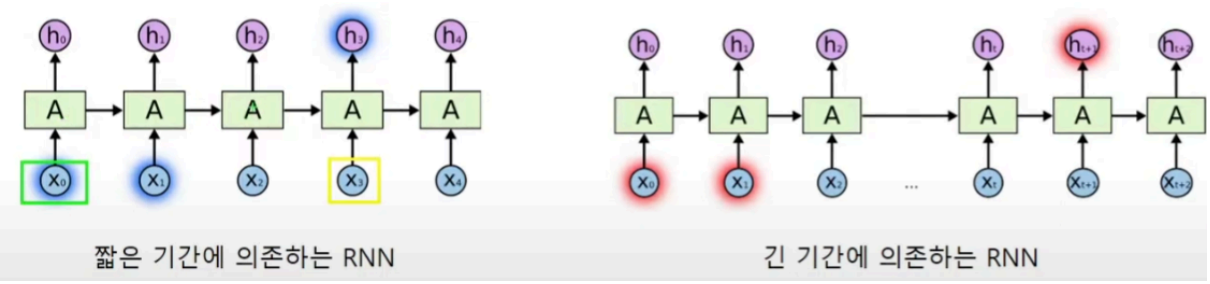
LSTM

https://www.youtube.com/watch?v=qKgqrdrf224

장기 의존 관계 (Long-term Dependency)

- 바닐라 인공 신경망은 input data: label 관계를 parameter에 학습
- RNN은 input data, hidden state: label 관계를 parameter에 학습
- hidden state가 담고 있는 오래전 정보와의 관계를 학습하려면 gradient가 시간 방향으로 적절히 전달되어야



The clouds are in the *sky*.

I grew up in France...I speak fluent *French*.

RNN의 장기 의존 관계 문제

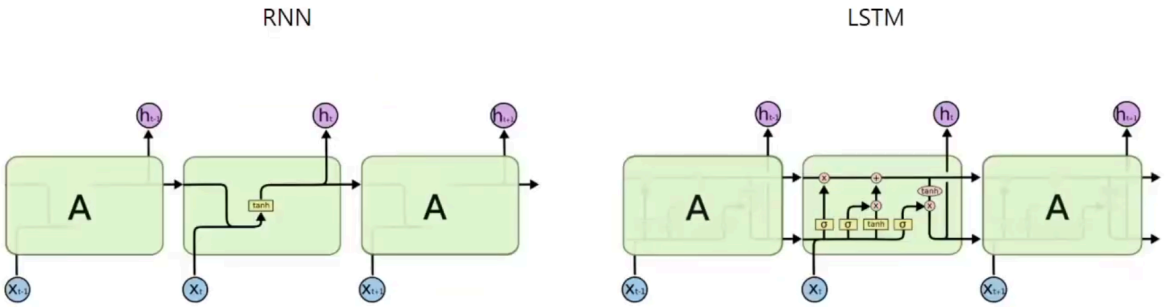
- RNN에서는 장기 의존 관계 학습하는 데에 vanishing gradient 문제가 있어 기울기 소멸되거나 발산하는 문제가 있다. → 학습이 제대로 되지 않음

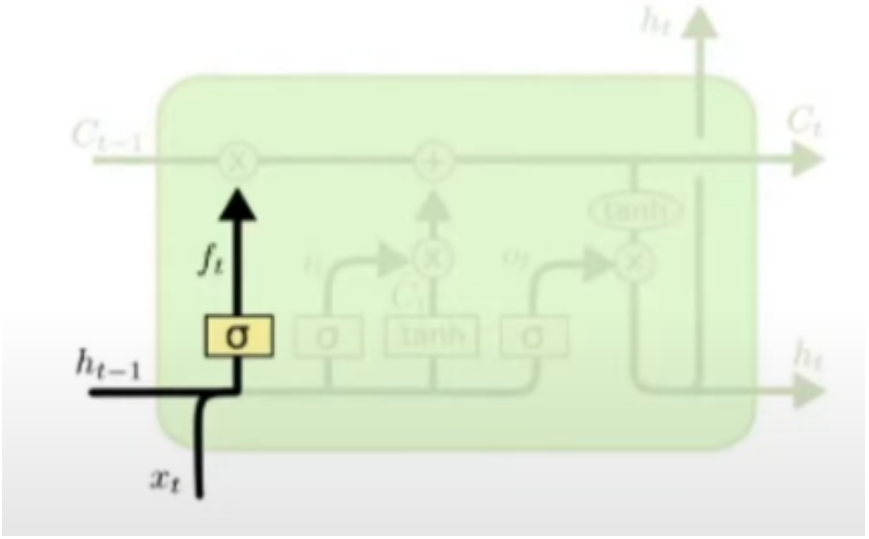
Gradient Clipping

- gradient divergence(발산) 문제를 해결
- 임계값을 설정하고 gradient의 norm이 임계값을 넘어가면 gradient를 normalize한 후 임계값을 곱해서 norm이 임계값이 되도록 한다.

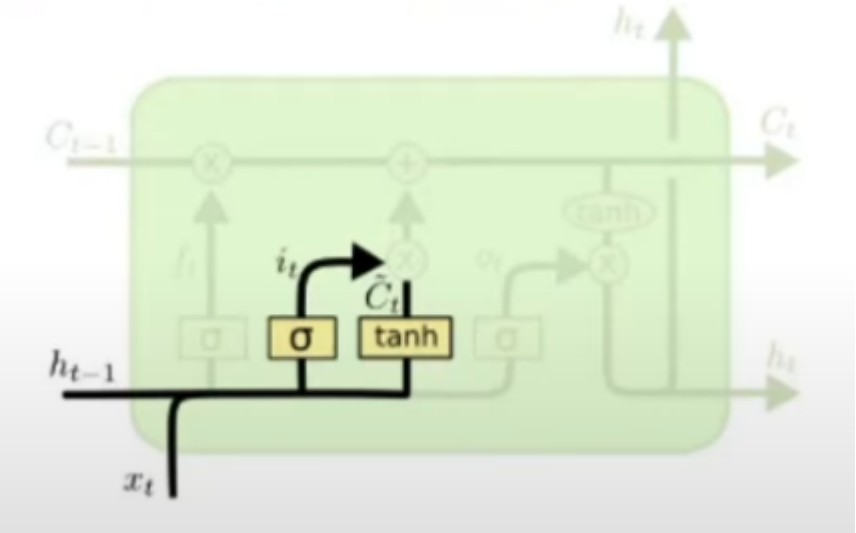
LSTM (Long Short-Term Memory)

- RNN: input and hidden state를 받음
  - RNN은 앞 정보의 기억을 위해 hidden state를 출력 시킬 뿐 아니라 순환
- LSTM: input, hidden state, cell state를 받음
  - LSTM은 hidden state 뿐 아니라 cell state도 순환
  - 단, cell state는 출력시키지 않음
  - 3개의 gate는 0~1사이의 값을 취하며 벨브의 역할을 함
  - gate는 sigmoid를 통해 얻고, 정보는 hyperbolic tangent를 통해 얻음
  - 3개의 gate는 기존 cell state의 정보를 얼마나 잊을지 (forget gate), 기존 cell state에 얼마나 정보를 추가할지 (input gate), 새로 얻은 cell state의 정보를 hidden state로 얼마나 출력할지 (output state)를 결정

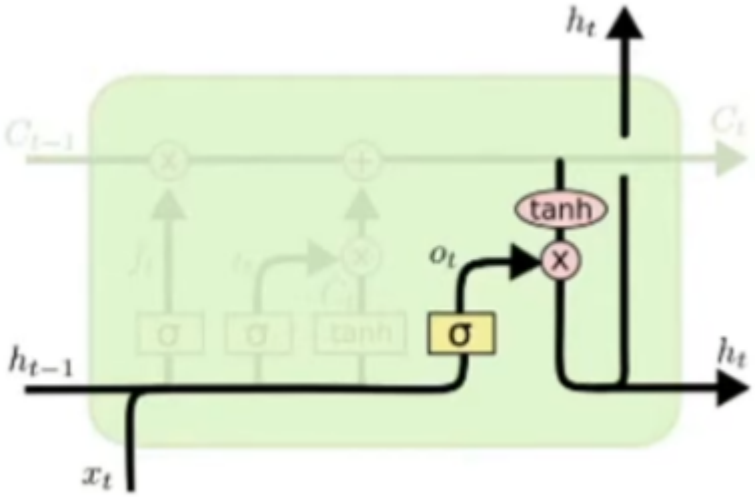




forget gate



input gate



output gate

STUDY Project 관련

- Many-to-one type
- Preporcessing
  - 주식 가격을 학습 및 테스트 데이터로 사용
- Optimization
- Validation 과정 생략
- Test

- 2/12 Coding Test Homework

```
from collections import deque
import sys # speed up
input = sys.stdin.readline
# print = sys.stdout.write

N, M = map(int, input().split()) # Maze size
A = [[0 for i in range(M+1)] for i in range(N+1)] # Maze graph init
visited = [[False for i in range(M+1)] for i in range(N+1)] # visited list init
path = [] # path list

# Maze graph assignment
for i in range(1,N+1):
    temp = list(input())
    for j in range(1,M+1):
        A[i][j] = int(temp[j-1])

# Problem Characteristics
dx = [1,0,-1,0] ;dy = [0,1,0,-1]

def BFS():
    x=1;y=1 # startNode
    q = deque()
    q.append([x,y])
    visited[x][y] = True

    while q:
        currentNode = q.popleft()
        # path.append(currentNode)
        for i in range(4):
            x = currentNode[0] + dx[i]
            y = currentNode[1] + dy[i]
            if 1<= x <= N and 1 <= y <= M:
                if A[x][y] != 0 and not visited[x][y]:
                    q.append([x,y])
                    visited[x][y] = True
                    A[x][y] = A[currentNode[0]][currentNode[1]] + 1
```

```
    if A[N][M] > 1:
        return A[N][M]
    else:
        return 'Fail'

print(BFS())
```

- 다음주 문제 추천
  - Programmers
    - 게임 맵 최단거리 (BFS)
    - 네트워크 (DFS)
  - https://www.acmicpc.net/problem/1874