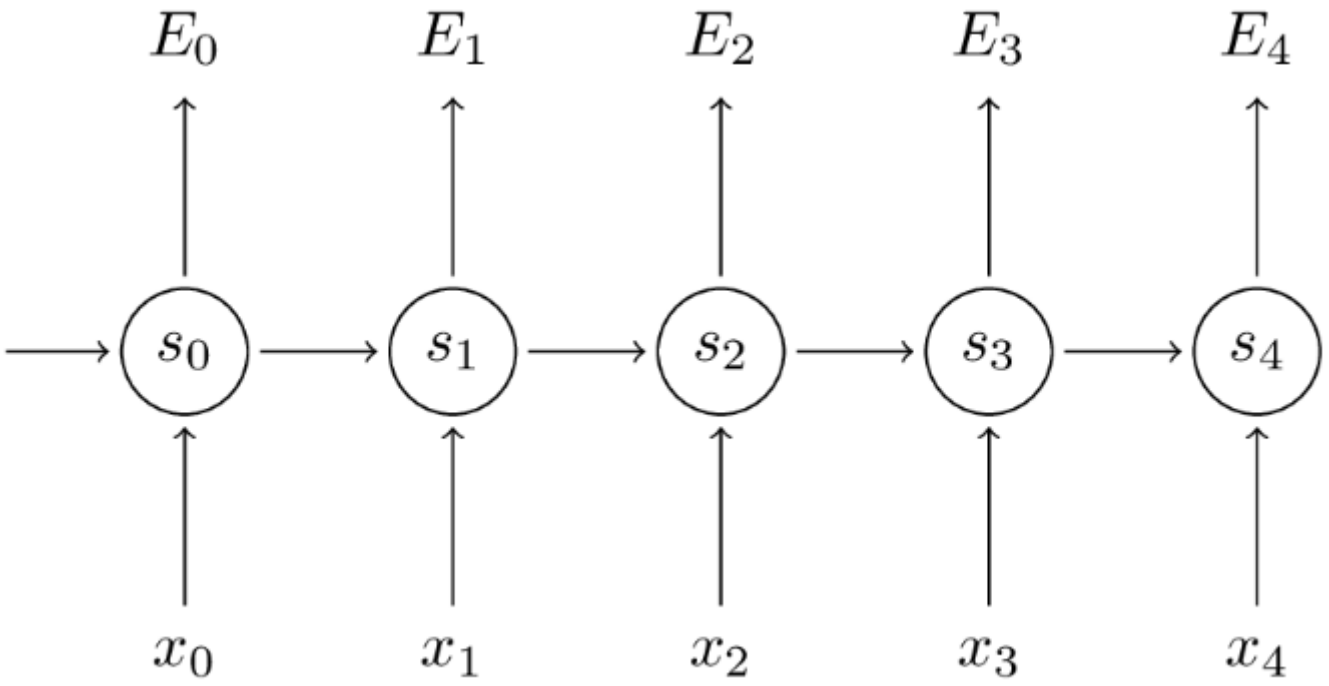
Vanishing Gradients and Fancy RNNs

Wongyu Kim (김원규)

Internet Computing Laboratory
Department of Computer Science
Yonsei University

2020.08.28

Vanishing Gradient Problem



$$\frac{\partial E}{\partial W} = \sum_t \frac{\partial E_t}{\partial W}.$$

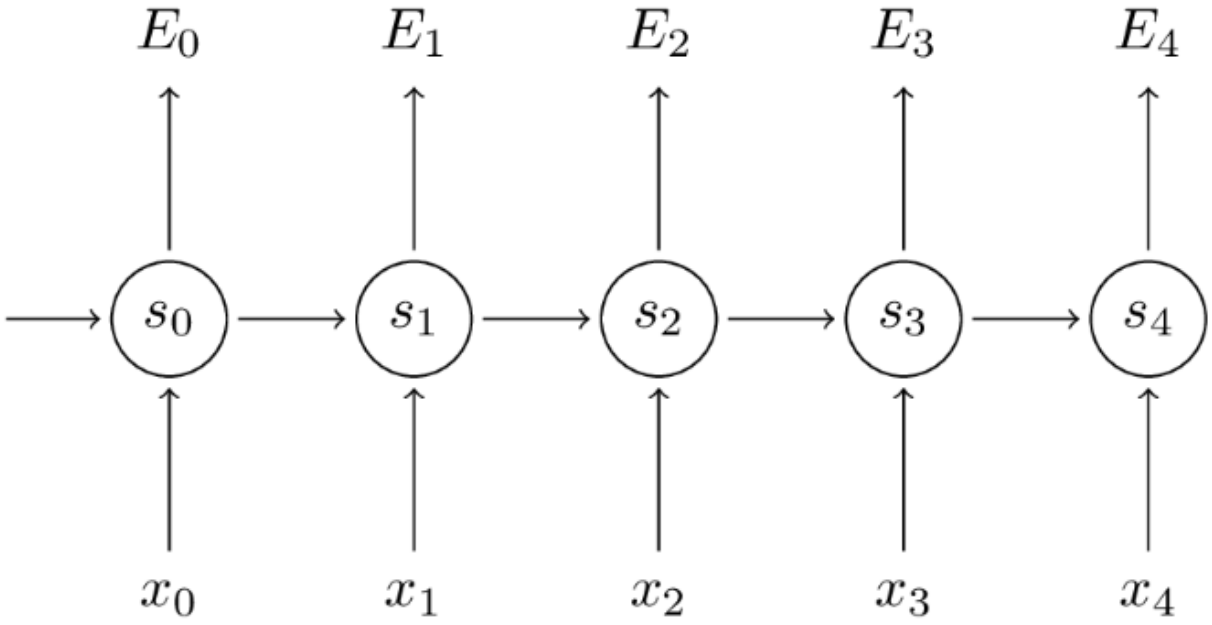
핵심 Equation (1)

$$\frac{\partial E_3}{\partial W} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \frac{\partial s_3}{\partial s_k} \frac{\partial s_k}{\partial W}$$

핵심 Equation (2)

VGP의 원인이 되는 식
Long Range Dependency 활용을 못하게 막는 가장 큰 원인

Vanishing Gradient Problem



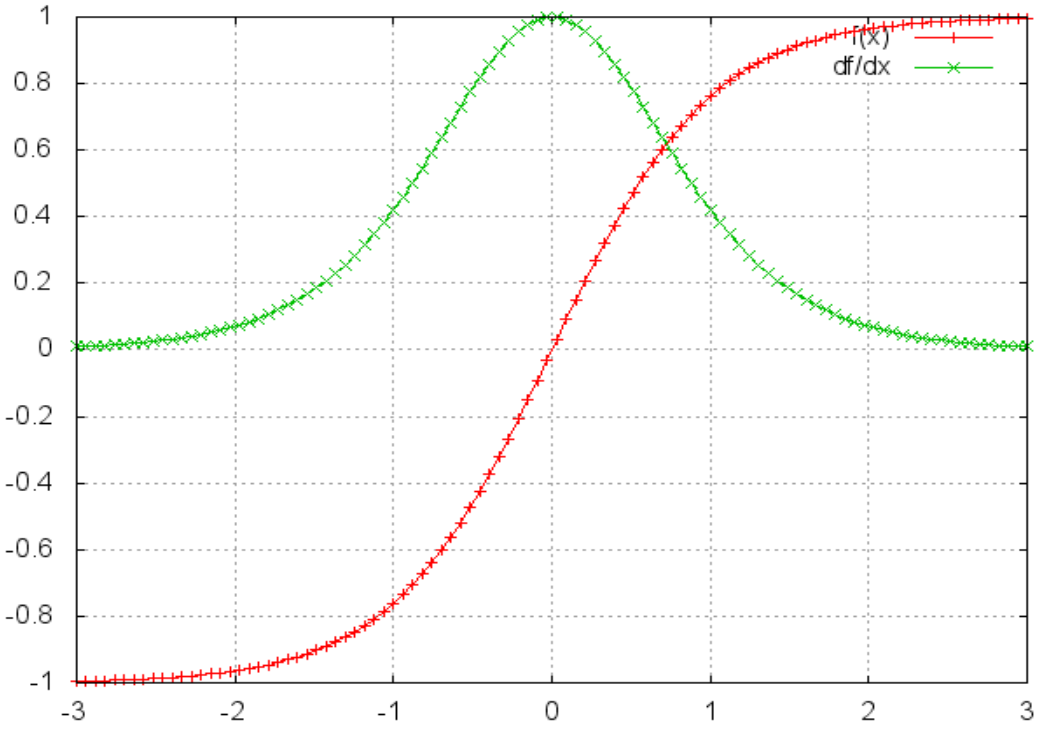
(2-1) $$\frac{\partial E_3}{\partial W} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \frac{\partial s_3}{\partial s_k} \frac{\partial s_k}{\partial W} \Longrightarrow \frac{\partial E_3}{\partial W} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \left(\prod_{j=k+1}^{3} \frac{\partial s_j}{\partial s_{j-1}} \right) \frac{\partial s_k}{\partial W}$$

(2-2) $$\frac{\partial \boldsymbol{s}_j}{\partial \boldsymbol{s}_{j-1}} = \frac{\partial \boldsymbol{h}^{(t)}}{\partial \boldsymbol{h}^{(t-1)}} = \mathrm{diag}\left(\sigma'\left(\boldsymbol{W}_h \boldsymbol{h}^{(t-1)} + \boldsymbol{W}_x \boldsymbol{x}^{(t)} + \boldsymbol{b}_1\right)\right) \boldsymbol{W}_h$$

(2-3) $$\frac{\partial J^{(i)}(\theta)}{\partial \boldsymbol{h}^{(j)}} = \frac{\partial J^{(i)}(\theta)}{\partial \boldsymbol{h}^{(i)}} \prod_{j<t\leq i} \frac{\partial \boldsymbol{h}^{(t)}}{\partial \boldsymbol{h}^{(t-1)}}$$

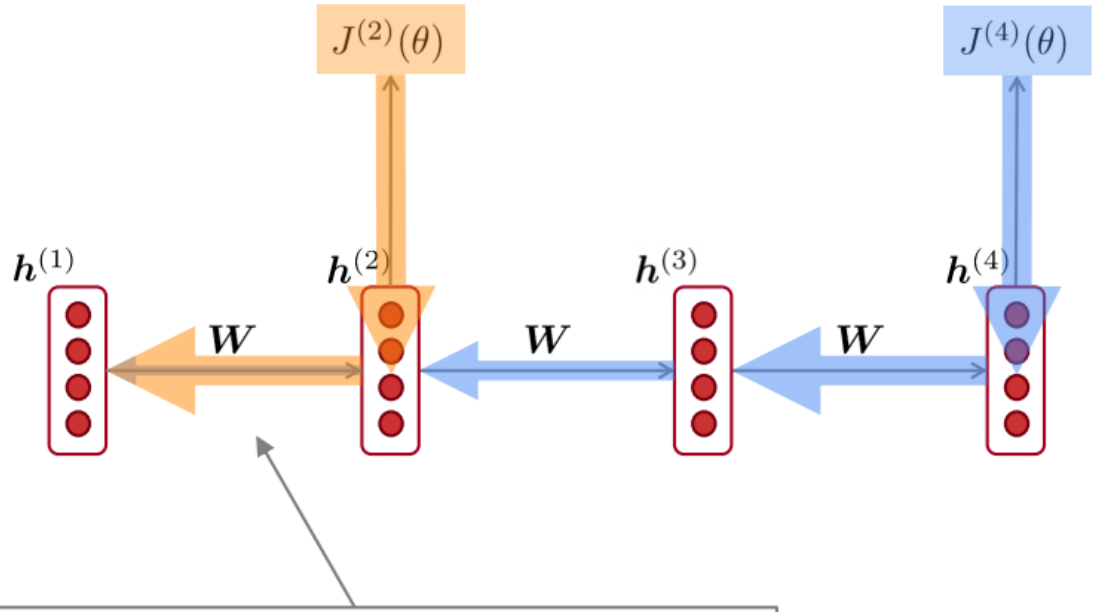
$$= \frac{\partial J^{(i)}(\theta)}{\partial \boldsymbol{h}^{(i)}} \boldsymbol{W}_h^{(i-j)} \prod_{j<t\leq i} \mathrm{diag}\left(\sigma'\left(\boldsymbol{W}_h \boldsymbol{h}^{(t-1)} + \boldsymbol{W}_x \boldsymbol{x}^{(t)} + \boldsymbol{b}_1\right)\right)$$

Vanishing Gradient Problem +



Vanishing Gradient Problem

Why is vanishing gradient a problem?



Gradient signal from faraway is lost because it's much smaller than gradient signal from close-by.

So model weights are only updated only with respect to near effects, not long-term effects.

- **LM task:** *When she tried to print her tickets, she found that the printer was out of toner. She went to the stationery store to buy more toner. It was very overpriced. After installing the toner into the printer, she finally printed her _____*

Why Gradient Vanishing

- 만약 gradient가 너무 크다면 Nan(Not a Number) or Inf로 떠서 쉽게 알아차려 어느 정도의 threshold를 주어 gradient를 clipping 할 수 있다.

Algorithm 1 Pseudo-code for norm clipping

$\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta}$
if $\|\hat{\mathbf{g}}\| \geq threshold$ **then**
  $\hat{\mathbf{g}} \leftarrow \frac{threshold}{\|\hat{\mathbf{g}}\|}\hat{\mathbf{g}}$
end if

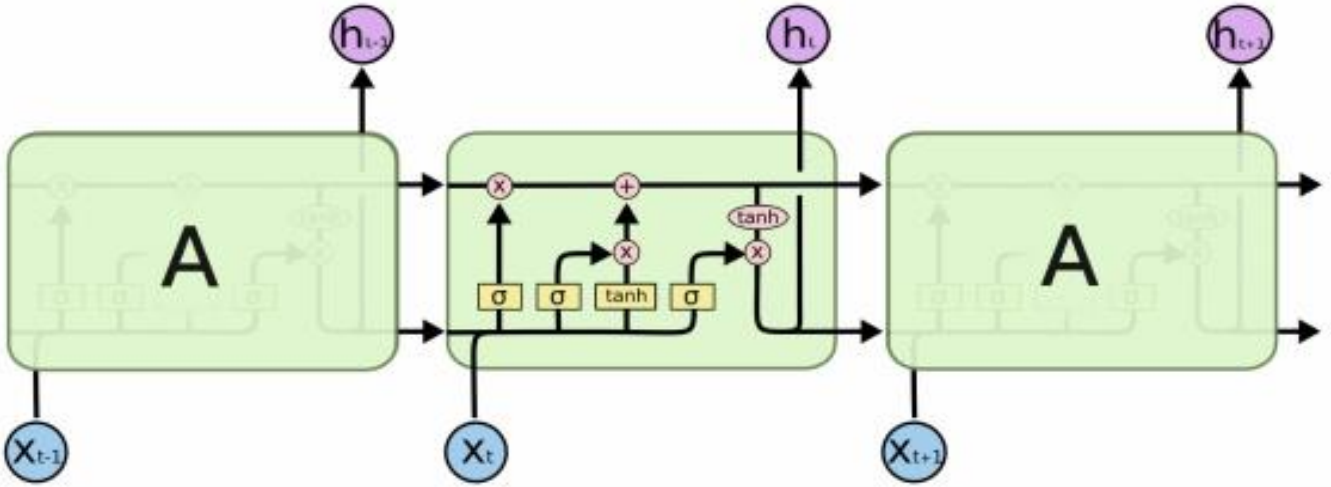
- 따라서 Vanishing 문제 해결이 더 시급하며 여러가지 해결책이 있다.
  1. W를 좋은 값으로 초기화
  2. Regularization(오버피팅 방지, W 값 저하, 테스트 일반화
  3. ReLU 사용(0 이하의 값 = gradient 0, 그렇지 않으면 gradient 1)
  4. LSTM or GRU

LSTM – Long Short-Term Memory

- Long-Range Dependency를 해결하기 위해서는 뒤의 timestep에서 앞의 정보를 이용하는 것이 어렵다는 것이다.
- 따라서 hidden state 를 계속 write하지 않고 따로 정보를 저장할 수 있는 separate memory를 만들자.
- 즉, hidden state를 바로 적용하지 말고 cell state에서 어느정도 이전 정보를 반영할지에 대해 처리한 후(=separate memory write) hidden state에 반영한다.

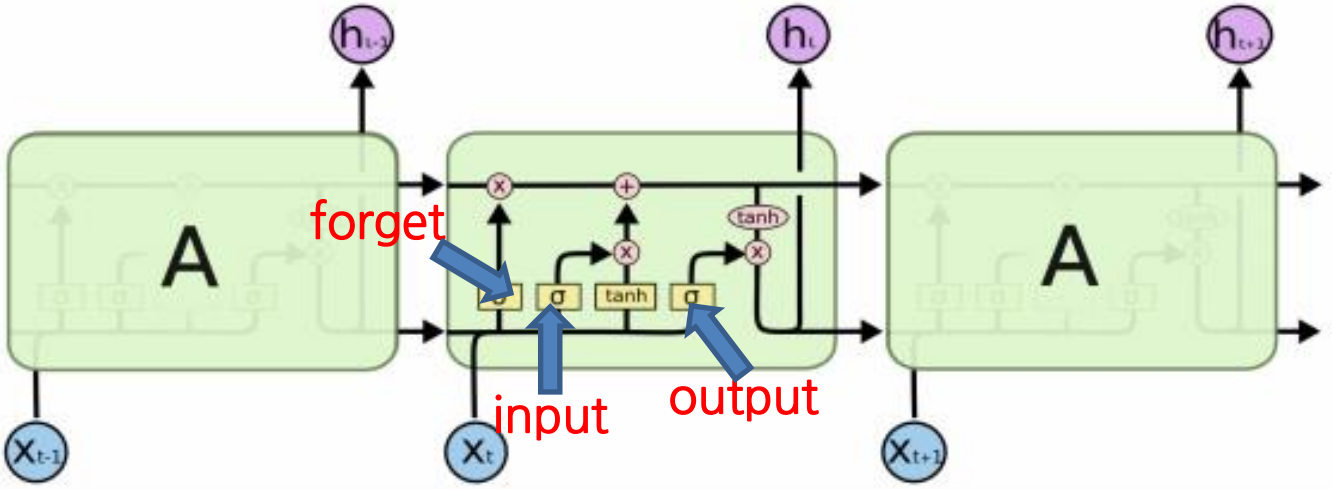
LSTM

- h(t): hidden state, c(t): cell state
  (둘 다 n-length vector / cell-state는 long-term info를 저장하며 cell에서 forget, write, read info 기능이 가능)
- Gate는 forget, input, output gate가 있으며 역시 n-length vector
  gate 성분은 0~1 사이의 값을 가짐(0 = closed, 1 = open)
- 각 state마다 gate weights는 공유되지 않는다.(RNN에서는 W를 공유하여 사용했다.)



LSTM

- f(t)는 과거 정보를 잊는 게이트(0이면 잊고, 1이면 온전히 기억)
- i(t)는 현재 정보를 기억하는 게이트(0이면 잊고, 1이면 온전히 기억)
- c(t)~는 이번에 새롭게 cell에 쓰여질 내용(현재 정보 with i(t))
- c(t-1)은 과거 정보 with f(t)
- c(t)는 최종 cellstate 정보
- h(t)는 c(t)의 정보를 o(t) gate를 통해 비율에 따라 반영한 정보(o(t) 역시 0이면 c(t)의 정보를 반영하지 않고, 1이면 온전히 반영)



LSTM

Long Short-Term Memory (LSTM)

We have a sequence of inputs $x^{(t)}$, and we will compute a sequence of hidden states $h^{(t)}$ and cell states $c^{(t)}$. On timestep t:

Forget gate: controls what is kept vs forgotten, from previous cell state

Input gate: controls what parts of the new cell content are written to cell

Output gate: controls what parts of cell are output to hidden state

New cell content: this is the new content to be written to the cell

Cell state: erase ("forget") some content from last cell state, and write ("input") some new cell content

Hidden state: read ("output") some content from the cell

Sigmoid function: all gate values are between 0 and 1

$$f^{(t)} = \sigma\left(W_f h^{(t-1)} + U_f x^{(t)} + b_f\right)$$
$$i^{(t)} = \sigma\left(W_i h^{(t-1)} + U_i x^{(t)} + b_i\right)$$
$$o^{(t)} = \sigma\left(W_o h^{(t-1)} + U_o x^{(t)} + b_o\right)$$
$$\tilde{c}^{(t)} = \tanh\left(W_c h^{(t-1)} + U_c x^{(t)} + b_c\right)$$
$$c^{(t)} = f^{(t)} \circ c^{(t-1)} + i^{(t)} \circ \tilde{c}^{(t)}$$
$$h^{(t)} = o^{(t)} \circ \tanh c^{(t)}$$

All these are vectors of same length n

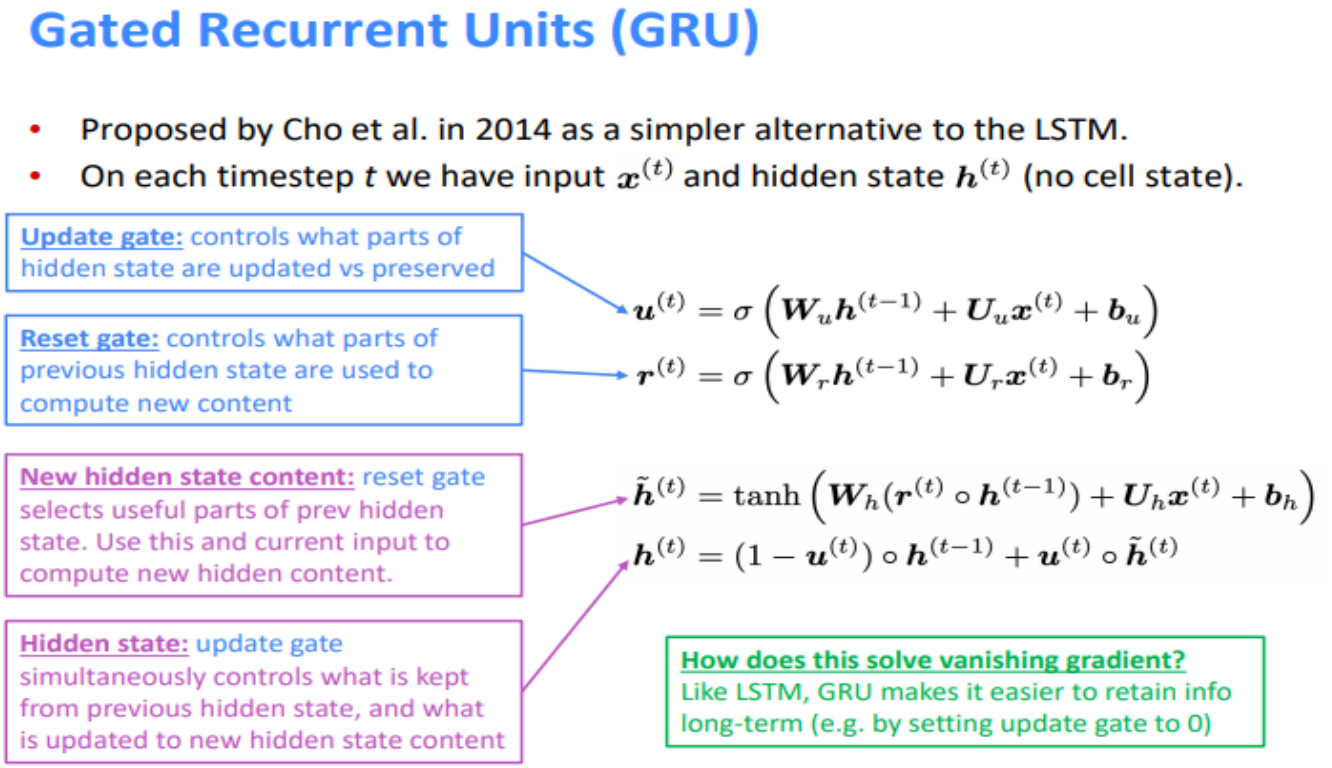
Gates are applied using element-wise product

23

여기서 Wf, Uf, bf, Wi, Ui, bi, Wo, Uo, bo, Wc, Uc, bc 의 파라미터는 RNN과 다르게 state마다 공유하지 않는다.

GRU – Gated Recurrent Units

- GRU는 LSTM의 간략화 버전
- Cell state가 없다.



- u(t): 이전 hidden state 내용이 얼마나 보존되는지
- r(t): h(t)~를 계산하는데 있어서 이전 hidden state 내용이 얼마나 보존되는지

LSTM 현 상황

LSTMs: real-world success

- In 2013-2015, LSTMs started achieving state-of-the-art results
  - Successful tasks include: handwriting recognition, speech recognition, machine translation, parsing, image captioning
  - LSTM became the dominant approach

- Now (2019), other approaches (e.g. Transformers) have become more dominant for certain tasks.
  - For example in **WMT** (a MT conference + competition):
  - In WMT 2016, the summary report contains ”RNN” 44 times
  - In WMT 2018, the report contains “RNN” 9 times and “Transformer” 63 times

LSTM vs GRU

LSTM vs GRU

- Researchers have proposed many gated RNN variants, but LSTM and GRU are the most widely-used
- The biggest difference is that GRU is quicker to compute and has fewer parameters
- There is no conclusive evidence that one consistently performs better than the other
- LSTM is a good default choice (especially if your data has particularly long dependencies, or you have lots of training data)
- Rule of thumb: start with LSTM, but switch to GRU if you want something more efficient

Other Models

- LSTM의 VGP 해결 방법의 개념을 이용해 여러 모델이 성능 향상을 보였음.
- 결국 LSTM은 현재 정보를 계산하기 위해 필요한 정보들을(이전 정보 & new input) 얼마나 반영하는지가 핵심이다.
- 기존 RNN의 repeated multiplication by the same weight matrix in BPTT 문제를 아예 다른 방식인 cell memory로 해결했다.

- 이렇게 이전 정보들을 조금만 반영한다면 그 state는 skip한다고 볼 수 있고 forward & backward propagation 시 VGP 해결에 도움이 되며 모델이 깊어질수록 효과가 큼.

- 이 개념을 CNN에 적용한 것이 ResNet, HighwayNet 등이 있음

Bidirectional RNNs

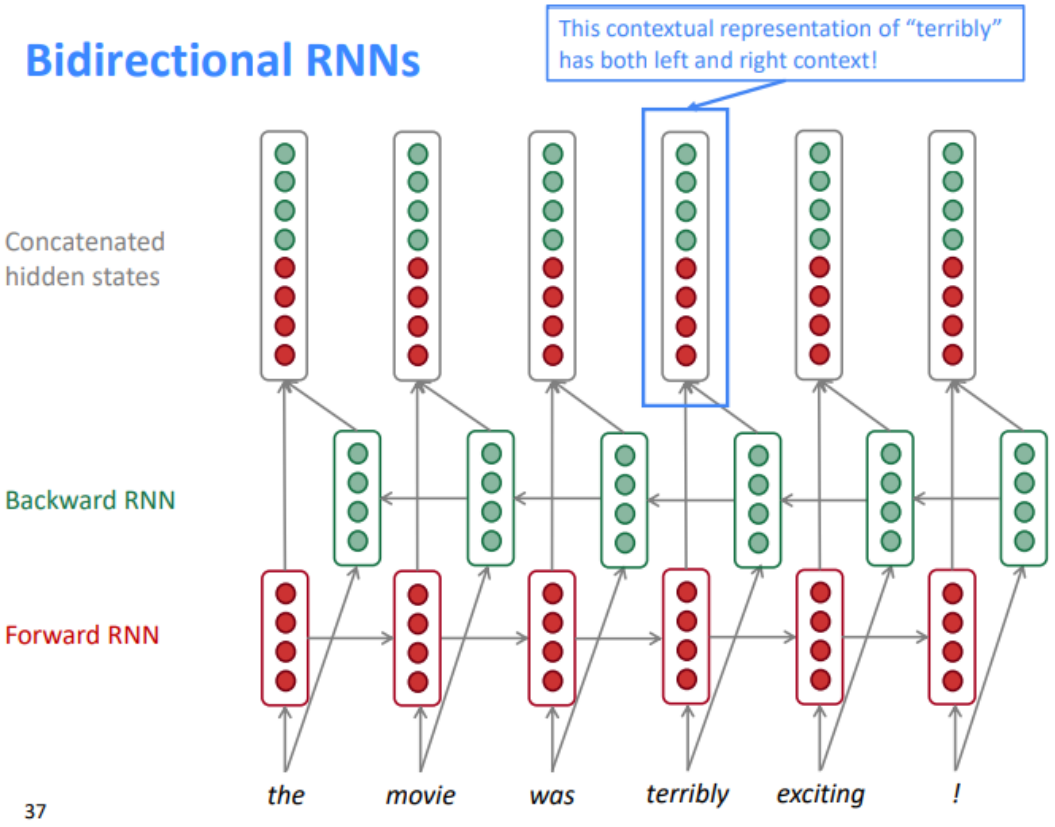
Bidirectional RNNs

- Note: bidirectional RNNs are only applicable if you have access to the entire input sequence.
  - They are **not** applicable to Language Modeling, because in LM you *only* have left context available.
- If you do have entire input sequence (e.g. any kind of encoding), bidirectionality is powerful (you should use it by default).
- For example, BERT (**Bidirectional** Encoder Representations from Transformers) is a powerful pretrained contextual representation system built on bidirectionality.
  - You will learn more about BERT later in the course!

40

Bidirectional RNNs

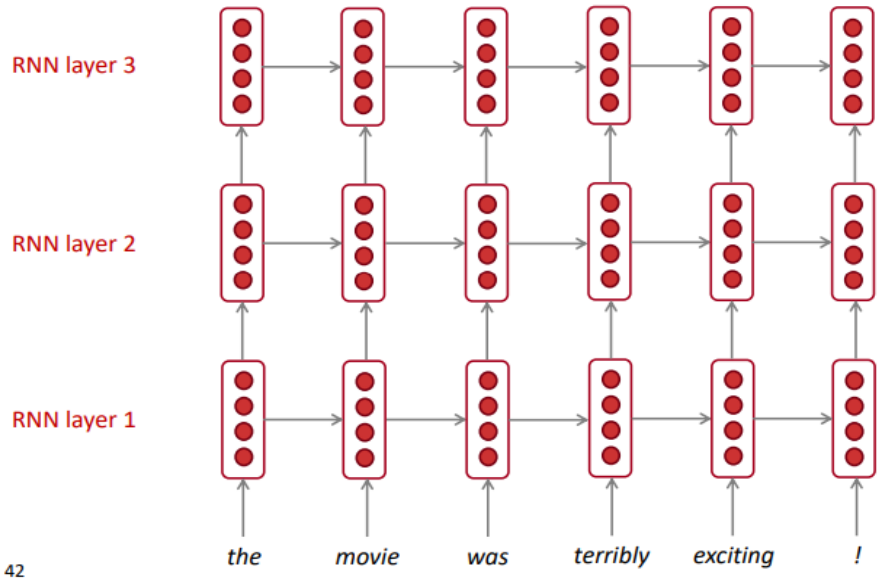
•



Multi-layer RNNs

Multi-layer RNNs

The hidden states from RNN layer *i* are the inputs to RNN layer *i+1*



42

Multi-layer RNNs

- RNNs are already "deep" on one dimension (they unroll over many timesteps)
- We can also make them "deep" in another dimension by applying multiple RNNs – this is a multi-layer RNN.
- This allows the network to compute more complex representations
  - The lower RNNs should compute lower-level features and the higher RNNs should compute higher-level features.
- Multi-layer RNNs are also called *stacked RNNs*.

참고 문헌

한글 블로그 자료

- https://aikorea.org/blog/rnn-tutorial-3/
- https://jeongukjae.github.io/posts/cs224n-lecture-7-vanishing-gradients-fancy-rnns/
- https://ratsgo.github.io/natural%20language%20processing/2017/03/09/rnnlstm/

Cs224n-winter 2019-lecture 7

https://www.youtube.com/watch?v=QEw0qEa0E50&list=PLoROMvodv4rOhcuXMZkNm7j3fVwBBY42z&index=7

CS224n-winter 2019-syllabus-lecture 7

https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1194/

Questions & Answers

Thank you