Sequence Modeling

Sequential Data Modeling

Sequential Data

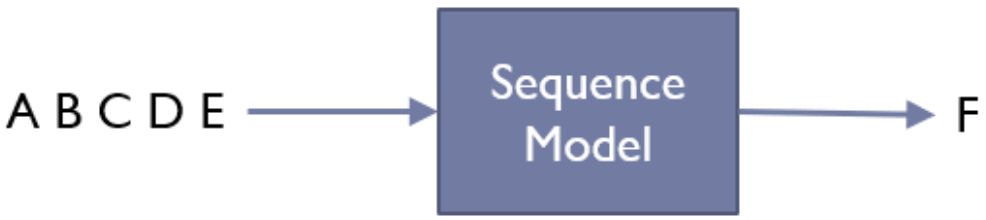
- 순차 데이터란 '데이터 집합 내의 객체들이 어떤 순서를 가진 데이터'로 그 순서가 변경될 경우 고유의 특성을 잃어버리는 특징이 있다.
- Most of data are sequential
- Speech,Text,Image,...

Deep Learnings for Sequential Data

- Convolution Neural Networks(CNN)
  - Try to find local features from a sequence
- Recurrent Neural Networks : LSTM,GRU
  - Try to capture the feature of the **past**

Three Types of Problems

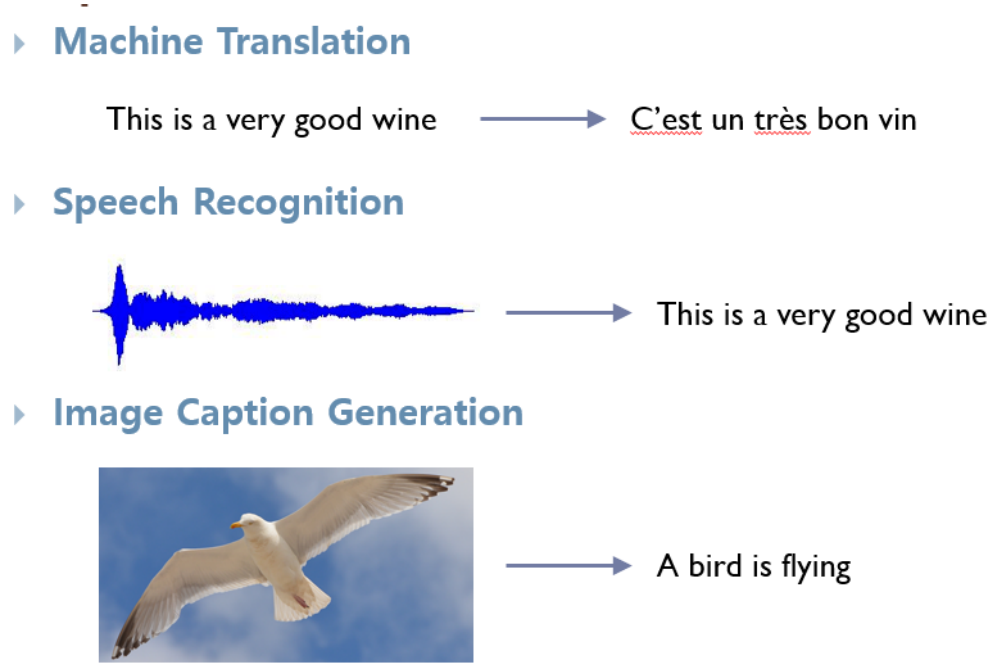
- Next Step Prediction



- Classification

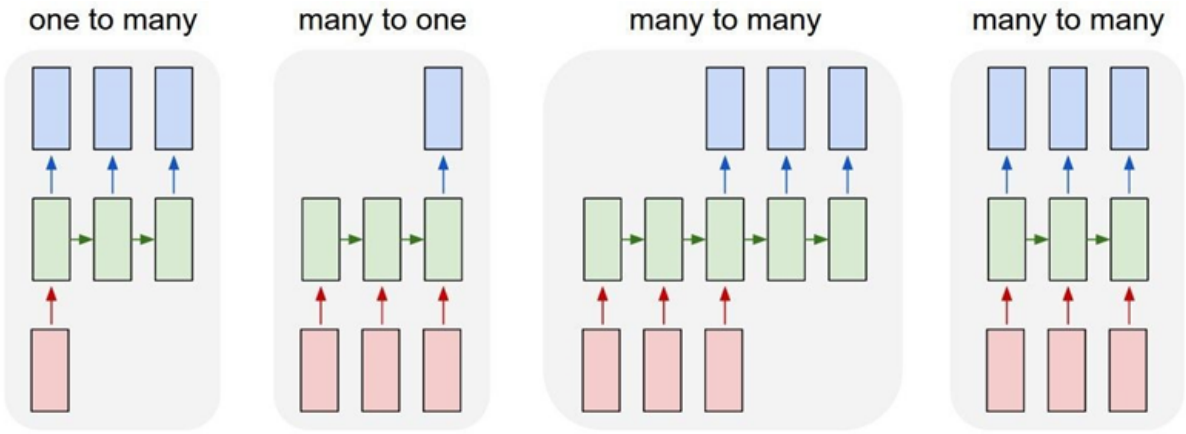
- Sequence Generation

- Machine Translation
- Speech Recognition
- Image Caption Generation



Types of Processes

- one to many>>Image Captiong :image→sequence of words
- many to one>>Sentiment Classification : sentence → sentiment
- many to many>>Machine Translation : sentence→sentence
- synched many to many>>Stock Price Prediction,Prediction of next word



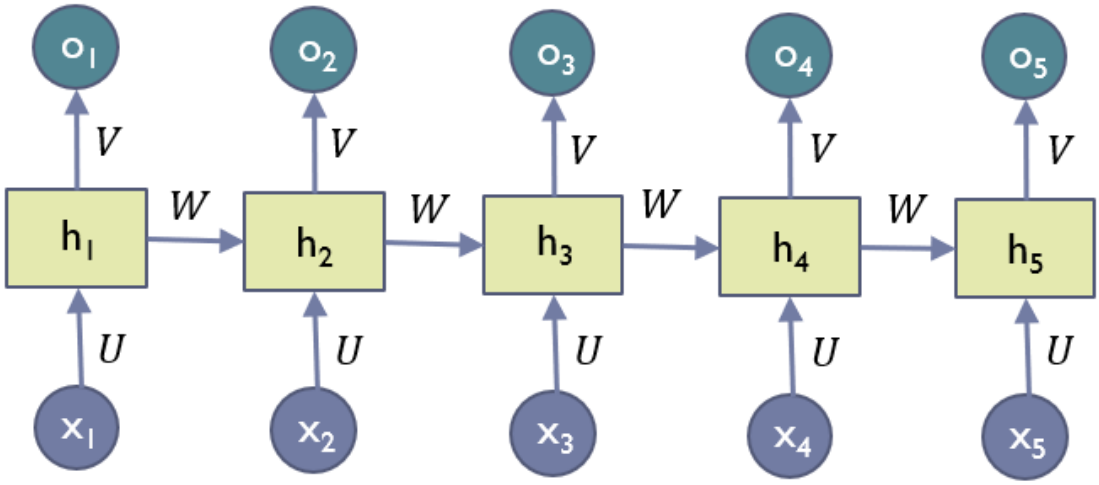
그런데 앞서 찾아본 RNN은 synched many to many구조이다. 위 type들은 결국 다 RNN이므로

synched many to many형태로 변환해야함!!

Synched Many to Many

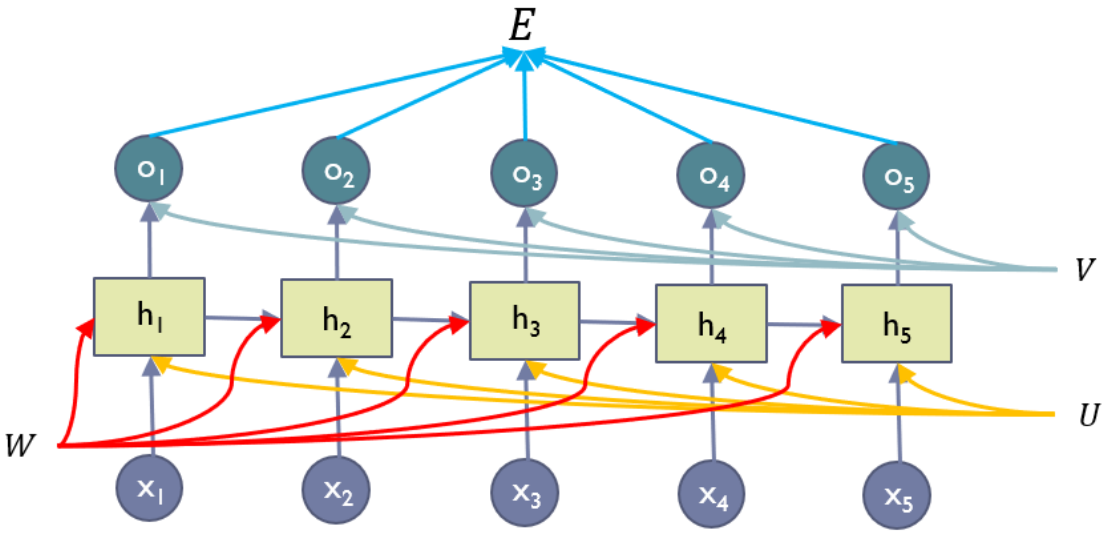
- Training

$$x_1 x_2 x_3 \cdots x_n \rightarrow y_1 y_2 y_3 \cdots y_n$$

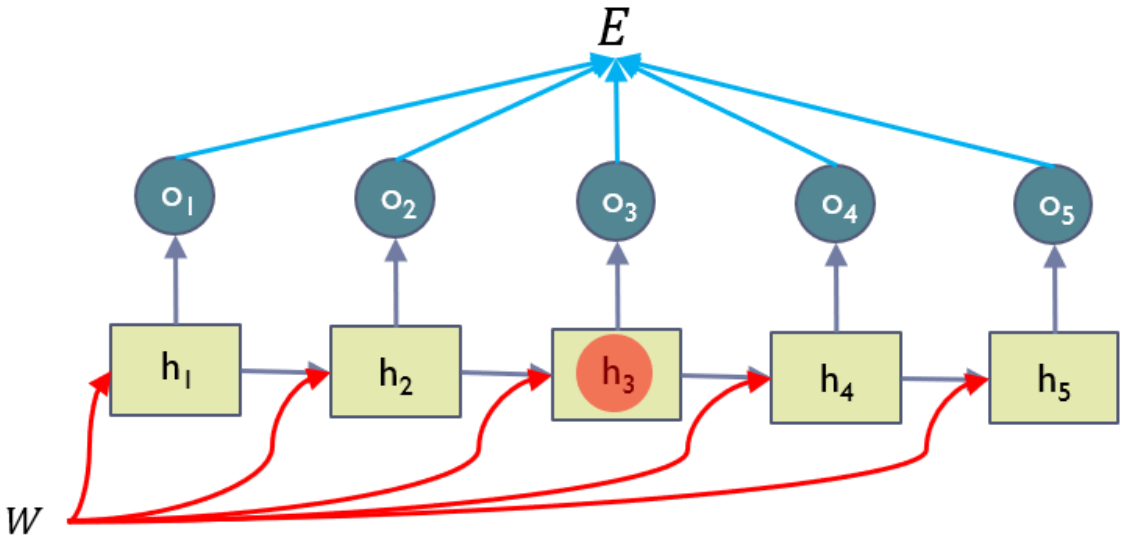


$$E = \sum_{i=1}^{n} (y_i - o_i)^2$$

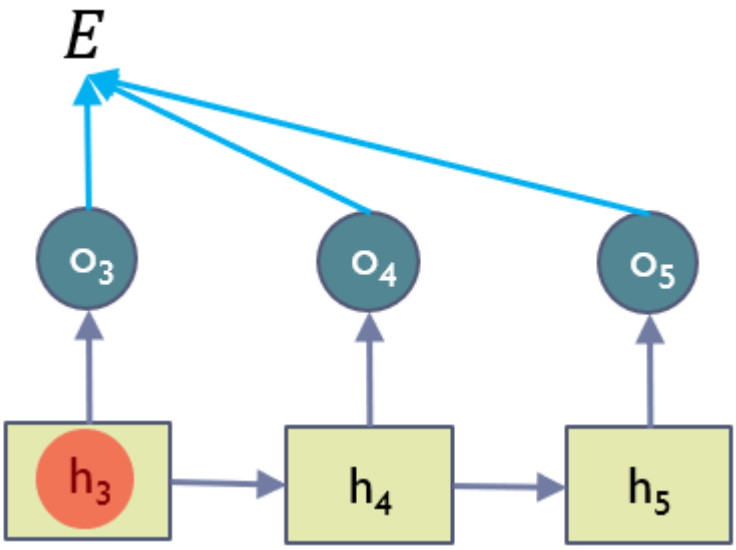
error function으로 cross-entropy도 가능!



$$\frac{\partial E}{\partial w} = ?$$



$$\frac{\partial E}{\partial w} = \sum_{i=1}^{n} \frac{\partial E}{\partial h_i} \frac{\partial h_i}{\partial w}$$



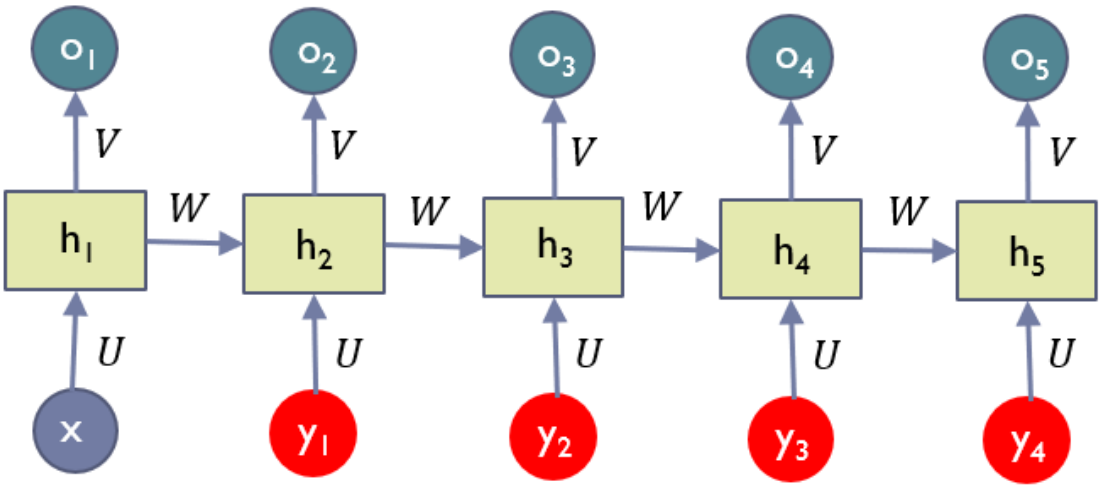
$$\frac{\partial E}{\partial h_i} = \frac{\partial E}{\partial o_i}\frac{\partial o_i}{\partial h_i} + \frac{\partial E}{\partial h_{i+1}}\frac{\partial h_{i+1}}{\partial h_i}$$

$$\frac{\partial E}{\partial w} = \sum_{i=1}^{n} \frac{\partial E}{\partial h_i}\frac{\partial h_i}{\partial w}$$

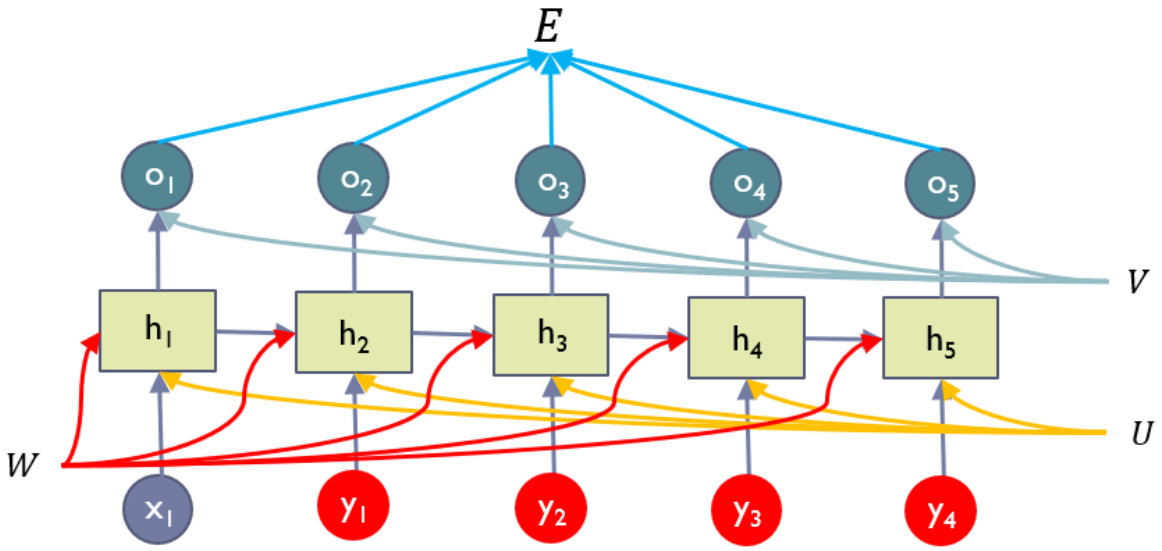
One to Many

- Training

$$x \rightarrow y_1 y_2 y_3 \cdots y_n$$



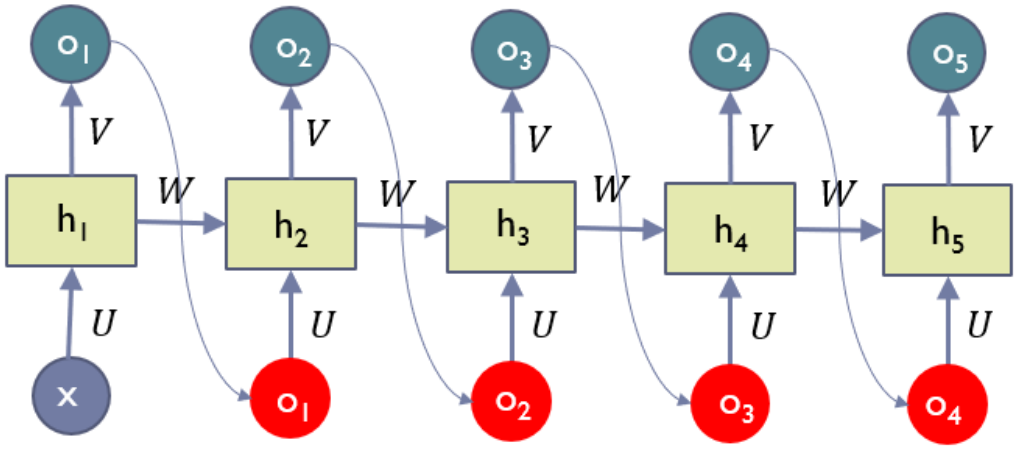
$$E = \sum_{i=1}^{n} (y_i - o_i)^2$$



$$\frac{\partial E}{\partial w} = \sum_{i=1}^{n} \frac{\partial E}{\partial h_i} \frac{\partial h_i}{\partial w} \qquad \frac{\partial E}{\partial h_i} = \frac{\partial E}{\partial o_i} \frac{\partial o_i}{\partial h_i} + \frac{\partial E}{\partial h_{i+1}} \frac{\partial h_{i+1}}{\partial h_i}$$

- Testing

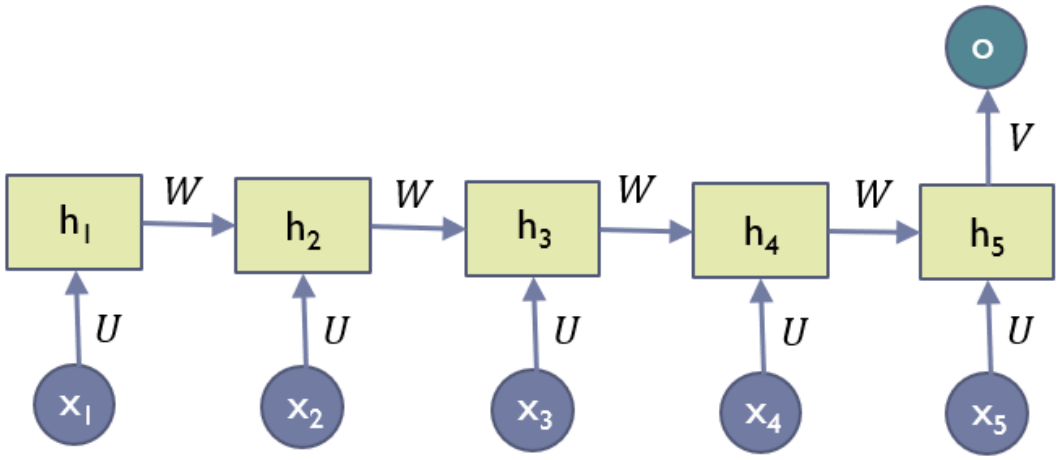
$$x \rightarrow ? ? ? ? ? ?$$



Many to One

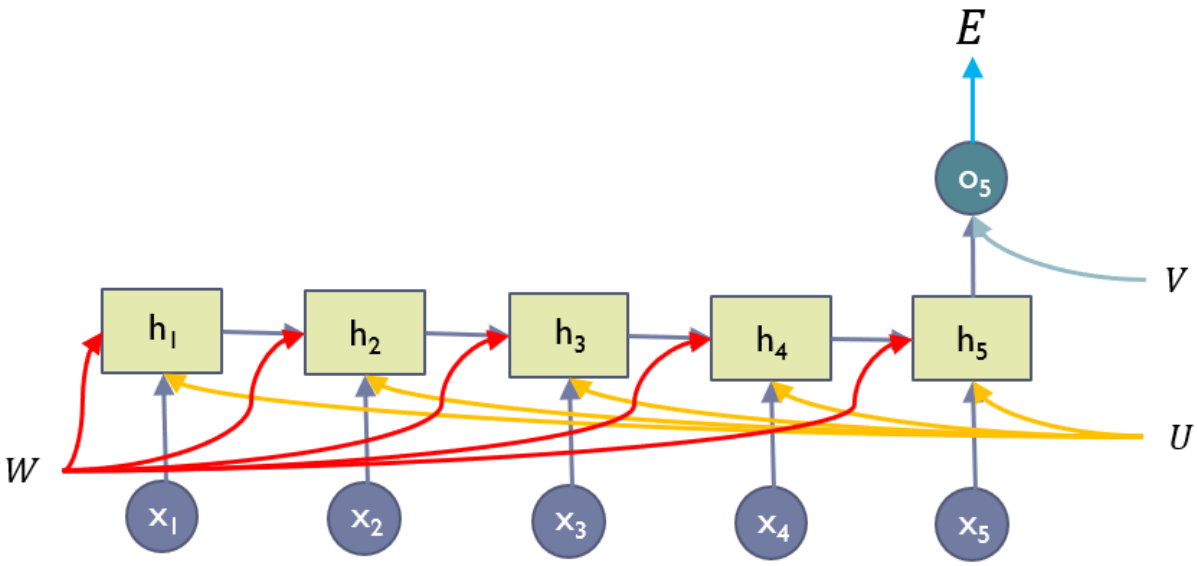
- Training

$$x_1 x_2 x_3 \cdots x_n \rightarrow y$$



$$E = (y - o)^2$$

h1,h2,h3,h4에서 출력값이 안나오는것이 아니다. 다만 안쓸뿐!

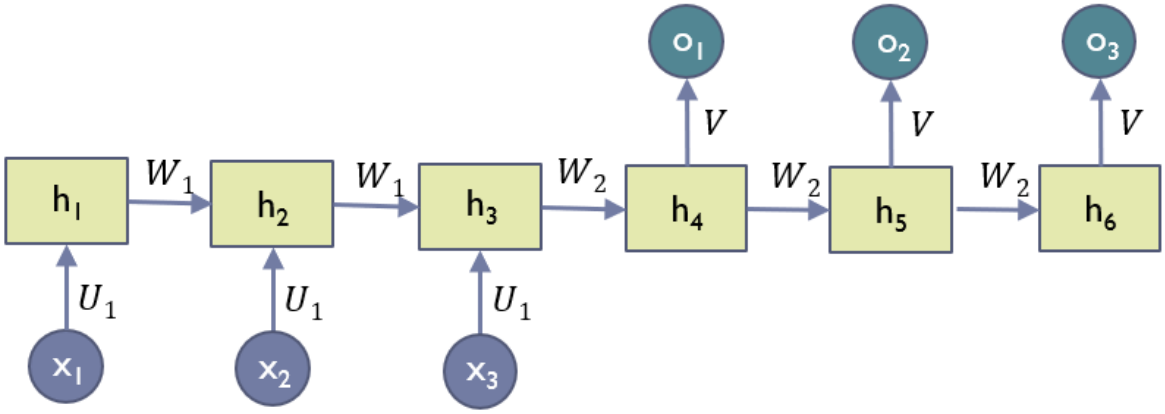


$$\frac{\partial E}{\partial w} = \sum_{i=1}^{n} \frac{\partial E}{\partial h_i} \frac{\partial h_i}{\partial w} \qquad \frac{\partial E}{\partial h_i} = \frac{\partial E}{\partial h_{i+1}} \frac{\partial h_{i+1}}{\partial h_i}$$

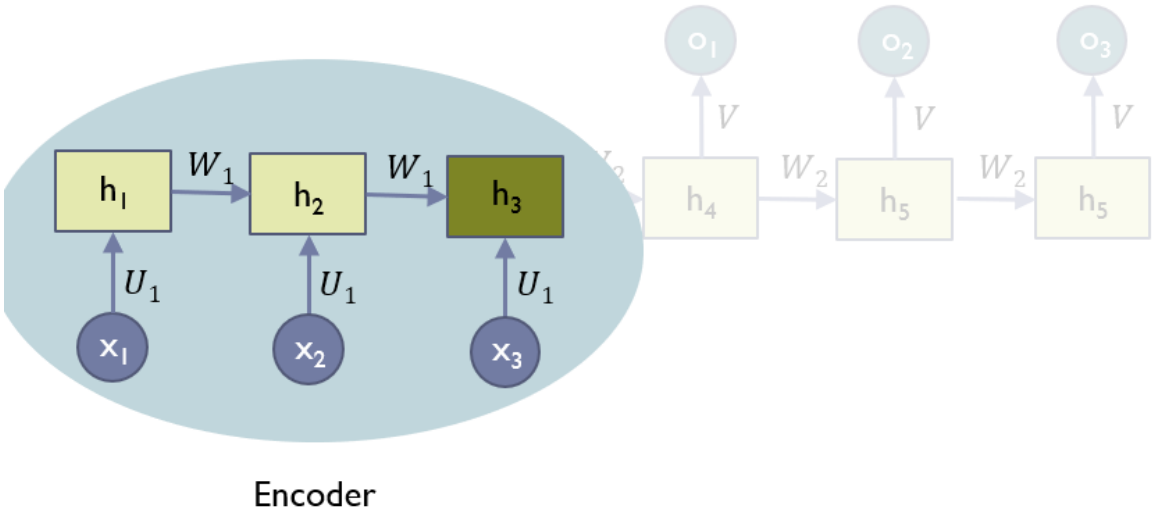
Many to Many

- Training

$$x_1 x_2 x_3 \cdots x_n \rightarrow y_1 y_2 y_3 \cdots y_n$$

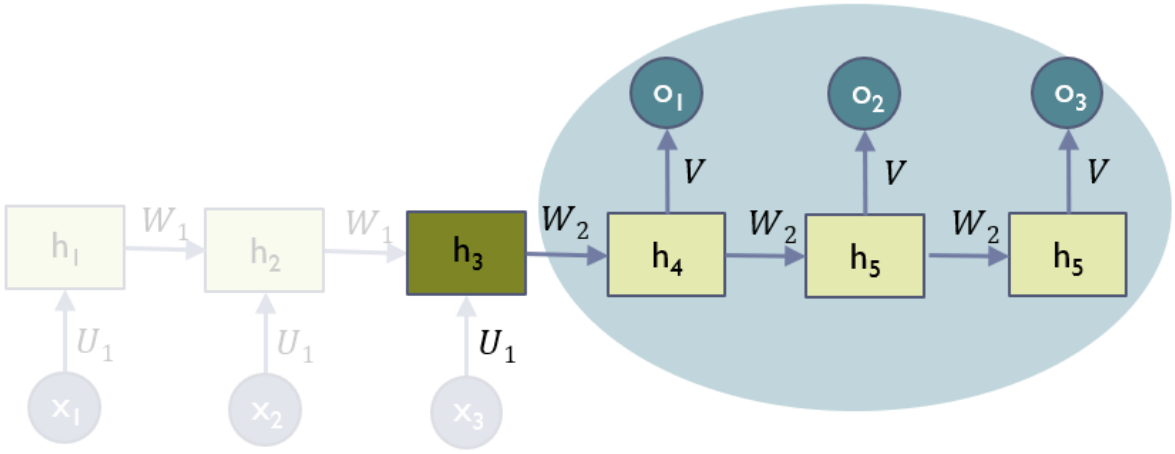


$$x_1 x_2 x_3 \cdots x_n \rightarrow y_1 y_2 y_3 \cdots y_n$$



encoder : many to one

$$x_1 x_2 x_3 \cdots x_n \rightarrow y_1 y_2 y_3 \cdots y_n$$



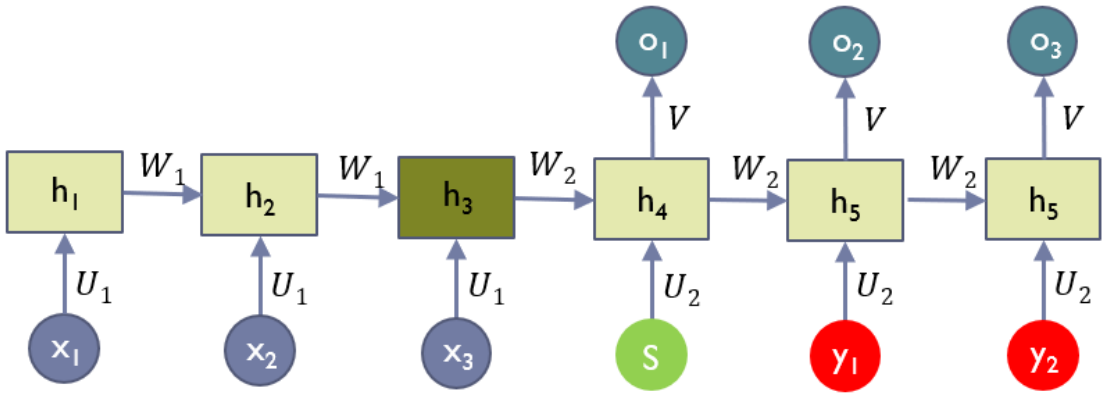
Decoder

Encoder

decoder : one to many

$$E = \sum_{i=1}^{n} (y_i - o_i)^2$$

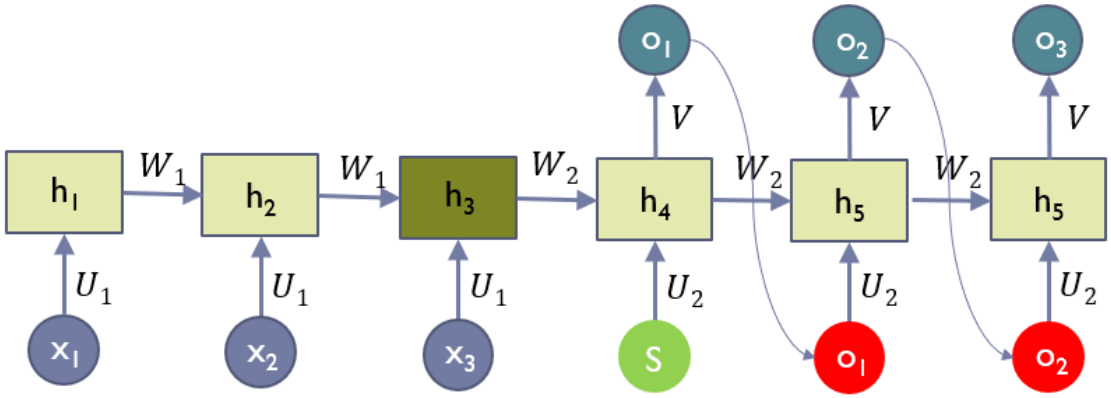
$$x_1 x_2 x_3 \cdots x_n \rightarrow y_1 y_2 y_3 \cdots y_n$$



$$\frac{\partial E}{\partial w} = ??$$ Combination of [Many to One] and [One to Many]

- Testing

$$x_1 x_2 x_3 \cdots x_n \rightarrow ?????$$

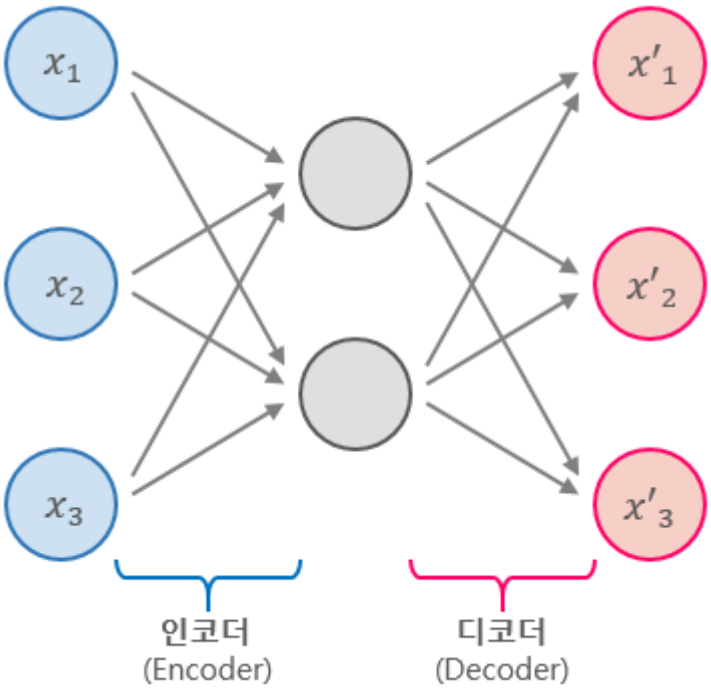


many to many 구조에서 encoder,decoder이 쓰였는데 이게 뭘까??>>AutoEncoder

AutoEncoder?

오토인코더는 아래의 그림과 같이 단순히 입력을 출력으로 복사하는 신경망이다. 어떻게 보면 간단한 신경망처럼 보이지만 네트워크에 여러가지 방법으로 제약을 줌으로써 어려운 신경망으로 만든다.

예를들어 아래 그림처럼 hidden layer의 뉴런 수를 input layer(입력층) 보다 작게해서 데이터를 압축(차원을 축소)한다거나, 입력 데이터에 노이즈(noise)를 추가한 후 원본 입력을 복원할 수 있도록 네트워크를 학습시키는 등 다양한 오토인코더가 있다. 이러한 제약들은 오토인코더가 단순히 입력을 바로 출력으로 복사하지 못하도록 방지하며, 데이터를 효율적으로 표현(representation)하는 방법을 학습하도록 제어한다.



https://excelsior-cjh.tistory.com/187

Autoencoder는 위의 그림에서 볼 수 있듯이 항상 encoder와 decoder 두 부분으로 구성되어 있다.

- Encoder : 인지 네트워크(recognition network)라고도 하며, 입력을 내부 표현으로 변환한다.
- Decoder : 생성 네트워크(generative network)라고도 하며, 내부 표현을 출력으로 변환한다.

오토인코더는 위의 그림에서 처럼, 입력과 출력층의 뉴런 수가 동일하다는 것만 제외하면 일반적인 MLP와 동일한 구조이다. 오토인코더는 입력을 재구성하기 때문에 출력을 재구성(reconstruction)한다고 하며, loss function는 입력과 재구성(출력)의 차이를 가지고 계산한다.

위 그림의 오토인코더는 히든 레이어의 뉴런(노드, 유닛)이 입력층보다 작으므로 입력이 저차원으로 표현되는데, 이러한 오토인코더를

Undercomplete Autoencoder라고 한다. undercomplete 오토인코더는 저차원을 가지는 히든 레이어에 의해 입력을 그대로 출력으로 복사할 수 없기 때문에, 출력이 입력과 같은 것을 출력하기 위해 학습해야 한다. 이러한 학습을 통해 undercomplete 오토인코더는 입력 데이터에서 가장 중요한 특성(feature)을 학습하도록 만든다.

그렇다면 autoencoder는 무엇일까? 입력과 출력을 같도록하는 구조를 말한다. 노이즈 제거에 탁월하며, unsupervsied learning 방법 중 하나이다. 입력값에 특별히 label 혹은 정답 데이터가 따로 있는 것이 아니라, 입력값 그 자체가 정답이되므로 supervised되는 것이 없다. 따라서 unsupervsied 이지만, 좀 더 엄밀히 말하자면 self-supervised learning이다. 입력값이 정답이 되므로, 스스로 답을 주는 self-supervised라고 할 수 있다.

그렇다면 오토인코더가 갖는 의미는 무엇일까?

1. 모델이 데이터를 바라보는, 이해하는 시각/표현을 latent vector를 통해 얻을 수 있다.

일반적으로 **인코더는 차원을 축소하기 때문에 입력받는 것을 문제를 해결하기 위한 어떠한 벡터로 변환된다**. 결국 풀고자 하는 문제는 입력값을 출력값으로 되돌리는 것이기 때문에, 그 벡터는 입력값에 대한 정보를 최대한 압축하듯이 잘 담아낸다. 이러한 값은 representation, (latent) feature, embedding 등의 용어로 혼용된다. latent vector를 잘 생성하고 활용하는 것은 이후에 이 정보를 이용해 어떠한 작업을 하는지에 영향을 미치기 때문에 마치 데이터를 가공하는 전처리과정처럼 매우 중요하다.

2. 오토인코더에서 decoder 부분은 특정한 vector로부터 학습된 형태로 변환(복원)하는 기능을 갖는다.

일반적으로 autoencoder는 encoder와 decoder가 대칭적인 구조를 갖는다. 위에서 생성/추출한 latent vector를 이용해 원하는 결과 이미지를 생성해낼 수 있다. 이러한 '생성'의 측면으로 보았을때, decoder는 generator가 될 수 있는 것이다. 대표적인 generative 모델로는 GAN이 있지만, VAE도 있다.

autoencoder는 linear autoencoder, 각 단계의 weight를 초기화하는 Stacking Autoencoder, 학습 데이터에 노이즈를 추가한 Denoising AutoEncoder(DAE), regularizer를 추가한(=noise 역할) Stochastic Contractive Autoencoder(SCAE), 테일러 전개를 이용해서 stochastic을 deterministic하게 바꾼 Contractive Autoencoder(CAE), latent variable을 찾고자 만들어진 Variational Autoencoder(VAE)

Linear Autoencoder는 비선형의 activation function이 없어서, 기존의 머신러닝(PCA)으로도 매칭이 된다.

원-핫 인코딩은 단어간 유사성을 계산할 수 없다는 단점이 있다.&고차원,sparse representation

이 문제를 해결하고자 word embedding

워드 임베딩은 단어를 벡터로 표현하는 방법으로, 단어를 밀집 표현으로 변환한다.

sparse Representation(희소 표현)

서 원-핫 인코딩을 통해서 나온 원-핫 벡터들은 표현하고자 하는 단어의 인덱스의 값만 1이고, 나머지 인덱스에는 전부 0으로 표현되는 벡터 표현 방법이었습니다. 이렇게 벡터 또는 행렬(matrix)의 값이 대부분이 0으로 표현되는 방법을 희소 표현(sparse representation)이라고 합니다. 그러니까 원-핫 벡터는 희소 벡터(sparse vector)입니다.

이러한 희소 벡터의 문제점은 단어의 개수가 늘어나면 벡터의 차원이 한없이 커진다는 점입니다. 원-핫 벡터로 표현할 때는 갖고 있는 코퍼스에 단어가 10,000개였다면 벡터의 차원은 10,000이어야만 했습니다. 심지어 그 중에서 단어의 인덱스에 해당되는 부분만 1이고 나머지는 0의 값을 가져야만 했습니다. 단어 집합이 클수록 고차원의 벡터가 됩니다. 예를 들어 단어가 10,000개 있고 강아지란 단어의 인덱스는 5였다면 원 핫 벡터는 이렇게 표현되어야 했습니다.

Ex) 강아지 = [0 0 0 0 1 0 0 0 0 0 0 0 ... 중략 ... 0] # 이 때 1 뒤의 0의 수는 9995개.

이러한 벡터 표현은 공간적 낭비를 불러일으킵니다. 잘 생각해보면, **공간적 낭비**를 일으키는 것은 원-핫 벡터뿐만은 아닙니다. 희소 표현의 일종인 DTM과 같은 경우에도 특정 문서에 여러 단어가 다수 등장하였으나, 다른 많은 문서에서는 해당 특정 문서에 등장했던 단어들이 전부 등장하지 않는다면 역시나 행렬의 많은 값이 0이 되면서 공간적 낭비를 일으킵니다. 뿐만 아니라, 원-핫 벡터는 **단어의 의미를 담지 못한다는 단점을 갖고있습니다.**

Dense Representation(밀집 표현)

이러한 희소 표현과 반대되는 표현이 있으니, 이를 밀집 표현(dense representation)이라고 합니다. 밀집 표현은 벡터의 차원을 단어 집합의 크기로 상정하지 않습니다. 사용자가 설정한 값으로 모든 단어의 벡터 표현의 차원을 맞춥니다. 또한, 이 과정에서 더 이상 0과 1만 가진 값이 아니라 실수값을 가지게 됩니다. 다시 희소 표현의 예를 가져와봅시다.

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

예를 들어 10,000개의 단어가 있을 때 강아지란 단어를 표현하기 위해서는 위와 같은 표현을 사용했습니다. 하지만 밀집 표현을 사용하고, 사용자가 밀집 표현의 차원을 128로 설정한다면, 모든 단어의 벡터 표현의 차원은 128로 바뀌면서 모든 값이 실수가 됩니다.

Ex) 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 합니다.

word Embedding(워드 임베딩)

단어를 밀집 벡터(dense vector)의 형태로 표현하는 방법을 **워드 임베딩(word embedding)**이라고 합니다. 그리고 이 밀집 벡터를 워드 임베딩 과정을 통해 나온 결과라고 하여 **임베딩 벡터(embedding vector)**라고도 합니다.

워드 임베딩 방법론으로는 LSA, Word2Vec, FastText, Glove 등이 있습니다. 케라스에서 제공하는 도구인 Embedding()는 앞서 언급한 방법들을 사용하지는 않지만, 단어를 랜덤한 값을 가지는 밀집 벡터로 변환한 뒤에, 인공 신경망의 가중치를 학습하는 것과 같은 방식으로 단어 벡터를 학습하는 방법을 사용합니다.

아래의 표는 앞서 배운 원-핫 벡터와 지금 배우고 있는 임베딩 벡터의 차이를 보여줍니다.

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