7장 | 시계열 분석

WEEK	10주차

7.1 시계열 문제

시계열 분석이란?

시간에 따라 변하는 데이터를 사용하여 추이를 분석하는 것
추세를 파악하거나 향후 전망 등을 예측하기 위한 용도로 사용
e.g.) 주가/환율 변동 및 기온/습도 변화 등

시계열 형태(the components of time-series) 데이터 변동 유형에 따른 구분

- **불규칙 변동(irregular variation)**
  시계열 자료에서 시간에 따른 규칙적인 움직임과 달리 어떤 규칙성이 없어 예측 불가능하고 우연적으로 발생하는 변동
- **추세 변동(trend variation)**
  시계열 자료가 갖는 장기적인 변화 추세
  추세: 장기간에 걸쳐 지속적으로 증가, 감소하거나 또는 일정한 상태(stationary)를 유지하려는 성향
  → 짧은 기간 동안에는 추세 변동을 찾기 어려운 단점
- **순환 변동(cyclical variation)**
  대체로 2~3년 정도의 일정한 기간을 주기로 순환적으로 나타나는 변동
  1년 이내 주기로 곡선을 그리며 추세 변동에 따라 변동
- **계절 변동(seasonal variation)**
  시계열 자료에서 보통 계절적 영향과 사회적 관습에 따라 1년 주기로 발생하는 것을 의미

→ 시계열 데이터는 결국 **규칙적 시계열**과 **불규칙적 시계열**로 나눌 수 있음

규칙적 시계열 - 트렌드와 분산이 불변하는 데이터

불규칙적 시계열 - 트렌드 혹은 분산이 변화하는 시계열 데이터

시계열 데이터를 잘 분석한다 = 불규칙성을 갖는 시계열 데이터에 특정한 기법이나 모델을 적용하여 규칙적 패턴을 찾거나 예측하는 것

불규칙적 시계열 데이터에 규칙성을 부여하는 방법: AR, MA, ARMA, ARIMA 모델 적용
△ 최근에는 딥러닝을 이용하여 시계열 데이터의 연속성을 기계 스스로 찾아내도록 하는 방법이 더 좋은 성능을 내고 있음

7.2 AR, MA, ARMA, ARIMA

시계열 분석 특징

: 독립 변수(independent variable)를 사용하여 종속 변수(dependent variable)를 예측하는 일반적인 머신 러닝에서 **시간**을 **독립 변수**로 사용
→ 분석하는 데 있어 일반적인 방법론들과 차이

7.2.1 AR 모델

AR(AutoRegressive)(자기 회귀) 모델
: 이전 관측 값이 관측 값에 영향을 준다는 아이디어에 대한 모형

$$\underset{①}{Z_t} = \underset{②}{\Phi_1 Z_{t-1} + \Phi_2 Z_{t-2} + \cdots + \Phi_p Z_{t-p}} + \underset{③}{a_t}$$

1) 시계열 데이터에서 현재 시점을 의미
2) 과거가 현재에 미치는 영향을 나타내는 모수에 시계열 데이터의 과거 시점을 곱한 것
3) 오차 항 (= 백색 잡음)

→ 수식은 p 시점을 기준으로 그 이전의 데이터에 의해 현재 시점의 데이터가 영향을 받는 모형

7.2.2 MA 모델

MA(Moving Average)(이동 평균) 모델
: 트렌드(평균 혹은 시계열 그래프에서 y값)가 변화하는 상황에 적합한 회귀 모델

'윈도우' 개념 사용 - 시계열에 따라 윈도우 크기만큼 슬라이딩(moving)된다고 하여 이동 평균 모델이라고 함

$$\underset{①}{Z_t} = \underbrace{\theta_1 a_{t-1} + \theta_2 a_{t-2} + \cdots + \theta_p a_{t-p}}_{②} + \underset{③}{a_t}$$

1) 시계열 데이터에서 현재 시점을 의미
2) 매개변수에서 과거 시점의 오차를 곱한 것
3) 오차 항

→ 수식은 AR 모델처럼 이전 데이터의 '상태'에서 현재 데이터의 상태를 추론하는 것이 아닌, 이전 데이터의 오차에서 현재 데이터의 상태를 추론하겠다는 의미

7.2.3 ARMA 모델

ARMA(AutoRegressive Moving Average)(자기 회귀 이동 평균) 모델
: AR, MA 두 가지 관점에서 과거의 데이터를 사용하는 모델로 연구 기관에서 주로 사용

$$Z_t = a + \Phi_1 Z_{t-1} + \cdots + \Phi_p Z_{t-p} + \theta_1 a_{t-1} + \cdots + \theta_q a_{t-q} + a_t$$

7.2.4 ARIMA 모델

ARIMA(AutoRegressive Integrated Moving Average)(자기 회귀 누적 이동 평균) 모델
: 자기 회귀와 이동 평균을 둘 다 고려하는 모형 △ ARMA와 달리 과거 데이터의 선형 관계뿐만 아니라 추세(cointegration)까지 고려한 모델

statsmodels 라이브러리를 이용한 ARIMA 모델 구현 절차

1. ARIMA() 함수를 호출하여 사용하는데, ARIMA(p,d,q) 함수에서 쓰는 파라미터는 다음과 같음
   - p: 자기 회귀 차수
   - d: 차분 차수
   - q: 이동 평균 차수
2. fit() 메서드를 호출하고 모델에 데이터를 적용하여 훈련시킴
3. predict() 메서드를 호출하여 미래의 추세 및 동향에 대해 예측

NOTE) statsmodels 라이브러리

statsmodels는 다음 통계 분석 기능을 제공하는 파이썬 패키지

- 검정 및 추정(test and estimation)
- 회귀 분석(regression analysis)
- 시계열 분석(time-series analysis)

파이썬에서 사용하려면 pip install statsmodels 명령으로 사전 설치 작업 필요

7.3 순환 신경망(RNN)

RNN(Recurrent Neural Network)
: 시간적으로 연속성이 있는 데이터를 처리하려고 고안된 인공 신경망
'Reccurent(반복되는)' : 이전 은닉층이 현재 은닉청의 입력이 되면서 '반복되는 순환 구조를 갖는다'는 의미

RNN이 기존 네트워크와 다른 점 - '**기억(memory)**'을 갖는다는 것
기억: 현재까지 입력 데이터를 요약한 정보
→ 새로운 입력이 네트워크로 들어올 때마다 기억은 조금씩 수정됨
→ 최종적으로 남겨진 기억은 모든 입력 전체를 요약한 정보가 됨

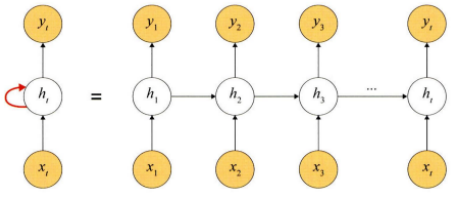


그림 7-4 순환 신경망(RNN)

첫 번째 입력(x_1)이 들어오면 첫 번째 기억(h_1)이 만들어지고, 두 번째 입력(x_2)이 들어오면 기존 기억(h_1)과 새로운 입력을 참고하여 새 기억(h_2)을 만듦 → 입력 길이만큼 과정 반복

= RNN은 외부 입력과 자신의 이전 상태를 입력 받아 현재 상태를 갱신

입력과 출력에 따른 RNN 유형

1. **일대일**

   순환이 없기 때문에 RNN이라고 말하기 어려움
   e.g.) 순방향 네트워크

2. **일대다**

   입력이 하나이고, 출력이 다수인 구조
   e.g.) 이미지 캡션(image captioning): 이미지를 입력해서 이미지에 대한 설명을 문장으로 출력

3. **다대일**

   입력이 다수이고, 출력이 하나인 구조
   e.g.) 감성 분석기: 문장을 입력해서 긍정/부정 출력

```
self.em = nn.Embedding(len(TEXT.vocab.stoi), embedding_dim) # 임베딩 처리
self.rnn = nn.RNNCell(input_dim, hidden_size) # RNN 적용
self.fc1 = nn.Linear(hidden_size, 256) # 완전연결층
self.fc2 = nn.Linear(256, 3) # 출력층
```

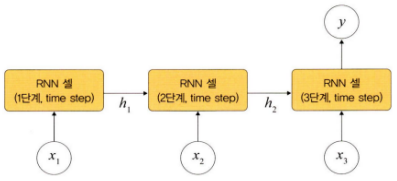


그림 7-5 다대일 모델

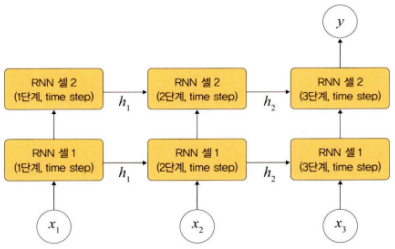


그림 7-6 적층된 다대일 모델

4. **다대다**

   입력과 출력이 다수인 구조
   e.g.) 자동 번역기

   - 텐서플로 구현

     `keras.layers.SimpleRNN(100, return_sequences=True, name='RNN')`
     →
     `return_sequences = True` 옵션으로 시퀀스를 리턴하도록 쉽게 구현 가능

   - △ 파이토치 구현

     문장 번역에서 많이 사용되는 시퀀스-투-시퀀스(seq2seq)를 이용하는 방식으로 사용됨

```
Seq2Seq(
    (encoder): Encoder(
        (embedding): Embedding(7855, 256)
        (rnn): LSTM(256, 512, num_layers=2, dropout=0.5)
        (dropout): Dropout(p=0.5, inplace=False)
    )
    (decoder): Decoder(
        (embedding): Embedding(5893, 256)
        (rnn): LSTM(256, 512, num_layers=2, dropout=0.5)
        (fc_out): Linear(in_features=512, out_features=5893, bias=True)
        (dropout): Dropout(p=0.5, inplace=False)
    )
```

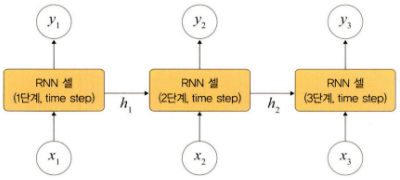


그림 7-7 다대다 모델

5. **동기화 다대다**

   4 처럼 입력과 출력이 다수인 구조
   e.g.) 문장에서 다음에 나올 단어를 예측하는 언어 모델, 프레임 수준의 비디오 분류 등

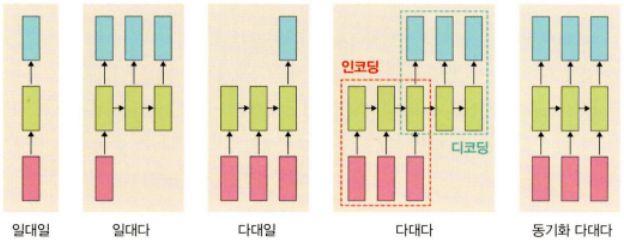


그림 7-8 RNN 모델 유형

7.3.1 RNN 계층과 셀

RNN의 구성 - RNN 계층(layer), RNN 셀(cell)

RNN은 내장된(built-in) 계층, 셀 레벨의 API 제공
- RNN 계층: 입력된 배치 순서대로 모두 처리
- RNN 셀: 오직 하나의 단계 처리
→ RNN 셀은 RNN 계층의 for loop 구문을 갖는 구조

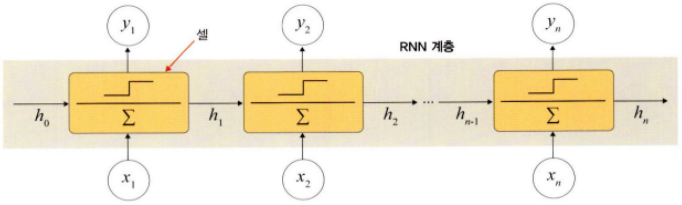


그림 7-9 RNN 계층과 RNN 셀

RNN 계층은 셀을 래핑하여 동일한 셀을 여러 단계에 적용
그림 7-9에서도 X1, X2, ..., Xn 등이 전체 RNN 셀에서 사용되고 있음
= 셀은 실제 계산에 사용되는 RNN 계층의 구성 요소로, 단일 입력과 과거 상태를 가져와서 출력과 새로운 상태를 생성

셀 유형

- **nn.RNNCell**: SimpleRNN 계층에 대응되는 RNN 셀
- **nn.GRUCell**: GRU 계층에 대응되는 GRU 셀
- **nn.LSTMCell**: LSTM 계층에 대응되는 LSTM 셀

RNN의 계층과 셀을 분리해서 설명하는 이유
: 파이토치에서 이 둘을 분리해서 구현이 가능하기 때문

RNN 활용 분야

- **자연어 처리** : 연속적 단어들의 나열인 언어(자연어) 처리는 음성 인식, 단어 의미 판단 및 대화 등에 대한 처리 가능
- 손글씨, 센서 데이터 등 시계열 데이터 처리

7.4 RNN 구조

RNN - 은닉층 노드들이 연결되어 이전 단계 정보를 은닉층 노드에 저장할 수 있도록 구성한 신경망

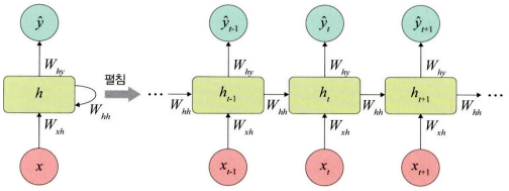


그림 7-10 RNN 구조

x_{t-1}에서 h_{t-1}을 얻음
→ 다음 단계에서
h_{t-1}과 x_t 사용해서 과거 정보와 현재 정보를 모두 반영
→
h_t와 x_{t+1}의 정보 이용해서 과거와 현재 정보를 반복해서 반영

RNN의 가중치

- W_{xh} : 입력층 → 은닉층
- W_{hh} : t 시점의 은닉층 → t+1 시점의 은닉층
- W_{hy} : 은닉층 → 출력층

주의) 세 가중치가 모든 시점에 동일 = 가중치를 공유

t단계에서의 RNN 계산

1. **은닉층** 계산을 위해 x_t, h_{t-1} 필요
   = (이전 은닉층 x 은닉층 → 은닉층 가중치 + 입력층 → 은닉층 가중치 x (현재) 입력 값)

   RNN에서 은닉층은 일반적으로 하이퍼볼릭 탄젠트 활성화 함수 사용

$$h_t = \tanh(\hat{y}_t)$$
$$\hat{y}_t = W_{hh} \times h_{t-1} + W_{xh} \times x_t$$

2. **출력층**은 심층 신경망과 계산 방법이 동일
   = (은닉층 → 출력층 가중치 x 현재 은닉층) 에 소프트맥스 함수 적용

$$\hat{y}_t = \text{softmax}(W_{hy} \times h_t)$$

3. RNN의 **오차**(E)는 심층 신경망에서 전방향 학습과 달리 각 단계(t)마다 오차를 측정
   = 각 단계마다 실제 값(
   y_t)과 예측 값($\hat{y}_t$)으로 오차(평균 제곱 오차 적용)를 이용하여 측정

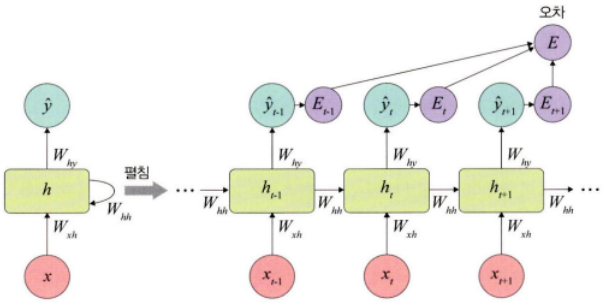


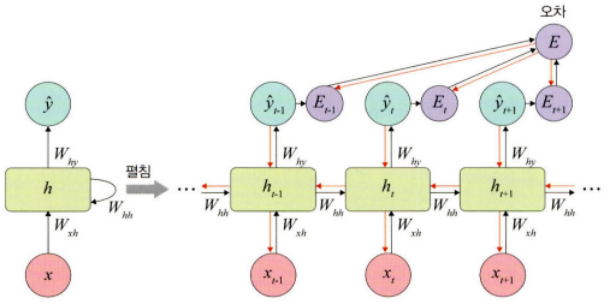
그림 7-11 RNN의 순방향 학습

4. RNN에서 **역전파**는 BPTT(BackPropagation Through Time)를 이용해서 모든 단계마다 처음부터 끝까지 역전파 함

   오차는 각 단계(t)마다 오차를 측정하고 이전 단계로 전달: BPTT

   3에서 구한 오차를 이용해서 W_{xh}, W_{hh}, W_{hy} 및 바이어스를 업데이트 - 이 때 BPTT는 오차가 멀리 전파될 때(왼쪽으로 전파) 계산량이 많아지고 전파되는 양이 점차 적어지는 문제점(기울기 소멸 문제)이 발생
   → 기울기 소멸 문제 보완 : 오차를 몇 단계 까지만 전파시키는 생략된_BPTT(truncated BPTT) 사용, LSTM 및 GU 사용

> **NOTE) 생략된-BPTT**
>
> 계산량을 줄이기 위해 현재 단계에서 일정 시점까지만(보통 5단계 이전까지만) 오류를 역전파 하는 것



NOTE) IMDB 데이터셋

영화 리뷰에 대한 데이터 5만 개
훈련 데이터 2만 5000개, 테스트 데이터 2만 5000개
각각 50%씩 긍정 리뷰와 부정 리뷰
이미 전처리 되어있어 각 리뷰가 숫자로 변환되어 있음

7.5 LSTM

RNN의 결정적 단점 - 기울기 소멸 문제
→ 해결 - LSTM, GRU 같은 확장된 RNN 방식 사용

7.5.1 LSTM 구조

LSTM 순전파

LSTM은 기울기 소멸 문제를 해결하기 위해 **망각 게이트, 입력 게이트, 출력 게이트**라는 새로운 요소를 은닉층의 각 뉴런에 추가함

- **망각 게이트** forget gate

  과거 정보를 어느 정도 기억할지 결정
  → 과거 정보와 현재 데이터를 입력 받아 시그모이드를 취한 후 그 값을 과거 정보에 곱해 줌
  → 시그모이드의 출력이 0이면 과거 정보는 버리고, 1이면 과거 정보는 온전히 보존

  0과 1 사이의 출력 값을 가지는 h_{t-1}과 x_t를 입력 값으로 받음
  -
  x_t: 새로운 입력 값
  -
  h_{t-1}: 이전 은닉층에서 입력되는 값
  →
  h_{t-1}과 x_t를 이용하여 이전 상태 정보를 현재 메모리에 반영할지 결정하는 역할

  - 계산한 값이 1이면 바로 직전의 정보를 메모리에 유지
  - 계산한 값이 0이면 초기화

$$f_t = \sigma(w_f[h_{t-1}, x_t])$$
$$c_t = f_t \cdot c_{t-1}$$

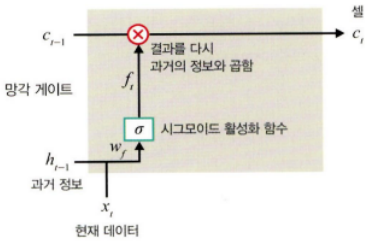


그림 7-16 망각 게이트

- **입력 게이트** input gate

  현재 정보를 기억하기 위해 만들어짐
  → 과거 정보와 현재 데이터를 입력 받아 시그모이드와 하이퍼볼릭 탄젠트 함수를 기반으로 현재 정보에 대한 보존량을 결정
  → 현재 메모리에 새로운 정보를 반영할지 결정하는 역할

  - 계산한 값이 1이면 입력 x_t가 들어올 수 있도록 허용(open)

- 계산한 값이 0이면 차단

$$i_t = \sigma(w_i[h_{t-1}, x_t])$$
$$\tilde{c}_t = \tanh(w_c[h_{t-1}, x_t])$$
$$c_t = c_{t-1} + i_t \cdot \tilde{c}_t$$

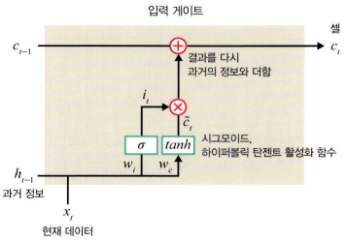


그림 7-17 입력 게이트

- **셀**

  각 단계에 대한 은닉 노드(hidden node) = 메모리 셀
  총합(sum)을 사용하여 셀 값을 반영 → 기울기 소멸 문제 해결

  셀 업데이트 방법
  : 망각 게이트와 입력 게이트의 이전 단계 셀 정보를 계산하여 현재 단계의 셀 상태를 업데이트

$$f_t = \sigma(w_f[h_{t-1}, x_t])$$
$$c_t = c_{t-1} + i_t \cdot \tilde{c}_t$$

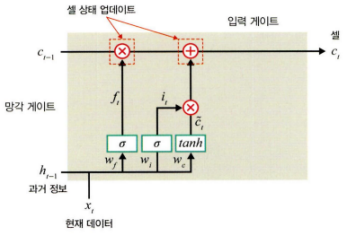


그림 7-18 셀

- **출력 게이트** output gate

  과거 정보와 현재 데이터를 사용하여 뉴런의 출력을 결정
  이전 은닉 상태와 t번째 입력을 고려해서 다음 은닉 상태를 계산
  LSTM에서는 이 은닉 상태가 그 시점에서의 출력이 됨
  → 출력 게이트는 갱신된 메모리의 출력 값을 제어하는 역할을 함

  - 계산한 값이 1이면 의미 있는 결과로 최종 출력
  - 계산한 값이 0이면 해당 연산 출력을 하지 않음

$$o_t = \sigma(w_o[h_{t-1}, x_t])$$
$$h_t = o_t \cdot \tanh(c_{t-1})$$

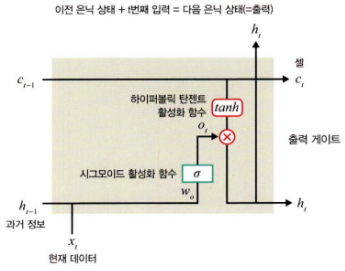


그림 7-19 출력 게이트

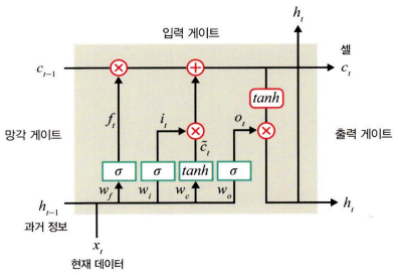


그림 7-20 LSTM 전체 게이트

LSTM 역전파

LSTM은 셀을 통해서 역전파를 수행 → 중단 없는 기울기(uninterrupted gradient flow)라고도 함
= 최종 오차는 모든 노드에 전파되는데, 이때 셀을 통해서 중단 없이 전파

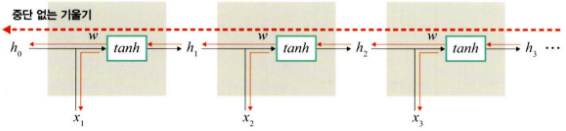


그림 7-21 LSTM 셀 단위 역전파

$$
\begin{aligned}
t_t &= \tanh(w_{hh}h_{t-1} + w_{xh}x_t) \\
&= \tanh((w_{hh} \quad w_{xh})\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}) \\
&= \tanh(w\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix})
\end{aligned}
$$

주의) 셀 단위로 오차가 전파된다고 해서 입력 방향으로 오차가 전파되지 않는 것은 아님
셀 내부적으로 오차가 입력(x_t)으로 전파됨

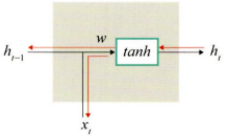


그림 7-22 입력층으로의 역전파

7.6 게이트 순환 신경망(GRU)

GRU(Gated Recurrent Unit)

- 게이트 메커니즘이 적용된 RNN 프레임워크의 한 종류
- LSTM보다 간단한 구조

7.6.1 GRU 구조

GRU는 LSTM에서 사용하는 망각 게이트와 입력 게이트를 하나로 합친 것이며, 별도의 업데이트 게이트로 구성되어 있음

하나의 게이트 컨트롤러(gate controller)가 망각 게이트와 입력 게이트를 모두 제어
→ 게이트 컨트롤러가 1을 출력 → 망각 게이트는 열리고 입력 게이트는 닫힘
→ 게이트 컨트롤러가 0을 출력 → 망각 게이트는 닫히고 입력 게이트는 열림
→ 이전 기억이 저장될 때마다 단계 별 입력은 삭제

GRU는 출력 게이트가 없어 전체 상태 벡터가 매 단계마다 출력 됨
이전 상태의 어느 부분이 출력될지 제어하는 새로운 게이트 컨트롤러가 별도로 존재

- **망각 게이트** reset gate

  과거 정보를 적당히 초기화 시키려는 목적
  시그모이드 함수를 출력으로 이용하여 (0,1) 값을 이전 은닉층에 곱함
  이전 시점의 은닉층 값에 현시점의 정보에 대한 가중치를 곱한 것

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$

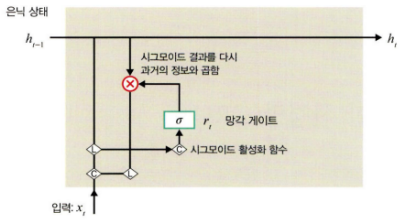


그림 7-26 망각 게이트

- **업데이트 게이트** update gate

  과거와 현재 정보의 최신화 비율을 결정하는 역할

  시그모이드로 출력된 결과(z_t)는 현시점의 정보량을 결정하고 1에서 뺀 값(1-z_t)을 직전 시점의 은닉층 정보와 곱함

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$$

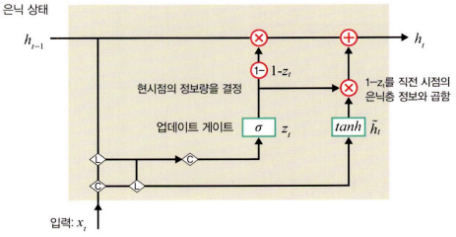


그림 7-27 업데이트 게이트

- **후보군** candidate

  현시점의 정보에 대한 후보군을 계산

  과거 은닉층의 정보를 그대로 이용하지 않고 망각 게이트의 결과를 이용하여 후보군을 계산

$$\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t])$$

(*는 점 단위 연산(pointwise operation)입니다. 예를 들어 벡터를 더할 때
각각의 차원(dimension)에 맞게 곱하거나 더하는 것이 가능해집니다)

- **은닉층 계산**

  업데이트 게이트 결과와 후보군 결과를 결합하여 현시점의 은닉층 계산

  - 시그모이드 함수의 결과는 현시점에서 결과에 대한 정보량을 결정
  - 1-시그모이드 함수의 결과는 과거의 정보량을 결정

$$h_t = (1 - z_t) * h_{t-1} + z_t \times \tilde{h}_t$$

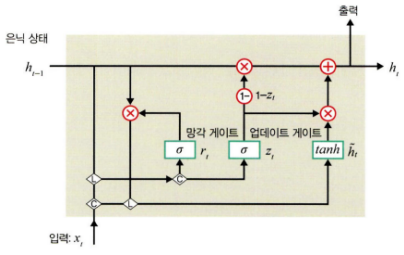


그림 7-28 GRU 내부 구조

7.7 양방향 RNN

RNN은 이전 시점의 데이터들을 참고해서 정답 예측
△ 실제 문제에서는 과거 시점이 아닌 미래 시점의 데이터에 힌트가 있는 경우 多

→ **양방향 RNN**(bidirectional RNN)은 이전 시점의 데이터 뿐만 아니라 이후 시점의 데이터도 함께 활용하여 출력 값을 예측하고자 함

7.7.1 양방향 RNN 구조

하나의 출력 값을 예측하는 데 메모리 셀 두 개 사용

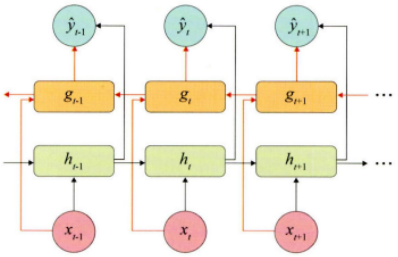


그림 7-30 양방향 RNN

- 첫 번째 메모리 셀(초록색 메모리 셀)

  이전 시점의 은닉 상태를 전달 받아 현재의 은닉 상태 계산
- 두 번째 메모리 셀(노란색 메모리 셀)

  다음 시점의 은닉 상태를 전달 받아 현재의 은닉 상태 계산

→ 이 값 두 개를 모두 출력층에서 출력 값을 예측하는 데 사용

양방향 RNN에 대한 개념은 RNN뿐만 아니라 LSTM이나 GRU에도 적용