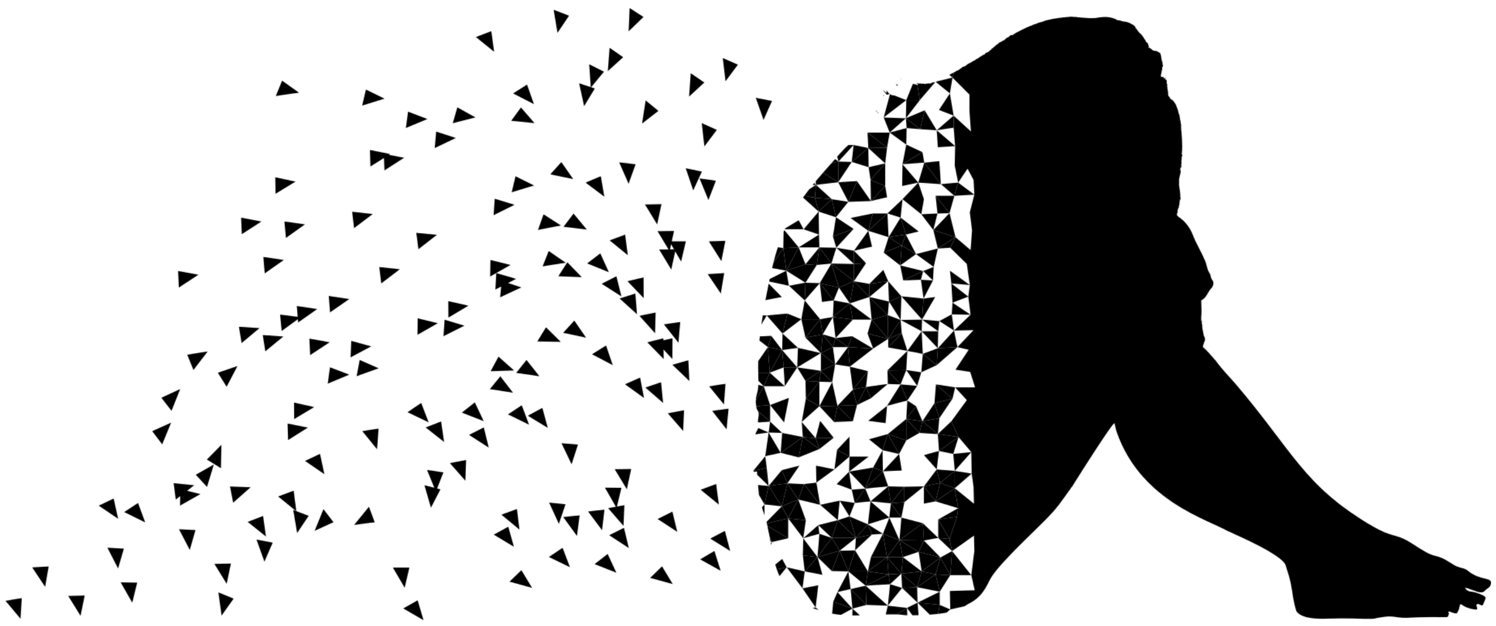
Andrey Sakryukin Follow
Jun 7 · 6 min read



Under The Hood of Neural Networks. Part 2: Recurrent.

In Part 1 of this series, we have studied the Forward and Backward passes of a Feed Forward Fully-Connected network. In spite of the fact, that Feed Forward networks are widespread and find a lot of real-world applications, they have a main limitation. Feed Forward networks cannot handle sequential data. This means that they cannot work with inputs of different sizes and they do not store information about previous states (memory). Thus, in this article, we will talk about Recurrent Neural Networks (RNNs) allowing overcome named limitations.

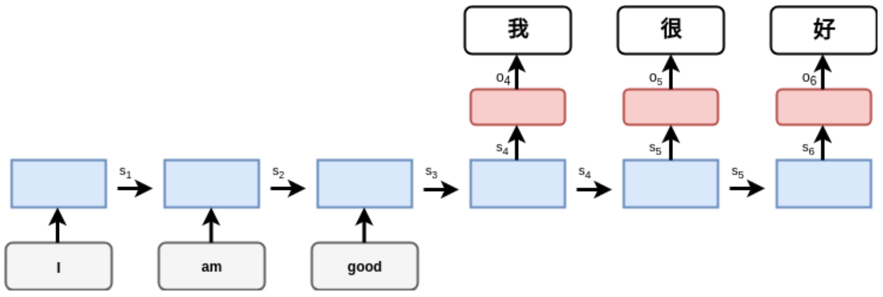
Forward Pass

Technically, *Recurrent Networks* can be represented as Feed Forward networks expanded with a state variable and a recurrent loop. So mathematically each element of the input sequence of RNNs is processed as follows:

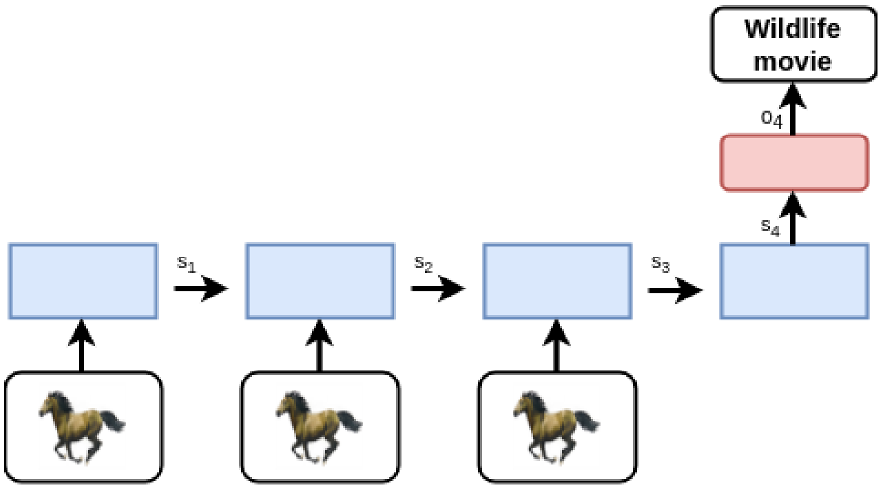
$$\begin{aligned} s_t^{in} &= I_t W_s + s_{t-1} U + B_s \\ s_t &= f_s(s_t^{in}) \\ o_t^{in} &= s_t * W_o + B_o \\ o_t &= f_o(o_t^{in}) \end{aligned}$$

where **It** is the input at time-step **t** (**t**-th element of the input sequence), **st** is the hidden state at time-step **t** and **yt** is the output at time-step t, **fs** and **fo** are activation functions at the respective layers. As we can see from the equation the state of the Recurrent Network is basically an output of the hidden layer. The *recurrence* of the network is explained by the presence of **t-1** term in calculations at time-step **t**. So each calculation of the Recurrent Network depends on previous states and inputs.

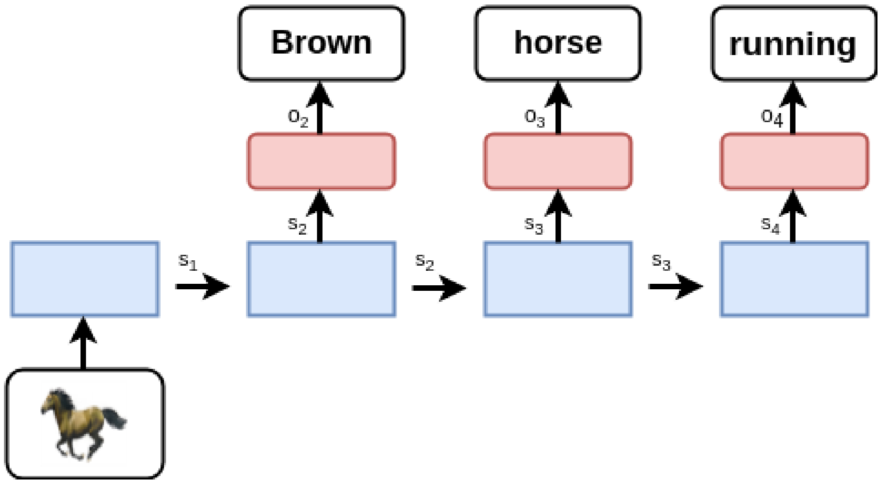
As Recurrent Networks are applied in a variety of real-world problems, we can see different variations of the RNN architecture, which do not significantly alter the math behind. The examples of such application and respective models can be: (a) language translation, (b) video classification, (c) image captioning, (d) autocomplete system.



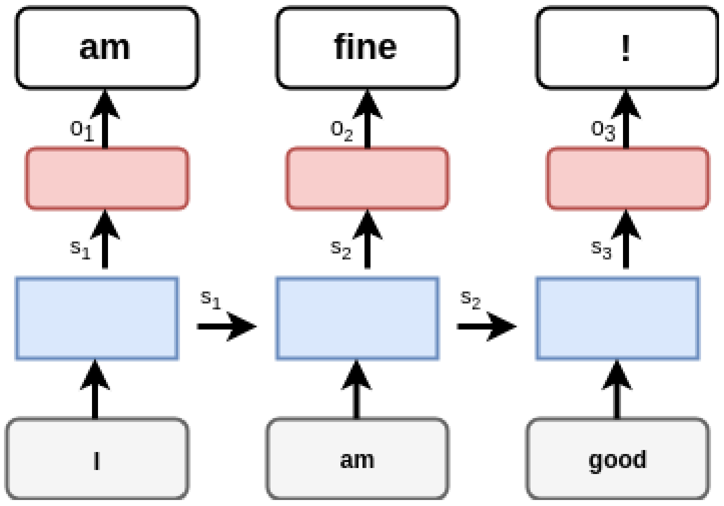
(a) RNN for translation



(b) RNN for video classification



(c) RNN for image captioning



(d) RNN for the autocomplete system

As the mathematical derivation of the training process for the case (a) can be easily applied to the rest of the cases, we will consider a simple example of a sequence traversing network. Suppose we have a sequence of numbers (represented as binary) 1,3,5 and after feeding them one-by-one to our network, we want to get 5,3,1 as an output. First lets randomly initialize the variables:

$$W_s = \begin{bmatrix} -0.75 & 0.50 \\ -0.25 & 0.00 \\ 0.25 & 0.25 \end{bmatrix} \qquad W_o = \begin{bmatrix} -0.90 & 0.45 & 0.00 \\ 0.15 & 0.25 & 0.10 \end{bmatrix} \quad U = \begin{bmatrix} 0.10 & -0.25 \\ 0.50 & 0.50 \end{bmatrix}$$

$$B_s = \begin{bmatrix} 0.50 & 0.35 \end{bmatrix} \qquad\qquad B_o = \begin{bmatrix} 0.80 & 0.60 & -0.25 \end{bmatrix}$$

Note: here, for the sake of simplicity of calculations, we set state size as 2, however state variable is usually a high-dimensional variable.

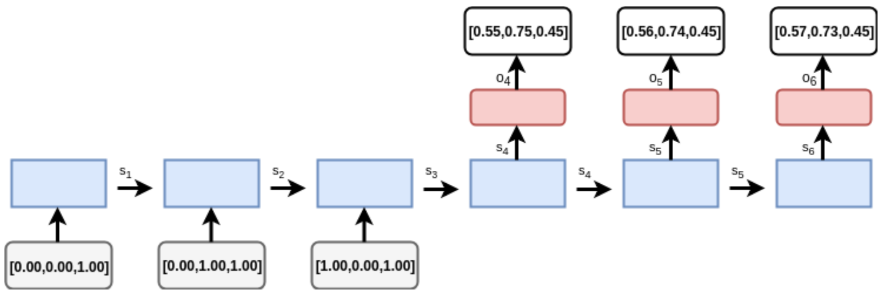
As we can have multiple 1s at the output we will use sigmoid activation function (**fo**), while **fs** will use **tanh**.

Now, having the variables initialized we can calculate the Forward Pass of our Network using equations introduced above. As a result we will get:

$$\begin{aligned}
s_1 &= \tanh(\begin{bmatrix} 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} -0.75 & 0.50 \\ -0.25 & 0.00 \\ 0.25 & 0.25 \end{bmatrix} + \begin{bmatrix} 0.50 & 0.35 \end{bmatrix}) \\
&= \tanh(\begin{bmatrix} 0.75 & 0.60 \end{bmatrix}) = \begin{bmatrix} 0.64 & 0.54 \end{bmatrix} \\
s_2 &= \tanh(\begin{bmatrix} 0 & 1 & 1 \end{bmatrix} * \begin{bmatrix} -0.75 & 0.50 \\ -0.25 & 0.00 \\ 0.25 & 0.25 \end{bmatrix} + \begin{bmatrix} 0.64 & 0.54 \end{bmatrix} * \begin{bmatrix} 0.10 & -0.25 \\ 0.50 & 0.50 \end{bmatrix} + \begin{bmatrix} 0.50 & 0.35 \end{bmatrix}) \\
&= \begin{bmatrix} 0.68 & 0.61 \end{bmatrix} \\
s_3 &= \tanh(\begin{bmatrix} 1 & 0 & 1 \end{bmatrix} * \begin{bmatrix} -0.75 & 0.50 \\ -0.25 & 0.00 \\ 0.25 & 0.25 \end{bmatrix} + \begin{bmatrix} 0.68 & 0.61 \end{bmatrix} \begin{bmatrix} 0.10 & -0.25 \\ 0.50 & 0.50 \end{bmatrix} + \begin{bmatrix} 0.50 & 0.35 \end{bmatrix}) \\
&= \begin{bmatrix} 0.36 & 0.84 \end{bmatrix} \\
s_4 &= \tanh(\begin{bmatrix} 0.36 & 0.84 \end{bmatrix} * \begin{bmatrix} 0.10 & -0.25 \\ 0.50 & 0.50 \end{bmatrix} + \begin{bmatrix} 0.50 & 0.35 \end{bmatrix}) \\
&= \begin{bmatrix} 0.74 & 0.59 \end{bmatrix} \\
s_5 &= \tanh(\begin{bmatrix} 0.74 & 0.59 \end{bmatrix} * \begin{bmatrix} 0.10 & -0.25 \\ 0.50 & 0.50 \end{bmatrix} + \begin{bmatrix} 0.50 & 0.35 \end{bmatrix}) \\
&= \begin{bmatrix} 0.70 & 0.43 \end{bmatrix} \\
s_6 &= \tanh(\begin{bmatrix} 0.70 & 0.43 \end{bmatrix} * \begin{bmatrix} 0.10 & -0.25 \\ 0.50 & 0.50 \end{bmatrix} + \begin{bmatrix} 0.50 & 0.35 \end{bmatrix}) \\
&= \begin{bmatrix} 0.66 & 0.37 \end{bmatrix}
\end{aligned}$$

$$
\begin{aligned}
o_4 &= \sigma\left(\begin{bmatrix}0.74 & 0.59\end{bmatrix} * \begin{bmatrix}-0.90 & 0.45 & 0.00\\ 0.15 & 0.25 & 0.10\end{bmatrix} + \begin{bmatrix}0.80 & 0.60 & -0.25\end{bmatrix}\right)\\
&= \sigma\left(\begin{bmatrix}0.22 & 1.08 & -0.19\end{bmatrix}\right) = \begin{bmatrix}0.55 & 0.75 & 0.45\end{bmatrix}\\
o_5 &= \sigma\left(\begin{bmatrix}0.70 & 0.43\end{bmatrix} * \begin{bmatrix}-0.90 & 0.45 & 0.00\\ 0.15 & 0.25 & 0.10\end{bmatrix} + \begin{bmatrix}0.80 & 0.60 & -0.25\end{bmatrix}\right)\\
&= \begin{bmatrix}0.56 & 0.74 & 0.45\end{bmatrix}\\
o_6 &= \sigma\left(\begin{bmatrix}0.66 & 0.37\end{bmatrix} * \begin{bmatrix}-0.90 & 0.45 & 0.00\\ 0.15 & 0.25 & 0.10\end{bmatrix} + \begin{bmatrix}0.80 & 0.60 & -0.25\end{bmatrix}\right)\\
&= \begin{bmatrix}0.57 & 0.73 & 0.45\end{bmatrix}
\end{aligned}
$$

Schematically our network looks like:



Backward Pass

The first difference we have in case of Recurrent Networks compared to Feed Forward is that we can also have a sequence as an output. As a result, the loss is then represented as a sum of losses at each output (or an average of them). Consequently:

$$
\begin{aligned}
L &= \sum_t L_t\\
\frac{\partial L}{\partial \theta} &= \sum_t \frac{\partial L_t}{\partial \theta}
\end{aligned}
$$

where **theta** can be represented by any trainable variable.

Here we will again use Cross-Enthropy loss (introduced in Part 1). In this case the loss from our forward pass is **3.30**. The derivative of the loss with respect to **o** before the activation function is:

$$
\frac{\partial L_t}{\partial o_t^{in}} = -y_t \circ (1 - \sigma(o_t^{in})) = -y_t \circ (1 - o_t) = y_t \circ (o_t - 1)
$$

As **Wo** and **Bo** do not depend on previous time-steps and has no recurrent loop, we will start our derivation with them:

$$\begin{aligned}
\frac{\partial L_t}{\partial W_o} &= \frac{\partial L_t}{\partial o_t^{in}} \frac{\partial o_t^{in}}{\partial W_o} = y_t \circ (o_t - 1) \frac{\partial o_t^{in}}{\partial W_o} = y_t \circ (o_t - 1) \frac{\partial s_t W_o + B_o}{\partial W_o} = \\
&= s_t^T y_t \circ (o_t - 1) \\
\frac{\partial L_t}{\partial B_o} &= \frac{\partial L_t}{\partial o_t^{in}} \frac{\partial o_t^{in}}{\partial B_o} = y_t \circ (o_t - 1) \frac{\partial o_t^{in}}{\partial B_o} = y_t \circ (o_t - 1) \frac{\partial s_t W_o + B_o}{\partial B_o} = \\
&= y_t \circ (o_t - 1)
\end{aligned}$$

Proceeding with the rest variables:

$$\begin{aligned}
\frac{\partial L_t}{\partial W_s} &= \frac{\partial L_t}{\partial o_t^{in}} \frac{\partial o_t^{in}}{\partial W_s} = y_t \circ (o_t - 1) \frac{\partial o_t^{in}}{\partial W_s} = y_t \circ (o_t - 1) \frac{\partial s_t W_o + B_o}{\partial W_s} = \\
&= y_t \circ (o_t - 1) W_o^T \frac{\partial s_t}{\partial W_s} = y_t \circ (o_t - 1) W_o^T \frac{\partial f_s(s_{t-1}U + I_t W_s + B_s)}{\partial W_s} = \\
&= y_t \circ (o_t - 1) W_o^T \circ f_s'(s_{t-1}U + I_t W_s + B_s) \frac{\partial s_{t-1}U + I_t W_s + B_s}{\partial W_s} = \\
&= I_t^T y_t \circ (o_t - 1) W_o^T \circ f_s'(s_t^{in}) + y_t \circ (o_t - 1) W_o^T \circ f_s'(s_t^{in}) \frac{\partial s_{t-1}U}{\partial W_s} = \\
&= I_t^T y_t \circ (o_t - 1) W_o^T \circ f_s'(s_t^{in}) + y_t \circ (o_t - 1) W_o^T \circ f_s'(s_t^{in}) U^T \frac{\partial f_s(s_{t-2}U + I_{t-1} W_s + B_s)}{\partial W_s} = \\
&= I_t^T y_t \circ (o_t - 1) W_o^T \circ f_s'(s_t^{in}) + I_{t-1}^T y_t \circ (o_t - 1) W_o^T \circ f_s'(s_t^{in}) U^T \circ f_s'(s_{t-1}^{in}) + \\
&\quad + y_t \circ (o_t - 1) W_o^T \circ f_s'(s_t^{in}) U^T \circ f_s'(s_{t-1}^{in}) U^T \frac{\partial s_{t-3}}{\partial W_s} \\
\Delta_N &= y_N \circ (o_N - 1) W_o^T \circ f_s'(s_N^{in}) \\
\Delta_{t-1} &= \Delta_t U^T \circ f_s'(s_{t-1}^{in}) \\
\frac{\partial L_i}{\partial W_s} &= \sum_{t=0}^{i} I_t \Delta_t
\end{aligned}$$

$$\begin{aligned}
\frac{\partial L_t}{\partial B_s} &= \frac{\partial L_t}{\partial o_t^{in}} \frac{\partial o_t^{in}}{\partial B_s} = y_t \circ (o_t - 1) \frac{\partial o_t^{in}}{\partial B_s} = y_t \circ (o_t - 1) \frac{\partial s_t W_o + B_o}{\partial B_s} = \\
&= y_t \circ (o_t - 1) W_o^T \frac{\partial s_t}{\partial B_s} = y_t \circ (o_t - 1) W_o^T \frac{\partial f_s(s_{t-1}U + I_t W_s + B_s)}{\partial B_s} = \\
&= y_t \circ (o_t - 1) W_o^T \circ f_s'(s_t^{in}) + y_t \circ (o_t - 1) W_o^T \circ f_s'(s_t^{in}) U^T \frac{\partial f_s(s_{t-2}U + I_{t-1} W_s + B_s)}{\partial W_s} = \\
&= \sum_{t=0}^{i} \Delta_t
\end{aligned}$$

$$\begin{aligned}
\frac{\partial L_t}{\partial U} &= \frac{\partial L_t}{\partial o_t^{in}} \frac{\partial o_t^{in}}{\partial U} = y_t \circ (o_t - 1) \frac{\partial o_t^{in}}{\partial U} = y_t \circ (o_t - 1) \frac{\partial s_t W_o + B_o}{\partial U} = \\
&= y_t \circ (o_t - 1) W_o^T \frac{\partial s_t}{\partial U} = y_t \circ (o_t - 1) W_o^T \frac{\partial f_s(s_{t-1}U + I_t W_s + B_s)}{\partial U} = \\
&= s_{t-1}^T y_t \circ (o_t - 1) W_o^T \circ f_s'(s_t^{in}) + y_t \circ (o_t - 1) W_o^T \circ f_s'(s_t^{in}) U^T \frac{\partial f_s(s_{t-2}U + I_{t-1} W_s + B_s)}{\partial U} = \\
&= s_{t-1}^T y_t \circ (o_t - 1) W_o^T \circ f_s'(s_t^{in}) + s_{t-2}^T y_t \circ (o_t - 1) W_o^T \circ f_s'(s_t^{in}) U^T \circ f_s'(s_{t-1}^{in}) + \\
&\quad + s_{t-2}^T y_t \circ (o_t - 1) W_o^T \circ f_s'(s_t^{in}) U^T \circ f_s'(s_{t-1}^{in}) U^T \frac{\partial s_{t-2}}{\partial U} = \\
&= \sum_{t=0}^{i} s_{t-1}^T \Delta_t
\end{aligned}$$

Note: derivative of the **tanh** activation function:

$$\frac{\partial \tanh x}{\partial x} = 1 - \tanh^2 x$$

Now we have all the equations for the calculation of the gradients and applying of the Gradient Descent algorithm (discussed in Part 1). Let's start with the calculation of gradients for each output separately, we will start with output at time-step 4:

$$\begin{aligned}
&L_4 : \\
&\Delta_4 = \begin{bmatrix} 1.0 & 0 & 1.0 \end{bmatrix} \circ \left(\begin{bmatrix} 0.22 & 1.08 & -0.19 \end{bmatrix} - 1 \right) \begin{bmatrix} -0.90 & 0.15 \\ 0.45 & 0.25 \\ 0.00 & 0.10 \end{bmatrix} \circ \left(1 - \begin{bmatrix} 0.55 & 0.35 \end{bmatrix} \right) = \\
&\quad = \begin{bmatrix} -0.78 & -0.00 & -1.19 \end{bmatrix} \begin{bmatrix} -0.90 & 0.15 \\ 0.45 & 0.25 \\ 0.00 & 0.10 \end{bmatrix} \circ \begin{bmatrix} 0.45 & 0.65 \end{bmatrix} = \begin{bmatrix} 0.40 & -0.12 \end{bmatrix} \circ \begin{bmatrix} 0.45 & 0.65 \end{bmatrix} = \\
&\quad = \begin{bmatrix} 0.17 & -0.08 \end{bmatrix} \\
&\Delta_3 = \begin{bmatrix} 0.17 & -0.08 \end{bmatrix} \begin{bmatrix} 0.10 & 0.50 \\ -0.25 & 0.50 \end{bmatrix} \circ \begin{bmatrix} 0.87 & 0.29 \end{bmatrix} = \begin{bmatrix} 0.03 & 0.01 \end{bmatrix} \\
&\Delta_2 = \begin{bmatrix} 0.00 & 0.01 \end{bmatrix} \\
&\Delta_1 = \begin{bmatrix} 0.00 & 0.01 \end{bmatrix}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial L_4}{\partial W_o} &= \begin{bmatrix} 0.74 \\ 0.59 \end{bmatrix} \begin{bmatrix} 1.00 & 0.00 & 1.00 \end{bmatrix} \circ \left(\begin{bmatrix} 0.55 & 0.75 & 0.45 \end{bmatrix} - 1 \right) \\
&= \begin{bmatrix} -0.33 & 0.00 & -0.41 \\ -0.26 & 0.00 & -0.32 \end{bmatrix} \\
\frac{\partial L_4}{\partial B_o} &= \begin{bmatrix} 1.00 & 0.00 & 1.00 \end{bmatrix} \circ \left(\begin{bmatrix} 0.55 & 0.75 & 0.45 \end{bmatrix} - 1 \right) = \begin{bmatrix} -0.45 & 0.00 & -0.55 \end{bmatrix} \\
\frac{\partial L_4}{\partial W_s} &= \left(\begin{bmatrix} 0.00 \\ 0.00 \\ 0.00 \end{bmatrix} \begin{bmatrix} 0.17 & -0.08 \end{bmatrix} + \begin{bmatrix} 1.0 \\ 0 \\ 1.0 \end{bmatrix} \begin{bmatrix} 0.03 & 0.01 \end{bmatrix} + \begin{bmatrix} 0.00 \\ 1.00 \\ 1.00 \end{bmatrix} \begin{bmatrix} 0.00 & 0.01 \end{bmatrix} + \begin{bmatrix} 0.00 \\ 0.00 \\ 1.00 \end{bmatrix} \begin{bmatrix} 0.00 & 0.01 \end{bmatrix} \right) = \\
&= \begin{bmatrix} 0.03 & 0.01 \\ 0.00 & 0.01 \\ 0.03 & 0.03 \end{bmatrix} \\
\frac{\partial L_4}{\partial B_s} &= \left(\begin{bmatrix} 0.17 & -0.08 \end{bmatrix} + \begin{bmatrix} 0.03 & 0.01 \end{bmatrix} + \begin{bmatrix} 0.00 & 0.01 \end{bmatrix} + \begin{bmatrix} 0.00 & 0.01 \end{bmatrix} \right) = \\
&= \begin{bmatrix} 0.21 & -0.04 \end{bmatrix} \\
\frac{\partial L_4}{\partial U} &= \left(\begin{bmatrix} 0.36 \\ 0.84 \end{bmatrix} \begin{bmatrix} 0.17 & -0.08 \end{bmatrix} + \begin{bmatrix} 0.68 \\ 0.61 \end{bmatrix} \begin{bmatrix} 0.03 & 0.01 \end{bmatrix} + \begin{bmatrix} 0.75 \\ 0.60 \end{bmatrix} \begin{bmatrix} 0.00 & 0.01 \end{bmatrix} + \begin{bmatrix} 0.00 \\ 0.00 \end{bmatrix} \begin{bmatrix} 0.00 & 0.01 \end{bmatrix} \right) = \\
&= \begin{bmatrix} 0.09 & -0.01 \\ 0.17 & -0.05 \end{bmatrix}
\end{aligned}$$

Performing the same calculation for outputs at time-steps 5 and 6, we will get:

$L_5:$

$$\Delta_5 = \begin{bmatrix}-0.06 & -0.10\end{bmatrix}$$
$$\Delta_4 = \begin{bmatrix}0.01 & -0.05\end{bmatrix}$$
$$\Delta_3 = \begin{bmatrix}0.01 & -0.01\end{bmatrix}$$
$$\Delta_2 = \begin{bmatrix}0.00 & 0.00\end{bmatrix}$$
$$\Delta_1 = \begin{bmatrix}0.00 & 0.00\end{bmatrix}$$
$$\frac{\partial L_5}{\partial W_o} = \begin{bmatrix}0.00 & -0.19 & -0.39\\ 0.00 & -0.11 & -0.24\end{bmatrix}$$
$$\frac{\partial L_5}{\partial B_o} = \begin{bmatrix}0.00 & -0.26 & -0.55\end{bmatrix}$$
$$\frac{\partial L_5}{\partial W_s} = \begin{bmatrix}0.01 & -0.01\\ 0.00 & 0.00\\ 0.01 & 0.00\end{bmatrix}$$
$$\frac{\partial L_5}{\partial B_s} = \begin{bmatrix}-0.04 & -0.15\end{bmatrix}$$
$$\frac{\partial L_5}{\partial U} = \begin{bmatrix}-0.03 & -0.09\\ -0.02 & -0.10\end{bmatrix}$$

$L_6:$

$$\Delta_6 = \begin{bmatrix}0.00 & -0.05\end{bmatrix}$$
$$\Delta_5 = \begin{bmatrix}0.01 & -0.02\end{bmatrix}$$
$$\Delta_4 = \begin{bmatrix}0.00 & 0.00\end{bmatrix}$$
$$\Delta_3 = \begin{bmatrix}0.00 & 0.00\end{bmatrix}$$
$$\Delta_2 = \begin{bmatrix}0.00 & 0.00\end{bmatrix}$$
$$\Delta_1 = \begin{bmatrix}0.00 & 0.00\end{bmatrix}$$
$$\frac{\partial L_6}{\partial W_o} = \begin{bmatrix}0.00 & 0.00 & -0.36\\ 0.00 & 0.00 & -0.21\end{bmatrix}$$
$$\frac{\partial L_6}{\partial B_o} = \begin{bmatrix}0.00 & 0.00 & -0.55\end{bmatrix}$$
$$\frac{\partial L_6}{\partial W_s} = \begin{bmatrix}0.00 & 0.00\\ 0.00 & 0.00\\ 0.00 & 0.00\end{bmatrix}$$
$$\frac{\partial L_6}{\partial B_s} = \begin{bmatrix}0.01 & -0.07\end{bmatrix}$$
$$\frac{\partial L_6}{\partial U} = \begin{bmatrix}0.01 & -0.05\\ 0.01 & -0.04\end{bmatrix}$$

Having all the needed gradients we can calculate the final gradients with respect to the total loss as a sum of gradients with respect to every separate loss:

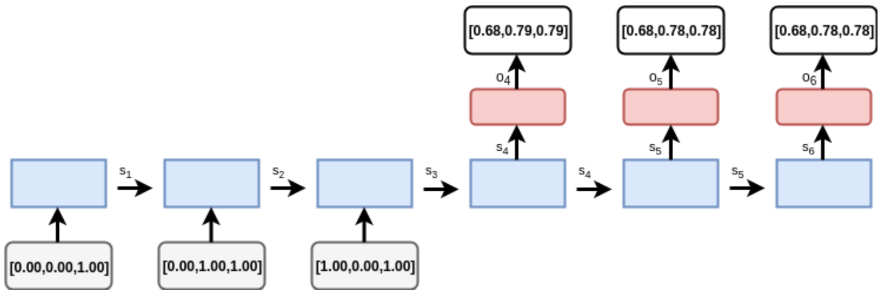
$$\frac{\partial L}{\partial W_o} = \begin{bmatrix} -0.33 & -0.19 & -1.16 \\ -0.26 & -0.11 & -0.77 \end{bmatrix}$$
$$\frac{\partial L}{\partial B_o} = \begin{bmatrix} -0.45 & -0.26 & -1.65 \end{bmatrix}$$
$$\frac{\partial L}{\partial W_s} = \begin{bmatrix} 0.05 & 0.01 \\ 0.00 & 0.02 \\ 0.04 & 0.03 \end{bmatrix}$$
$$\frac{\partial L}{\partial B_s} = \begin{bmatrix} 0.18 & -0.27 \end{bmatrix}$$
$$\frac{\partial L}{\partial U} = \begin{bmatrix} 0.06 & -0.15 \\ 0.16 & -0.19 \end{bmatrix}$$

After application of the Gradient Descent algorithm with learning rate of 0.5 we will get new variables:

$$W_s = \begin{bmatrix} -0.77 & 0.50 \\ -0.25 & -0.01 \\ 0.23 & 0.23 \end{bmatrix} \quad W_o = \begin{bmatrix} -0.73 & 0.54 & 0.58 \\ 0.28 & 0.31 & 0.48 \end{bmatrix} \quad U = \begin{bmatrix} 0.07 & -0.17 \\ 0.42 & 0.60 \end{bmatrix}$$

$$B_s = \begin{bmatrix} 0.41 & 0.48 \end{bmatrix} \qquad B_o = \begin{bmatrix} 1.02 & 0.73 & 0.58 \end{bmatrix}$$

Now, updated network will result in following Forward Pass:



And the new total loss equals **1.36**.

Summary

In this article, I have introduced *Recurrent Neural Networks* (RNNs) and derived the essential equations needed for the training process. This type of network is particularly useful for the sequence processing tasks.

In spite of the fact that the training loss after the Backward Pass significantly reduced, the training process of such a network could still be tricky. First of all, the dimensionality of the state should be increased. More importantly classical (or vanilla) RNNs have difficulties remembering old states. To tackle this problem exists a number of solutions. The most famous among them, probably, would be:

1. adding residual connections between states (so that every state depends not only on the previous state, but f.e. on t-2 or t-3 state as well)
2. using Long-Short Term Memory (LSTM) cells, which needs a separate article to be described.

In the next part, I will describe Convolutional Neural Networks, which became an essential part of modern Deep Learning and Computer Vision fields.

Don’t forget to clap if you found this article useful and follow me to see more posts like this!

Please share your feedback and ideas on future posts in the comments below!

Thanks for reading!