ME490 자율주행시스템을 위한 프로그래밍 Programming for Autonomous System

카이스트 기계공학과

Goal of this course

- Let students learn and experience how recent AI/Autonomous vehicle techniques are implemented in the mechanical system
- 학생들이 최근 주목받는 인공지능/자율주행 등 4차산업혁명 기술이 어떻게 기계공학과에서 다루는 시스템에 적용되는지를 배우고 실습하도록 하는 것

- Ultimately, students would know that the AI TOOLS are (just) tools that you could use for making machines autonomous
- 그리하여 학생들이 시스템을 발전시키는데 AI 관련 기술을 도구로 사용할 수 있다는 것을 알게 하는 것

Feedback from Capstone 1

- Pros
  - Challenging, fruitful curriculum
  - 도전적이지만 많은 것을 배울 수 있었음
  - Excellent TA support 조교들의 지원에 감사

- Cons
  - Nonsystematic coding curriculum 소프트웨어에 대한 체계적 교육이 없어
  - Time consuming coding and debugging 프로그래밍에 시간이 너무 많이 할애 되었음
  - Rule changes 규정 등의 변경에 대한 불만

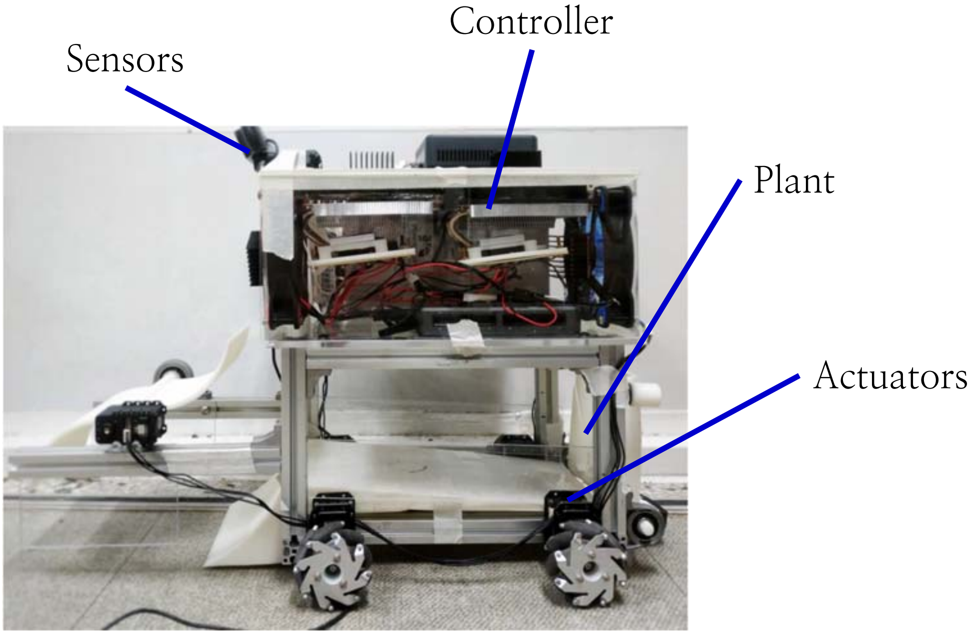
Plans for Capstone 2

- Pros
  - Challenging, fruitful curriculum
  - 도전적이지만 많은 것을 배울 수 있었음
  - Excellent TA support 조교들의 지원에 감사

- Attempt to improve the Cons
  - Programming for autonomous system ME491 창시구프로그래밍 교육 및 실습
  - Rule setting: TA pre-runs (software wise)

The changes in rule (if any) would be introduced to make your life easy !

Recall, your autonomous vehicle system



Autonomous vehicle system

Controller

- Go forward
- With speed v
- With direction θ, combinations of n-wheels

– direct controller by X-box

– Previous 창시구 (시모제, 자동제어)

– New 창시구 (자율주행, 딥러닝)

– autonomous control: find out where & how to go and come back

- TOOLs for making system autonomous

"SLAM" "Machine learning"
"Neural net CNN"
"Reinforcement learning"
"Deep learning"

SLAM: Simultaneous Localization And Mapping

Programming for autonomous system

(Thanks to) Open source software

- Open-source software (OSS) is a type of computer software whose source code is released under a license in which the copyright holder grants users the rights to study, change, and distribute the software to anyone and for any purpose.
- Open-source software may be developed in a collaborative public manner.
- According to scientists who studied it, open-source software is a prominent example of open collaboration. [from wiki]

– ME490 Programming for autonomous system

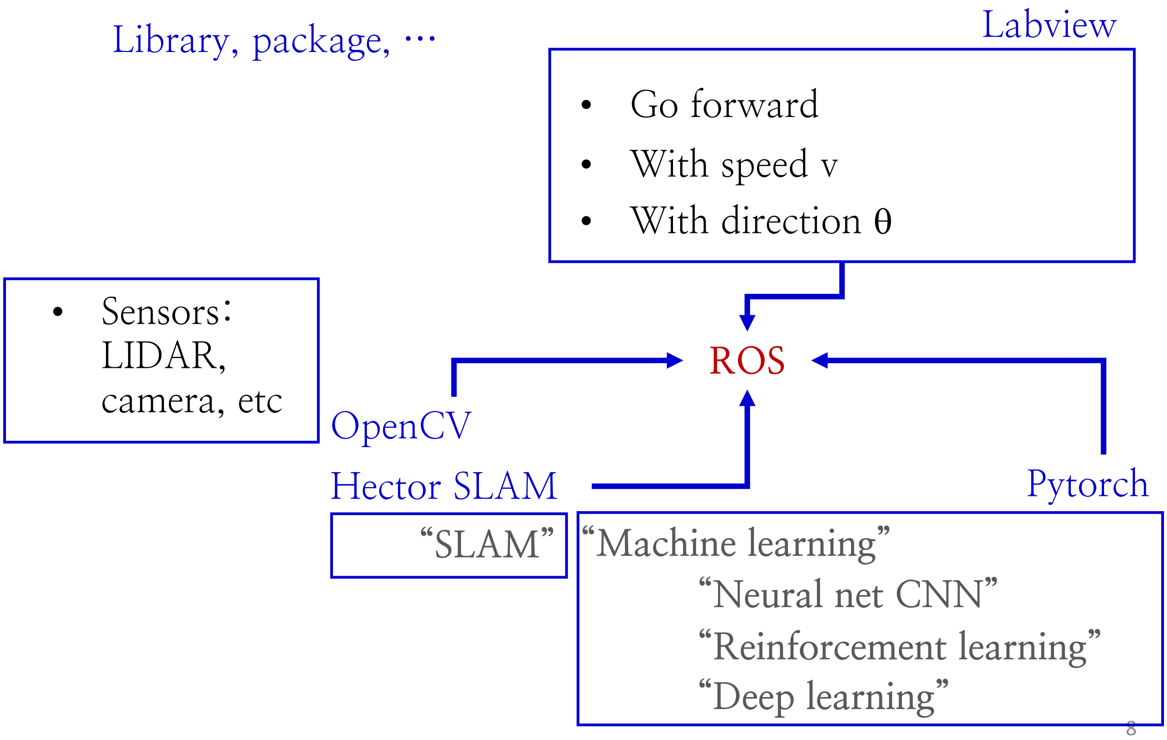
– ME401 Capstone design 2

"SLAM" CE481
"Machine learning" CS376
"Neural net CNN" CS470
CS672 "Reinforcement learning"
CS774 "Deep learning"

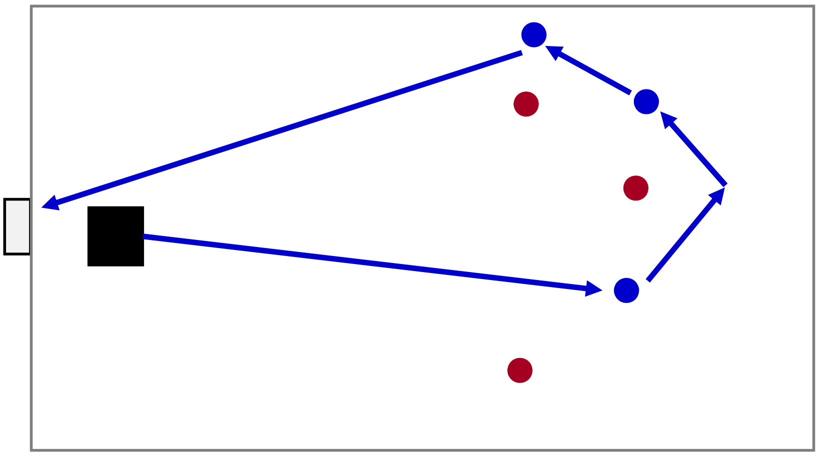
Programming for autonomous system

Open source software

Library, package, …



Recall, final mission of Capstone 1



ROS

Hector SLAM

Pytorch

"SLAM"

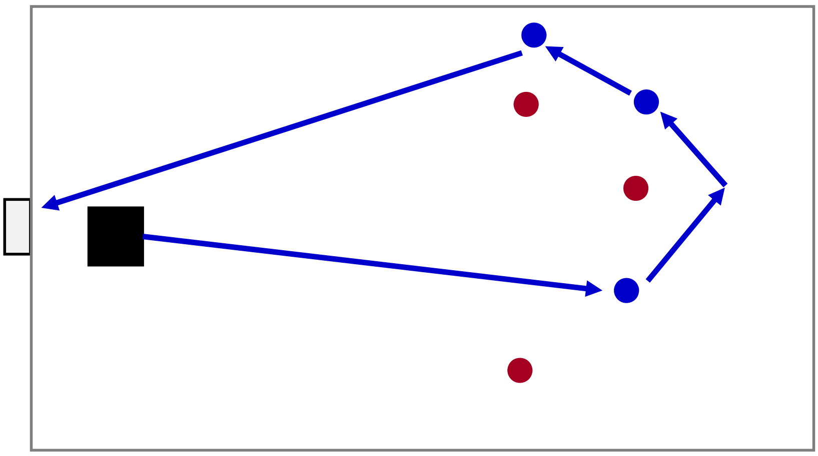
"Machine learning"

"Neural net CNN"

"Reinforcement learning"

"Deep learning"

Recall, final mission of Capstone 1



ROS

Hector SLAM

Pytorch

"SLAM"

"Machine learning"

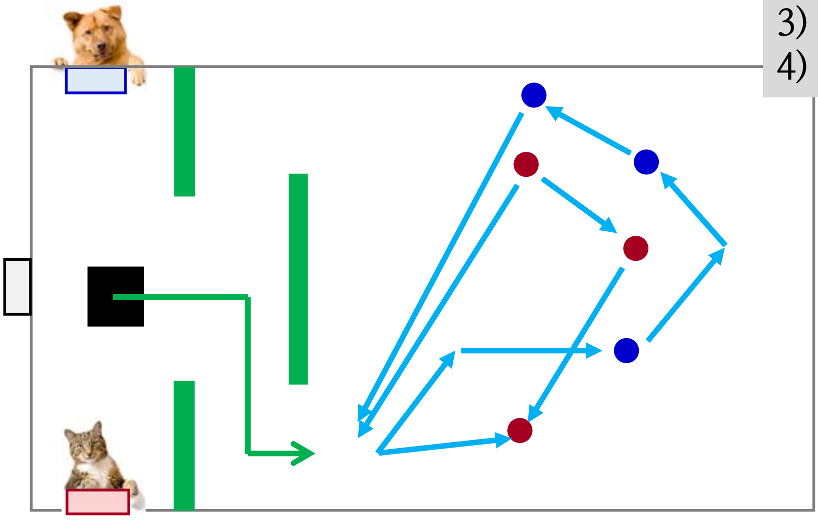
"Neural net CNN"

"Reinforcement learning"

"Deep learning"

Final mission (draft)

Goals
1) Quickly (Design)
2) Escape the maze (SLAM)
3) Pick all balls (DQN)
4) Drop to each basket (CNN)



ROS

Hector SLAM

Pytorch

"SLAM"

"Machine learning"

"Neural net CNN"

"Reinforcement learning"

"Deep learning"

Active learning (Education 4.0)

- 트랙 1: 플립드 러닝 방식 (교수님께서 사전 제작하신 동영상강의(필수)를 KLMS에서 학습 + 오프라인 상호협력학습)
- 트랙 2: 액티브 러닝 50% 이상 (수업시간 중 일방향 강의비중을 50%미만으로 줄이고, 토론/ 문제풀이/ 프로젝트 등의 다양한 학습활동(액티브 러닝) 으로 수업설계)
- Track 2: One-directional lecture: Active learning = 50:50
- 트랙 3: 액티브 러닝 100% (수업시간 중 일방향 강의 없이, 액티브러닝으로 수업설계)

	Mon	Tue	Wed	Thu
4:00-5:00	**Class**			
5:00-6:00	Practice			
6:00-7:00				
7:00-8:00				
8:00-9:00				

Course schedule

W	Date	Class	Active learning
1	8/27	0. Course introduction	Programing structure: Python, PyTorch
2	9/3	1.0 Machine learning basic	Programing of MSE
3	9/10	1.1 Neural Network	Programing of MNIST or Dog/cat
4	9/17	1.2 Convolutional Neural Network	Quiz #1 Python & PyTorch Programing of CNN *9/21 Capstone 1st design review*
5	9/24-26	Choosuk	No Class
6	10/1	2.0 SLAM basic	Code review #2 due: Customized CNN Programing structure: ROS
7	10/8	2.1 SLAM package in ROS	Programing of Hector SLAM Code Assignment #3 due: Basic ROS
8	10/15-17	Midterm	No Class, No exam

W	Date	Class	Active learning
9	10/22	2.2 SLAM for Capstone project	Programing of SLAM & path planning *10/26 Capstone 2nd design review*
10	10/29	3.0 Reinforcement learning	Code review #4 due: Customized SLAM Programing of DQN
11	11/5	3.1 Deep Q-learning: environment	Programing of simulator / emulator
12	11/12	3.2 DQN: learning	Programing of Capstone-vehicle learning network
13	11/19	3.3. DQN: testing	Code review #5 due: DQN code Programing of Capstone-vehicle testing
14	11/26	Real system implementation	Capstone system testing *11/30 Capstone Final demo*
15	12/3		Code review #6 due: Total system code

Course Syllabus

- Lecture
  - Class lecture & active learning: Mon 4:00~5:00 pm, ME building 2OOO
- Evaluation guideline
  1) Six code review reports : #1~#5 (10%, each), #6 (30% each)
  2) Attendance and class participation 20%
- Syllabus, weekly notices, forms, and lecture notes will be uploaded on the course web at KLMS, and github

Programming structure & code review

- Data prep programming
- Architecture programing
  - Module programing
  - Inter-module control
- Interface programing
  - Interface for software modules
  - Interface for different devices
- Procedure
- Sample code(s) will be provided and you are supposed to modify them to fit your system
  - Code review of sample code
  - Code review of your own (customized) code

Ex) Programming structure & code review

```
%matplotlib inline
import gym
import math
import random
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
from collections import namedtuple
from itertools import count
from PIL import Image
from tensorboardX import SummaryWriter

import torch
import torch.nn as nn
import torch.optim as optim
import torch.nn.functional as F
import torchvision.transforms as T

env = gym.make('CartPole-v0').unwrapped
writer = SummaryWriter()

# set up matplotlib
is_ipython = 'inline' in matplotlib.get_backend()
if is_ipython:
    from IPython import display
```

파이썬의 필요한 모듈을 임폴트
import necessary module

*모듈(Module)은 파이썬 코드를 논리적으로 묶어서 관리하고 사용할 수 있도록 하는 것으로, 보통 하나의 파이썬 .py 파일이 하나의 모듈이 된다. 모듈 안에는 함수, 클래스, 혹은 변수들이 정의될 수 있으며, 실행 코드를 포함할 수도 있다

모듈안의 함수만 임폴트
import necessary function from module

*하나의 모듈 안에는 여러 함수들이 존재할 수 있는데, 이 중 하나의 함수만을 불러 사용

머신 러닝 라이브러리인 파이톨치 임폴트
import PyTorch, a machine learning library

시뮬레이션 라이브러리 gym에서 cartpole 시뮬레이션 환경을 불러서 env라는 클라스명으로 저장

학습 시키는 네트워크의 성능을 실시간으로 저장하여 관측하기위해 tensorboardX의 함수 정의

Ex) Programming structure & code review

```
    def sample(self, batch_size):
        return random.sample(self.memory, batch_size)

    def __len__(self):
        return len(self.memory)

class DQN(nn.Module):

    def __init__(self):
        super(DQN, self).__init__()
        self.conv1 = nn.Conv2d(3, 16, kernel_size=5, stride=2)
        self.bn1 = nn.BatchNorm2d(16)
        self.conv2 = nn.Conv2d(16, 32, kernel_size=5, stride=2)
        self.bn2 = nn.BatchNorm2d(32)
        self.conv3 = nn.Conv2d(32, 32, kernel_size=5, stride=2)
        self.bn3 = nn.BatchNorm2d(32)
        self.head = nn.Linear(448, 2)

    def forward(self, x):
        x = F.relu(self.bn1(self.conv1(x)))
        x = F.relu(self.bn2(self.conv2(x)))
        x = F.relu(self.bn3(self.conv3(x)))
        return self.head(x.view(x.size(0), -1))
```

batch 사이즈에 맞게 저장된 데이터 셋을 랜덤하게 선택해주는 함수

메모리에 할당된 사이즈 출력 함수

DQN 모델을 정의하는 클래스

DQN 네트워크의 layer구조를 설정 여기서는 CNN을 사용

Forward step에서 넣어줄 activation function 정의

Ex) Programming structure & code review

```
TARGET_UPDATE = 10

policy_net = DQN().to(device)
target_net = DQN().to(device)
target_net.load_state_dict(policy_net.state_dict())
target_net.eval()

optimizer = optim.RMSprop(policy_net.parameters())
memory = ReplayMemory(10000)


steps_done = 0


def select_action(state):
    global steps_done
    sample = random.random()
    eps_threshold = EPS_END + (EPS_START - EPS_END) * \
        math.exp(-1. * steps_done / EPS_DECAY)
    steps_done += 1
    if sample > eps_threshold:
        with torch.no_grad():
            return policy_net(state).max(1)[1].view(1, 1)
    else:
        return torch.tensor([[random.randrange(2)]], device=device, dtype=torch.long)
```

위에서 설정한 모델 정의(double DQN을 두 모델 정의)

최적화 알고리즘 정의(여기서는 RMS 사용)

Replaybuffer사이즈 및 함수 선언

Epsilon greedy 알고리즘에 따른 action 선택 함수

Ex) Programming structure & code review

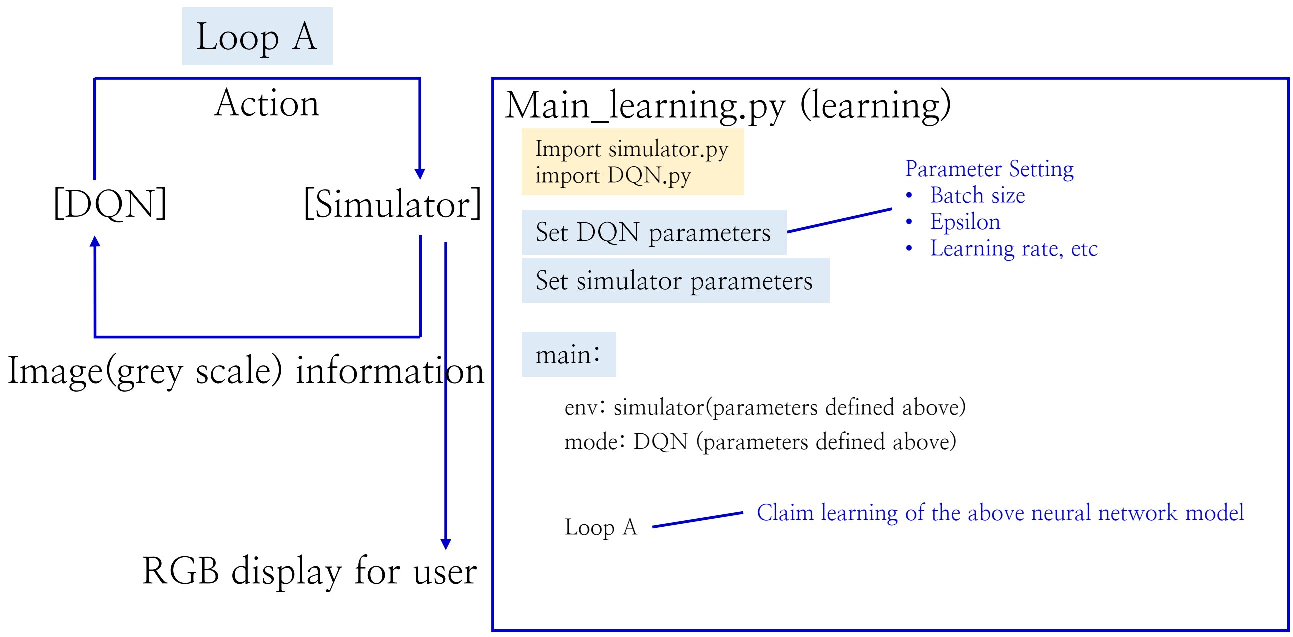
전체 프로그래밍의 기능적 모듈 등에 대한 분석

Also analyze the functional modules of main-simulator-test_DQN code

Main.py (learning)	Simulator.py	Test_DQN.py

Ex) Programming structure & code review

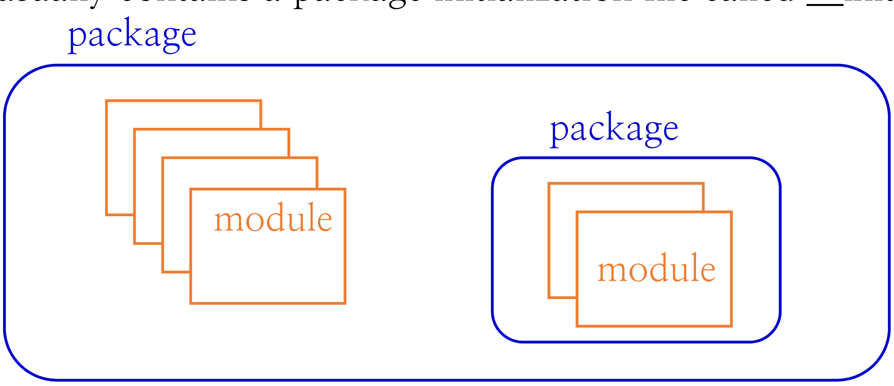
- 각 모듈 코드의 개념, 구조, 세부 내용에 대한 리뷰 작성
- Review the concept, sub-structure of the code



Q: Are you familiar to functions, modules, packages, class in Python?

Python package

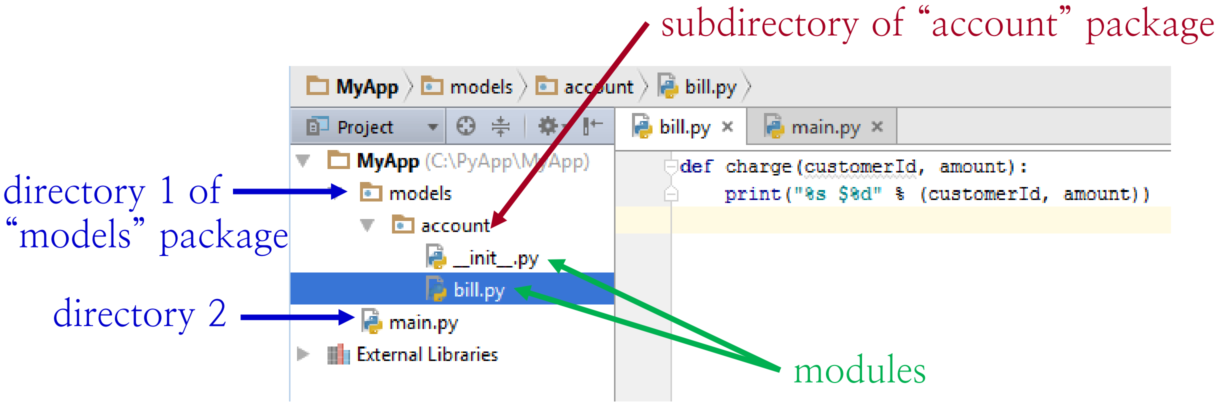
- 파이썬에서 모듈은 하나의 .py 파일을 가리키며, 패키지는 이러한 모듈들을 모은 컬렉션을 가리킴
- In Python, a module means a single .py file, and a package means a collection of these modules.
- 파이썬의 패키지는 하나의 디렉토리에 놓여진 모듈들의 집합을 가리키는데, 그 디렉토리에는 일반적으로 __init__.py 라는 패키지 초기화 파일이 존재
- A package in Python refers to a set of modules placed in a directory, which usually contains a package initialization file called __init__.py



http://pythonstudy.xyz/python/article/

Python package

- 파이썬으로 큰 프로젝트를 수행하게 될 때, 모든 모듈을 한 디렉토리에 모아 두기 보다는 각 영역별로 디렉토리/서브디렉토리를 만들고 계층적인 카테고리로 묶어서 패키지별로 관리하는 것이 편리하고 효율적이다.
- When you run a large project with Python, it's convenient and efficient to group all the modules in a hierarchical under the directories/subdirectories rather than a directory and manage them on a package-by-package basis



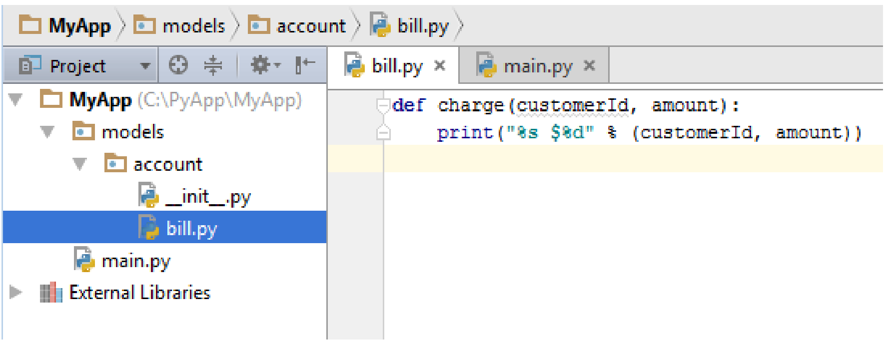
http://pythonstudy.xyz/python/article/

Import Python package

- 패키지내 모듈을 import하기 위해서는 "import 패키지명.모듈명"과 같이 패키지명을 앞에 붙여 사용한다.
- To import the module in the package, use the package name prefixed with "import package name.module name".

```
# 모듈 import
# import 패키지.모듈
import models.account.bill
models.account.bill.charge(1, 50)
```

package name module name function name



Import Python package

- To import a module, try "from package import module"
- To import a function, try “from package.module import function”

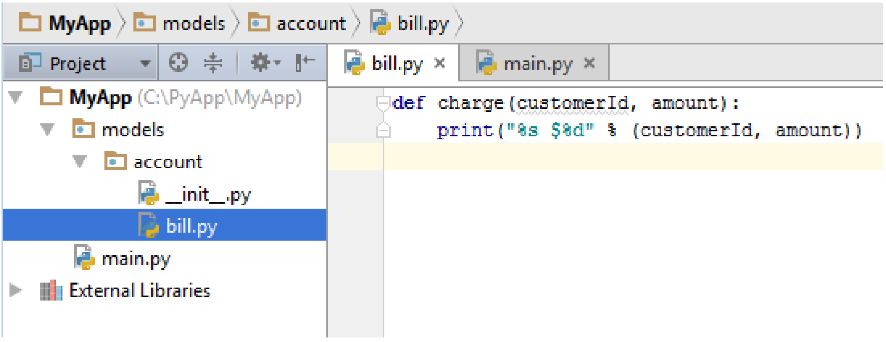
```
# 모듈안의 모든 함수 import
# from 패키지명 import 모듈명
from models.account import bill
bill.charge(1, 50)

# 특정 함수만 import
# from 패키지명.모듈명 import 함수명
from models.account.bill import charge
charge(1, 50)
```

module name

package name

function name

package + module name



http://pythonstudy.xyz/python/article/

Machine learning code…

```
import torch
class MyReLU(torch.autograd.Function):
    """ XXXX
def forward(ctx, input):
    """ XXXX


def backward(ctx, grad_output):
    """ XXXX
input, = ctx.saved_tensors
    grad_input = grad_output.clone()
    grad_input[input < 0] = 0
    return grad_input


dtype = torch.float
```

[http://busterworld.tistory.com/64 [BW]]

Python Class

- 클래스는 데이터와 그에 대한 처리를 하나로 정의한 것으로 함수와 유사 개념
- A class defines data and its processing as one group

```
class Bdb:
    """XXXXXXXX

    count = 0

    def __init__(self, skip=None):
        self.skip = set(skip) if skip else None
        self.breaks = {}
        self.fncache = {}

    def canonic(self, filename):
        if filename == "<" + filename[1:-1] + ">":
            return filename
```

Class name

Description code

http://www.dongwun.com/tc/142

Method: member of Python class

- 메서드(method): 클래스의 행위를 표현하는 것으로 클래스 내의 함수
- Method is a function in a class expressing the behavior of the class

```
class Bdb:
    """XXXXXXXX

    count = 0

    def __init__(self, skip=None):
        self.skip = set(skip) if skip else None
        self.breaks = {}
        self.fncache = {}

    def canonic(self, filename):
        if filename == "<" + filename[1:-1] + ">":
            return filename
```

Method

http://pythonstudy.xyz/python/article/

Class variables

- 클래스 변수 (class variable)은 메서드 밖에 존재하는 변수로 "클래스명.변수명" 으로 엑세스 할 수 있다.
- Class variables is defined outside the method by “class.variable”

```
class Bdb:

    """XXXXXXXX

    count = 0

    def __init__(self, skip=None):
        self.skip = set(skip) if skip else None
        self.breaks = {}
        self.fncache = {}

    def canonic(self, filename):
        if filename == "<" + filename[1:-1] + ">":
            return filename
```

Class variable, accessed by ‘Bdb.count’

Instance variables

- 인스턴스 변수(Instant variable)는 메서드 안에서 사용되면서 "self.변수명" 처럼 사용된다.
- Instant variable is used in a method, such as "self.variable"

```
class Bdb:

    """XXXXXXXX

    count = 0

    def __init__(self, skip=None):
        self.skip = set(skip) if skip else None
        self.breaks = {}
        self.fncache = {}

    def canonic(self, filename):
        if filename == "<" + filename[1:-1] + ">":
            return filename
```

Class initializer that sets the
- Instance variable
- Initialize the object

Instance variable

Object in Python Class

- 클래스를 사용하기 위해 객체(Object)를 생성해야 하며 "객체변수명 = 클래스명()"과 같이 클래스명을 함수 호출하는 것처럼 사용하면 된다.
- To use a class, you first need to create an object from the class. To create an object in Python, use the class name as if you were calling a function like "object variable name = class name ()".

```python
class Rectangle:
    count = 0  # 클래스 변수

    # 초기자(initializer)
    def __init__(self, width, height):
        # self.* : 인스턴스변수
        self.width = width
        self.height = height
        Rectangle.count += 1

    # 메서드
    def calcArea(self):
        area = self.width * self.height
        return area
```

Class “Rectangle” is assigned to an object “r”

```python
# 객체 생성
r = Rectangle(2, 3)

# 메서드 호출
area = r.calcArea()
print("area = ", area)

# 인스턴스 변수 엑세스
r.width = 10
print("width = ", r.width)

# 클래스 변수 엑세스
print(Rectangle.count)
print(r.count)
```


Inheritance of Class

- (부모)클래스는 다른(자식) 클래스로 상속되며 상속 받기 위해서는 파생클래스(자식클래스)에서 클래스명 뒤에 베이스클래스(부모클래스) 이름을 괄호와 함께 넣어 주면 된다.
- To inherit a class, you can put the base class (parent class) name in parentheses in the derived class (child class), followed by the class name.

```python
class Animal:
    def __init__(self, name):
        self.name = name
    def move(self):
        print("move")
    def speak(self):
        pass

class Dog (Animal):
    def speak(self):
        print("bark")

class Duck (Animal):
    def speak(self):
        print("quack")
```

Dog, Duck are child class that inherits the functions from parent class “Animal”

Inheritance of Class

```python
class Animal:
    def __init__(self, name):
        self.name = name
    def move(self):
        print("move")
    def speak(self):
        pass

class Dog (Animal):
    def speak(self):
        print("bark")

class Duck (Animal):
    def speak(self):
        print("quack")
```

자식클래스는 부모클래스의 멤버들을 호출하거나 사용할 수 있다

Child classes can call or use members of the parent class

```python
dog = Dog("doggy") # 부모클래스의 생성자
n = dog.name # 부모클래스의 인스턴스변수
dog.move()   # 부모클래스의 메서드
dog.speak()  # 파생클래스의 멤버
```

http://pythonstudy.xyz/python/article/

Active learning : week 1

- Python review
- Python module, package, class coding practice
- PyTorch review

1.0 Machine learning basic

AI, Machine learning, Deep learning

- Artificial Intelligence: technique which enables computers to mimic human behavior
  - Machine learning: Subset of AI technique which uses statistical methods to enables machines to improve with experiences
    - Deep learning: Subset of ML which make the computation of multi-layer neural networks feasible

CNN

RL

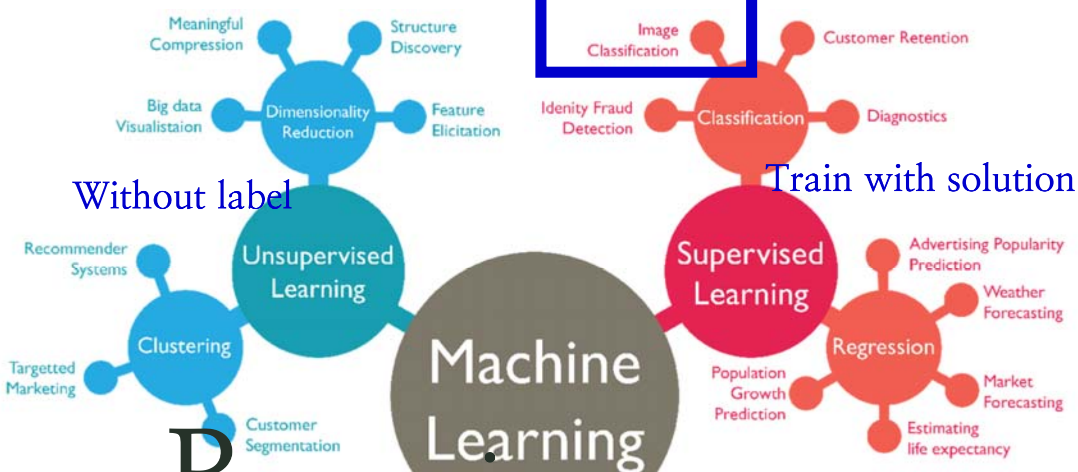
RNN

LSTM

추천: Youtube 기계학습(Machine learning) 입문하기 강의, Terry TaeWoong Um

Types of Machine learning

Capstone design 2



Without label

Train with solution(label)

Clustering Regression Classification, Regression

ɿuential actions

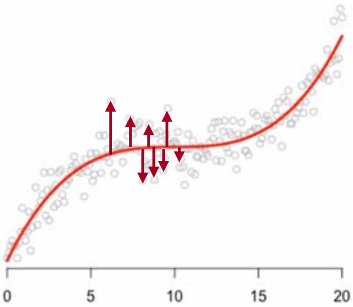
Caɿ

[images from google]

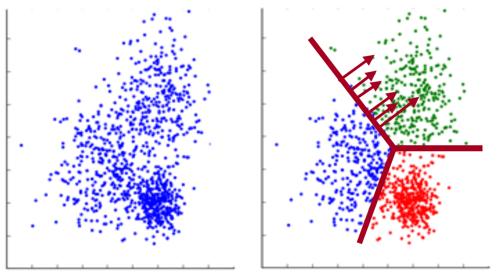
Machine learning (= evolving Regression)

- A computer program is said to learn from experience E with respect to some class of task T and performance measure P, if its performance at task in T, as measured by P, improves with experience E. (Mitchell, 1997)
- Task : classification, regression, clustering
- Performance (loss function): errors, distance
- Experience: data (labeled, unlabeled)

Rate of correct answers



Distance(=regression error) from model (y = f(x³))



Distance from lines
(Supporting Vector Machine)

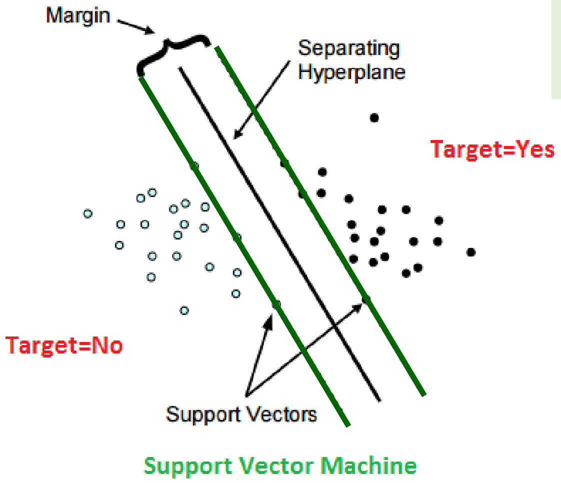
https://www.youtube.com/watch?v=j3za7nv7RfI, images from google

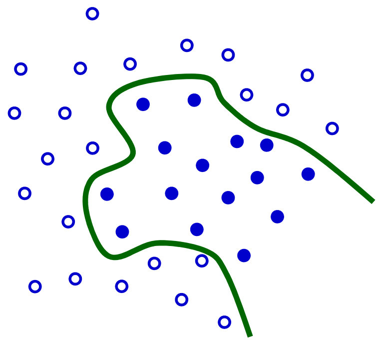
Classification algorithm (1 of many!)

- Support vector machine (SVM) model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible.

Q: how to find supporting vectors?

Q: how to find a character(s) that separates the data for classification?



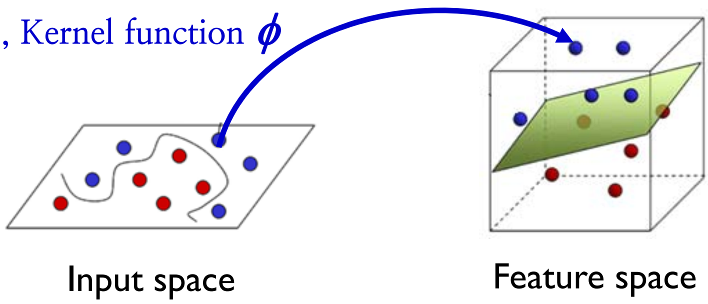


[images from google]

Feature

- 머신러닝에서 판별하고자 하는 데이터에 대해서 판별의 근거로 활용할 수 있는 데이터의 특징들을 feature라고 한다
- In machine learning and pattern recognition, a feature is an individual measurable property or characteristic of a phenomenon being observed
- Choosing informative, discriminating and independent features is a crucial step for effective algorithms in machine learning
- Kernel function which enable the data in raw representation to operate in a high–dimensional, implicit "feature" space

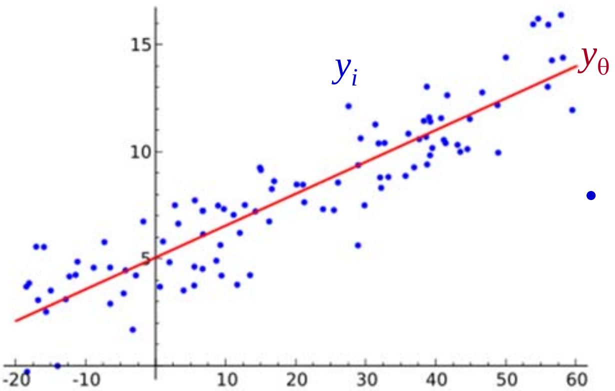
Coordinate transfer,
i.e., Kernel function ϕ



✓ Then find SVM = maximize the distance from separation lines(planes)

[images from google]

Mathematical formulation of regression



- Model/feature (= regression function)

$$y_\theta(x_i) = \theta^T x_i + b$$

- Loss function (= cost function)

$$J(\theta) = \sum_{i=1} \left(y_\theta(x_i) - y_i \right)^2$$

- Mean Squared Error (MSE)

$$MSE = \frac{1}{N} \sum_{i=1}^{N} \left(y_i - \bar{y}_i \right)^2$$

- Optimization

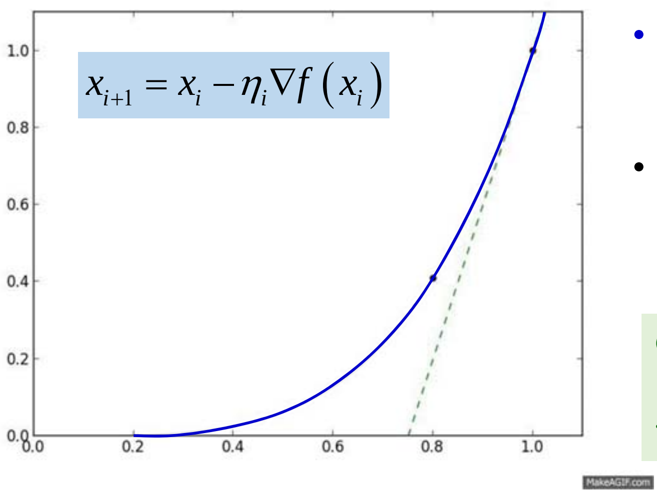
$$\theta^* = \arg\min_\theta J(\theta)$$

Q: How to solve this?

[images from google]

Finding min MSE by Gradient Descent

- Gradient descent is a first-order iterative optimization algorithm for finding the minimum of a function. To find a local minimum of a function using gradient descent, one takes steps proportional to the negative of the gradient (or approximate gradient) of the function at the current point iteratively. [wiki]



- 배치(batch): 모델 학습의 반복 1회, 즉 경사 업데이트 1회에 사용되는 예(데이터)의 집합.
- Batch: the set of examples used in one iteration (that is, one gradient update) of model training.

Q: If you have a VERY LARGE batch, even a single iteration may take a very long time to compute.

[images from google]

Stochastic gradient descent (SGD)

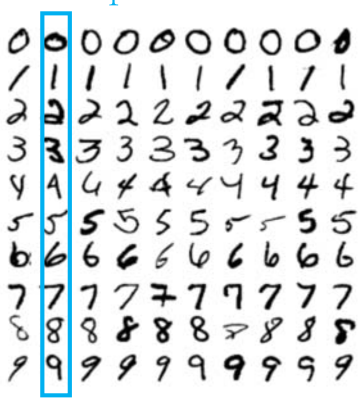
- To minimize the loss function, a standard (or "batch") gradient descent method would perform the following iterations :

$$w' = w - \eta \nabla f(w) \quad = w - \eta \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(w)$$

i^{th} observation (example) in the dataset of total size n used for training

$\sum$

A sample



- A sample = instance = an observation = an input vector = a feature vector
- **When the training set is enormous and no simple formulas exist, evaluating the sums of gradients becomes very expensive, because evaluating the gradient requires evaluating all the summand functions' gradients.**

[https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/ & Wikipedia]

Batch, Epoch

- Epoch is an ENTIRE dataset passed through neural network only ONCE
  - Number of epochs defines the number times that the learning algorithm will work through the entire training dataset (sort of 재활용)
  - Since, one epoch is too big to feed to the computer at once we divide it in several smaller batches
  - Updating the weights using iterative GD with one epoch may be not enough

A batch

- Batch Size: a number of samples processed before the model is updated
  - Tune it (BS) for training speed, model quality, computational power

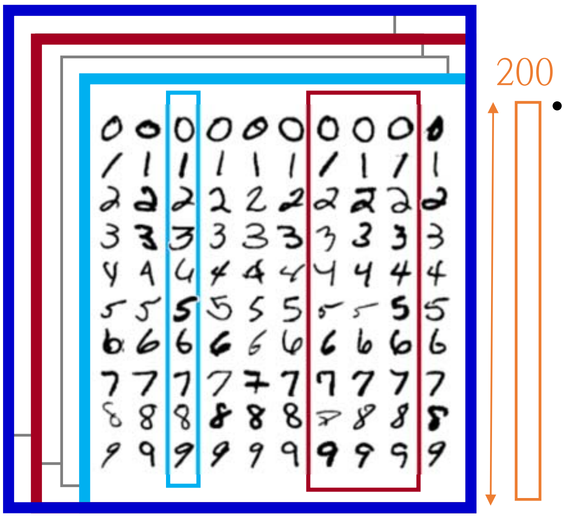
[https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/ & Wikipedia]

Stochastic gradient descent (SGD)

- Batch Gradient Descent: Batch Size = Size of Training Set
- Stochastic Gradient (randomly select) Descent: Batch Size = 1
  - Pros: Faster calculation, Cons: May cause nosy results

$$\boxed{w' = w - \eta \nabla f(w) \approx w - \eta \nabla f_i(w)}$$

- Mini-Batch Gradient Descent: 1 〈 Batch Size 〈 Size of Training Set



- Ex) A dataset with 200 samples, a batch size = 5, epochs = 1,000
  - One epoch will involve 40 batches or 40 updates to the model
  - With 1,000 epochs, the model will pass through the whole dataset 1,000 times. That is a total of 40,000 batches during the entire training process

[https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/ & Wikipedia]

Stochastic gradient descent (SGD)

- As the algorithm sweeps through the training set, it performs the above update for each training example. Several passes can be made over the training set until the algorithm converges.
- If this is done, the data can be shuffled for each pass to prevent cycles.
- Typical implementations may use an adaptive learning rate so that the algorithm converges.
- Pseudocode
  - Choose an initial vector of parameters w and learning rate η
  - Repeat until an approximate minimum is obtained:
    - Randomly shuffle examples in the training set.
    - For i=1,2,...,n, do:

$$w' = w - \eta \nabla f_i(w)$$

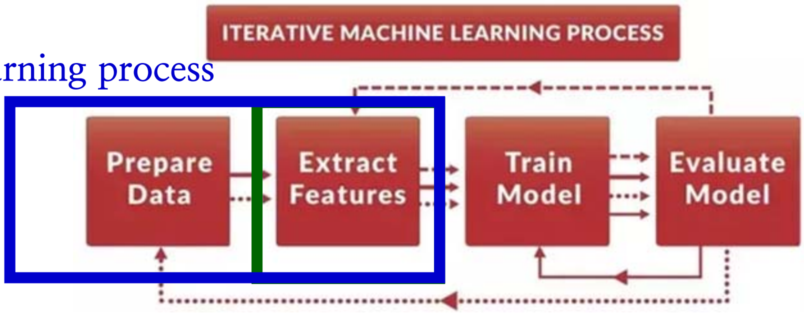
[https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/ & Wikipedia]

Machine learning process

Greater number of features, or even features with non-linear characteristics will make the regression process infeasible: Neural Net

1) Select a parametric/nonparametric model (linear, kernel etc.)
2) Set a performance measurement (loss function)
3) Training data (optimizing model parameter)
4) Evaluate the final performance using test data

Deep learning process

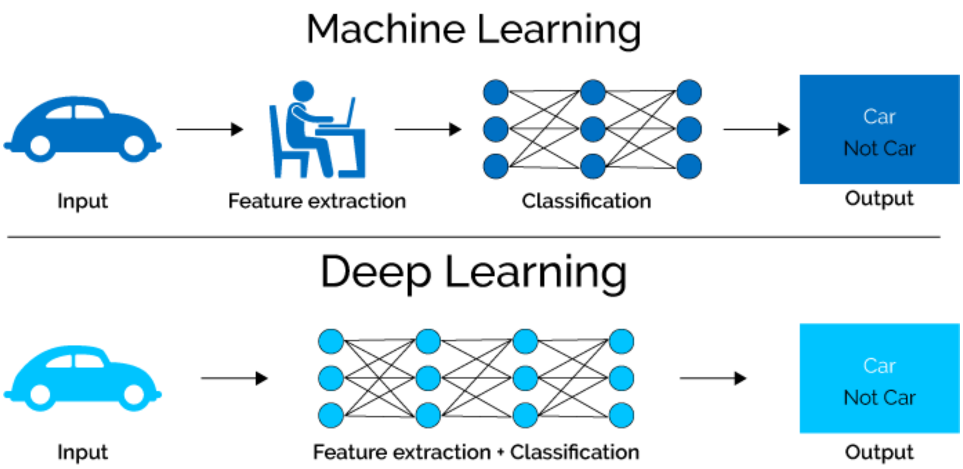


- Deep learning uses a cascade of multiple layers of nonlinear processing units for feature extraction and transformation.

[images from google]

Deep learning

Feature learning, end-to-end learning

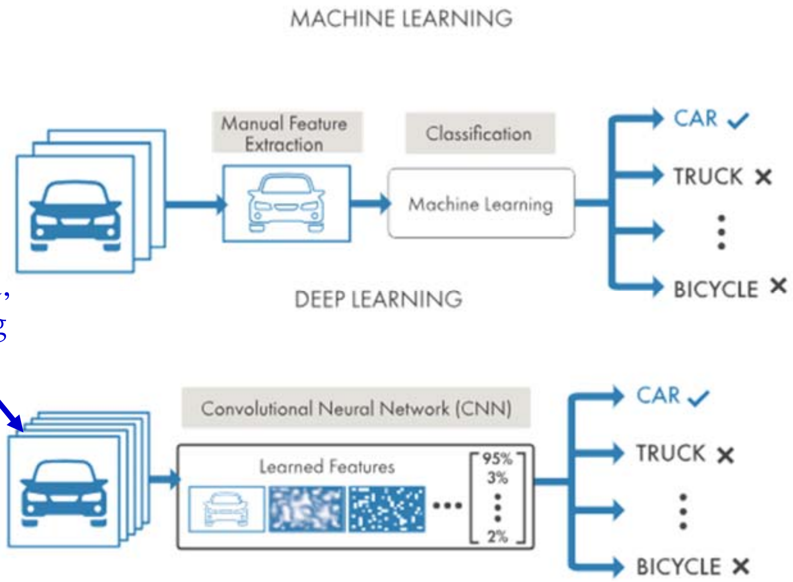


http://www.umiacs.umd.edu/~kanazawa/
https://medium.com/swlh/ill-tell-you-why-deep-learning-is-so-popular-and-in-demand-5aca72628780

Machine learning vs. deep learning

- In machine learning, you manually choose features and a classifier to sort images. With deep learning, feature extraction and modeling steps are automatic (so called, end−to−end learning).
- A key advantage of deep learning networks is that they often continue to improve as the size of your data increases.

require MANY data, hard to debug



https://www.mathworks.com/discovery/deep-learning.html#withmatlab

Machine learning vs. deep learning

	Machine Learning	**Deep Learning**
Training dataset	Small	Large
Choose your own features	Yes	No
# of classifiers available	Many	Few
Training time	Short	Long

From Matlab machine learning tutorial

In class review

RL：Reinforcement learning

AI

SVM：support vector machine

Feature

ML：Machine learning

Regression/clustering/classification

Deep learning

SGD：stochastic gradient descent

Kernel function

Supervised/unsupervised learning

NN：Neural net

Loss function

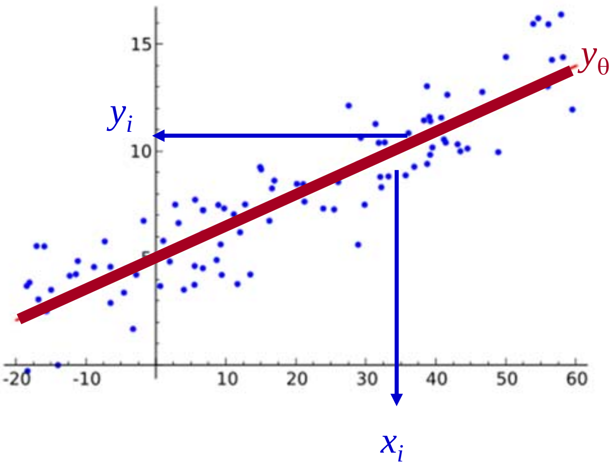
Programming ML

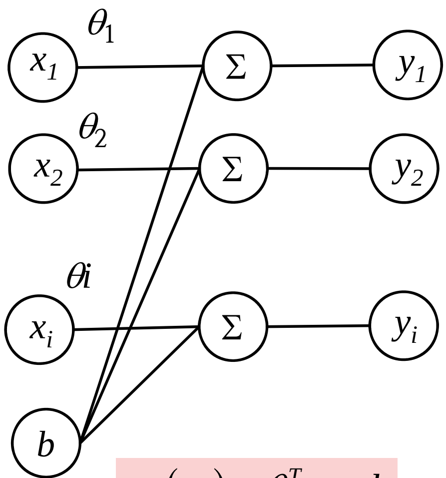
- Regression
- Basic PyTorch programming

1.1 Neural Network

Machine learning

- Try your own definition

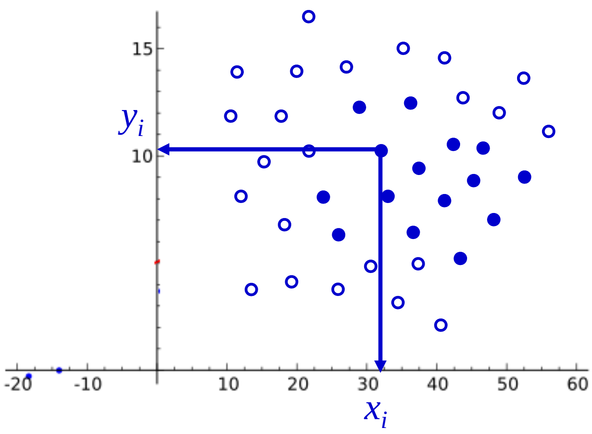


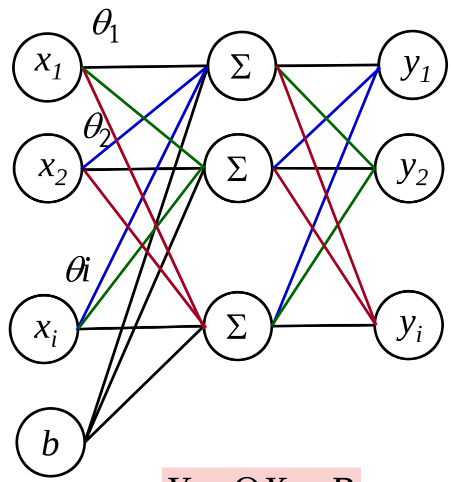


$$y_\theta(x_i) = \theta^T x_i + b$$

- Loss function (= cost function) $J(\theta) = \sum_{i=1} \left(y_\theta(x_i) - y_i \right)^2$
- Mean Squared Error (MSE) $MSE = \frac{1}{N} \sum_{i=1}^{N} (y_i - \bar{y}_i)^2$

Machine learning ⊇ Neural network



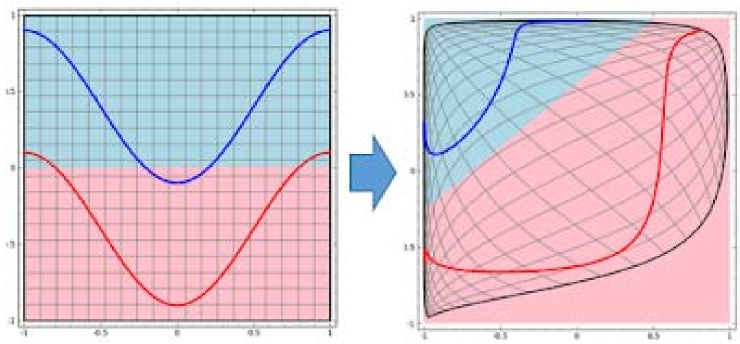


$$Y = \Theta X + B$$

- Loss function (= cost function) $$J(\theta) = \sum_{i=1} \left(y_\theta (x_i) - y_i \right)^2$$
- Mean Squared Error (MSE) $$MSE = \frac{1}{N} \sum_{i=1}^{N} \left(y_i - \bar{y}_i \right)^2$$

Neural Network

- 신경망이란 데이터를 잘 구분하기 위해 데이터 공간들을 잘 왜곡해 (e.g. kernels) 선들을 긋고 (e.g. SVM) 선형 맞춤 (linear fitting)과 비선형 변환 (nonlinear transformation or activation)을 반복하여 구분결과가 더 잘 나오게 하는 과정(optimization)을 포함하는 구조라고 할 수 있다
- Neural networks are structures that are repeatedly built up with linear fitting and nonlinear transformation or activation to distinguish data

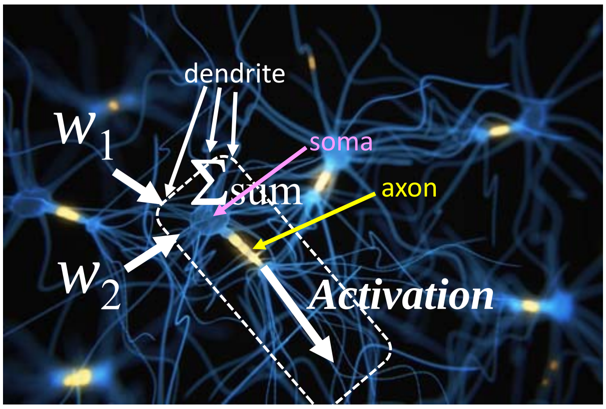


(사진출처: colah's blog)

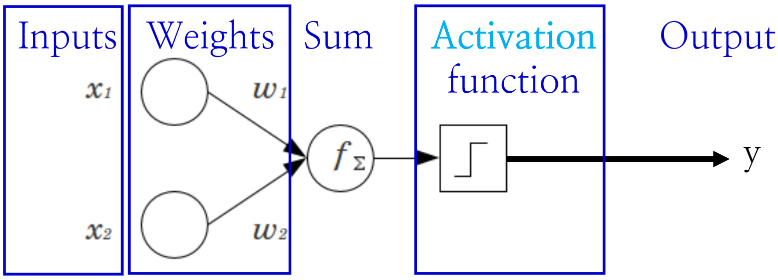
- 파란선과 빨간선의 영역을 직선으로 구분한다고 하면 불완전한 구분이 되므로 (왼쪽), 공간을 왜곡하면 오른쪽과 같이 직선으로 구분가능하다
- 이처럼 인공신경망은 선 긋고, 구기고, 합하고를 반복하여 데이터를 처리한다

http://t-robotics.blogspot.com/2015/05/deep-learning.html#.W2ke9SgzY2w

Perceptron: a basic unit of neural network



Activation function
- Step function
- Sigmoid(=logistic) function
- ReLU
- Softmax

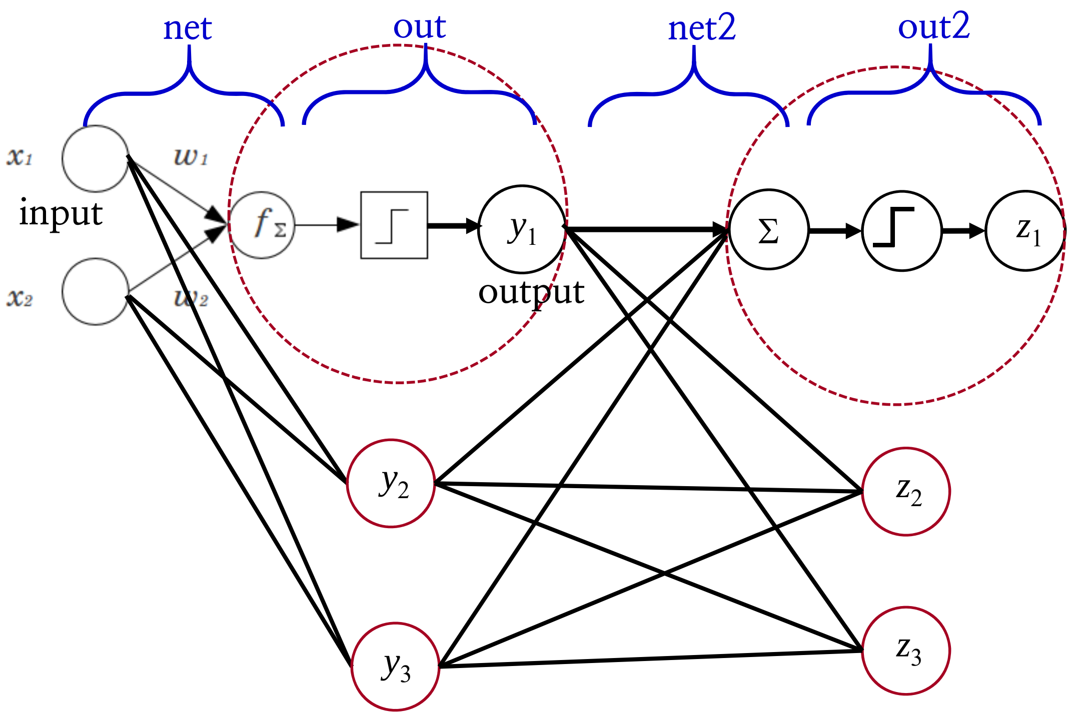


node

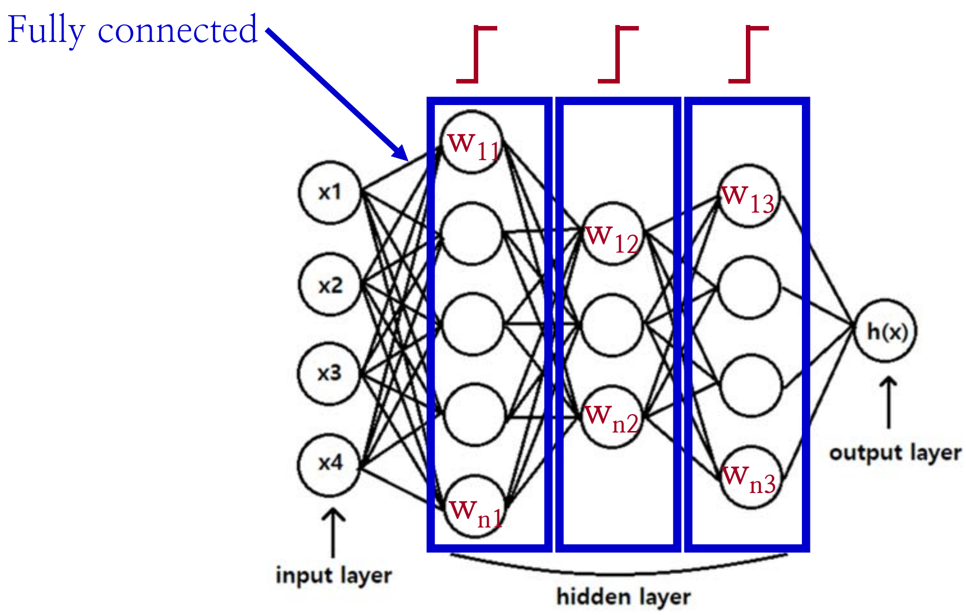
So an artificial neuron simply calculates a "weighted sum" of its input and then decides whether it should be "fired" or not (= activation)

[images from google]

Multilayer perceptron



Multilayer perceptron



Q: how to find w's?

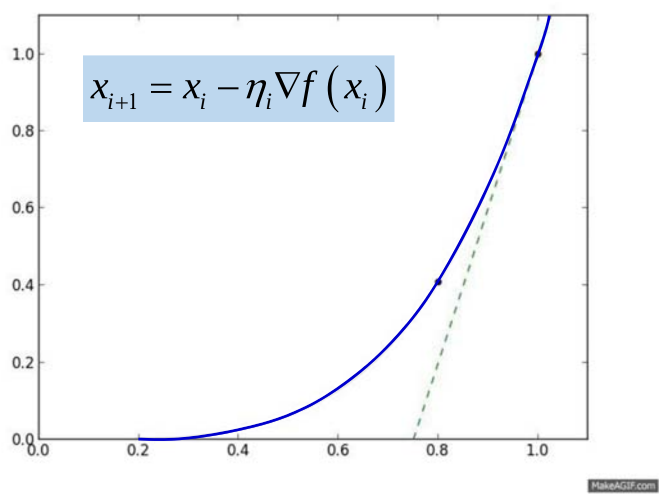
A: Tune 'w' to minimize the error btw. the true outputs and the estimated outputs ≡ loss function

[images from google]

Tune weight w to minimize the loss function

- Recall, Gradient descent

  Find x^* such that min $f(x^*)$



- Backpropagation

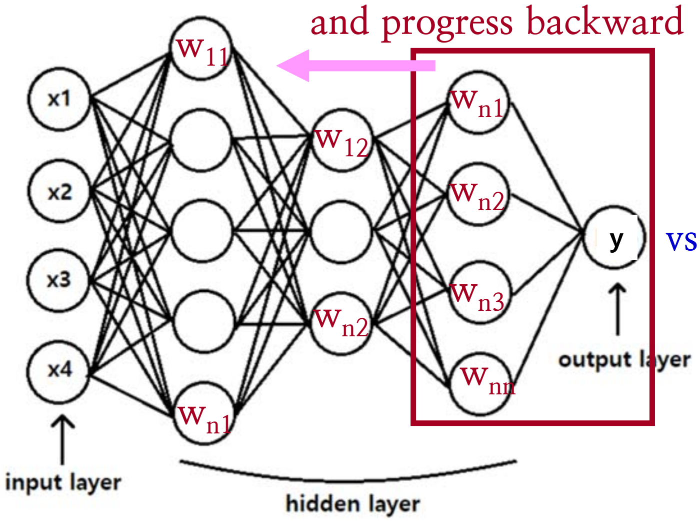
$$w' = w - \eta \underline{\nabla f}\left(w\right)$$

Q: What is the loss function f(w)?

A: Error between the NN output & true values

Backpropagation

- To tune ‘w’, starting from the outer most output, and progress backward



vs. y_{true}

$$f(w) = f\left(y_{true} - \underline{y_{NN}}\right)^2$$

$$= f\left(y_{true} - \underline{Wx}\right)^2$$

Adjust “w” to minimize loss function f

$$w' = w - \eta \nabla \underline{f}(w)$$

Q: What is the loss function f(w)?

A: Error between the NN output & true values

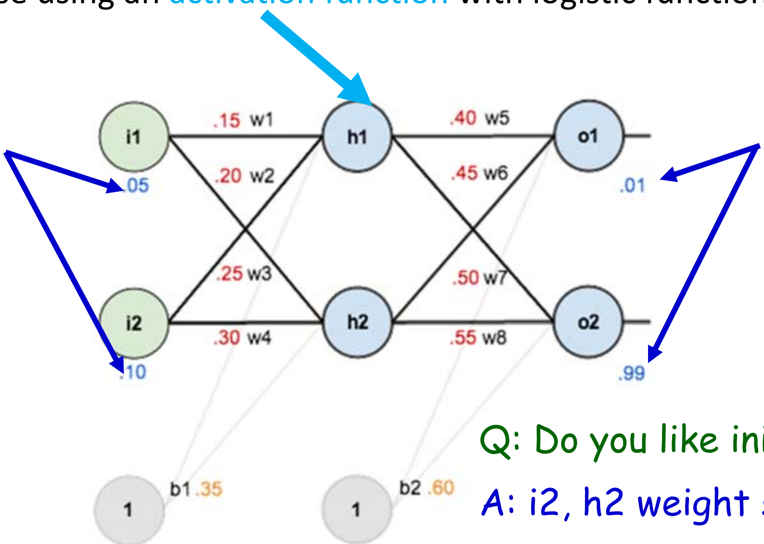
Backpropagation

- Backpropagation is a method used to calculate a gradient of the loss function to adjust the weights of nodes in the network.

Ex) Here are the initial weights, the biases, and training inputs/outputs

Suppose using an activation function with logistic function $y(x) = \frac{1}{1 + Ae^{Bx}}$

Given inputs 0.05 and 0.10



We want the neural network to output 0.01 and 0.99.

Q: Do you like initial guess?

A: i2, h2 weight should be greater

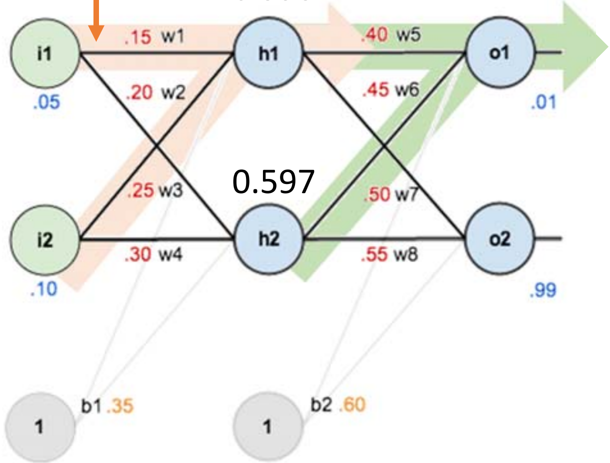
https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

Backpropagation

- Here's how we calculate the total net input for h1:

$net_{h1} = w1 * i1 + w2 * i2 + b1 * 1 = 0.15 * 0.05 + 0.2 * 0.1 + 0.35 * 1 = 0.3775$

- Using logistic function to get the output of h1: $out_{h_1} = \frac{1}{1+e^{-h_1}} = \frac{1}{1+e^{-0.3775}} = 0.593$
- Carrying out the same process for h2 we get: $out_{h2} = 0.597$



Repeat this process for the output layer neurons, using the output from the hidden layer neurons as inputs

+ (plus)

Activation function

Backpropagation

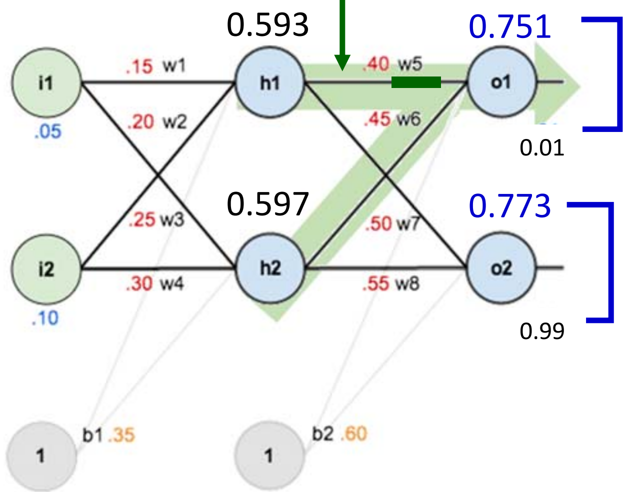
- Here's the output for o1:

$net_{o1} = w5 * out_{h1} + w6 * out_{h2} + b2 * 1 = 0.4 * 0.593 + 0.45 * 0.597 + 0.6 * 1 = 1.106$

$$out_{o_1} = \frac{1}{1+e^{-net_{o_1}}} = \frac{1}{1+e^{-1.106}} = 0.751$$

- Similarly, $out_{o2} = 0.773$

Q: how to change w5?



- Then calculate the loss function

$$E_{total} = \sum \tfrac{1}{2}(target - output)^2$$

$$E_{o_1} = \frac{1}{2}(0.01 - 0.751)^2 = 0.275$$

Q: what affects to out_{o1}?

$$E_{o_2} = 0.024$$

$$E_{total} = E_{o_1} + E_{o_2} = 0.298$$

Q: Which output component contributes to error more?

Backward pass

Q: We want to know how much a change in w5 affects the total error?

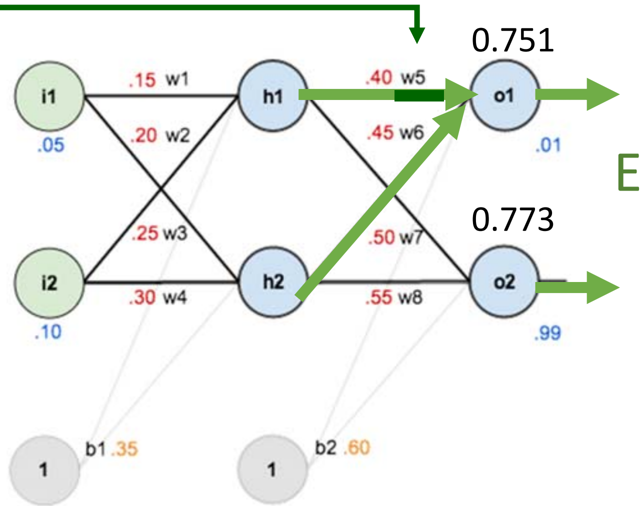
$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o_1}} \cdot \frac{\partial out_{o_1}}{\partial net_{o_1}} \cdot \frac{\partial net_{o_1}}{\partial w_5}$$

w_5 affects to E_{total} through o_1 by network + activation function

$$\frac{\partial E_{total}}{\partial out_{o_1}} = -(0.01 - o_1) = -(0.01 - 0.751) = 0.741$$

$$\frac{\partial out_{o_1}}{\partial net_{o_1}} = \frac{\partial}{\partial net_{o_1}}\left(\frac{1}{1+e^{-net_{o_1}}}\right) = 0.187$$

$$= \left(1 - \frac{1}{1+e^{-net_{o_1}}}\right)\left(\frac{1}{1+e^{-net_{o_1}}}\right) = (1-0.751)0.751$$



E_{total}

$$E_{o_1} = \frac{1}{2}(0.01 - \underbrace{0.751}_{out_{o1}})^2 = 0.275$$

$$E_{o_2} = 0.024$$

$$E_{total} = E_{o_1} + E_{o_2} = 0.298$$

Backward pass

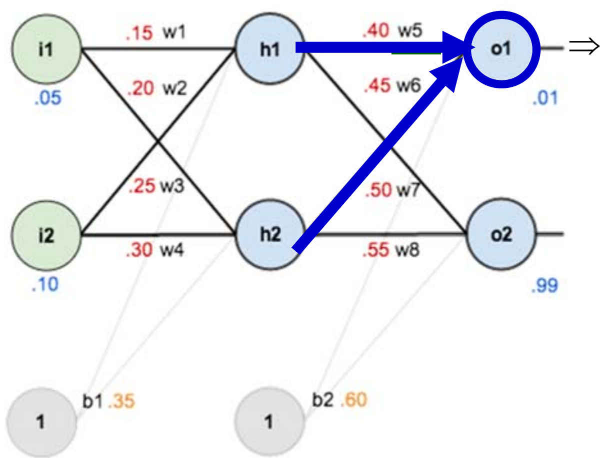
- Finally, how much does the total net input of o1 change with respect to w5?

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o_1}} \cdot \frac{\partial out_{o_1}}{\partial net_{o_1}} \cdot \frac{\partial net_{o_1}}{\partial w_5}$$

$$\frac{\partial E_{total}}{\partial out_{o_1}} = -(0.01 - o_1) = -(0.01 - 0.751) = 0.741$$

$$\frac{\partial out_{o_1}}{\partial net_{o_1}} = \frac{\partial}{\partial net_{o_1}}\left(\frac{1}{1+e^{-net_{o_1}}}\right) = 0.187$$

Recall, net_{o1} = w5 * out_{h1} + w6 * out_{h2} + b2 * 1



$$\Rightarrow \frac{\partial net_{o_1}}{\partial w_5} = out_{h_1} = 0.593$$

- Putting all together

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o_1}} \cdot \frac{\partial out_{o_1}}{\partial net_{o_1}} \cdot \frac{\partial net_{o_1}}{\partial w_5}$$

$$= 0.741 \cdot 0.187 \cdot 0.593 = 0.082$$

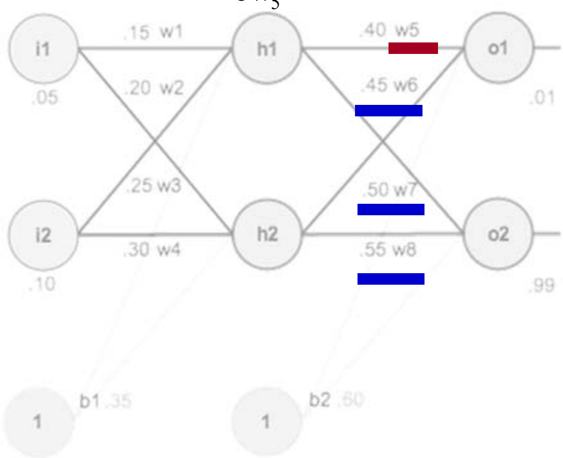
Backward pass

- Update weight: to decrease the error, we then subtract this value from the current weight (optionally multiplied by some learning rate, eta, which we'll set to 0.5):

$$w_5^{+} = w_5 - \eta \cdot \frac{\partial E_{total}}{\partial w_5} = 0.4 - 0.5 \cdot 0.082 = 0.359$$

Reduce weight w5 to reduce error (= loss function)

$$\frac{\partial E_{total}}{\partial w_5} = 0.082$$



We can repeat this process to get the new weights w6, w7, and w8:

$$w_6^{+} = 0.409$$
$$w_7^{+} = 0.511$$
$$w_8^{+} = 0.561$$

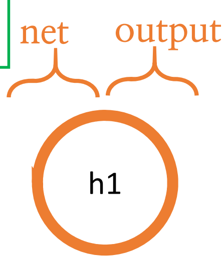
https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

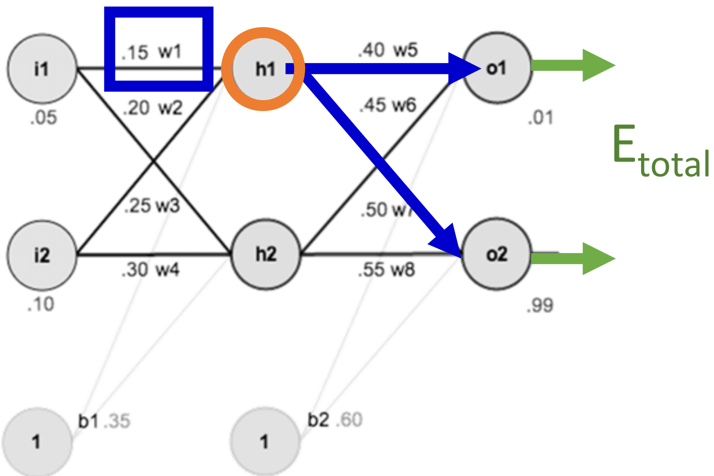
Hidden layer

- Next, we'll continue the backwards pass by calculating new values for w1, w2, w3, w4.

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h_1}} \cdot \frac{\partial out_{h_1}}{\partial net_{h_1}} \cdot \frac{\partial net_{h_1}}{\partial w_1} \quad \Rightarrow \frac{\partial E_{total}}{\partial w_1} = 0.00044$$

$$\frac{\partial E_{total}}{\partial out_{h_1}} = \frac{\partial E_{o_1}}{\partial out_{h_1}} \cdot \frac{\partial E_{o_2}}{\partial out_{h_1}}$$





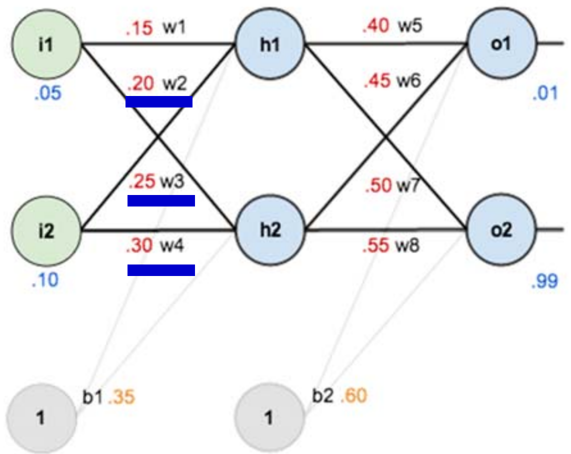
Hidden layer

- We can now update all of our weights w1, w2, w3, w4.

$$w_1^+ = w_1 - \eta \cdot \frac{\partial E_{total}}{\partial w_1} = 0.15 - 0.5 \cdot 0.00044 = 0.1497$$

Reduce weight w1 (very slightly) to reduce error (= loss function)

$$w_2^+ = 0.1996$$
$$w_3^+ = 0.2498$$
$$w_4^+ = 0.2995$$

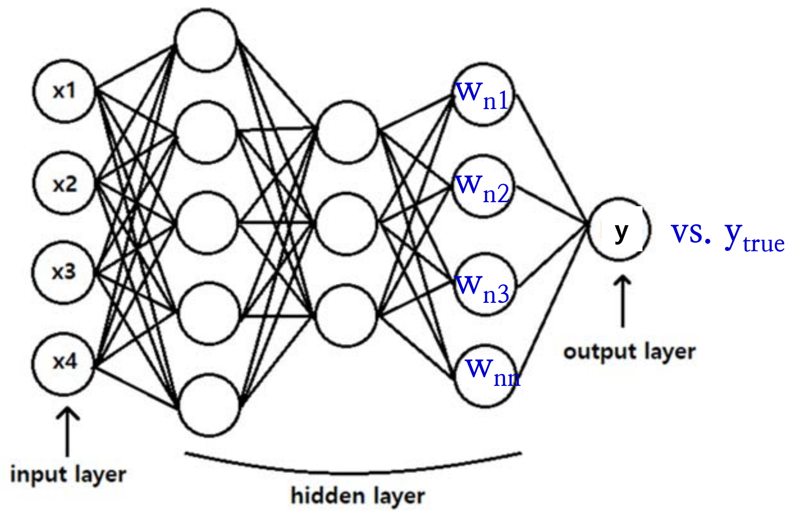


- Original network error E_{total} = 0.298
- After the first round of backpropagation, E_{total} = 0.291

It seems to be very small change, but after training (repeating) this 10^4 times (with different batches), for example, the error plummets to 0.0000351085.

https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

Recall, Backpropagation

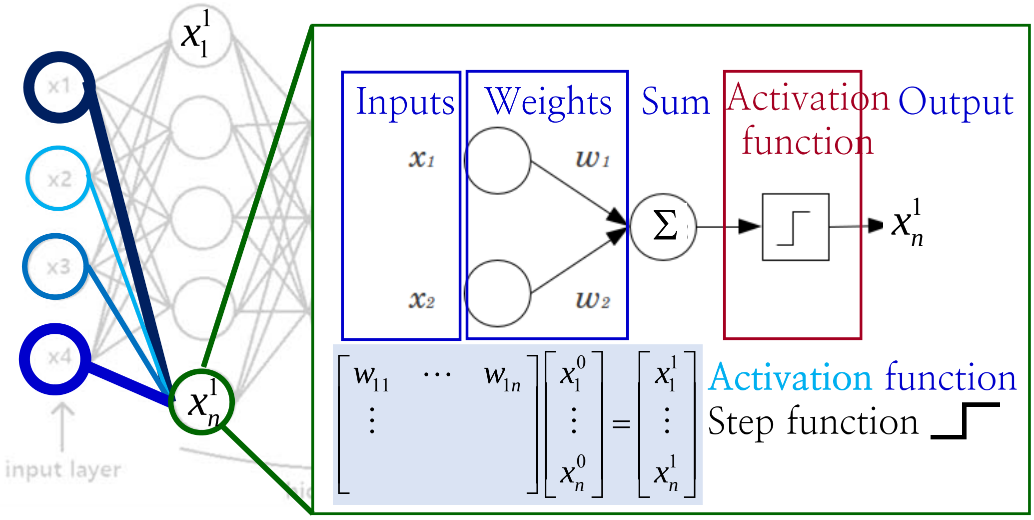


$$w' = w - \eta \nabla f(w)$$

- Derivative of output errors
- Output is a weighted sum of inputs with activation function

Images from google

Activation function in NN



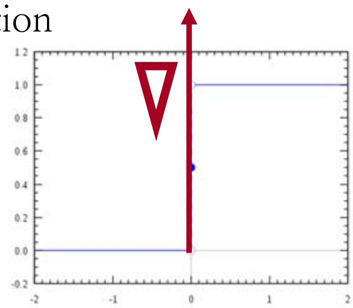
$$w' = w - \eta \nabla f(w)$$

- Derivative of output errors
- Output is a weighted sum of inputs with activation function

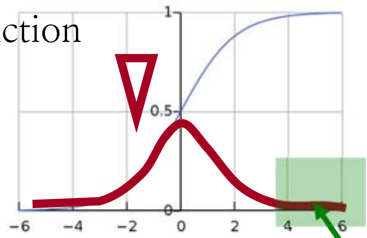
Images from google

Activation function

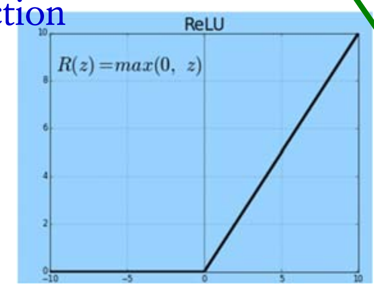
Step function

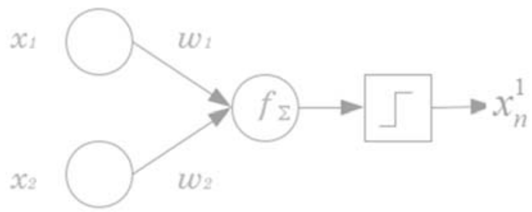


Logistic function



ReLU function





$$x_n^1 = \sigma\left(w_{1,1}x_1^0 + w_{1,2}x_2^0 + \cdots + w_{1,n}x_n^0 + b_1 \right)$$

Bias

Activation function, e.g. logistic

$$w' = w - \eta \nabla f(w)$$

To update the weight w, take a derivative of *f*, i.e., activation funciton

- Update of the weight (~ proportional to the partial derivative ∇ of the error function) would not occur if the gradient will be vanishingly small. This may completely stop the neural network from further training.

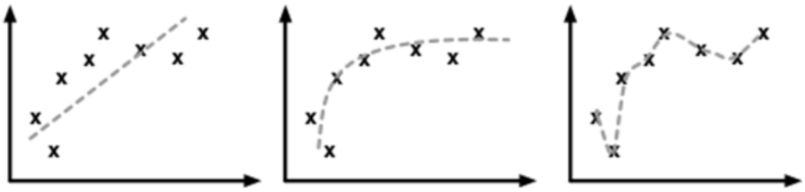
⇒ ‘Vanishing gradient’ problem

Images from google

Overfitting with training images

Under fitting

Over fitting



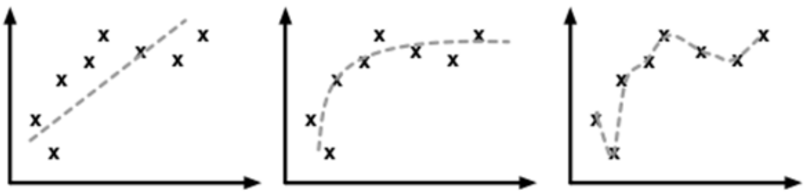
Theses are not cats because it's different from what has been trained

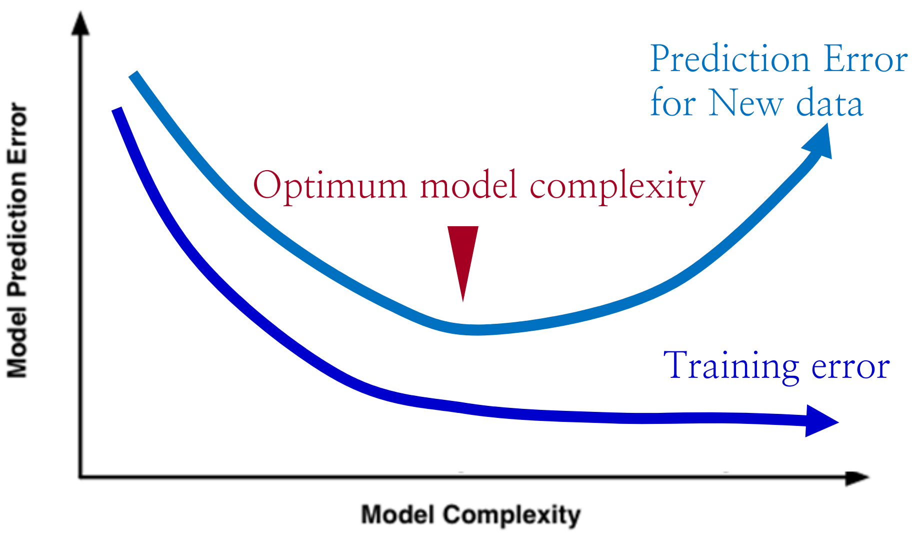
Q: How to avoid 'overfitting'?

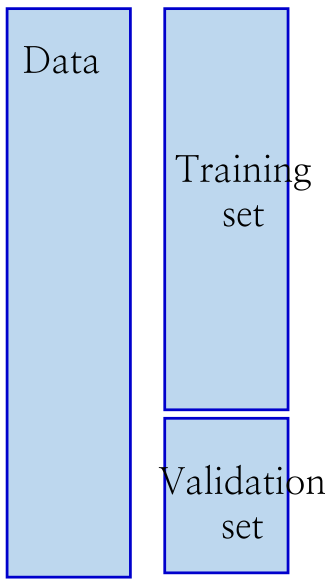
Training & test (,validation)

Under fitting

Over fitting



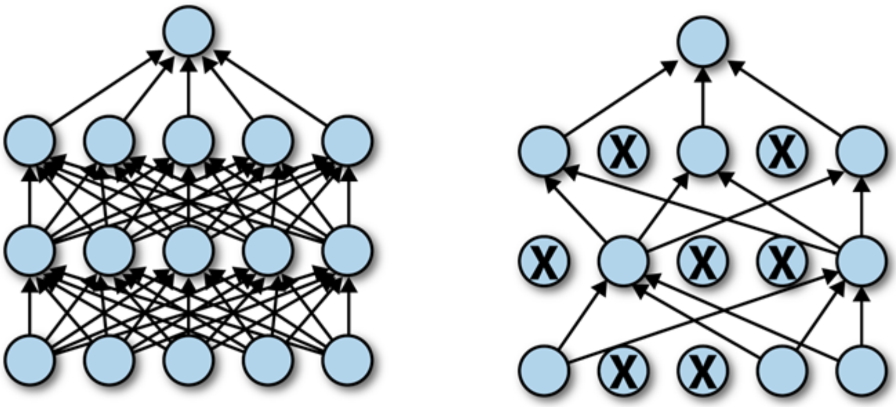




https://www.quora.com/What-is-generalization-in-machine-learning

Dropout

- Dropout is a form of regularization that randomly drops some proportion of the nodes that feed into a fully connected layer.
- This helps prevent the net from relying on one node in the layer too much.
- Here, dropping a node means that its contribution to the corresponding activation function is set to 0. Since there is no activation contribution, the gradients for dropped nodes drop to zero as well.



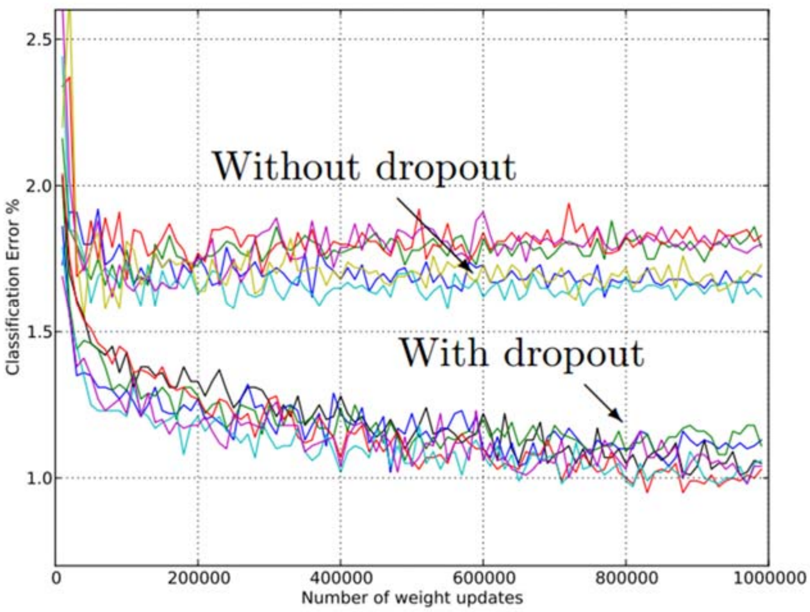
(a) Standard Neural Net

(b) After applying dropout

https://www.safaribooksonline.com/library/view/tensorflow-for-deep/9781491980446/ch04.html

Dropout

- Bellow we have a classification error (Not including loss), observe that the test/validation error is smaller using dropout



https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/dropout_layer.html

In class review

Backpropagation

Hidden layer

Neural Network (ANN)

Overfitting

Dropout

ReLU

Softmax

SGD: stochastic gradient descent

Fully connected

Vanishing gradient

Loss function

Programing NN

Basic code structures & details

1.2 Convolutional Neural Network

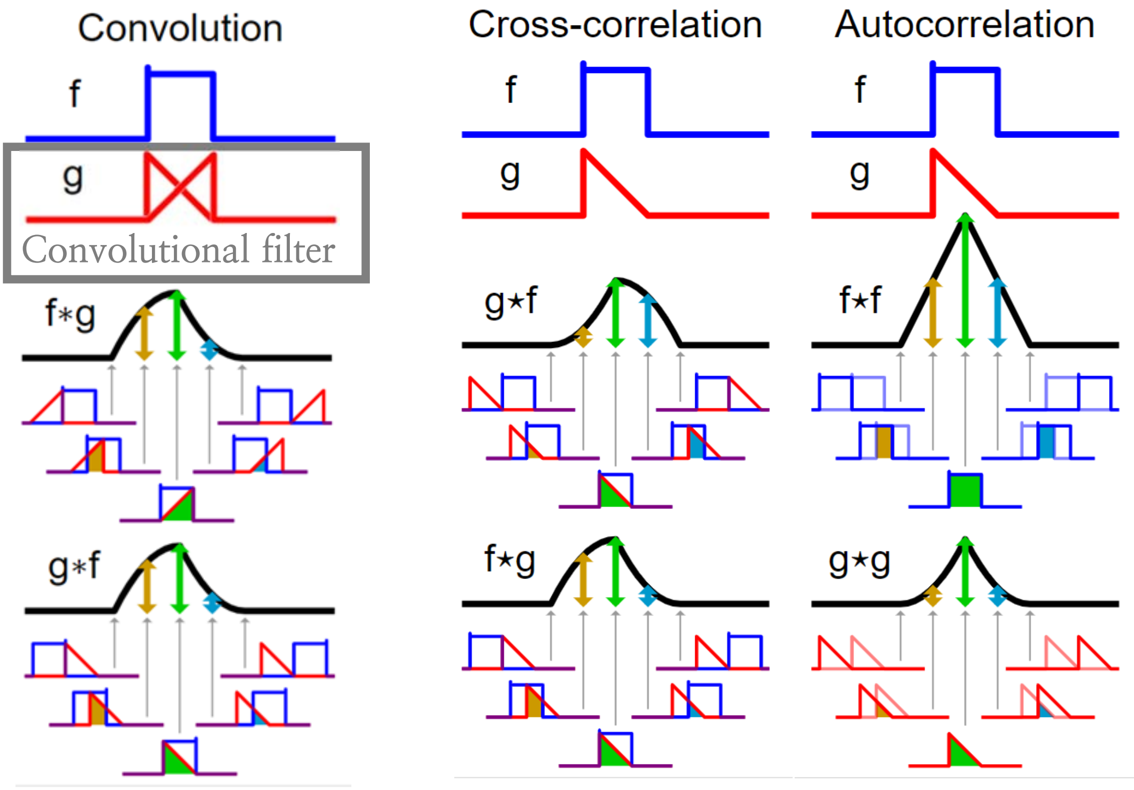
Convolution

- Convolve 감다; 감기다; 휘감다; 둘둘 말다[감다]; 빙빙 돌다
- Convolution
  - 대단히 복잡한[난해한] 것 ex) the bizarre convolutions of the story
  - (나선형의) 주름[구불구불한 것] the convolutions of the brain

$$(f*g)(t) \equiv \int_{-\infty}^{\infty} \underline{f(\tau)g(t-\tau)}\,d\tau = \int_{-\infty}^{\infty} f(t-\tau)g(\tau)\,d\tau$$

- 합성곱(convolution)은 하나의 함수와 또 다른 함수를 반전 이동한 값을 곱한 다음, 구간에 대해 적분하여 새로운 함수를 구하는 수학 연산자이다. a mathematical operation on two functions (f and g) to produce a third function that expresses how the shape of one is modified by the other

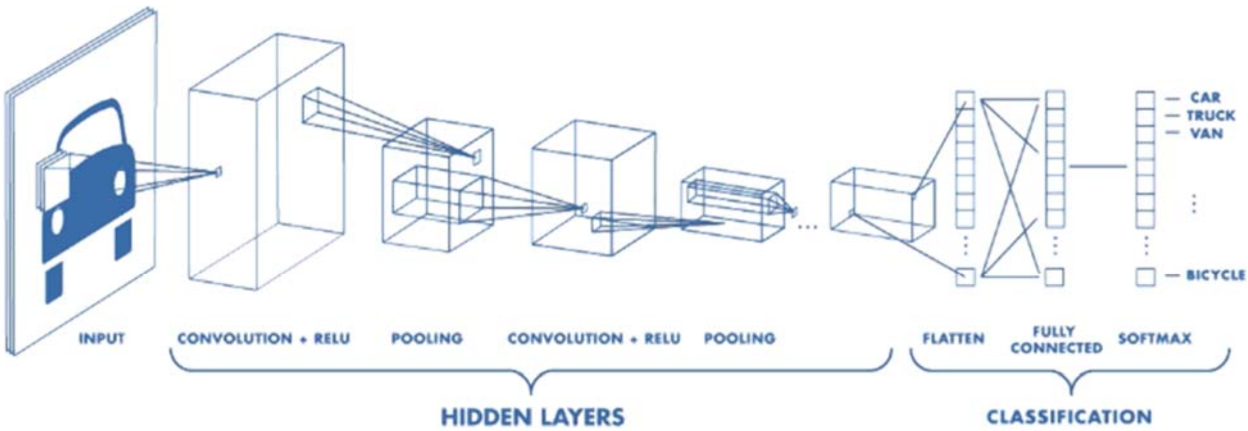
Convolution



[Wikipedia]

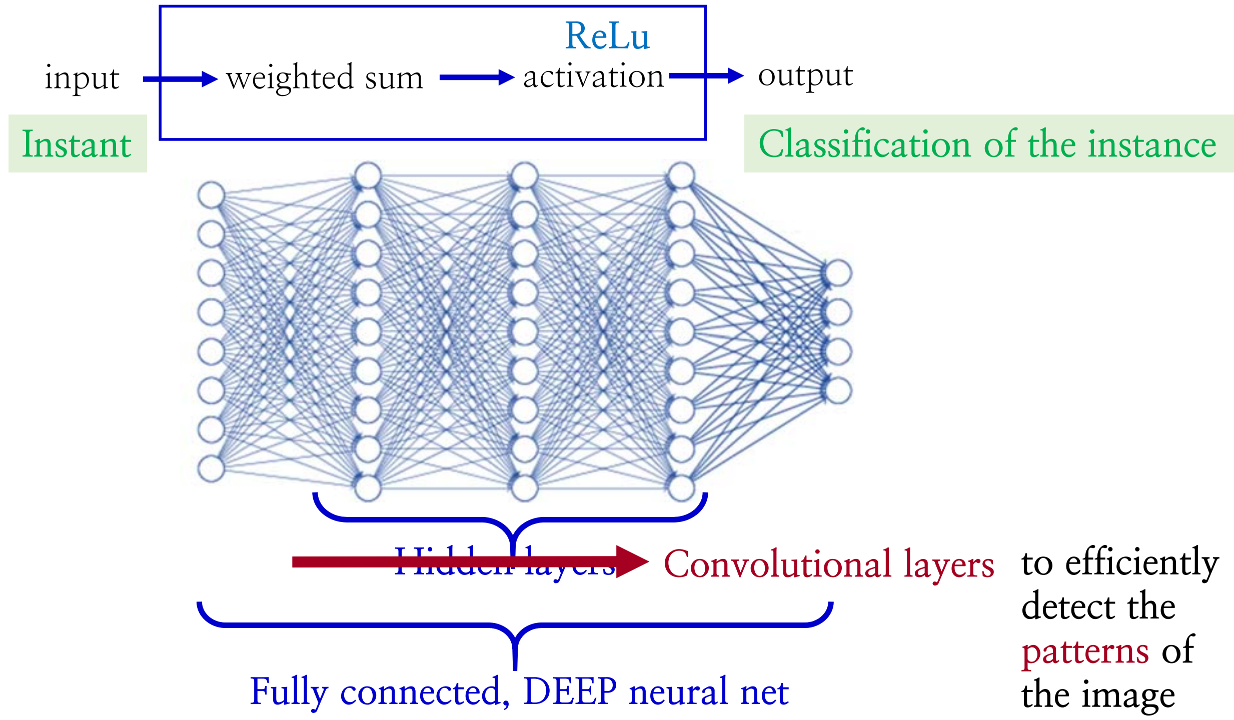
Convolutional neural network (CNN)

- 이미지 입력과 같은 경우 데이터량을 줄여 효율적으로 계산하기 위해 입력값에서 특징점(feature)을 추출하는 **필터(convolutional filter)를 사용하여 입력 데이터를 스캔하여 feature map (=activation map)을 생성**하는 신경망
- A neural network method that scans input data using a convolutional filter that extracts feature points from the input values to efficiently calculate the data amount in the case of image input.



Images from google

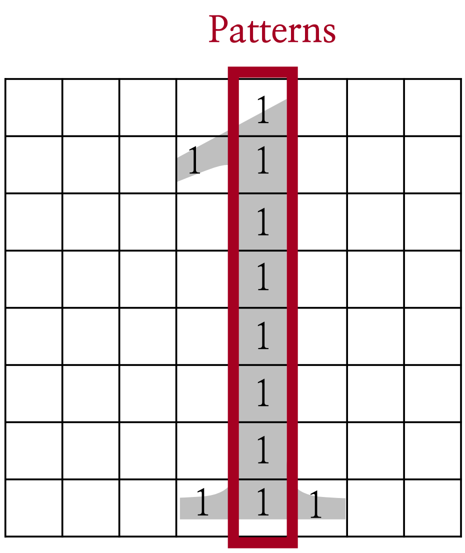
Artificial NN, i.e., fully connected NN

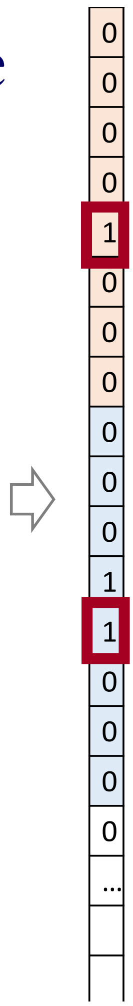


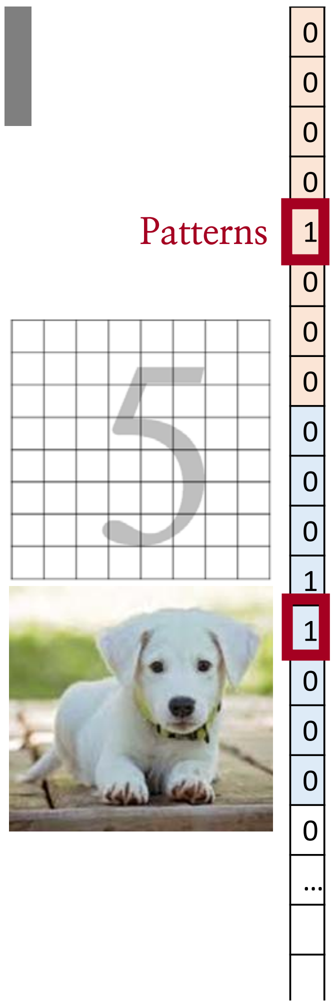
to efficiently detect the patterns of the image

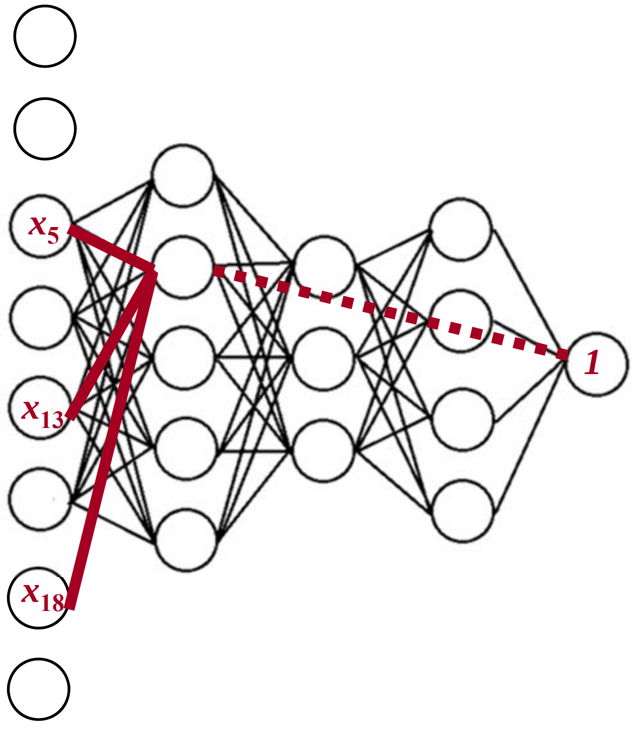
Images from google

Patterns of an image



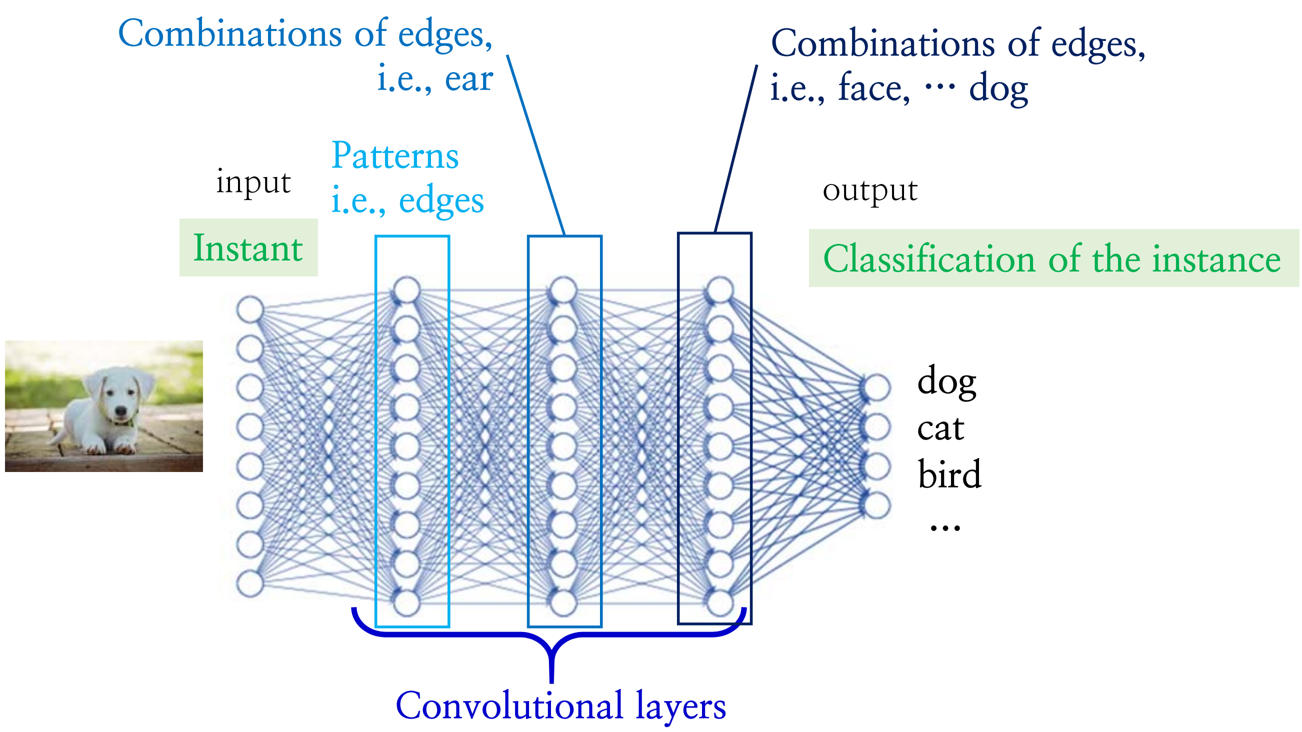




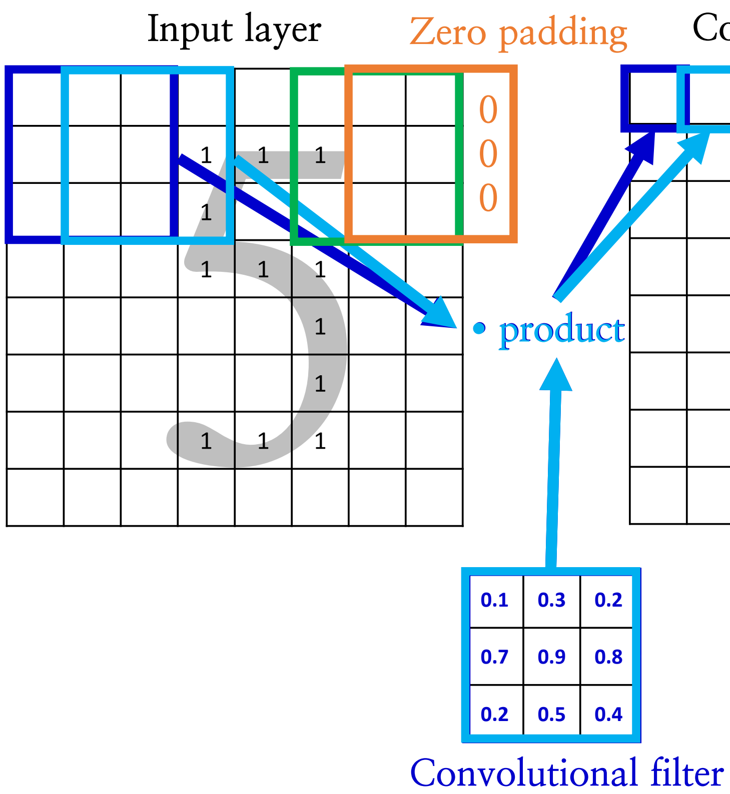


Q: How to extract the pattern of complicated input?

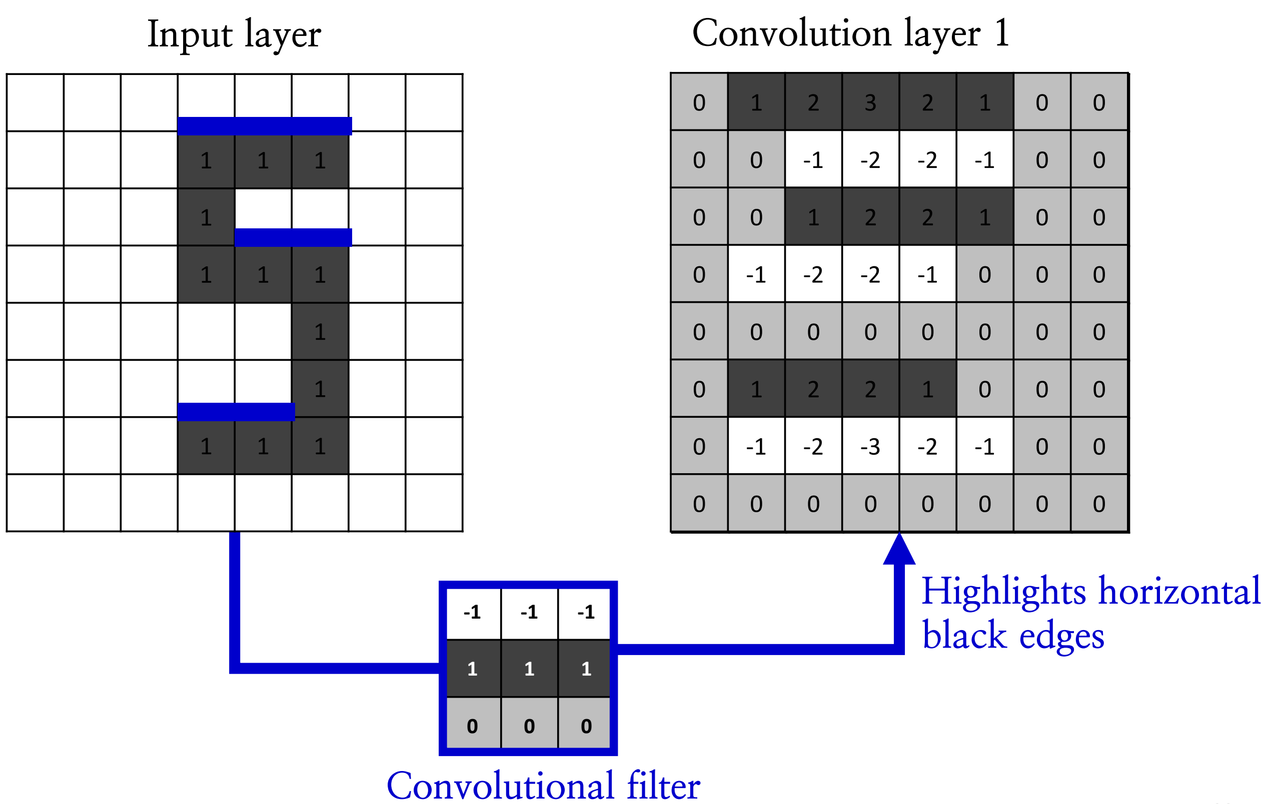
Convolutional neural network



Convolution filter



Ex) Convolution filter detecting edges



Ex) Convolution filter detecting edges

Convolution layer 1

0	1	2	3	2	1	0	0
0	0	-1	-2	-2	-1	0	0
0	0	1	2	2	1	0	0
0	-1	-2	-2	-1	0	0	0
0	0	0	0	0	0	0	0
0	1	2	2	1	0	0	0
0	-1	-2	-3	-2	-1	0	0
0	0	0	0	0	0	0	0

Convolution layer 2

-4	-9	-12	-11	-6	-2	0	0
2	6	10	10	6	2	0	0
-4	-8	-10	-8	-4	-1	0	0
3	5	5	3	1	0	0	0
3	5	5	3	1	0	0	0
-6	-11	-12	-9	-4	-1	0	0
3	6	7	6	3	1	0	0
0	0	0	0	0	0	0	0

-1	-1	-1
1	1	1
0	0	0

Convolutional filter

Highlights horizontal black edges

Ex) Convolution filter detecting edges

Input layer

			1	1	1		
			1				
			1	1	1		
					1		
					1		
			1	1	1		

Convolution layer 1

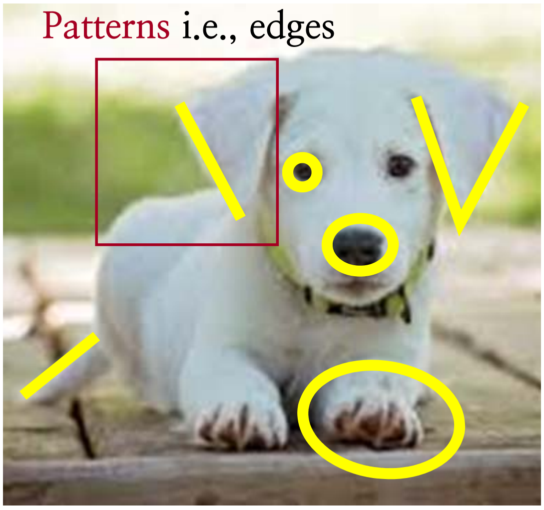
0	0	2	-1	0	-1	0	0
0	0	3	-1	0	-2	0	0
0	0	2	-1	1	-2	0	0
0	0	1	0	2	-3	0	0
0	0	1	0	2	-3	0	0
0	0	1	0	1	-2	0	0
0	0	1	0	0	-1	0	0
0	0	0	0	0	0	0	0

-1	1	0
-1	1	0
-1	1	0

Convolutional filter 2

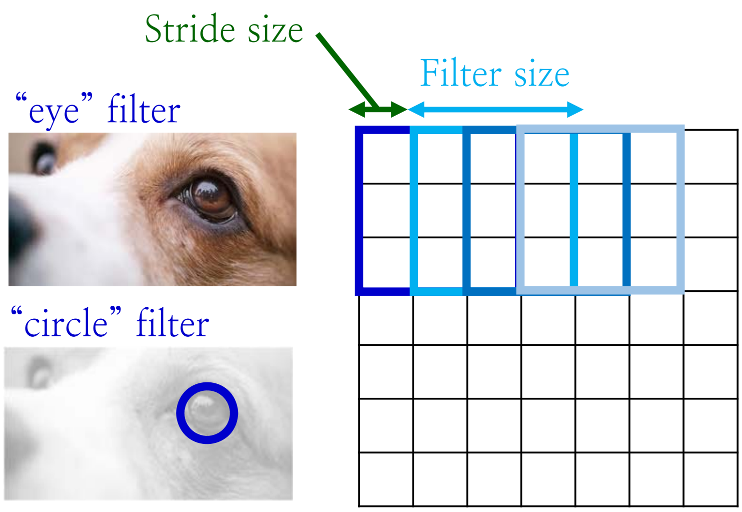
Patterns of the image

Apply multiple convolutional filters to extract feature map like edges



Convolution filter

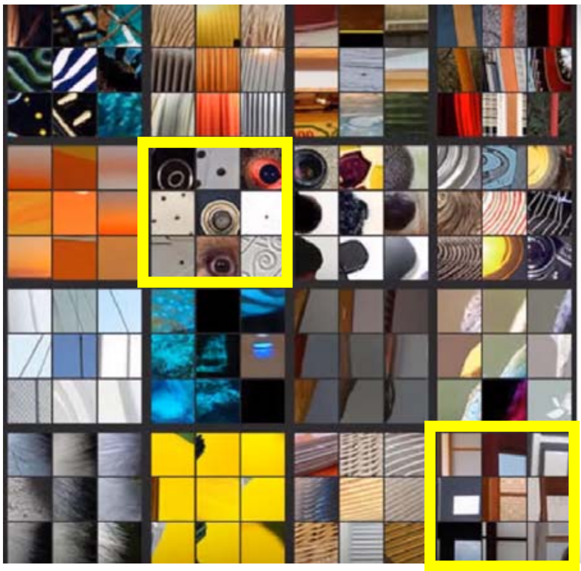
- 입력(이미지)으로 부터 여러가지 특징(예-테두리)을 추출하는 다중의 컨볼루션 필터를 적용
- Apply multiple convolutional filters to extract feature map like edges



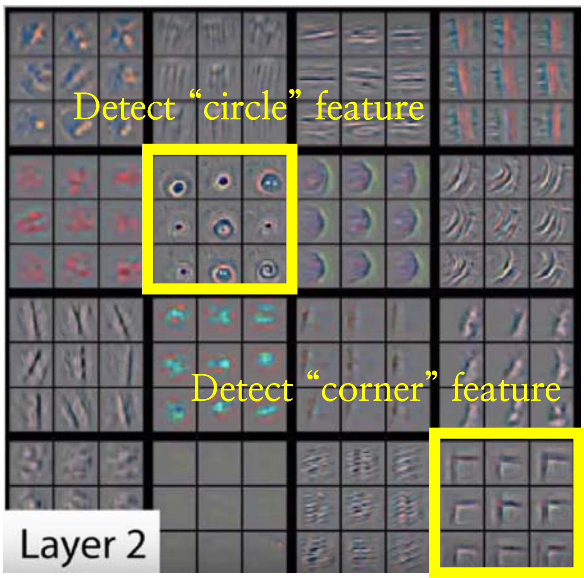
Ex) 위의 이미지를 파란 사각형 윈도우 (컨볼루션필터)로 rolling 하며 추출
Suppose you detect the image by viewing through a blue window

Ex) Convolution filter detecting features

Input layer

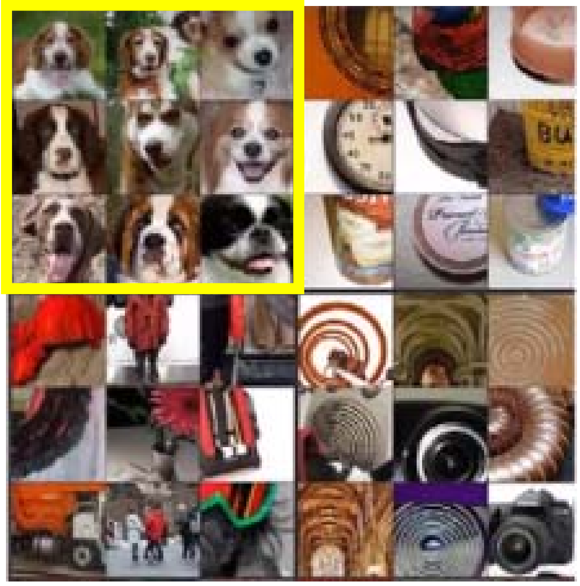


Convolution layer 2



Ex) Convolution filter detecting features

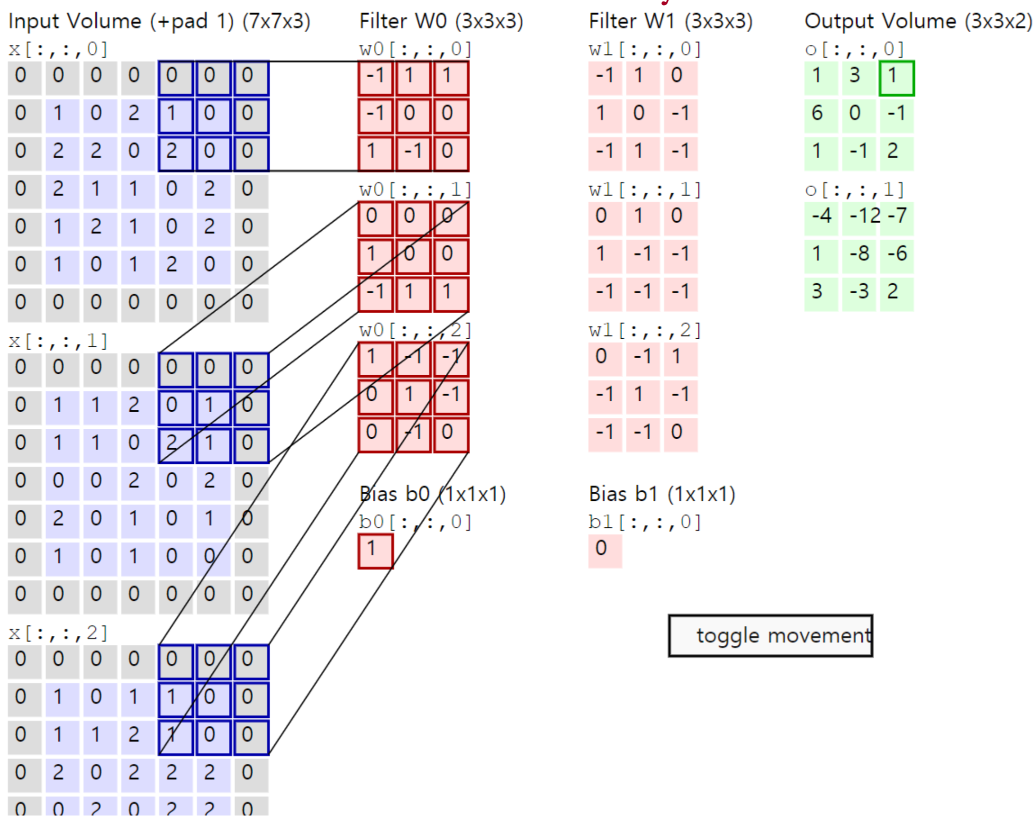
Input layer



Convolution layer 4

Ex) Convolution with layered input (RGB)

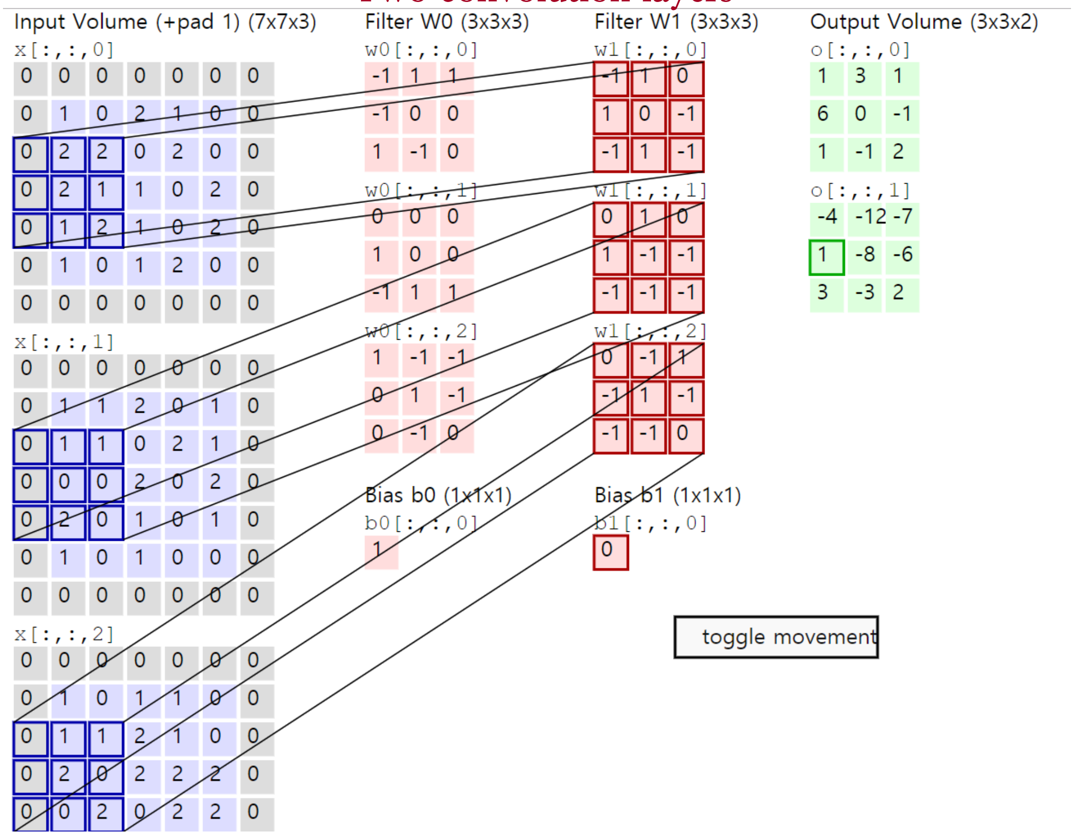
Two convolution layers



http://cs231n.github.io/convolutional-networks/

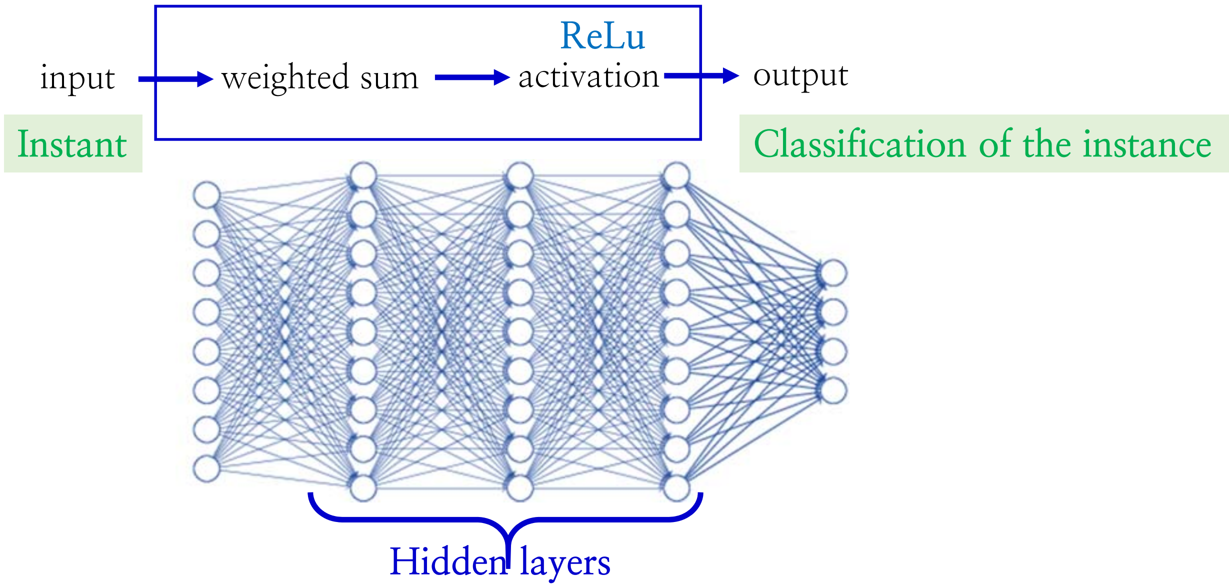
Ex) Convolution with layered input

Two convolution layers



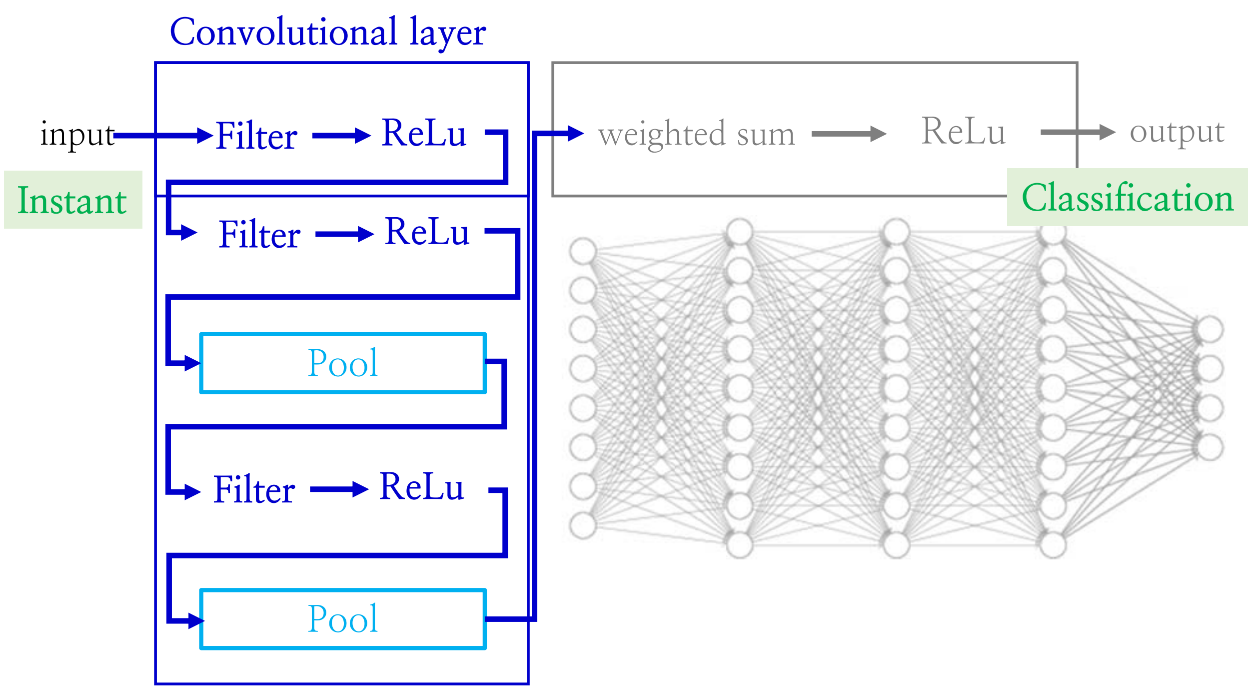
http://cs231n.github.io/convolutional-networks/

Recall, Artificial NN

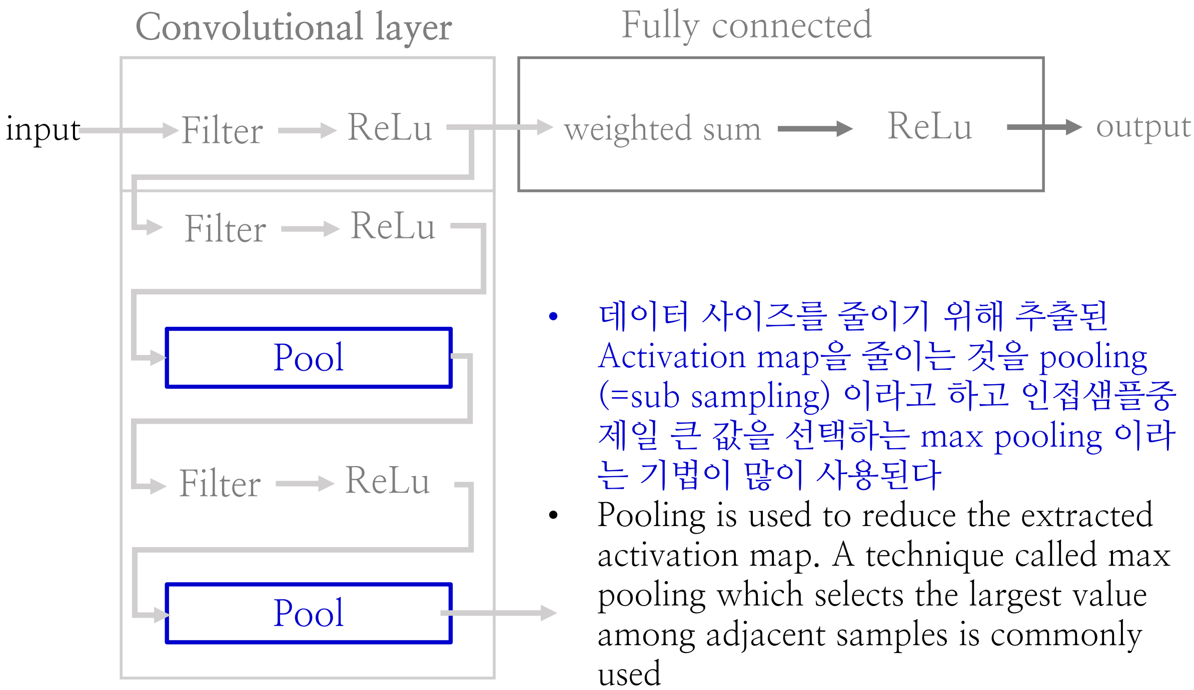


Images from google

Convolutional neural network



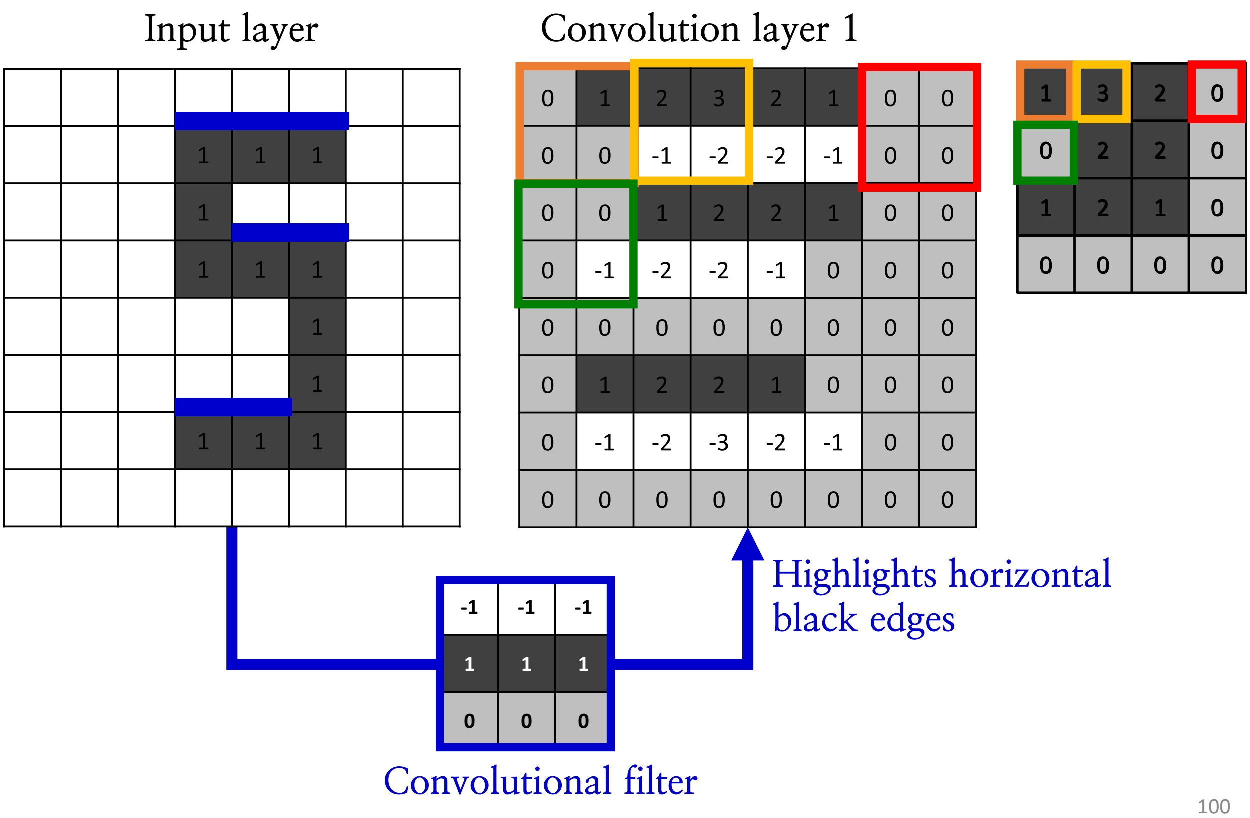
Convolutional neural network



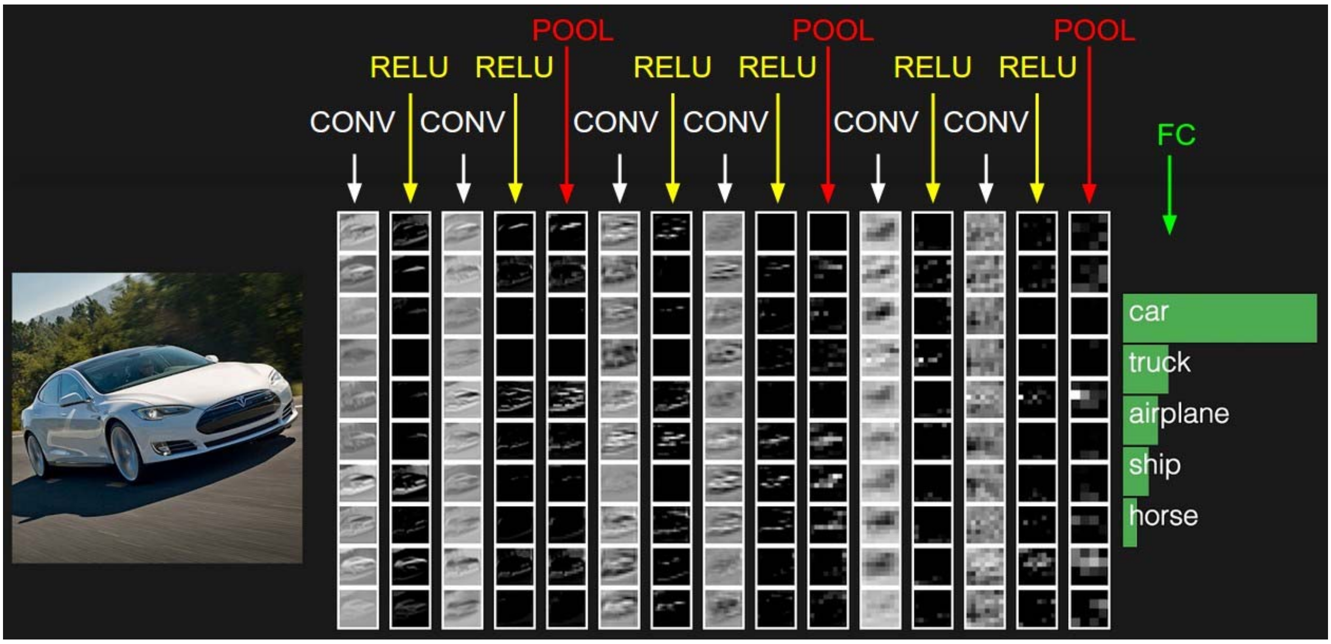
- 데이터 사이즈를 줄이기 위해 추출된 Activation map을 줄이는 것을 pooling (=sub sampling) 이라고 하고 인접샘플중 제일 큰 값을 선택하는 max pooling 이라는 기법이 많이 사용된다
- Pooling is used to reduce the extracted activation map. A technique called max pooling which selects the largest value among adjacent samples is commonly used

Max pooling example

Filter size = 2
Stride size = 2



CNN example

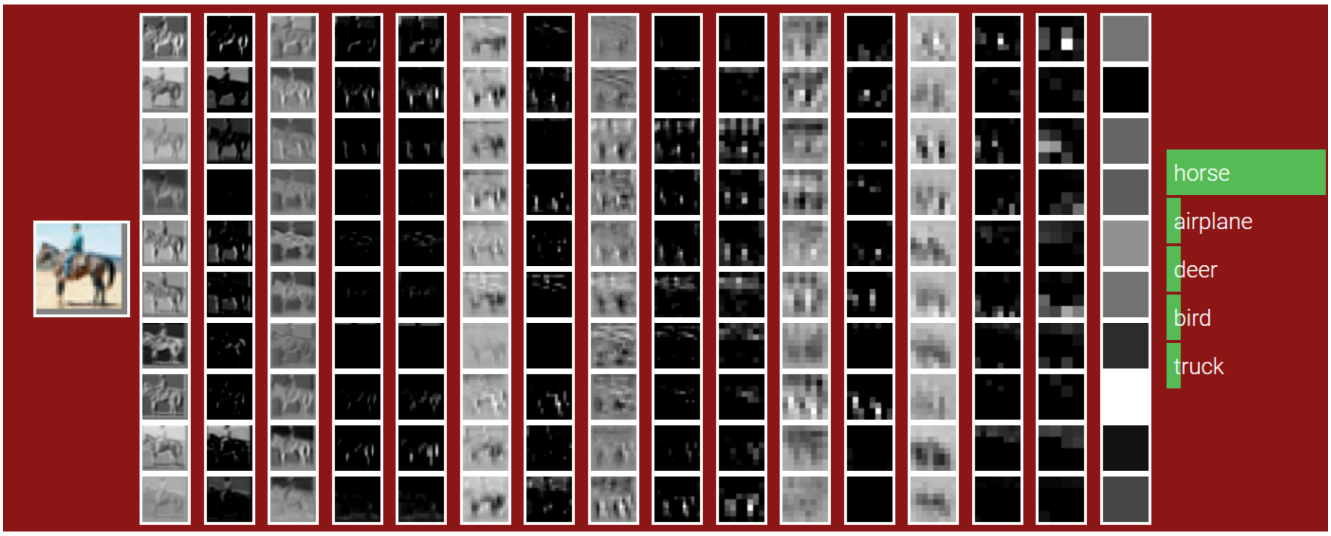


추천: https://www.slideshare.net/yongho/ss-79607172, 자습해도 모르겠던 딥러닝, 머리속에 인스톨 시켜드립니다, 하용호

http://goodtogreate.tistory.com/entry/Convolutional-Neural-Network-CNN

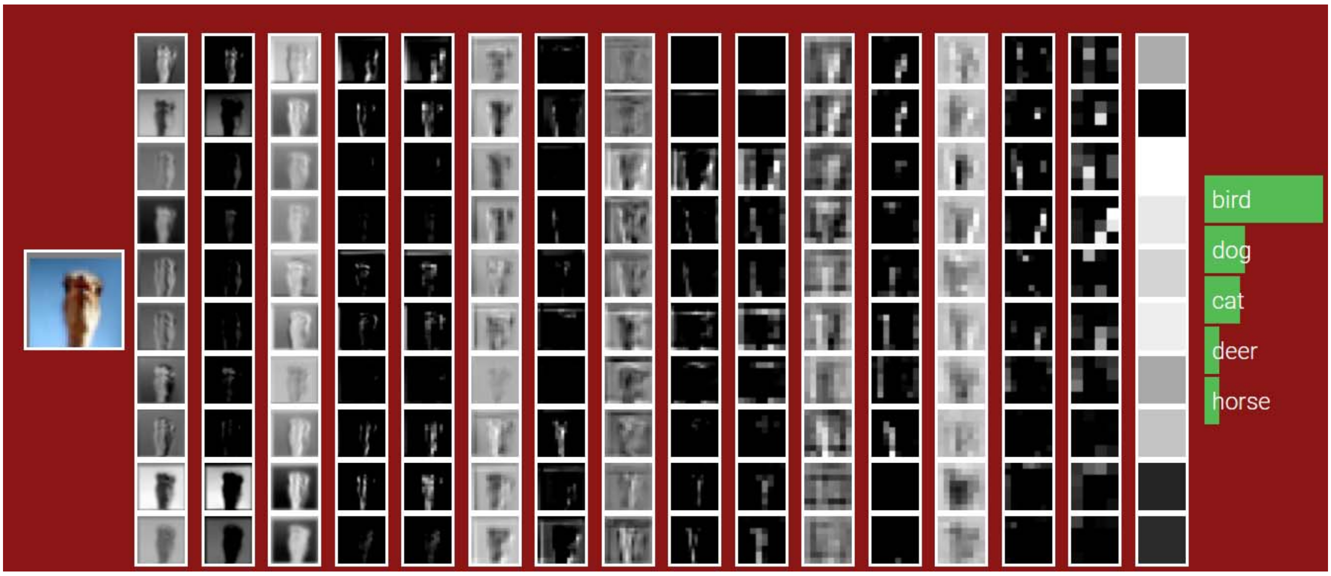
CNN example

http://cs231n.stanford.edu/



CNN example

http://cs231n.stanford.edu/



Programing CNN

[import & setting parameters]

[Visualize a few images]

[Train the model]

loss.backward()
optimzer.step()

[Visualize the model prediction]

[Fine tune the convolution net]

optim.SGD(model_ft.parameters(), lr = 0.001, momentum = 0.9)

[Train & evaluate]

2.0 SLAM basic

2.1 SLAM package in ROS

2.2 SLAM for Capstone project

Recall,

Backpropagation

Hidden layer

Convolutional Neural Network (CNN)

Overfitting

Maxpooling

Dropout

ReLU

Softmax

Feature

SGD：stochastic gradient descent

Vanishing gradient

Fully connected NN

Loss function

Recall,

RL: Reinforcement learning

AI

ML: Machine learning

Regression/clustering/classification

Deep learning

Supervised/unsupervised learning

NN: Neural net

3.0 Reinforcement learning
Deep Q-learning Network

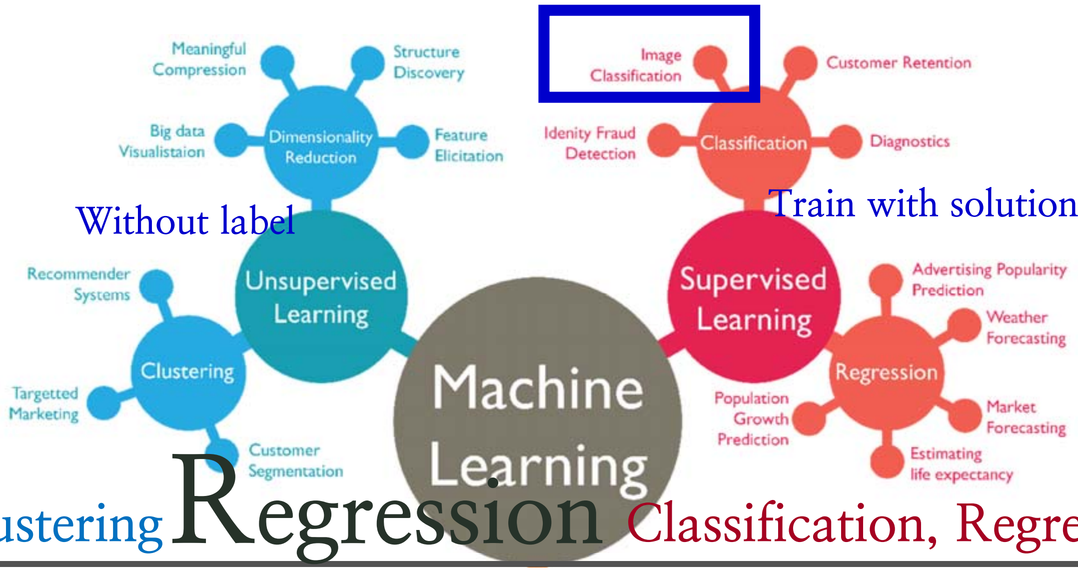
모두를 위한 RL 강좌 by Sung Kim, https://www.youtube.com/playlist?list=PLlMkM4tgfjnKsCWav-Z2F-MMFRx-2gMGG

Types of Machine learning

Capstone design 2

Without label

Train with solution(label)



Clustering Regression Classification, Regression

quential actions

Ca

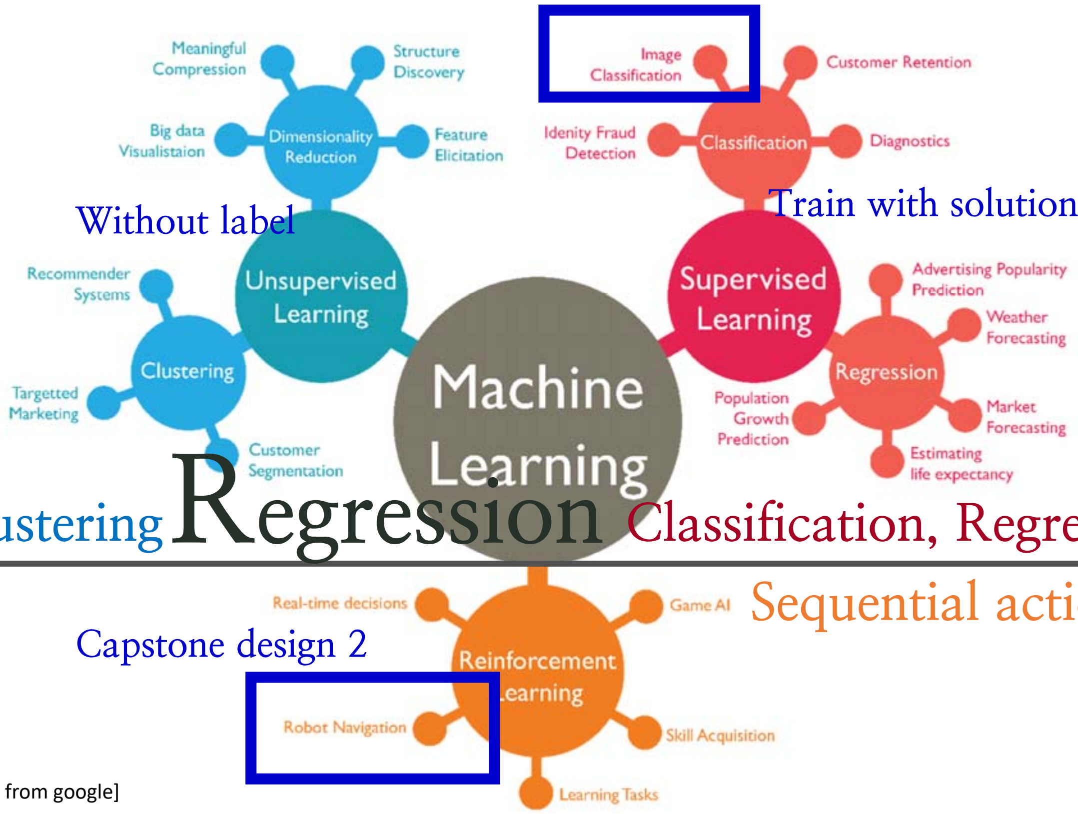
[images from google]

Types of Machine learning

Capstone design 2

Without label

Train with solution(label)



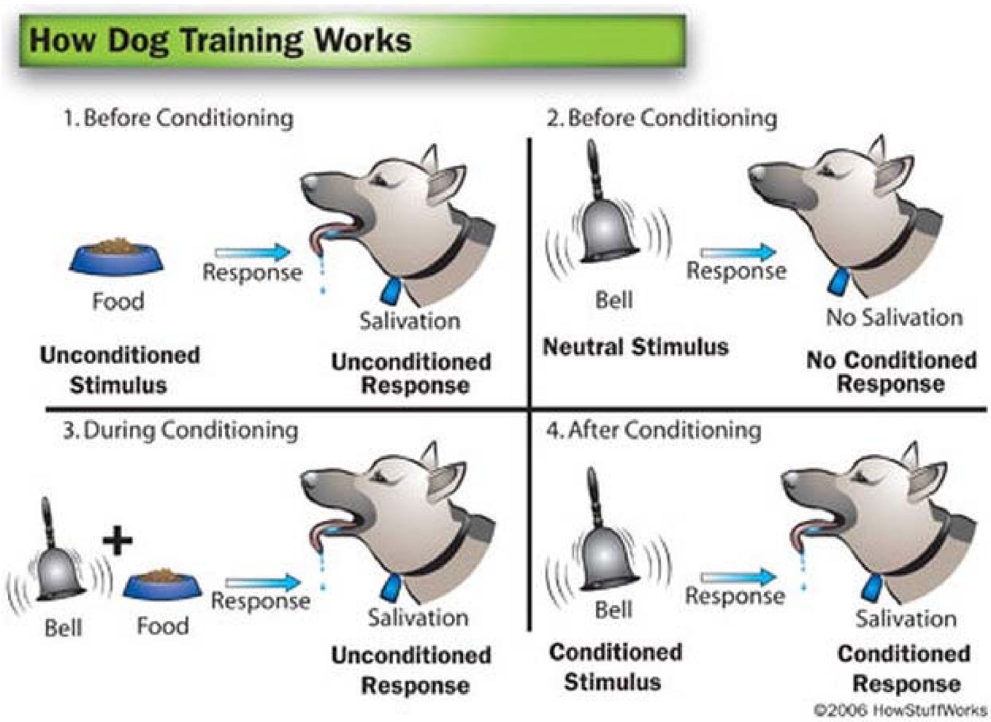
Clustering Regression Classification, Regression

Sequential actions

Capstone design 2

[images from google]

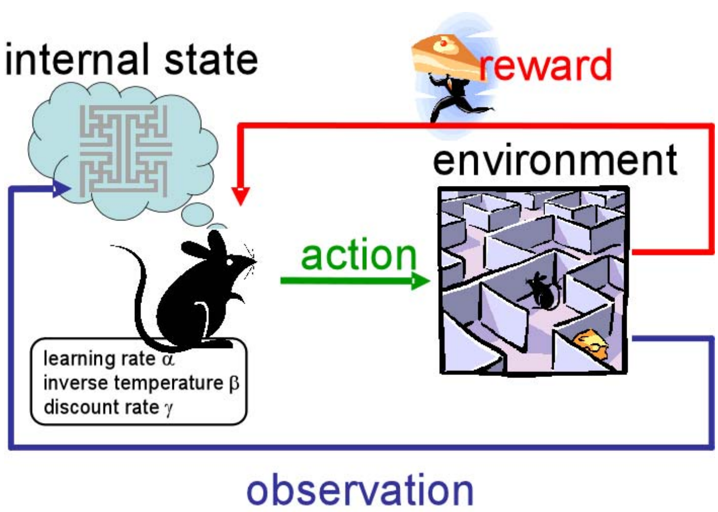
Reinforcement learning



https://www.krigolsonteaching.com/reinforcement-learning.html

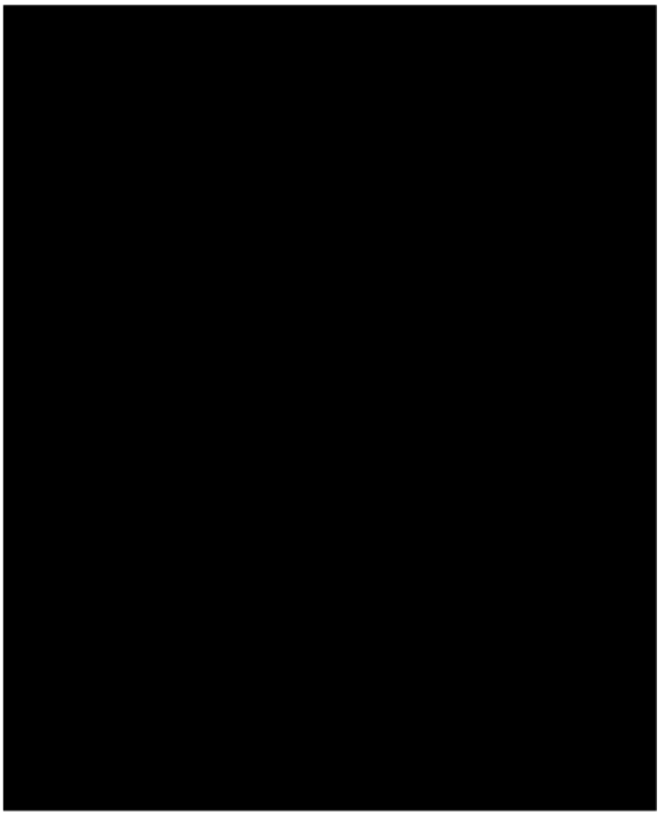
Reinforcement learning in AI

- Computational approach to learning from interaction
- Reinforcement learning (RL) is an area of machine learning concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward. [wiki]



https://becominghuman.ai/the-very-basics-of-reinforcement-learning-154f28a79071

Atari Breakout Game by Google Deepmind



https://www.youtube.com/watch?v=V1eYniJ0Rnk

Reinforcement learning in OpenAI Gym

- OpenAI is a non-profit research company that is focussed on building out AI in a way that is good for everybody. It was founded by Elon Musk and Sam Altman.
- OpenAI Gym is a toolkit for developing and comparing reinforcement learning algorithms

- *"A 2016 Nature survey indicated that more than 70 percent of researchers have tried and failed to reproduce another scientist's experiments, and more than half have failed to reproduce their own experiments."*

- OpenAI is created for removing this problem of lack of standardization in papers along with an aim to create better benchmarks by giving versatile numbers of environment with great ease of setting up.
- Aim of this tool is to increase reproducibility in the field of AI and provide tools with which everyone can learn about basics of AI.

https://towardsdatascience.com/reinforcement-learning-with-openai-d445c2c687d2

OpenAI Gym ex) cart-pole

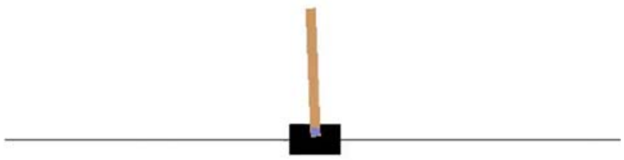
```
import gym
env = gym.make('CartPole-v0')
env.reset()
for _ in range(1000):
    env.render()
    env.step(env.action_space.sample())
# take a random action
```

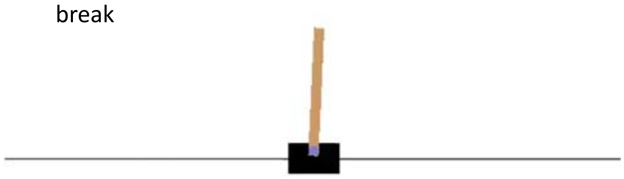
→화면출력

→ 입력(화면)에 대해 취할 액션, 창시구 2에서는 공 화면이 입력(화면)이 되고 모터로 보내는 명령이 액션임

```
import gym
env = gym.make('CartPole-v0')
for i_episode in range(20):
    observation = env.reset()
    for t in range(100):
        env.render()
        print(observation)
        action = env.action_space.sample()
        observation, reward, done, info = env.step(action)
        if done:
            print("Episode finished after {} timesteps".format(t+1))
            break
```

done: 게임 종료

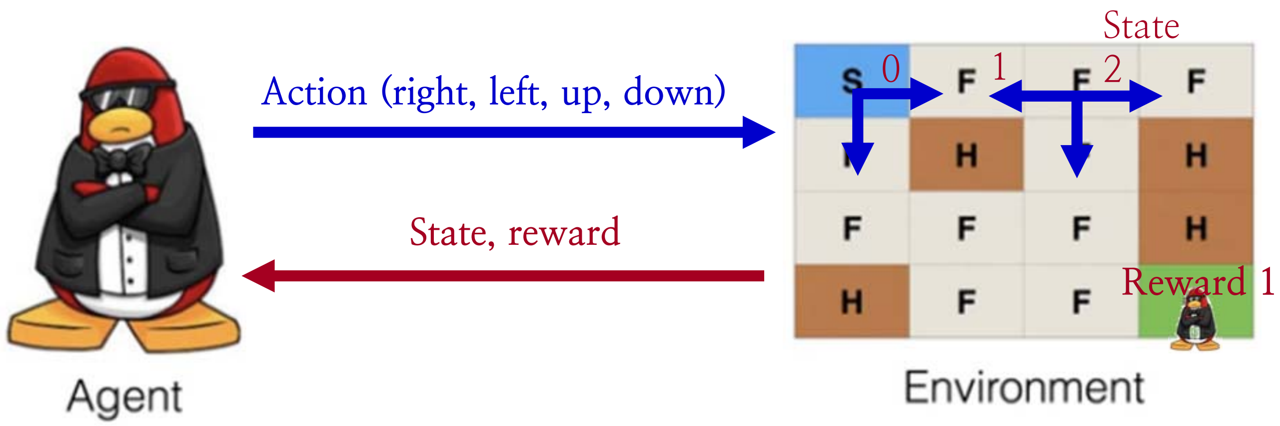




※Controlling an inverted pendulum w/ vs. without domain knowledge

https://gym.openai.com/docs/

OpenAI Gym ex) Frozen lake

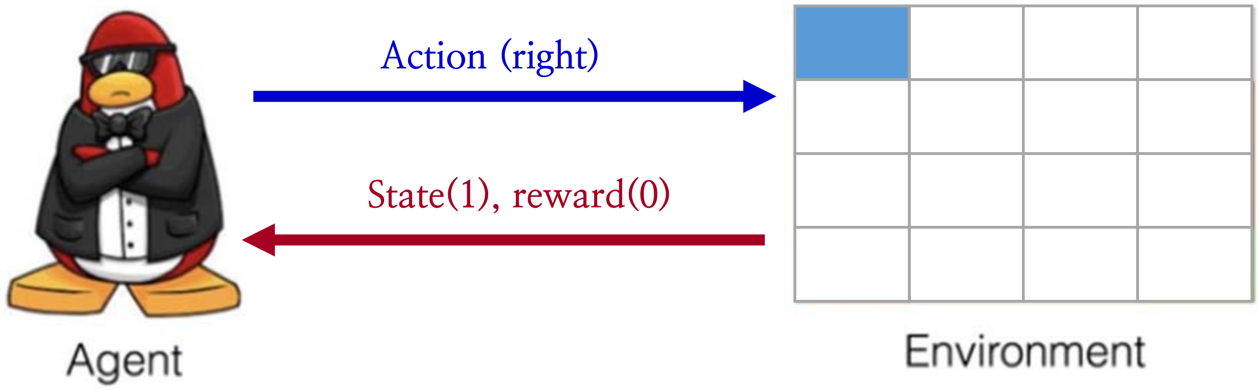


[S] safe/start
[F] frozen
[H] hole
[G] goal

https://towardsdatascience.com/reinforcement-learning-with-openai-d445c2c687d2

OpenAI Gym ex) Frozen lake

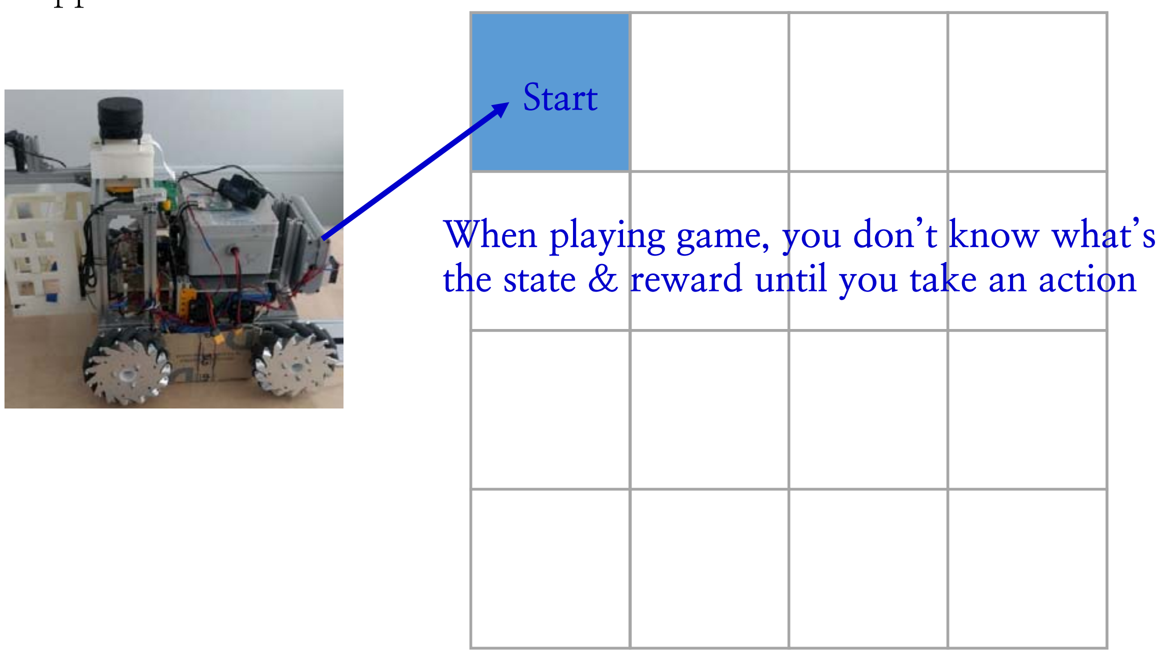
When playing game, you don't know what's the state & reward until you take an action



[S] safe/start
[F] frozen
[H] hole
[G] goal

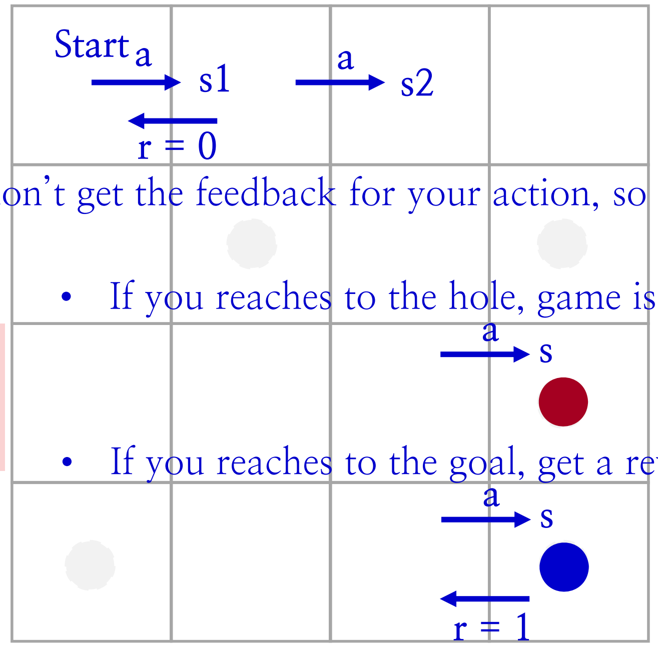
Lookup table method: Q(quality)-table

- Mission: (quickly) pick the blue ball, then get reward (+1)
- Suppose the wheels could go right, left, up, down (4 actions)
- Suppose red balls are bombs



Q-learning (look-up table)

- Initially, vehicle don't know which is where
- So just randomly take an action and then get the information about the states and reward



- Mostly, you don't get the feedback for your action, so r = 0
- If you reaches to the hole, game is over
- If you reaches to the goal, get a reward 1

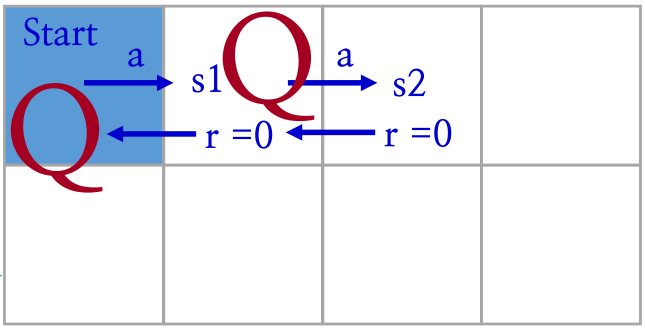
- You get the reward (not for each action but) for the "whole action set"

Q1: What if you have a something (or someone) to ask where to go?

Q function : State-action-value function

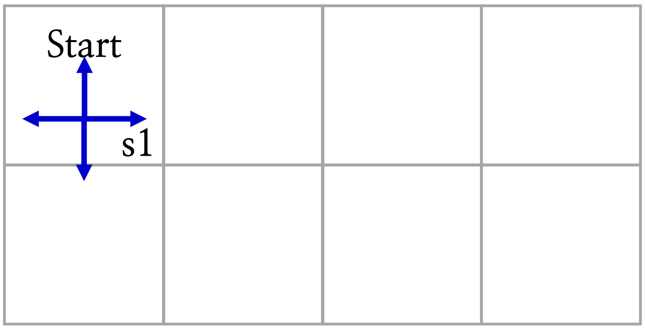
- Suppose a Q gives you the feedback about the reward "r" you would get when you take an action "a" at the state "s"

function Q(state, action) = reward



Select action from Q function

- Suppose you trust Q and take an action "a" that maximize the reward "r"



Q

Q(state, action) = reward

- Q(s1, left) = 0
- Q(s1, right) = 0.5
- Q(s1, up) = 0
- Q(s1, down) = 0.3

① Choose this to maximize the Q in response to various *a*,

$\max_{a} Q(s_1, a)$

② Then take a corresponding action, i.e. "right"

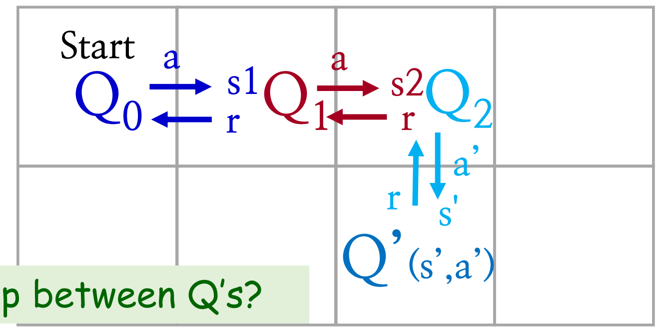
Policy

$\arg\max_{a} Q(s_1, a) \triangleq \pi^*(s)$

Optimal policy

How to find(know) Q?

- Suppose you follow the advice of Q (= optimal policy) at each state "s" and take an action "a" that maximize the reward "r"



Q: What is relationship between Q's?

Suppose you know Q'(s',a'), how to obtain Q(s,a)?

s' is the state reached by an action "a" from the previous state "s"

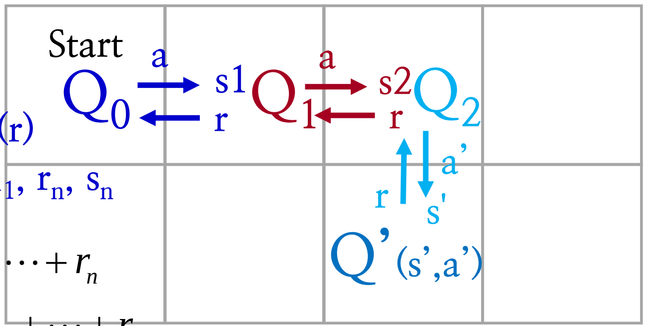
$$Q(s,a) = r + \max_{a'} Q(s', a')$$

How to find(know) Q?

−or− different explanation

State (s) → action (a) → reward (r)

$s_0, a_0, r_1, s_1, a_1, r_2, s_2, \cdots, s_{n-1}, a_{n-1}, r_n, s_n$



Total reward R $\quad R = r_1 + r_2 + r_3 + \cdots + r_n$

$$R_t = r_t + \underline{r_{t+1} + r_{t+2} + \cdots + r_n} \qquad R_{t+1} = r_{t+1} + r_{t+2} + \cdots + r_n$$

Then optimal R_t could be obtained as follows

$$R^*(t) = r_t + \max_a R(t+1)$$

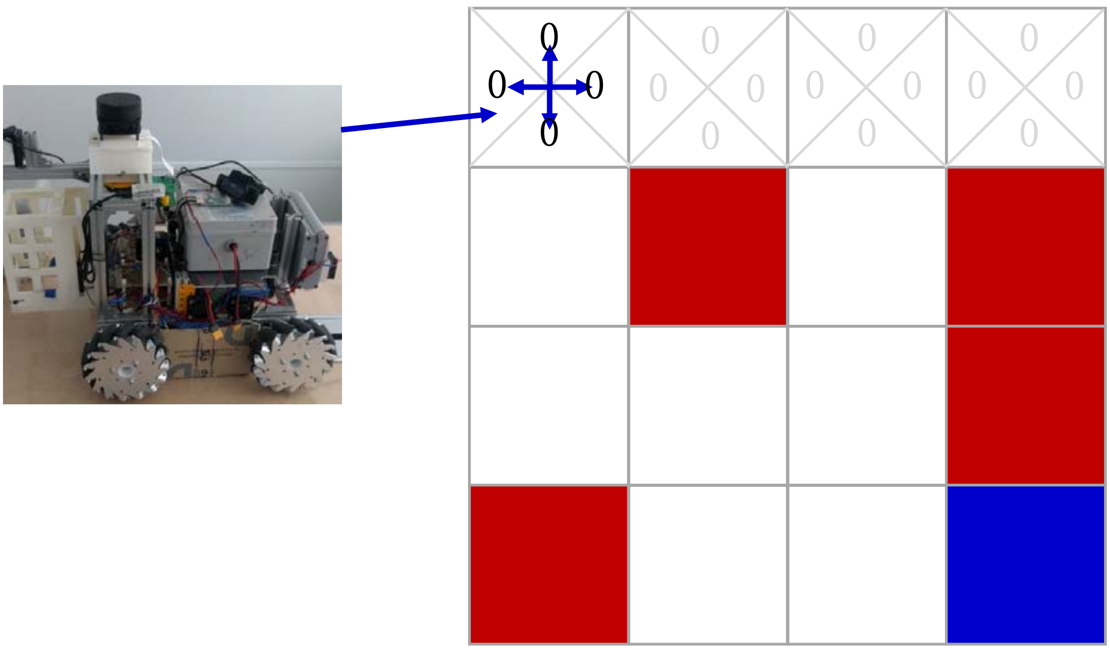
$$\hat{Q}(s,a) \leftarrow r + \max_{a'} \hat{Q}(s',a')$$

Update $\hat{Q}$ this way until you get the optimal Q*

Ex) Learning Q

- 16 states and 4 actions (up, down, left, right)

Start Initial Q values are 0

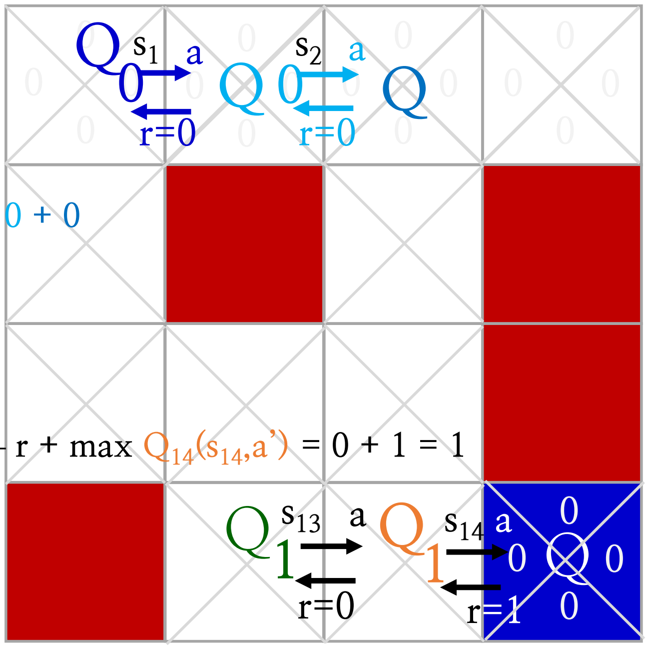


Ex) Learning Q

$$Q(s,a) = r + \max_{a'} Q(s',a')$$

$Q(s_0,a_{right}) \leftarrow r + \max Q(s_1,a') = 0 + 0$

Initial Q values are 0



$Q(s_1,a_{right}) \leftarrow r + \max Q(s_2,a') = 0 + 0$

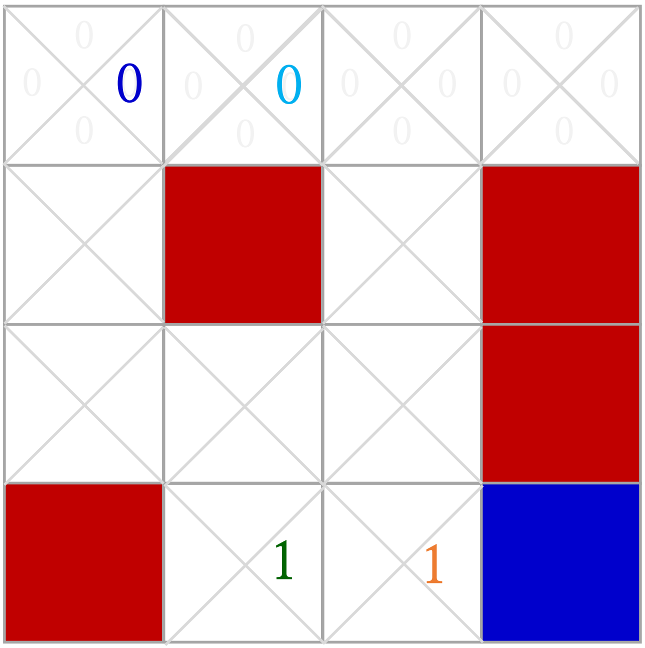
$Q_{13}(s_{13},a_{right}) \leftarrow r + \max Q_{14}(s_{14},a') = 0 + 1 = 1$

$Q_{14}(s_{14},a_{right}) \leftarrow r + \max Q_{15}(s_{15},a') = 1 + 0 = 1$

Ex) Learning Q

$$Q(s,a) = r + \max_{a'} Q(s',a')$$

Initial Q values are 0



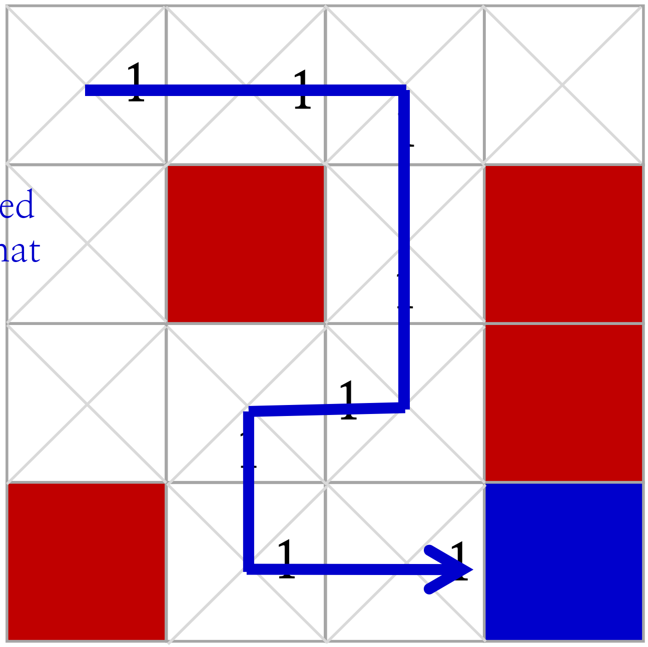
Ex) Learning Q table after one success

$$\hat{Q}(s,a) \leftarrow r + \max_{a'} \hat{Q}(s',a')$$

- Update the Q's for many visits

- Optimal "policy" π^* is obtained by taking a series of actions that maximize Qs

$$\pi^*(s) = \arg\max_a Q(s,a)$$



Programming Q−learning algorithm

- Initialize a Q table (Set the size of the Q, etc.)
- Start with a state *s*
- For loop
  - Select an action *a*
  - Execute action *a* and receive immediate reward *r*
  - Go to a new state *s'*
  - Update the Q table as follows

$$\hat{Q}(s,a) \leftarrow r + \max_{a'} \hat{Q}(s',a')$$

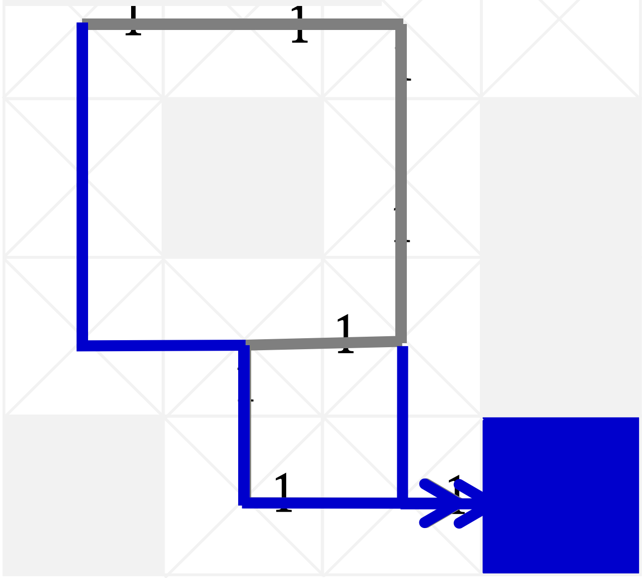
  - Replace the current state s to a new state s'

By repeating this, Q learning occurs

Exploit & exploration | ε−greedy

Q: Do you satisfy with the below "optimal" policy?

$$\pi^*(s) = \arg\max_a Q(s,a)$$



Exploit & exploration | ε−greedy

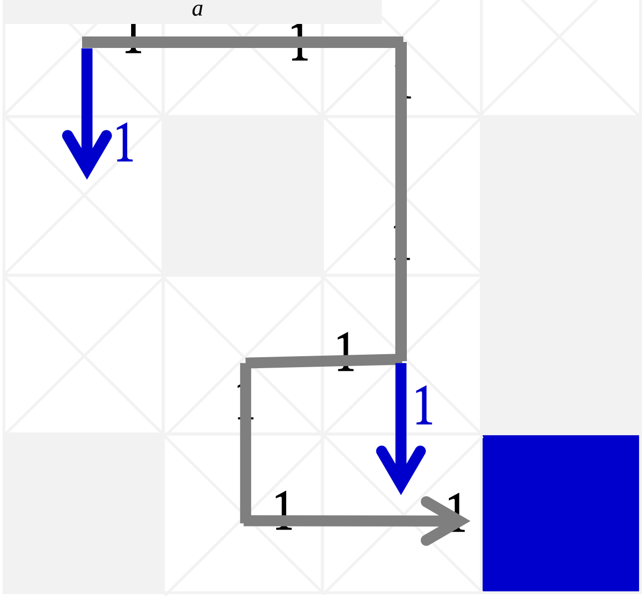
Q: Do you satisfy with the below "optimal" policy?

A: How about we try "random" action for the portion of ε % of total trials?

$$\varepsilon = 0.1$$
$$\text{if} \quad rand < \varepsilon$$
$$a = random$$
$$\text{else}$$
$$a = \arg\max Q(s,a)$$

- Also, decaying e−greedy, add random noise, etc

$$\pi^*(s) = \arg\max_a Q(s,a)$$



Discounted future reward

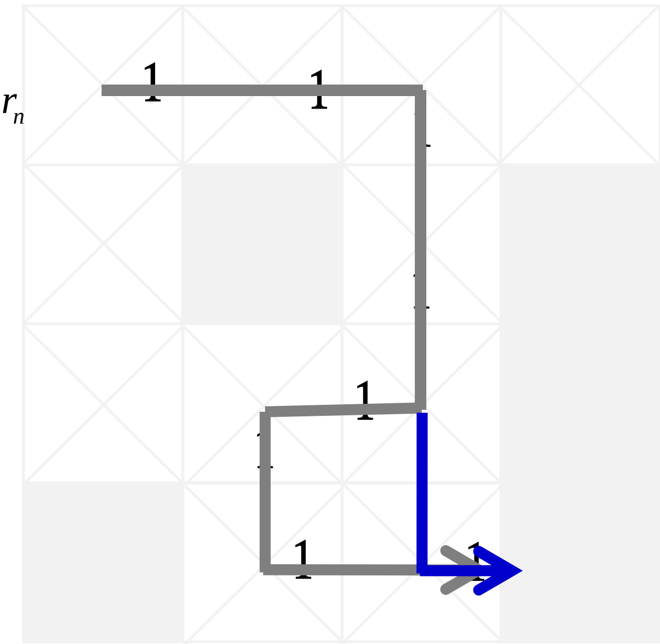
Introduce a discount rate γ btw. [0,1] s.t. $\hat{Q}(s,a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s',a')$

- −or− different approaches as a Discounted future reward

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots + \gamma^{n-t} r_n$$
$$= r_t + \gamma\left(r_{t+1} + \gamma\left(r_{t+2} + \cdots\right)\right)$$
$$= r_t + \gamma R_{t+1}$$

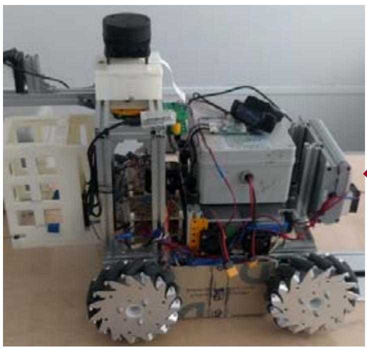
$$\Rightarrow Q(s,a) = r + \gamma \max_{a'} Q(s',a')$$

✓ Choose the action that maximizes the (discounted) future reward

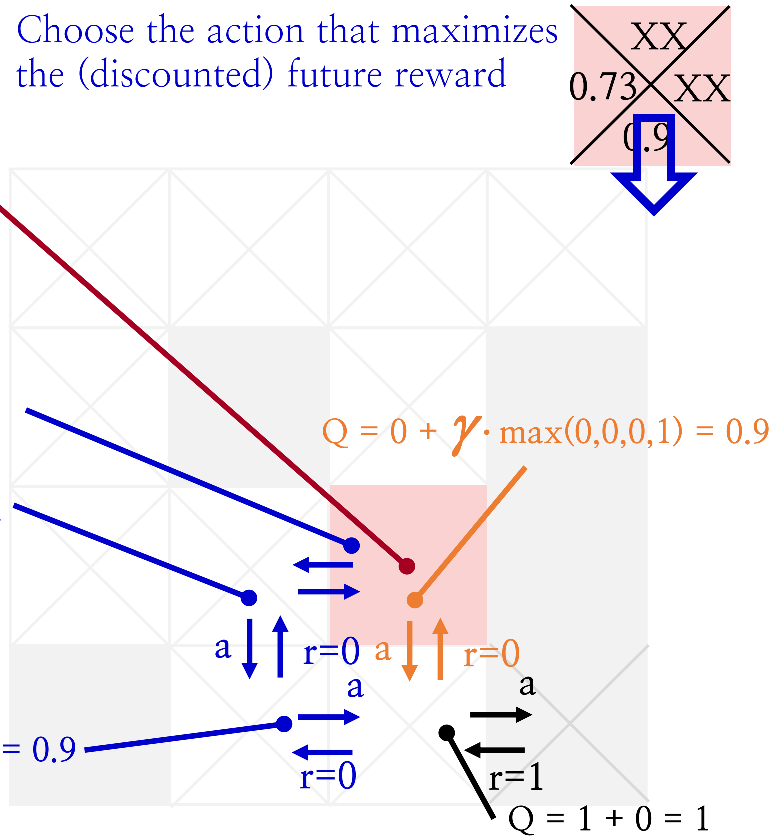


Q: Which path (policy) is better?

Ex) Discounted reward $\gamma = 0.9$



✓ Choose the action that maximizes the (discounted) future reward



Programming Q−learning algorithm

- Initialize a Q table (Set the size of the Q, etc.)
- Start with a state s
- For loop
  - Select an action a with ε−greedy
  - Execute action a and receive immediate reward r
  - Go to a new state s'
  - Update the Q table as follows

Discounted future reward

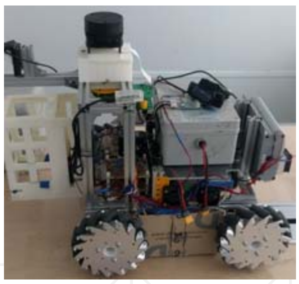
$$\hat{Q}(s,a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s',a')$$

※ Q^ converges to Q

  - Replace the current state s to a new state s'

Q: What if the response to action is different from what you expected?

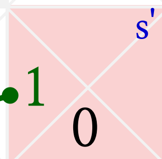
Slippery or rough surface



'Right' action may result in 'Down' or stay or others not intended to do so

⇒ Stochastic (= Non−deterministic) world

Q: Would you trust Q(left)?



This Q (s, left) = 1 may be obtained from trial of Q (s, down (but slip to left))

⇒ Trust Q(s') just a little bit and update Q(s) a little bit (learning rate)

Learning with learning rate α

- Recall, update Q(s) from next state s' information

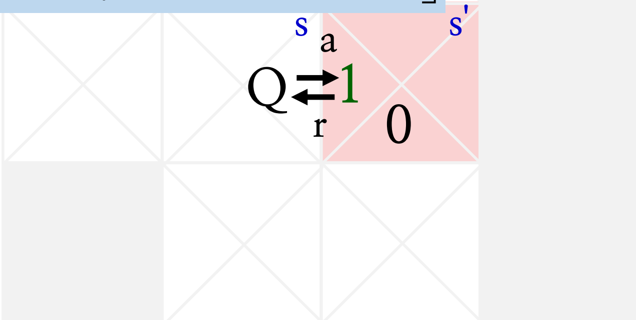
$$Q(s,a) \leftarrow r + \gamma \max_{a'} Q(s',a')$$

- If you partially trust states Q(s'), and stick to the current Q(s), then

$$\hat{Q}(s,a) \leftarrow (1-\alpha)\hat{Q}(s,a) + \alpha\left[r + \gamma \max_{a'} \hat{Q}(s',a')\right]$$

α : Learning rate

$$\hat{Q}(s,a) \quad \leftarrow \quad \hat{Q}(s,a) + \alpha \cdot \left[r + \gamma \max_{a'} \hat{Q}(s',a') - \hat{Q}(s,a)\right]$$



Programming Q−learning algorithm

- Initialize a Q table (Set the size of the Q, etc.)
- Start with a state *s*
- For loop
  - Select an action *a* with ε−greedy
  - Execute action *a* and receive immediate reward *r*
  - Go to a new state *s'*
  - Update the Q table as follows

Learning rate

$$\hat{Q}(s,a) \leftarrow (1-\alpha)\hat{Q}(s,a) + \alpha\left[r + \gamma \max_{a'} \hat{Q}(s',a')\right]$$

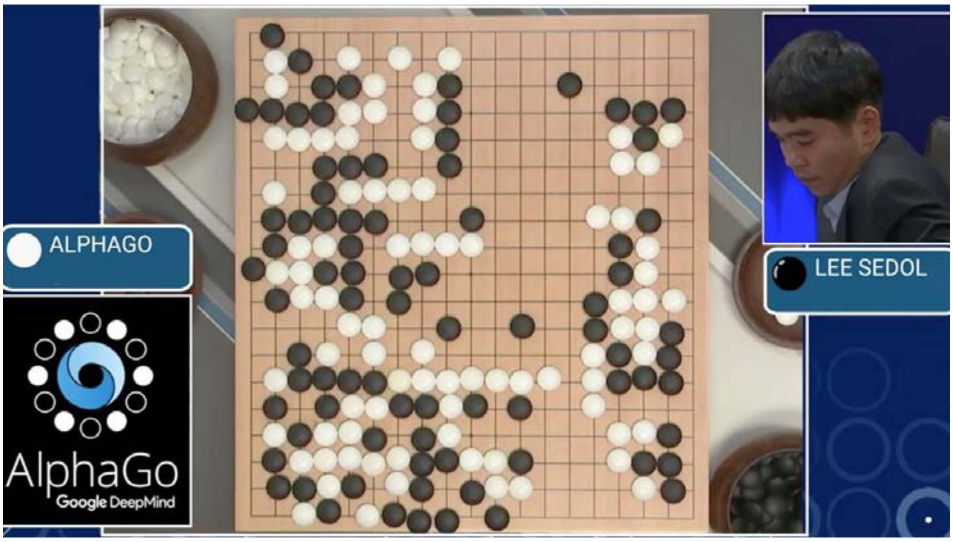
  - Replace the current state s to a new state s'

※ Q^ converges to Q

Lab: http://computingkoreanlab.com/app/jAI/jQLearning/

Q: What if number of states are VERY big?

Q−Table for AlphaGo

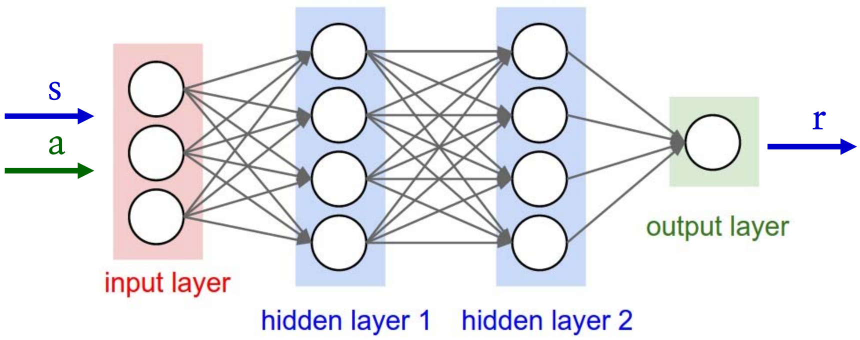


$\sim 10^{171}$ =208,1681,9938,1979,9846,9947,8633,3448,6277,0286,5224,5388,4530,5484,2563,9456,8209,2741,9612,7380,1537,8525,6484,5169,8519,6439,0725,9916,0156,2812,8546,0898,8831,4427,1297,1531,9317,5577,3662,0397,2470,6484,0935 $\Rightarrow$ Try Network!

Images from google

Q function approximation using Network

- Network of inputs of state ‘s’ & action ‘a’ and output as reward ‘r’

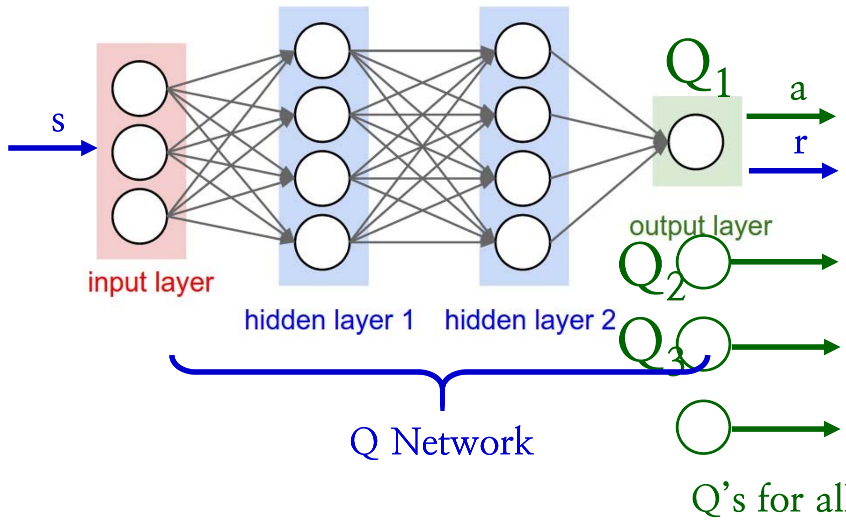


Reward for the given action, e.g., r_{LEFT}: 0.5

http://cs231n.github.io/neural-networks-1/

Q function approximation using Network

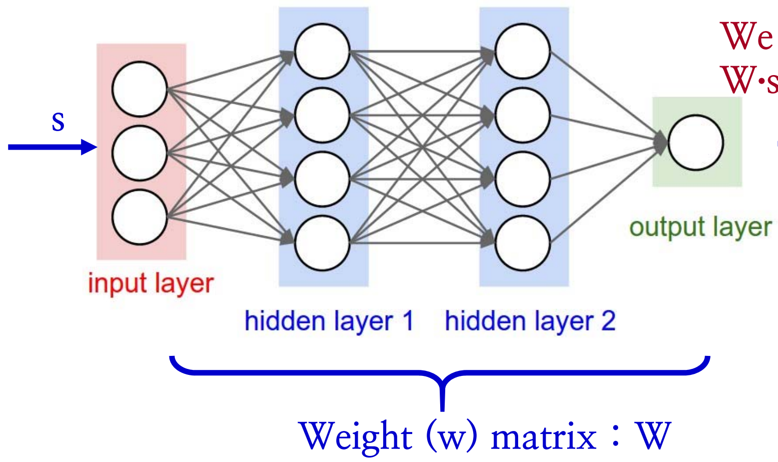
- Network of inputs of state ‘s’ and output as reward ‘r’ @ action ‘a’



Reward (Q) for all actions, e.g., [0.5, 0.1, 0, 0.8]

LEFT: 0.5
Right: 0.1
UP: 0
DOWN: 0.8

Q–Network training



We want output of Q–network W·s to match with Q^*(=*y*, label)

W·s ~ Q^*

⇒ Equivalent to linear regression problem

Formulate linear regression of finding “W” to match Wx ~ Q*(=y, label)

$$\min loss(W) \sim \min \text{cost}(W) \sim \min f\,(Wx - y)^2$$

Q−Network training

Formulate linear regression of finding "W" to match Wx ~ Q*(=y, label)

$$\min loss(W) \sim \min cost(W) \sim \min f\,(Wx - y)^2$$

$$\hat{Q}(s,a|\theta) = \quad \hat{Q} \equiv Ws \quad y \equiv Q^*(s,a) = r + \gamma \max_{a'} Q(s')$$

Q prediction (^) at a given state 's' and action 'a', as a function of weight (θ) of the network

- Through the network, find weight (θ) to minimize the cost, i.e., loss

$$\min_{\theta} \sum_{j=0}^{N} \left[\hat{Q}(s_j, a_j) - Q^*(s_j, a_j) \right]^2 \quad \min_{\theta} \sum_{j=0}^{N} \left[\hat{Q}(s_j, a_j|\theta) - \left(r_j + \gamma \max_{a'} Q(s_{j+1}, a'|\theta) \right) \right]^2$$

Function of weight (θ), from Network Ws= Θs

Approximation from next state's $Q_{s'}$ function and optimization (note max Q')

Also function of "θ"

Programming Q−learning algorithm

- Initialize a Q table (Set the size of the Q, etc.)
- Start with a state *s*
- For loop
  - Select an action *a* with ε−greedy
  - Execute action *a* and receive immediate reward *r*
  - Go to a new state *s'*
  - Update the Q table as follows

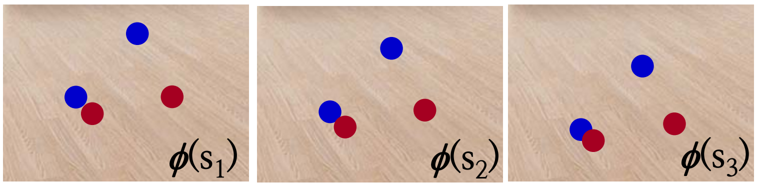
$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a')\right]$$

  - Replace the current state s to a new state s'

Programming Q−learning algorithm

- Initialize a ~~Q table (Set the size of the Q, etc.)~~ Q function with random weight θ_0
- Start with a state s Initialize $s_1 = \{x_1\}$
  −or− preprocessed(mostly images) sequenced $\phi_1 = \phi(s_1)$

Ex) x1 = state to motor command "forward", then $\phi_1 = \phi(s_1)$ is the changed image with respect to forward movement



Recall, e−greedy

- Initialize a Q function with random weight θ_0
- Initialize $s_1 = \{x_1\}$ −or− preprocessed sequenced $\phi_1 = \phi(s_1)$
- For loop
  - Select an action a with ε−greedy

$$a_t = \begin{cases} random & for\ rand < \varepsilon \\ \arg\max_a Q(s,a) & otherwise \end{cases}$$

Programming Q−learning algorithm

- Initialize a Q function with random weight θ_0
- Initialize $s_1 = \{x_1\}$ −or− preprocessed sequenced $\phi_1 = \phi(s_1)$
- For loop
  - Select an action a with ε−greedy

Execute action 'a' in EMULATOR and observe the image x_{t+1} and reward

  - ~~Execute action *a* and receive immediate reward *r*~~
  - Go to a new state *s'* set $s_{t+1} = \{x_{t+1}\}$, or $\phi_{t+1} = \phi(s_{t+1})$(= here s_{t+1})
  - Update the ~~Q table as follows~~ Q(=Ws=θs) to match the target y using gradient descent

$$Q_j^* \triangleq y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a' | \theta) & \text{otherwise} \end{cases}$$

$$\text{Loss function} = \min \Big(\underbrace{Q(\phi_j, a_j | \theta)}_{Ws} - \underbrace{y_j}_{\text{Target, label}} \Big)^2 \text{ by gradient descent}$$

Programming Q−learning algorithm

- Initialize a Q function with random weight θ_0

For loop for episodes

- Initialize $s_1 = \{x_1\}$ −or− preprocessed sequenced $\phi_1 = \phi(s_1)$
- For loop
  - Select an action a with ε−greedy
  - Execute action a in emulator and observe the image x_{t+1} and reward
  - Go to a new set $s_{t+1} = \{x_{t+1}\}$, or $\phi_{t+1} = \phi(s_{t+1})$(= here s_{t+1})
  - Update the Q(=Ws=θs) to match the target y using gradient descent

$$y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a' | \theta) & \text{otherwise} \end{cases}$$

$$\textit{then} \quad \min \Big(Q(\phi_j, a_j | \theta) - y_j \Big)^2$$

Key algorithm of Deep Learning used for Atari by Google DeepMind

Convergence

- Q^ converges to Q^* using table lookup representation BUT
- Q^ diverges using neural networks due to

① Correlations between samples
② Non-stationary targets

Q: How DeepMind resolve this issue?

Google DeepMind

✓ Go deep
✓ Capture and Replay (sol'n for ①)
✓ Separated networks: create a target network (sol'n for ②)

Images from google

DQN

DQN paper
www.nature.com/articles/nature14236

DQN source code
sites.google.com/a/deepmind.com/dqn/

Two issues to overcome

- Q^ diverges using neural networks due to

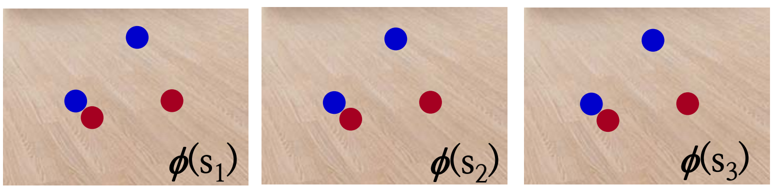
① Correlations between samples
② Non-stationary targets

1. Correlation between samples

- Initialize a Q function with random weight θ_0

For loop for episodes

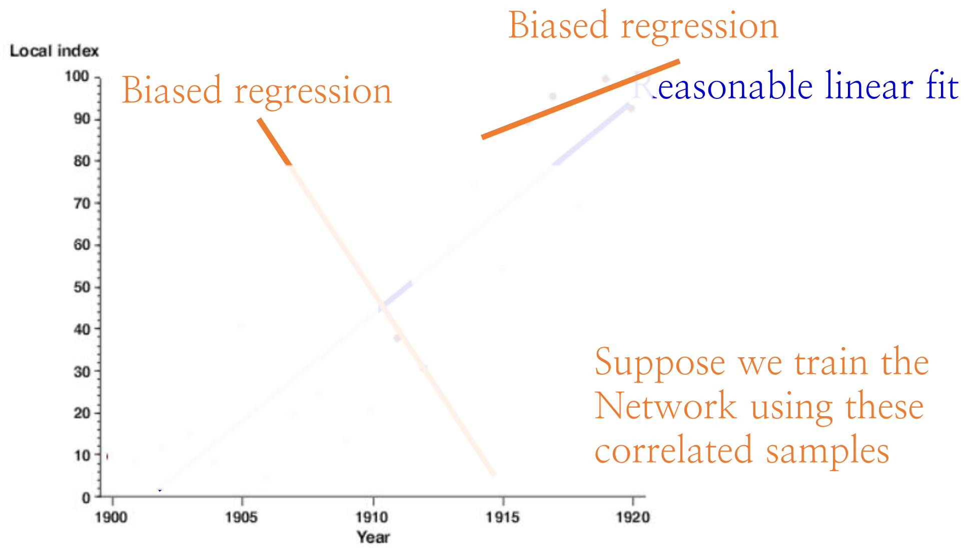
- Initialize $s_1 = \{x_1\}$ –or– preprocessed sequenced $\phi_1 = \phi(s_1)$
- For loop
  - Select an action *a* with ε–greedy, random or $a_j = \max_a Q^*(\phi_j, a|\theta)$
  - Execute action a in emulator and observe the image x_{t+1} and reward
  - Go to a new set $s_{t+1} = \{x_{t+1}\}$, or $\phi_{t+1} = \phi(s_{t+1})$(= here s_{t+1})
  -



Very similar samples and correlated with each other

1. Correlation between samples

✓ If samples have high correlation, the regression results could be biased

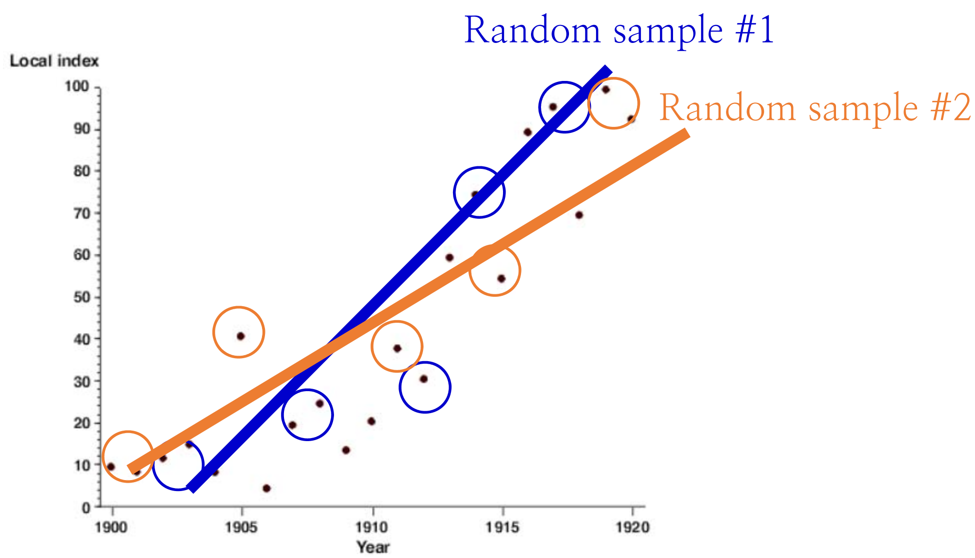


Suppose we train the Network using these correlated samples

※ Solution: **Experience replay**, i.e., random sample & replay

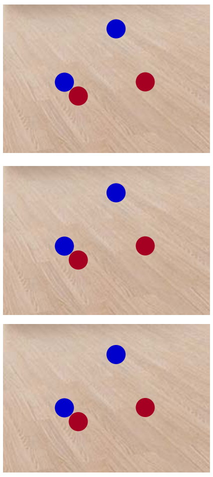
1. Experience replay

- Capture few samples (to reduce the intra-sample correlation), and perform a regression, and replay the "captured sample"



1. Experience replay

- Capture few samples (to reduce the intra-sample correlation), and perform a regression, and replay the "captured sample"



Capture many samples →

Stored in a buffer

s_1, a_1, r_2, s_2
s_2, a_2, r_3, s_3
s_3, a_3, r_4, s_4
…
$s_t, a_t, r_{t+1}, s_{t+1}$

→ Random sample & replay for training

$$\min_{\theta} \sum_{j=0}^{N} \left[\hat{Q}\left(s_j, a_j \middle| \theta\right) - \left(r_j + \gamma \max_{a'} Q\left(s_{j+1}, a' \middle| \theta\right)\right) \right]^2$$

Programming: experience replay

- Initialize a Q function with random weight θ_0

For loop for episodes

- Initialize $s_1 = \{x_1\}$ –or– preprocessed sequenced $\phi_1 = \phi(s_1)$
- For loop
  - Select an action a with ε–greedy, random or $a_j = \max_a Q^*(\phi_j, a|\theta)$
  - Execute action a in emulator and observe the image x_{t+1} and reward
  - Go to a new set $s_{t+1} = \{x_{t+1}\}$, or $\phi_{t+1} = \phi(s_{t+1})$(= here s_{t+1})
  - Store transition (ϕ_t, a_t, r_t, ϕ_{t+1}) in D (buffer)
  - Sample random minibatch of transitions (ϕ_j, a_j, r_j, ϕ_{j+1}) from D
  - Update the Q(=Ws=θs) to match the target y using gradient descent

$$y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'|\theta) & \text{otherwise} \end{cases}$$

$$\textit{then} \quad \min\left(Q(\phi_j, a_j) - y_j\right)^2$$

2. Non-stationary target

Moving

Note, both learning parameter θ(=W), which is updated through the network, is also affect the target

$$\min_{\theta}\sum_{j=0}^{N}\Big[\underbrace{\hat{Q}\big(s_j,a_j\,\boxed{\theta}\big)}_{\text{Q prediction = Ws}}-\underbrace{\big(r_j+\gamma\max_{a'}Q\big(s_{j+1},a'\,\boxed{\theta}\big)\big)}_{\text{Target(y) to match}}\Big]^2$$

※ Solution: Separate networks by creating a target network

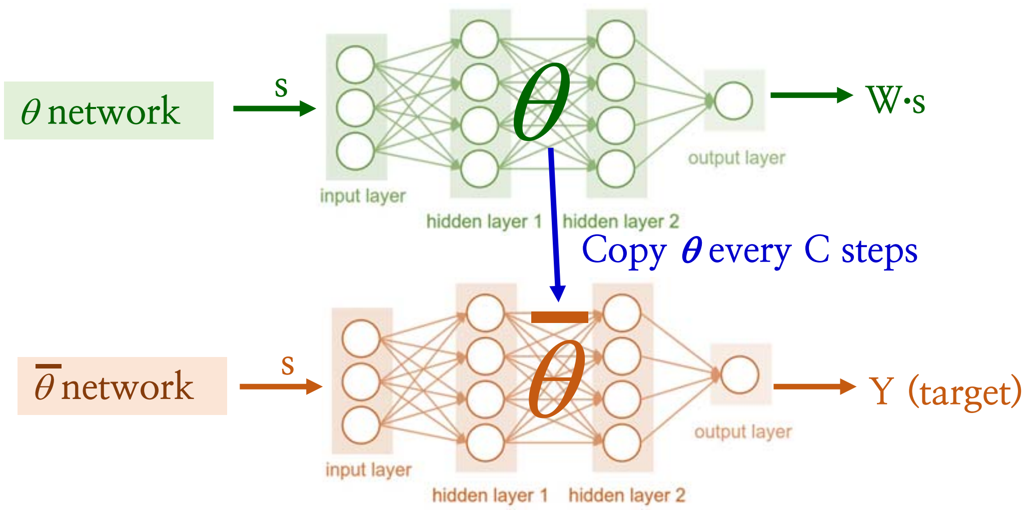
$$\Rightarrow\ \min_{\theta}\sum_{t=0}^{T}\Big[\hat{Q}\big(s_t,a_t|\theta\big)-\big(r_t+\gamma\max_{a'}Q\big(s_{t+1},a'|\bar{\theta}\big)\big)\Big]^2$$

⇒ Separate target network

And only update $\boldsymbol{\theta}$ in $\hat{Q}$

2. Separate target network

$$\min_{\theta}\sum_{j=0}^{N}\Big[\hat{Q}\big(s_j,a_j|\theta\big)-\big(r_j+\gamma\max_{a'}Q\big(s_{j+1},a'|\bar{\theta}\big)\big)\Big]^2$$



2. Separate target network

Prediction Q is obtained from θ network

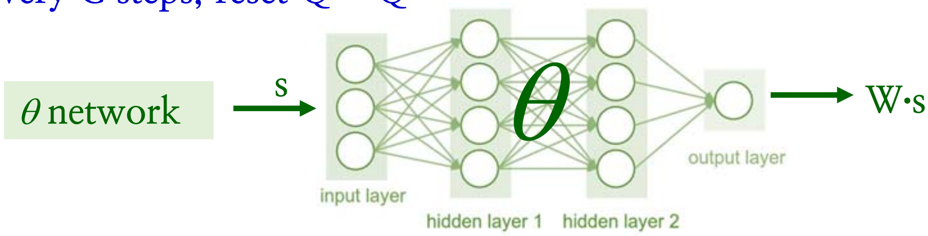
- Update the Q(=Ws=θs) to match the target y using gradient descent

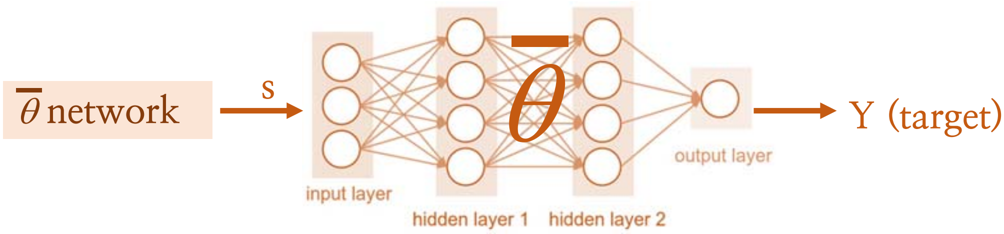
$$y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a' | \bar{\theta}) & \text{otherwise} \end{cases}, \text{then } \min\left(Q(\phi_j, a_j | \theta) - y_j\right)^2$$

Target y is obtained from $\bar{\theta}$ network

Update $\boldsymbol{\theta}$ by gradient descent but no update for y

- Every C steps, reset $\hat{Q}$ = Q





Programming: experience replay

- Initialize two networks of Q/Q^ functions with random weight $\theta_0 = \theta\hat{}_0$

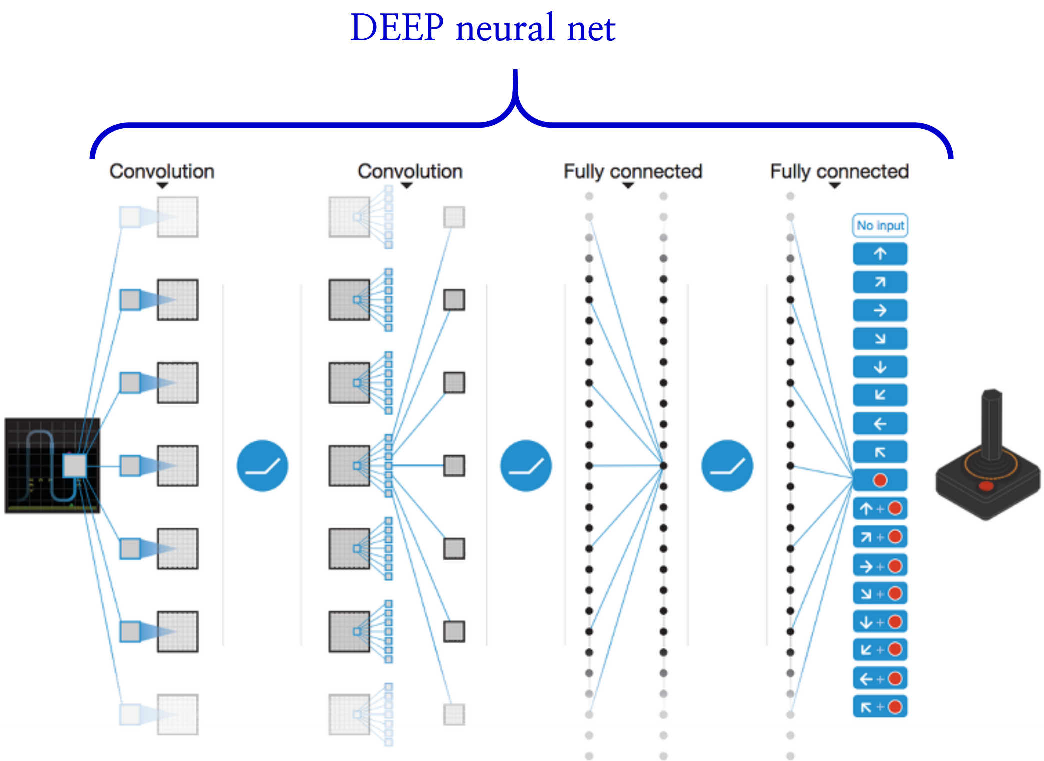
For loop for episodes

- Initialize $s_1 = \{x_1\}$ −or− preprocessed sequenced $\boldsymbol{\phi}_1 = \boldsymbol{\phi}(s_1)$
- For loop
  - Select an action a with ε−greedy, random or $a_j = \max_a Q^*(\phi_j, a | \theta)$
  - Execute action a in emulator and observe the image x_{t+1} and reward
  - Go to a new set $s_{t+1} = \{x_{t+1}\}$, or $\boldsymbol{\phi}_{t+1} = \boldsymbol{\phi}(s_{t+1})$(= here s_{t+1})
  - Store transition (ϕ_t, a_t, r_t, ϕ_{t+1}) in D
  - Sample random minibatch of transitions (ϕ_j, a_j, r_j, ϕ_{j+1}) from D
  - Update the Q(=Ws=θs) to match the target y using gradient descent

$$y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a' | \bar{\boldsymbol{\theta}}) & \text{otherwise} \end{cases} \Rightarrow \min\left(Q(\phi_j, a_j | \boldsymbol{\theta}) - y_j\right)^2$$

  - Every C steps, reset Q^ = Q

and Go Deep



Images from google

DQN programming