Clockwork RNN

Reference

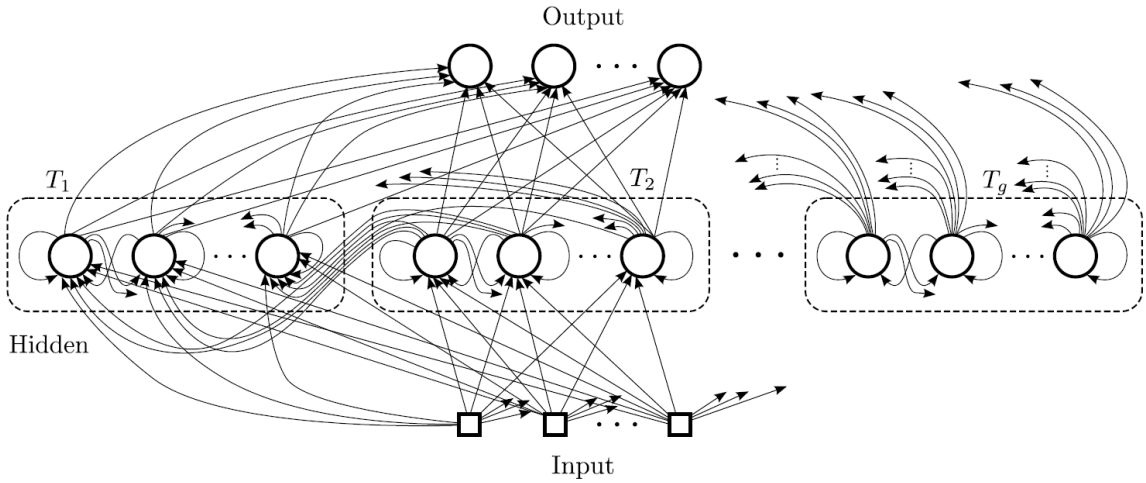
- A Clockwork RNN, Jan Koutnik et al.

RNN이 세상에 처음 나왔을 때, sequential data를 처리하는 탁월한 능력을 보이며 사람들의 주목을 받기 시작했다. 하지만 RNN도 단점이 있었는데, 바로 vanishing gradient 문제이다. Language Modeling에서 t번째 단어를 예측하고 싶을 때, $t-1$번째 단어가 중요하게 쓰일 수도 있지만, 더 먼 시점에 있는 단어를 무시할 수는 없다. RNN은 vanishing gradient 문제로 인해 먼 시점에 대한 gradient가 0과 근접하며, 따라서 먼 시점이 현재 시점에 미치는 영향을 제대로 파악할 수 없다. 그 영향이 실제로 작다면 관계 없지만, 실제로 작은지 아니면 vanishing gradient로 인해서 작게 보이는 것인지 알 수 없기 때문에 문제가 되는 것이다.

이러한 RNN의 문제점을 보완하는 대표적인 모델이 LSTM이다. LSTM과 이를 조금씩 변형한 모델에 대해서는 여기에 자세히 정리해두었다.

한편, 다른 흐름으로 LSTM을 수정하지 않고 새로운 방법으로 vanishing gradient 문제를 해결하고자 하는 모델이 바로 Clockwork RNN이다.

Model



위 그림은 clockword RNN (CW-RNN)의 아키 텍쳐이다. 특징은 아래와 같다.

- CW-RNN은 기존의 RNN과 유사하게 input, hidden, output layers로 구성되어 있다.
- 기존의 RNN과는 다르게, hidden layer가 크기가 k인 g개의 module로 partition 되어 있다.
- g개의 modle은 clock period, T_n라는 것을 부여 받는다: $T_n \in \{T_1, \cdots, T_g\}$. T_n이 크다는 것은 slow module임을, 작다는 것은 fast module임을 의미한다.
- 각 module은 internally fully-interconnect 되어 있다. 위 그림에서 각 module안에 있는 node 들이 서로를 향해 화살표를 쏘고 있다.
- 모듈 간의 recurrent connections은 다음과 같이 정의된다: 모듈 j에서 모듈 i로의 connections은 $T_i < T_j$인 경우에만 성립한다. 즉, 위 그림에서 모듈 간의 화살표를 보면 오른쪽에서 왼쪽으로 되어 있다. 논문에서 위 그림은 clock period을 오름차순으로 정렬했다고 한다. 즉, 왼쪽 모듈일수록

clock period가 작아지므로 화살표가 왼쪽으로 향하는 것이다. (slower modules to faster modules)

기존의 t 시점 에서 RNN output인 $y_O^{(t)}$는 아래와 같이 hidden layer와 input layer을 이용하여 계산된다.

$$y_H^{(t)} = f_H(W_H \cdot y^{(t-1)} + W_I \cdot x^{(t)})$$

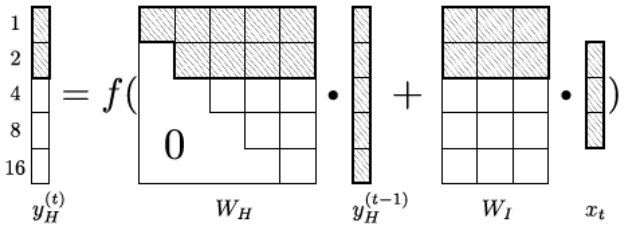
$$y_O^{(t)} = f_O(W_O \cdot y_H^{(t)})$$

CW-RNN과 RNN의 가장 큰 차이점은 t시점 에서 $(t\ MOD\ T_i) = 0$ 을 만족하는 i번째 module의 output만 실행된다. time period T_n의 선택은 aritrary하며 이 논문에서는 exponential series periods을 사용하였다. 즉, i번째 module은 clock period $T_i = 2^{i-1}$을 가진다.

hidden layer을 크기가 k인 g개의 module로 나눈다고 했으므로 행렬 W_H, W_I 또한 g개의 blocks-rows로 나눌 수 있다.

$$\mathbf{W}_H = \begin{pmatrix} \mathbf{W}_{H_1} \\ \vdots \\ \mathbf{W}_{H_g} \end{pmatrix} \qquad \mathbf{W}_I = \begin{pmatrix} \mathbf{W}_{I_1} \\ \vdots \\ \mathbf{W}_{I_g} \end{pmatrix}$$

아래 예시를 자세히 살펴보자.



hidden layer을 $g = 5$, 다섯 개로 나눈 상황이고 각 module에 대해서 time period는 1, 2, 4, 8, 16 즉 $T_i = 2^{i-1}$로 두었다. time step $t = 6$에서 $(t\ MOD\ T_i) = 0$ 을 만족하는 i는 무엇일까? $T_i = 1, 2$가 위 조건을 만족하므로 위 예시에서도 두 module에 대해서만 outputs을 계산하는 것이다. 즉, W_H, W_I의 첫 두 행만 outputs 계산에 관여한다. 이를 아래와 같이 표현할 수 있다.

$$\mathbf{W}_{H_i} = \begin{cases} \mathbf{W}_{H_i} & \text{for } (t \bmod T_i) = 0 \\ \mathbf{0} & \text{otherwise} \end{cases}$$

결국 time period을 도입하는 하고 이를 이용하여 일부만 output 계산에 관여하게 한다는 것이 이 논문의 골자인데, 그러면 time period을 도입하는 것이 어떤 장점이 있는 것일까?

time period가 작은 module에 대해서 생각해보자. 이 module은 $(t\ MOD\ T_i) = 0$ 을 만족할 가능성이 크다. 왜냐하면 시점이 커짐에 따라서, 즉 t가 커지면서 T_i는 작기 때문에 위 조건을 만족할 기회가 많이 주어지는 것이다.

반면에 time period가 큰 module에 대해서 생각해보자. T_i가 크다면 t가 작을 때는 위 조건을 만족시킬 수 없을 것이다. 우선 $t < T_i$라면 나머지가 0이 될 수가 없기 때문에 시점 t가 T_i에 도달하기 이전까지 i번째 module은 output 계산에 관여하지 못할 것이다.

즉, T_i가 작은 module (= low-clock-rate module)은 시점 t가 작을 때부터 output 계산에 관여하여, input sequences에서 얻은 long-term 정보를 처리하는 역할을 맡는다. 반면 T_i가 크다면 시점이 어느 정도 지난 후에 output 계산에 관여할 것이므로 local 정보를 처리하는 역할을 맡는다.

이런식으로 LSTM의 cell state처럼, vanishing gradient의 문제를 해결하기 위해 이전의 정보를 기억하는 module이 존재하는 것이다.