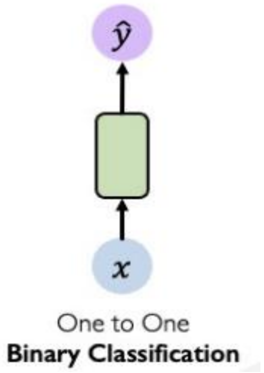
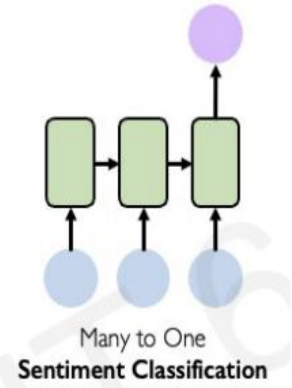
Sequence Modeling

A training system to make predictions by estimating the probability distribution over the next values, given the sequence of past context



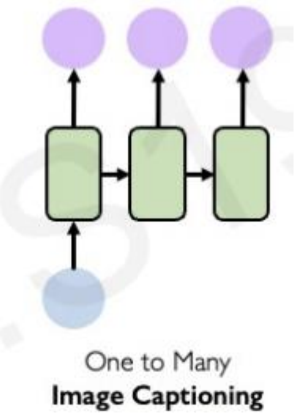
1. One to One

- binary classification
- single input, generate a single output
- based on the lecture from week 1



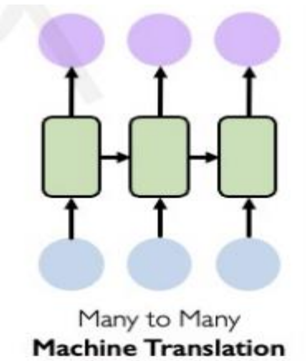
2. Many to One

- sentiment classification
- many inputs, generate a single output
- input: claim, your, winning, prize -> output: spam



3. One to Many

- image captioning
- one input, generates many outputs
- input: an image -> output: an explanation of the image

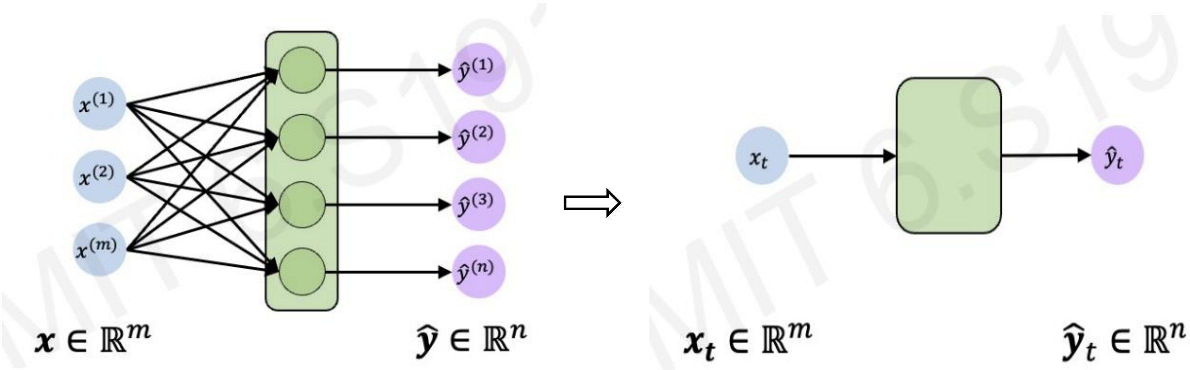


4. Many to Many

- machine translation
- many inputs, many outputs
- input: a sentence -> output: a sentence in another language.

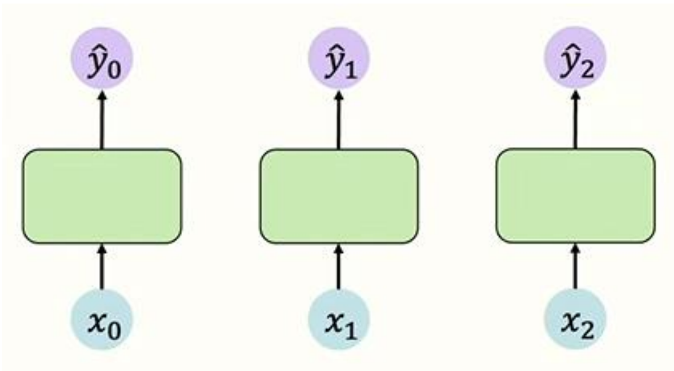
Neurons with Recurrence

Simplify the previous Perceptron layers for convenience



Sequence data: data over time t=0, 1, 2, …

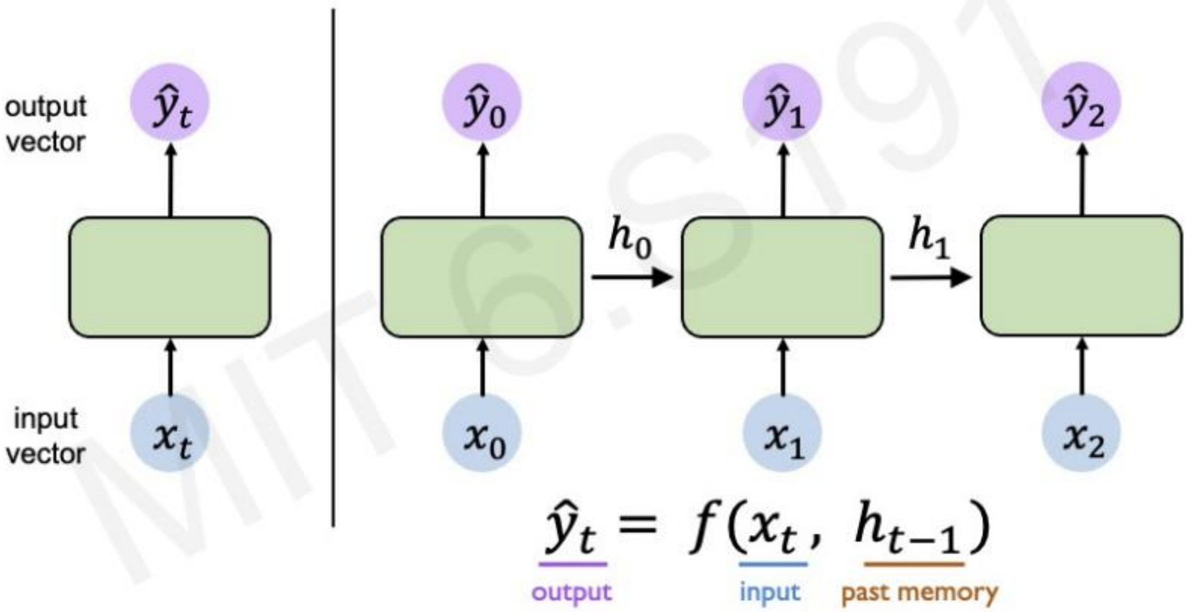
Let's take the given model and apply it over time



- isolated: no linking between time steps

- sequence model should have dependence: want y_2 to depend on the input x_0 and x_1

Introduction of variable h, representing state of the neural network



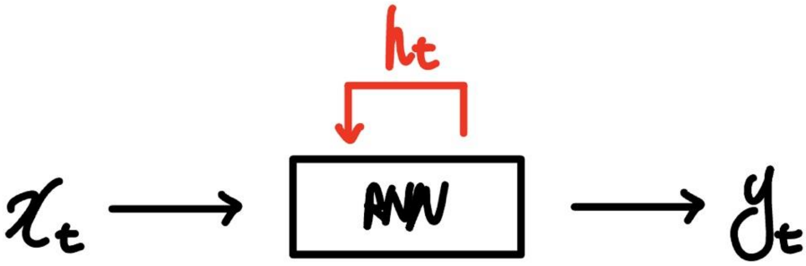
- h is learned and computed by the neuron and passed to the next sequence

- Now the output depends on the input and the state from prior time step

Recurrent neural networks (RNNs)

RNNs apply the same network to each element in a sequence

- Preserving and passing on relevant information
- enable the machine to learn temporal dependencies that simple network cannot



h_t : cell State, 해당 RNN에서 학습되어 계산된 변수

$$= f_w(\underbrace{x_t}_{\text{input}}, \underbrace{h_{t-1}}_{\text{t-1 시점의 State}})$$

- here, f_w is the function that gives the specific weight as we did in week 1
- same function, and same weight

Take a look at h_t more

In time step t

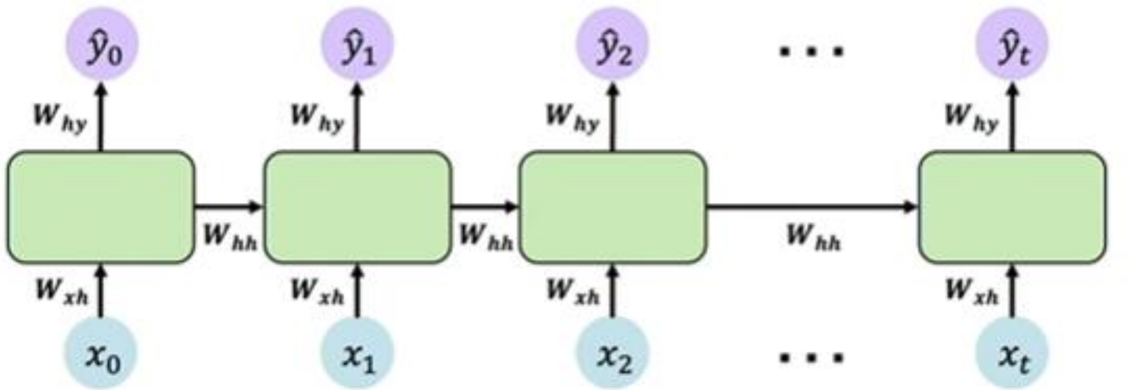
$$h_t = \tanh(W_{hh}^T h_{t-1} + W_{xh}^T x_t)$$

1. W_{hh}^T : h_{t-1}을 위한 가중치 벡터
2. W_{xh}^T : x_t(입력)를 위한 가중치 벡터

- tanh is an activating function, we apply this for non-linearity; however, doesn't have to be tanh it could be sigmoid, hyperbolic etc.

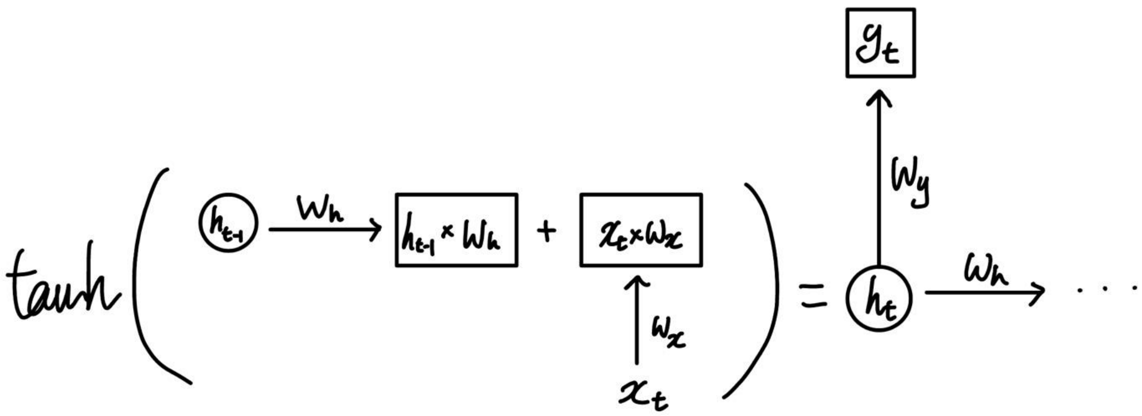
Finally, we will get the output by multiplying the calculated h_t and weight

Unfolding RNNs



- same weight at every step

+) Handwriting of personal study



모든 t에 대해 가중치 W_h, W_x, W_y는 동일
bcz 똑같은 퍼셉트론을 적용하고 있기 때문

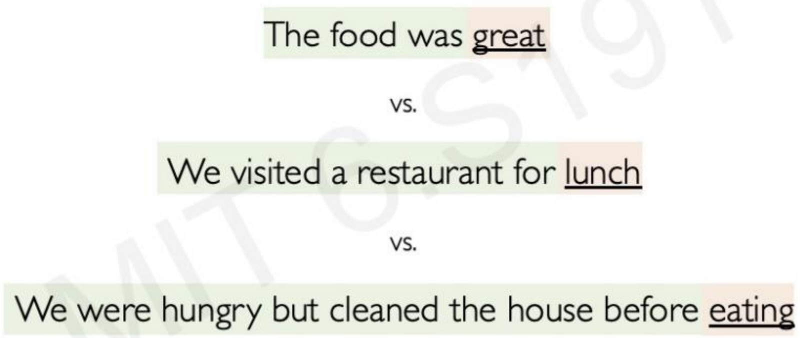
To compute predictions, we need more than this

- to update the weights, getting the loss of each layer by backpropagation is critical
- the total value of the loss L: sum of each loss

Sequence Modeling: designing criteria & example

There are four conditions to make a flexible sequence model

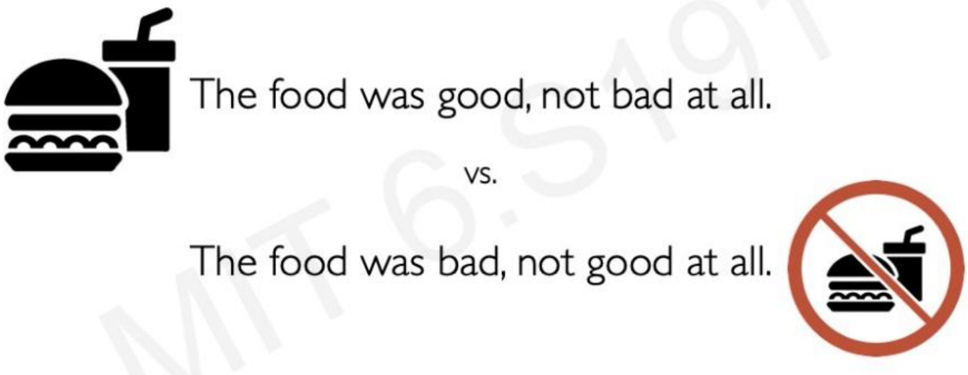
- Handle variable-length sequence: the length of the word can be varied



- Track long-term dependencies: tracking dependencies is important

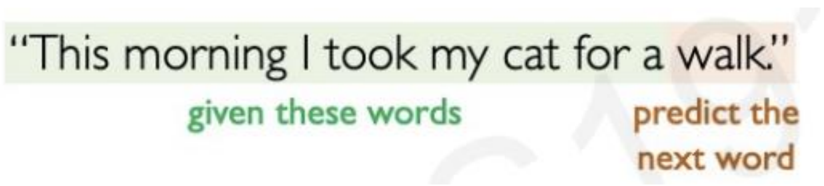
  -> we need information from the past sometimes

- maintain information about order: preserving inherent order is crucial



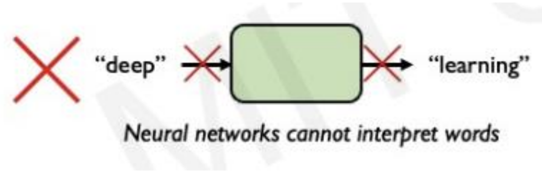
- share parameters across the sequence

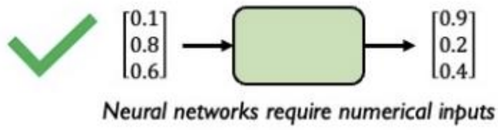
Example: predict the next word



-> Predicting the word "walk" is our goal

As neural networks cannot interpret the words of people, we need to translate them into numerical inputs





how to encode natural language into numerical inputs

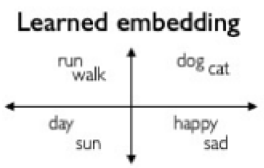
- Embedding: transform indexes into a vector of fixed size

1. make corpus: putting words together
2. index: giving a number to each word
3. embedding: index to fixed-sized vector, two methods are introduced

3-1. One-hot embedding

- each column represents one unique category.

- a vector representation that assigns a value of 1 to the index of the word you want to represent, and 0 to any other index.



3-2. Learned embedding

- categorize the words with similar meanings so that they can be represented in similar numerical values