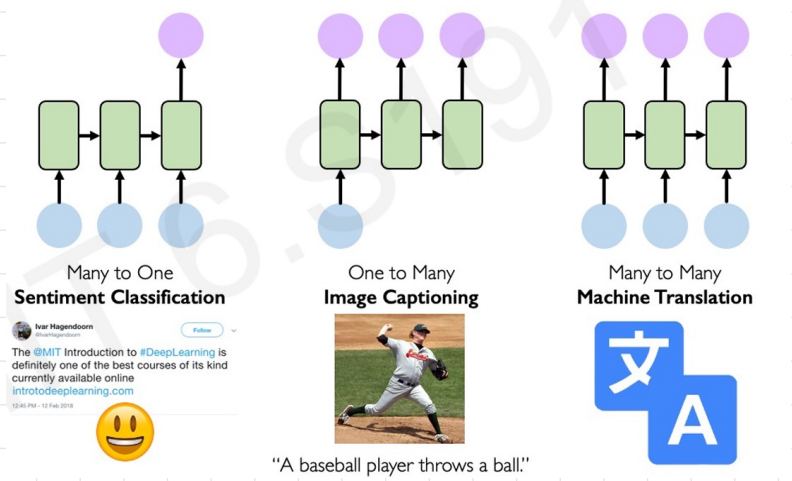
Sequential data

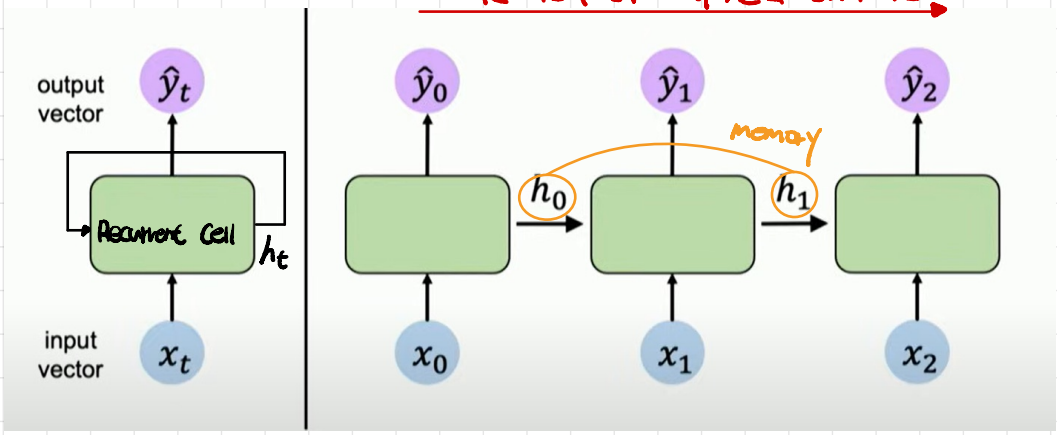
- 음파, 또는 글의 움직임, 텍스트, 주가...

Sequential Modeling Application



Neurons with Recurrence

이전 시간의 정보가 이후 시점으로 넘어가도록한다

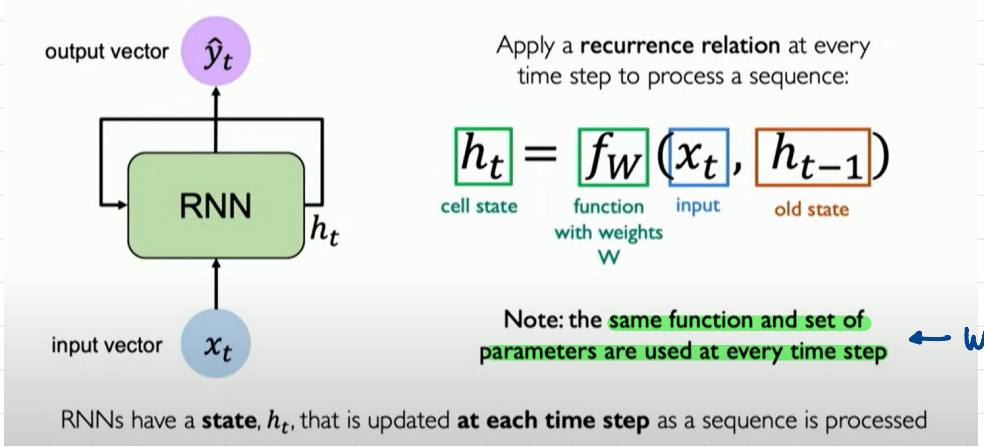


$$\hat{Y}_t = f(x_t, h_{t-1})$$

t: time , h_0 : Inner State / Memory term

- $\hat{Y}_t$는 t 때의 입력과 과거 메모리에 따라 달라진다.

Recurrent Neural Networks (RNNs)



Output Vector

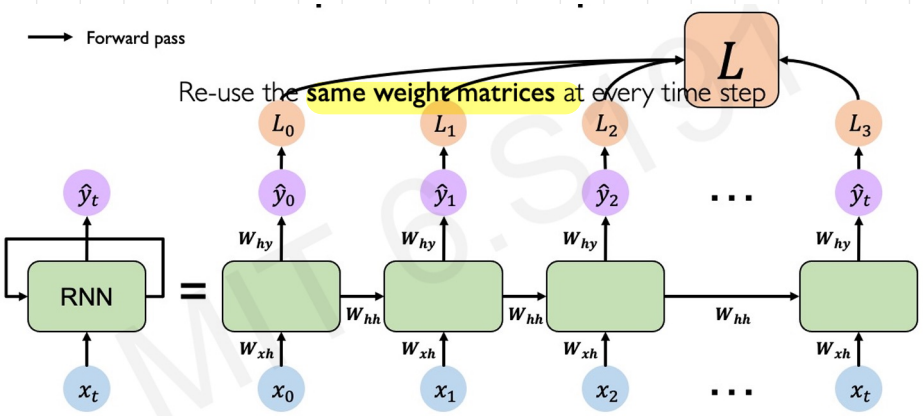
$\hat{Y}_t = W_{hy}^T h_t$

Update Hidden State

$h_t = \tanh(W_{hh}^T h_{t-1} + W_{xh} x_t)$

(when input vector is x_t)

← W, H_t는 계속 달라진다.



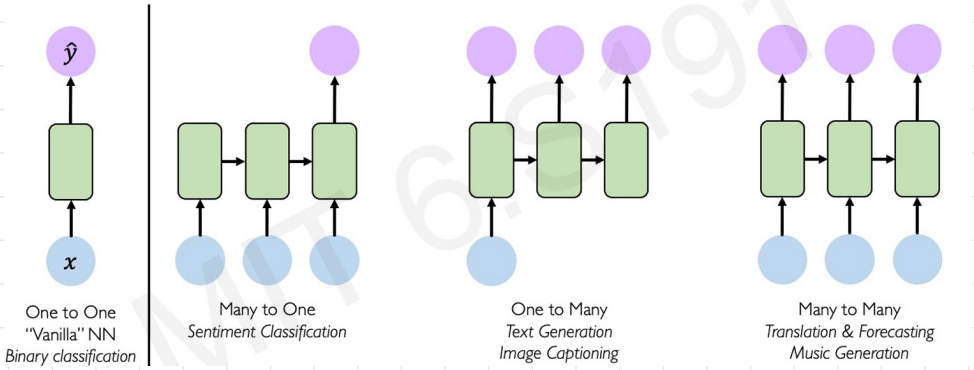
L_t : t번째 time 에서의 일시적인 손실

L : L_t (모든 개별 손실 값)을 합계 가져다 활용하여 총손실 구한다.

↳ 총손실

$$L = \sum_{t=0}^{T} L_t(\hat{Y}_t, Y_t)$$

RNN Models



What Sequential Models need to deal with

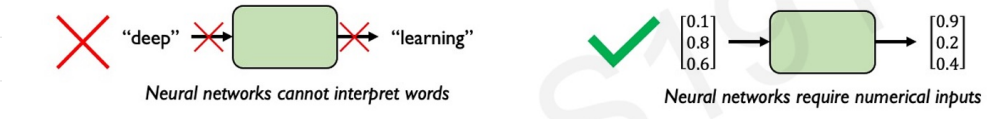
1. Variable-length sequences
2. Track long-term dependencies
3. Maintain information about order
4. Share parameters across the sequence

→ RNN은 4가지 모두 만족함.

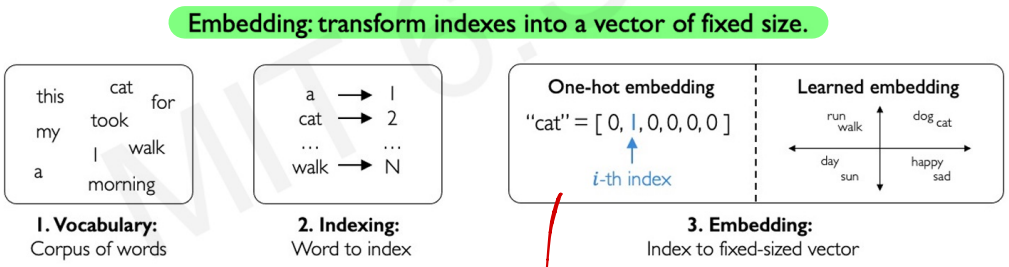
Example of RNN : Word prediction

Neural Network는 Text data가 무엇을 의미하는지 정확히 알지 못하고 숫자 계산만 할 수 있다.

→ 우리는 Text를 Numerical inputs로 변환해야한다.



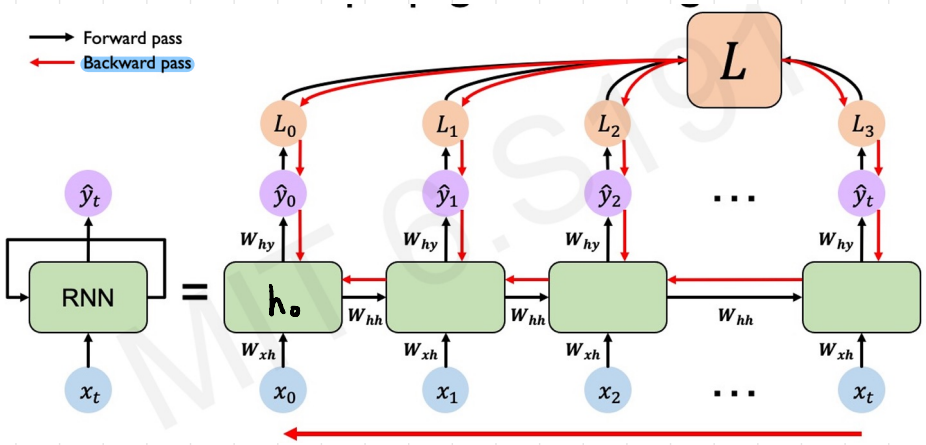
- Training을 통해 내재적인 의미를 해석하고 최대한 유사한 숫자 벡터를 가지도록 하는 것이 목표이다.



Vector 중 단 하나의 값만 1이고 나머지 모든 값은 0인 Encoding

Training weights of RNNs

- Backpropagation Through Time (BPTT)



1. $\frac{\partial L_t}{\partial W_{hy}} = \frac{\partial L_t}{\partial \hat{y}_t} \cdot \frac{\partial \hat{y}_t}{\partial W_{hy}}$

2. $\frac{\partial L_t}{\partial W_{hh}} = \frac{\partial L_t}{\partial h_t} \cdot \frac{\partial h_t}{\partial W_{hh}}$

3. $\frac{\partial L_t}{\partial W_{xh}} = \frac{\partial L_t}{\partial h_t} \cdot \frac{\partial h_t}{\partial W_{xh}}$

$h_t = f(W_{hh} h_{t-1} + W_{xh} x_t)$

EX) 4-Layer에서 $\frac{\partial L}{\partial h_0}$

완전한 식

$$\frac{\partial L}{\partial h_0} = \delta_3 W_{hy} f'(h_3) W_{hh} f'(h_2) W_{hh} f'(h_1) W_{hh} + \delta_2 W_{hy} f'(h_2) W_{hh} f'(h_1) W_{hh} + \delta_1 W_{hy} f'(h_1) W_{hh} + \delta_0 W_{hy}$$

이제 줄임없이 **완전한 역전파 과정**을 확인할 수 있어! 🚀🔥

→ W_{hh}가 매우 작아질 때 $\forall L \approx 0$ 이 될 수도 있다.

Gradient of h_0은 위에서 W_{hh} + repeated gradient Computation을 포함한다.

Case 1) Weight가 클 때

Exploding gradient 발생 가능

→ Gradient clipping으로 해결

Case 2) Weight가 매우 작을 때 (W_{hh})

Vanishing gradient 발생 가능

Solution 1) Activation function
2) Weight initialization
3) Network architecture

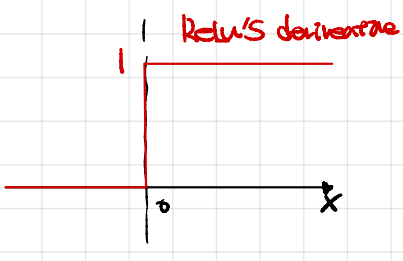
→ Gradient가 0에 수렴되지 제대로 된 학습이 이루어지지 않을 수 있다.

→ Long-term dependencies에 대해 너무 작은 값이 계속 곱해져서 소멸되어 학습이 불가능해진다.

Solution of Vanishing gradient

1. Activation function

- ReLU를 사용



- ReLU는 미분하면 $x > 0$ 에서의 항상 1이기 때문에 곱해도 괜찮다.

2. Parameter Initialize

$$I_n = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix}$$

Identity matrix로 weight를 초기화하면 곱해도 괜찮다.

3. Gated Cells

Gated cell

LSTM, GRU, etc

선택적으로 정보를 add or remove 할지 결정을 한다.

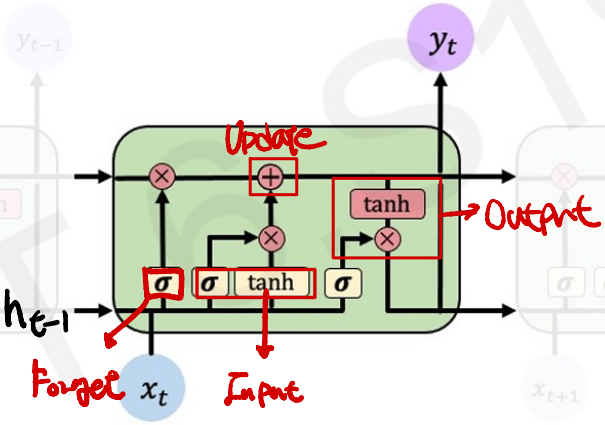
- 필요한 것 add, 필요없는 것 remove한다.

Example of Gated cell : LSTM

LSTM: RNN의 vanishing gradient 문제 해결 가능

Gated LSTM cells **control information flow:**

1) Forget 2) Store 3) Update 4) Output



1) Sigmoid 역할: forget gate

$0 \le \nabla \le 1$ 이고 0: forget, 1: maintain

$f_t = \nabla(W_f[h_{t-1}, x_t] + b_f)$

2) Input gate

[Sigmoid(∇): 새 정보의 중요도 결정
tanh: 새로운 정보의 벡터값 생성

$i_t = \nabla(W_i[h_{t-1}, x_t] + b_i)$

$\tilde{C}_t = \tanh(W_c[h_{t-1}, x_t] + b_c)$

3) Update gate

Forget cell + Input cell

$C_t = f_t C_{t-1} + i_t \tilde{C}_t$

4) Output gate

[Sigmoid : 어떤 정보로 출력할지 결정
tanh : 최종 h_t 계산

$O_t = \nabla(W_o[h_{t-1}, x_t] + b_o)$

$h_t = \tanh(C_t)$

→ Uninterrupted Gradient descent를 이용해 Backpropagation하여 Vanishing Gradient 문제를 줄 수 있다.

Limitations of RNNs

1. Encoding bottleneck

- 정보가 많다면 Encoding할 수도 많아지고, 이 모든 정보가 적절하게 유지되고 업데이트 되기 쉽지 않다.

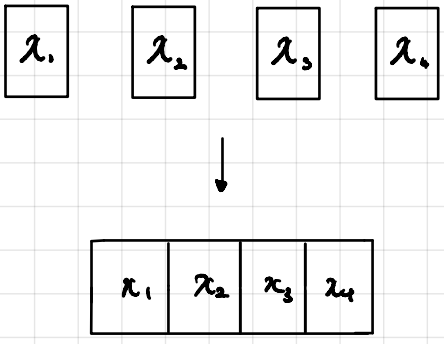
2. 시간별로 처리하면 매우 느릴 수 있다.

→ 병렬로 계산을 하는 방법도 있다.

3. RNN의 용량이 크지 않아 Long memory에 취약하다.

RNN의 문제점을 극복하려면?

→ 시간 단위의 입력 대신, 한 번에 입력해야한다.



하지만 RNN을 아무리 병렬로 한다 해도, 중요한 정보가 사라질 수 있고, 순서가 보장되지 않으며, Long term memory에 취약하다.

Attention : Transformer model의 핵심

Ex) 1. 내가 현재 본 것(query)과 저장된 것(key)의 연관성 비교

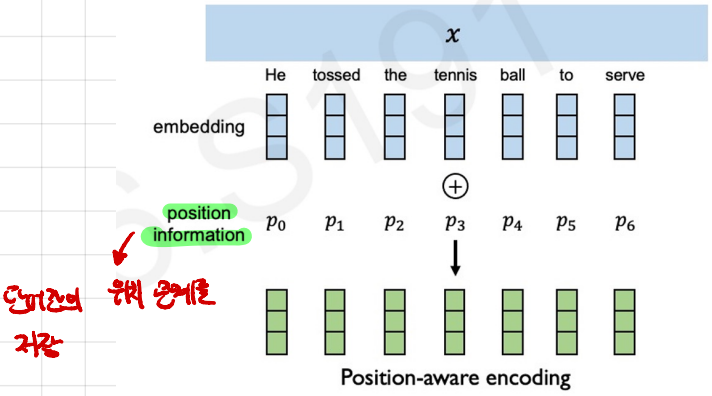
- Attention mask

2. 연관성이 높은 것에 Attention

- Extract values based on attention

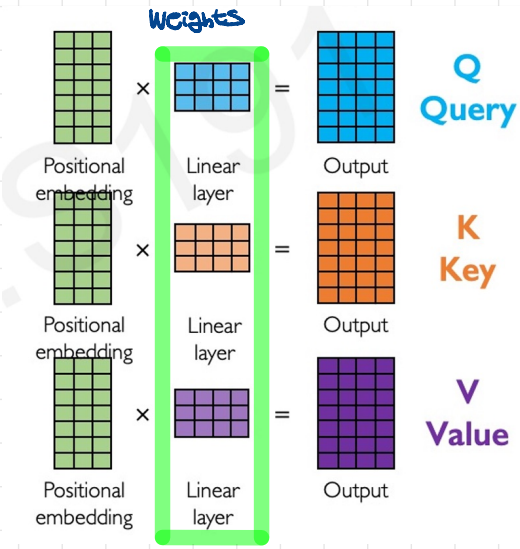
Learning Self-Attention With Neural Networks

1. Encode position information

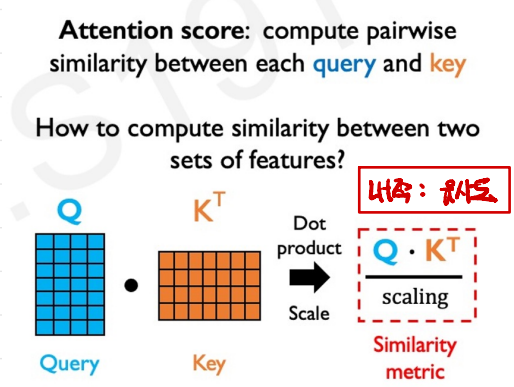


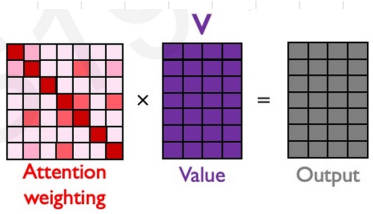
단어간의 위치 정보를 저장

2. Extract query, key, value

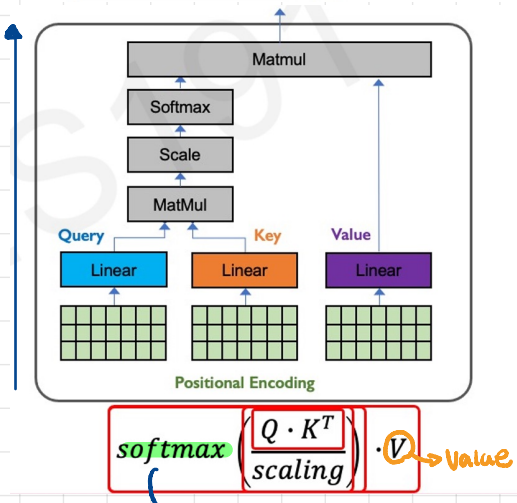


3. Compute attention weighting





$$softmax\left(\frac{Q \cdot K^T}{scaling}\right) \cdot V = A(Q,K,V)$$



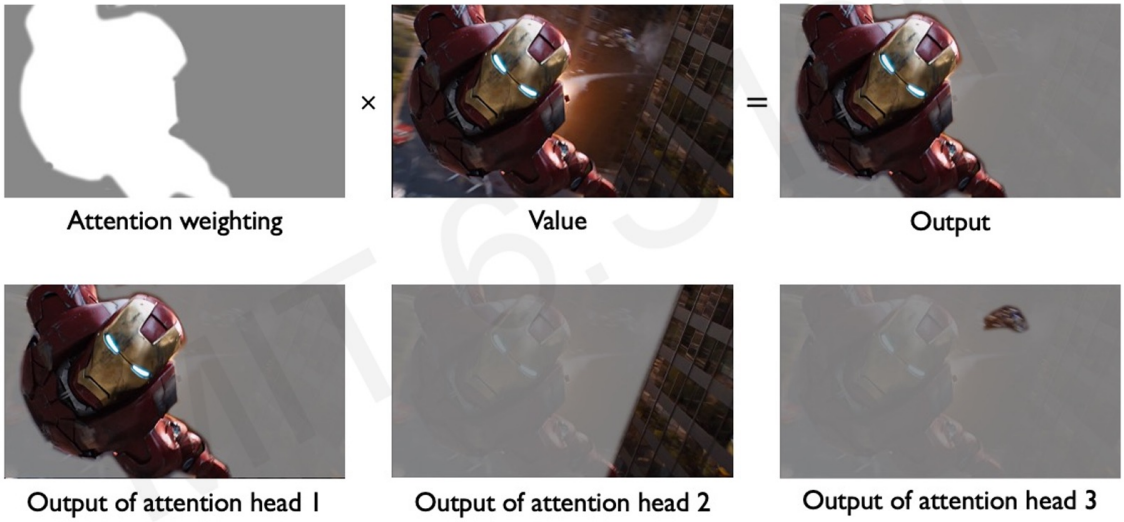
$0 \leq \quad \leq 1$

→ 상대적인 중요도를 알려줌.

위의 3가지 과정을 다 거친 마지막으로 Attention이 높은 features만을 선택한다.

Self Attention head

각 head는 W_Q, W_K, W_V를 가져와 각기 다른 방법으로 Attention을 계산할 수 있다.



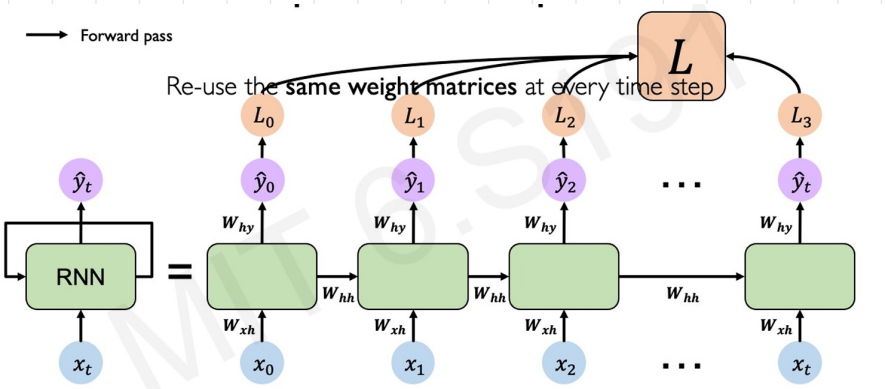
→ 각각의 attention head가 각각 얼굴, 배경, 옷선을 추출하고 있다.

Multi Attention

$$MultiHead(Q,K,V) = Concat(head_1, \dots, head_h)W_0$$

Concat: head를 연결 / W_0: 출력가중치

=> Long dependency를 효율적으로 학습할 수 있다.



$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

$\in [-1, 1]$

L_t: Cross entropy

MSE 등등...

- $h_0 = f(x_0 W_{xh})$, $\hat{y}_0 = W_{hy} \cdot h_0$, $L_0 = y_0 - \hat{y}_0$
- $h_1 = f(h_0 W_{hh} + x_1 W_{xh})$, $\hat{y}_1 = h_1 \cdot W_{hy}$, $L_1 = y_1 - \hat{y}_1$
- $h_2 = f(h_1 W_{hh} + x_2 W_{xh})$, $\hat{y}_2 = h_2 \cdot W_{hy}$, $L_2 = y_2 - \hat{y}_2$
- $h_3 = f(h_2 W_{hh} + x_3 W_{xh})$, $\hat{y}_3 = h_3 \cdot W_{hy}$, $L_3 = y_3 - \hat{y}_3$
- $L = \sum_{t=0}^{3} L_t$

Gradient - Multi Layer RNN

$$\frac{\partial L}{\partial W_{hy}^{(L)}} = \sum_{t=0}^{T} \frac{\partial L_t}{\partial \hat{y}_t} \cdot \frac{\partial \hat{y}_t}{\partial W_{hy}^{(L)}}$$

$$\frac{\partial L}{\partial W_{hh}^{(L)}} = \sum_{t=0}^{T} \frac{\partial L_t}{\partial h_t^{(L)}} \cdot \frac{\partial h_t^{(L)}}{\partial W_{hh}^{(L)}}$$

$$\frac{\partial L}{\partial W_{xh}^{(L)}} = \sum_{t=0}^{T} \frac{\partial L_t}{\partial h_t^{(L)}} \cdot \frac{\partial h^{(t)}}{\partial W_{xh}^{(L)}}$$

Example of RNN (4 Layers) - Feed forward

Data x: $\begin{bmatrix} 0.5 & 0.1 & -0.3 \\ -0.2 & 0.4 & 0.3 \\ 0.1 & -0.5 & 0.2 \\ 0.4 & 0.2 & -0.1 \end{bmatrix}$ y: $\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$ $W_{xh} = \begin{bmatrix} 0.2 & -0.3 \\ 0.4 & 0.1 \\ -0.5 & 0.2 \end{bmatrix}$ $W_{hh} = \begin{bmatrix} 0.6 & -0.2 \\ 0.1 & 0.5 \end{bmatrix}$ $W_{hy} = \begin{bmatrix} 0.3 & -0.7 \\ -0.5 & 0.8 \end{bmatrix}$

$$h_0 = f(W_{xh} \cdot x_0) = \begin{bmatrix} 0.2 & -0.3 \\ 0.4 & 0.1 \\ -0.5 & 0.2 \end{bmatrix}^T \cdot \begin{bmatrix} 0.5 \\ 0.1 \\ 0.3 \end{bmatrix} = \begin{bmatrix} (0.2 \times 0.5) + (0.4 \times 0.1) + (-0.5 \times 0.3) \\ (-0.3 \times 0.5) + (0.1 \times 0.1) + (0.2 \times 0.3) \end{bmatrix} = \underset{\tanh}{f}\left(\begin{bmatrix} 0.28 \\ -0.20 \end{bmatrix}\right) = \begin{bmatrix} 0.2728 \\ -0.1974 \end{bmatrix}$$

$$\hat{y}_0 = W_{hy} h_0 = \begin{bmatrix} 0.3 & -0.7 \\ -0.5 & 0.8 \end{bmatrix} \cdot \begin{bmatrix} 0.2728 \\ -0.1974 \end{bmatrix} = \begin{bmatrix} 0.2205 \\ -0.2469 \end{bmatrix}$$

$$L_0 = \frac{1}{2} \| y_0 - \hat{y}_0 \|^2 = \frac{1}{2} \left[(1 - 0.2205)^2 + (0 - (-0.2469))^2 \right] = 0.3478$$

전체 L: $\sum_{t=0}^{T} L_t$

★ h_t, $\hat{y}_0$, L_t는 모두 다르나 출력: W_{hh}, W_{xh}, W_{hy}는 동일하다.

BPTT formula induction $\quad \delta_t = \hat{y}_t - y_t$ 라 하자

- $\frac{\partial L}{\partial W_{hy}} = \frac{\partial}{\partial W_{hy}} \left(\sum_{t=0}^{T} L_t \right) = \frac{1}{2} \frac{\partial}{\partial W_{hy}} \left(\sum_{t=0}^{T} \| W_{hy} h_t - y_t \|^2 \right) = \sum_{t=0}^{T} (\| \hat{y}_t - y_t \|) \cdot h_t^T = \sum_{t=0}^{T} \delta_t \cdot h_t^T$
- $\frac{\partial L}{\partial W_{hh}} = \frac{\partial}{\partial W_{hh}} \left(\sum_{t=0}^{T} L_t \right) = \frac{1}{2} \frac{\partial}{\partial W_{hh}} \left(\sum_{t=0}^{T} \| W_{hy} h_t - y_t \|^2 \right) = \frac{\partial}{\partial W_{hh}} \left(\sum_{t=0}^{T} (\| W_{hy} \cdot \tanh(W_{xh} \cdot x_t + W_{hh} \cdot h_{t-1}) - y_t \|^2) \right)$
  $= \sum_{t=0}^{T} \delta_t W_{hy}^T (1 - h_t^2) h_{t-1}^T$ $\quad \left(\frac{d}{dz} \tanh(z) = 1 - \tanh^2(z) \right)$
- $\frac{\partial L}{\partial W_{xh}} = \frac{\partial}{\partial W_{xh}} \left(\sum_{t=0}^{T} L_t \right) = \frac{1}{2} \frac{\partial}{\partial W_{xh}} \left(\sum_{t=0}^{T} \| W_{hy} h_t - y_t \|^2 \right) = \frac{\partial}{\partial W_{xh}} \left(\sum_{t=0}^{T} (\| W_{hy} \cdot \tanh(W_{xh} \cdot x_t + W_{hh} \cdot h_{t-1}) - y_t \|^2) \right)$
  $\cdot \sum_{t=0}^{T} \delta_t W_{hy}^T (1 - h_t^2) x_t^T$