Recurrent Neural Network

Minlie Huang

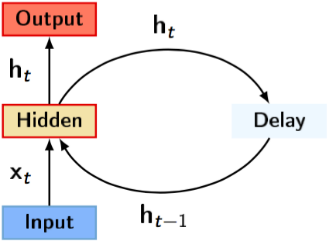
aihuang@tsinghua.edu.cn
Department of Computer Science
Tsinghua University
Some slides are from Dr. Xipeng Qiu

Summer, 2018

Recurrent Neural Network (RNN)

The RNN has recurrent hidden states whose output at each time is dependent on that of the previous time.
Given a sequence $x(1:n) = (x_1, x_2, ..., x_t, ..., x_n)$ (each x_i is a vector), the RNN updates its recurrent hidden state h(t) by

$$h_t = \begin{cases} 0 & \text{if t=0} \\ f(h_{t-1}, x_t) & \text{otherwise} \end{cases} \tag{1}$$



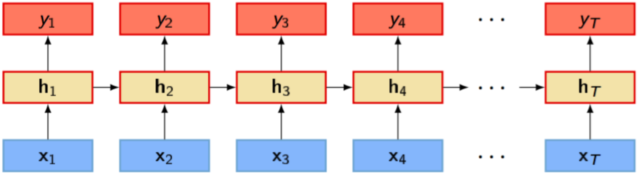
Simple Recurrent Network

- The recurrence function:

$$h_t = f(Uh_{t-1} + Wx_t + b) \tag{2}$$

where f is non-linear function (for instance, *sigmoid* or *tanh*).

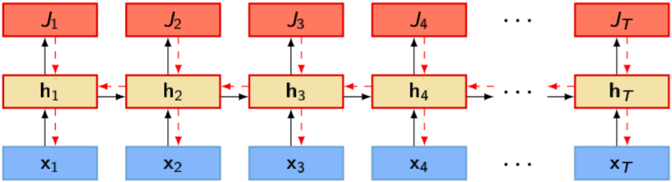
- A typical RNN structure can be illustrated as follows:



Backpropagation Through Time, BPTT

Suppose loss at time t is J_t, then the total loss is $\Sigma_{t=1}^{T} J_t$. The gradient of J is

$$\frac{\partial J}{\partial U} = \sum_{t=1}^{T} \frac{\partial J_t}{\partial U} = \sum_{t=1}^{T} \frac{\partial h_t}{\partial U} \frac{\partial J_t}{\partial h_t} \tag{3}$$



Matrix Calculus

$$\frac{\partial y}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y}{\partial x_1} & \frac{\partial y}{\partial x_2} & \cdots & \frac{\partial y}{\partial x_n} \end{bmatrix}$$

Figure: $y = f(x_1, x_2, ..., x_n)$

$$\frac{\partial \mathbf{Y}}{\partial x} = \begin{bmatrix} \frac{\partial y_{11}}{\partial x} & \frac{\partial y_{12}}{\partial x} & \cdots & \frac{\partial y_{1n}}{\partial x} \\ \frac{\partial y_{21}}{\partial x} & \frac{\partial y_{22}}{\partial x} & \cdots & \frac{\partial y_{2n}}{\partial x} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_{m1}}{\partial x} & \frac{\partial y_{m2}}{\partial x} & \cdots & \frac{\partial y_{mn}}{\partial x} \end{bmatrix}$$

Figure: Y is a matrix.

Matrix Calculus (cont'd)

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_m}{\partial x_1} & \frac{\partial y_m}{\partial x_2} & \cdots & \frac{\partial y_m}{\partial x_n} \end{bmatrix}$$

Figure: Y, X are column vectors.

$$\frac{\partial \mathbf{F}}{\partial \mathbf{X}} = \begin{bmatrix} \frac{\partial \mathbf{F}}{\partial X_{1,1}} & \cdots & \frac{\partial \mathbf{F}}{\partial X_{1,n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial \mathbf{F}}{\partial X_{m,1}} & \cdots & \frac{\partial \mathbf{F}}{\partial X_{m,n}} \end{bmatrix}$$

Figure: F, X are matrices.

Gradient of RNN

$$\frac{\partial J}{\partial U} = \sum_{t=1}^{T} \sum_{k=1}^{t} \frac{\partial h_k}{\partial U} \frac{\partial h_t}{\partial h_k} \frac{\partial y_t}{\partial h_t} \frac{\partial J_t}{\partial y_t} \tag{4}$$

$$\frac{\partial h_t}{\partial h_k} = \prod_{i=k+1}^{t} \frac{\partial h_i}{\partial h_{i-1}} = \prod_{i=k+1}^{t} U^T diag[f'(h_{i-1})] \tag{5}$$

$$\frac{\partial J}{\partial U} = \sum_{t=1}^{T} \sum_{k=1}^{t} \frac{\partial h_k}{\partial U} \left(\prod_{i=k+1}^{t} U^T diag[f'(h_{i-1})] \right) \frac{\partial y_t}{\partial h_t} \frac{\partial J_t}{\partial y_t} \tag{6}$$

Long-Term Dependencies

Define $\gamma = ||U^T diag[f'(h_{i-1})]||$,

- Exploding Gradient Problem: When $\gamma > 1$ and $t - k \to \infty$, $\gamma^{t-k} \to \infty$.
- Vanishing Gradient Problem: When $\gamma < 1$ and $t - k \to \infty$, $\gamma^{t-k} \to 0$.

There are various ways to solve Long-Term Dependency problem.

Gradient Explosion & Vanishing

- Gradient norm clipping (Mikolov thesis 2012; Pascanu, Mikolov, Bengio, ICML 2013)

$$\hat{g} \leftarrow \frac{\partial \mathcal{J}}{\partial \theta}$$

IF

$$||\hat{g}|| > \mathit{Threshold}$$

THEN

$$\hat{g} \leftarrow \frac{\mathit{Threshold}}{||\hat{g}||} * \hat{g}$$

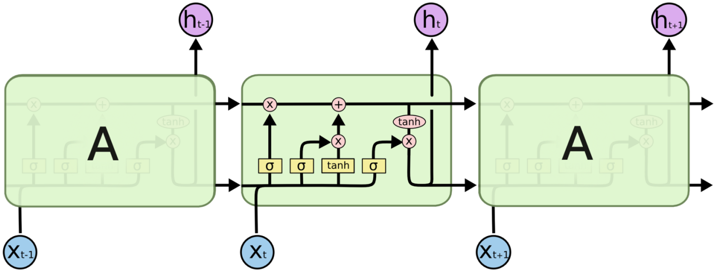
- Gradient propagation regularizer (avoid vanishing gradient)
- LSTM self-loops (avoid vanishing gradient)

Long Short-Term Memory (LSTM)

Key difference to RNN: a memory cell c which is controlled by three gates:

- input gate i:
$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{7}$$
- output gate o:
$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{8}$$
- forget gate f:
$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{9}$$
- update of memory cell and hidden state:
$$\widetilde{C_t} = tanh(W_c x_t + U_c h_{t-1} + b_C) \tag{10}$$
$$C_t = f_t \otimes C_{t-1} + i_t \otimes \widetilde{C_t} \tag{11}$$
$$h_t = o_t \otimes tanh(C_t) \tag{12}$$

LSTM Architecture



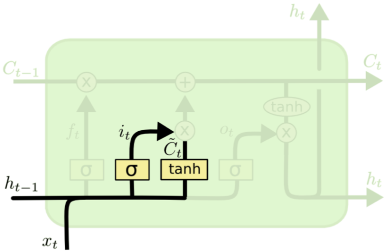
$$\widetilde{C_t} = tanh(W_c x_t + U_c h_{t-1} + b_C)$$

$$C_t = f_t \otimes C_{t-1} + i_t \otimes \widetilde{C_t}$$

$$h_t = o_t \otimes tanh(C_t)$$

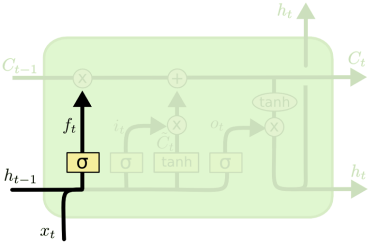
The picture and following 4 pictures are from http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

LSTM: Input and Forget Gates



$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$$

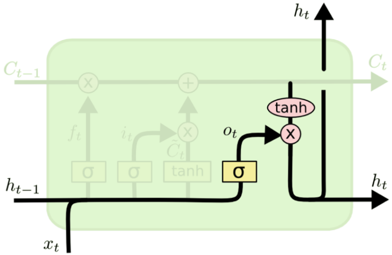
$$\widetilde{C}_t = tanh(W_c x_t + U_c h_{t-1} + b_C)$$



$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$$

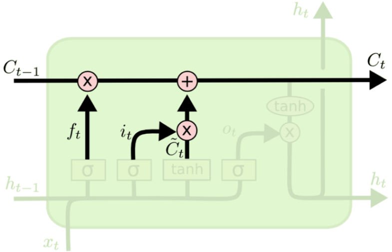
The picture is from http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

LSTM:Output Gates and State Update



$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)$$

$$h_t = o_t \otimes tanh(C_t)$$



$$C_t = f_t \otimes C_{t-1} + i_t \otimes \widetilde{C_t}$$

The picture is from
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Gated Recurrent Unit, GRU

Instead of using three gates, only two gates is employed:

- update gate z:

$$z_t = \sigma(W_z x_t + U_z h_{t-1}) \tag{13}$$

- reset gate r:

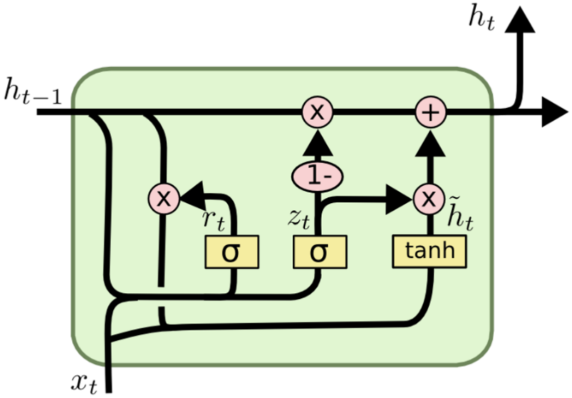
$$r_t = \sigma(W_r x_t + U_r h_{t-1}) \tag{14}$$

- state update:

$$\widetilde{h_t} = tanh(W_c x_t + U(r_t \otimes h_{t-1})) \tag{15}$$

$$h_t = (1 - z_t) \otimes h_{t-1} + z_t \otimes \widetilde{h_t} \tag{16}$$

GRU Architecture



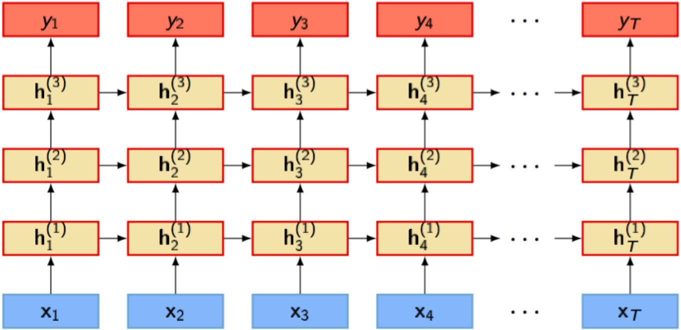
The picture is from
http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

More Exploring to LSTM...

- How does LSTM deal with the explosion and vanishing gradient problems?
- refer to: Hochreiter S, Schmidhuber J. Long short-term memory[J]. Neural computation, 1997, 9(8): 1735-1780.

Stacked RNN/LSTM

- Stack multiple RNN/LSTM in vertical direction



- More common is seen that raw input (x_i) is connected to all layers as input.

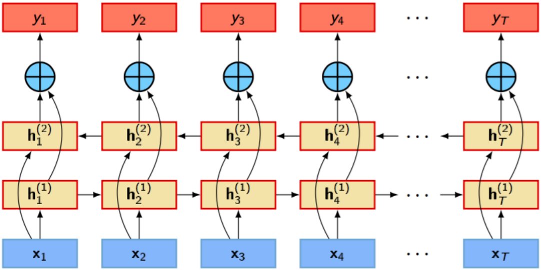
Bidirectional RNN/LSTM

- Modeling the prefix context and suffix context at the same time.

$$h_t^{(1)} = f(U^{(1)} h_{t-1}^{(1)} + W^{(1)} x_t + b^{(1)}) \tag{17}$$

$$h_t^{(2)} = f(U^{(2)} h_{t+1}^{(2)} + W^{(2)} x_t + b^{(2)}) \tag{18}$$

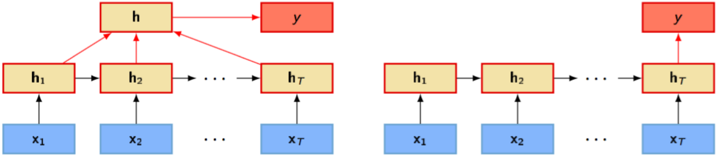
$$h_t = h_t^{(1)} \oplus h_t^{(2)} \tag{19}$$



Application of RNN: Classification

Applicable to many classification problems.

- Text Classification
- Sentiment Classification

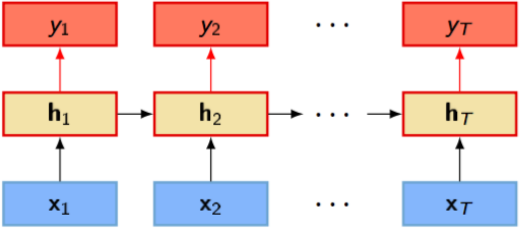


(c) Mean (d) Last

The left: can be enhanced with attention mechanisms.

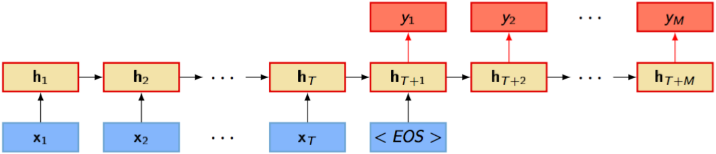
Application of RNN: Sequence Labeling

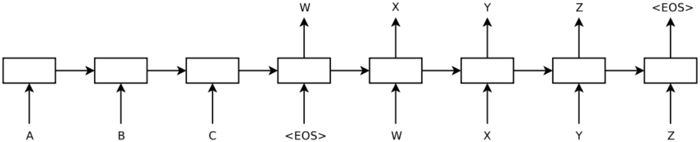
- Sequence Labeling, such as Chinese word segmentation, part-of-speech tagging, semantic role labeling



Application of RNN: Sequence to Sequence Learning

- Machine Translation
- Sequence to sequence generation (for dialogue, question answering, etc.)
- Image captioning: from image to text

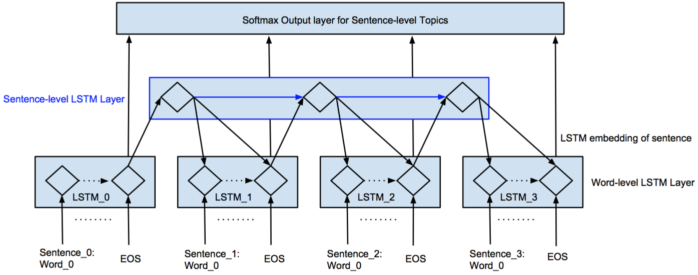




Sutskever I, Vinyals O, Le Q V. Sequence to sequence learning with neural networks[C]//NIPS. 2014: 3104-3112.

Hierarchical LSTM

Hierarchical LSTM (low-level LSTMs are taken as input to the high-level LSTM)



Ghosh S, Vinyals O, Strope B, et al. Contextual LSTM (CLSTM) models for Large scale NLP tasks[J]. arXiv preprint arXiv:1602.06291, 2016.

Summary

- RNN is very good at modeling text/language
- Basic RNN models: RNN, GRU, LSTM
  - Stacked RNN
  - Bi-directional RNN
  - Hierarchical RNN
- Typical Learning Problems with RNN
  - Classification
  - Sequential Labeling
  - Sequence to Sequence translation: MT, Dialogue Generation