05

함수 근사법

3. 함수 근사법

https://github.com/multicore-it/rl

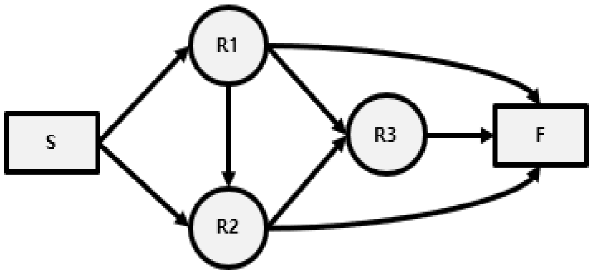
경사하강법 단순 환경 데이터 표현

https://github.com/multicore-it/rl

단순 환경 데이터 표현

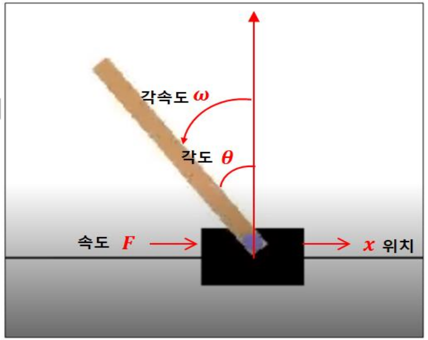
- 경우의 수가 작은 환경에서는 데이터를 배열로 표현할 수 있다.

Environment



State	Actions
S	0.5
	0.0
	0.5
R1	0.2
	0.4
	0.4
R2	0.4
	0.0
	0.6
R3	0.0
	1.0
	0.0

action values

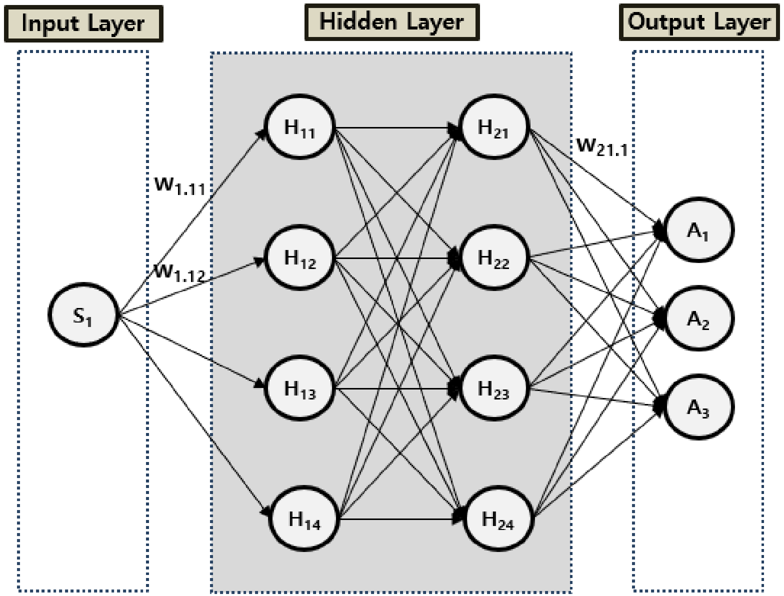


com/multicore-it/rl

경사하강법 복잡 환경 데이터 표현

https://github.com/multicore-it/rl

복잡 환경 데이터 표현



- 상태와 행동의 개수가 굉장히 많은 복잡한 환경에서 모든 데이터를 배열로 표현 불가능
- 근사함수를 사용해서 상태, 행동, 정책 등 모든 요소를 표현 가능
- 인공신경망이 근사함수로 많이 사용됨
- 인공신경망은 이론적으로 모든 데이터를 표현 가능
- 학습을 통해 가중치와 편향을 찾아야 함

https://github.com/multicore-it/rl
https://github.com/multicore-it/rl

경사하강법 인공신경망 활용

https://github.com/multicore-it/rl

인공신경망 활용한 함수 근사

- 가중치 w로 표현되는 인공신경망으로 상태가치함수(v)와 행동가치함수(q)를 근사할 수 있음

Neural Network		Array
$\hat{v}(s, w)$	≒	$v_\pi(s)$
$\hat{q}(s, a, w)$	≒	$q_\pi(s, a)$

https://github.com/multicore-it/rl
https://github.com/multicore-it/rl

경사하강법 평균제곱오차

평균제곱오차(MSE: Mean Squared Error)

- 평균제곱오차 : 두 값의 차이를 구하고 그 결과를 제곱해서 평균을 구하는 것
- 차에 제곱을 하는 것은 값의 방향(음, 양)이 중요한 것이 아니라 그 크기가 중요하기 때문
- 목표 함수를 MSE 형태로 도출하고 SGD(Stochastic Gradient decent)를 활용해서 MSE를 최소화하는 방향으로 학습을 진행

Neural Network		Array
$\hat{v}(s, w)$	≒	$v_\pi(s)$
$\hat{q}(s, a, w)$	≒	$q_\pi(s, a)$

https://github.com/multicore-it/rl
https://github.com/multicore-it/rl
https://github.com/multicore-it/rl

경사하강법 함수 근사법

https://github.com/multicore-it/rl

목표함수 MSE

- 참가치함수(신만이 알 수 있는 아주 정확한 가치함수)를 알고 있다고 가정할 때, 변수 w로 표현되는 인공신경망으로 근사
- 목표함수 MSE를 최소화하는 변수 w를 찾는 것이 학습의 목표

Goal(MSE)

$$J(w) = E_{\pi}\left[(v_{\pi}(s) - \hat{v}(s, w))^2\right]$$

https://github.com/multicore-it/rl

https://github.com/multicore-it/rl

경사하강법 함수 근사법

경사하강법 사용

- MSE를 학습률 변수(α)를 추가해서 값을 최소화하는 방향(-)으로 학습
- 미분 공식을 활용해서 프로그래밍이 편리한 방법으로 식을 변경
- 문제는 기댓값(E_π)

Gradient Decent

$$\Delta w = -\frac{1}{2} \propto \nabla_w J(w) \quad ②\text{-}1$$

$$J(w) = E_\pi [(v_\pi(s) - \hat{v}(s, w))^2]$$

$$= \propto E_\pi [(v_\pi(s) - \hat{v}(s, w)) \nabla_w \hat{v}(s, w)] \quad ②\text{-}3$$

미분공식

②-2

$$y = (a - x)^2$$

$$y' = 2(a - x)(a - x)'$$

$$= -2(a - x) \nabla_w x$$

https://github.com/multicore-it/rl

경사하강법 함수 근사법

https://github.com/multicore-it/rl

확률적 경사하강법 사용

- SGD는 전체 데이터를 사용하는 것이 아니라 샘플링을 하기 때문에 기대값(E_π) 을 없앨 수 있다.
- 앞에서 배운 MC와 TD 개념을 사용해서 간단히 할 수 있음

Stochastic Gradient Decent	$\Delta w = \propto (v_\pi(s_t) - \hat{v}(s_t, w))\ \nabla_w \hat{v}(s_t, w)$ ①
MC	$\Delta w = \propto (G_t - \hat{v}(s_t, w))\ \nabla_w \hat{v}(s_t, w)$ ②
TD	$\Delta w = \propto (R_{t+1} + \hat{v}(s_{t+1}, w) - \hat{v}(s_t, w))\ \nabla_w \hat{v}(s_{t+1}, w)$ ③

https://github.com/multicore-it/rl

https://github.com/multicore-it/rl

경사하강법 함수 근사법

확률적 경사하강법 사용

- SGD는 전체 데이터를 사용하는 것이 아니라 샘플링을 하기 때문에 기대값(E_{π}) 을 없앨 수 있다.
- 앞에서 배운 MC와 TD 개념을 사용해서 간단히 할 수 있음

Stochastic Gradient Decent	$\Delta w = \propto (v_{\pi}(s_t) - \hat{v}(s_t, w))\ \nabla_w \hat{v}(s_t, w)$	①
MC	$\Delta w = \propto (G_t - \hat{v}(s_t, w))\ \nabla_w \hat{v}(s_t, w)$	②
TD	$\Delta w = \propto (R_{t+1} + \hat{v}(s_{t+1}, w) - \hat{v}(s_t, w))\ \nabla_w \hat{v}(s_{t+1}, w)$	③

경사하강법 함수 근사법

https://github.com/multicore-it/rl

Q함수 사용 사용

Goal(MSE)	$J(w) = E_\pi [(q_\pi(S,A) - \hat{q}(S,A,\ w))^2]$
Gradient Decent	$\Delta w = -\frac{1}{2} \propto \nabla_w J(w)$ $= \propto E_\pi [(q_\pi(S,A) - \hat{q}(S,A,\ w))\ \nabla_w \hat{q}(S,A,\ w)]$
Stochastic Gradient Decent	$\Delta w = \propto (q_\pi(S,A) - \hat{q}(S,A,\ w))\ \nabla_w \hat{q}(S,\ A,w)$
MC	$\Delta w = \propto (G_t - \hat{q}(S_t, A_t,\ w))\ \nabla_w \hat{q}(S_t, A_t,\ w)$
TD	$\Delta w = \propto (R_{t+1} + \gamma \hat{q}(S_{t+1}, A_{t+1}, w) - \hat{q}(S_t, A_t, w))\ \nabla_w \hat{q}(S_t, At, w)$

↓

DQN

- 가치함수 대신 Q함수 사용 가능
- 행동가치함수를 인공신경망을 사용해서 표현하는 방법을 DQN(Deep Q Learning)이라 한다.

https://github.com/multicore-it/rl

https://github.com/multicore-it/rl