인공지능 실습 Chapter 1

AI 프로그래밍

포항공과대학교 컴퓨터공학과

TA 소개

류성한

- CH 1, 3, 5, 7 담당
- ryush@postech.ac.kr

이동하

- CH 2, 4, 6 담당
- dongha0914@postech.ac.kr

강의 개요

- 수업 내용
  - 이론 시간에 배운 지식을 활용해 각 단원 별 AI 문제 해결

- 수업 목표
  - AI 기술을 바탕으로 주어진 문제를 해결 할 수 있다
  - 실습을 통해 AI 이론을 이해한다

- 수업 방법
  - 각 단원 별 이론 리뷰
  - 실습 문제 소개
  - 단계적으로 Python 코드 구현

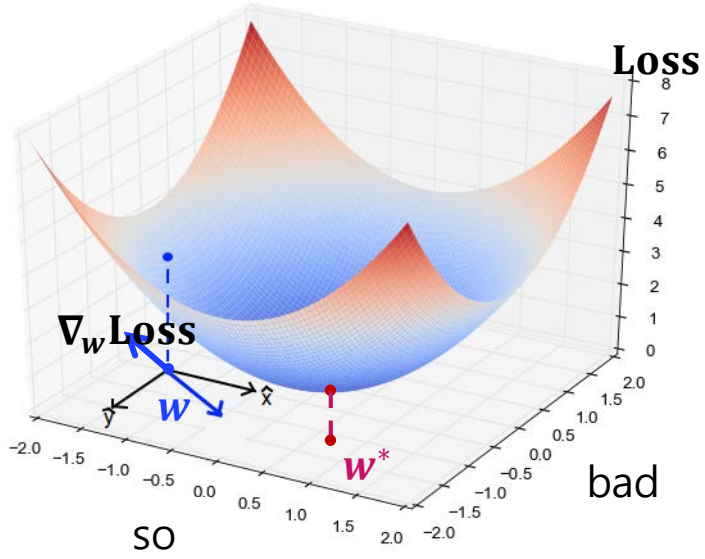
목차

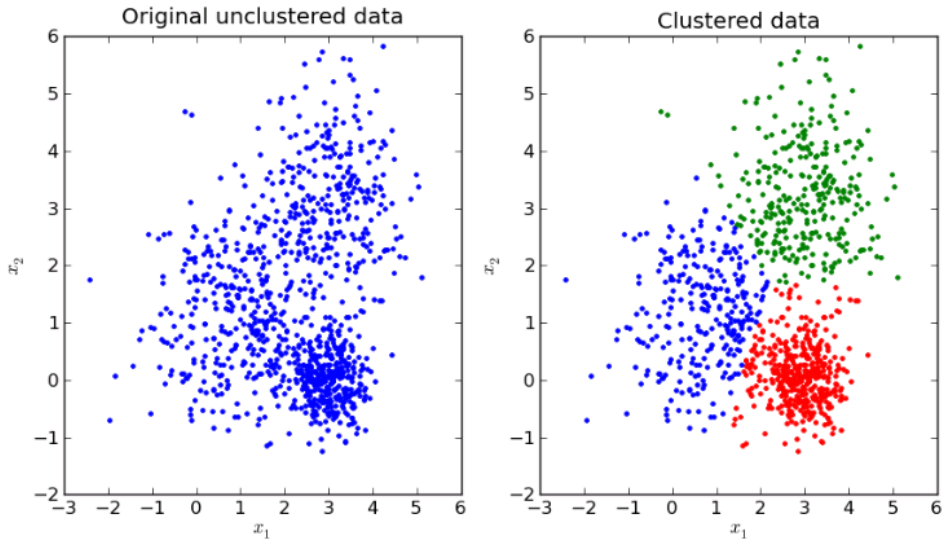
CH1. AI 프로그래밍 연습

- 최장 단어 검색 구현
- 최빈도 단어 검색 구현
- 최장 Palindrome 계산 구현
- 문장 변이 (Mutation) 구현
- 맨해튼 거리 계산 구현
- Vector의 Dot Product 구현
- Vector의 합연산 구현

CH2. Machine Learning

- Supervised Learning (지도 학습)
  - Sentiment Analysis
- Unsupervised Learning (비지도 학습)
  - k-means Clustering



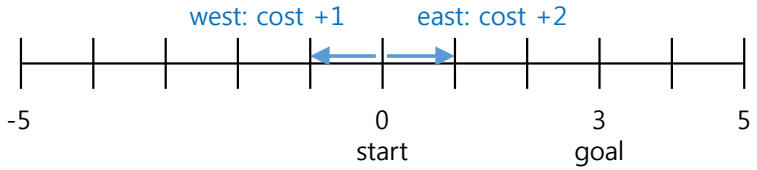


CH3. Search

- Back-tracking search
- Dynamic programming
- Uniform cost search
- A* Search

Transportation

Number Line



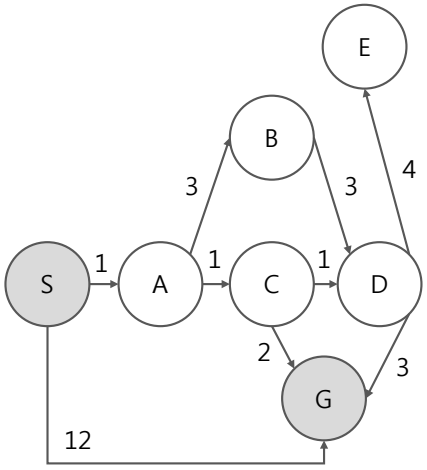
Word Segmentation

"thisisnotmybeautifulhouse"
=> "this is not my beautiful house"

Vowel Insertion

"thts m n th crnr"
=> "thats me in the corner"

Graph Search



CH4. Markov Decision Process

- Markov decision process
  - Value iteration
- Reinforcement learning
  - Q-learning

Blackjack

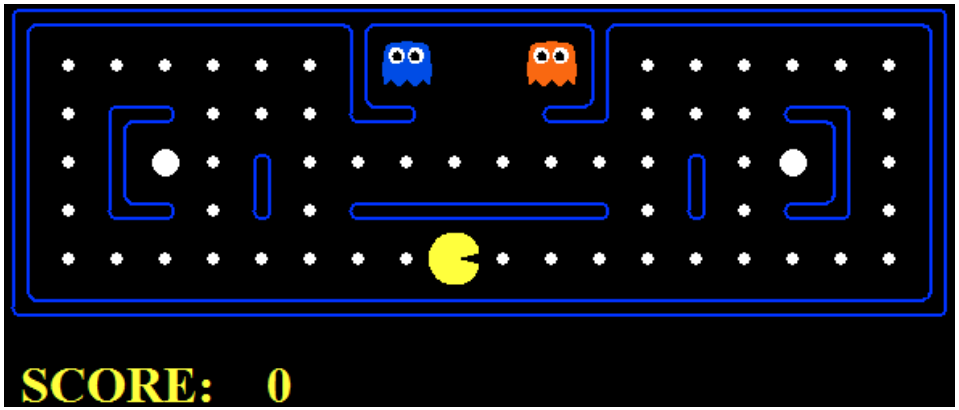
CH5. Adversarial Search

- Minimax
- Expectimax
- Alpha-beta pruning
- Evaluation function

Tic-tac-toe

X		
	O	
		X

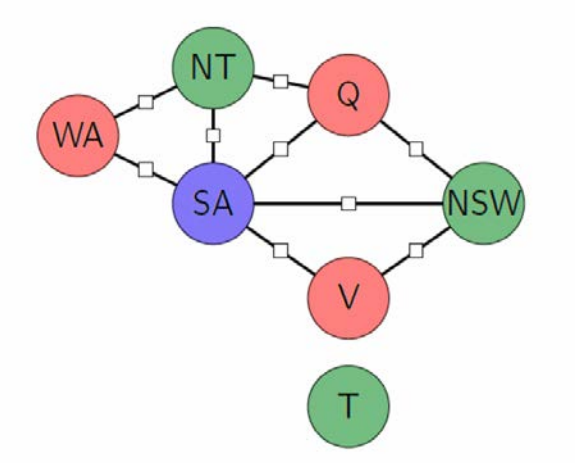
Pacman



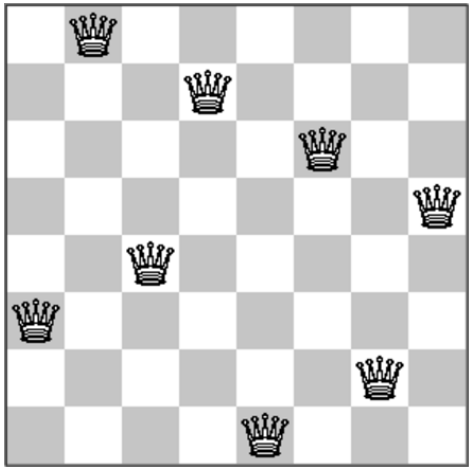
CH6. Constraint Satisfaction Problem

- Factor graphs
- Dynamic ordering
- Arc consistency
- Course Scheduling

Map-coloring problem



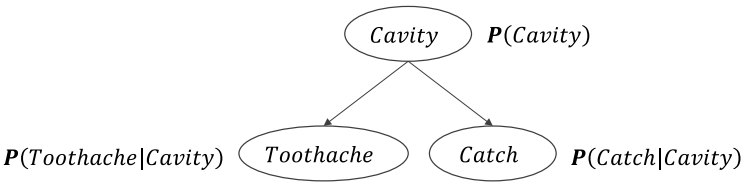
N-queens problem



CH7. Bayesian Network

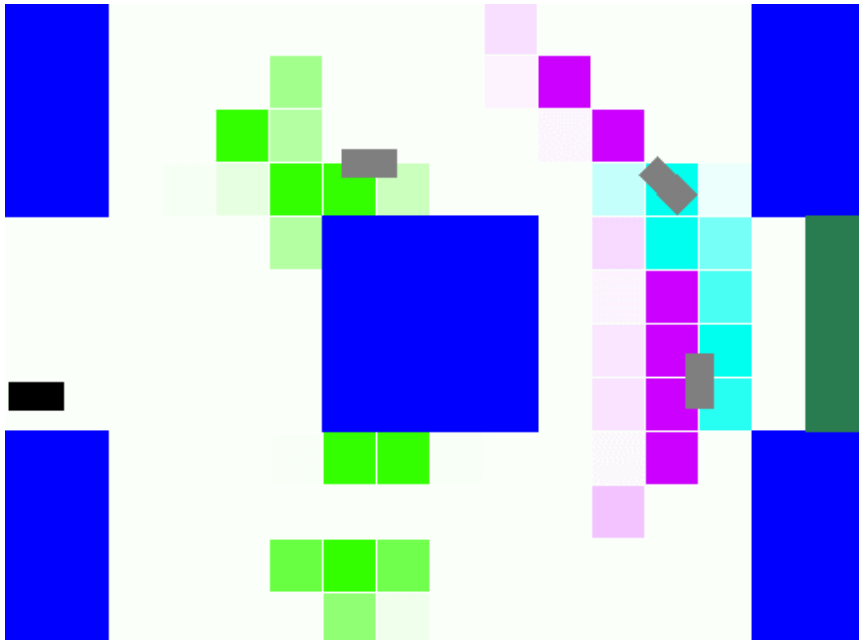
- 기본 확률 이론
- Naive Bayes model
- Bayesian network

충치 진단 시스템



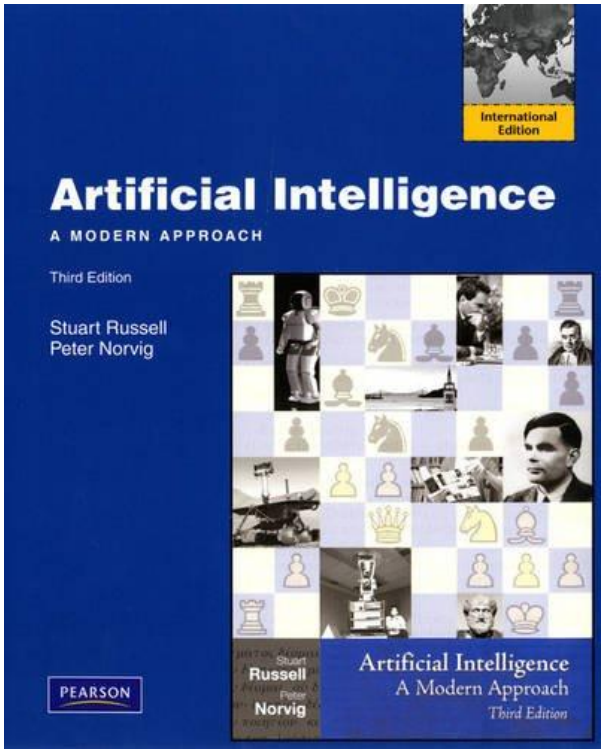
Q. 치통이 있을 때 충치가 있을 확률은?

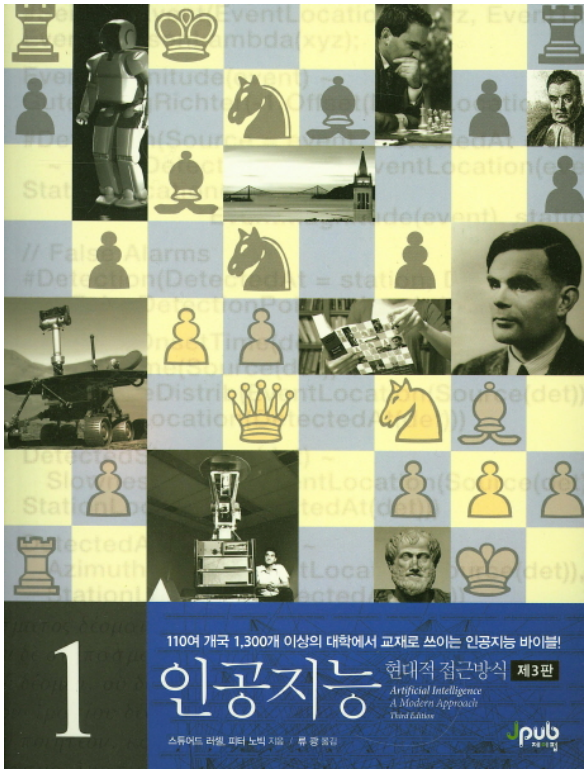
Car Tracking



AI 관련 교재

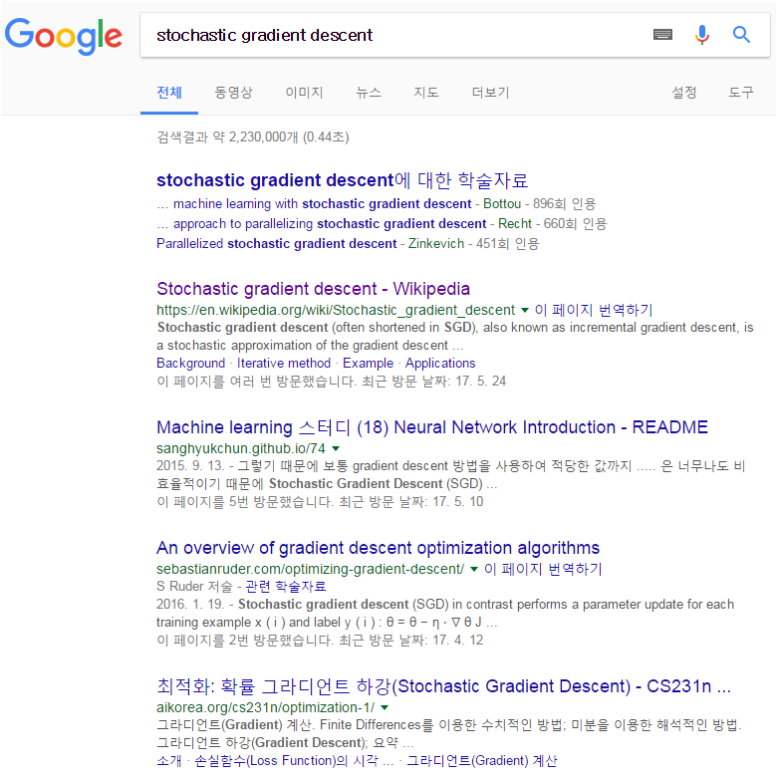
- [AIMA] Artificial Intelligence: A Modern Approach (3rd Edition)
  인공지능: 현대적 접근방식
  - by Stuart Russell, Peter Norvig

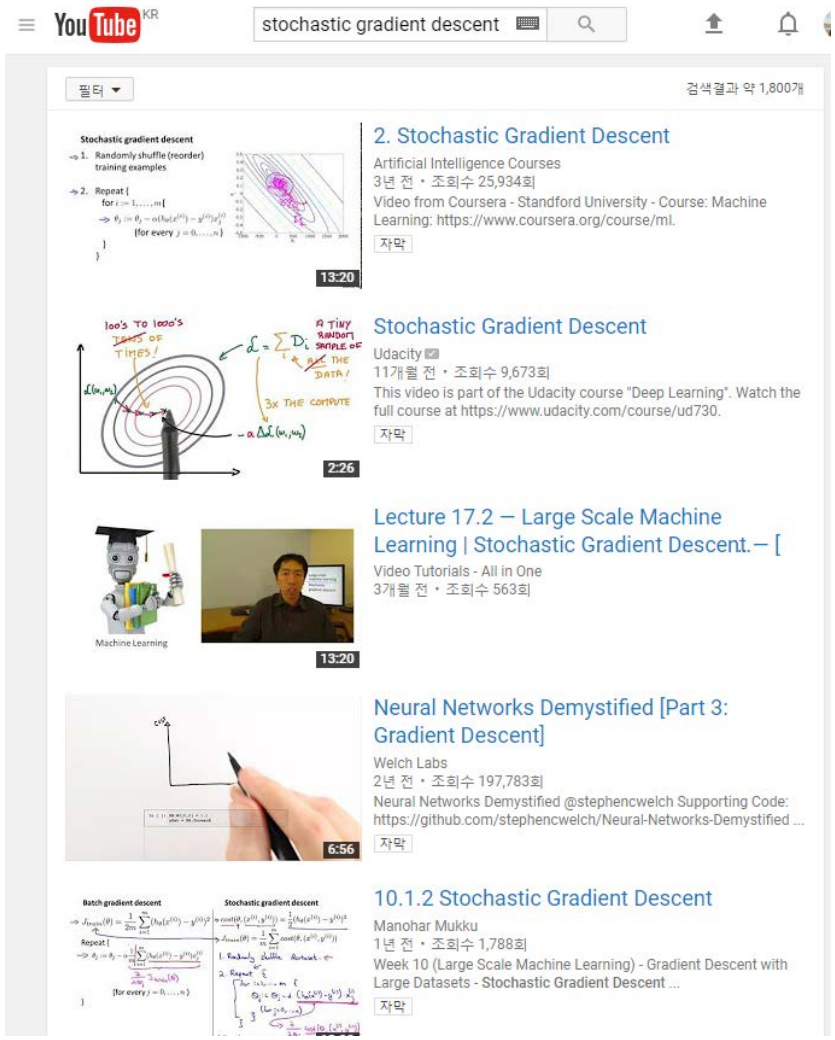




AI 관련 교재

- 구글 검색
- Youtube
- Wikipedia





A. 최장 단어 검색 구현

- 다음 조건을 만족하는 computeMaxWordLength(text) 함수를 구현
  - 입력으로 문자열 text가 주어짐 (단어는 공백으로 구분)
    - 예. 'which is the longest word'
  - 가장 긴 단어를 반환(return)
  - 최대길이 단어가 여러 개 있는 경우 알파벳 기준으로 가장 뒤에 나오는 단어를 반환

- 테스트 케이스
  - 1: 'which is the **longest** word'
  - 2: 'cat **sun** dog'
  - 3: '0 1 2 3 ... 99998 **99999**'

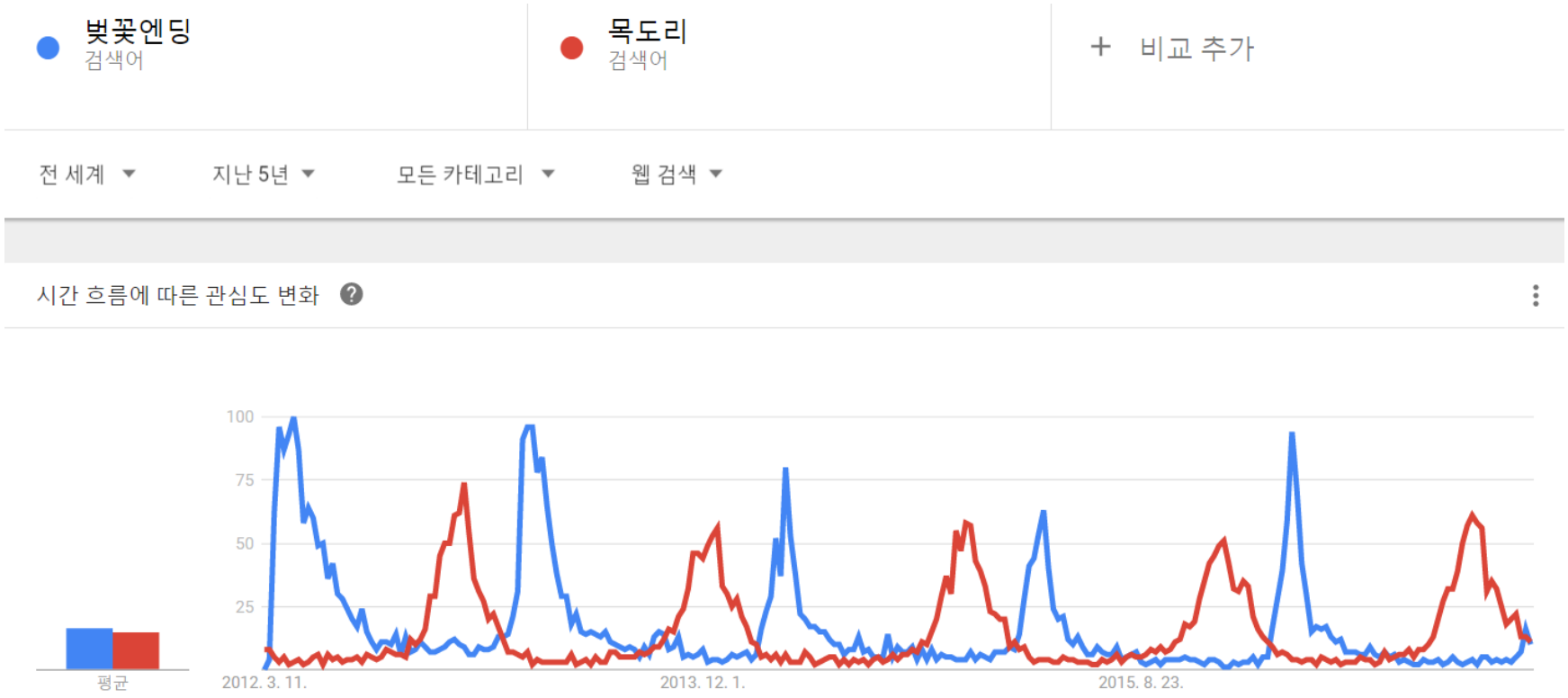
A. 최장 단어 검색 구현

- computeMaxWordLength(text):
  1. 문장(text)을 여러 단어로 분리 => words
  2. 단어(words)의 최대 길이 계산 => max_len
  3. 최대 길이 단어 찾기 => longest_words
  4. 최대 길이 단어 정렬 => longest_words
  5. 가장 마지막 단어 반환 => longest_words[-1]

1a.py 구현

B. 최빈도 단어 검색 구현

- Google Trends - https://trends.google.com/trends/
  - 웹문서에서의 단어 빈도수 측정



B. 최빈도 단어 검색 구현

- 최빈도 단어의 set 및 해당 빈도수 반환하는 computeMostFrequentWord(text) 구현 (tuple 형태)

- 테스트 케이스
  - f-0: 'the quick brown fox jumps over the lazy fox'
    =>(set(['the', 'fox']), 2)
  - f-1: 'the quick brown fox jumps over the lazy dog'
    =>(set(['the']), 2)

B. 최빈도 단어 검색 구현

- computeMostFrequentWord(text)
  1. 문장(text)을 여러 단어로 분리 => words
  2. 단어 빈도수 저장 => word_freqs
  3. 최대 빈도수 저장 => max_cnt
  4. 단어 빈도수(word_freqs)가 max_cnt인 단어 찾기 => frequent_words
  5. 결과 반환 => (frequent_words, max_cnt)

1b.py 구현

C. 최장 Palindrome 계산 구현

- Palindrome (회문): “거꾸로 읽어도 제대로 읽는 것과 같은 문장이나 낱말” by 위키
  - 예) 소주 좀 주소

- 입력 텍스트에서 임의의 문자를 지워 가장 긴 회문의 길이를 찾는 computeLongestPalindrome(text) 구현

- 테스트 케이스
  - 1: ‘’ => 0
  - 2: ‘ab’ => 1
  - 3: ‘aa’ => 2
  - 4: ‘animal’ => 3 (‘ana’, ‘aia’, ‘ama’)
  - 5: something long

C. 최장 Palindrome 계산 구현

- 알고리즘을 recursion 형태로 정의

$$f(text) = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ 2 + f(text[2, \ldots, n-1]) & \text{if } n > 1 \text{ and } text[1] = text[n] \\ \max(f(text[1, \ldots, n-1]), f(text[2, \ldots, n])) & \text{if } n > 1 \text{ and } text[1] \neq text[n] \end{cases}$$

- 첫번째 문자와 마지막 문자가 동일한 경우

f(| 소 | 주 | 좀 | 주 | 소 |)

= 2 + f(| 주 | 좀 | 주 |)

= 2 + 3 = 5

C. 최장 Palindrome 계산 구현

$$f(text) = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ 2 + f(text[2, \ldots, n-1]) & \text{if } n > 1 \text{ and } text[1] = text[n] \\ \max(f(text[1, \ldots, n-1]), f(text[2, \ldots, n])) & \text{if } n > 1 \text{ and } text[1] \neq text[n] \end{cases}$$

- 첫번째 문자와 마지막 문자가 다른 경우

f(신 도 제 제 도)

= max(f(신 도 제 제), f(도 제 제 도))

= max(2, 4) = 4

C. 최장 Palindrome 계산 구현

$$f(text) = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ 2 + f(text[2, \ldots, n-1]) & \text{if } n > 1 \text{ and } text[1] = text[n] \\ \max(f(text[1, \ldots, n-1]), f(text[2, \ldots, n])) & \text{if } n > 1 \text{ and } text[1] \neq text[n] \end{cases}$$

- 위와 같은 recursion 형태의 알고리즘을 구현

1c.py 구현

- 5번째 테스트 케이스는 긴 실행시간이 필요
  - 동일한 연산을 반복하기 때문
  - Dynamic 프로그래밍으로 해결

C. 최장 Palindrome 계산 구현

- Recursion을 이용한 피보나치 수 (Fibonacci Numbers)
  - 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...

점화식:

$$F_n := \begin{cases} 0 & \text{if } n = 0; \\ 1 & \text{if } n = 1; \\ F_{n-1} + F_{n-2} & \text{if } n > 1. \end{cases}$$

Python 코드:

```
In [1]: def fib(n):
            if n == 0:
                return 0
            elif n == 1:
                return 1
            else:
                return fib(n-1) + fib(n-2)

In [2]: fib(5)
Out[2]: 5
```

C. 최장 Palindrome 계산 구현

- Recursion을 이용한 피보나치 수 (Fibonacci Numbers)
  - fib(5)
    = fib(4) + fib(3)
    = (fib(3) + fib(2)) + (fib(2) + fib(1))
    = ((fib(2) + fib(1)) + (fib(1) + fib(0))) + ((fib(1) + fib(0)) + fib(1))
    = (((fib(1) + fib(0)) + fib(1)) + (fib(1) + fib(0))) + ((fib(1) + fib(0)) + fib(1))
  - Dynamic programming
    - 복잡한 문제를 간단한 여러 개의 문제로 나누어 푸는 방법

C. 최장 Palindrome 계산 구현

- Dynamic 프로그래밍을 이용한 피보나치 수 계산

In [1]:
```
from collections import defaultdict
```

In [2]:
```
memory = defaultdict(int, {0:0, 1:1})
```

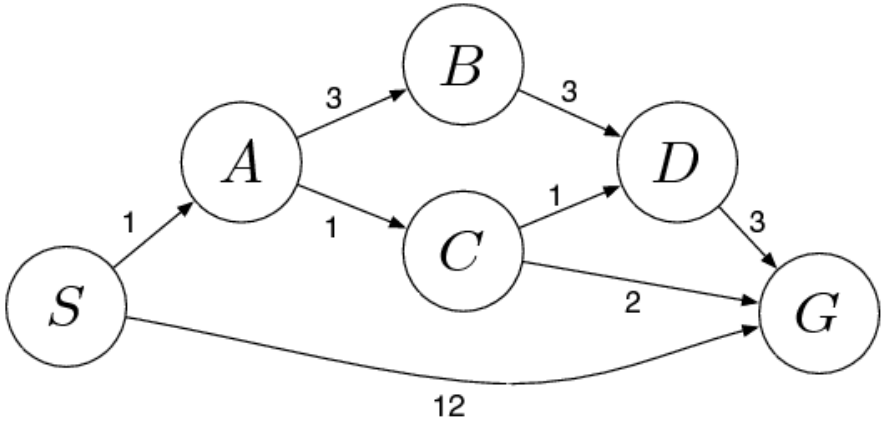
In [3]:
```
def fib(n):
    if n not in memory:
        memory[n] = fib(n-1) + fib(n-2)
    return memory[n]
```

In [4]:
```
fib(5)
```

Out[4]: 5

C. 최장 Palindrome 계산 구현

- Dynamic programming 사용 예
  - 최단 경로 탐색 (CH3)



- 앞서 구현한 recursion 형태의 palindrome 알고리즘을 dynamic 프로그래밍으로 수정

1c.py 수정

D. 문장 변이 (Mutation) 구현

- 주어진 입력 문장을 바탕으로 동일한 길이의 문장을 생성하는 mutateSentences(sentence) 구현
  - 조건 : 생성된 문장에서 인접한 단어들은 입력 문장에서도 인접해야 함
- 예) 입력 문장 'the cat and the mouse‘
  - 인접 단어 목록
    - the cat
    - cat and
    - and the
    - the mouse
  - 문장 생성
    - 'and the cat and the‘
    - 'the cat and the mouse‘
    - 'the cat and the cat‘
    - 'cat and the cat and'

D. 문장 변이 (Mutation) 구현

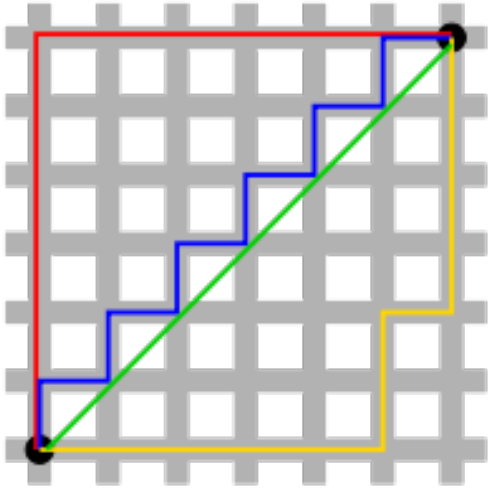
- 테스트 케이스
  - 1: 'a a a a a' -> ['a a a a a']
  - 2: 'the cat' -> ['the cat']
  - 3: 'the cat and the mouse' -> ['and the cat and the', ...]

- mutateSentences(sentence):
  1. 문장(text)을 여러 단어로 분리 => words
  2. 모든 가능한 단어 조합(word pairs) 찾기 => word_pairs
  3. 길이가 1인 문장 리스트 생성 => results
  4. 반복문을 통해 길이를 확장 => results

1d.py 구현

E. 맨해튼 거리 계산 구현

- Manhattan distance (or L_1 distance, taxicab metric)를 계산하는 manhattanDistance(loc1, loc2) 구현

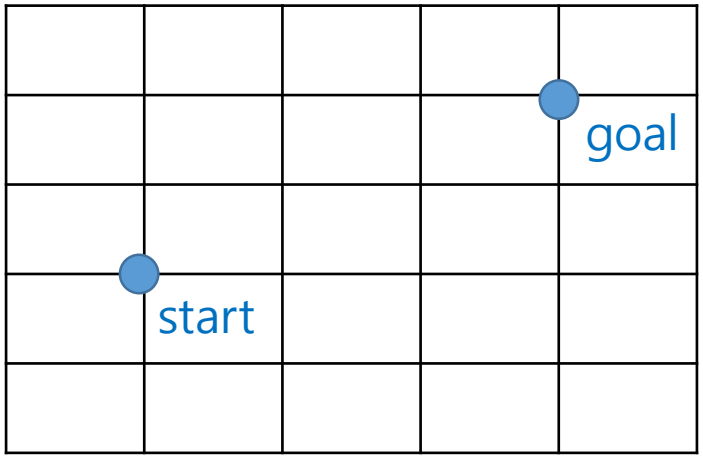


Green line:
Euclidean distance

- L1 distance between (x_1, y_1) and (x_2, y_2) = $|x_1-x_2| + |y_1-y_2|$

E. 맨해튼 거리 계산 구현

- Manhattan distance 사용 예
  - 최단 경로 탐색 (CH3)



E. 맨해튼 거리 계산 구현

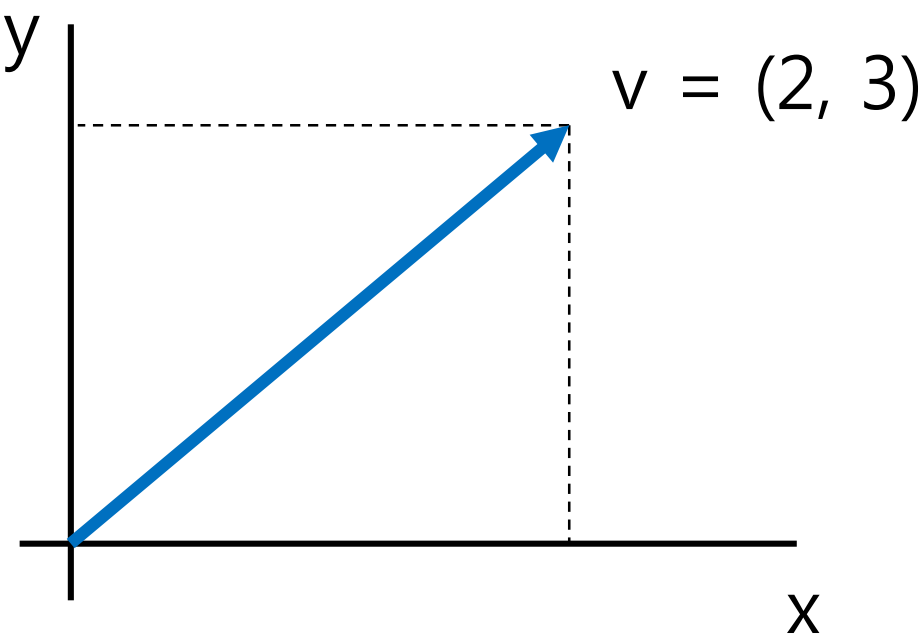
- 테스트 케이스
  - 1: loc1=(0, 0), loc2=(5, 5), answer=10
  - 2: loc1=(2, 3), loc2=(2, 3), answer=0
  - 3: loc1=(3, 5), loc2=(1, 9), answer=6

1e.py 구현

Vectors in AI

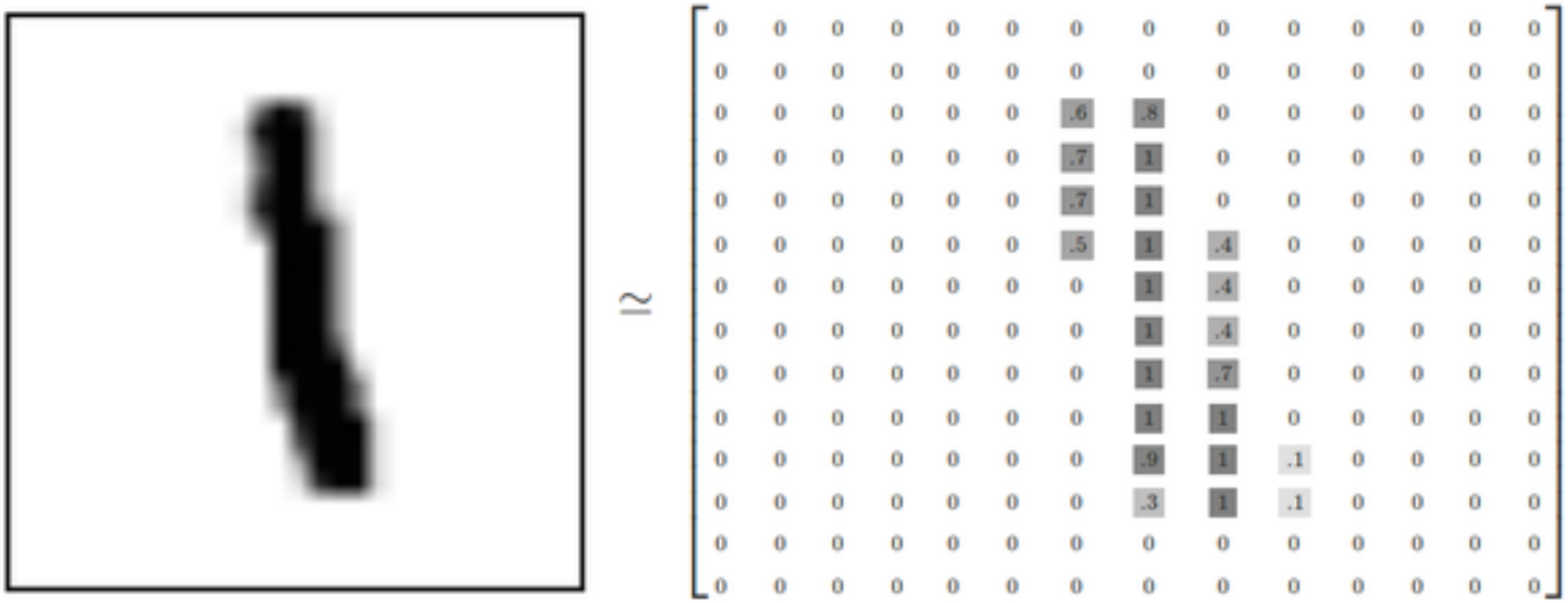
- 스칼라 (scalar): 어떤 공간에서의 값 하나
  - 예. x = 13.93

- 벡터 (vector): 여러 스칼라로 표현되는 것
  - 방향과 크기를 가짐
  - 예.



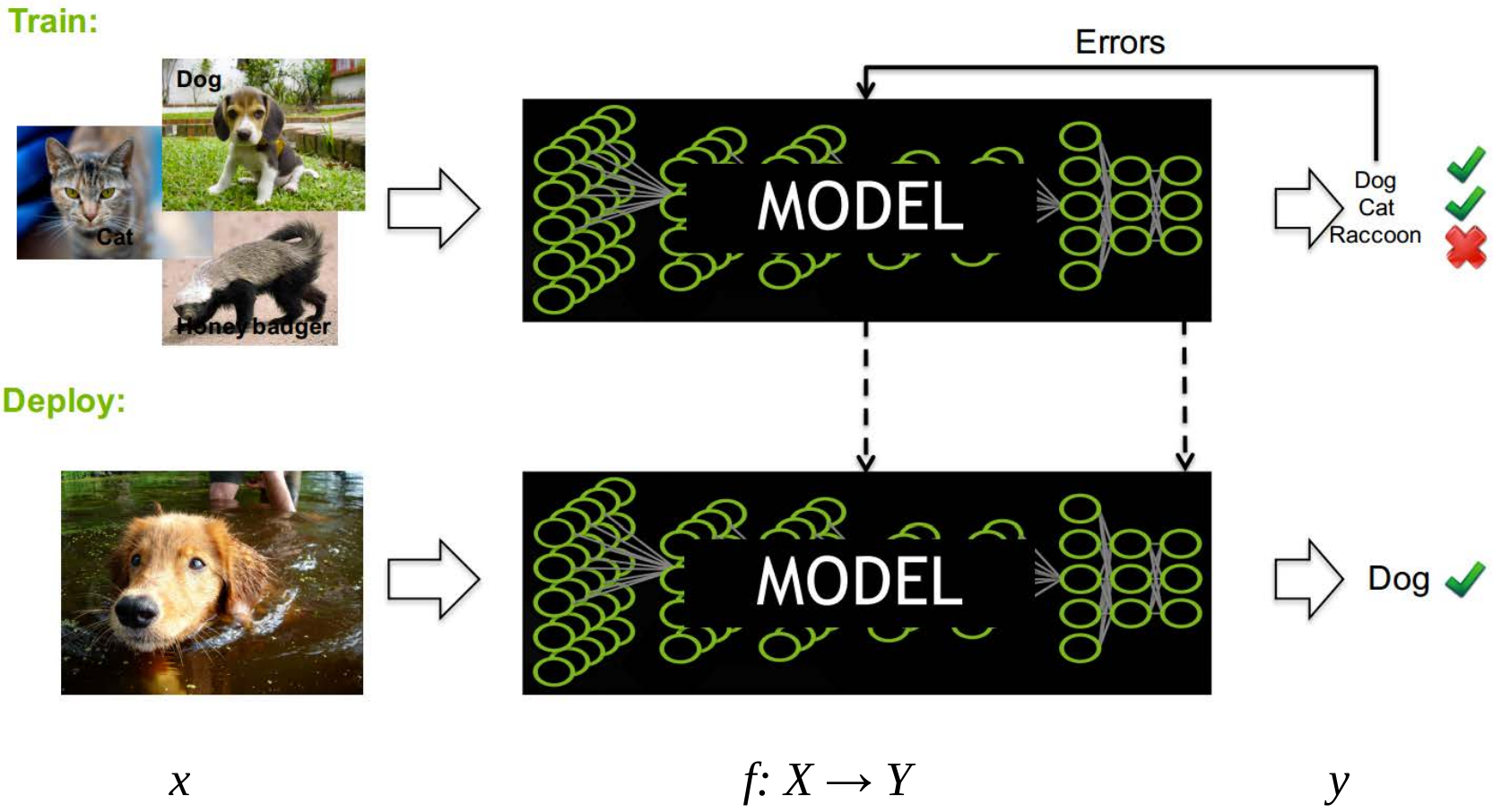
Vectors in AI

- 이미지
  - MNIST: 필기체 숫자 데이터



Vectors in AI

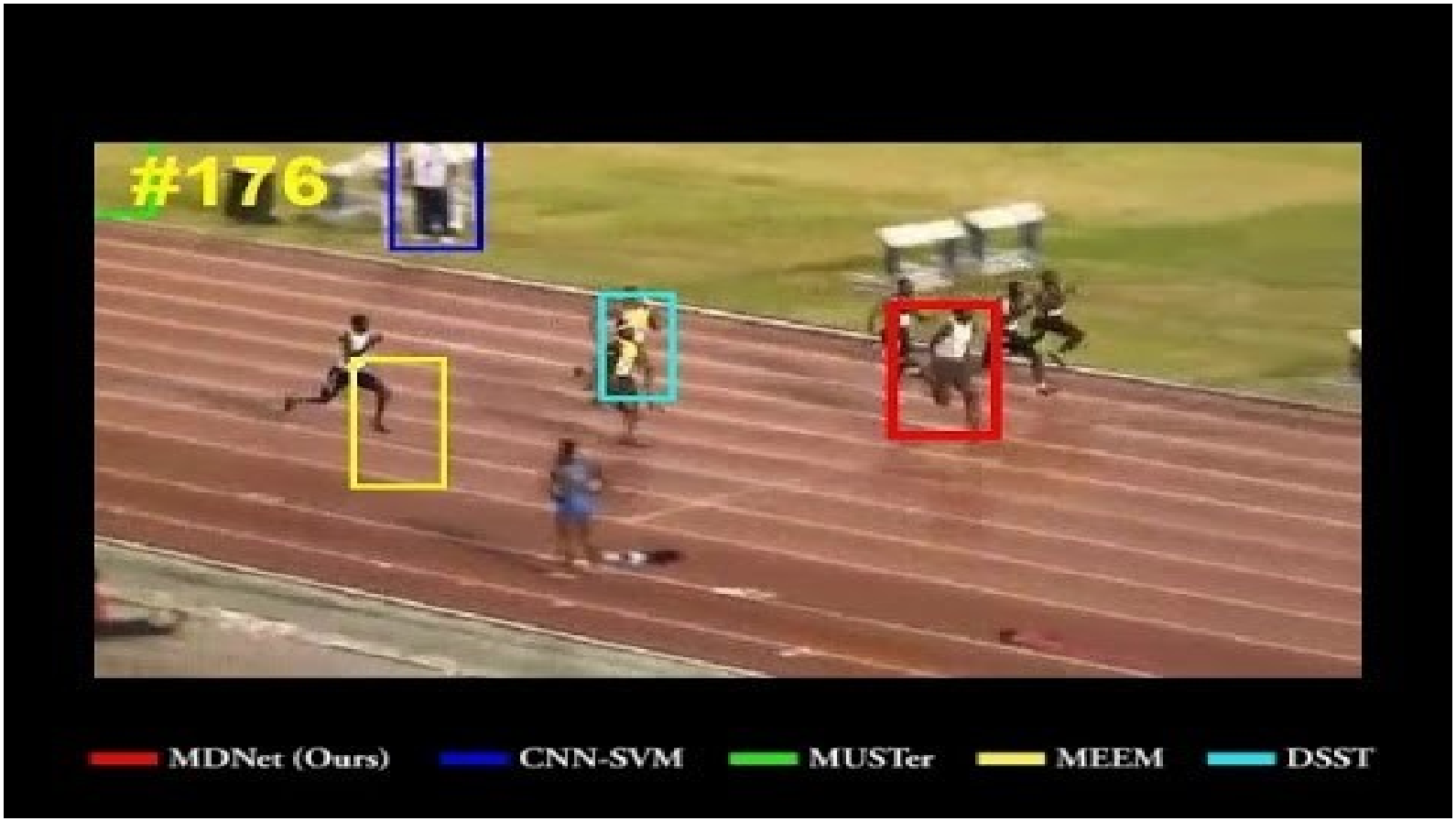
- 이미지



https://developer.nvidia.com/deep-learning-courses

Vectors in AI

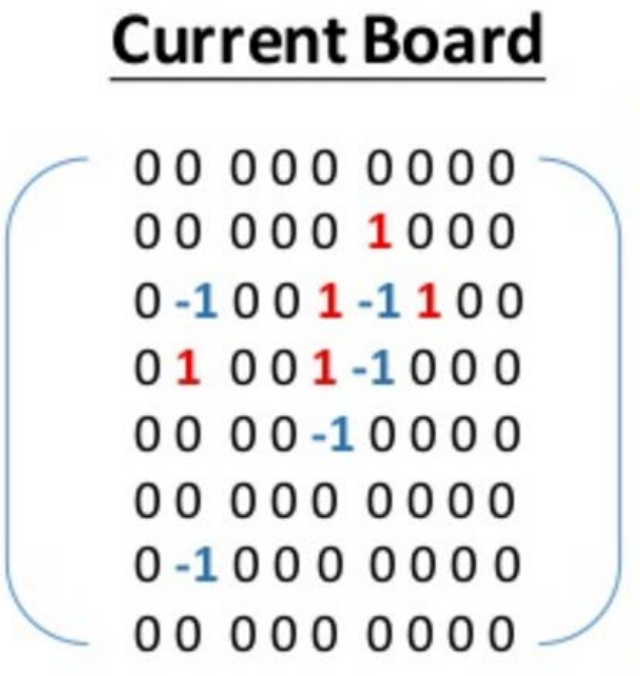
- 동영상



Hyeonseob Nam and Bohyung Han, Learning Multi-Domain Convolutional Neural Networks for Visual Tracking, CoRR

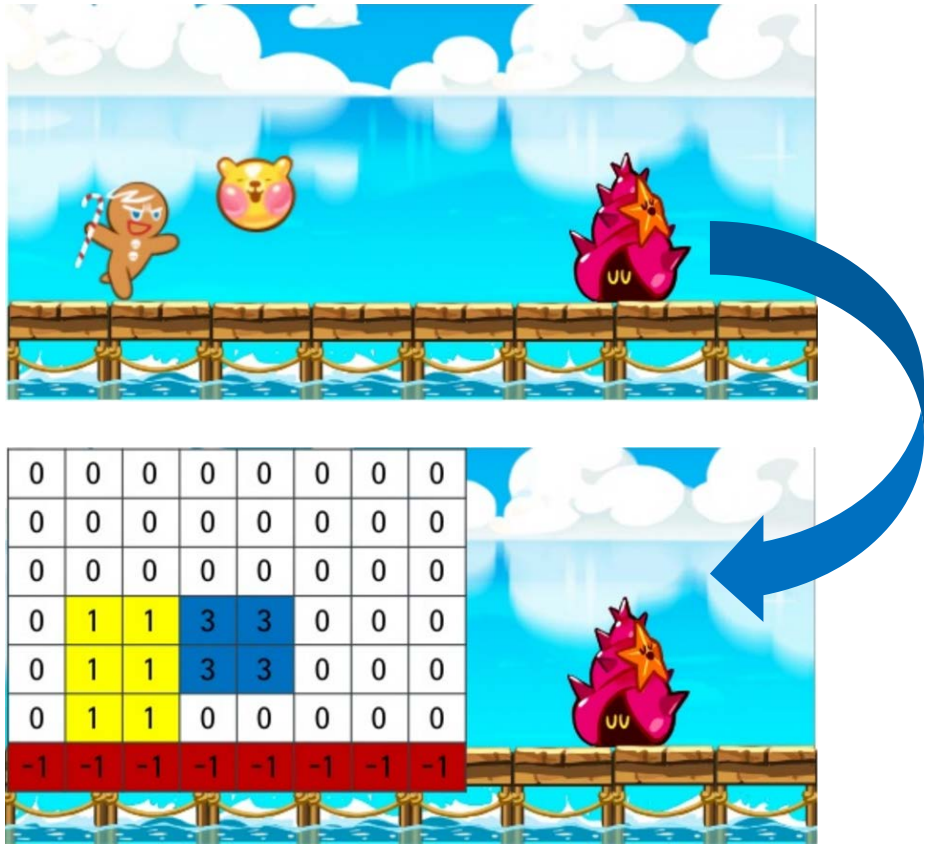
Vectors in AI

- 게임



Action: 바둑알의 위치 선택

Shane Moon, How AlphaGo Works

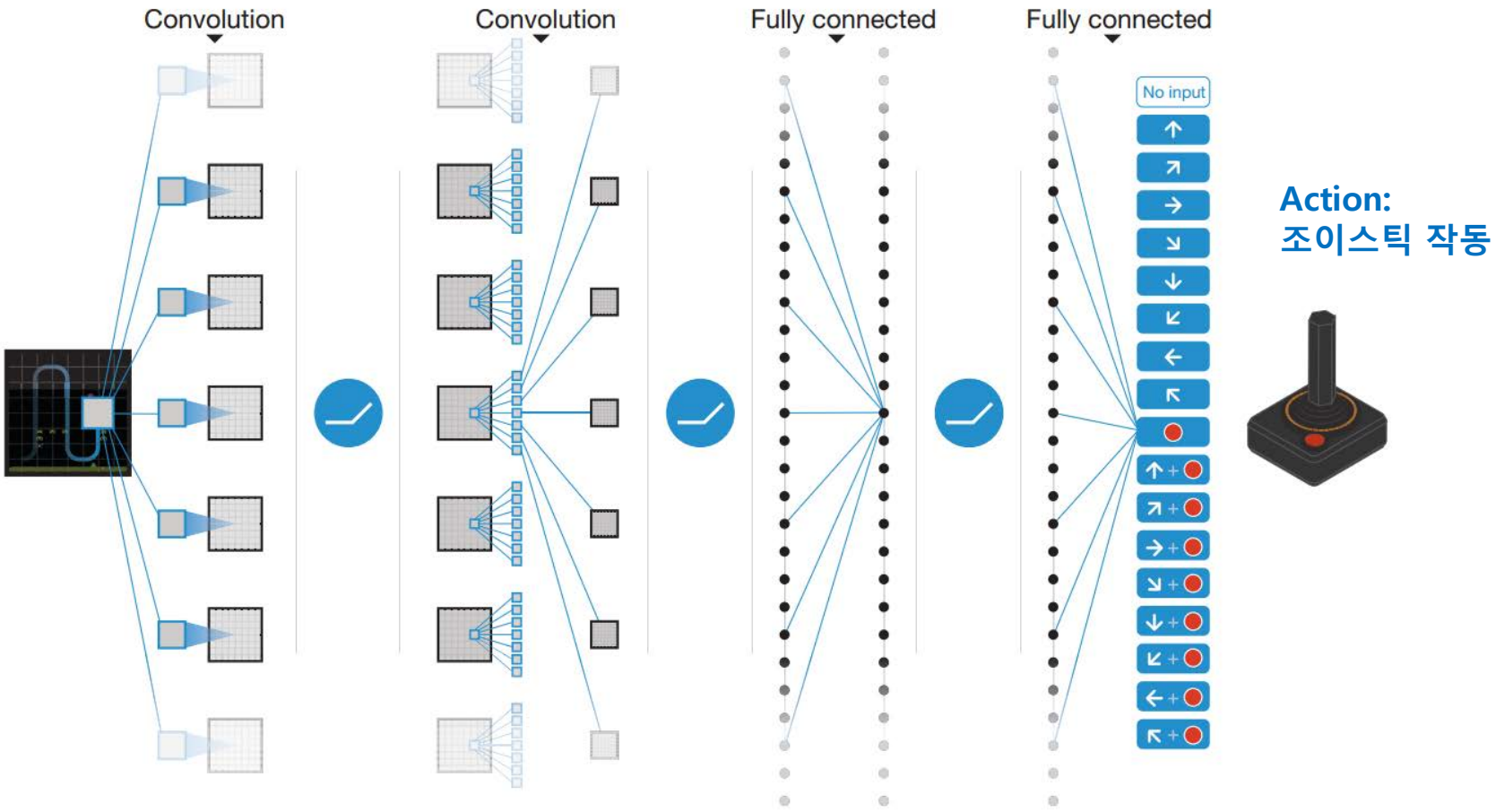


Action: 점프, 슬라이드, 가만히

김태훈, 2016, 딥러닝과 강화 학습으로 나보다 잘하는 쿠키런 AI 구현하기, Naver DEVIEW.

Vectors in AI

- 게임



Mnih et al., Deep Q-Network, 2015

Vectors in AI

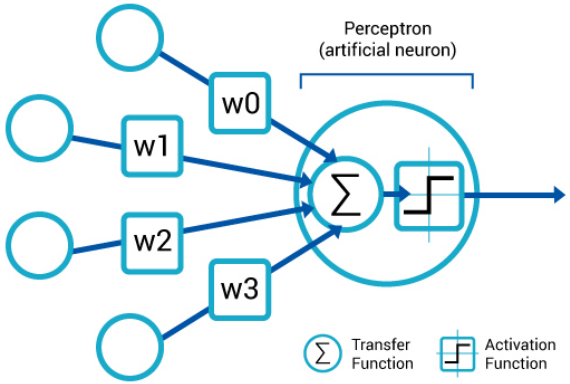
- 게임

Google DeepMind's Deep Q-learning playing Atari Breakout

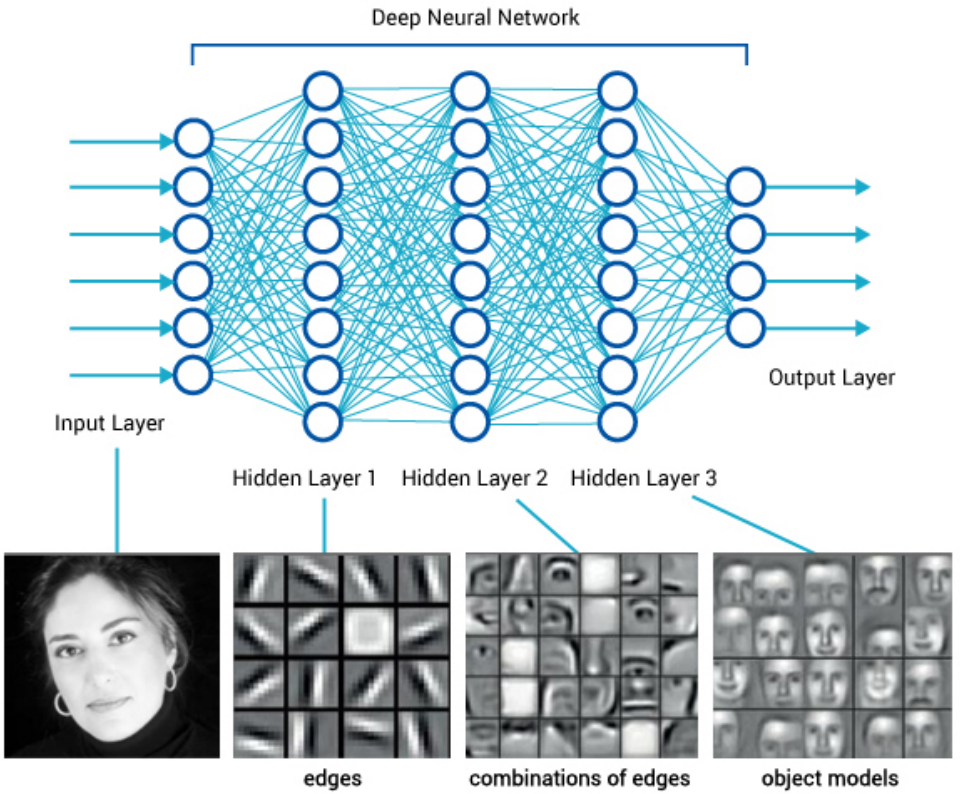
Vectors in AI

- Many items in AI are vectors (or matrices)

Single neuron



Deep neural network



https://www.amax.com/blog/?p=804

Vectors in AI

- 텍스트의 vectorization

텍스트	차원 (dimension) = 특징 (feature)				
	i	me	you	hate	love
"I love you"	1	0	1	0	1
"you hate me"	0	1	1	1	0

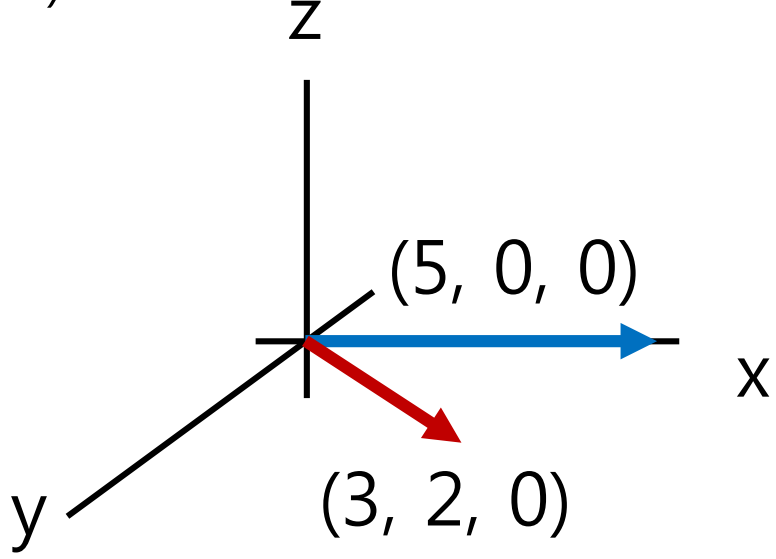
- Python에서의 표현 방법
  - Vector as a list: [1 0 1 0 1]
  - Vector as a dictionary: {'i':1, 'love':1, 'you':1}

F. Vector의 Dot Product 구현

- Dot product (스칼라곱, 내적)을 계산하는 sparseVectorDotProduct(v1, v2) 구현

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$$

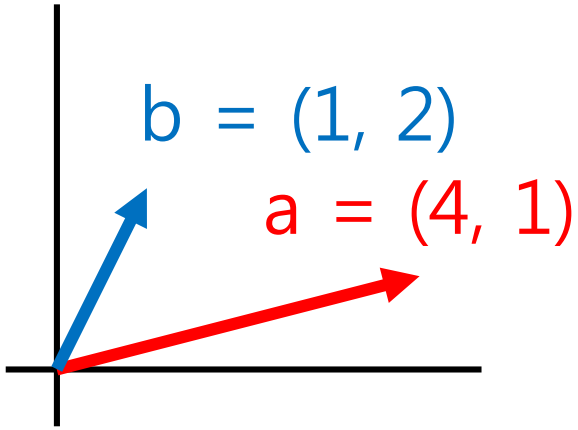
- 예. (5, 0, 0) · (3, 2, 0) = 5 X 3 + 0 X 2 + 0 X 0 = 15



F. Vector의 Dot Product 구현

- Geometric 의미
  - Dot product와 vector의 크기를 알면 사이각을 구할 수 있다

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$



$\cos(\theta) = a \cdot b \,/\, (|a|\,|b|)$

$= 6 \,/\, (\sqrt{4^2 + 1^2} \times \sqrt{1^2 + 2^2}$

$= 6 \,/\, 9.21954445\ldots$

$= 0.650791373\ldots$

$\therefore \theta = 49.398705354\ldots'$

F. Vector의 Dot Product 구현

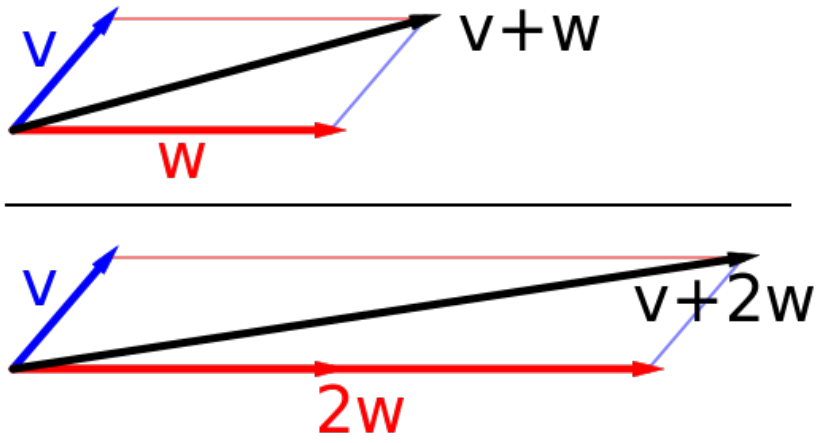
- 테스트 케이스
  - 1: v_1={a:5}, v_2={a:3, b:2}, $v_1 \cdot v_2 = 15$
  - 2: v_1={c:5}, v_2={a:2, b:1}, $v_1 \cdot v_2 = 0$
  - 3: v_1={a:5, b:4}, v_2={a:-1, b:2}, $v_1 \cdot v_2 = 3$

1f.py 구현

G. Vector의 합연산 구현

- incrementSparseVector(v1, scale, v2) 구현
  - v1 += scale * v2

- Geometric 의미



- 테스트 케이스
  - 1: v_1={a:5}, scale=2, v_2={a:3, b:2}, answer = {a:11, b:4}

1g.py 구현

Vectorization

- Vector 연산 사용 예
  - 감정 분석 (CH2)
    - $\phi(x_1) = \{`very': 1, `interesting': 1, `movie': 1\}$
    - $\phi(x_2) = \{`very': 1, `boring': 1\}$
    - Loss function

Vectorization

$$\text{Loss}_{\text{hinge}}(x, y, \mathbf{w}) = \max\{0, 1 - \mathbf{w} \cdot \phi(x) y\}$$

Dot product

    - Stochastic Gradient Descent

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} \text{Loss}_{\text{hinge}}(x, y, \mathbf{w})$$

Vector의 합연산

Vectorization

- Vector 연산 사용 예
  - K-means Clustering (CH2)
    - 두 Vector 간의 유사도(혹은 거리) 계산