

인공지능을 위한 머신러닝 알고리즘

10. 재현 신경망

CONTENTS

1

재현 신경망의 원리

2


GRU, LSTM: 재현 신경망의 한계를 넘어

3

어떻게 기계가 글을 생성할 수 있을까

학습 목표

- 재현 신경망의 학습 원리를 이해할 수 있다.
- 재현 신경망이 가진 그라디언트 손실 문제를 이해할 수 있다.
- 언어 모델로써 재현 신경망이 어떻게 글을 생성하는지 이해할 수 있다.

A person's hands are shown holding a smartphone with a white screen. The background is dark with out-of-focus, warm-toned bokeh lights in shades of yellow, orange, and blue. A semi-transparent dark banner is at the bottom, containing a yellow decorative element and the title text.

1. 재현 신경망의 원리

■ 시계열 데이터 학습 모델

❖ 재현 신경망 (**Recurrent Neural Networks**)은 시계열 데이터를 확률적으로 모델링

● 연속된 데이터들의 집합 $X = (x_1, x_2, \dots, x_T)$

● x 가 나타날 확률

$$p(X) = p(x_1, x_2, \dots, x_T) = p(x_1)p(x_2 | x_1)p(x_3 | x_1, x_2) \dots p(x_T | x_1, \dots, x_{T-1}) = \prod_{t=1}^T p(x_t | x_{<T})$$

● 재현 신경망은 매 시간 단위 t 마다 다음을 계산

$$p(x_T | x_{<T}) = g(h_{T-1})$$

$$h_{T-1} = \phi(x_{T-1}, h_{T-1}) \quad \phi \text{ is non-linear activation function}$$

● 은닉층 = 컨텍스트 층 = 히스토리 층

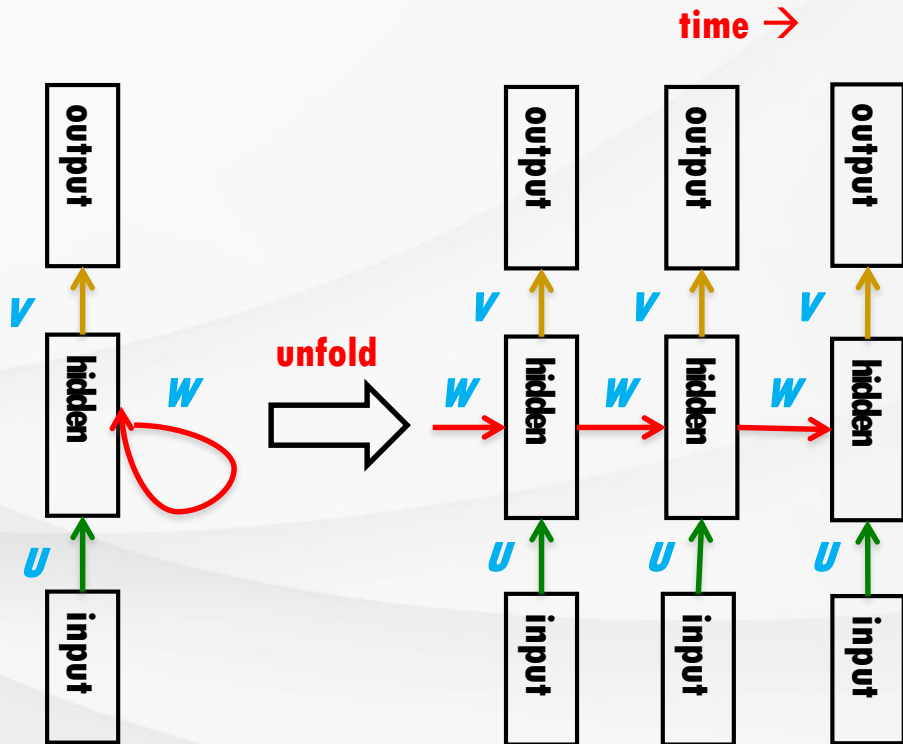
● 역전파를 사용하여 모델 파라미터 학습

■ 학습 가능한 파라미터

U, W, V

$$s_t = \tanh(Ux_t + Ws_{t-1})$$
$$y_t' = \text{softmax}(Vs_t)$$

- 은닉 유닛들은 연속된 벡터공간에서 오래 전 데이터 정보를 저장
- 많은 수의 뉴런과 시간이 주어진다면, 재현 신경망은 어떠한 시계열 데이터도 학습할 수 있음

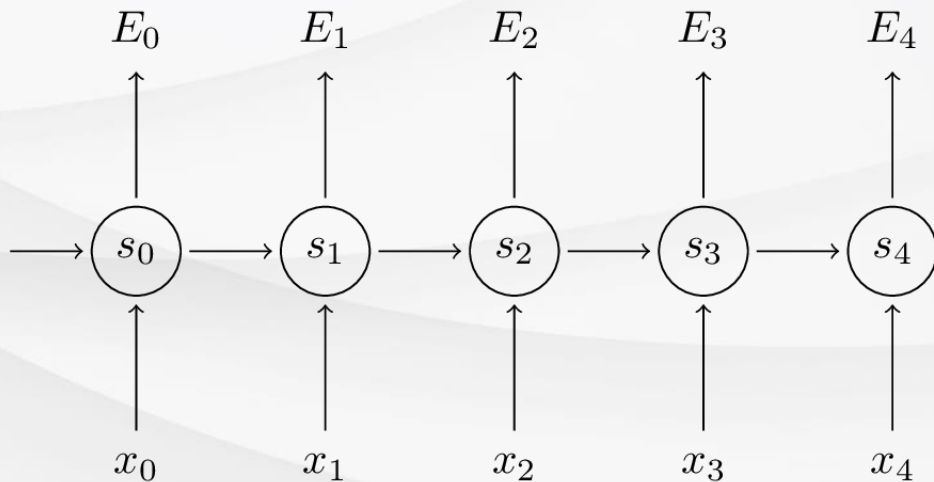


■ 재현 신경망의 손실 함수

- y_t 는 클래스 정보를 갖고 있는 이진 벡터
- t 번째 시간에서의 손실 함수는
정답 클래스의 마이너스
로그 확률
- 시계열 데이터 집합
(예> $\{x_0, x_1, x_2, x_3, x_4\}$)의
손실 함수는 각 시간
단위의 손실 함수의 총합

$$E_t(y_t, y'_t) = -y_t \log y'_t$$

$$E(y, y') = \sum E_t(y_t, y'_t) \\ = -\sum y_t \log y'_t$$



■ 시간에 대한 오류 역전파 과정

- **Back Propagation Through Time (BPTT)**

- 예> t=3에서 손실 함수에 대한 파라미터 W의 그라디언트 값 계산

- 체인을 사용

- t=3에서 손실 함수에 영향을 미치는 변수들로 y'_3 와 s_3 이 존재

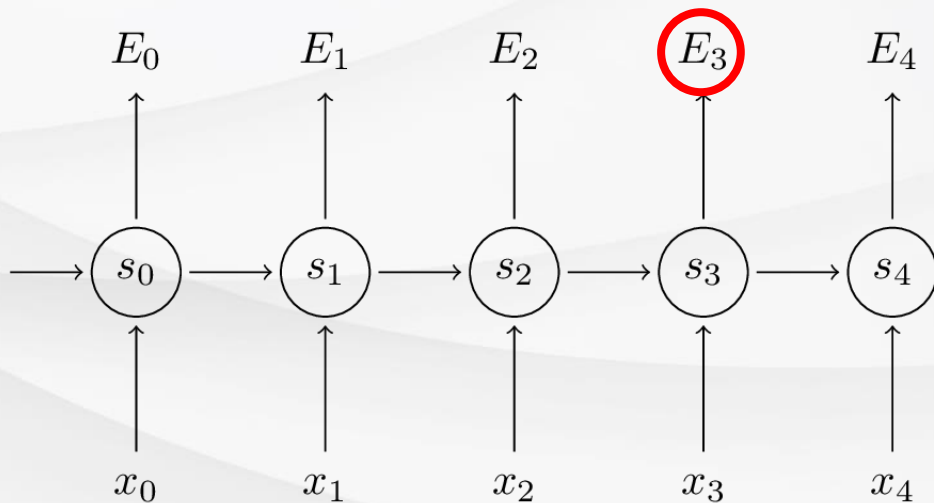
$$s_t = \tanh(Ux_t + Ws_{t-1})$$
$$y_t' = \text{softmax}(Vs_t)$$

$$\frac{\delta E_3}{\delta W} = \frac{\delta E_3}{\delta y'_3} \frac{\delta y'_3}{\delta s_3} \frac{\delta s_3}{\delta W}$$

Hidden weight parameter

Output vector

Hidden vector



■ 분류 과정

- 재현 신경망에서 $t=3$ 일 때, 은닉 유닛 s_3 의 값 계산

$$s_3 = \tanh(Ux_t + Ws_2)$$

- s_3 은 s_2 에 의해 영향을 받고, s_2 는 다시 W 와 s_1 에 영향을 받음
→ s_3 에 대한 s_2 , W , s_1 의 변화량을 계산해야 함
- 하지만, s_2 는 다시 s_1 과 W 에 의해 영향을 받으므로 상수로 볼 수 없음

$$\frac{\delta E_3}{\delta W} = \frac{\delta E_3}{\delta y'_3} \frac{\delta y'_3}{\delta s_3} \frac{\delta s_3}{\delta W}$$



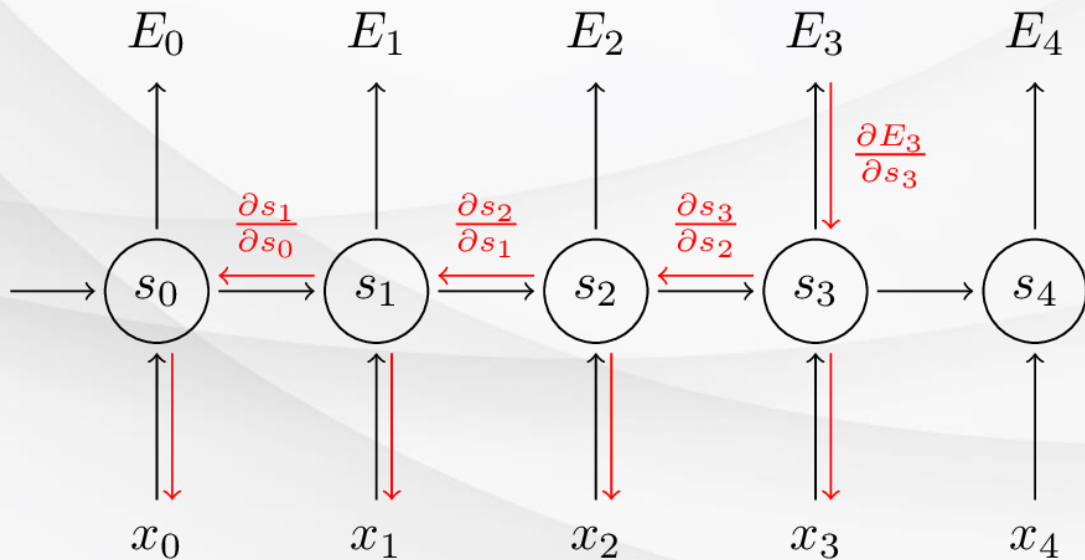
$$\begin{aligned} \frac{\delta E_3}{\delta W} &= \frac{E_3}{\delta y'_3} \frac{\delta y'_3}{\delta s_3} \frac{\delta s_3}{\delta W} + \frac{\delta E_3}{\delta y'_3} \frac{\delta y'_3}{\delta s_3} \frac{\delta s_3}{\delta s_2} \frac{\delta s_2}{\delta W} \\ &\quad + \frac{\delta E_3}{\delta y'_3} \frac{\delta y'_3}{\delta s_3} \frac{\delta s_3}{\delta s_1} \frac{\delta s_1}{\delta W} + \frac{\delta E_3}{\delta y'_3} \frac{\delta y'_3}{\delta s_3} \frac{\delta s_3}{\delta s_0} \frac{\delta s_0}{\delta W} \\ &= \sum_{k=0}^3 \frac{\delta E_3}{\delta y'_3} \frac{\delta y'_3}{\delta s_3} \frac{\delta s_3}{\delta s_k} \frac{\delta s_k}{\delta W} \\ &= \sum_{k=0}^3 \frac{\delta E_3}{\delta y'_3} \frac{\delta y'_3}{\delta s_3} \prod_{j=k+1}^3 \frac{\delta s_j}{\delta s_{j-1}} \frac{\delta s_k}{\delta W} \end{aligned}$$

$\frac{\delta s_3}{\delta s_2} \frac{\delta s_2}{\delta s_1} \frac{\delta s_1}{\delta s_0}$

■ 시간에 대한 오류 역전파 과정

- 그라디언트 계산 과정에서 다수의 곱셈이 필요함
- 시간에 따른 오류 역전파 과정이 생기는 이유는 같은 파라미터 \mathbf{w} 를 매시간 단위마다 반복해서 사용하기 때문

$$\frac{\partial E_3}{\partial W} = \sum_{k=0}^3 \frac{\partial E_3}{\partial y'_3} \frac{\partial y'_3}{\partial s_3} \prod_{j=k+1}^3 \frac{\partial s_j}{\partial s_{j-1}} \frac{\partial s_k}{\partial W}$$

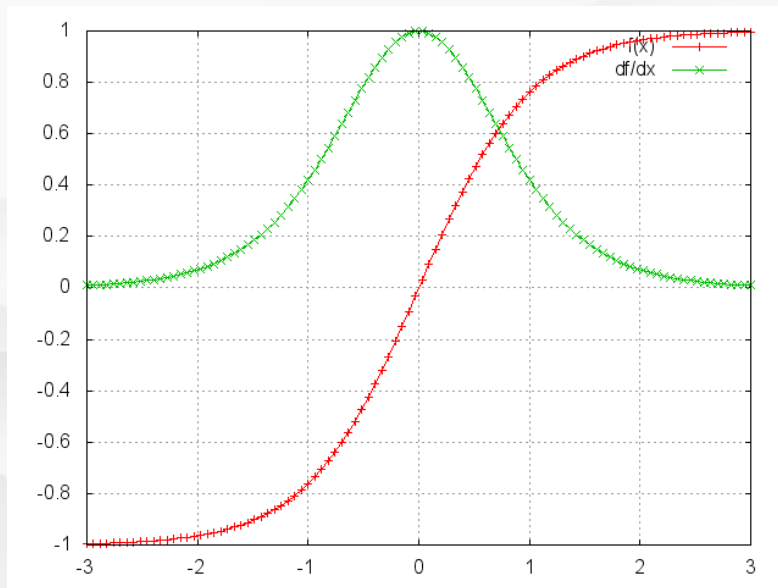


■ 그라디언트 손실 발생

$$\frac{\delta s_3}{\delta s_2} \frac{\delta s_2}{\delta s_1} \frac{\delta s_1}{\delta s_0} = Ws'(z_3)Ws'(z_2)Ws'(z_1) \quad (* zt = \tanh(Uxt + Wst_{1j}))$$


활성함수의 미분 값

- **tanh: 0~1**
- **시그모이드 : 0~0.25**
- 활성함수의 미분 값을 여러 번 곱해줄 경우 그라디언트의 값이 0으로 수렴할 수 있음
- 시간에 대한 길이가 깊어질수록 학습이 잘 되지 않음



■ 그라디언트 폭발 발생

- ◉ 만약 파라미터의 값이 매우 클 경우 (예 > 100 이상)
- ◉ $Ws'(z_3)Ws'(z_2)Ws'(z_1)$ 식에서 매우 큰 값을 계산하게 됨
- ◉ 그라디언트 값이 매우 커져서 **NaN**이 되거나 프로그램 종료 발생
- ◉ 해결책: 그라디언트 클리핑 (**Gradient Clipping**)
- ◉ 그라디언트 손실이 더욱 문제가 됨
 - 그라디언트 폭발은 프로그램 종료에 의해 알아채기 쉽고 해결책이 분명하지만, 그라디언트 손실은 발생 사실을 알기 어렵고 해결책이 불분명

A person's hands are shown holding a smartphone, with the screen glowing. The background is dark with out-of-focus, colorful bokeh lights in shades of yellow, orange, and blue. A semi-transparent dark blue horizontal bar spans the width of the image, containing the title text and a yellow decorative element on the left.

2. GRU, LSTM: 재현 신경망의 한계를 넘 어

■ GRU(Gated Recurrent Units)

일반적인 재현 신경망은 매 시간마다 은닉층을 다음과 같이 직접 계산함

$$h_t = f(Ux_t + Wh_{t-1})$$

GRU는 먼저 현재 입력 데이터와 은닉층을 기반으로 업데이트 게이트(**Update Gate**)를 계산

$$z_t = \sigma(U^{(z)}x_t + W^{(z)}h_{t-1})$$

리셋 게이트(**Reset Gate**)도 같은 식이지만 다른 파라미터를 사용하여 계산

$$r_t = \sigma(U^{(r)}x_t + W^{(r)}h_{t-1})$$

■ 잠재적 은닉 유닛 계산

업데이트 게이트

$$z_t = \sigma(U^{(z)}x_t + W^{(z)}h_{t-1})$$

리셋 게이트

$$r_t = \sigma(U^{(r)}x_t + W^{(r)}h_{t-1})$$

잠재적 은닉 유닛 계산

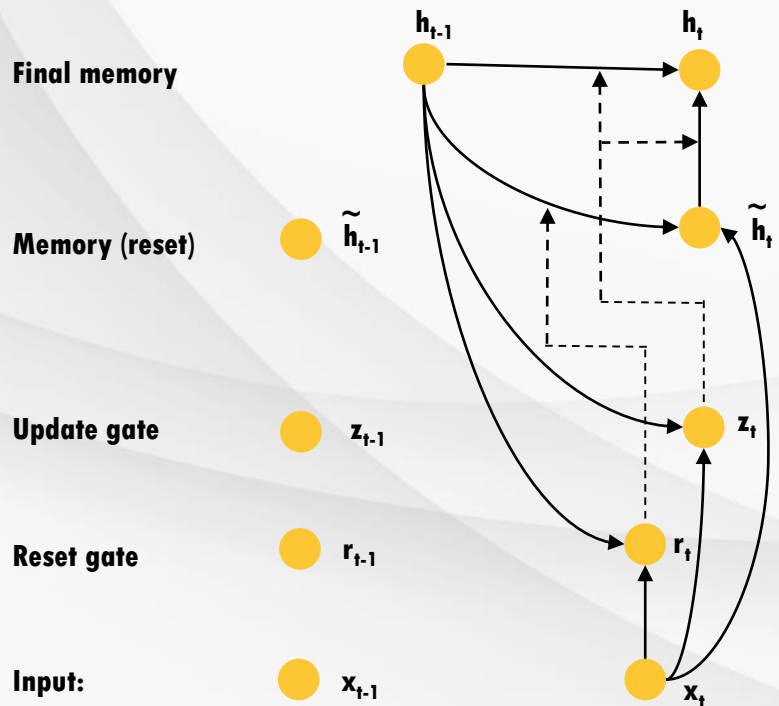
만약 리셋 게이트가 0으로 설정되면, 이전 단계까지 계산한 은닉 유닛을 고려하지 않음,
그리고 현재 시간 단계에서만 계산한 정보만 반영

“잠재적 은닉 유닛” ←
$$h'_t = \tanh(Ux_t + r_t \circ Wh_{t-1})$$

최종 사용할 은닉 유닛은 현재 시간에 계산한 잠재적 은닉 유닛과
이전 시간 단계까지 계산한 은닉 유닛을 업데이트 게이트를 사용하여 조합

$$h_t = z_t \circ h_{t-1} + (1 - z_t) \circ h'_t$$

GRU의 은닉 유닛 계산 과정



$$z_t = \sigma(U^{(z)}x_t + W^{(z)}h_{t-1})$$

$$r_t = \sigma(U^{(r)}x_t + W^{(r)}h_{t-1})$$

$$h'_t = \tanh(Ux_t + r_t \circ Wh_{t-1})$$

$$h_t = z_t \circ h_{t-1} + (1 - z_t) \circ h'_t$$

GRU의 은닉 유닛 계산 과정

- 만약 리셋 게이트(r_t)를 **0**으로 설정하면
잠재적 은닉 유닛 계산 시 이전 시간 단계의
정보를 고려하지 않음
- 업데이트 게이트(z_t)는 최종 은닉 유닛을 계산할 때,
이전 시간의 정보를 얼마큼 반영할지 결정
 - 만약 z_t 가 **1**에 가까울 경우,
이전 시간의 정보를 거의 그대로 복사하게 됨.
(그라디언트 손실이 적어짐)
- 만약 리셋 게이트를 모두 **1**로 설정하고,
업데이트 게이트를 **0**으로 설정하면 일반적인 재현 신경망이 될 수 있음

$$z_t = \sigma(U^{(z)}x_t + W^{(z)}h_{t-1})$$

$$r_t = \sigma(U^{(r)}x_t + W^{(r)}h_{t-1})$$

$$h'_t = \tanh(Ux_t + r_t \circ Wh_{t-1})$$

$$h_t = z_t \circ h_{t-1} + (1 - z_t) \circ h'_t$$

LSTM (Long Short-Term Memory)

- ◉ **LSTM**은 **GRU**보다 게이트가 하나 더 많음(출력 게이트 추가)
- ◉ 최종 은닉 유닛을 계산하는 과정에서 한 단계 더 추가됨
 - 두 가지 보조 메모리가 존재 (잠재 은닉 유닛 g_t , 셀 메모리 c_t)
- ◉ 데이터의 개수가 더 많을 경우 **LSTM**을 사용하면 더 좋은 성능을 나타낼 수 있음

GRU

$$\begin{aligned}z_t &= \sigma(U^{(z)}x_t + W^{(z)}h_{t-1}) \\r_t &= \sigma(U^{(r)}x_t + W^{(r)}h_{t-1}) \\h'_t &= \tanh(Ux_t + r_t \circ Wh_{t-1}) \\h_t &= z_t \circ h_{t-1} + (1 - z_t) \circ h'_t\end{aligned}$$



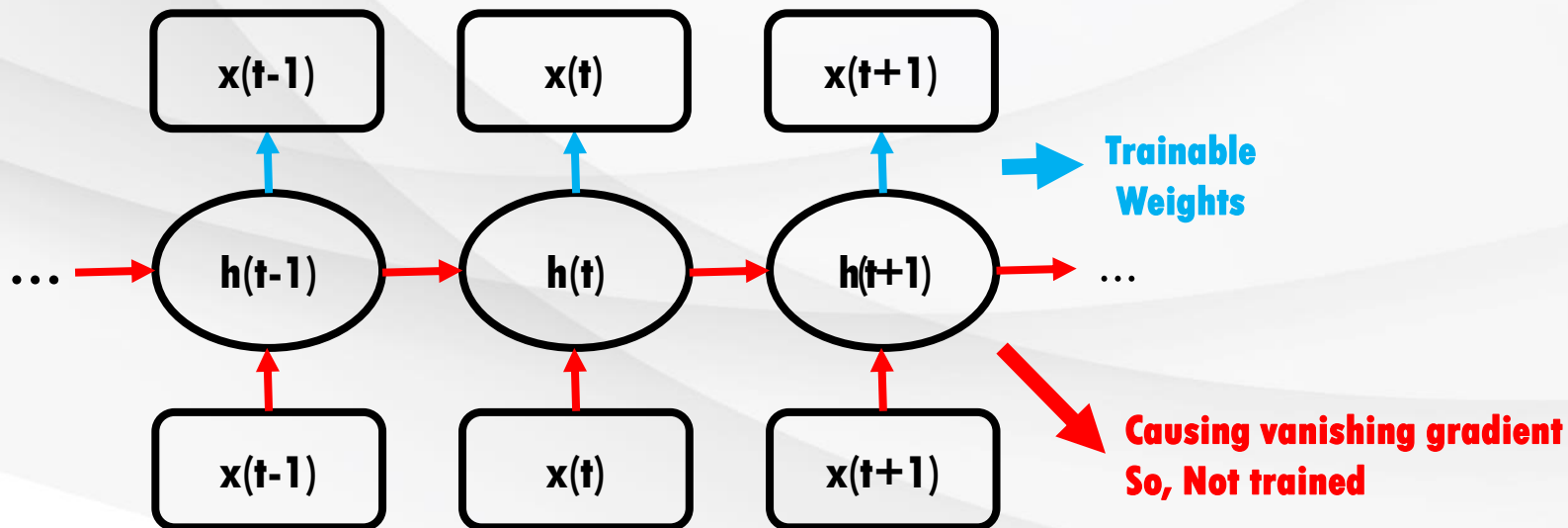
LSTM


$$\begin{aligned}i_t &= \sigma(U^{(i)}x_t + W^{(i)}h_{t-1}) \\f_t &= \sigma(U^{(f)}x_t + W^{(f)}h_{t-1}) \\o_t &= \sigma(U^{(o)}x_t + W^{(o)}h_{t-1}) \\g_t &= \tanh(U^{(g)}x_t + W^{(g)}h_{t-1}) \\c_t &= c_{t-1} \circ f_t + g_t \circ i_t \\s_t &= \tanh(c_t) \circ o_t\end{aligned}$$

Candidate hidden state
Input gate
Output gate
Forget gate

■ ESN (Echo State Networks)

- ◉ 파라미터 u, w 는 학습하지 않음
- ◉ 많은 수의 은닉 유닛 개수가 필요하고 유닛들 사이는 매우 섬겨야 함 (Sparse)
- ◉ 길이가 긴 시계열 데이터를 효율적으로 학습할 수 있음



A person's hands are shown holding a smartphone, with the screen glowing. The background is dark with out-of-focus, warm-toned bokeh lights. A semi-transparent dark banner is at the bottom, containing a yellow decorative element and Korean text.

3. 어떻게 기계가 글을 생성할 수 있을까

■ 언어 모델링 문제

- 문제의 목표는 문장 **W**가 나타날 확률 **P(W)**를 계산하는 것
- P(W)**는 체인룰을 사용하여 단어들의 조건부 확률들의 곱으로 나타낼 수 있음

$$P(W) = P(w(1), w(2), w(3), \dots, w(M)) = \prod_{k=1}^{M+1} P(w(k)|w(1), \dots, w(k-1))$$

- 예를 들어, 언어 모델은 다음 문장이 나타날 확률을 계산할 수 있음

$W = \text{He went to buy some chocolate}$

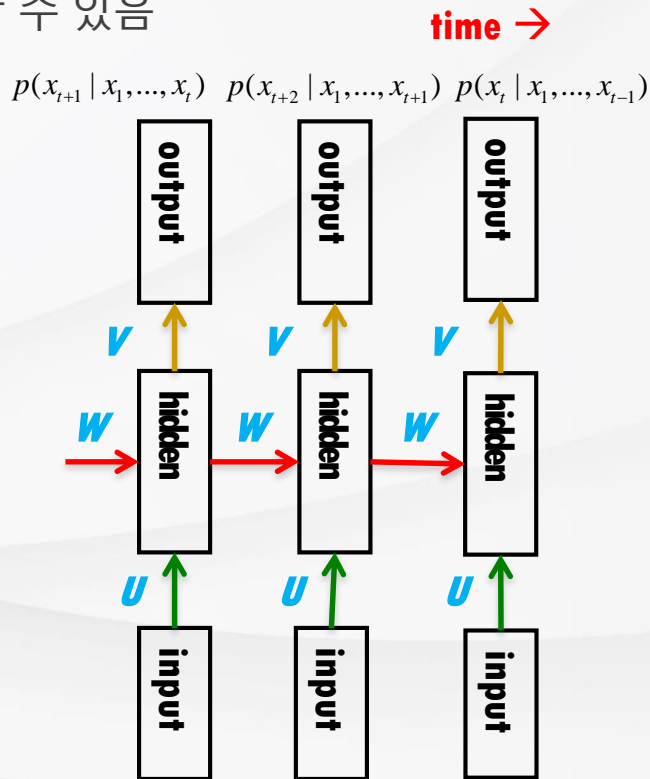
$$\begin{aligned} P(W) &= P(\text{chocolate} \mid \text{He went to buy some}) * P(\text{He went to buy some}) \\ &= P(\text{chocolate} \mid \text{He went to buy some}) * P(\text{some} \mid \text{He went to buy}) * P(\text{He went to buy}) \\ &= \dots \end{aligned}$$

- 이전 단어들의 집합이 컨텍스트로 주어졌을 때,
다음 단어가 등장할 확률의 연속적인 곱셈으로 나타남

언어 모델링 문제

- 재현 신경망을 사용하면 매시간 단위마다 단어를 생성할 수 있음
- t 번째 시간 단위에서 생성된 단어는 t+1 번째 시간 단위의 입력이 됨
- He went to buy some chocolate.**
- 매시간 단위마다 은닉 유닛은 이전 시간까지 생성한 단어들의 정보를 가지고 있음
- 길이가 매우 긴 문장의 경우, 의미 있는 문장을 생성하기 어려움

예> The girl whom I met yesterday is very pretty



■ 생성된 글의 예

PANDARUS:

Alas, I think he shall be come approached and the day When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death, I should not sleep.

Second Senator:

They are away this miseries, produced upon my soul, Breaking and strongly should be buried,
when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:

Well, your wit is in the care of side and that.

Second Lord:

They would be ruled after this chamber, and my fair nues begun out of the fact, to be conveyed, Whose noble
souls I'll have the heart of the wars.

Clown:

Come, sir, I will make did behold your worship.

영어 - 셰익스피어의 글을 대신 생성



학습정리

지금까지 [재현 신경망]에 대해서 살펴보았습니다.

재현 신경망의 원리

재현 신경망은 시계열 데이터를 확률적으로 모델링

학습 가능 파라미터: U, W, V

은닉 유닛의 계산: $s_t = \tanh(Ux_t + Ws_{t-1})$ / 출력 계산: $y_t' = \text{softmax}(Vs_t)$

그라디언트 손실

활성함수의 미분값을 여러 번 곱셈하는 과정에서 발생 $Ws'(z_3)Ws'(z_2)Ws'(z_1)$

활성함수의 미분값: $\tanh(0 \sim 1)$, 시그모이드($0 \sim 0.25$)

프로그래머가 알아차리기 어려워서 그라디언트 폭발보다 더 문제가 됨

언어모델로써 재현 신경망

문장 W 의 확률인 $P(W)$ 는 체인룰을 사용하여 단어들의 조건부 확률들의 곱으로 나타낼 수 있음

$$P(W) = P(w(1), w(2), w(3), \dots, w(M)) = \prod_{k=1}^{M+1} P(w(k) | w(1), \dots, w(k-1))$$

재현 신경망의 t 번째 시간 단위에서 생성된 단어는 $t+1$ 번째 시간 단위의 입력이 됨