Machine Learning for Games

Learning Basics

- A characters (agent) earns its environment and the player.
  - As a player plays more, his characteristics traits can be better anticipated by the computer ➔ the behavior of the character can be tuned to playing styles.
- The majority of learning in Game AI is done offline (after tested exhaustively, before the game leaves the office), not online.

Learning Basics

- Intra-behavior learning vs inter-behavior learning
  - Intra-behavior: learning not a new behavior(action).

    a small area of a character's behavior is changed 🡐 example: learning the best patrol routes, where cover points are in a room.
  - Inter-behavior: learning a new behavior. 🡐 example: learning a way to kill an enemy such as laying an ambush.

Parameter Modification

Parameter Modification

- The simplest learning algorithm: calculates the value of one or more parameters, such as cost functions for pathfinding, the radius for arrival steering behavior.
- Method: define a **fitness** (goodness) value, and the search for the best parameter to **maximize (or minimize) the fitness value**.
- 만약 parameter가 가질 수 있는 “값의 범위”가 굉장히 넓은 경우는?

Parameter Modification

- Hill Climbing
- Simulated Annealing
- Genetic Algorithm

Hill Climbing

- **Step 1) Initially, a guess** is made as to the best parameter value. (randomly or deliberately). This parameter value is evaluated to get a score(fitness).
- Step 2) The algorithm then tries to work out in **what direction to change the parameter** in order to improve its score.
  - It does this by looking at nearby values for each parameter.
  - If it sees that the score increases in one or more directions, then it moves up the **steepest gradient(가장 크게 증가)**.

예) parameter 1개: p

direction 1: new v' 🡐 v - delta / direction 2: new v 🡐 v + delta

parameter 2개: p1, p2

direction 1: new v1 🡐 v1 - delta1, new v2 🡐 v2 – delta2

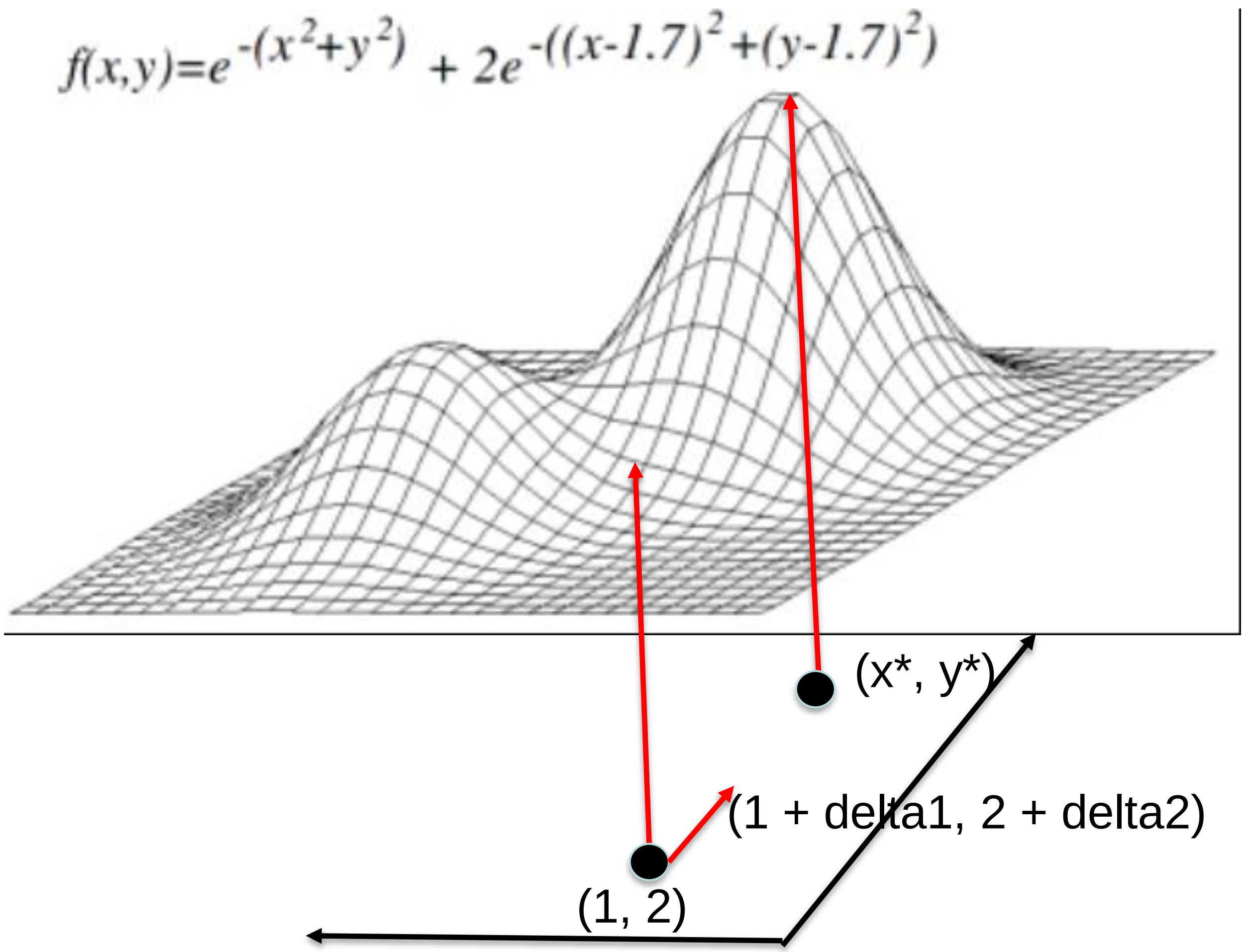
direction 2: new v1 🡐 v1 - delta1, new v2 🡐 v2 + delta2

direction 3: new v1 🡐 v1 + delta1, new v2 🡐 v2 – delta2

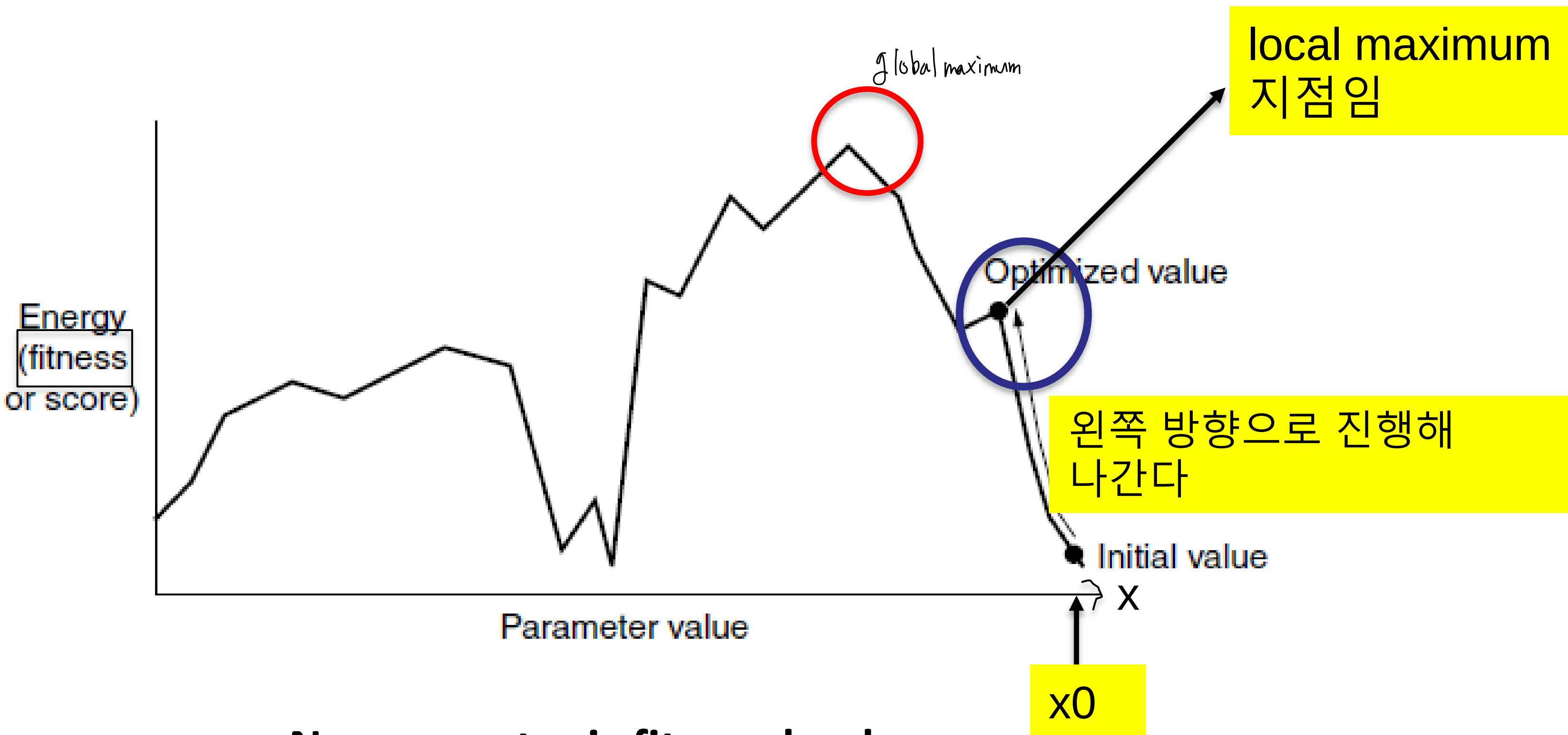
direction 4: new v1 🡐 v1 + delta1, new v2 🡐 v2 + delta2

Parameters: x and y

Find (x, y) such that the following f(x,y) is maximized.



Problem: Local Maximum
어떻게 해결할까?



Non-monotonic fitness landscape with sub-optimal hill climbing

여러개의 x0을 시도
down hill 이더라도 당분간 계속 전진.

Reinforcement Learning

- Motivation: To give a character free choice of any action in any circumstance and for it to work out **which actions are best for any given situation**.
- To give **feedback** a character only when something significant (event) happens **after the character take an action**. The character should **learn** that “all the actions leading up to the good result” are also good things to do.

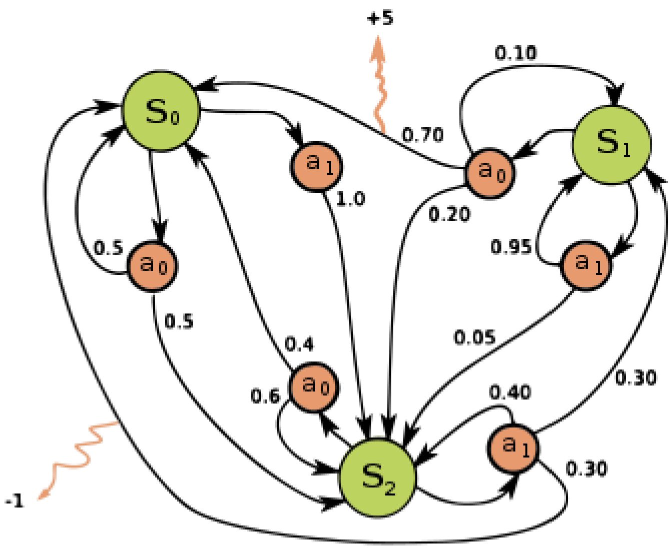
Definition of RL (Reinforcement Learning)

"a way of programming agents by reward and punishment without needing to specify *how* the task is to be achieved"

[Kaelbling, Littman, & Moore, 96]

음성 설명 없음

Markov Decision Process 와 RL 관계



States, Actions, Probabilities for going to the next state after taking an action, Rewards obtained after taking actions.

Find a policy π s)
 that determines an action for each state s so as to maximize cumulative rewards obtained from start state s0.

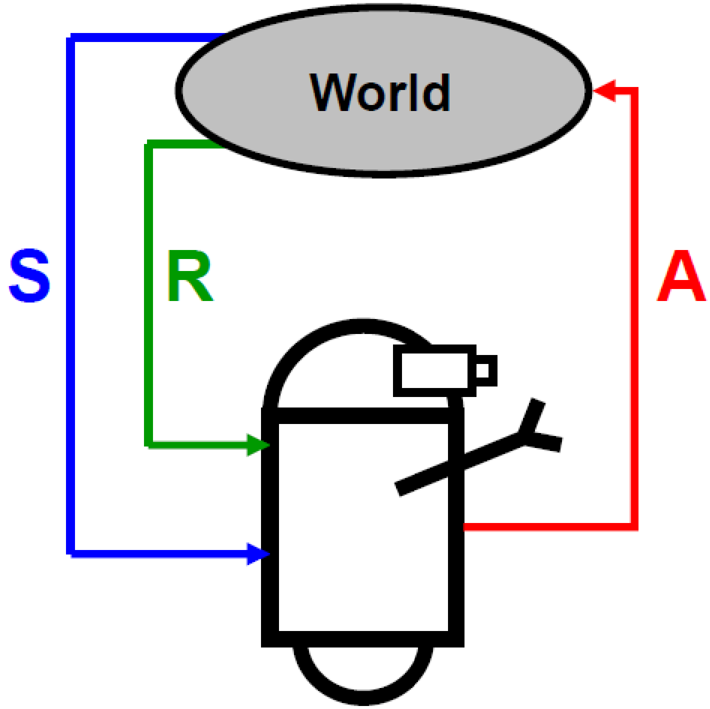
What if we don't how the whole MDP ? ➔ RL

음성 설명 없음

Basic RL Model

음성 설명 없음

1. Observe state, s_t
2. Decide on an action, a_t
3. Perform action
4. Observe new state, s_{t+1}
5. Observe reward, r_{t+1}
6. Learn from experience
7. Repeat



S_t : 시간 t에서의 상태, S_{t+1} : 시간 (t+1)에서의 상태
a_t : 시간 t에서, 선택하는 액션 (여러 후보 액션들 중에서 미래 총 reward가 가장 큰 것으로 추정되는 액션)
r_{t+1} :: a_t 를 수행 후, 상태가 S_{t+1} 으로 바뀌고 나서 얻게 되는 reward

Q-learning에서는, **Q(s,a)**가 "상태 s에서 액션 a를 선택해 수행 후, 예상되는 미래 총 reward" 를 나타냄. Q-learning에서는, 위의 Step 6에서 Q(s, a)를 update 함을 말 함.

음성 설명 없음

An Example: Gridworld

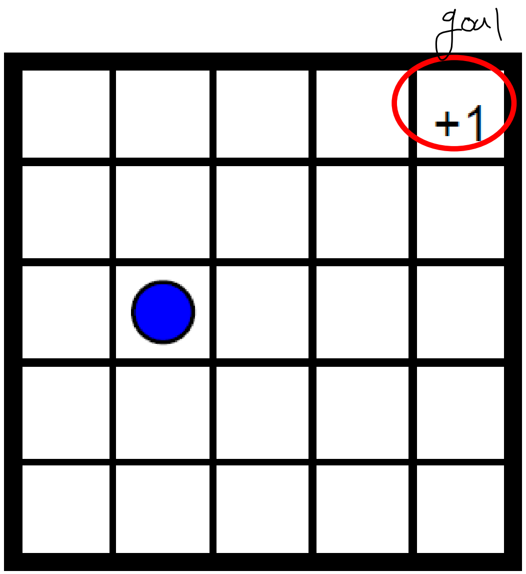
강화학습은 사례들을 모으는 것임

S0 a1 ==> (S1). 총보상 1 100
S0 a2 ==> (S2) 총보상 2 200
S0 a3 ==> (S3) 총보상3 150
S1. a1 ==> (S4) 총보상 4

Canonical RL domain

- States are grid cells
- 4 actions: N, S, E, W
- Reward for entering top right cell
- -0.01 for every other move

동서남북 action 가능



Finding a path for maximizing sum of rewards
➔ Shorted Path

A RL Example for Angry Birds Game

Angry Birds 게임 설명: https://en.wikipedia.org/wiki/Angry_Birds_(video_game)

음성 설명 없음

We define the action to be $a \in \{0, 1, 2, 3, , 90\}$
where each discrete number represents the angle of the shot

$$reward = \begin{cases} 1, & \text{if } score > 3000 \\ -1, & \text{otherwise} \end{cases}$$

가정: pig들 위치와 구조물 상태에 맞추어, 가장 적절한 reference point(angry bird가 날라와서 부딪히는 지점)은 직접 계산함.

Q-Learning

- *Q*-learning treats the game world as **a state machine**. At any point in time, the algorithm is in some state.
- The **state** should encode **all the relevant details about the character's environment** and internal data, such as position, proximity of the enemy, health level, etc.
- For each state, the algorithm needs to **understand the actions that are available** to it.
- After the character carries out one action in the current state, the **reinforcement function should give it feedback**. Feedback can be positive or negative and is often zero. After carrying out an action, the character is likely to **enter a new state**.

Q-Learning

gamma

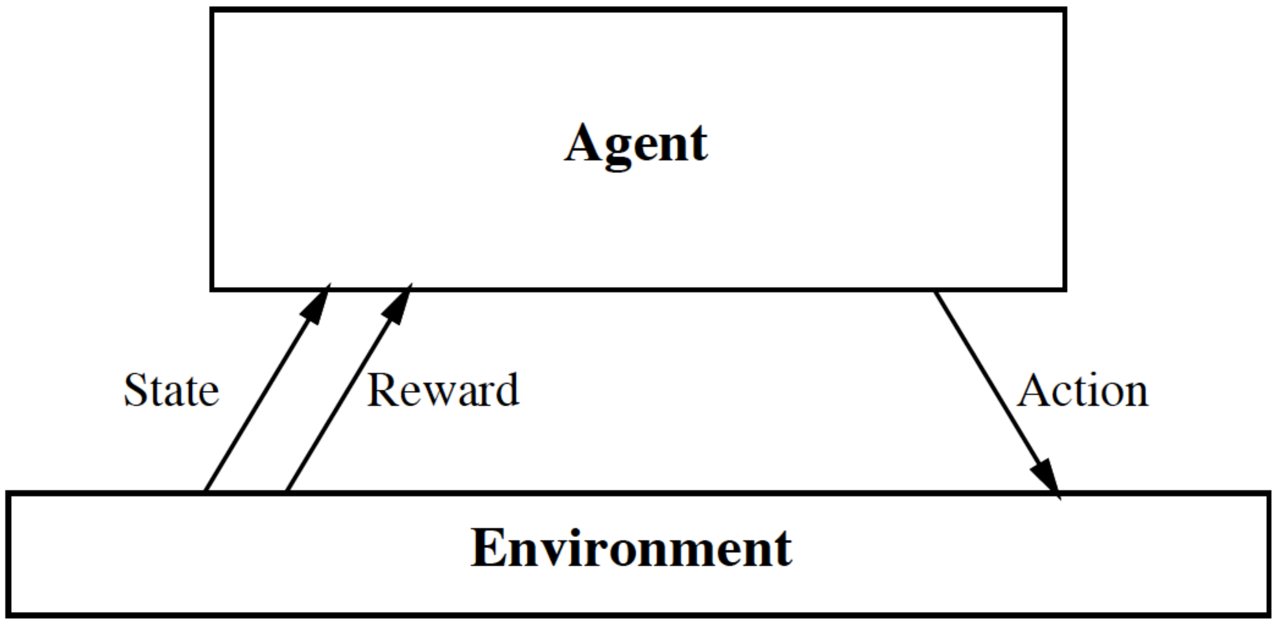
- *Experience tuple: f*our elements -- the start state, the action taken, the **reinforcement value**, and the resulting state – are written as (*s, a, r, s'*)
- *Q*-value: it represents how good it thinks that action is to take when in that state.
- Updated by Q-learning rule:

$$Q(s,a) = (1-\alpha)\boxed{Q(s,a)} + \alpha\boxed{\left(r + \gamma \max\left(Q(s',a')\right)\right)},$$

α is the **learning rate**, and γ is the **discount rate**. Both are parameters of the algorithm. The r value is the **new reinforcement** from the experience tuple.

$$\boxed{\max_{a' \in A(s')}\left(Q(s',a')\right)}$$

The Big Picture



$$s_0 \xrightarrow[r_0]{a_0} s_1 \xrightarrow[r_1]{a_1} s_2 \xrightarrow[r_2]{a_2} \dots$$

Your action influences the state of the world which determines its reward

Q-Learning

Q-learning system code:

*Set state **s** to a randomly chosen state*

For in 0 …Iterations:

if random() < nu, then pick a new state s;

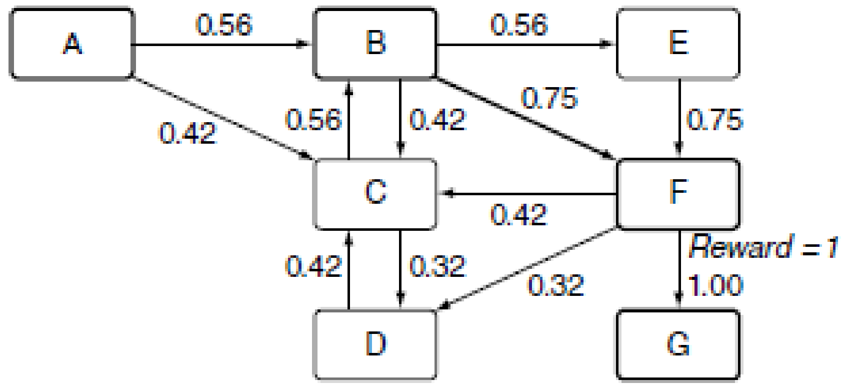
*Pick an action **a**; //policy for selection action*

***reward**, **newstate** = problem.takeAction(s,a)*

*Update Q(**s**,**a**)*

***s** = **newstate**;*

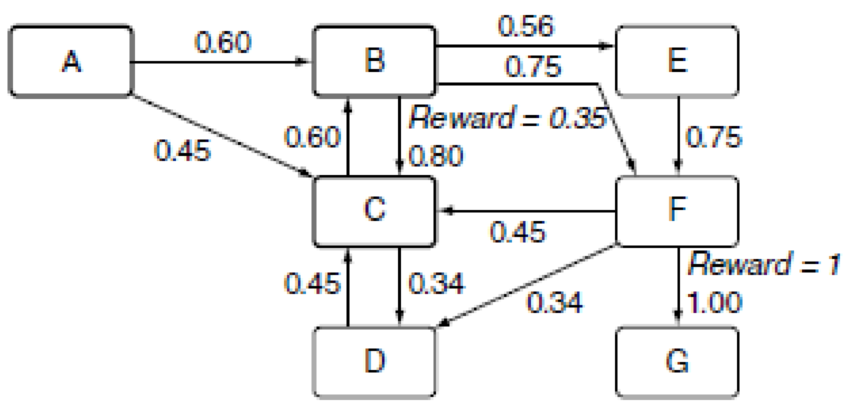
$$Q(s,a) = (1-\alpha)Q(s,a) + \alpha\left(r + \gamma \max\left(Q(s',a')\right)\right),$$



A learned state machine

Labels are Q-values

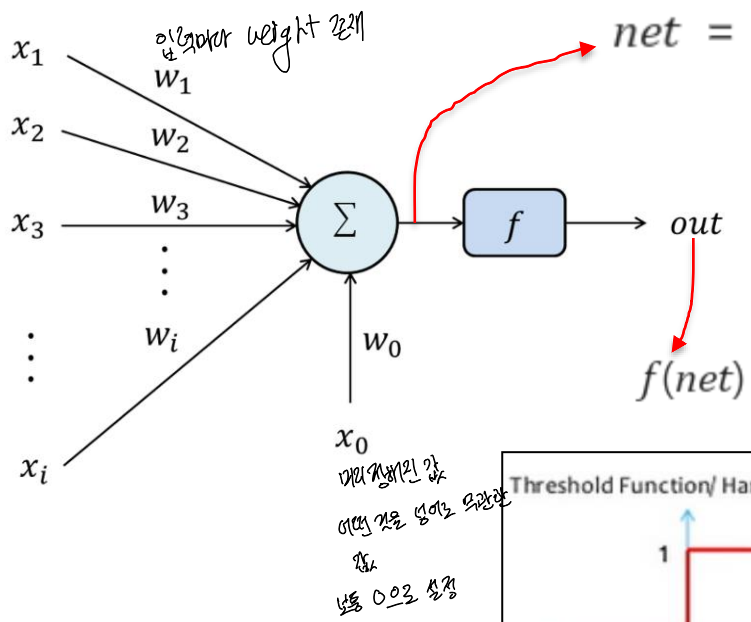
Very sensitive to reward.



A learned machine with additional rewards

Artificial Neural Network Based Learning

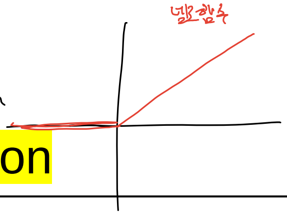
- # Neural Network(NN) Basic Operation
  - Perceptron



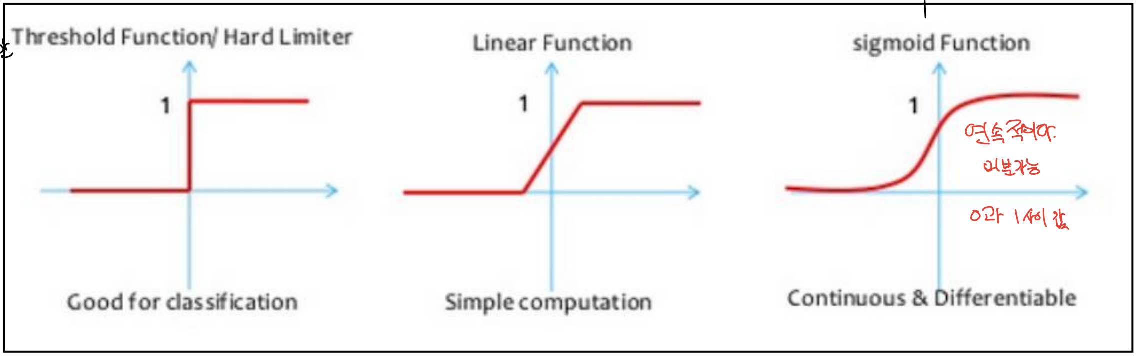
$$net = \sum_{i}^{N} w_i x_i + \boxed{w_0 x_0}$$

bias

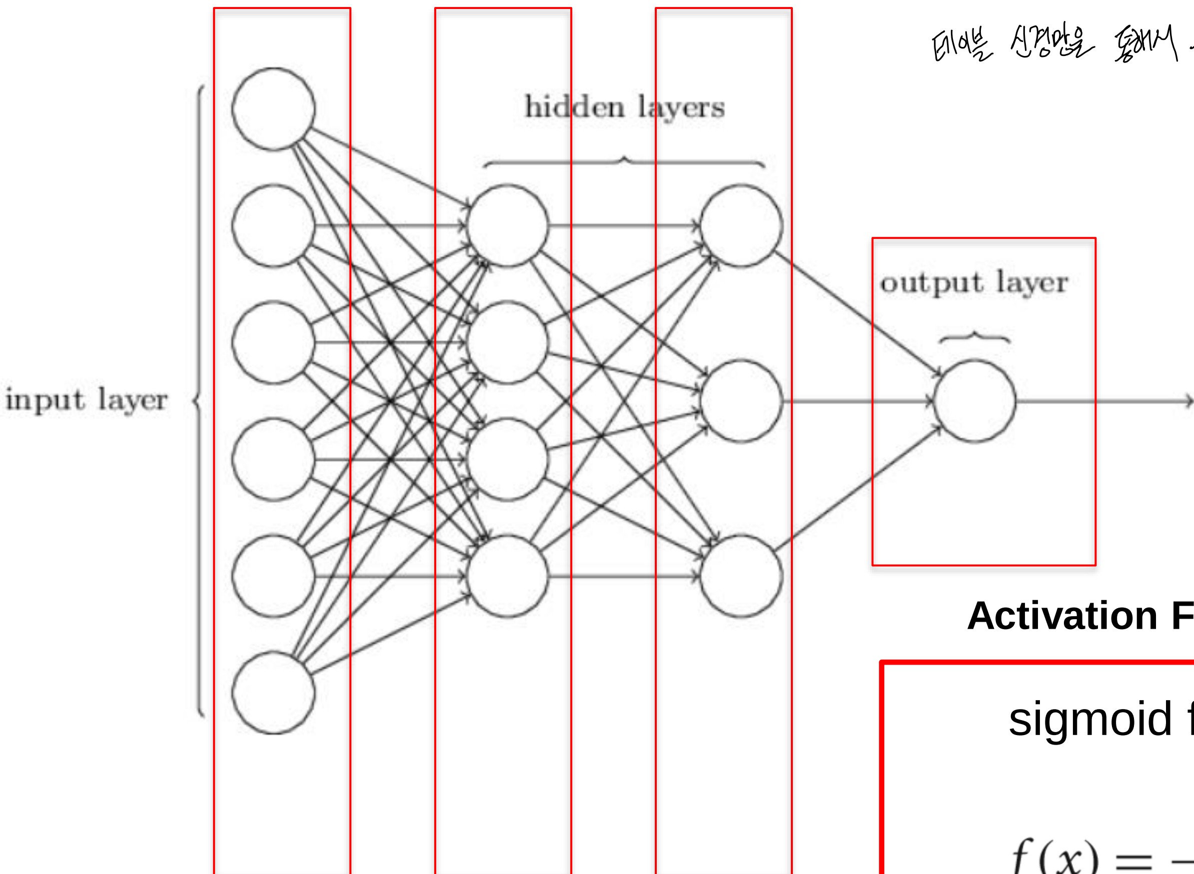
활성화 함수



f: activation function



– Four layer NN



디어블 신경망을 통해서 유사하게 사용

최적의 w 값을 찾아서 학습

Activation Function

sigmoid function

$$f(x) = \frac{1}{1 + e^{-hx}}$$

feedforward network

The multi-layer perceptron takes inputs from all the nodes in the preceding layer and sends its single output value to all the nodes in the next layer.

recurrent network 주식처럼 time series data들의 과거에 따르는 데이터를 훈련

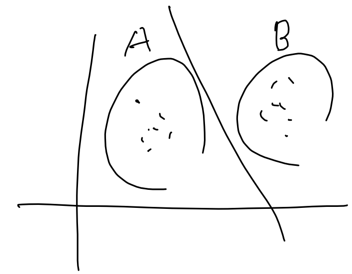
connections that lead from a later layer back to earlier layers. This architecture is known as a *recurrent network*

음성 설명 없음

- **Deep** Neural Net
  - an artificial neural network (ANN) with one or multiple layers (**hidden layers**) between the input and output layers.

인공신경망(ANN) 용도

- Every function
  - Classification　　　　　제일 많이 씀
  - Feature detection
  - Regression
  - **Clustering**
  - **Association**
  - Various computations
  - etc
- Applications: Image Processing, Character recognition, Forecasting, etc.



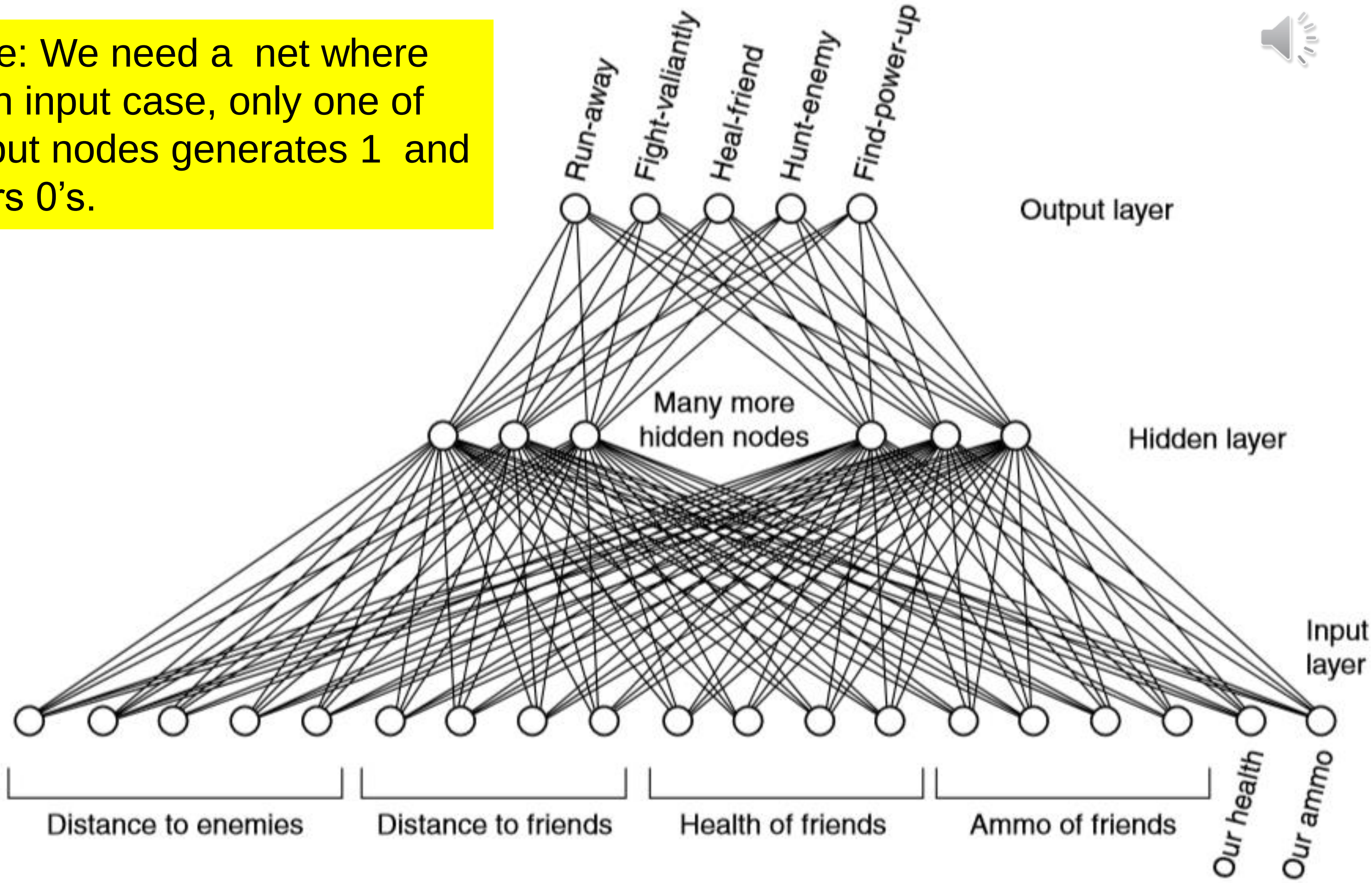
We'd like to group a set of input values (such as distances to enemies, health values for friendly units, or ammo levels) together so that **we can act differently for each group**.

1) a group of **"safe" situations**, where health and ammo are high and enemies are a long way off. ➔ Our AI can go looking for power-ups or lay a trap.

2) a group of **life-threatening situations** where ammo is spent, health is perilously low, and enemies are bearing down. ➔ This might be a good time to run away in blind panic.

3) a **"fight-valiantly" group**. If the character was healthy, with ammo and enemies nearby, it would naturally do its stuff. But it might do the same if it was on the verge of death, but had ammo, and it might do the same even in improbable odds to altruistically allow a squad member to escape.

Example: We need a net where given an input case, only one of the output nodes generates 1 and all others 0's.



Backpropagation : a Learning Rule

- for **supervised learning**, the **most popular learning algorithm**
- 원리: 마지막 layer에서 우리가 원하는 target output과 현재 network가 produce한 estimated output끼리의 **loss (또는 error)를 계산하는 function**을 정의하고 **➔ 그 값을 minimize하도록 weight value 조절**. **➔** 마지막 layer부터 뒤쪽(즉 input layer쪽)으로 차례로 weight values를 update해 나감.
- In our example: See the next page

$\delta_{j:}$

음성 설명 없음.

- In our example:

Let the set of neuron states be o_j, where j is the neuron, and w_{ij} is the weight between neurons i and j. The equation for the updated weight value is

$$w'_{ij} = w_{ij} - \eta \nabla C$$

$$w'_{ij} = w_{ij} + \eta \delta_j o_i,$$

node i 에서 node j 로 가는 edge의 현 weight value.

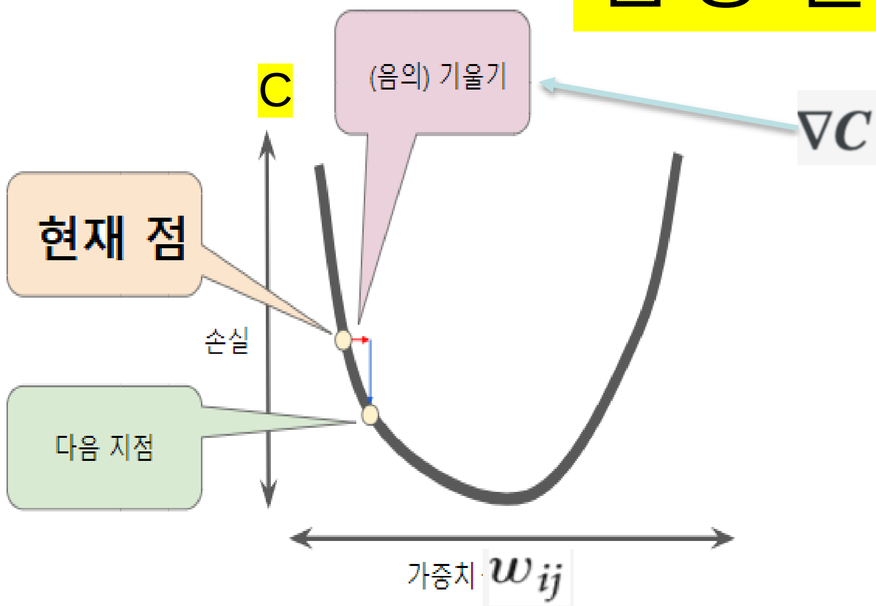
where η is a gain term, and δ_j is an error term

node i 에서 node j 로 가는 edge의 새로운 weight value.

δ_j: is based ont the difference (loss) between the expected output and the current output .
∇C : gradient of cost function C (C is defined by difference, named "loss", between the expected output and the current output of the network for a given input)

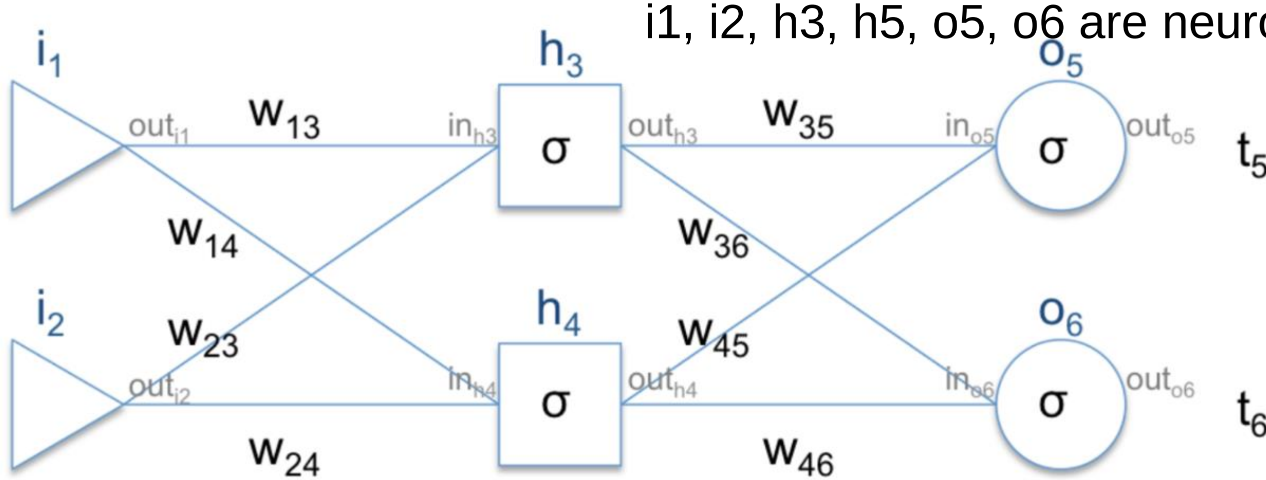
Activation Function: A requirement for backpropagation algorithm is a **differentiable** activation function.

음성 설명 없음.



gradient of cost function C (C is defined by difference, named “loss”, between the expected output, named “target output:, and the current output of the network for a given input)

t5, t6이 expected(target) output.
i1, i2, h3, h5, o5, o6 are neurons.



Supervised Learning vs 지도학습 Unsupervised Learning vs 비지도학습 Reinforcement Learning with ANN(Artificial Neural Network)

음성 설명 없음.

음성 설명 없음.

Supervised Learning with ANN:

- a full set of labeled data while training an algorithm.
- each example in the training dataset is tagged with the answer (a label) the algorithm should come up with on its own.
- backpropagation for error-correction
- 응용 분야:

숫자 이미지를 입력 받아서, 어떤 숫자 인지를 알아냄

- predicting the price of an apartment in San Francisco based on square footage, location and proximity to public transport.

Unsupervised Learning with ANN:

- The training dataset is a collection of examples without a specific desired outcome or correct answer.
- The neural network then attempts to automatically find structure in the data
- Approaches: SOM(Self-Organizing Map), Autoencoder.

음성 설명 없음.

- 응용 분야:

고객의 구매금액, 구매시간, 구매물품 데이터르 이용해, 고객을 4가지 그룹으로 clustering.

Fill an online shopping cart with diapers, applesauce and sippy cups and the site just may recommend that you add a bib and a baby monitor to your order.

Reinforcement Learning with ANN

Store a huge Q(s,a) table using a neural network. How ?

음성 설명 없음.