Deep Learning

Recurrent Neural Network

By
Subhasis Dasgupta
Asst Professor
Praxis Business School, Kolkata

Why Deep Learning

In many situations traditional learning techniques were found inadequate such as

- Speech recognition
- Image recognition
- Sequence modeling

Common problems:

- Correlated features
- Correlated samples
- Variable features

So far only ANN could show significant amount of performance boost in handling such situation but with different architecture

What is special in RNN

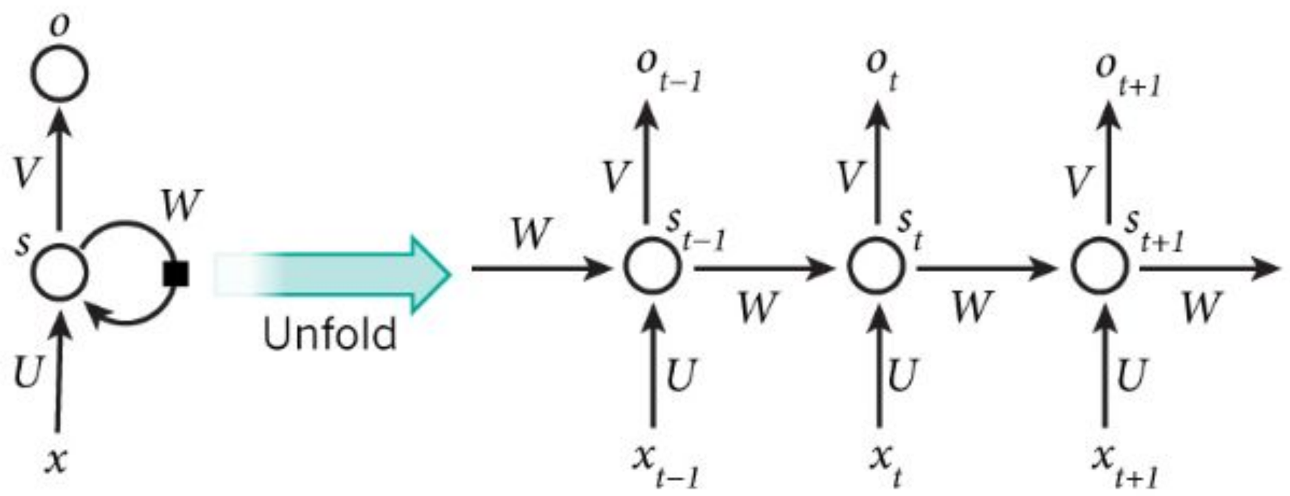
An RNN structure has some sort of memory associated with it which is helpful in sequence prediction

Has at least one feedback connection which helps in looping the activations

May have stochastic activation functions as well

It has been used extensively in NLP and other sequence modeling

Architecture of RNN



$$s_t = \sigma_s[Ux_t + Ws_{t-1} + b_s]$$
$$o_t = \sigma_o[Vs_t + b_o]$$

RNN training

RNN can be trained using the concepts of back-propagation of errors and applying gradient descent technique

$$s_t = \sigma_s[Ux_t + Ws_{t-1} + b_s]$$
$$o_t = \sigma_o[Vs_t + b_o]$$

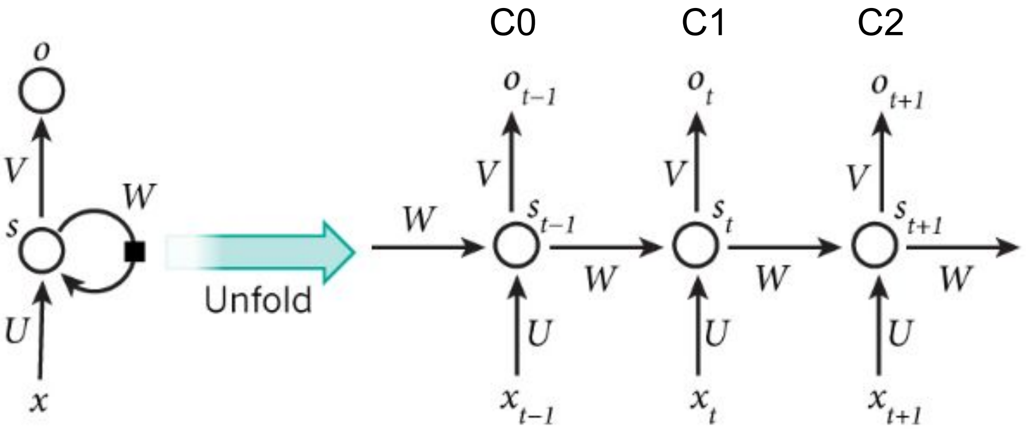
$$\frac{\partial C}{\partial W} = \sum_t \frac{\partial C_t}{\partial W}$$

Gradient Example

$$\frac{\partial C_{t+1}}{\partial W} = \frac{\partial C_{t+1}}{\partial O_{t+1}} \cdot \frac{\partial O_{t+1}}{\partial s_{t+1}} \cdot \frac{\partial s_{t+1}}{\partial a} \cdot \frac{\partial a}{\partial W}$$

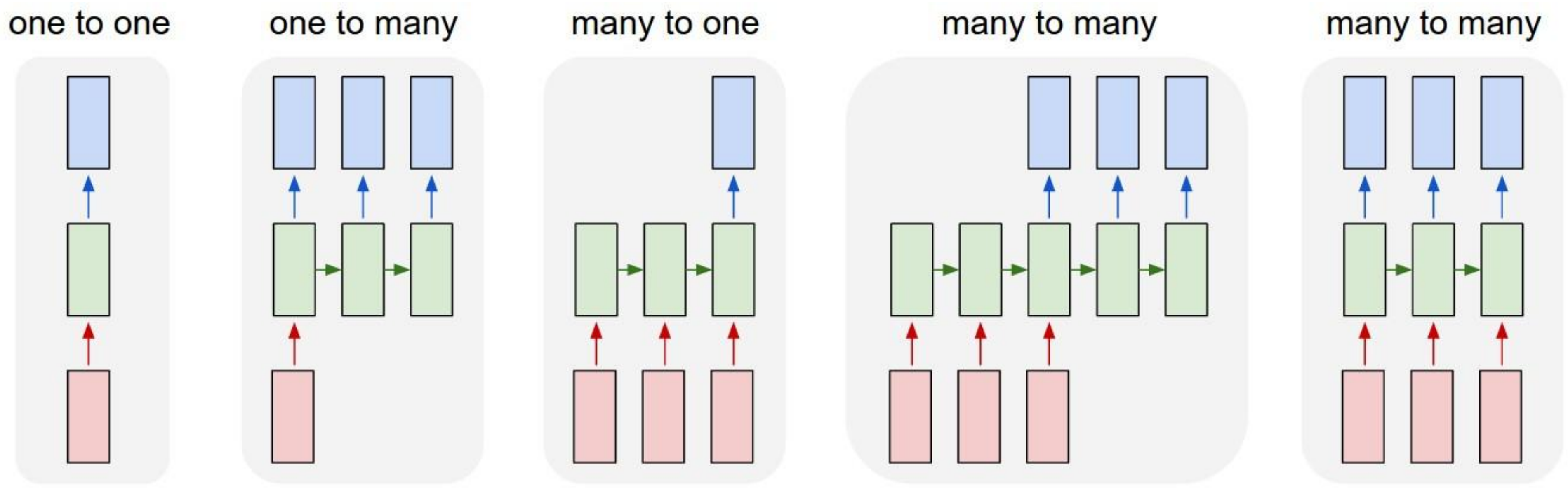
$$a = [Ux_t + Ws_t + b_s]$$

Depends on W as well



Types of RNN

Depending on the objectives in hand, RNN can be used in different ways



Typical Usage of RNN

RNN have shown high level of predictive accuracy in text analysis such as

- Sentiment mapping
- Character predicting in texts
- Translation of languages
- Image processing

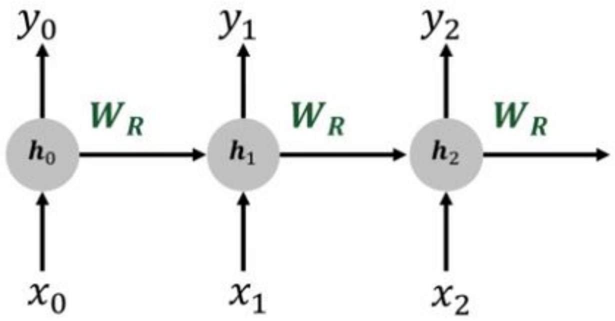
Problem in RNN training

Common problems in RNN training are

- Diminishing gradient
- Exploding gradient

Suppose there are 100 such sequences and the gradient of 100th node is required to be calculated.

If the gradient of each time step is above 1, then the final gradient would be very large and if the gradient at time steps are less than 1 then the value would be close to zero



$$\frac{\partial C_{100}}{\partial W_R} = \frac{\partial C_{100}}{\partial y_{100}} \ldots W_R \frac{\partial g_{100}}{\partial a_{100}} \ldots W_R \frac{\partial g_{99}}{\partial a_{99}} \ldots$$

$$\frac{\partial C_T}{\partial W_R} \propto |W_R|^T \left|\frac{\partial g}{\partial a}\right|^T$$

Problem in RNN training

Exploding Gradient

- Easier to detect
- Truncated backpropagation through time (BPTT)
- Clip gradients at thresholds
- RMSprop for adjusting learning rate

Vanishing Gradient

- Hard to detect
- ReLu (Rectified Linear Units) activation function which has no zero gradient
- RMSprop for adjusting learning rate
- LSTM and GRUs

LSTM (Long Short Term Memory)

Proposed by Sepp Hochreiter and Jurgen Schmidhuber in 1997

LSTM has a compelling feature of not getting stuck by vanishing gradient problem

It has four layers of neural net along with gates to control the state of the cell

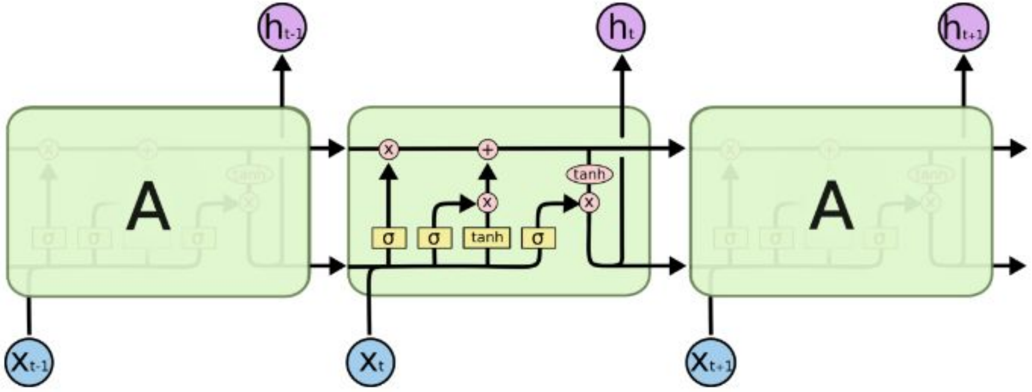
The additional layers have both sigmoid and tanh activation functions to control the inputs

Understanding LSTM

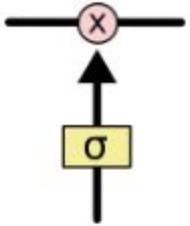
LSTM is capable of removing or adding information to the cell state

Gates are important addition to LSTM

Gates are composed on sigmoid layer neural net and pointwise multiplication operation



The repeating module in an LSTM contains four interacting layers.

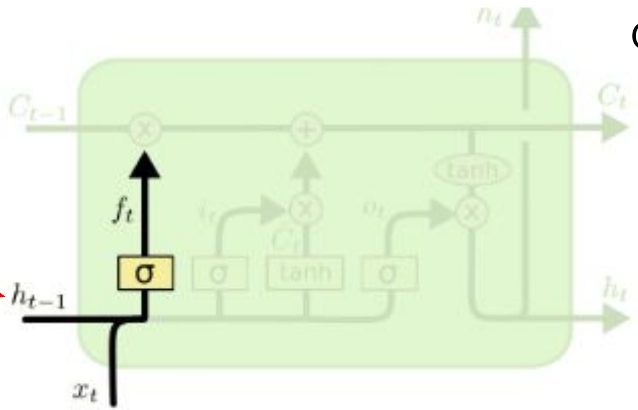


Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Understanding LSTM

Step 1: Decide what is to be forgotten (forget gate layer comes into play)

How?



Output of past cell state

Input for current cell state

Output in [0,1]

Weights of the neural network

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$

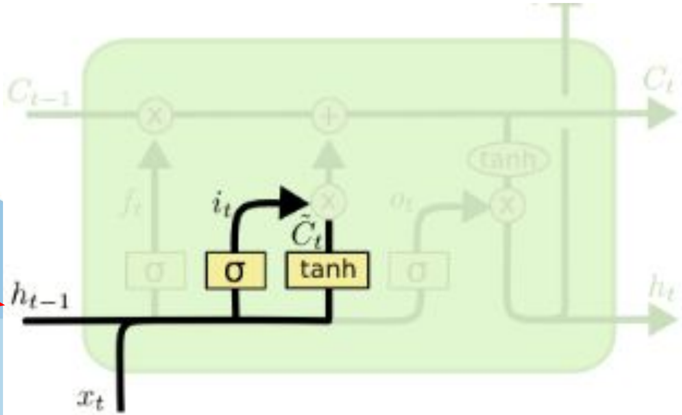
Sigmoid function for deciding the level of forgetfulness of past state

Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Understanding LSTM

Step 2: Decide what new information we're going to store in the cell state

How?



Output of past cell state

Input for current cell state

Sigmoid function for deciding the level of information store

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

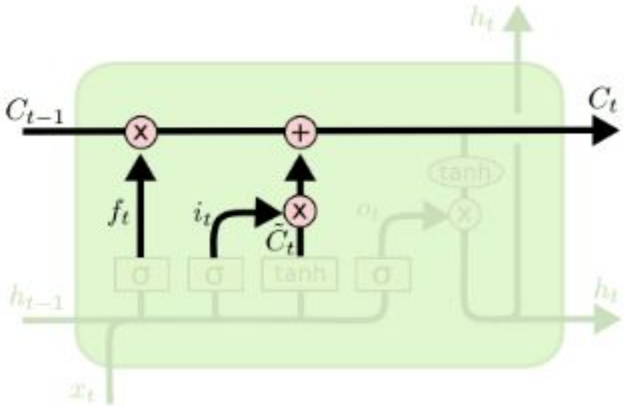
Tanh function for candidate sets

Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Understanding LSTM

Step 3: Update old state $C_{(t-1)}$ to its new state i.e. C_t

How?



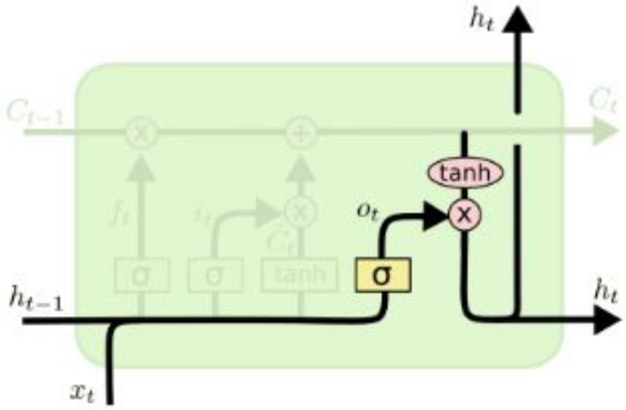
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Understanding LSTM

Step 4: Decide Output

- run a sigmoid layer which decides what parts of the cell state we're going to output
- put the cell state through tanh (to push the values to be between −1 and 1) and multiply it by the output of the sigmoid gate, so that we only output the parts we decided to



Part of current state to be output

$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Output of state C_t

GRU (Gated Recurrent Unit)

Proposed by Kyunghyun Cho in 2014

It can be considered as a simplified version of LSTM

It has two gates rather than three gates as in LSTM

GRU has reset gate and update gate but it doesn’t have internal memory (i.e. C_t)

GRU (Gated Recurrent Unit)