Background

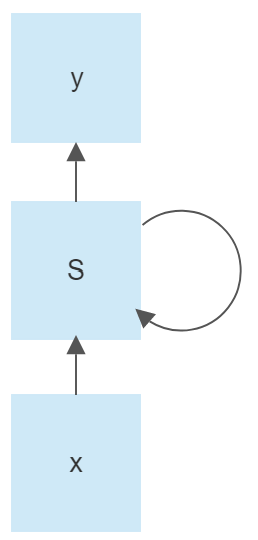
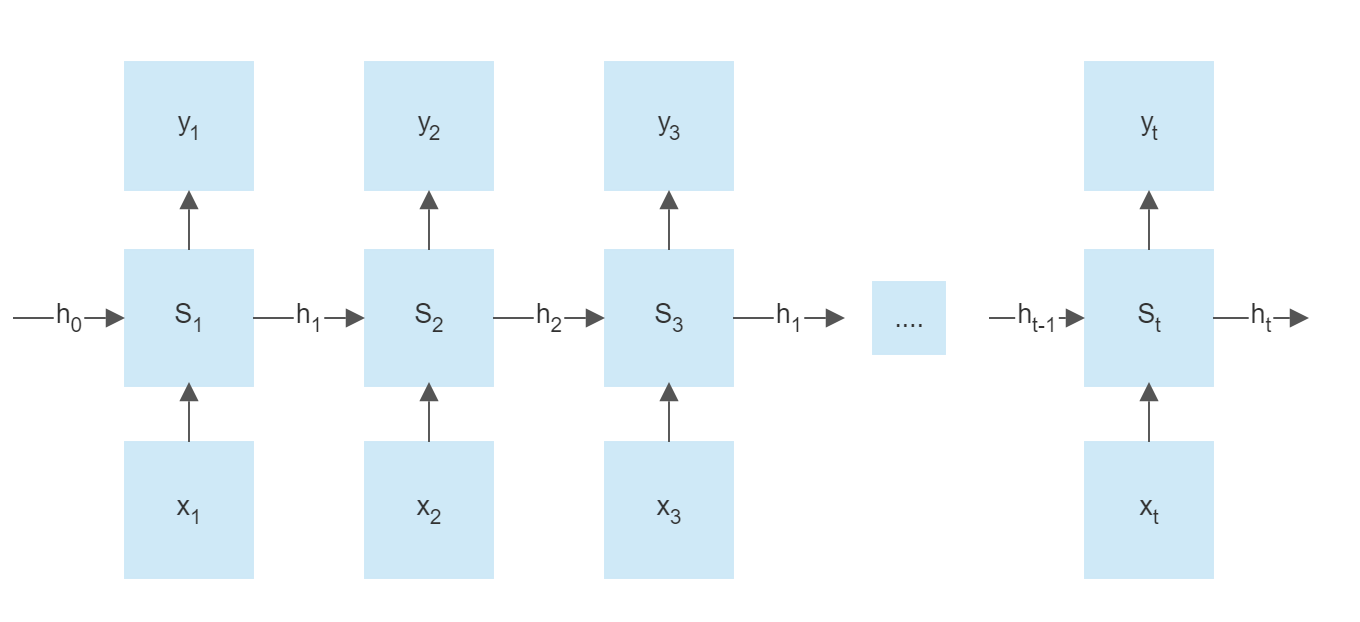
We need a neural network that can do sequential processing. How to do sequential processing, first, we need to input the values sequentially. Below is demonstrated RNN. Each RNN block (St) will take information from the previous blocks (ht-1), and the current input (xt), then create a new output (yt). The problem becomes more manageable when we can control the history of previous blocks. That reason is why the sequence model can handle processing time series data.

Figure 1: RNN architecture

Below is demonstrated memory cell of a block. With

xt: data input

yt: data output

ht: the next hidden state

ht-1: the old hidden state

Wh, Wy, Wx: weight of each output.

The hidden state is the network's memory, which includes information of the previously hidden state ht-1 plus with current data input xt. The activation function is often non-linear, like Tanh, ReLu, or Sigmoid. In this model, we use Tanh. Then we can have output yt and continue transfer to the next state with ht. The variable of the hidden unit is the number of “neurons” in the hidden state.

Diagram

Description automatically generated

Figure 2: Memory Cell

**Algorithm:**

1. Forward Propagation

Graphical user interface, text, application

Description automatically generated

**Figure 3:** Forward Propagation

The figure above shows forward pass values involving a loop over each step in the sample (xt) to get the output yt and calculate MSE loss. Use error to update the next state.

1. Backward Propagation (main algorithm)

A picture containing text

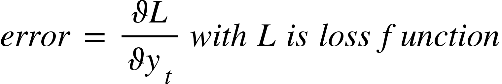
Description automatically generated

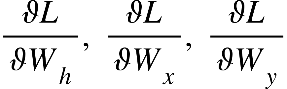
**Figure 4:** Backward Propagation

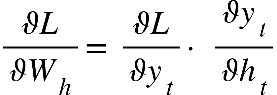
The figure above shows backward propagation, which handle the main algorithm of the program.

We can have hidden state equation h subscript t equals space tan h open parentheses W subscript x times x subscript t space plus space W subscript h times h subscript t minus 1 end subscript close parentheses

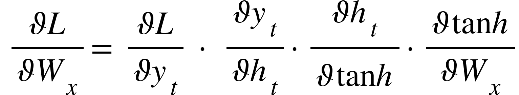
Let consider:

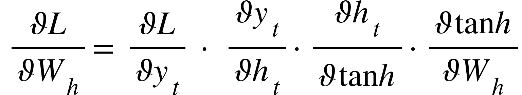


We need to know  to figure out Wh, Wx , Wy.

First we can calculate: 

For each sample in the data, we have to learn from every data for each epoch. Now we have:





Now we have enough information to compute Wh, Wx , Wy. we can choose learning rate to help the program can decrease error.

W subscript y space equals space W subscript y space minus space L e a r n i n g R a t e space times space fraction numerator ϑ L over denominator ϑ W subscript y end fraction
W subscript x space equals space W subscript x space minus space L e a r n i n g R a t e space times space fraction numerator ϑ L over denominator ϑ W subscript x end fraction
W subscript h space equals space W subscript h space minus space L e a r n i n g R a t e space times space fraction numerator ϑ L over denominator ϑ W subscript h end fraction

From that, we can calculate weights and help RNN improve the result by managing the history of the last stage to help the next stage improve for a better result.