IFI 9000 Analytics Methods
Neural Networks & Deep Learning

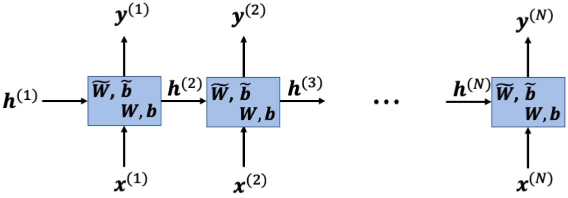
by **Houping Xiao**

Spring 2021

Introduction

Recurrent neural networks

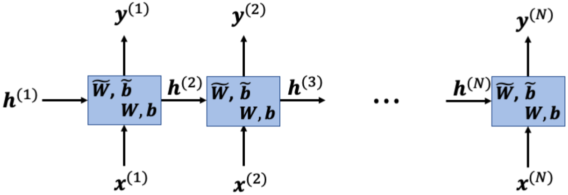
- While CNNs work quite promising for images, they may not be the best modeling tools for other data sets such as time series data
- For temporal, or time-series data and stream inputs (e.g., text streams), recurrent neural networks (RNNs) are of major attention



- We assume a sequence of data is streamed as N time instances, and mapped to a sequence of response (here of the same length).
- For now let's assume that the input and output have similar lengths

RNN: governing equations

- Remember in standard neural network the output of the hidden layer was in the form $\boldsymbol{h} = \sigma(\boldsymbol{Wx}x + \boldsymbol{b})$



- In RNNs the input is a stream x(t) and we have another coefficient matrix that makes the current hidden output dependent on the previous one:

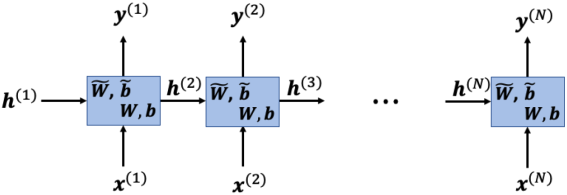
$$\boldsymbol{h}^{(t)} = \sigma\left(\boldsymbol{W}\begin{pmatrix}\boldsymbol{h}^{(t-1)}\\ \boldsymbol{x}^{(t-1)}\end{pmatrix} + \boldsymbol{b}\right),$$

$$\boldsymbol{y}^{(t)} = \sigma\left(\tilde{\boldsymbol{W}}\boldsymbol{h}^{(t)} + \tilde{\boldsymbol{b}}\right), t = 1, \cdots, N$$

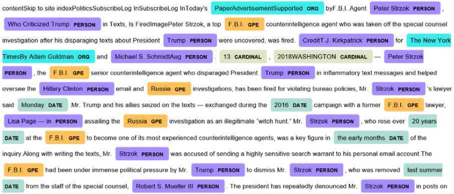
- Training cost per sample: $\mathcal{L}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \sum_{t=1}^{N} L\left(\boldsymbol{y}^{(t)}, \hat{\boldsymbol{y}}^{(t)}\right)$

Types of RNN and applications

- The following architecture is many-to-many, with the input and output having the same length

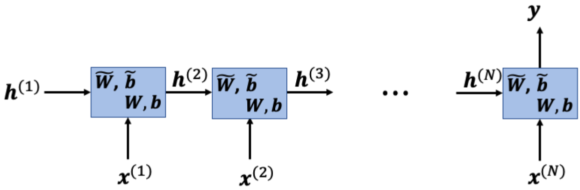


- Application example is named-entity recognition (classify unstructured text into predefined classes)



Types of RNN and applications

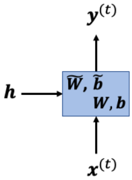
- The following architecture is many-to-one



- Application example is sentiment classification (review systems, scoring systems)

Types of RNN and applications

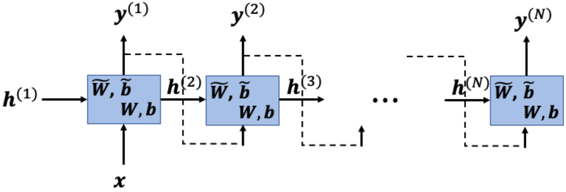
- The following architecture is one-to-one



- This is somehow equivalent to traditional one-layer network (real-time mapping)

Types of RNN and applications

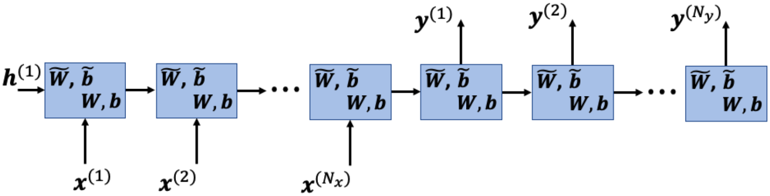
- The following architecture is one-to-many



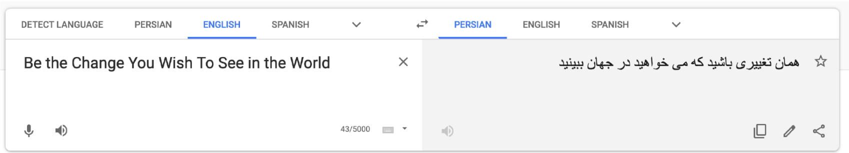
- Application example is music generation, image captioning

Types of RNN and applications

- The following architecture is many-to-many, with the input and output having different lengths

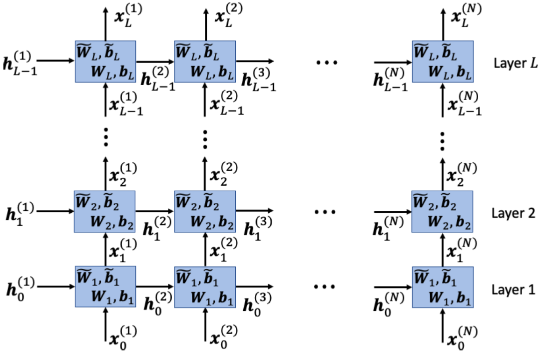


- Application example is machine translation, video captioning



Deep RNNs

- All the architectures we explained so far can become deep and layered



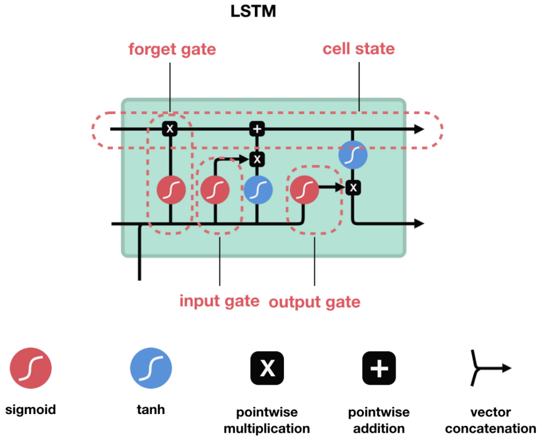
- In practice we do not need very deep RNNs (unlike standard DNNs which can be very deep)

Problems with standard RNNs and remedies

- Hard to train and vanishing gradient
- Difficulty accessing information from long time ago
- Two main variants of RNNs:
  - Long Short-Term Memory (LSTM)
  - Gated Recurrent Units (GRUs)
- To learn more and see some cool applications see: https://www.youtube.com/watch?v=6niqTuYFZLQt=1850s

LSTM: Long Short-term Memory

- LSTM developed for resolving short-term memory, i.e., RNN can forget what it seen in longer sequences
- LSTM uses “gates” to regulate the flow of information, keep important information or throw away unimportant information



LSTM: a toy example

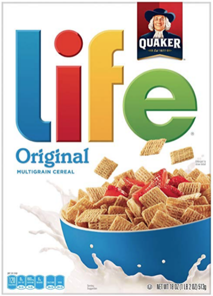
Customers Review 2,491

Thanos

September 2018

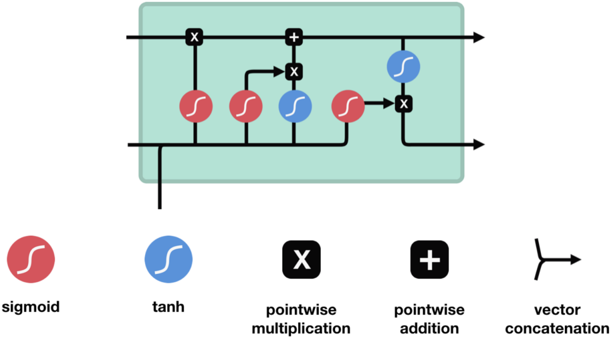
Verified Purchase

Amazing! This box of cereal gave me a perfectly balanced breakfast, as all things should be. I only ate half of it but will definitely be buying again!



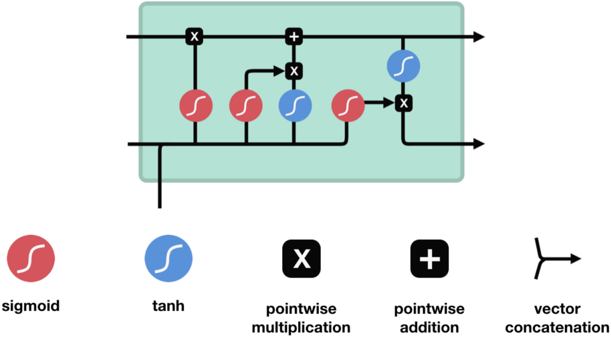
A Box of Cereal
$3.99

- Keep or forget information using gates
- Gates are different neural networks, contains sigmoid functions
- Sigmoid squishes values between 0 (forgotten) and 1 (kept).



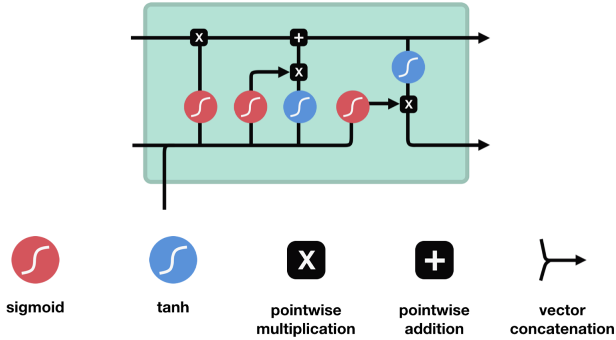
LSTM: forget gate

- Based on the previous hidden state and current input, the forget gate decides what information should be forgotten or kept
- output value closer to 0 means to forget, and the closer to 1 means keep



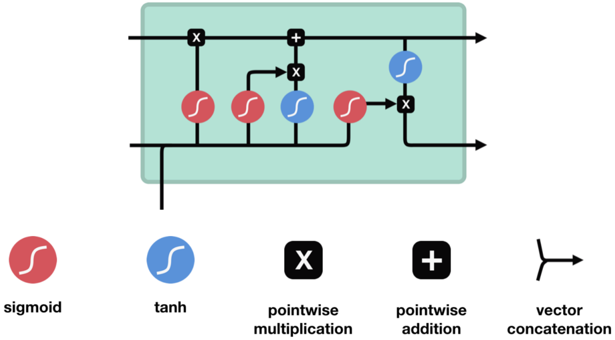
LSTM: input gate

- Based on the previous hidden state and current input, the input gate uses sigmoid to decide what information should be updated
- uses tanh function to regulate the network
- the sigmoid output will decide which information is important to keep from the tanh output



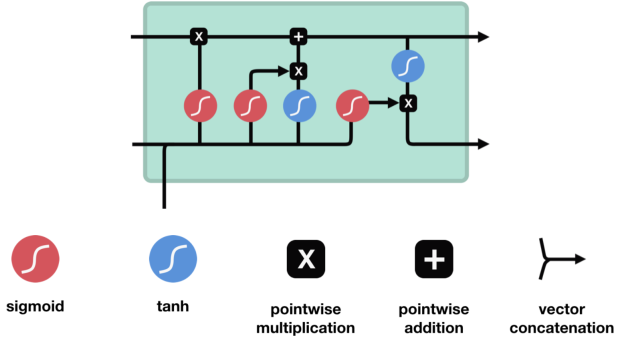
LSTM: cell state

- Based on the previous cell state, output from the input gate, to update the new cell state.



LSTM: output gate

- Output the new cell state
- output the hidden state, based on the previous hidden state and the current input



The End