# 7강 Recurrent Neural Networks

## Recurrent Hidden Units

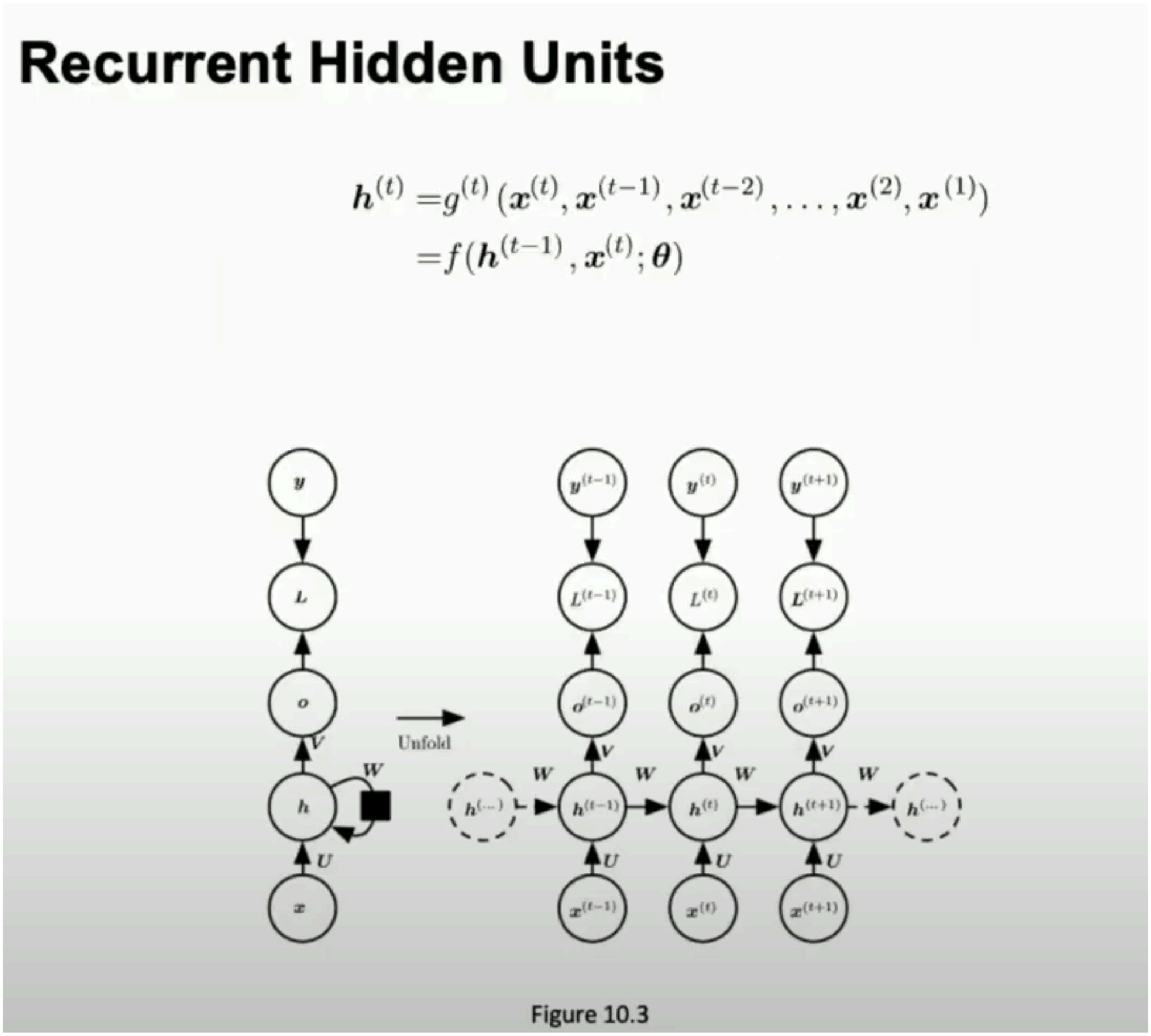



그림 부분의 검은색 네모 = delay
-> 이전 상태를 현재 상태에 전달하는 역할
-> RNN은 이전 시간 단계의 정보를 기억하고 시간적 종속성을 학습할 수 있음
-> 같은 정보가 같은 시간안에 input으로 들어가는 것을 방지

결과적으로 output이 다음 시간대의 input으로 들어감

# RNN

# Plain RNN

- The RNN in Fig. 10.3 is universal in the sense that any function computable by a Turing machine can be computed by such a recurrent network of a finite size.

$$
\begin{aligned}
\boldsymbol{a}^{(t)} &= \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)} \\
\boldsymbol{h}^{(t)} &= \tanh(\boldsymbol{a}^{(t)}) \\
\boldsymbol{o}^{(t)} &= \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)} \\
\hat{\boldsymbol{y}}^{(t)} &= \operatorname{softmax}(\boldsymbol{o}^{(t)})
\end{aligned}
$$

$$
\begin{aligned}
& L\left(\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(\tau)}\}, \{\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(\tau)}\}\right) \\
&= \sum_t L^{(t)} \\
&= -\sum_t \log p_{\text{model}}\left(y^{(t)} \mid \{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(t)}\}\right),
\end{aligned}
$$

**Recurrent Neural Network (RNN) 은 시간에 따라 데이터의 종속 관계를 학습하는 네트워크, 은닉 상태를 사용하여 과거 시간의 정보를 현재 상태로 전달**

## (1) 은닉 상태 업데이트

$$a^{(t)} = b + Wh^{(t-1)} + Ux^{(t)}$$

- $a^{(t)}$: 시간 $t$에서의 **은닉 상태 입력 (pre-activation)**입니다.
- $b$: **편향(bias)** 항입니다.
- $W$: 이전 은닉 상태 $h^{(t-1)}$와 곱해지는 **가중치 행렬**입니다.
- $U$: 현재 입력 $x^{(t)}$와 곱해지는 **가중치 행렬**입니다.
- 이 식은 은닉 상태를 **이전 은닉 상태와 현재 입력**의 선형 결합으로 나타냅니다.

### (2) 은닉 상태 $h^{(t)}$

$$h^{(t)} = \tanh(a^{(t)})$$

- $h^{(t)}$: 시간 $t$에서의 **은닉 상태 (hidden state)**입니다.
- **tanh**: 비선형 활성화 함수로, 값의 범위를 $-1$에서 $1$로 제한합니다.
- 이 비선형성을 통해 RNN은 복잡한 패턴을 학습할 수 있습니다.

### (3) 출력 계산

$$o^{(t)} = c + Vh^{(t)}$$

- $o^{(t)}$: 시간 $t$에서의 **출력 전 상태 (pre-output)**입니다.
- $c$: 출력의 편향(bias)입니다.
- $V$: 은닉 상태 $h^{(t)}$와 곱해지는 **가중치 행렬**입니다.

---

### (4) 최종 출력 $\hat{y}^{(t)}$

$$\hat{y}^{(t)} = \mathrm{softmax}(o^{(t)})$$

- $\hat{y}^{(t)}$: 시간 $t$에서의 **예측 출력**입니다.
- **softmax** 함수는 출력을 확률 분포로 변환하며, 이를 통해 각 클래스에 대한 예측 확률을 구할 수 있습니다.

# 3. 손실 함수 $L$

## (1) 손실 함수 정의

$$L\left(\{x^{(1)},\ldots,x^{(T)}\},\{y^{(1)},\ldots,y^{(T)}\}\right)=\sum_t L^{(t)}$$

- $L$: 전체 시퀀스에 대한 손실 함수입니다.
- $L^{(t)}$: 시간 $t$에서의 개별 손실입니다.

---

## (2) 로그 우도 기반 손실

$$L=-\sum_t \log p_{\text{model}}\left(y^{(t)}\mid\{x^{(1)},\ldots,x^{(t)}\}\right)$$

- 로그 우도 손실: 모델이 예측한 확률 분포 $p_{\text{model}}$와 실제 정답 $y^{(t)}$ 사이의 차이를 측정합니다.
- RNN은 시퀀스 전체에서 각 시간 $t$의 예측 손실을 누적합니다.

## 4. Plain RNN의 특징

- **시간 의존성**: 이전 시간 단계의 은닉 상태 $h^{(t-1)}$가 현재 시간 단계 $h^{(t)}$의 계산에 영향을 미칩니다.
- **은닉 상태 공유**: 모든 시간 단계에서 **같은 가중치 행렬 $W, U, V$**를 공유합니다.
- **손실 함수**: 로그 우도를 기반으로 하며, 전체 시퀀스의 손실을 합산합니다.

---

## 5. RNN의 한계

1. **Vanishing Gradient Problem (기울기 소실 문제)**
   - 시간 단계가 길어질수록 은닉 상태에 대한 기울기가 점점 작아져 학습이 어려워집니다.
2. **Long-term Dependency**
   - 멀리 떨어진 시간 단계 간의 관계를 학습하는 데 어려움이 있습니다.

## RNN Backpropagation Turough Time(BPTT)

# RNN Backpropagation Through Time (BPTT)

$$\frac{\partial L}{\partial L^{(t)}} = 1.$$

$$(\nabla_{\boldsymbol{o}^{(t)}} L)_i = \frac{\partial L}{\partial o_i^{(t)}} = \frac{\partial L}{\partial L^{(t)}} \frac{\partial L^{(t)}}{\partial o_i^{(t)}} = \hat{y}_i^{(t)} - \mathbf{1}_{i, y^{(t)}}.$$

$$\begin{aligned} \nabla_{\boldsymbol{h}^{(t)}} L &= \left(\frac{\partial \boldsymbol{h}^{(t+1)}}{\partial \boldsymbol{h}^{(t)}}\right)^{\top} (\nabla_{\boldsymbol{h}^{(t+1)}} L) + \left(\frac{\partial \boldsymbol{o}^{(t)}}{\partial \boldsymbol{h}^{(t)}}\right)^{\top} (\nabla_{\boldsymbol{o}^{(t)}} L) \\ &= W^T diag\left(1 - \left(h^{(t+1)}\right)^2\right) \nabla_{h^{(t+1)}} L + V^T \left(\nabla_{o^{(t)}} L\right) \end{aligned}$$

## 1. BPTT란?

- **Backpropagation Through Time (BPTT)**: RNN의 시간 단계 $t$를 따라 **손실 함수의 기울기(gradient)**를 역전파하여 **가중치(W, U, V)**를 업데이트하는 알고리즘입니다.
- 이는 **시간** 축을 따라 펼쳐진 RNN(Unfolded RNN)에서의 Backpropagation입니다.

---

## 2. 수식 설명

### (1) 손실 $L$에 대한 기울기

$$\frac{\partial L}{\partial L^{(t)}} = 1$$

- 시간 $t$에서의 **손실 $L^{(t)}$**에 대한 변화율은 1입니다.

### (2) 출력층 $o^{(t)}$에 대한 기울기

$$(\nabla_{o^{(t)}} L)_i = \frac{\partial L}{\partial o_i^{(t)}} = \frac{\partial L}{\partial L^{(t)}} \frac{\partial L^{(t)}}{\partial o_i^{(t)}} = \hat{y}_i^{(t)} - 1_{i,y^{(t)}}$$

- $o^{(t)}$: 출력층 $t$에서의 **pre-activation** 값입니다.
- $\hat{y}_i^{(t)}$: $i$-번째 클래스에 대한 예측 확률값입니다 (softmax 출력).
- $1_{i,y^{(t)}}$: 실제 클래스 $y^{(t)}$에 대한 **indicator** 함수입니다.
- 결과적으로 출력층에 대한 기울기는 예측값 $\hat{y}_i^{(t)}$와 실제값 사이의 차이입니다.

### (3) 은닉 상태 $h^{(t)}$에 대한 기울기

$$\nabla_{h^{(t)}} L = \left(\frac{\partial h^{(t+1)}}{\partial h^{(t)}}\right)^\top \nabla_{h^{(t+1)}} L + \left(\frac{\partial o^{(t)}}{\partial h^{(t)}}\right)^\top \nabla_{o^{(t)}} L$$

- 은닉 상태 $h^{(t)}$에 대한 기울기는 두 항으로 나뉩니다:
  1. **다음 시간 단계 $t+1$**의 은닉 상태 $h^{(t+1)}$로부터 전파되는 기울기.
  2. 현재 시간 단계 $t$의 출력 $o^{(t)}$로부터 전파되는 기울기.

**(i) 첫 번째 항 (시간 역전파)**

$$\left(\frac{\partial h^{(t+1)}}{\partial h^{(t)}}\right)^\top \nabla_{h^{(t+1)}} L$$

- $\frac{\partial h^{(t+1)}}{\partial h^{(t)}}$: 은닉 상태 간의 전파를 나타내는 항입니다.
  - 이 값은 $W^\top \text{diag}(1-(h^{(t+1)})^2)$로 계산됩니다.
  - $1-(h^{(t+1)})^2$: tanh 활성화 함수의 도함수입니다.

**(ii) 두 번째 항 (출력층 역전파)**

$$\left(\frac{\partial o^{(t)}}{\partial h^{(t)}}\right)^\top \nabla_{o^{(t)}} L$$

- $\frac{\partial o^{(t)}}{\partial h^{(t)}}$: 출력 $o^{(t)}$가 은닉 상태 $h^{(t)}$로부터 어떻게 영향을 받는지를 나타냅니다.
  - 이는 가중치 행렬 $V^\top$입니다.
- $\nabla_{o^{(t)}} L$: 출력층에서의 손실 기울기입니다.

### (4) 결합된 기울기 수식

최종적으로 $h^{(t)}$에 대한 기울기는 다음과 같이 표현됩니다:

$$\nabla_{h^{(t)}} L = W^\top \mathrm{diag}(1 - (h^{(t+1)})^2) \nabla_{h^{(t+1)}} L + V^\top (\nabla_{o^{(t)}} L)$$

- **첫 번째 항:** 다음 시간 단계 $h^{(t+1)}$로부터의 기울기 전파.
- **두 번째 항:** 출력층으로부터의 기울기 전파.

---

## 3. 핵심 개념 요약

- **BPTT (Backpropagation Through Time)**는 RNN의 시간에 따른 은닉 **상태**와 출력의 손실 기울기를 계산합니다.
- 기울기 계산은 두 부분으로 나뉩니다:
    1. **시간 역전파:** 다음 시간 단계 $t+1$에서 전파된 기울기.
    2. **출력 역전파:** 현재 시간 단계 $t$의 출력층에서 전파된 기울기.
- 기울기의 누적 과정이 반복되며, 이는 **시간에 따라 펼쳐진 RNN 구조**에서의 backpropagation과 동일합니다.

## RNN Backpropagation Through Time (BPTT)

$$
\begin{aligned}
\nabla_{\boldsymbol{c}} L &= \sum_t \left( \frac{\partial \boldsymbol{o}^{(t)}}{\partial \boldsymbol{c}} \right)^\top \nabla_{\boldsymbol{o}^{(t)}} L = \sum_t \nabla_{\boldsymbol{o}^{(t)}} L \\
\nabla_{\boldsymbol{b}} L &= \sum_t \left( \frac{\partial \boldsymbol{h}^{(t)}}{\partial \boldsymbol{b}^{(t)}} \right)^\top \nabla_{\boldsymbol{h}^{(t)}} L = \sum_t \mathrm{diag}\left(1 - \left(\boldsymbol{h}^{(t)}\right)^2\right) \nabla_{\boldsymbol{h}^{(t)}} L \\
\nabla_{\boldsymbol{V}} L &= \sum_t \sum_i \left( \frac{\partial L}{\partial o_i^{(t)}} \right) \nabla_{\boldsymbol{V}} o_i^{(t)} = \sum_t \left(\nabla_{\boldsymbol{o}^{(t)}} L\right) \boldsymbol{h}^{(t)^\top} \\
\nabla_{\boldsymbol{W}} L &= \sum_t \sum_i \left( \frac{\partial L}{\partial h_i^{(t)}} \right) \nabla_{\boldsymbol{W}^{(t)}} h_i^{(t)} \\
&= \sum_t \mathrm{diag}\left(1 - \left(\boldsymbol{h}^{(t)}\right)^2\right) \left(\nabla_{\boldsymbol{h}^{(t)}} L\right) \boldsymbol{h}^{(t-1)^\top} \\
\nabla_{\boldsymbol{U}} L &= \sum_t \sum_i \left( \frac{\partial L}{\partial h_i^{(t)}} \right) \nabla_{\boldsymbol{U}^{(t)}} h_i^{(t)} \\
&= \sum_t \mathrm{diag}\left(1 - \left(\boldsymbol{h}^{(t)}\right)^2\right) \left(\nabla_{\boldsymbol{h}^{(t)}} L\right) \boldsymbol{x}^{(t)^\top}
\end{aligned}
$$

- **전체 손실에 대한 그래디언트 ($\nabla L$)**:
  - 시간 스텝 $t$에 따른 손실 $\nabla_{o(t)} L$의 총합으로 표현된다.
- **출력 바이어스($\nabla b L$)**:
  - 출력 바이어스 $b$에 대한 그래디언트는 각 스텝 $t$의 $\nabla_{h(t)} L$의 합으로 계산된다.
- **출력 가중치($\nabla V L$)**:
  - 출력 가중치 $V$에 대한 그래디언트는 다음과 같이 계산된다.
- **재귀 가중치($\nabla W L$)**:
  - 재귀 가중치 $W$는 은닉 상태 간 연결의 가중치이다.
- **입력 가중치($\nabla U L$)**:
  - 입력 가중치 $U$는 입력 $x$와 은닉 상태 $h$를 연결하는 가중치이다.

## Fully Connected Graphical Model

# Fully Connected Graphical Model

$$P(\mathbb{Y}) = P(\mathbf{y}^{(1)}, \dots, \mathbf{y}^{(\tau)}) = \prod_{t=1}^{\tau} P(\mathbf{y}^{(t)} \mid \mathbf{y}^{(t-1)}, \mathbf{y}^{(t-2)}, \dots, \mathbf{y}^{(1)})$$

$$L = \sum_t L^{(t)}$$

$$L^{(t)} = -\log P(\mathrm{y}^{(t)} = y^{(t)} \mid y^{(t-1)}, y^{(t-2)}, \dots, y^{(1)}).$$

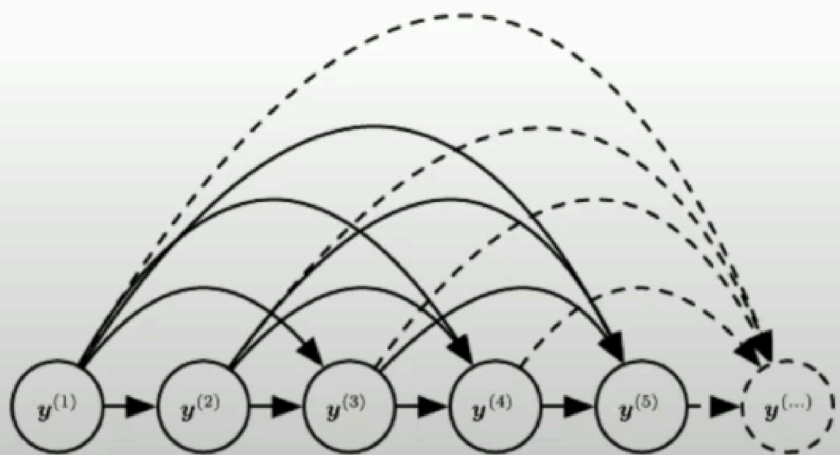



Figure 10.7

- Fully Connected Graphical Model 정의:
  - Fully Connected Graphical Model은 시간 축 $t$를 따라 모든 이전 상태 $y^{(t-1)}, y^{(t-2)}, \dots, y^{(1)}$가 현재 상태 $y^{(t)}$를 예측하는 데 영향을 미치는 구조이다.
  - 확률은 아래와 같이 정의된다: $P(Y) = P(y^{(1)}, \dots, y^{(\tau)}) = \prod_{t=1}^{\tau} P(y^{(t)} \mid y^{(t-1)}, y^{(t-2)}, \dots, y^{(1)})$
- 손실 함수 정의:
  - Fully Connected Graphical Model의 손실 함수는 각 시간 스텝 $t$에서의 손실 $L^{(t)}$의 합으로 표현된다: $L = \sum_t L^{(t)}$
  - 각 시간 스텝 $t$의 손실은 다음과 같이 정의된다: $L^{(t)} = -\log P(y^{(t)} = y^{(t)} \mid y^{(t-1)}, y^{(t-2)}, \dots, y^{(1)})$
- Figure 10.7:
  - 그림에서는 Fully Connected Graphical Model에서 시간 $t$의 현재 출력 $y^{(t)}$가 모든 이전 출력 $y^{(t-1)}, y^{(t-2)}, \dots, y^{(1)}$와 연결되어 있는 구조를 보여준다.
  - 실선과 점선은 모든 이전 시점이 현재 시점에 영향을 미치는 경로를 나타낸다.

## 요약

Fully Connected Graphical Model은 모든 이전 출력 값을 고려해 현재 출력 값을 계산하는 방식으로, 시간적 의존성이 강한 시퀀스 데이터를 모델링하는 데 효과적이다. 이러한 모델은 손실 함수로 로그 가능도를 사용해 모델이 각 시간 스텝에서의 조건부 확률을 최대화하도록 훈련된다.

# Hidden and Output Recurrence

- The RNN of figure 10.3 is only able to represent distributions in which the y values are conditionally independent from each other given the x values.

$$P(\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(\tau)} \mid \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(\tau)}) = \prod_t P(\boldsymbol{y}^{(t)} \mid \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(t)}).$$

- In this Fig. 10.10, the output values are not forced to be conditionally independent in this model. These connections allow this RNN to model an arbitrary distribution over sequences of y given sequences of x of the same length.

$$P(y^{(1)}, \ldots, y^{(\tau)} | x^{(1)}, \ldots, x^{(\tau)}) = \Pi_t P(y^{(t)} | y^{(1)}, \ldots y^{(t-1)}, x^{(1)}, \ldots, x^{(t)})$$

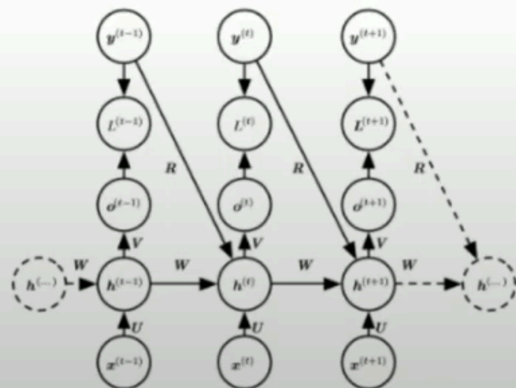
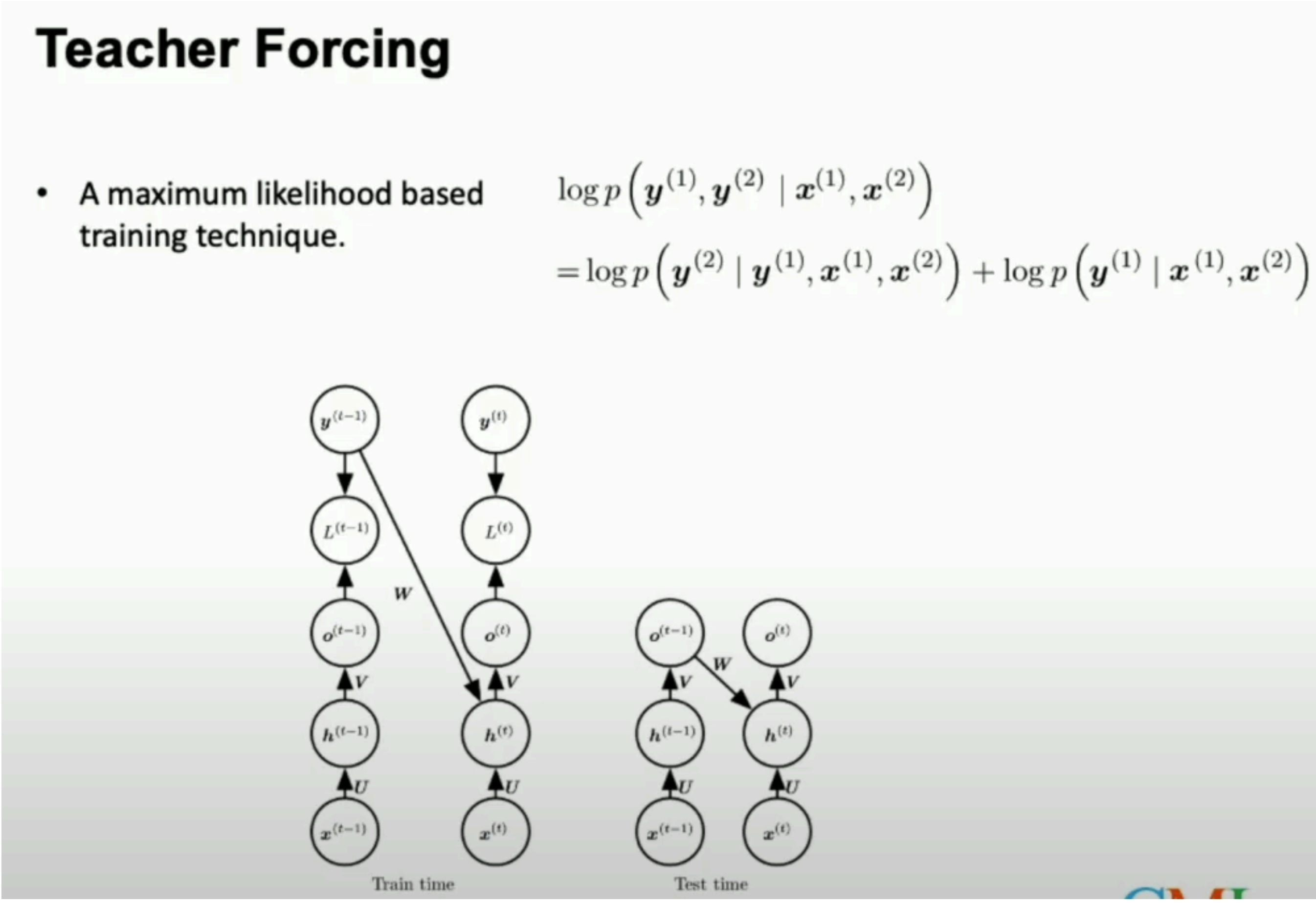



Figure 10.10

- **조건부 독립성(Conditional Independence)**:
  - Figure 10.3의 RNN은 $x$ 값을 알았을 때, $y$ 값들이 서로 **조건부 독립적**인 분포만 표현할 수 있다.
  - 이 경우 확률은 아래와 같이 나타난다: $P(y^{(1)}, \ldots, y^{(\tau)} \mid x^{(1)}, \ldots, x^{(\tau)}) = \prod_t P(y^{(t)} \mid x^{(1)}, \ldots, x^{(t)})$
- **Figure 10.10의 개선**:
  - Figure 10.10에서는 출력 $y$ 값들 간의 조건부 독립성을 강제하지 않는다.
  - 이 연결 구조 덕분에, RNN은 동일 길이의 $x$ 시퀀스가 주어질 때 $y$ 시퀀스의 **임의의 분포**를 모델링할 수 있다.
  - 확률은 다음과 같이 나타난다: $P(y^{(1)}, \ldots, y^{(\tau)} \mid x^{(1)}, \ldots, x^{(\tau)}) = \prod_t P(y^{(t)} \mid y^{(1)}, \ldots, y^{(t-1)}, x^{(1)}, \ldots, x^{(t)})$
- **구조의 차이**:
  - Figure 10.3에서는 $y^{(t)}$가 $x^{(t)}$만 고려하여 독립적으로 결정된다.
  - Figure 10.10에서는 $y^{(t)}$가 이전 출력 값들 $y^{(1)}, \ldots, y^{(t-1)}$ 및 입력 값들 $x^{(1)}, \ldots, x^{(t)}$ 모두를 고려하여 결정된다.

## Teacher Forcing




Teacher Forcing은 시퀀스 예측 모델(RNN, LSTM, Transformer 등)을 훈련할 때 사용하는 최대우도(MLE) 기반 훈련 기법이다.

## 핵심 개념:

이전 스텝의 모델 출력 대신 실제 정답(ground truth) 을 다음 스텝의 입력으로 사용해 훈련을 빠르게 수렴하도록 돕는 방법이다.

## 훈련 vs 테스트:

- 훈련 시: $y^{(t-1)}$ (실제 정답)을 $y^{(t)}$를 예측하는 입력으로 사용한다.
- 테스트 시: $y^{(t-1)}$ (모델 출력)을 $y^{(t)}$를 예측하는 입력으로 사용한다.(정확한 것은 아니지만 GT를 알지 못하므로 대신 사용하는 것)

## 장점:

훈련 초기의 불안정성을 줄이고, 학습을 빠르게 진행할 수 있다.

## 단점:

테스트 시 실제 출력에 의존하기 때문에, 모델이 이전 출력 오류를 적절히 처리하지 못할 가능성이 있다. 이러한 문제를 "exposure bias" 라고 한다.

## 수식:

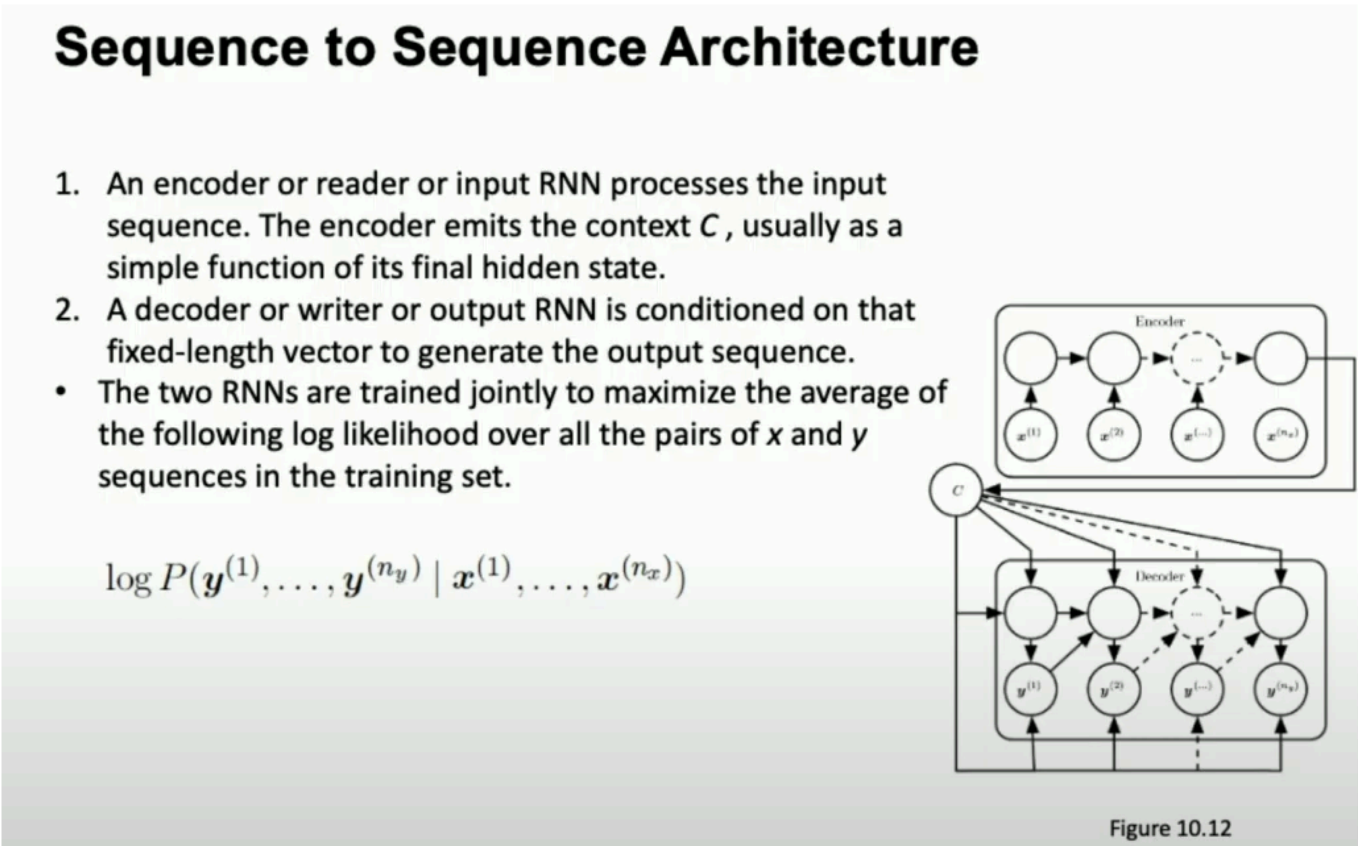
$logp(y^{(1)}, y^{(2)} \mid x^{(1)}, x^{(2)})$ = $\log p(y^{(2)} \mid y^{(1)}, x^{(1)}, x^{(2)}) + \log p(y^{(1)} \mid x^{(1)}, x^{(2)})$

이는 조건부 확률을 최대화하는 방식으로 훈련된다.

### 결론:

Teacher Forcing은 모델이 빠르게 학습하도록 돕는 유용한 방법이지만, 테스트 환경과 훈련 환경이 달라지는 문제가 발생할 수 있다. 이를 보완하기 위해 Scheduled Sampling과 같은 기법이 활용되기도 한다.

## Sequence to Sequence Architecture

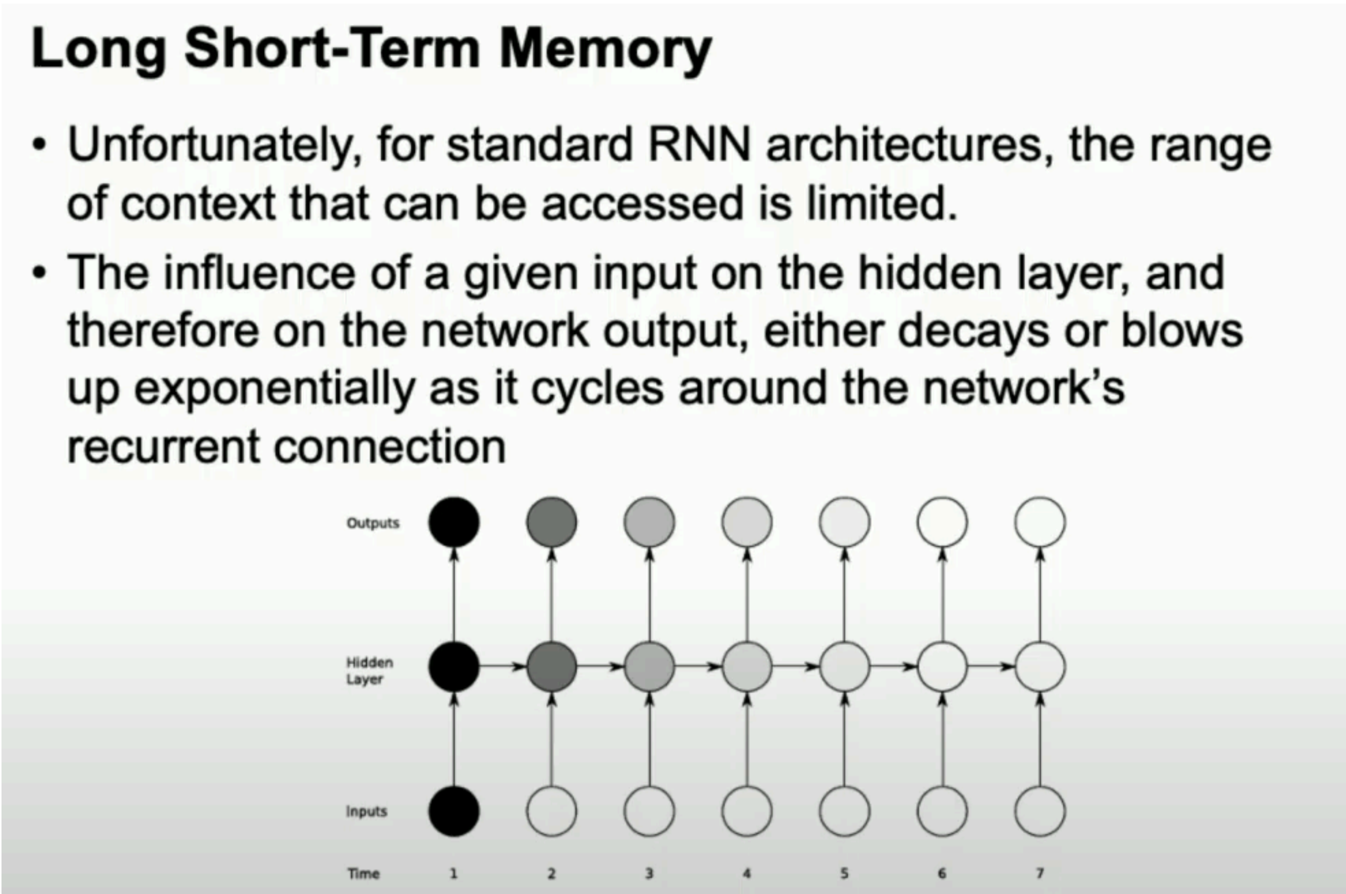



- **Encoder (입력 RNN)**:
  - Encoder는 입력 시퀀스 $x^{(1)}, \ldots, x^{(n_x)}$를 처리한다.
  - 입력 시퀀스를 처리한 후, 마지막 은닉 상태를 바탕으로 고정 길이의 컨텍스트 벡터 $C$ 를 생성한다.
  - $C$는 보통 마지막 은닉 상태 $h^{(n_x)}$의 간단한 함수로 표현된다.
- **Decoder (출력 RNN)**:
  - Decoder는 Encoder로부터 전달받은 컨텍스트 벡터 $C$를 기반으로 출력 시퀀스 $y^{(1)}, \ldots, y^{(n_y)}$를 생성한다.
  - Decoder는 생성된 이전 출력 값 $y^{(t-1)}$와 $C$를 사용하여 $y^{(t)}$를 예측한다.
- **목표 (Log-Likelihood 최대화)**:
  - Encoder와 Decoder는 $x, y$ 시퀀스 쌍의 전체 데이터셋에서 평균 로그 가능도를 최대화하도록 공동으로 훈련된다.
  - 로그 가능도는 아래와 같이 정의된다: $\log P(y^{(1)}, \ldots, y^{(n_y)} \mid x^{(1)}, \ldots, x^{(n_x)})$
- 그림은 Seq2Seq 구조에서 Encoder와 Decoder의 연결 방식을 보여준다.
- **Encoder**는 입력 $x^{(1)}, \ldots, x^{(n_x)}$를 처리하며, 컨텍스트 벡터 $C$를 생성한다.
- **Decoder**는 $C$를 기반으로 출력 $y^{(1)}, \ldots, y^{(n_y)}$를 생성하며, 이전 출력 값과의 의존성을 포함한다.

Seq2Seq 아키텍처는 입력과 출력 시퀀스 간의 관계를 학습하는 방식. 모델은 입력 시퀀스를 압축해 고정된 컨텍스트 벡터로 표현한 뒤, 이를 바탕으로 출력 시퀀스를 생성하는 방식으로 작동

C = context를 의미함

# Long Short-Term Memory




## RNN의 근본적인 문제

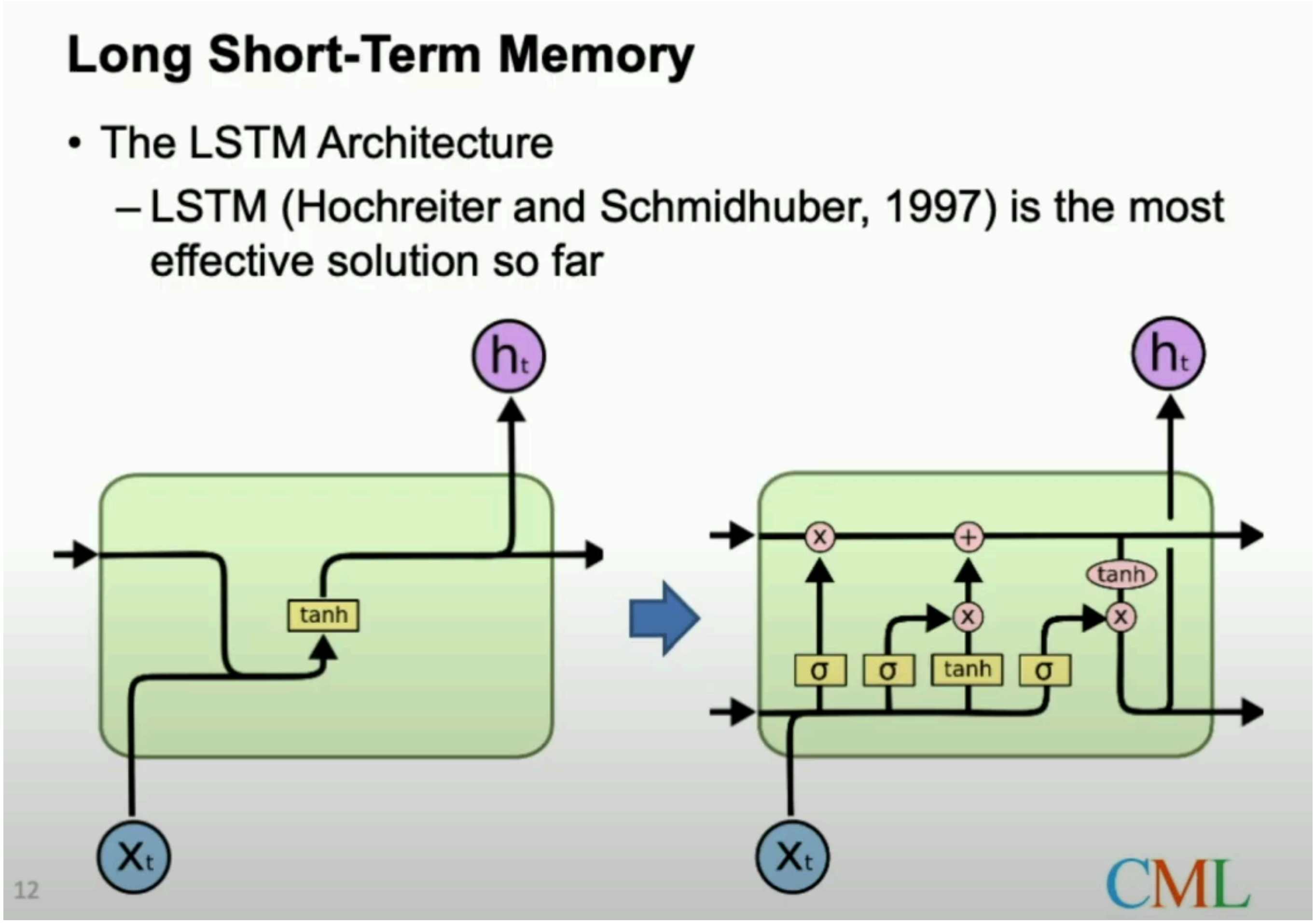
점화식 구조이기 때문에 발산 혹은 소멸해버리는 경우

예시) 등비수열 구조라면
r이 1보다 작은 경우 -> 소멸
r이 1보다 큰 경우 -> 발산

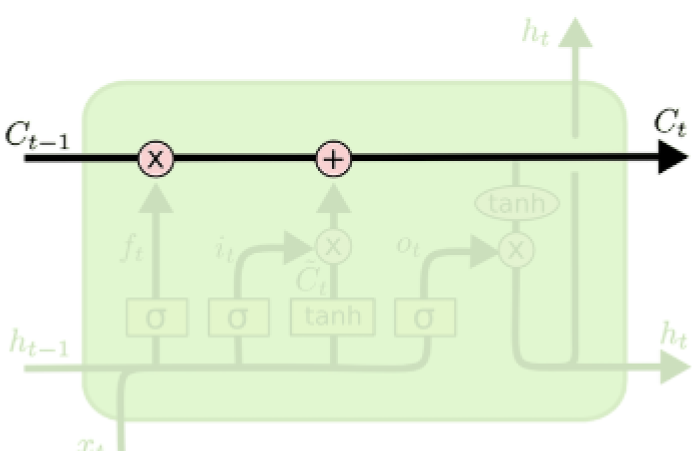
행렬 W가 계속 곱해지는 r의 역할을 함
따라서 W의 eigenvalue, singuler value가 1이어야 한다

## LSTM

https://dgkim5360.tistory.com/entry/understanding-long-short-term-memory-lstm-kr




이 구조에선 Skip connection 개념이 도입됨
플러스(+) 기호가 있는 곳에는 Skip connection 개념이 들어있음
플러스 기호가 있기 때문에 gradient가 제한 없이 넘어가 전달될 수 있음(backpropagation highway라고 생각하면 됨)
LSTM에서는 hidden vector 역할을 하는 곳이 두 개가 있음(C, h)




LSTM의 cell state

## LSTM 구조

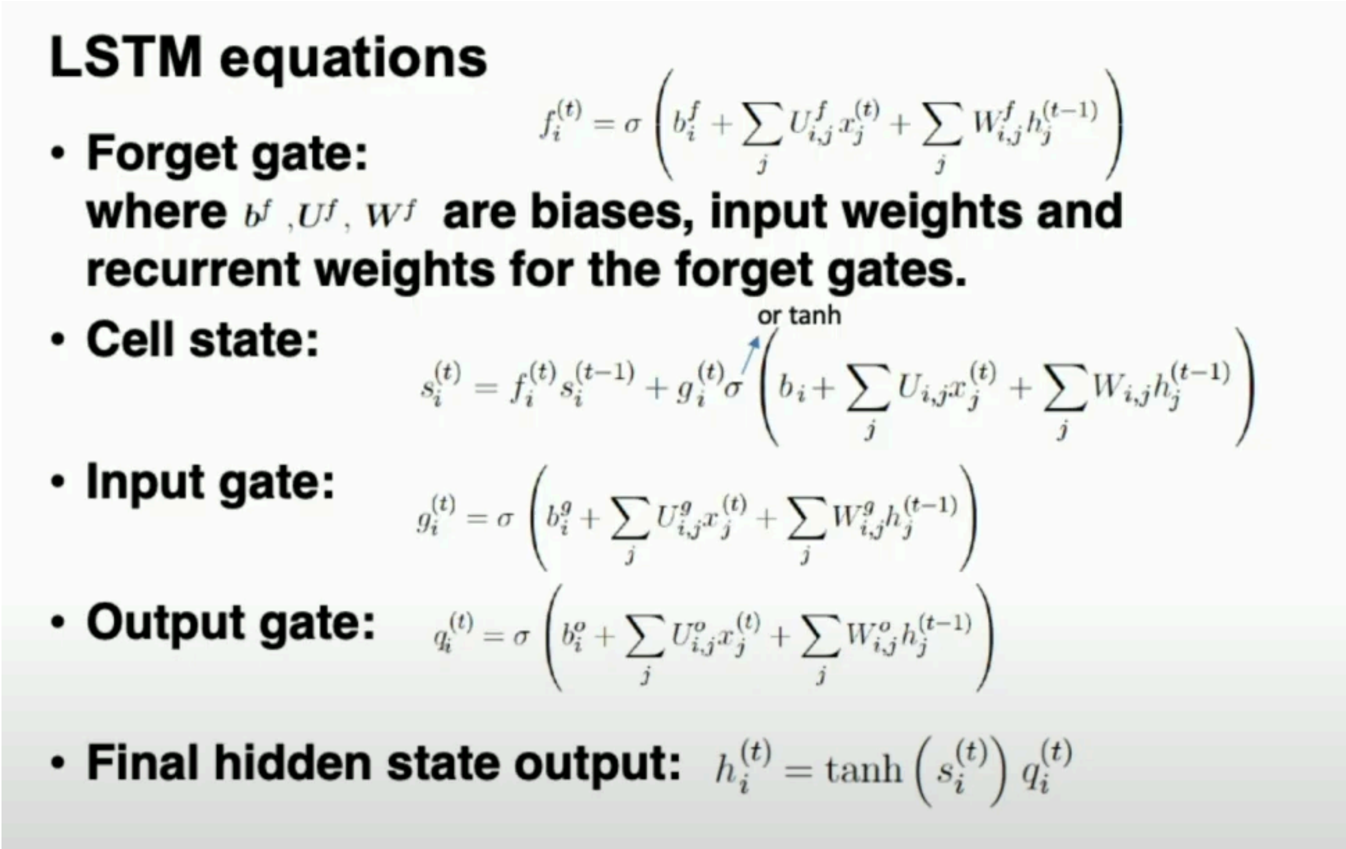
1. **입력과 출력**:
   - 입력: 현재 입력값 $x_t$와 이전 시간 스텝의 은닉 상태 $h_{t-1}$.

- 출력: 현재 시간 스텝의 은닉 상태 $h_t$와 셀 상태 $C_t$.

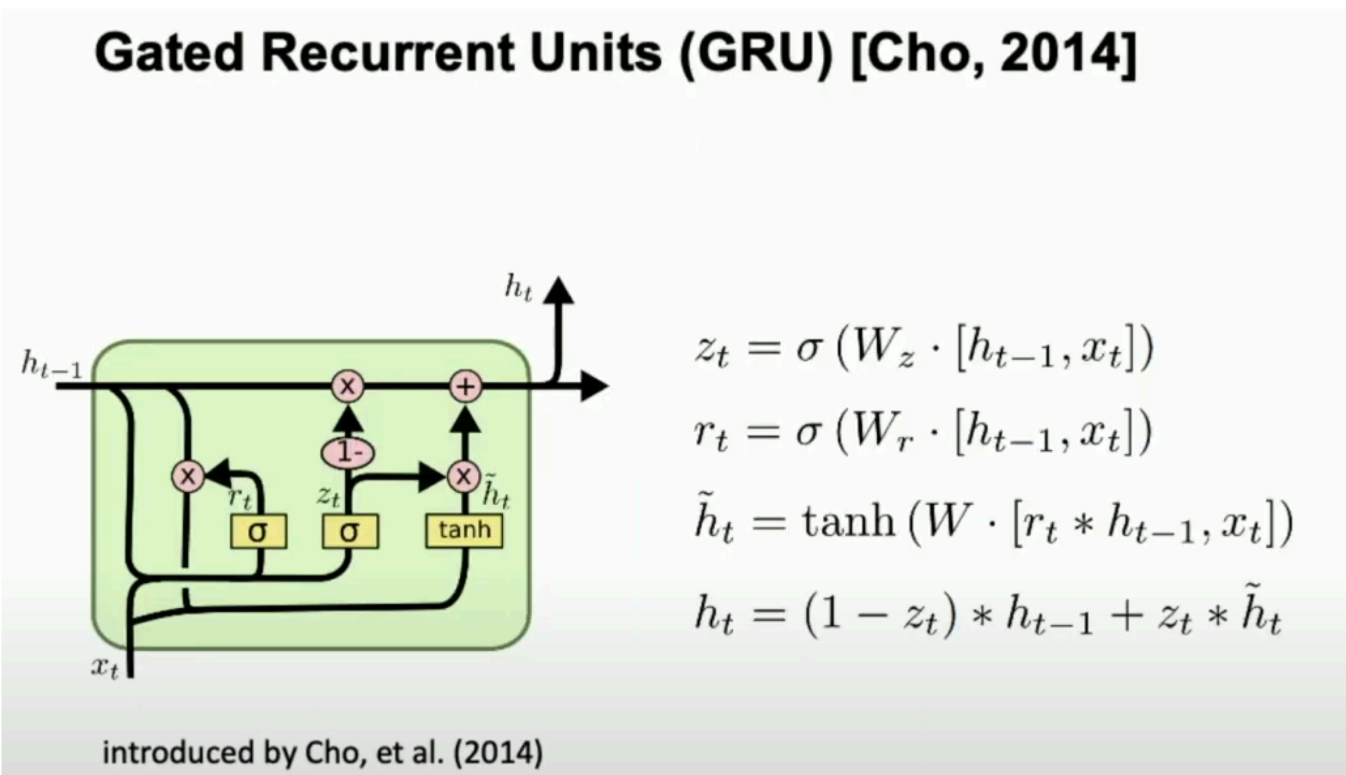
2. **주요 구성 요소**:
   - **Forget Gate**:
     - 현재 입력 $x_t$와 이전 은닉 상태 $h_{t-1}$을 사용해 셀 상태 $C_{t-1}$의 어느 부분을 유지할지 결정.
     - $\sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$
   - **Input Gate**:
     - 새로운 정보를 저장하기 위해 현재 입력 $x_t$와 이전 은닉 상태 $h_{t-1}$을 사용해 업데이트할 값을 결정.
     - $\sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$
   - **Candidate Cell State**:
     - 입력 게이트의 결정에 따라 셀 상태를 업데이트하기 위해 새로운 후보 값 생성.
     - $\tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$
   - **Cell State Update**:
     - Forget Gate와 Input Gate의 결과를 조합해 새로운 셀 상태 $C_t$ 계산.
     - $C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$
   - **Output Gate**:
     - 현재 시간 스텝의 출력 $h_t$를 생성하기 위해 현재 셀 상태 $C_t$를 기반으로 결정.
     - $\sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$
   - **Hidden State Update**:
     - $\tanh(C_t)$를 통해 활성화된 값을 Output Gate 결과와 조합해 은닉 상태 $h_t$ 계산.
     - $h_t = o_t \odot \tanh(C_t)$

## LSTM의 장점

1. **장기 의존성 해결**:
   - cell state를 통해 $C_t$를 사용해 정보가 시간 축을 따라 사라지지 않고 전달된다.
2. **효율적인 정보 업데이트**:
   - 게이트 메커니즘을 통해 필요하지 않은 정보는 제거하고 중요한 정보만 유지.




## GRU




앞서 LSTM에서 hidden vector가 두 개 (C, h)로 존재하는 것을 하나로 합친 모델
cell state가 사라지고 hidden vector가 하나만 남아있다

z 라는 파라미터를 기준으로 내분과 유사한 계산 과정을 수행
(1-z) : 이전의 정보를 보존하는 비율

z : 현재 정보를 보존하는 비율

여기서도 플러스 부분이 Skip connection 의 역할을 수행한다

## Gradient Clipping

### Optimization for Long-Term Dependencies

- **Gradient Clipping**
- **Regularization approach**
  - Add a regularization term to a loss function [Pascanu *et al.*, 2013]

$$\Omega = \sum_t \left( \frac{\left| \left| (\nabla_{\boldsymbol{h}^{(t)}} L) \frac{\partial \boldsymbol{h}^{(t)}}{\partial \boldsymbol{h}^{(t-1)}} \right| \right|}{||\nabla_{\boldsymbol{h}^{(t)}} L||} - 1 \right)^2$$

### RNN Backpropagation Through Time (BPTT)

$$\frac{\partial L}{\partial L^{(t)}} = 1.$$

$$(\nabla_{\boldsymbol{o}^{(t)}} L)_i = \frac{\partial L}{\partial o_i^{(t)}} = \frac{\partial L}{\partial L^{(t)}} \frac{\partial L^{(t)}}{\partial o_i^{(t)}} = \hat{y}_i^{(t)} - \mathbf{1}_{i,y^{(t)}}.$$

$$\nabla_{\boldsymbol{h}^{(t)}} L = \left( \frac{\partial \boldsymbol{h}^{(t+1)}}{\partial \boldsymbol{h}^{(t)}} \right)^\top (\nabla_{\boldsymbol{h}^{(t+1)}} L) + \left( \frac{\partial \boldsymbol{o}^{(t)}}{\partial \boldsymbol{h}^{(t)}} \right)^\top (\nabla_{\boldsymbol{o}^{(t)}} L)$$

$$= W^T diag\left(1 - \left(h^{(t+1)}\right)^2\right) \nabla_{h^{(t+1)}} L + V^T \left(\nabla_{o^{(t)}} L\right)$$

위 그림에서 $W^T$ 는 지속적으로 곱해지면 등비수열의 r 역할을 수행하고 있다
N의 값이 크면 곱해지는 횟수가 많아지는데, 이 과정에서 gradient가 소멸, 폭발할 위험이 있다.
이 상황에서 gradient clipping을 활용해 gradient를 최대한 1에 가깝게(정확히는 일정 임계값 내로 그래디언트의 최대 크기를 제한한다) 클리핑한다.
W의 singular value 들이 거의 1에 가깝도록 유지해야한다.

## Singular value 관련 내용 참고

Singular Value(특이값) 는 행렬의 중요한 속성을 나타내는 값으로, 행렬의 선형 변환 특성을 분석하는 데 사용된다.
Singular Value는 Singular Value Decomposition(SVD) 의 결과로 얻어진다.

---

## 1. Singular Value Decomposition(SVD)

주어진 $m \times n$ 행렬 $A$에 대해, SVD는 다음과 같이 표현된다:
$A = U\Sigma V^T$

- $U$ : $m \times m$ 직교 행렬 (왼쪽 특이벡터)
- $\Sigma$ : $m \times n$ 대각 행렬 (대각선 요소가 특이값)
- $V^T$ : $n \times n$ 직교 행렬 (오른쪽 특이벡터)
  $\Sigma$의 대각선에 위치한 값들이 $A$의 Singular Value(특이값) 이다.

## 2. Singular Value의 의미

- 선형 변환의 크기:
  - Singular Value는 행렬 $A$가 특정 방향의 벡터를 얼마나 늘리거나 줄이는지를 나타낸다.
  - $\sigma_i$ (특이값)는 $A$에 의해 변환된 벡터의 스케일링 크기를 나타낸다.
- 행렬의 안정성:
  - 큰 특이값은 변환 과정에서 값이 폭발할 가능성을, 작은 특이값은 정보 손실(소멸)을 초래할 수 있다.

## 3. 특이값과 RNN/LSTM의 관계

- $W$ 행렬의 특이값:
  - RNN에서는 반복적으로 $W$ 행렬을 곱하기 때문에 $W$의 특이값이 모델의 안정성에 큰 영향을 미친다.
  - 특이값이 1보다 크면 그래디언트가 폭발(Exploding Gradient)하고, 1보다 작으면 그래디언트가 소멸(Vanishing Gradient)한다.
- 특이값 안정화:
  - $W$의 Singular Value를 1에 가깝게 유지하면 그래디언트가 소멸하거나 폭발하지 않고 안정적으로 유지된다.

## 4. 특이값을 제어하는 방법

1. Orthogonal Initialization:
   - $W$를 직교 행렬로 초기화하면 모든 Singular Value가 1로 설정된다.
2. Spectral Normalization:
   - $W$의 가장 큰 특이값 $\sigma_{\max}$를 1로 조정해 안정성을 높인다.
3. Regularization:
   - SVD를 통해 계산한 특이값들에 규제를 가해 특정 범위 내에서 유지하도록 한다.

## 5. 특이값의 활용

특이값은 선형 변환, 차원 축소(PCA), 이미지 압축 등 다양한 응용에서 활용된다. 특히 머신러닝에서 행렬의 안정성과 효율성을 보장하는 데 중요한 역할을 한다.