조문기

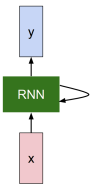
RNN에 대해서

RNN의 정의

- RNN (recurrnet neural network)는 main이 LSTM(long short term memory models)
- RNN의 가장 큰 특징 : hidden layer (node)가 존재
- Voice / Text detection or modification에 data처리 작업에 자주 쓰임

RNN의 특징

- Input 대비 Output (I : O)에 따라서 다양한 type가 존재한다.



- one to one

하나의 input에 해당하는 하나의 output : | 을 ,로 전환하거나 하는 방식

- one to many

사진 -> 단어들 (사진설명 붙이기)

초원의 그림 하나가 들어오는 경우 이에 대해서 웅장함, 평화로움 등등 설명할 수 있는 단어들을 매길 수 있음

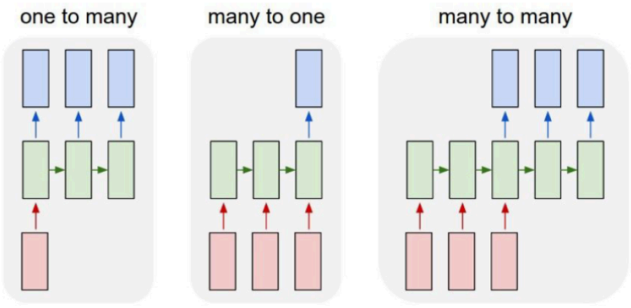
- many to one

편지 -> 감성정도 (편지의 감정의 정도 구하기)

편지 (string들의 조합)이 들어왔을때 이 편지는 어떤 감정을 가질지, 즉 공허함, 슬픔 등을 가지고 있는지 표현 할 수 있음

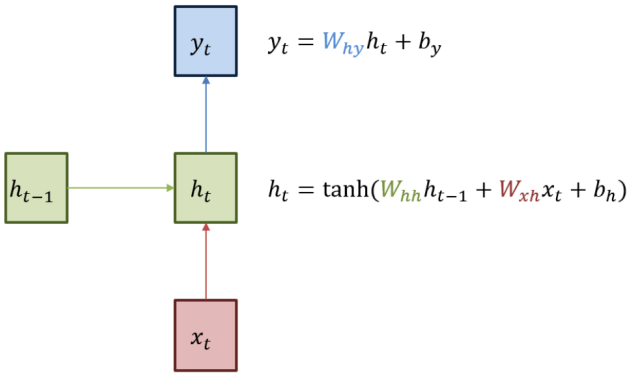
- many to many

번역 프로그램과 같은 것들이 여기에 속한다. 호메로스의 서사를 입력 했을때 우리 말로 표현이 가능한 경우를 뜻함

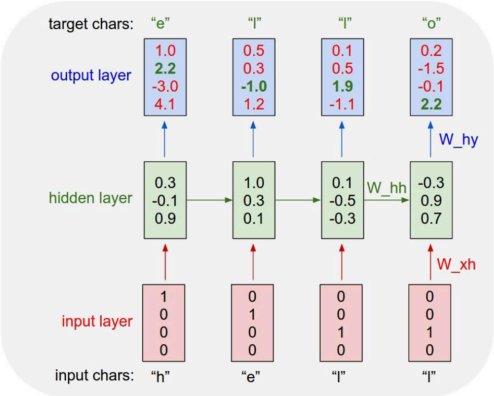


RNN의 구현 방식과 문제점 및 해결점

- RNN의 활성함수 (activation function)은 tanh을 채택하고 있음
- tanh - 비선형 함수 (Relu등과 다름)
- 그 이유는 다음과 같음, 만약 선형함수를 활성함수로 채택해서 3차원을 쌓는다면 f(x) = cx인 경우 f(f(f(x))) = c^3x가 된다. 이건 결국 constant인 c를 3제곱한 a가 존재하면 g(x) = ax랑 같은 것이 되어서 활성함수를 채택하는 장점이 없는 것임

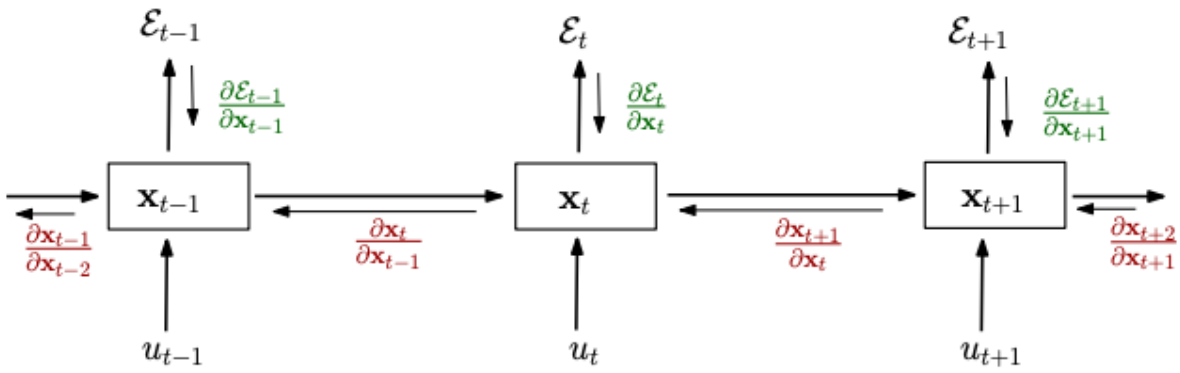


h1을 구할때 h0은 값이 주어지지 않는 상황은 h0을 random value를 통해 계산하게 됨



- hell 이 입력되면 hello를 출력하는 프로그램을 짜려고 한다고 해보자, h e l o에 대해서 one-hot-vector를 설정하면 각각 [1,0,0,0], [0,1,0,0], [0,0,1,0], [0,0,0,1] 이 할당이 됨 그 이후 각자의 input에 따라서 output stream이 나오게 되는데 이 상황을 foward propagation(순전파) 라고 칭함
- 이 값들을 보면 이제 컴퓨터는 가장 큰 값을 선택하게 되어 있는데 이 것이 옳지 않은 경우 고쳐주어야 함, 수정이 이루어지는 부분은 Whh Whx Why 이 세 부분임
- 첫번째 h에 대해서 e를 산출해야만 하는데, 가장 큰 값은 4번째, o를 산출하려고 하기에 이를 loss에 따른 수정을 하고자 함, 이 상황을 backward propagation(역전파) 라고 칭함

조문기



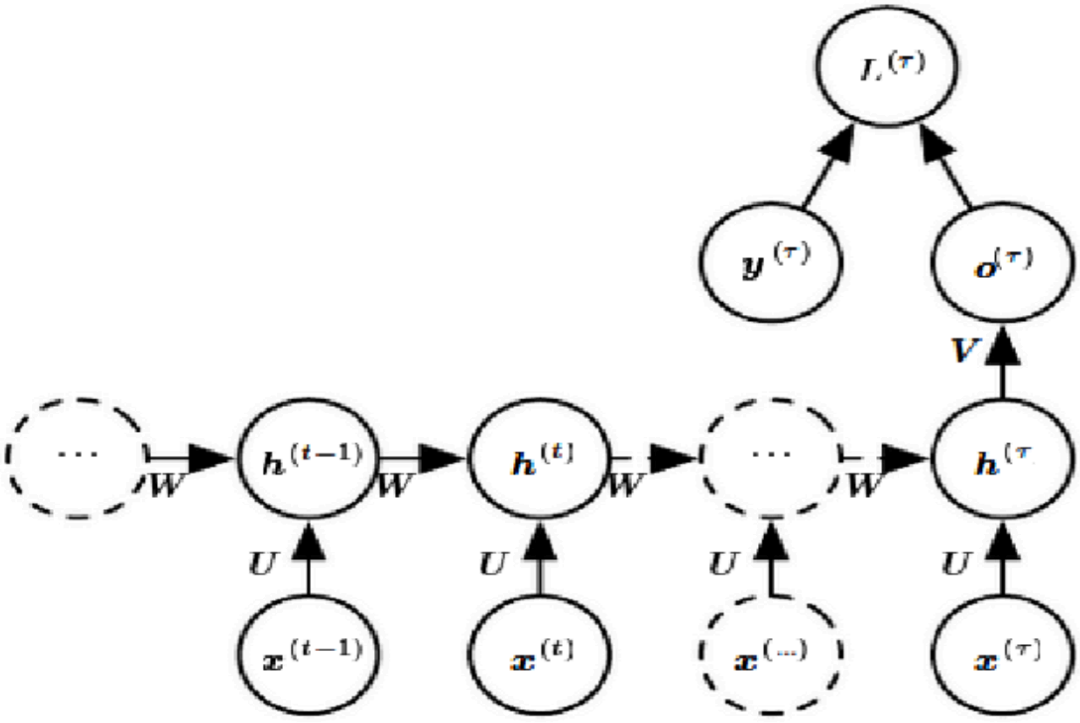
- 주목해야 하는 부분은 tanh를 통한 활성함수를 지나갔던 부분임, tanh의 역함수를 통해 역전파가 진행하게 되는데, 이때 gradient로 인한 문제점이 생김
- sigmoid계열의 함수들이 일으키는 문제점으로, 1986년부터 확인이 되었으나, 2006년 CIFAR의 hinton교수가 해결함 3차 이상의 네트워크를 쌓을 때, 값이 급격하게 내려가는 것이 주 문제점 이었음
- 음수값의 back propagation이 문제를 일으키고 cost의 급증을 야기함
- 이 문제를 vanishing gradient problem이라고 함
- 이를 해결하기 위해서 LSTM이라는 것이 제시 되었음

RNN과 LSTM의 차이점 및 그 특징

- LSTM은 state를 그냥 보내지 않고 저장하는 방식을 채택해서 layer가 높아질때 (ex 1 - 6차원의 차이를 1 - 3, 3 - 6) 간극을 낮춰주는 기능을 수행함
- 과거 정보들을 사용할때 sigmoid함수를 채택함 (⊙는 hadamard function, element다 합)

$$
\begin{aligned}
f_t &= \sigma(W_{xh\_f} x_t + W_{hh\_f} h_{t-1} + b_{h\_f}) \\
i_t &= \sigma(W_{xh\_i} x_t + W_{hh\_i} h_{t-1} + b_{h\_i}) \\
o_t &= \sigma(W_{xh\_o} x_t + W_{hh\_o} h_{t-1} + b_{h\_o}) \\
g_t &= \tanh(W_{xh\_g} x_t + W_{hh\_g} h_{t-1} + b_{h\_g}) \\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
h_t &= o_t \odot \tanh(c_t)
\end{aligned}
$$

조문기



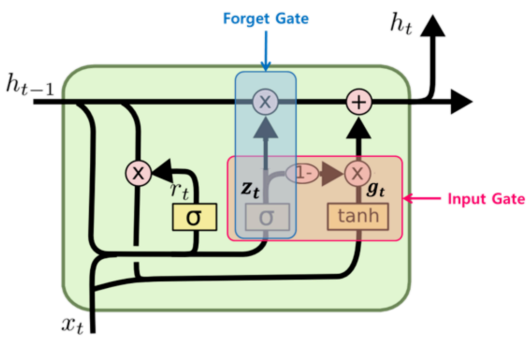
- f_t는 과거 정보를 잊기 위한 게이트임, 시그모이드 함수의 출력 범위는 0~1사이기 때문에 값이 만약 0이라면 잊어버리고, 1이면 이전 상태의 정보를 기억하게 됨
- i_t는 현재 정보를 기억하기 위한 것임, ht는 ft, it, gt, ot모두를 통합한 행렬임

LSTM과 GRU

- LSTM이 이렇게 RNN의 문제를 해결하려고 해도 sigmoid와 tanh를 사용하는 기본적 문제에 봉착해 낮은 ratio와 높은 계산 요구를 가지게 되는 것이 대두댐
- 이를 완화 (해결이 아님 애초에 근본적 문제라서) 하기 위해 GRU라는 것이 나옴
- GRU는 LSTM의 계산 요구를 낮추는 구현임

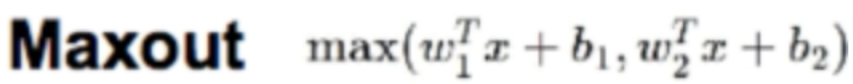
$$
\begin{aligned}
\mathbf{r}_t &= \sigma\left(\mathbf{W}_{xr}^T \cdot \mathbf{x}_t + \mathbf{W}_{hr}^T \cdot \mathbf{h}_{t-1} + \mathbf{b}_r\right) \\
\mathbf{z}_t &= \sigma\left(\mathbf{W}_{xz}^T \cdot \mathbf{x}_t + \mathbf{W}_{hz}^T \cdot \mathbf{h}_{t-1} + \mathbf{b}_z\right) \\
\mathbf{g}_t &= \tanh\left(\mathbf{W}_{xg}^T \cdot \mathbf{x}_t + \mathbf{W}_{hg}^T \cdot (\mathbf{r}_t \otimes \mathbf{h}_{t-1}) + \mathbf{b}_g\right) \\
\mathbf{h}_t &= \mathbf{z}_t \otimes \mathbf{h}_{t-1} + (1 - \mathbf{z}_t) \otimes \mathbf{g}_t
\end{aligned}
$$

- GRU (gated recurrent unit)
- LSTM Cell에서의 두 상태 벡터 ct / ht -> 가 하나의 벡터 ht로 통합
- zt가 gate controller이고, 이 값은 0 또는 1을 가져서 0일 경우 forget는 닫히고, input가 열림, 1일 경우 정반대로 작동
- ht에 있어서 ht-1의 어느 부분이 작동할지를 선택하게 해주는 rt가 존재



sigmoid / tanh 그러면 어쩌란 말이냐?

- sigmoid 함수는 결국 0~1로 mapping을 하는데 모든 값이 결국 0으로 수렴하려고 하며 뉴런들이 0으로 되어 죽어버리는 문제가 생겼었음, 이를 해결하기 위한 것이 tanh로 -1 ~ 1로 mapping을 하는 방법인데, 이것도 음수가 생길때 back propagation의 문제가 생겼음
- 그래서 이 둘을 모두 해결하기 위해서 relu function이 생긴 것인데, 여기의 문제점은 결국 0보다 작은 값은 다 0으로 mapping을 하게 되고, 양수는 그대로 내보내게 되어서 네트워크를 쌓는 것의 이점을 다 보지 못하는 것이 생김
- ReLU를 그래서 업그레이드 하기 위해서 Leakly ReLU (0.01x, x) 나 ELU(alpha * (e^x - 1), x)를 사용했는데, 높은 ratio를 보여주고 있음 (ELU의 경우 대신 exp의 계산 소요가 생김)
- 그러나 ReLU가 0이나 0에 가까워지는 순간 계속 죽어버리는 dying ReLU problem에서 자유롭진 못했음 (ELU의 경우 자유롭긴 함)
- 이 모든 것중 가장 좋은 것이 생기게 되었는데, 이 것이 maxout activation function임
- maxout : 이 함수가 ReLU function의 모든 것들을 포함함



- 어떤 뜻이냐면, ReLU는 결국 w1, b1이 0일때의 maxout과 같은 것이고, 나머지 모든 것들을 포함 할수 있어서 많은 케이스들을 적용해 높은 값을 산출할 수 있어진 것임
- 그러나 parameter가 2배로 증가해서 계산량도 2배인 것이 문제임

결론

- sigmoid / tanh 둘다 RNN빼곤 사용하지 않는게 좋음 (구조상 이득을 취하고 싶어서임)
- 거의 모든 뉴럴 네트워크에서는 maxout함수를 활성함수로 채택하는 것이 큰 performance를 보여주게 됨