[6교시]

1)RNN

- RNN학습
- 워드투벡터(Word2Vec)

2)RNN, LSTM, GRU

- RNN, LSTM, GRU 설명

3)GAN

- GAN이란 무엇인가?
- 체험: GAN을 이용한 음악 생성

[1교시]

1. 순환 인공 신경망(Recurrent neural network, RNN)

1) 자연어 처리(natural language processing)란?
- 우리가 일상생활에서 사용하는 언어를 말한다.
- 자연어의 의미를 분석하여 컴퓨터가 처리할 수 있도록 하는 일
- 자연어 처리는 음성 인식, 내용 요약, 번역, 사용자의 감성 분석, 텍스트 분류 작업(스팸 메일 분류, 뉴스 기사 카테고리 분류), 질의 응답 시스템, 챗봇과 같은 곳에서 사용되는 분야

2) 텍스트 전처리(Text preprocessing)
- 텍스트 전처리는 용도에 맞게 텍스트를 사전에 처리하는 작업
- 제대로 된 전처리를 하지 않으면 자연어 처리 기법들이 제대로 동작하지 않는다.

3) 토큰화(Tokenization)
- 토큰(Token)이란 문법적으로 더 이상 나눌 수 없는 언어요소를 뜻한다.
- 텍스트 토큰화(Text Tokenization)란 말뭉치로부터 토큰을 분리하는 작업을 뜻함
- 토큰화 규칙)
  규칙 1. 하이푼으로 구성된 단어는 하나로 유지한다.
  규칙 2. doesn't와 같이 아포스트로피로 '접어'가 함께하는 단어는 분리해준다.
  예) home-based는 하나의 토큰, doesn't의 경우 does와 n't는 분리

토큰화의 예)
입력: Time is an illusion. Lunchtime double so!
출력 : "Time", "is", "an", "illustion", "Lunchtime", "double", "so"

2. 워드투벡터(Word2Vec)

- 원-핫 인코딩에서 원-핫 벡터는 단어 간 유사도를 계산할 수 없다는 단점이 있음
- 그래서 **단어 간 유사도를 반영할 수 있도록 단어의 의미를 벡터화 할 수 있는 방법**이 필요
- 이를 위해서 사용되는 대표적인 방법이 워드투벡터(Word2Vec)
- 방법적으로 CBOW(Continuous Bag of Words)와 Skip-Gram 두 가지 방식이 있다.

◑ **CBOW(Continuous Bag of Words)**
- CBOW은 주변에 있는 단어들을 가지고 중심에 있는 단어를 맞추는 방식
  예:

나는 ____에 간다.

위 문장에 들어갈 수 있는 단어는 다양하다. '학교','집','회사'등등....이렇듯 주변 단어를 가지고 중심에 있는 단어를 맞춤으로써 단어 벡터들을 만들어 내는 방법이 CBOW이다.

◑ **Skip-Gram**
- 중심에 있는 단어로 주변 단어를 예측하는 방법

예:

-- 외나무다리 --

'외나무다리' 앞에는 어떤 단어가 올까? 아마도 '-는'이겠고. 그 앞에는 '원수'가 오겠고.
'외나무다리' 뒤에는 어떤 단어가 등장할까? '-에서'와 '만난다'일 가능성이 높겠다.

- Word2Vec는 코사인과 내적의 성질을 목적함수 구축에 적극 활용
- Word2Vec는 자주 같이 등장하는 단어들의 정보(co-occurrence)를 보존한다.
- Word2Vec의 문제점: 보존하려는 정보가 '동시등장 여부'이기 때문에 생기는 근본적인 한계상황

1) 희소 표현(Sparse Representation)

: one-hot vector란?
원-핫 인코딩을 통해서 나온 원-핫 벡터들은 표현하고자 하는 단어의 인덱스의 값만 1이고, 나머지 인덱스에는 전부 0으로 표현되는 벡터 표현 방법.
- 이렇게 **벡터 또는 행렬(matrix)의 값이 대부분이 0으로 표현되는 방법을 희소 표현(sparse representation)**이라고 함
- 원-핫 벡터는 희소 벡터(sparse vector)이다.
- 원-핫 벡터의 단점:
각 단어간 유사성을 표현할 수 없다
- 이를 위한 대안으로 단어의 '의미'를 **다차원 공간에 벡터화하는 방법**을 찾게 되는데, 이러한 표현 방법을 **분산 표현(distributed representation)**이라고 함
- 분산 표현을 이용하여 단어의 유사도를 벡터화하는 작업은 워드 임베딩(embedding) 에 속하기 때문에, **임베딩 벡터(embedding vector)**라고 하며, 저차원을 가지므로 밀집 벡터(dense vector)에도 속한다.

※ 밀집 표현(Dense Representation)
- 밀집표현은 희소 표현과는 반대되는 표현
- 벡터의 차원을 단어 집합의 크기로 상정하지 않고, 사용자가 설정한 값으로 모든 단어의 벡터표현의 차원을 맞춘다. 또한, 이 과정에서 더 이상 0과 1만 가진 값이 아니라 **실수 값을 가지게 된다**.
희소 표현의 예: Ex) 강아지 = [0 0 0 0 1 0 0 0 0 0 0 0 ... 중략 ... 0]
이 때 1 뒤의 0의 수는 9995개. 차원은 10,000
10,000개의 단어가 있을 때 강아지란 단어를 표현하기 위해서는 위와 같은 표현을 사용했다.
=> 하지만, 밀집 표현의 차원을 **128로 설정**한다면, 모든 단어에 대한 벡터 표현의 차원은 **128로 바뀌면서 모든 값이 실수가 된다.**
즉
강아지 = [0.2 1.8 1.1 -2.1 1.1 2.8 ... 중략 ...] # 이 벡터의 차원은 128,
이 경우 **벡터의 차원이 조밀해졌다고 하여 밀집 벡터(dense vector)라고 한다.**

▶ 비교 표

-	원-핫 벡터	임베딩 벡터
차원	고차원 (단어 집합의 크기)	저차원
다른 표현	희소 벡터의 일종	밀집 벡터의 일종
표현 방법	수동	훈련 데이터로부터 학습
값의 타입	1과 0	실수

※ 워드 임베딩(Word Embedding)

- **단어를 밀집 벡터(dense vector)의 형태로 표현하는 방법=워드임배딩(Word Embedding)**
- **밀집 벡터가 워드 임베딩 과정을 통해 나온 결과**라고 하여 **임베딩 벡터(embedding vector)**라고도 함.
- 워드 임베딩 방법론으로는 LSA, Word2Vec, FastText, Glove 등이 있다.

가) 잠재 의미 분석(Latent Semantic Analysis, LSA)

- 단어의 빈도 수를 이용한 수치화 방법에서 **단어의 의미를 고려하지 못한다는 단점**이 있다. 이를 위한 대안으로 **잠재된(Latent) 의미를 이끌어내는 방법**으로 잠재 의미 분석(Latent Semantic Analysis, LSA)이라는 방법이 있다.
- LSA는 차원을 축소시키고, 단어들의 잠재적인 의미를 끌어 낸다.

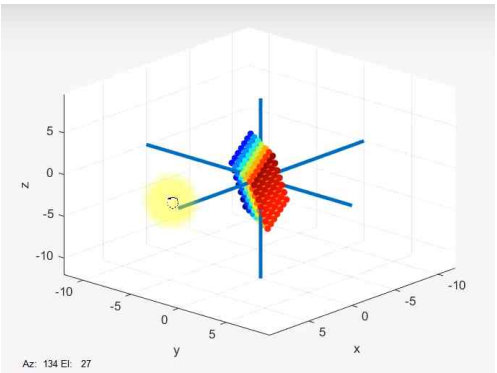
나) FastText

- FastText는 구글에서 개발한 Word2Vec을 기본으로 하되 **부분단어들을 임베딩하는 기법**

다) GloVe

- GloVe가 보존하려는 정보는 **단어 동시 등장** 여부
- GloVe로 임베딩된 단어 벡터끼리의 내적은 동시 등장확률의 로그값과 같다.

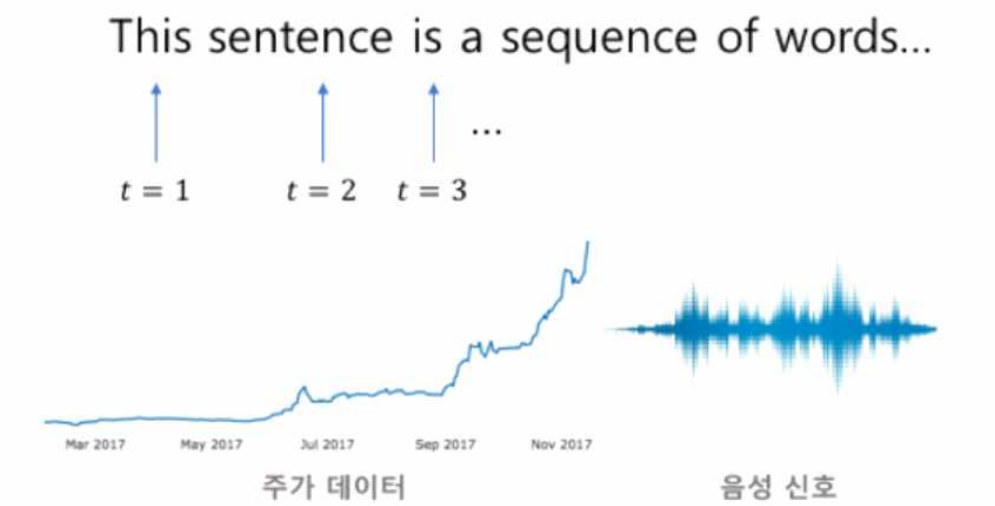
◈ 선형변환 예시: 특이값 분해(SVD)의 기하학적 의미와 활용 소개
https://www.youtube.com/watch?v=cq5qlYtnLoY



[2교시]

1. 순환 신경망 (RNN, Recurrent Neural Network)

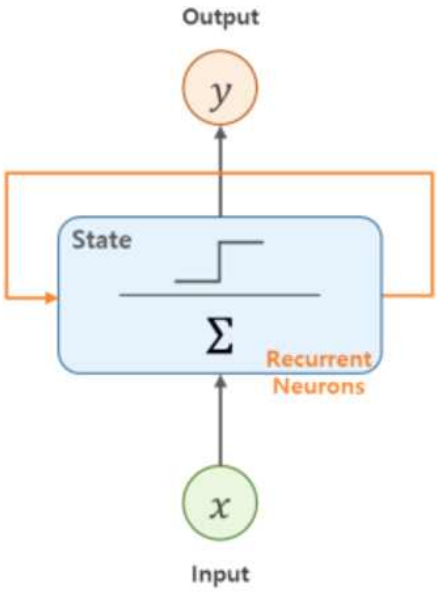
- 자연어(NL, Natural Language)나 음성신호, 주식과 같은 연속적인(sequential) 시계열(time series) 데이터에 적합한 모델인 RNN(Recurrent Neural Network)



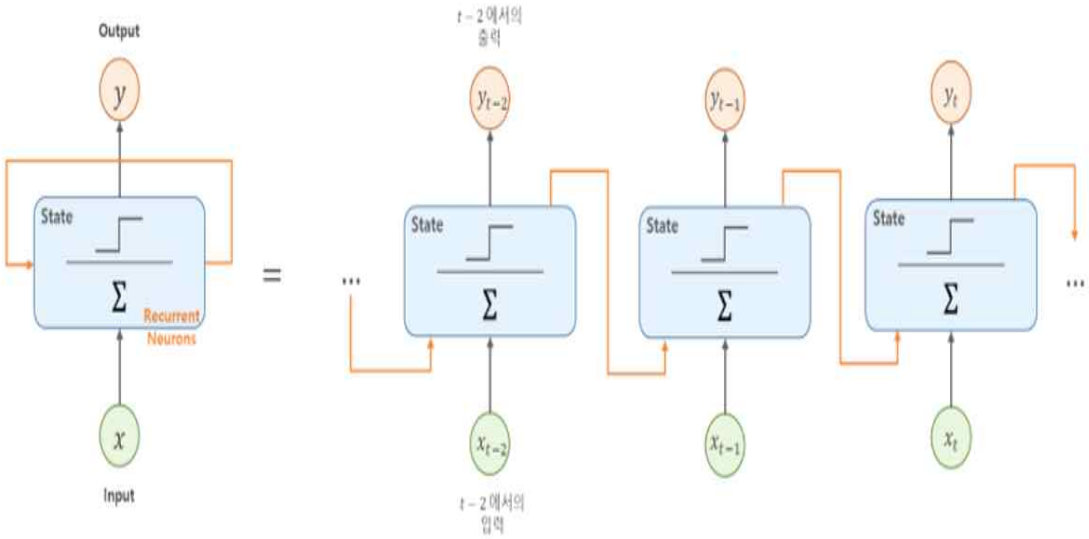
1) 순환 뉴런(Recurrent Neurons)

신경망은 입력층 → 출력층 한 방향으로만 흐르는 피드포워드(feedforward) 신경망이다.

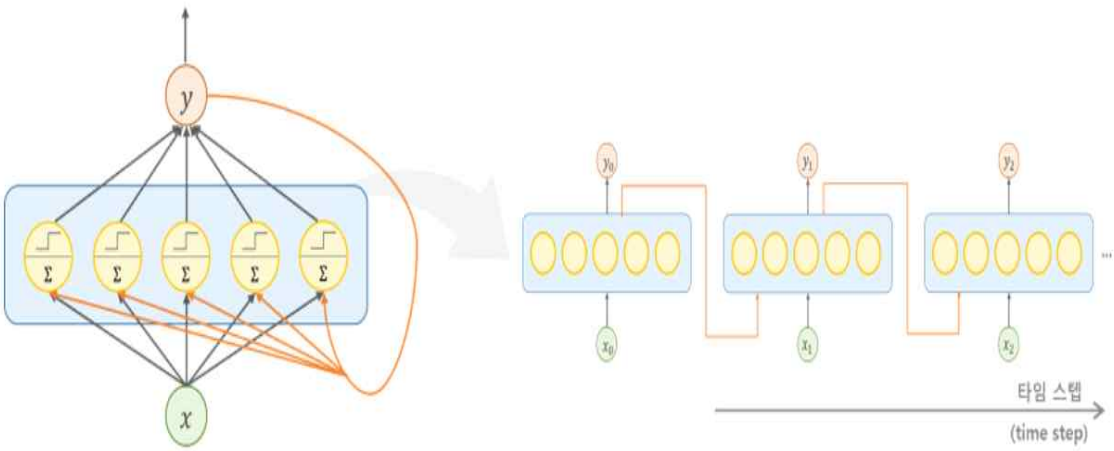
- RNN(순환 신경망)은 피드포워드 신경망과 비슷하지만, 출력이 다시 입력으로 받는 부분이 있다.



- RNN은 입력(x)을 받아 출력(y)를 만들고, 이 출력을 다시 입력으로 받는다. RNN을 그림으로 나타낼 때는 위의 그림처럼 하나로 나타내지 않고, 아래의 그림처럼 각 타임스텝(time step) t마다 순환 뉴런을 펼쳐서 타임스텝 별 입력(xt)과 출력(yt)을 나타낸다.



- 순환 뉴런으로 구성된 층(layer)은 아래의 그림처럼 나타낼 수 있는데, 타임 스텝 t마다 모든 뉴런은 입력 벡터 xt와 이전 타임 스텝의 출력 벡터yt-1 을 입력 받는다.



각 순환 뉴런은 두 개의 가중치 $\mathbf{w}_x$ 와 $\mathbf{w}_y$ 를 가지는데, $\mathbf{w}_x$ 는 $\mathbf{x}_t$ 를 위한 것이고 $\mathbf{w}_y$는 이전 타임 스텝의 출력 $\mathbf{y}_{t-1}$을 위한 것이다. 이것을 순환 층(layer) 전체로 생각하면 가중치 벡터 $\mathbf{w}_x$ 와 $\mathbf{w}_y$ 를 행렬 $\mathbf{W}_x$ 와 $\mathbf{W}_y$ 로 나타낼 수 있으며 다음의 식과 같이 표현할 수 있다.

$$\mathbf{y}_t = \phi\left(\mathbf{W}_x^T \cdot \mathbf{x}_t + \mathbf{W}_y^T \cdot \mathbf{y}_{t-1} + \mathbf{b}\right)$$

타임 스텝 t에서의 미니배치(mini-batch)의 입력을 행렬 Xt로 나타내어 아래와 같이 순환 층의 출력을 한 번에 계산할 수 있다.

$$\mathbf{Y}_t = \phi\left(\mathbf{X}_t \cdot \mathbf{W}_x + \mathbf{Y}_{t-1} \cdot \mathbf{W}_y + \mathbf{b}\right)$$
$$= \phi\left(\begin{bmatrix}\mathbf{X}_t & \mathbf{Y}_{t-1}\end{bmatrix}\begin{bmatrix}\mathbf{W}_x \\ \mathbf{W}_y\end{bmatrix} + \mathbf{b}\right)$$

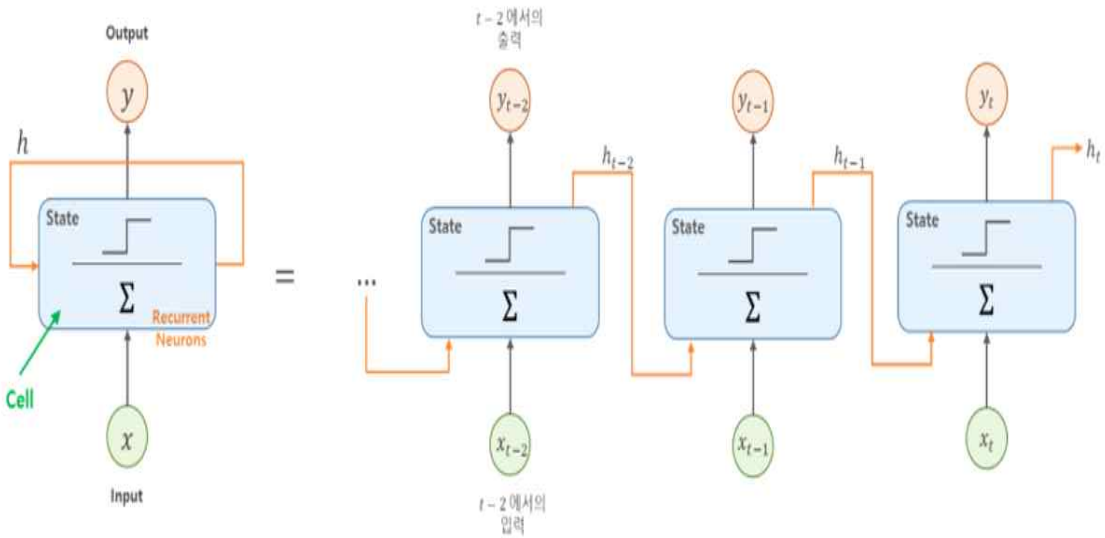
- $\mathbf{Y}_t$: 타임 스텝 t에서 미니배치에 있는 각 샘플(미니배치)에 대한 순환 층의 출력이며, $m \times n_{\text{neurons}}$ 행렬(m은 미니배치, n_{neurons}은 뉴런 수)
- $\mathbf{X}_t$: 모든 샘플의 입력값을 담고 있는 $m \times n_{\text{inputs}}$ 행렬 (n_{inputs}은 입력 특성 수)
- $\mathbf{W}_x$: 현재 타임 스텝 t의 입력에 대한 가중치를 담고 있는 $n_{\text{inputs}} \times n_{\text{neurons}}$ 행렬
- $\mathbf{W}_y$: 이전 타임 스텝 $t-1$ 의 출력에 대한 가중치를 담고 있는 $n_{\text{neurons}} \times n_{\text{neurons}}$ 행렬
- $\mathbf{b}$: 각 뉴런의 편향(bias)을 담고 있는 n_{neurons} 크기의 벡터

가) 메모리 셀

타임 스텝 t 에서 순환 뉴런의 출력은 이전 타임 스텝의 모든 입력에 대한 함수이기 때문에 이것을 **메모리**라고 볼 수 있다. 이렇게 타임 스텝에 걸쳐 어떠한 상태를 보존하는 신경망의 구성 요소를 **메모리 셀**(memory cell) 또는 셀(cell)이라고 한다. 일반적으로 타임 스텝 t 에서 셀의 상태 $\mathbf{h}_t$ (h = hidden)는 아래의 식과 같이 타임 스텝에서의 입력과 이전 타임 스텝의 상태에 대한 함수이다.

$$\mathbf{h}_t = f\left(\mathbf{h}_{t-1}, \mathbf{x}_t\right)$$

위에서 살펴본 RNN은 출력 $\mathbf{y}_t$ 가 다시 입력으로 들어갔지만, 아래의 그림과 같이 일반적으로 많이 사용되는 RNN은 출력 $\mathbf{y}_t$ 와 히든 상태(state) $\mathbf{h}_t$가 구분되며, 입력으로는 $\mathbf{h}_t$가 들어간다. RNN에서의 활성화 함수로는 $\tanh$가 주로 사용



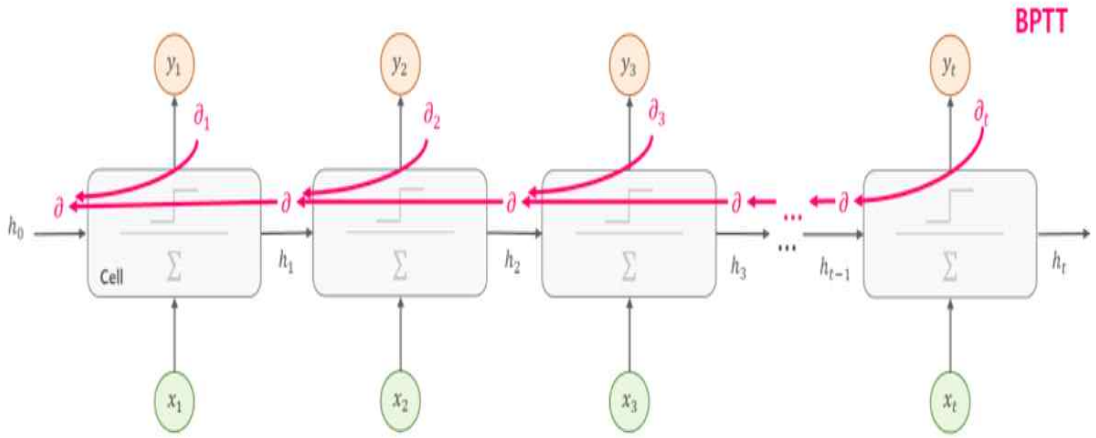
위의 그림을 식으로 나타내면 다음과 같다.

$$\mathbf{h}_t = \tanh(\mathbf{X}_t \cdot \mathbf{W}_x + \mathbf{h}_{t-1} \cdot \mathbf{W}_h + \mathbf{b})$$
$$= \tanh\left(\begin{bmatrix} \mathbf{X}_t & \mathbf{h}_{t-1} \end{bmatrix} \begin{bmatrix} \mathbf{W}_x \\ \mathbf{W}_h \end{bmatrix} + \mathbf{b} \right)$$
$$\mathbf{Y}_t = \mathbf{W}_y^T \cdot \mathbf{h}_t$$

2) RNN 학습시키기

1.1 BPTT (BackPropagation Through Time)

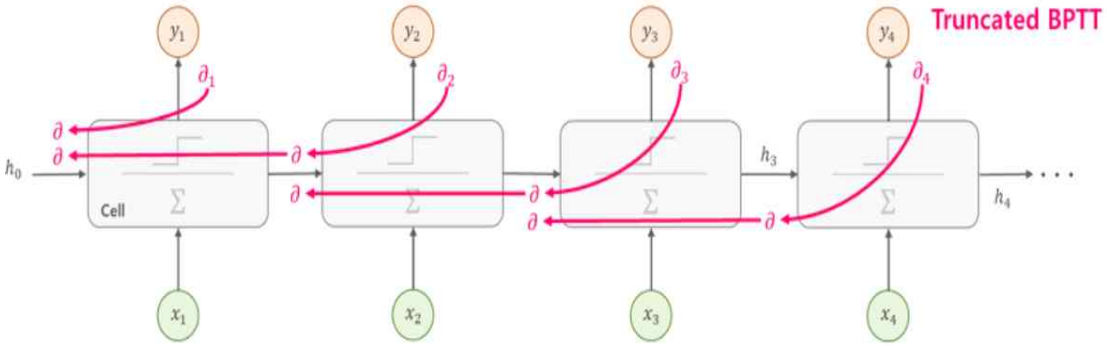
RNN은 타임 스텝별로 네트워크를 펼친 다음, 역전파 알고리즘을 사용하는데 이를 BPTT(BackPropagation Through Time)라고 한다.



- BPTT 또한 일반적인 역전파와 같이 먼저 순전파(forward prop)로 각 타임 스텝별 시퀀스를 출력한다.
- 그런 다음 이 출력 시퀀스와 손실(비용)함수를 사용하여 각 타임 스텝별 Loss를 구한다.
- 손실 함수의 그래디언트는 위의 그림과 같이 펼쳐진 네트워크를 따라 역방향으로 전파된다. BPTT는 그래디언트가 마지막 타임 스텝인 출력뿐만 아니라 손실함수를 사용한 모든 출력에서 역방향으로 전파된다.
- RNN은 각 타임 스텝마다 같은 매개변수 w와 b가 사용되기 때문에 역전파가 진행되면서 모든 타임 스텝에 걸쳐 매개변수 값이 합산된다. 이렇게 업데이트된 가중치는 순전파 동안에는 모든 타임 스텝에 동일한 가중치가 적용된다.

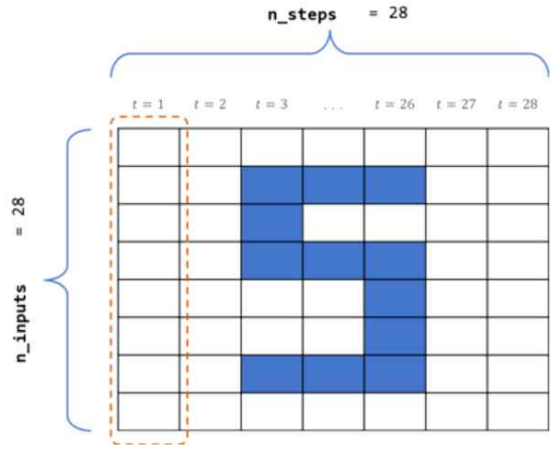
1.2 Truncated BPTT

- BPTT는 전체의 타임 스텝마다 처음부터 끝까지 역전파를 하기 때문에 타임 스텝이 클수록 계산량이 많아지는 문제가 있다.
- 이러한 계산량 문제를 해결하기 위해 전체 타임 스텝을 일정 구간(예를들어 3 또는 5 구간)으로 나눠 역전파를 하는 Truncated BPTT를 사용한다.



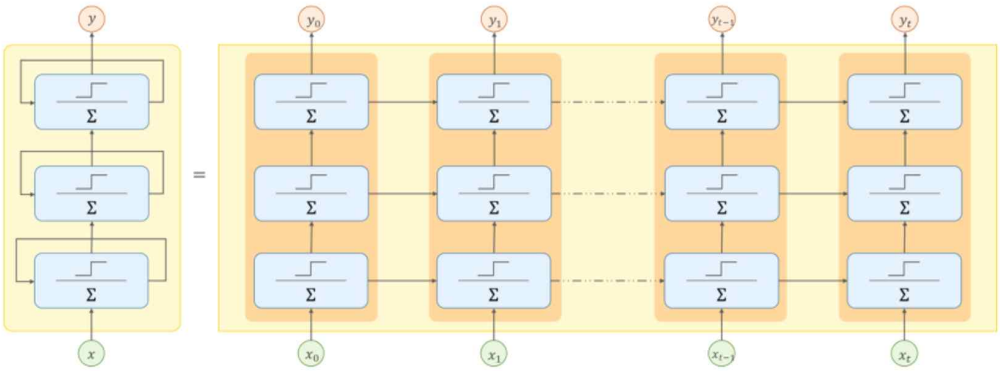
1.3 RNN을 이용한 분류기 구현

아래의 그림처럼 MNIST 데이터에서 28 x 28 픽셀을 시퀀스의 각원소는 28개의 픽셀을 가진 길이가 28 시퀀스 데이터로 볼 수 있다.



2) 심층 RNN

1.1 심층 RNN(deep RNN)은 RNN Cell을 여러 층으로 쌓은 것을 말한다.



2.2 Dropout 적용하기

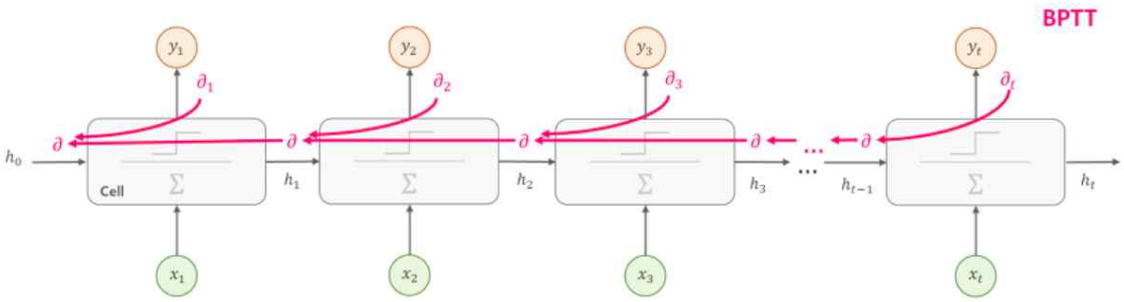
- 여러 층의 RNN Cell을 쌓게 되면 오버피팅(overfitting)되기 쉽기 때문에 RNN layer의 전과 후에 드롭아웃(dropout) layer를 추가할 수 있다.
- RNN 층(layer) 사이에도 드롭아웃을 적용할 수 있는데, 텐서플로에서는 tf.nn.rnn_cell.DropoutWrapper를 이용해 RNN 층 사이에도 드롭아웃을 적용할 수 있다.

2. 순환 신경망 LSTM, GRU

1) RNN Cell의 문제점

1.1 BPTT의 문제점

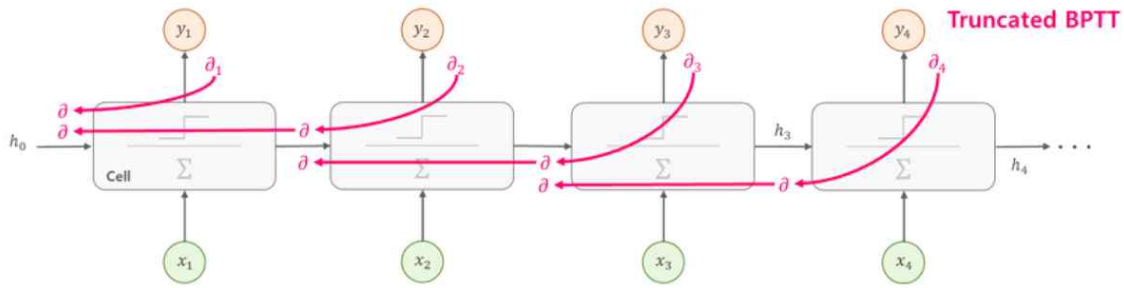
- RNN에서의 역전파 방법인 BPTT(BackPropagation Through Time)은 모든 타임스텝마다 처음부터 끝까지 역전파한다.



- 그렇기 때문에 타임 스텝이 클 경우, 위의 그림과 같이 RNN을 펼치게(unfold)되면 매우 깊은 네트워크가 될 것이며, 이러한 네트워크는 그래디언트 소실 및 폭주(vanishing & exploding gradient) 문제가 발생할 가능성이 크다. 그리고, 계산량 또한 많기 때문에 한번 학습하는데 아주 오랜 시간이 걸리는 문제가 있다.

◈ Truncated BPTT

BPTT의 문제를 해결하기 위해 아래의 타임 스텝을 일정 구간(보통 5-steps)으로 나누어 역전파(backprop)를 계산하여, 전체 역전파로 근사시키는 방법인 Truncated BPTT를 대안으로 사용할 수 있다.

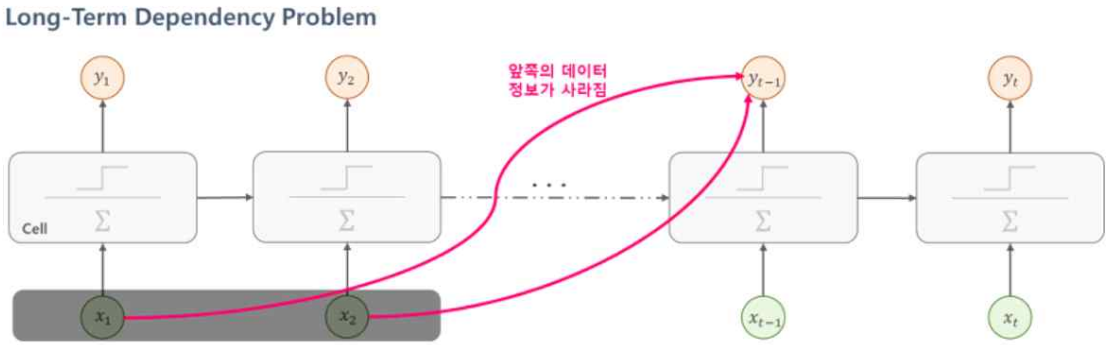


하지만 truncated-BPTT의 문제는 만약 학습 데이터가 장기간에 걸쳐 패턴이 발생한다고 하면, 이러한 **장기간(Long-Term)의 패턴을 학습할 수 없는 문제가 있다.**

1.2 장기 의존성(Long-Term Dependency) 문제

- RNN은 타임 스텝 에서 이전 타임 스텝(t-1)의 상태(state, h_{t-1})를 입력으로 받는 구조이기 때문에, 이전의 정보가 현재의 타임 스텝 t에 영향을 줄 수 있다. 따라서, RNN의 순환 뉴런(Reccurent Neurons)의 출력은 이전 타임 스텝의 모든 입력에 대한 함수이므로, 이를 메모리 셀(memory cell)이라고 한다.

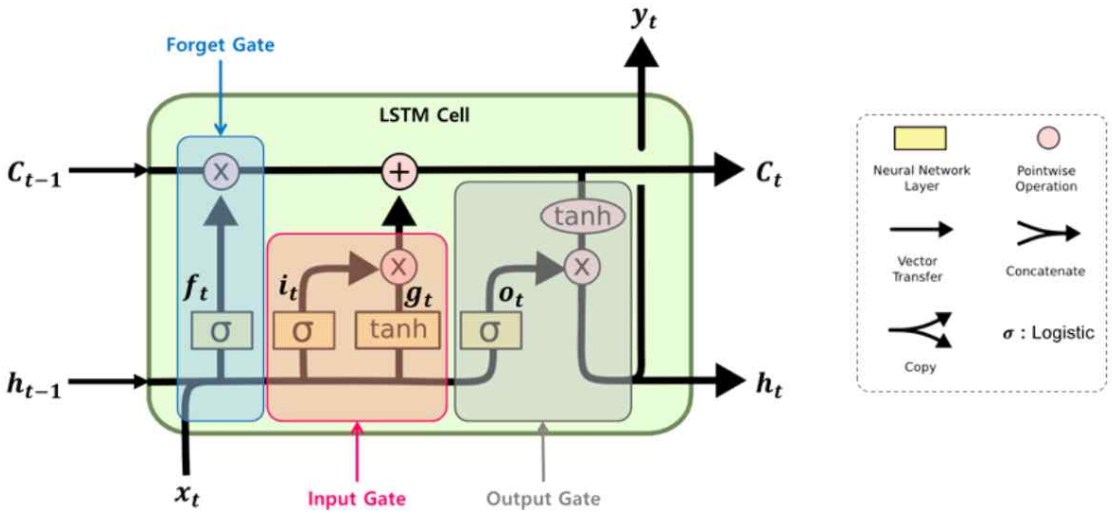
- RNN은 이론적으로 모든 이전 타임 스텝이 영향을 주지만 앞쪽의 타임 스텝(예를 들어 t=0, t=1)은 타임 스텝이 길어질수록 영향을 주지 못하는 문제가 발생하는데 이를 장기 의존성(Long-Term Dependency) 문제라고 한다. 이러한 문제가 발생하는 이유는 입력 데이터가 RNN Cell을 거치면서 특정 연산을 통해 데이터가 변환되어, 일부 정보는 타임 스텝마다 사라지기 때문이다.



이러한 문제를 해결하기 위해 **장기간의 메모리를 가질 수 있는** 여러 종류의 셀이 만들어졌는데, 그 중에서 대표적인 셀들이 **LSTM과 GRU 셀이다.**

■ LSTM Cell

LSTM(Long Short-Term Memory) 셀은 RNN 셀의 장기 의존성 문제를 해결할 뿐만 아니라 학습 또한 빠르게 수렴한다.



- 위의 그림에서 LSTM 셀에서 상태(state)가 두 개의 벡터 ht와 ct로 나누어 진다.
- ht를 단기 상태(short-term state), ct는 장기 상태(long-term state)를 뜻함
- ct에서 기억할 부분, 삭제할 부분, 읽어 들일 부분을 학습
- 장기 기억 ct-1은 셀의 왼쪽에서 오른쪽으로 통과하게 되는데 forget gate를 지나면서 일부를 기억(정보)을 잃고, 그 다음 덧셈(+) 연산으로 input gate로 부터 새로운 기억 일부를 추가한다.
- 이렇게 만들어진 ct는 별도의 추가 연산 없이 바로 출력
- 이러한 장기 기억 ct는 타임 스텝마다 일부의 기억을 삭제하고 추가하는 과정을 거치게

된다.
- 그리고 덧셈 연산 후에 ct는 복사되어 output gate의 함수로 전달되어 단기 상태 ht와 셀의 출력인 yt를 만든다.

- 현재 입력 벡터 와 이전의 단기 상태ht-1 이 네 개의 다른 FC-레이어(Fully-Connected layer)에 주입되는데, 이 레이어는gt를 출력, 현재 입력 데이터xt와 이전 타임스텝 단기데이터ht-1을 분석하는 역할을 함
- gt가 it의 곱셈(x)연산 후 장기상태ct에 일부분을 더한다.
- 기본 이 레이어만 있으면 RNN셀에서는 바로 yt와 ht로 출력된다.

- $\mathbf{f}_t, \mathbf{i}_t, \mathbf{o}_t$를 출력하는 세 개의 레이어에서는 활성화 함수로 시그모이드(sigmoid, logistic)를 사용한다. 시그모이드 함수의 출력의 범위는 0 ~ 1 이며, 이 출력값은 각 forget, input, output 게이트의 원소별(element-wise) 곱셈연산에 입력된다. 따라서, 출력이 `0` 일 경우에는 게이트를 닫고 `1` 일 경우에는 게이트를 열기 때문에 $\mathbf{f}_t, \mathbf{i}_t, \mathbf{o}_t$를 **gate controller**라고 한다.
  - **Forget gate** : $\mathbf{f}_t$에 의해 제어되며 장기 상태 $\mathbf{c}_t$의 어느 부분을 삭제할지 제어한다.
  - **Input gate** : $\mathbf{i}_t$에 의해 제어되며 $\mathbf{g}_t$의 어느 부분이 장기 상태 $\mathbf{c}_t$에 더해져야 하는지 제어한다.
  - **Output gate** : $\mathbf{o}_t$는 장기 상태 $\mathbf{c}_t$의 어느 부분을 읽어서 $\mathbf{h}_t$ 와 $\mathbf{y}_t$로 출력해야 하는지 제어한다.

- 타임 스텝 에서, 셀의 장기 상태, 단기 상태, 그리고 각 레이어의 출력을 구하는 식은

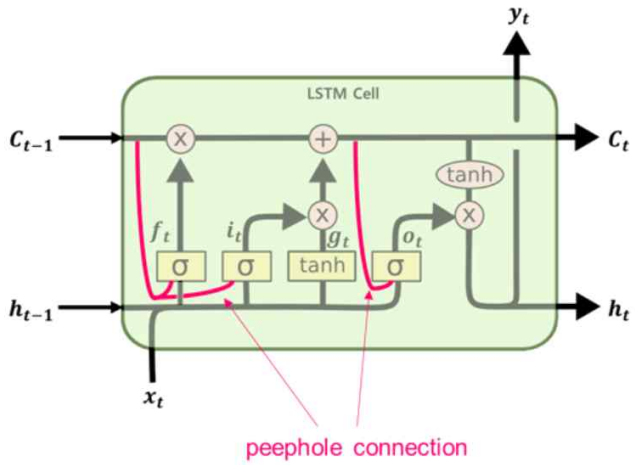
$$
\begin{aligned}
\mathbf{f}_t &= \sigma\left(\mathbf{W}_{xf}^T \cdot \mathbf{x}_t + \mathbf{W}_{hf}^T \cdot \mathbf{h}_{t-1} + \mathbf{b}_f\right) \\
\mathbf{i}_t &= \sigma\left(\mathbf{W}_{xi}^T \cdot \mathbf{x}_t + \mathbf{W}_{hi}^T \cdot \mathbf{h}_{t-1} + \mathbf{b}_i\right) \\
\mathbf{o}_t &= \sigma\left(\mathbf{W}_{xo}^T \cdot \mathbf{x}_t + \mathbf{W}_{ho}^T \cdot \mathbf{h}_{t-1} + \mathbf{b}_o\right) \\
\mathbf{g}_t &= \tanh\left(\mathbf{W}_{xg}^T \cdot \mathbf{x}_t + \mathbf{W}_{hg}^T \cdot \mathbf{h}_{t-1} + \mathbf{b}_g\right) \\
\mathbf{c}_t &= \mathbf{f}_t \otimes \mathbf{c}_{t-1} + \mathbf{i}_t \otimes \mathbf{g}_t \\
\mathbf{y}_t, \mathbf{h}_t &= \mathbf{o}_t \otimes \tanh(\mathbf{c}_t)
\end{aligned}
$$

- $\mathbf{W}_{xf}, \mathbf{W}_{xi}, \mathbf{W}_{xo}, \mathbf{W}_{xg}$: 입력 벡터 $\mathbf{x}_t$에 연결된 네 개의 레이어에 대한 가중치 행렬
- $\mathbf{W}_{hf}, \mathbf{W}_{hi}, \mathbf{W}_{ho}, \mathbf{W}_{hg}$: 이전 타임스텝의 단기 상태 $\mathbf{h}_{t-1}$에 연결된 네 개의 레이어에 대한 가중치 행렬
- $\mathbf{b}_f, \mathbf{b}_i, \mathbf{b}_o, \mathbf{b}_g$: 네 개의 레이어에 대한 편향(bias), **텐서플로**(TensorFlow)에서는 $\mathbf{b}_f$ 를 `1` 로 초기화하여 학습 시작시에 모든것을 잃어버리는 것을 방지한다.

◈ 핍홀(peephole) 연결
핍홀 연결(peephole connection)은 LSTM의 변종이다.
- 기존의 LSTM에서 gate controller(ft,it,ot)는 입력 xt와 이전 타임스텝의 단기 상태 ht-1만 입력으로 받는다.
- 하지만 핍홀 연결을 아래의 그림과 같이 연결 해주면서 gate controller에 이전 타임스텝의 장기 상태 ct-1가 입력으로 추가되며, 좀 더 많은 맥락(context)를 인식할 수

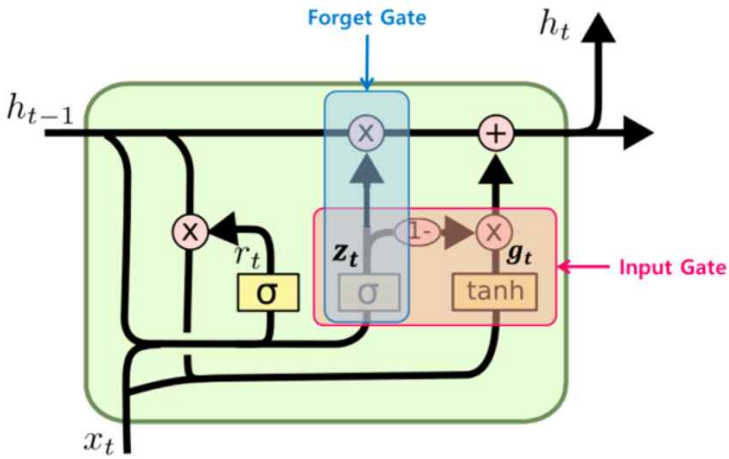
있다.

$$\mathbf{f}_t = \sigma\left(\mathbf{W}_{cf}^T \cdot \mathbf{c}_{t-1} + \mathbf{W}_{xf}^T \cdot \mathbf{x}_t + \mathbf{W}_{hf}^T \cdot \mathbf{h}_{t-1} + \mathbf{b}_f\right)$$
$$\mathbf{i}_t = \sigma\left(\mathbf{W}_{ci}^T \cdot \mathbf{c}_{t-1} + \mathbf{W}_{xi}^T \cdot \mathbf{x}_t + \mathbf{W}_{hi}^T \cdot \mathbf{h}_{t-1} + \mathbf{b}_i\right)$$
$$\mathbf{o}_t = \sigma\left(\mathbf{W}_{co}^T \cdot \mathbf{c}_t + \mathbf{W}_{xo}^T \cdot \mathbf{x}_t + \mathbf{W}_{ho}^T \cdot \mathbf{h}_{t-1} + \mathbf{b}_o\right)$$



■ GRU Cell

- GRU(Gated Recurrent Unit) 셀은 LSTM 셀의 간소화된 버전



- LSTM Cell에서의 두 상태 벡터 $\mathbf{c}_t$와 $\mathbf{h}_t$가 하나의 벡터 $\mathbf{h}_t$로 합쳐졌다.
- 하나의 gate controller인 $\mathbf{z}_t$가 **forget**과 **input** 게이트(gate)를 모두 제어한다. $\mathbf{z}_t$가 `1` 을 출력하면 forget 게이트가 열리고 input 게이트가 닫히며, $\mathbf{z}_t$가 `0` 일 경우 반대로 forget 게이트가 닫히고 input 게이트가 열린다. 즉, 이전$(t-1)$의 기억이 저장 될때 마다 타임 스텝 t의 입력은 삭제된다.
- GRU 셀은 output 게이트가 없어 전체 상태 벡터 $\mathbf{h}_t$가 타임 스텝마다 출력되며, 이전 상태 $\mathbf{h}_{t-1}$의 어느 부분이 출력될지 제어하는 새로운 gate controller인 $\mathbf{r}_t$가 있다.

- GRU 셀의 상태(state)와 각 레이어의 출력을 계산하는 식은

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

참조 사이트:
https://github.com/ExcelsiorCJH/Hands-On-ML/blob/master/Chap14-Recurrent_Neural_Networks/Chap14_3-Recurrent_Neural_Networks.ipynb

구글 콜렙 접속 주소:
https://colab.research.google.com/github/ExcelsiorCJH/Hands-On-ML/blob/master/Chap14-Recurrent_Neural_Networks/Chap14_3-Recurrent_Neural_Networks.ipynb?hl=ko

[3차시]

1. 머신러닝 종류

지도학습 (Supervised Learning)
비지도 학습 (Unsupervised Learning)

1) 지도학습 (Supervised Learning)

- 훈련 데이터에 레이블(Label)이라는 명시적인 정답이 주어진 상태에서 컴퓨터를 학습시키는 방법
가) 분류(classification); 숫자 인식, 스팸 메일을 분류등 정답을 맞추는 분야를 학습
나) 회귀(regression); 예측 변수(Predictor Variable)라 불리는 특성을 사용하여 최종 결과를 예측하는 방식, 3개월 뒤 아파트 가격을 예측하거나 몇 달 뒤 중고차 가격을 예측하는 것

◈ 지도 학습에 알고리즘
K-최근접 이웃 (KNN)
선형 회귀
로지스틱 회귀
서포트 백터 머신
결정트리와 랜덤 포레스트
신경망

2) 비지도 학습 (Unsupervised Learning)

- 학습에서 필요한 레이블(Label)이 필요하지 않으며, 정답이 없는 상태에서 학습 시키는 방식
- 데이터가 무작위로 분포 되어 있을 때 비슷한 특성을 가진 데이터들을 묶는 방식으로 대표적인 클러스터링(Clustering) 알고리즘
- 비지도 학습은 데이터의 숨겨진 특징이나 구조를 발견하는데 사용되는 알고리즘

◈ 비지도 학습 알고리즘

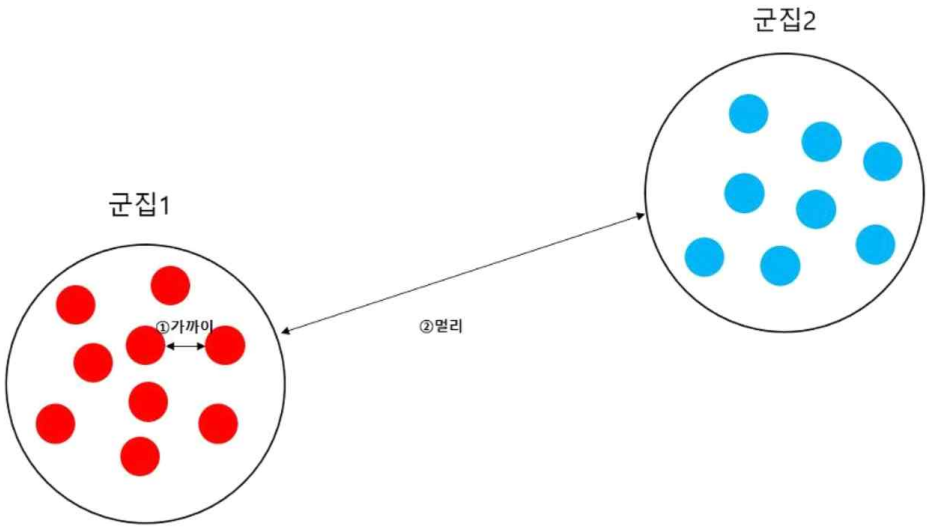
• 군집 (Clustering)
  K-평균 (K-Means)
  계측 군집 분석(HCA, Hierarchical Cluster Analysis)
  기댓값 최대화 (Expectation Maximization)

• 시각화(Visualization)와 차원 축소(Dimensionality Reduction)
  주성분 분석(PCA, Principal Component Analysis)
  커널 PCA(Kernel PCA)
  지역적 선형 임베딩(LLE, Locally-Linear Embedding)
  t-SNE(t-distributed Stochastic Neighbor Embedding)

• 연관 규칙 학습(Association Rule Learning)
  어프라이어리(Apriori)
  이클렛(Eclat)

3) 군집 (Clustering)

군집분석이란?
주어진 데이터들의 특성을 고려해 같은 그룹(클러스터)를 정의하고, 다른 클러스터의 개체보다 서로 더 유사한 개체가 되도록 그룹화하여 그룹의 대표성을 찾아내는 방법이다.



(1) cluster의 기준 : 거리
- 어떻게 군집화 할 것인가에 대한 기준에는 "거리"가 있다.
- 거리척도의 유형에는
① 유클리디안 거리(Euclidean Distance)

점 $\mathbf{p} = (p_1, p_2, \ldots, p_n)$와 $\mathbf{q} = (q_1, q_2, \ldots, q_n)$가 있을때,

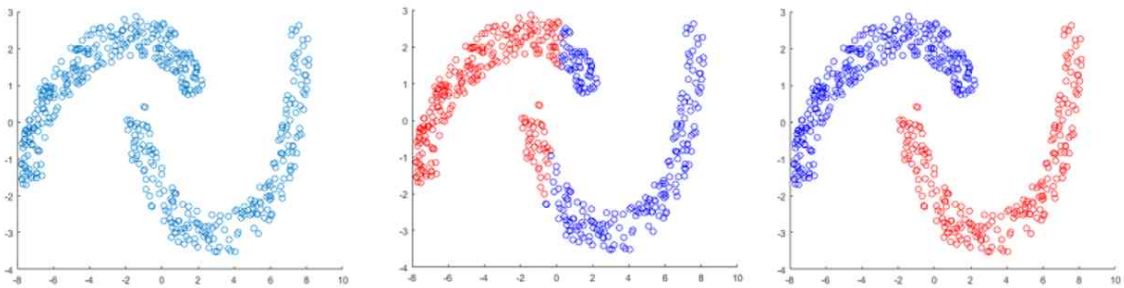
$$\|\mathbf{p}-\mathbf{q}\| = \sqrt{(\mathbf{p}-\mathbf{q})\cdot(\mathbf{p}-\mathbf{q})} = \sqrt{\|\mathbf{p}\|^2 + \|\mathbf{q}\|^2 - 2\mathbf{p}\cdot\mathbf{q}}.$$

② 맨하탄 거리(Manhattan distance)

$$d_1(\mathbf{p},\mathbf{q}) = \|\mathbf{p}-\mathbf{q}\|_1 = \sum_{i=1}^{n} |p_i - q_i|$$

(2) 군집분석 유형
① 분리형(비계층적) 군집화(Partitioning Clustering)
사전에 군집의 수를 정해주어 대상들이 군집에 할당되도록 하는 것, 대표적으로 K-Means algorithm이 있다.
①_1 중심 기반(Center-based clustering)
- 프로토타입 기반(Prototype-based)
- 동일한 군집에 속하는 데이터는 어떠한 중심을 기준으로 분포할 것이라는 가정을 기반
- 대표적으로 K-Means algorithm이 있다.

①_2 밀도 기반(Density-based clustering)
- 동일한 군집에 속하는 데이터는 서로 근접하게 분포할 것이라는 가정을 기반
- 대표적으로 DBSCAN algorithm이 있다.



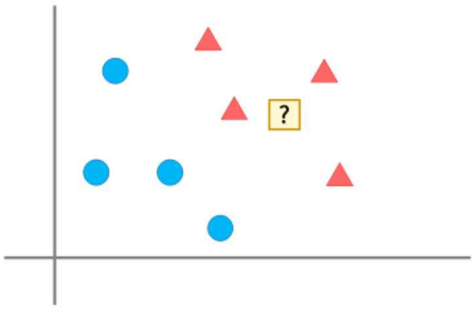
(a) 원본 데이터 (b) k-means clustering의 결과 (c) DBSCAN의 결과

② 계측정 군집화(Hierarchical Clustering)
각 객체가 n개(객체의 수)의 독립적인 각각의 군집에서 출발하여 점차 거리가 가까운 대상과 군집을 이루어 가는 것, 대표적으로 H-Clustering algorithm이 있다.

(3) 군집분석의 목적
- 유사한 성향을 가진 개체를 모아 군집을 형성하여 군집간의 특성을 관찰하거나 목표변수와의 관계를 파악하기 위함이다.
- 예측성 분석이 아니기 때문에 주로 분석 초기 단계에서 데이터의 특성을 파악하기 위해서 활용된다.

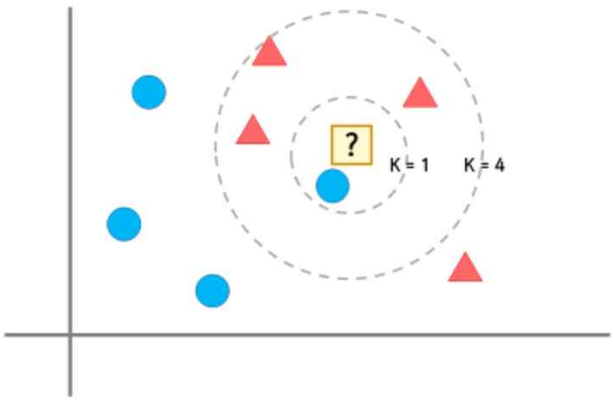
■ 최근접 이웃법 (Nearest Neighbor)

- 새로운 데이터를 입력 받았을 때 가장 가까이 있는 것이 무엇이냐를 중심으로 새로운 데이터의 종류를 정해주는 알고리즘



저 물음표에는 세모와 동그라미 중에 어떤게 들어갈까요?
최근접 이웃 알고리즘은 "?" 의 주변에 있는 것이 세모이기 때문에 세모라고 판단하는 알고리즘

- 단순히 주변에 무엇이 가장 가까이 있는가를 보는 것이 아니라 주변에 있는 몇 개의 것들을 같이 봐서 가장 많은 것을 골라내는 방식을 사용
- KNN 에서 K 는 주변의 개수를 의미



- K 가 1일때는 '?' 를 파란 동그라미라고 판단, K 를 4로 변경한다면 빨간 세모라고 판단
- 하지만 과연 K는 몇이어야 좋은 것일까. 최선의 k값을 선택하는 것은 데이터마다 다르게 접근해야한다.
- 일반적으로 k 값이 커질수록 분류에서 이상치의 영향이 줄어들고, 분류자체를 못하게 되는 상황이 발생한다.
- 일반적으로는 총데이터의 제곱근값을 사용

■ K-평균 군집화(K-means Clustering)

- 각 군집은 하나의 중심(centroid)을 가진다.
- 각 개체는 가장 가까운 중심에 할당되며, 같은 중심에 할당된 개체들이 모여 하나의 군집을 형성
- 사용자가 사전에 군집 수(k)가 정해야 알고리즘을 실행할 수 있다.
- k는 하이퍼파라메터(hyperparameter)를 뜻함

$$X = C_1 \cup C_2 \ldots \cup C_K, \quad C_i \cap C_j = \phi$$

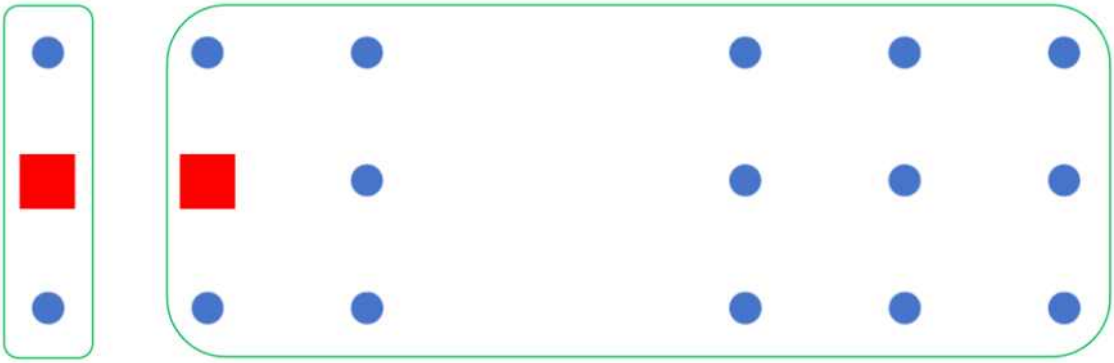
$$argmin_C \sum_{i=1}^{K} \sum_{x_j \in C_i} \|x_j - c_i\|^2$$

1) 학습과정

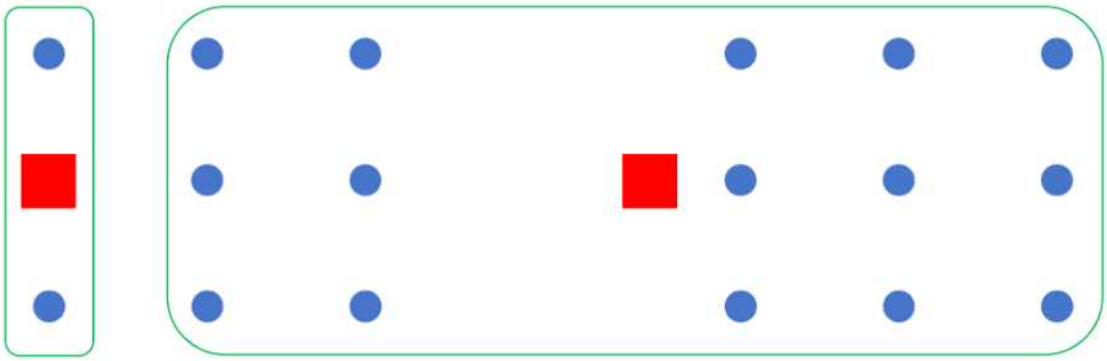
- KC는 수렴할 때까지 반복하는 방식
- 해를 찾기 어려운 문제를 풀 때 많이 사용되는 방법론
- KC의 경우 (1) 각 군집 중심의 위치 (2)각 개체가 어떤 군집에 속해야 하는지 멤버십, 이 두 가지를 동시에 찾아야 하기 때문에 EM 알고리즘을 적용

◑ 군집 수 k를 2, 군집의 중심(빨간색 점)을 랜덤 초기화

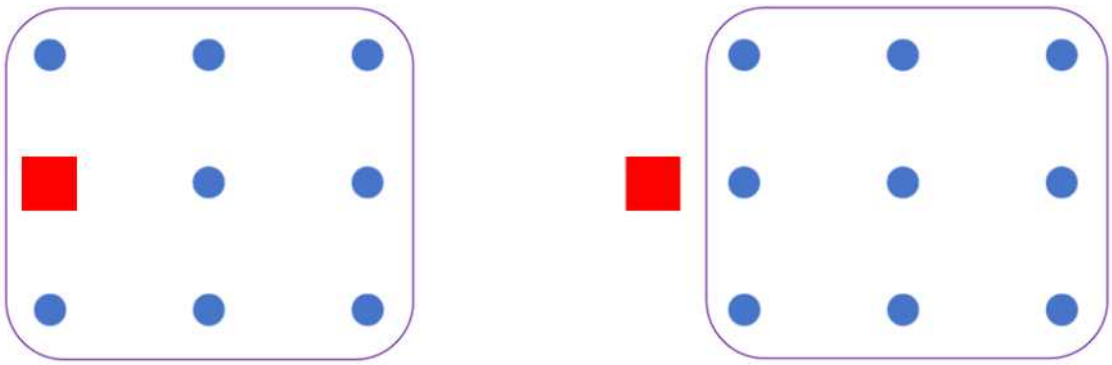
- Expectation 스텝; 모든 개체들(파란색 점)을 아래 그림처럼 가장 가까운 중심에 군집(녹색 박스)으로 할당



- Maximization 스텝; 중심을 군집 경계에 맞게 업데이트



- 다시 Expectation 스텝을 적용. 바꿔 말해 모든 개체들을 가장 가까운 중심에 군집(보라색 박스)으로 할당해주는 작업

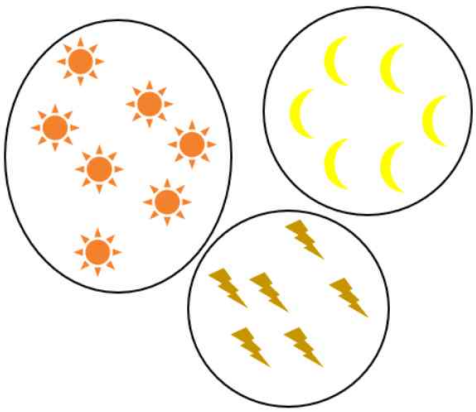


- Maximization 스텝을 또 적용해 중심을 업데이트
- Expectation과 Maximization 스텝을 반복 적용해도 결과가 바뀌지 않거나(=해가 수렴), 사용자가 정한 반복수를 채우게 되면 학습이 끝난다.

■ 계측 군집 분석(HCA, Hierarchical Cluster Analysis)

- 군집 분석은 기존 관측치의 Y값(Class)이 없는 데이터의 Clustering 알고리즘
- KNN 알고리즘과 동일한 거리 기반 모델로, 거리가 가까운 관측치들은 비슷한 특징을 가질 것이라는 전체하에 클러스터링을 수행하는 기법
- KNN과 차이점: 설명변수를 통한 예측 목적이 아닌 데이터 축소 목적
- 군집 분석에는 계층적인 방법(hierarchical clustering)과 비계층적인 방법(k-means)이

있다.
- 계층적인 방법은 가까운 대상끼리 순차적으로 군집을 묶어가는 것, 비계층적인 방법은 랜덤하게 군집을 묶어가는 것



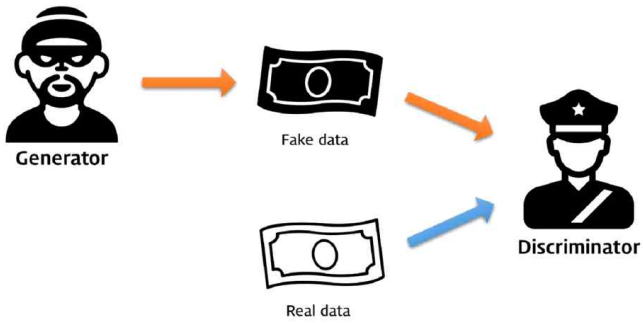
[GAN]

1.GAN이란

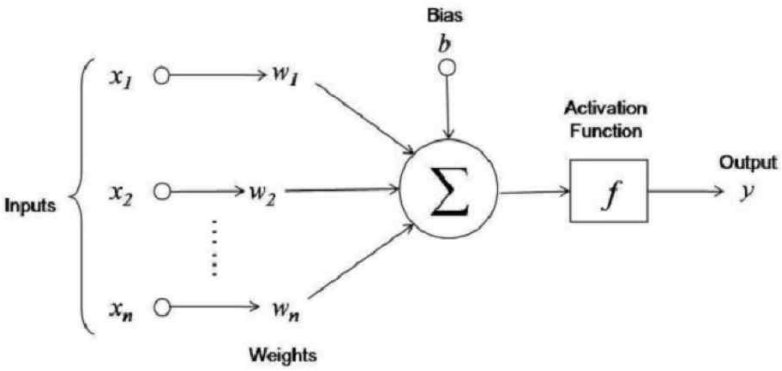
GAN은 'Generative Adversarial Network'의 약자다. 'Generative'는 GAN이 생성(Generation) 모델이라는 것을 뜻한다. 생성 모델이란 '그럴듯한 가짜'를 만들어내는 모델이다. 언뜻 보면 진짜 같은 가짜 사람 얼굴 사진을 만들어내거나 실제로 있을 법한 고양이 사진을 만들어내는 것이 생성 모델의 예다.

GAN의 두번째 단어인 'Adversarial'은 GAN이 두 개의 모델을 적대적(Adversarial)으로 경쟁시키며 발전시킨다는 것을 뜻한다. 위조지폐범과 경찰을 생각해보자. 이 둘은 적대적인 경쟁 관계다. 위조지폐범은 경찰을 속이기 위해 점점 지폐 위조 제조 기술을 발전시키고, 경찰은 위조지폐범을 잡기 위해 점점 위폐를 찾는 기술을 발전시킨다. 시간이 흐르면 위조지폐범의 위폐 제조 기술은 완벽에 가깝게 발전할 것이다.

이처럼 GAN은 위조지폐범에 해당하는 생성자(Generator)와 경찰에 해당하는 구분자(Discriminator)를 경쟁적으로 학습시킨다. 생성자의 목적은 그럴듯한 가짜 데이터를 만들어서 구분자를 속이는 것이며, 구분자의 목적은 생성자가 만든 가짜 데이터와 진짜 데이터를 구분하는 것이다. 이 둘을 함께 학습시키면서 진짜와 구분할 수 없는 가짜를 만들어내는 생성자를 얻을 수 있다. 이것이 GAN의 핵심적인 아이디어인 적대적 학습(Adversarial Training)이다.



GAN의 마지막 단어 '네트워크(Network)'는 이 모델이 인공신경망(Artificial Neural Network) 또는 딥러닝(Deep Learning)으로 만들어졌기 때문에 붙었다.



※ GAN 체험;

gan을 이용한 음악 생성)

gansynth_external.ipynb;

https://colab.research.google.com/notebooks/magenta/gansynth/gansynth_demo.ipynb#scrollTo=Vw9-tp6J5VV1

무료 MIDI 다운로드;

https://m.blog.naver.com/PostView.nhn?blogId=zin369_2&logNo=98725783&proxyReferer=https:%2F%2Fwww.google.com%2F

구글콜렙에서 체험)

gan외

https://magenta.tensorflow.org/demos/colab/