backpropagation algorithm in order to calculate the gradients of the loss function at the third timestep with respect to network weights .

Forward-Propagation

For $t = 3$, given the three inputs $x^{(1)}$, $x^{(2)}$, $x^{(3)}$, the three true labels $y^{(1)}$, $y^{(2)}$, $y^{(3)}$, and the initialized weights of the network, go through the RNN for three timesteps and calculate the predictions $\hat{y}^{(1)}$, $\hat{y}^{(2)}$, $\hat{y}^{(3)}$ and the values of all the other nodes in the network.($e^{(1)}$, $z^{(1)}$, etc).

or forward propagation we define the following:

$$
\begin{aligned}
& e^{(1)} = x^{(1)} L_x^{(1)} \in \mathbb{R}^{1.d} \\
& z^{(1)} = h^{(0)} H + e^{(1)} I + b_1 \in \mathbb{R}^{1.D_h} \\
& h^{(1)} = \sigma(z^{(1)}) \in \mathbb{R}^{1.D_h} \\
& q^{(1)} = h^{(t)} U + b_2 \in \mathbb{R}^{1.|V|} \\
& \hat{y}^{(1)} = softmax(q^{(1)}) \in \mathbb{R}^{1.|V|} \\
& J^{(1)} = CE(y^{(1)}, \hat{y}^{(1)})
\end{aligned}
$$

and the same goes for timesteps 2 and 3.

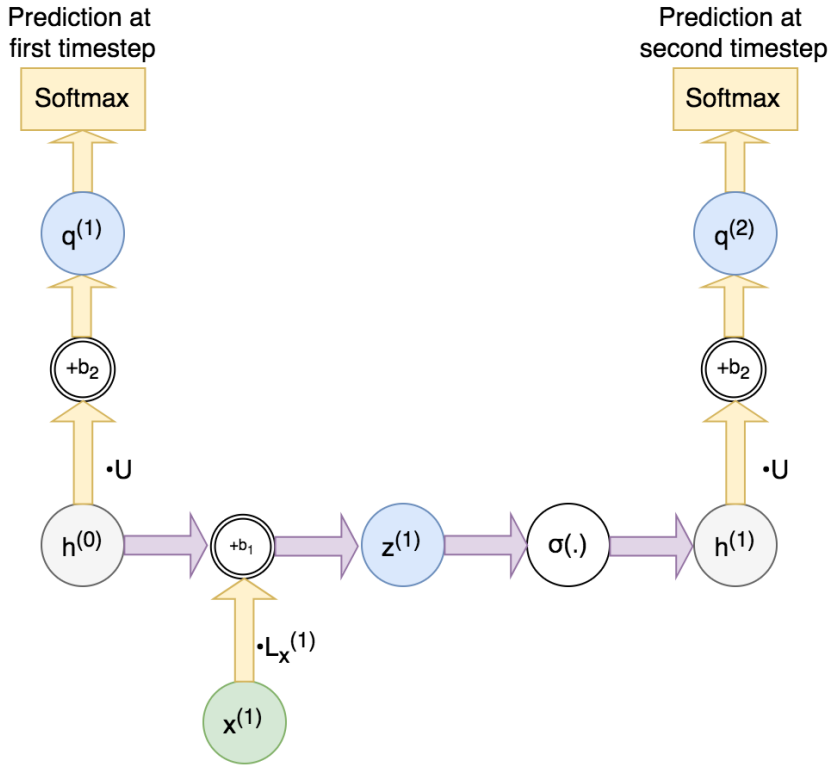

Figure 2: Unfolded network

Backward-Propagation

Now for backward propagation we want to calculate the gradient of the loss in third timestep with respect to network weights. We have:

$$\frac{\partial J^{(3)}}{\partial q^{(3)}} = \hat{y}^{(3)} - y^{(3)} = \delta_1^{(3)}$$
$$\frac{\partial J^{(3)}}{\partial h^{(3)}} = \frac{\partial J^{(3)}}{\partial q^{(3)}} \frac{\partial q^{(3)}}{\partial h^{(3)}} = \delta_1^{(3)} U^T$$
$$\frac{\partial J^{(3)}}{\partial z^{(3)}} = \frac{\partial J^{(3)}}{\partial h^{(3)}} \frac{\partial h^{(3)}}{\partial z^{(3)}} = (\delta_1^{(3)} U^T) \circ h^{(3)} \circ (1 - h^{(3)}) = \delta_2^{(3)}$$

Now for the gradient with respect to some of the network weights we have:

$$
\begin{aligned}
\frac{\partial J^{(3)}}{\partial b_2^{(3)}} &= \frac{\partial J^{(3)}}{\partial q^{(3)}}\frac{\partial q^{(3)}}{\partial b_2} = \delta_1^{(3)} \\
(\frac{\partial J^{(3)}}{\partial H})_{(3)} &= \frac{\partial J^{(3)}}{\partial z^{(3)}}\frac{\partial z^{(3)}}{\partial H} = h^{(3)T}\delta_2^{(3)} \\
(\frac{\partial J^{(3)}}{\partial I})_{(3)} &= \frac{\partial J^{(3)}}{\partial z^{(3)}}\frac{\partial z^{(3)}}{\partial I} = e^{(3)T}\delta_2^{(3)} = L^T x^{(3)T}\delta_2^{(3)} \\
\frac{\partial J^{(3)}}{\partial e^{(3)}} &= \frac{\partial J^{(3)}}{\partial z^{(3)}}\frac{\partial z^{(3)}}{\partial e^{(3)}} = \delta_2^{(3)} I^T \\
\frac{\partial J^{(3)}}{\partial L_x^{(3)}} &= \frac{\partial J^{(3)}}{\partial e^{(3)}}\frac{\partial e^{(3)}}{\partial L_x^{(3)}} = x^{(3)T}\delta_2^{(3)} I^T \\
\frac{\partial J^{(3)}}{\partial h^{(2)}} &= \frac{\partial J^{(3)}}{\partial z^{(3)}}\frac{\partial z^{(3)}}{\partial h^{(2)}} = \delta_2^{(3)} H^T = \delta^{(2)}
\end{aligned}
$$

Where $\circ$ stands for Hadamard (elementwise) product.

However, as the output of the third timestep is also dependent on first and second timesteps, the gradient flow continues its way.

To this point, we had:

$$
\delta^{(2)} = \frac{\partial J^{(3)}}{\partial h^{(2)}} = (((\hat{y}^{(3)} - y^{(3)})U^T) \circ h^{(3)} \circ (1 - h^{(3)}))H^T
$$

continuing the backpropagation we have:

$$
\begin{aligned}
(\frac{\partial J^{(3)}}{\partial H})_{(2)} &= \frac{\partial J^{(3)}}{\partial h^{(2)}}\frac{\partial h^{(2)}}{\partial z^{(2)}}\frac{\partial z^{(2)}}{\partial H} = h^{(t-2)T}(\delta^{(2)} \circ h^{(2)} \circ (1 - h^{(2)})) \\
(\frac{\partial J^{(3)}}{\partial I})_{(2)} &= \frac{\partial J^{(3)}}{\partial h^{(2)}}\frac{\partial h^{(2)}}{\partial z^{(2)}}\frac{\partial z^{(2)}}{\partial I} = e^{(2)T}(\delta^{(2)} \circ h^{(2)} \circ (1 - h^{(2)})) \\
(\frac{\partial J^{(3)}}{\partial L_x^{(2)}}) &= \frac{\partial J^{(3)}}{\partial h^{(2)}}\frac{\partial h^{(2)}}{\partial z^{(2)}}\frac{\partial z^{(2)}}{e^{(2)}}\frac{e^{(2)}}{\partial L_x^{(2)}} = x^{(2)T}(\delta^{(2)} \circ h^{(2)} \circ (1 - h^{(2)}))I^T
\end{aligned}
$$

and going further, we can calculate the gradient of the third timestep's loss with respect to the the network weights at first timestep. The most important take-away is that in the third timestep, the gradient of the loss with respect to each of the weights (H, U, etc) flows back to the very first time step and therefore each of the weights is updated multiple times.