Recurrent Neural Network for Sine Wave Prediction

Saikat Pal

Indian Institute of Technology Delhi

Prof. Sitikantha Roy

April 28, 2023

Outline

- Recurrent Neural Networks handle **sequence data** to predict the next event.
- The problem becomes much easier when we have a history of previous locations. This is why sequence models are handy in processing **time-series data** like language or audio files.

- The code template of RNN explained is a **many-to-one** type of architecture.

- The input dataset is the value of a **sinusoidal function** for 25-time steps. Such 175 samples each having 25 time-steps data is our entire training dataset. The shape of x would be (175,25,1).
- Each sample output would be the **26th-time step** value of the sinusoidal function. The shape of y would be (175,1).

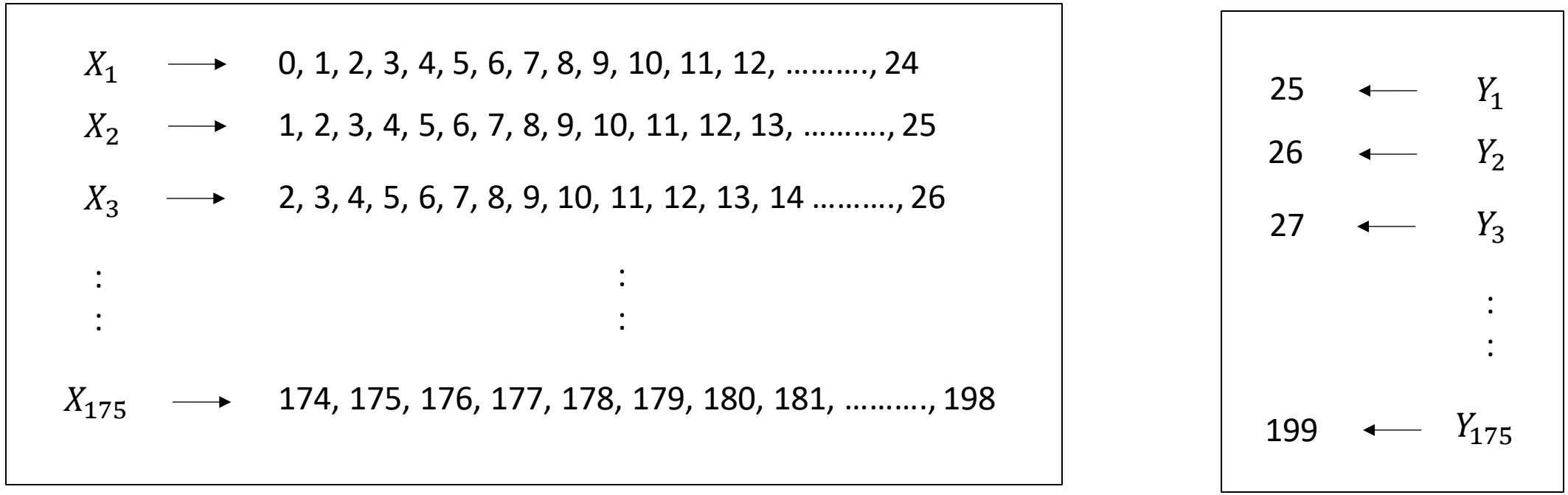
- **Hidden Units** (a variable in the forward pass which means the **number of neurons in a hidden state**) is taken as 100.

- Shape of the weight matrix ($\boldsymbol{W_{XH}}$) will be (100,1)

- Shape of the weight matrix ($\boldsymbol{W_{HH}}$) will be (100,100)

- Shape of the weight matrix ($\boldsymbol{W_{HY}}$) will be (1,100)

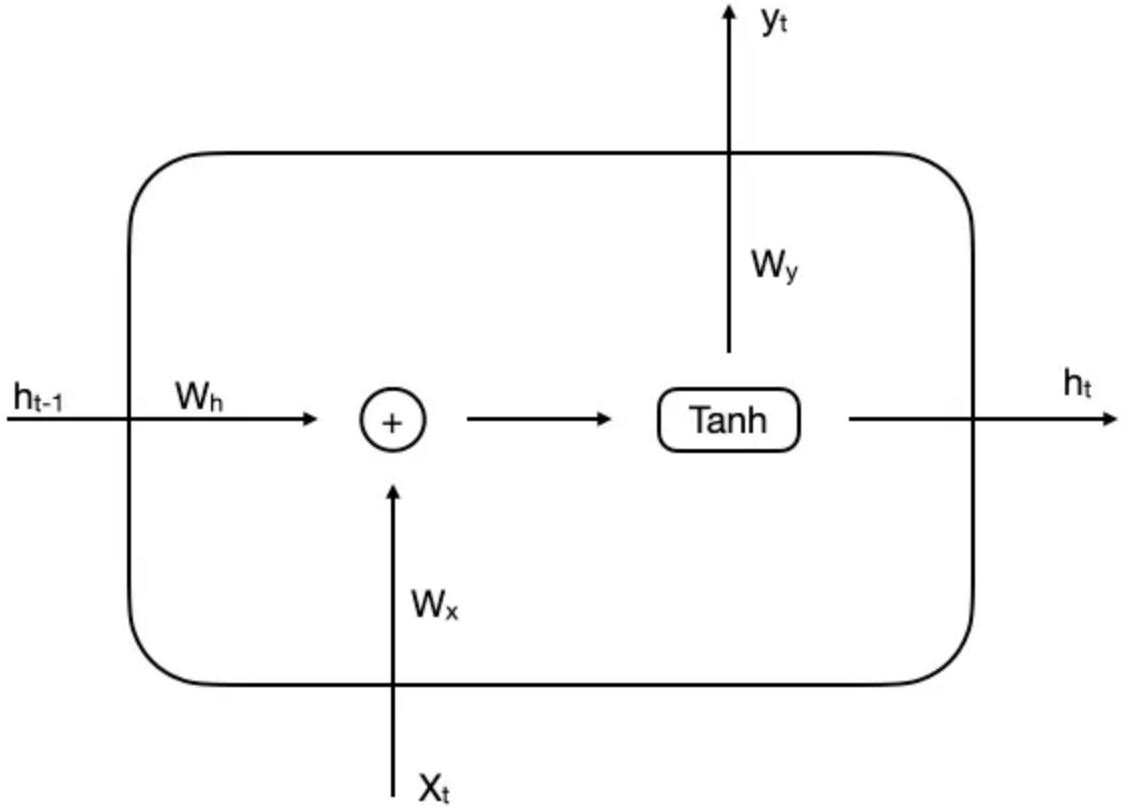
- All the weight matrices are initialized randomly. And the **initial hidden state** should be taken as a **zero vector**.


X_1 → 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, ……., 24
X_2 → 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, ……., 25
X_3 → 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14 ……., 26
X_{175} → 174, 175, 176, 177, 178, 179, 180, 181, ……., 198
25 ← Y_1
26 ← Y_2
27 ← Y_3
199 ← Y_{175}

Forward Pass:-

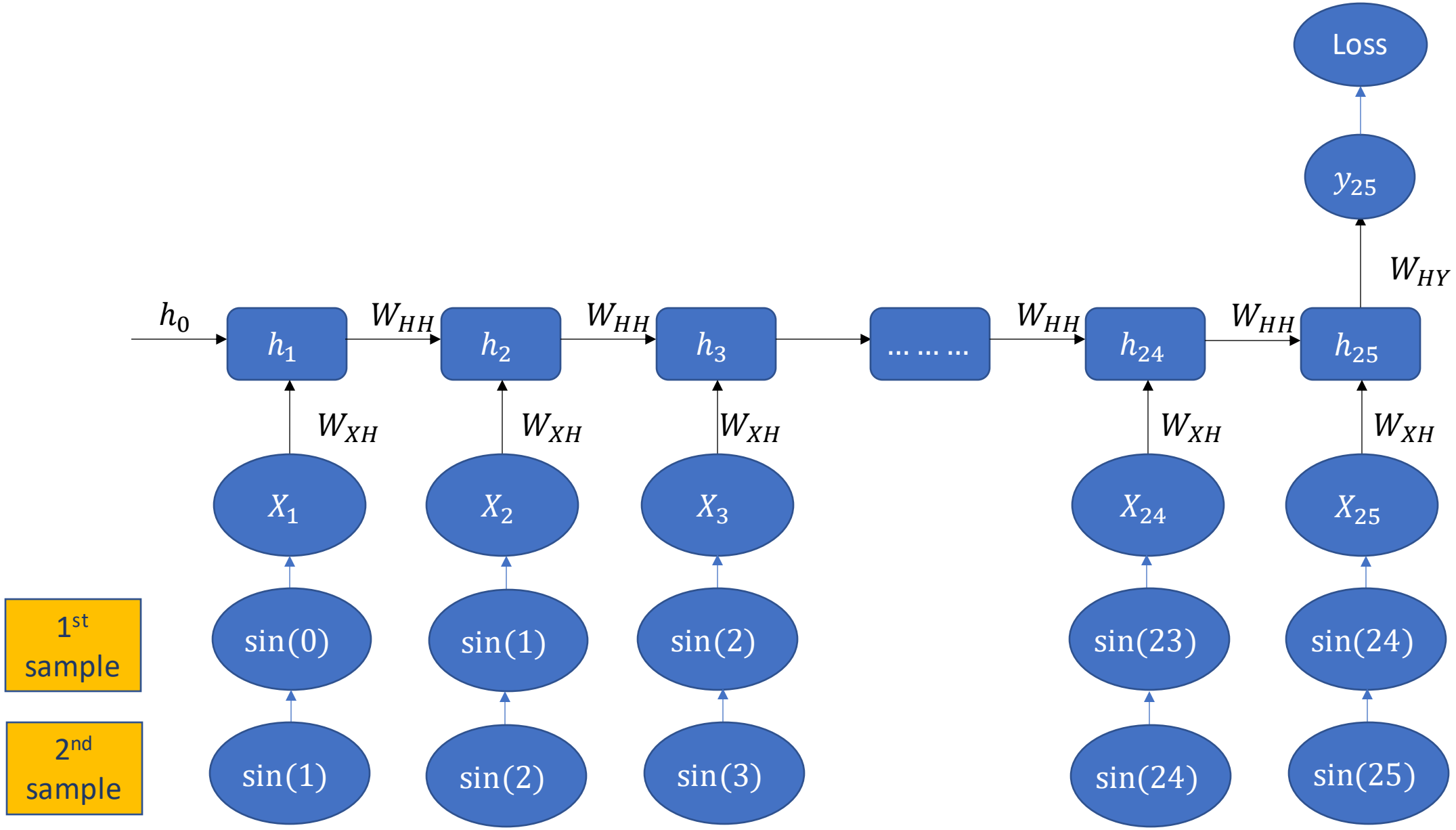
Operations that are performed inside each memory cell are given as:

- $h_t = \tanh(W_{XH} * X_t + W_{HH} * h_{t-1})$
- $y_t = W_{HY} * h_t$
- These two operations need to be written in the definition cell in the class RNN.
- All the h_t needs to be stored for the back-propagation through time (BPTT). In the template, they are all getting saved inside **hidden_states.**
- **Loss function is** $\boldsymbol{L = \frac{1}{2}\left(y_t - sample_y\right)^2}$



Memory Cell

Architecture:-

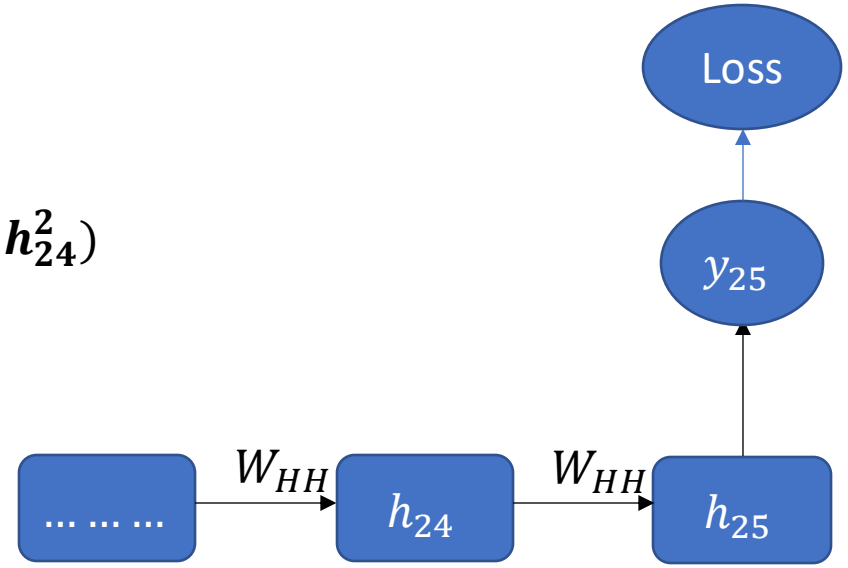


Back Propagation Through Time (BPTT):-

$$\frac{dL}{dW_{HH}} = \left[\frac{\partial L}{\partial y_{25}}\ \frac{\partial y_{25}}{\partial h_{25}}\ \frac{\partial h_{25}}{\partial W_{HH}}\right] + \left[\frac{\partial L}{\partial y_{25}}\ \frac{\partial y_{25}}{\partial h_{25}}\ \frac{\partial h_{25}}{\partial h_{24}}\ \frac{\partial h_{24}}{\partial W_{HH}}\right] + \left[\frac{\partial L}{\partial y_{25}}\ \frac{\partial y_{25}}{\partial h_{25}}\ \frac{\partial h_{25}}{\partial h_{24}}\ \frac{\partial h_{24}}{\partial h_{23}}\ \frac{\partial h_{23}}{\partial W_{HH}}\right] + \dots\dots$$

$$= \left[error\ * W_{HY}\ * \left(1 - h_{25}^2\right)h_{24}\right] + \left[error\ * W_{HY} * \left(1 - h_{25}^2\right)W_{HH} * (1 - h_{24}^2)h_{23}\right] + \cdots$$

- Initialise dht in the code as $dht = error * W_{HY}$
- In first term, which means for $step = 24$, temp stores $\mathbf{temp} = \boldsymbol{error} * \boldsymbol{W_{HY}} * \left(\mathbf{1} - \boldsymbol{h_{25}^2}\right) = \boldsymbol{dht} * (\mathbf{1} - \boldsymbol{h_{25}^2})$
  therefore, first term would be $\boldsymbol{temp} * \boldsymbol{h_{24}}$
- Then update the $\boldsymbol{dht}$ in the code as $\boldsymbol{dht} = \boldsymbol{temp} * \boldsymbol{W_{HH}}$
- In second term, which means for $step = 23$, temp stores $\mathbf{temp} = \boldsymbol{dht} * (\mathbf{1} - \boldsymbol{h_{24}^2})$
  therefore, second term would be $\boldsymbol{temp} * \boldsymbol{h_{23}}$

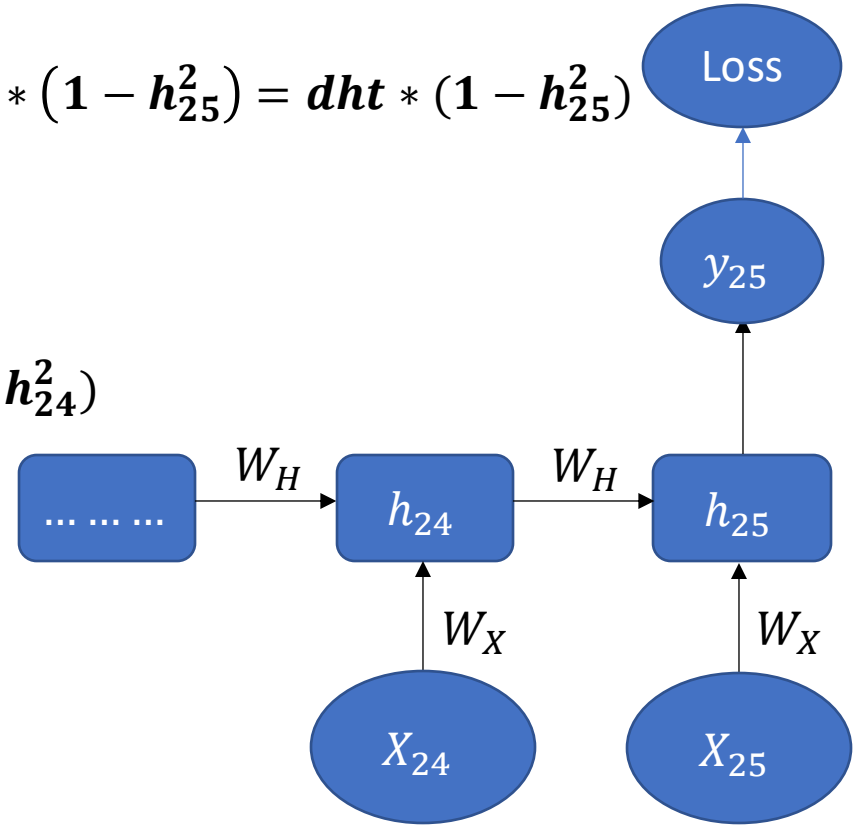


Back Propagation Through Time (BPTT):-

$$\frac{dL}{dW_{XH}} = \left[\frac{\partial L}{\partial y_{25}}\ \frac{\partial y_{25}}{\partial h_{25}}\ \frac{\partial h_{25}}{\partial W_{XH}}\right] + \left[\frac{\partial L}{\partial y_{25}}\ \frac{\partial y_{25}}{\partial h_{25}}\ \frac{\partial h_{25}}{\partial h_{24}}\ \frac{\partial h_{24}}{\partial W_{XH}}\right] + \left[\frac{\partial L}{\partial y_{25}}\ \frac{\partial y_{25}}{\partial h_{25}}\ \frac{\partial h_{25}}{\partial h_{24}}\frac{\partial h_{24}}{\partial h_{23}}\frac{\partial h_{23}}{\partial W_{XH}}\right] + \ldots\ldots$$

$$= \left[error\ * W_{HY}\ * \left(1 - h_{25}^2\right)X_{25}\right] + \left[error\ * W_{HY} * \left(1 - h_{25}^2\right)W_{HH} * (1 - h_{24}^2)X_{24}\right] + \cdots$$

- Initialise dht in the code as $dht = error * W_{HY}$
- In first term, which means for $step = 24$, temp stores $\mathbf{temp} = \boldsymbol{error} * \boldsymbol{W_{HY}} * \left(\mathbf{1} - \boldsymbol{h_{25}^2}\right) = \boldsymbol{dht} * (\mathbf{1} - \boldsymbol{h_{25}^2})$
  therefore, first term would be $\boldsymbol{temp} * \boldsymbol{X_{25}}$
- Then update the $\boldsymbol{dht}$ in the code as $\boldsymbol{dht} = \boldsymbol{temp} * \boldsymbol{W_{HH}}$
- In second term, which means for $step = 23$, temp stores $\mathbf{temp} = \boldsymbol{dht} * (\mathbf{1} - \boldsymbol{h_{24}^2})$
  therefore, first term would be $\boldsymbol{temp} * \boldsymbol{X_{24}}$



Question:-

- Complete the template of the code given on simple RNN architecture and backpropagation part. Plot the convergence of the training model and also plot the graph of prediction of sinusoidal function over the testing dataset and check how accurate the prediction is with actual value. Also, try using the activation function while calculating y_t as discussed in class to see if the gradients exploding or not are hereby decide clipping of gradients is helping or not in prediction. Also as told in class, please refer to the presentation for writing the codes.