Monte Carlo Tree Search와 Deep Neural Network를 활용한 근사적 동적계획법

저자
(Authors)

신교홍, 이태식

출처
(Source)

대한산업공학회 추계학술대회 논문집 , 2016.11, 8-33(26 pages)

발행처
(Publisher)

대한산업공학회
Korean Institute Of Industrial Engineers

URL

http://www.dbpia.co.kr/journal/articleDetail?nodeId=NODE07059983

APA Style

신교홍, 이태식 (2016). Monte Carlo Tree Search와 Deep Neural Network를 활용한 근사적 동적계획법. 대한산업공학회 추계학술대회 논문집, 8-33

Monte Carlo Tree Search와 Deep Neural Network를 활용한 근사적 동적계획법

신 교 홍, 이 태 식
한국과학기술원 산업 및 시스템 공학과
hong906@kaist.ac.kr, taesik.lee@kaist.edu

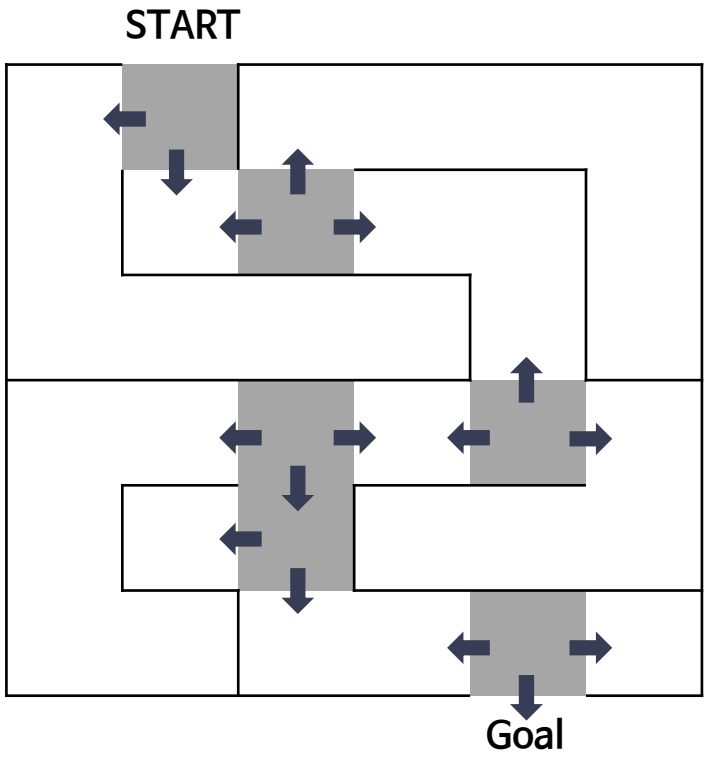
Complex Systems Design Laboratory
Dept. of Industrial and Systems Engineering
KAIST Korea Advanced Institute of Science and Technology

Nov 19, 2016
2016년 추계학술대회

근사적 동적계획법(Approximate Dynamic Programming) (1/2)

- START 지점에서 출발하여 Goal 지점에 도착한 경우 보상을 받는 시스템

작은 크기의 간단한 시스템



각 의사결정이 필요한 시점 × 가능한 모든 행동

모든 경우에 대해 수행

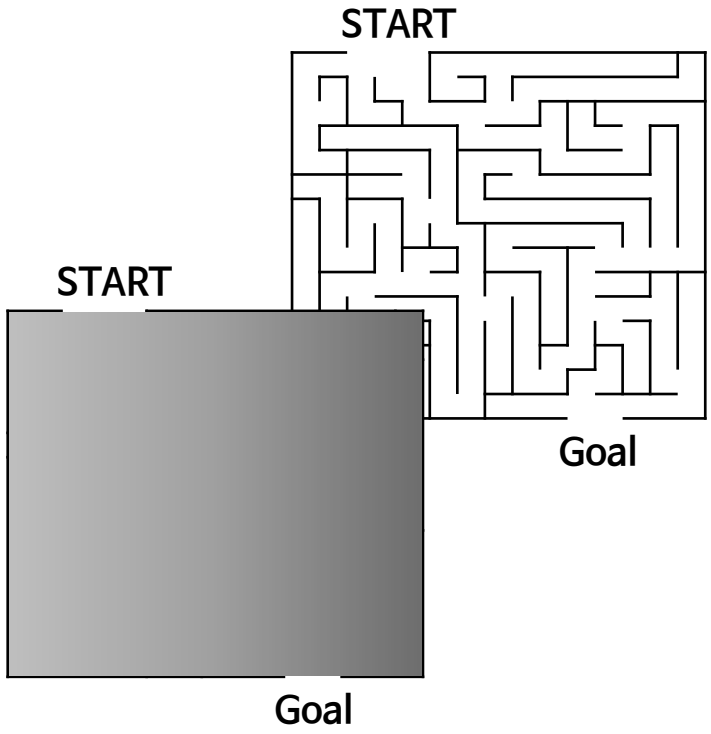
최적 의사결정 도출

* 동적 계획법(Dynamic Programming) - Appendix 참고

근사적 동적계획법 (Approximate Dynamic Programming) (2/2)

- START 지점에서 출발하여 Goal 지점에 도착한 경우 보상을 받는 시스템

큰 크기의 복잡한 시스템
작동 과정을 모르는 시스템



가능한 모든 행동들의 수행에 무리가 있음

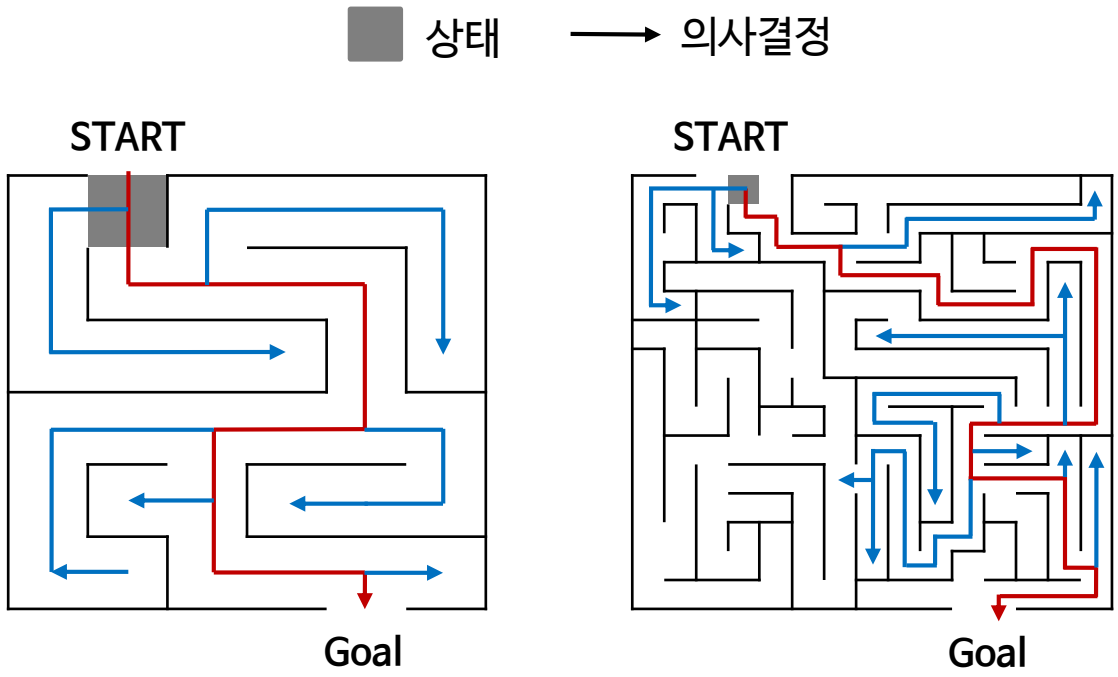
시스템을 통한 다수의 경험

- Reward
- Penalty
- None

경험의 누적을 통해
최적에 근접한 의사결정 도출

근사적 동적계획법의 활용 및 한계 (1/2)

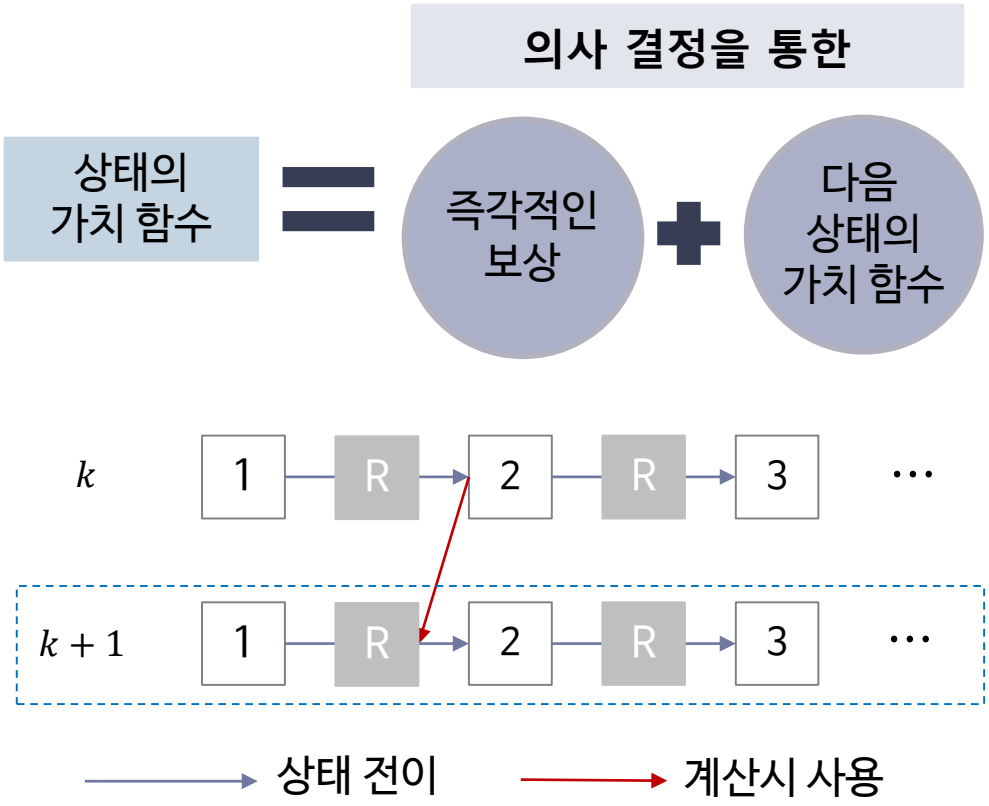
- 실제 시스템을 모사한 시뮬레이션 또는 확률분포 등을 통해 표본 경로(sample path) 생성함 [1]
  - 생성한 표본 경로 상에 있는 상태(State)의 가치 함수(Value function)만 계산하여 사용 → **근사적**



상태 공간 또는 의사 결정 공간이 넓은 경우, 방문하지 못하는 상태가 많아짐

근사적 동적계획법의 활용 및 한계 (2/2)

- 상태의 가치 함수 값을 잘 계산하기 위해서 각 상태를 많이 방문하는 것이 필요함



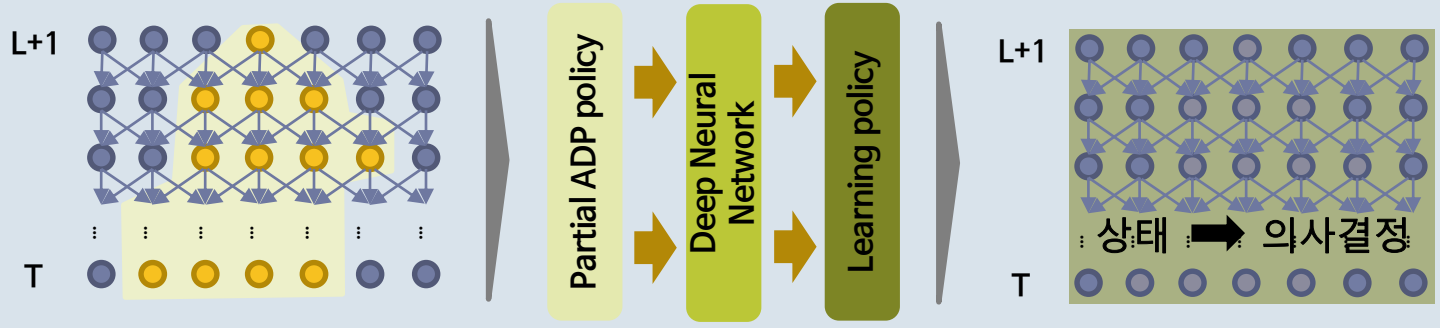
상태의 방문이 여러 번 이루어지지 않으면 상태의 가치 함수 값을 제대로 근사하지 못함

연구 목적

1. Monte Carlo Tree Search

Selection | Simulation | Backpropagation | Expansion

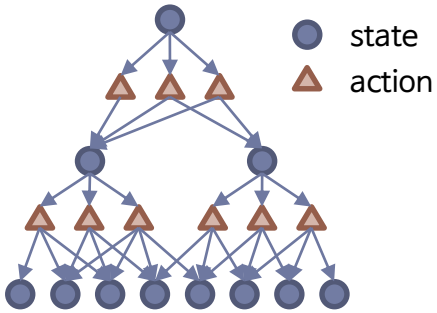
2. Deep Neural Network를 이용한 heuristic Policy 생성



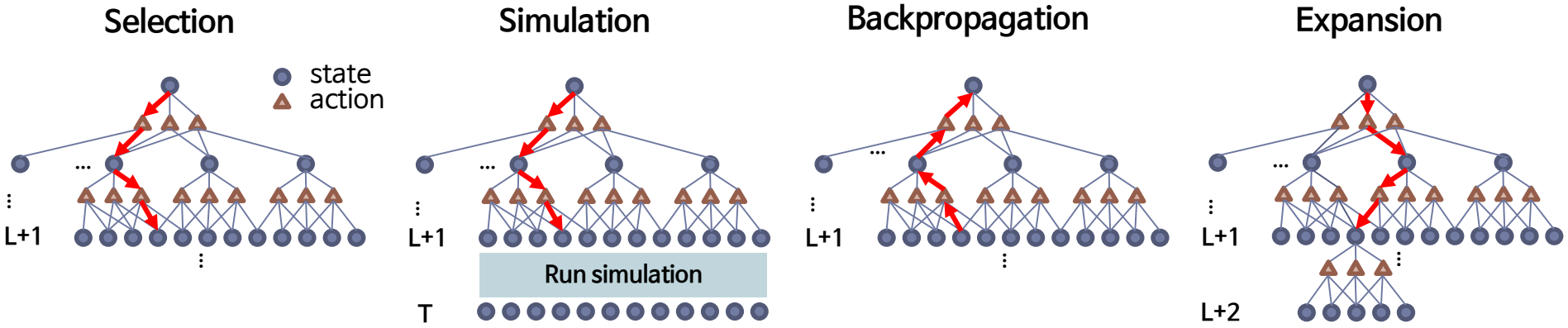
상태 공간 또는 의사결정 공간이 넓을 때
빠른 시간 안에 가치 함수를 잘 근사하여 최적에 가까운 의사결정 방침을 얻고자 함

Monte Carlo tree search (MCTS) 도입 (1/2)

- Tree Search
  - 가능한 모든 상태, 의사결정에 대해서 계산하는 방법
  - Decision tree 계산 방식



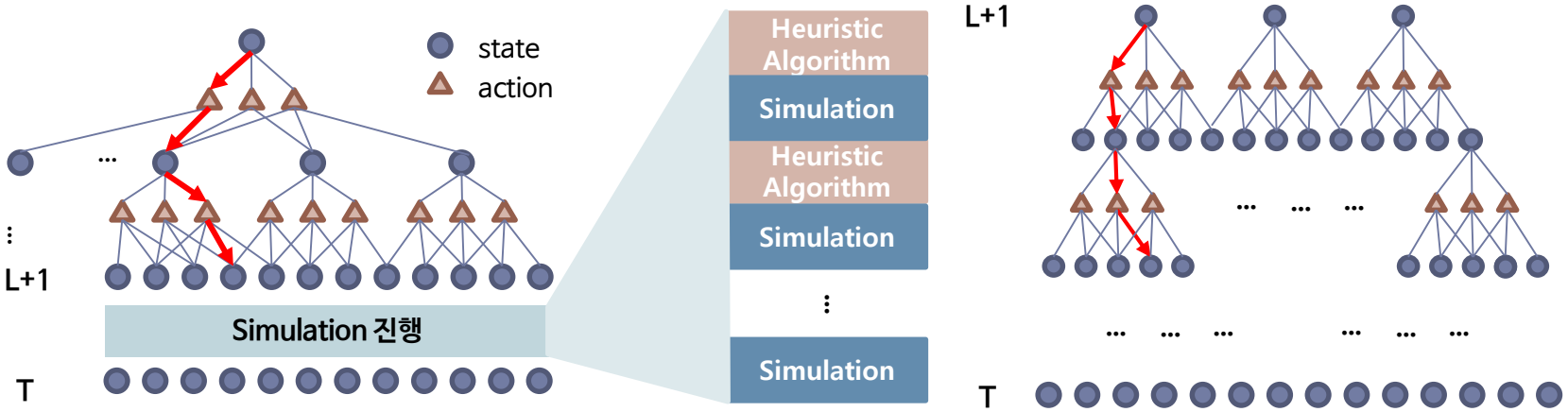
- Monte Carlo Tree Search [2]
  - 표본(Sample)을 통해 얻어진 상태의 가치 함수 값 계산함
  - 어떤 의사결정을 내리는 것이 좋을지 분석하면서 트리를 확장함



Monte Carlo tree search (MCTS) 도입 (2/2)

- Monte Carlo Tree Search의 Simulation 과정
  - L+1 번째 노드부터 최종 노드까지 heuristic algorithm 을 사용하여 도착한 상태에서 선택할 의사결정을 정함
  - 시뮬레이션 진행 동안 얻어진 보상 값을 모두 합하여 L+1 번째 노드의 가치 함수 값으로 사용함

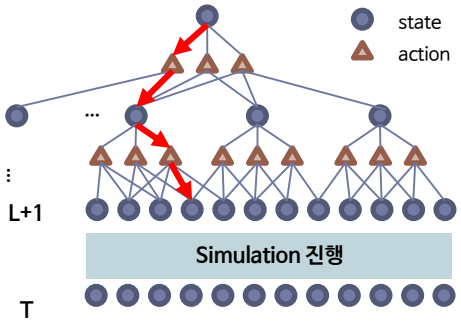
- heuristic algorithm
  - 기존에 알려져 있는 최적 의사결정을 선택함
  - 일반적인 순차적 의사결정 모델은 기존에 알려져 있는 최적 의사결정이 존재하지 않음



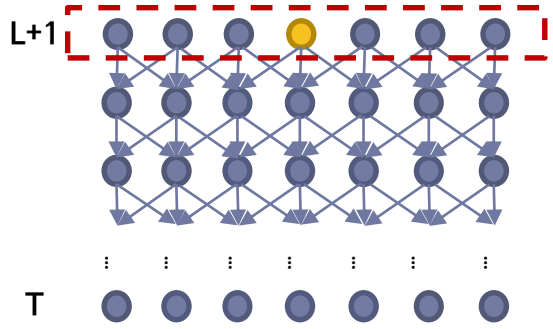
L+1번째 노드의 가치 함수를 제대로 근사하기 위한 heuristic algorithm이 필요함

Simulation 과정에 사용되는 heuristic algorithm (1/3)

- 제안하는 algorithm
  - ① 특정 L+1 노드부터 끝(T)까지 일반적인 근사적 동적계획법을 수행함
  - ② 근사적 동적계획법을 통해 얻어진 상태 공간 일부분의 policy를 학습함
  - ③ Policy를 학습한 network를 이용하여 본 문제의 근사적 동적계획법을 수행할 때 Simulation 과정에 필요한 의사결정 결정함

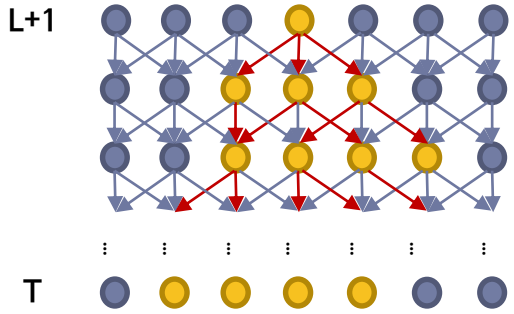


①-1



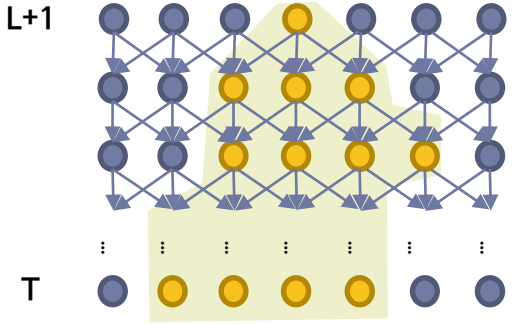
다수의 L+1번째 노드 중 임의의 하나를 선택

①-2



표본 경로를 생성하여 상태의 가치 함수 계산

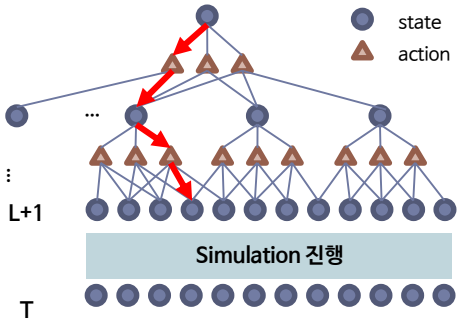
①-3



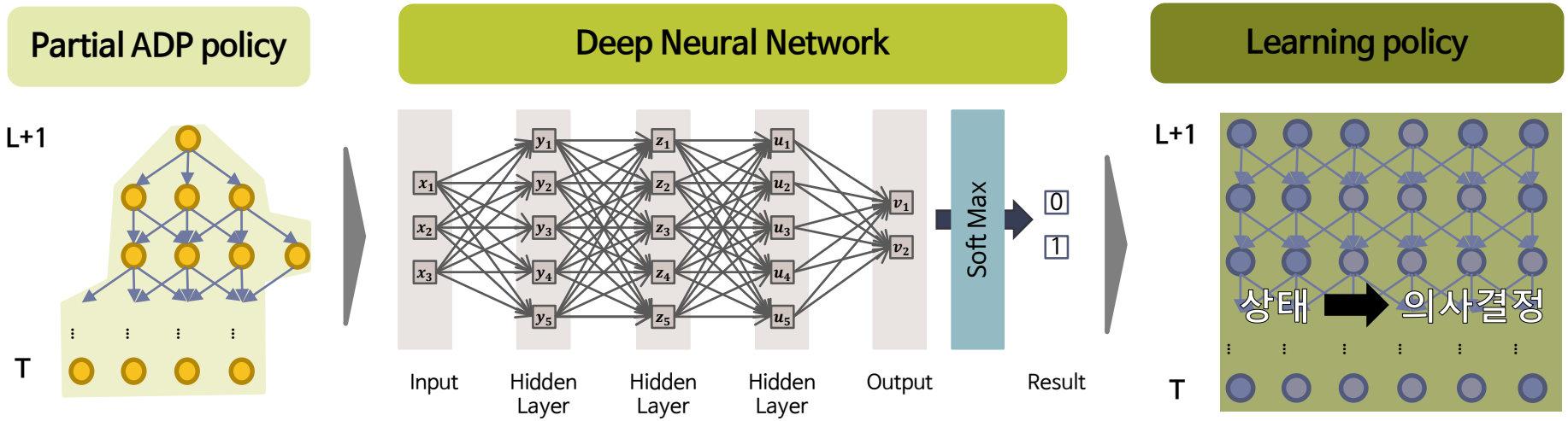
방문한 상태에서의 최적 의사결정 도출

Simulation 과정에 사용되는 heuristic algorithm (2/3)

- 제안하는 algorithm
  - ① 특정 L+1 노드부터 끝(T)까지 일반적인 근사적 동적계획법을 수행함
  - ② 근사적 동적계획법을 통해 얻어진 상태 공간 일부분의 policy를 학습함
  - ③ Policy를 학습한 network를 이용하여 본 문제의 근사적 동적계획법을 수행할 때 Simulation 과정에 필요한 의사결정 결정함



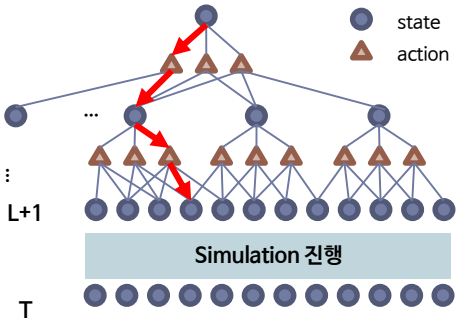
②



최종 결과가 주어진 training set을 가장 잘 설명할 수 있는 weight 학습 [3]

Simulation 과정에 사용되는 heuristic algorithm (3/3)

- 제안하는 algorithm
  - ① 특정 L+1 노드부터 끝(T)까지 일반적인 근사적 동적계획법을 수행함
  - ② 근사적 동적계획법을 통해 얻어진 상태 공간 일부분의 policy를 학습함
  - ③ Policy를 학습한 network를 이용하여 본 문제의 근사적 동적계획법을 수행할 때 Simulation 과정에 필요한 의사결정 결정함

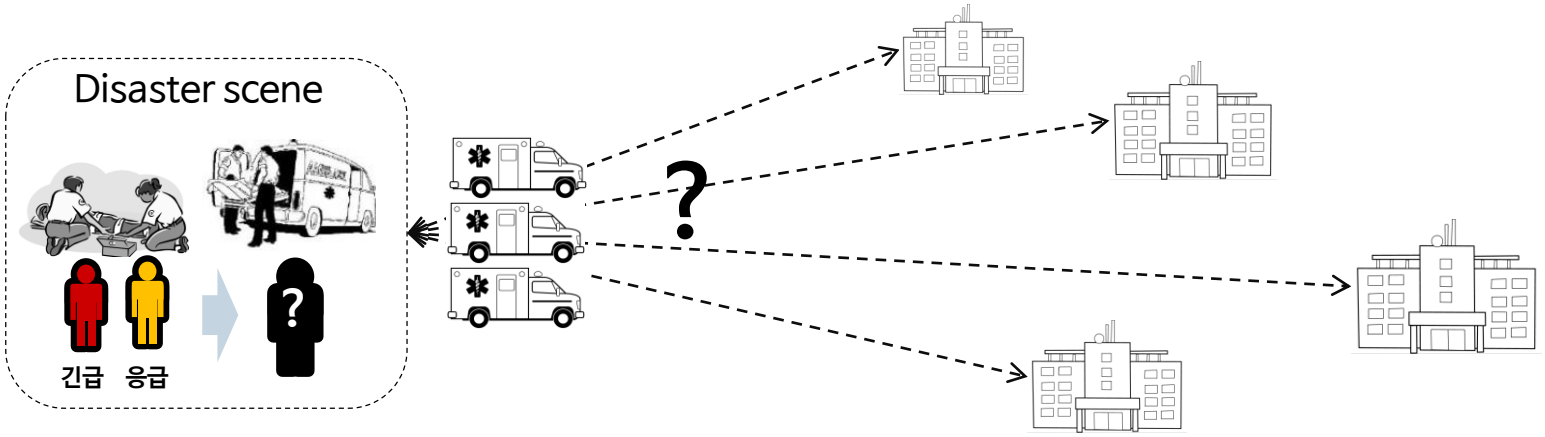


현재의 상태

현재의 상태에서의 최적 의사결정

성능 검증을 위한 순차적 의사결정 모델

- 다중손상사고 환자 이송 우선순위와 이송병원 결정 문제 [4]
  - 평균 생존자 수 최대화
  - 부상자의 생존율과 응급실 내 대기 시간 고려



- 모델 환경
  - 사고 현장에 발생한 환자는 두 유형으로 분류
  - N 대의 구급차가 M 곳의 응급실로 환자 이송
  - 응급실 내 긴급환자 우선처치 프로토콜 반영

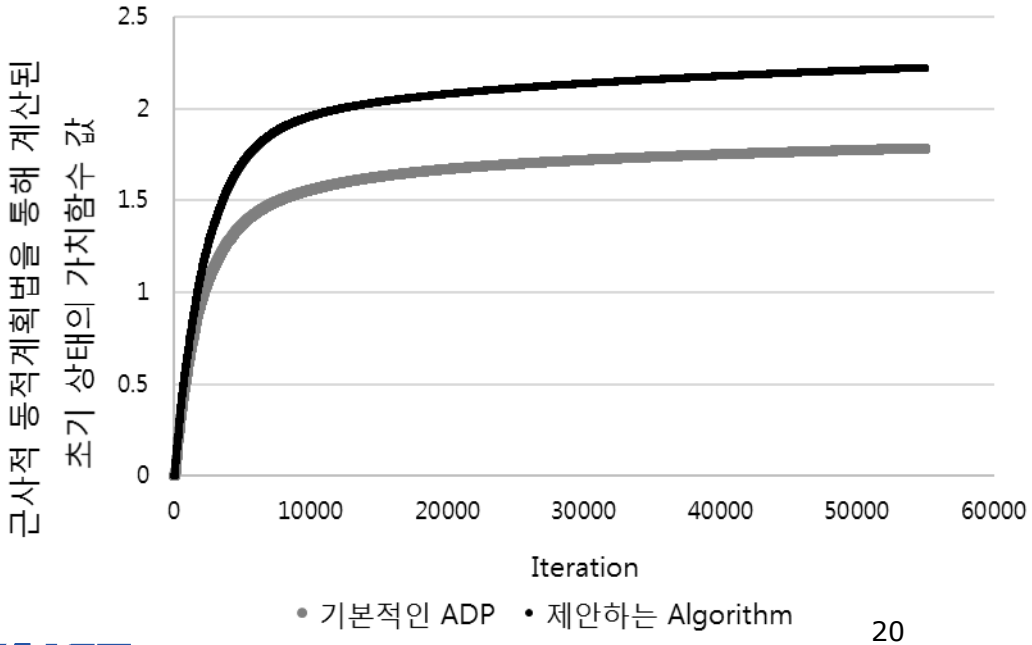
* 모델의 구성요소 및 실험 시나리오 - Appendix 참고

기본적인 근사적 동적계획법*과의 비교

- DNN 결과
  - 부분 모델의 초기 노드 : L = 10
    (사고 발생 16분 후, 긴급 환자 1명, 응급 환자 6명)
  - 원 모델의 초기 노드 : L = 0
    (사고 발생 직후, 긴급 환자 3명, 응급 환자 9명)

	Training set	Test set
상태 수	160200	16020
정확도	88.6%	88.5%

- Algorithm 성능 비교



	기존 방식	제안하는 Algorithm
Iteration 수	55000	55000
시작 상태의 가치 함수 값	1.785	2.323
소요 시간	66분	32분*

* 근사적 동적 계획법 Algorithm - Appendix 참고

* 32분 = 20분(partialADP) + 6분(DNN) + 6분(MCTS)

정리 및 결론

- 복잡한 시스템을 모사한 순차적 의사결정 모델의 최적 의사 결정을 구하기 위해 근사적 동적계획법이 주로 사용됨

- 상태 공간이 상당히 넓은 경우, 유사한 표본 경로 생성이 힘들고 각 상태의 충분한 수의 방문이 어려워 각 상태의 가치 함수 값을 제대로 근사하지 못함

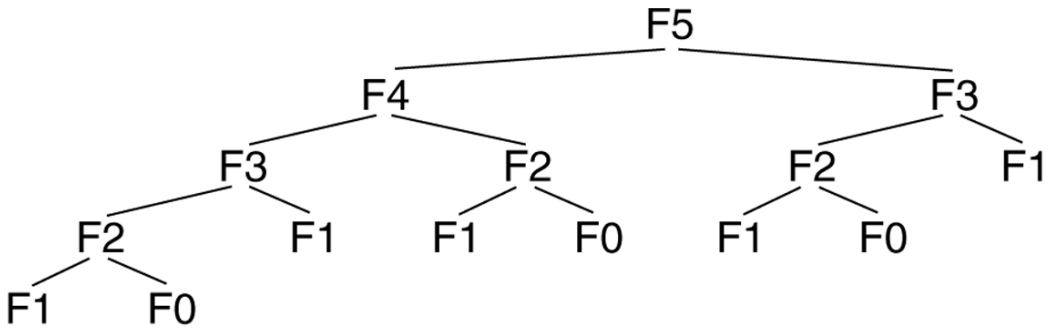
- Monte Carlo Tree Search와 Deep Neural Network의 활용
  - 특정 시점 이후, 상태의 가치 함수 값을 근사하기 위한 Simulation 단계의 heuristic policy가 필요함
  - 특정 시점 이후의 상태 공간 중 일부에 대해 근사적 동적계획법을 활용하여 policy를 생성하고 해당 policy를 DNN으로 학습하여 특정 시점 이후의 상태가 주어졌을 때 의사결정을 내려주는 heuristic policy로 사용함

- 예시 모델을 통해 기본적인 근사적 동적계획법과 비교하여 더 빠른 시간에 가치 함수 값을 잘 근사함을 확인함

Appendix

Dynamic Programming (1/3)

- 순차적인 의사결정 문제를 최적화하기 위한 방법으로 작은 문제들의 해를 이용하여 큰 문제의 해를 구하는 알고리즘
  - 유사한 문제를 반복적으로 풀며 작은 문제의 해를 사용 (recursion, divide and conquer 과 비교)
  - 작은 문제부터 시작하여 해당 문제의 결과값을 보다 큰 문제 해결에 사용하여 반복적으로 같은 문제를 푸는 경우를 줄임



Dynamic Programming (2/3)

- Infinite horizon problem을 해결하기 위한 DP로 두가지가 존재 [4]

Value Iteration

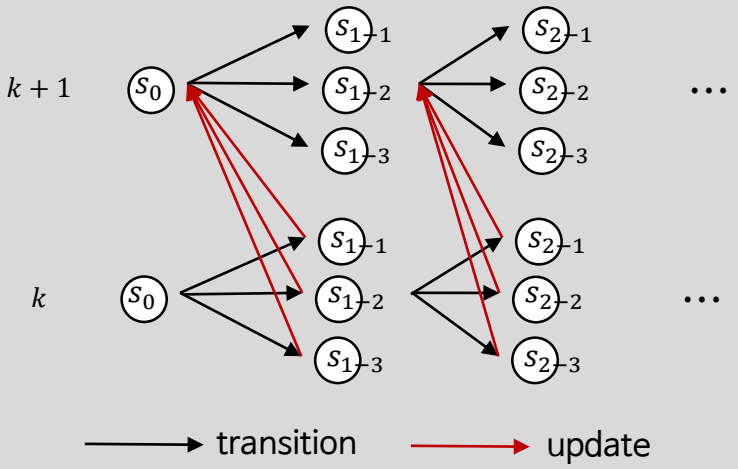
Bellman Optimality Equation 활용

가능한 모든 (i, a)의 value 값을 0으로 초기화 한 뒤

반복적으로 가능한 모든 (i, a)의 value 값에 대해

이전 iteration 결과를 활용하여 업데이트 해줌으로써

Bellman Optimality Equation에 수렴하도록 계산



Algorithm

Step 1.

$k = 1$ 로 설정

가능한 모든 (i, a) 에 대해 $Q^k(i, a) = 0$ 초기화

$\epsilon > 0$ 값 고정.

Step 2.

가능한 모든 (i, a) 에 대해 다음을 계산

$$Q^{k+1}(i, a) \leftarrow \sum_{j=1}^{|S|} p(i, a, j) \left[r(i, a, j) + \gamma \max_{b \in A(j)} Q^k(j, b) \right]$$

Step 3.

모든 $i \in S$에 대해 다음을 계산

$$J^{k+1}(i) = \max_{a \in A(i)} Q^{k+1}(i, a), \quad J^k(i) = \max_{a \in A(i)} Q^k(i, a)$$

$\|J^{k+1} - J^k\| < \epsilon(1-\gamma)/2\gamma$ 라면 step 4,

아니라면 $k = k + 1$ 로 변경 후 step 2로 이동.

Step 4.

모든 $i \in S$에 대해

$d^*(i) = \underset{a \in A(i)}{argmax}\, Q^k(i, a)$ 을 통해 최적 policy 도출.

Dynamic Programming (3/3)

- Infinite horizon problem을 해결하기 위한 DP로 두가지가 존재 [4]

Policy Iteration

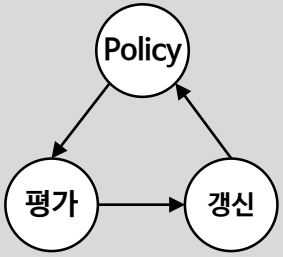
Bellman Policy Equation 활용

임의로 **한 policy 선정**

Bellman Policy Equation으로 수렴할 때까지 계산
해당 policy로 시스템을 운영했을 때의 value 값을 계산
(Policy Evaluation)

얻어진 Q-value 값을 통해 각 상태에서의 최적 의사결정 도출
(Policy Improvement)

동일한 Policy가 나올 때까지 갱신된 policy를 사용하여 Evaluation 및 Improvement 반복 수행



Algorithm

Step 1.
$k = 1$ 로 설정
임의의 policy $\hat{d}_k$ 선택

Step 2.
가능한 모든 (i, a) 에 대해 $Q^k(i, a) = 0$ 초기화 후
다음을 계산

$$Q_{\hat{d}_k}(i,a) \leftarrow \sum_{j=1}^{|S|} p(i,a,j)\left[r(i,a,j) + \gamma Q_{\hat{d}_k}(j, \hat{d}_k(j))\right]$$

(Q value가 수렴할 때까지)

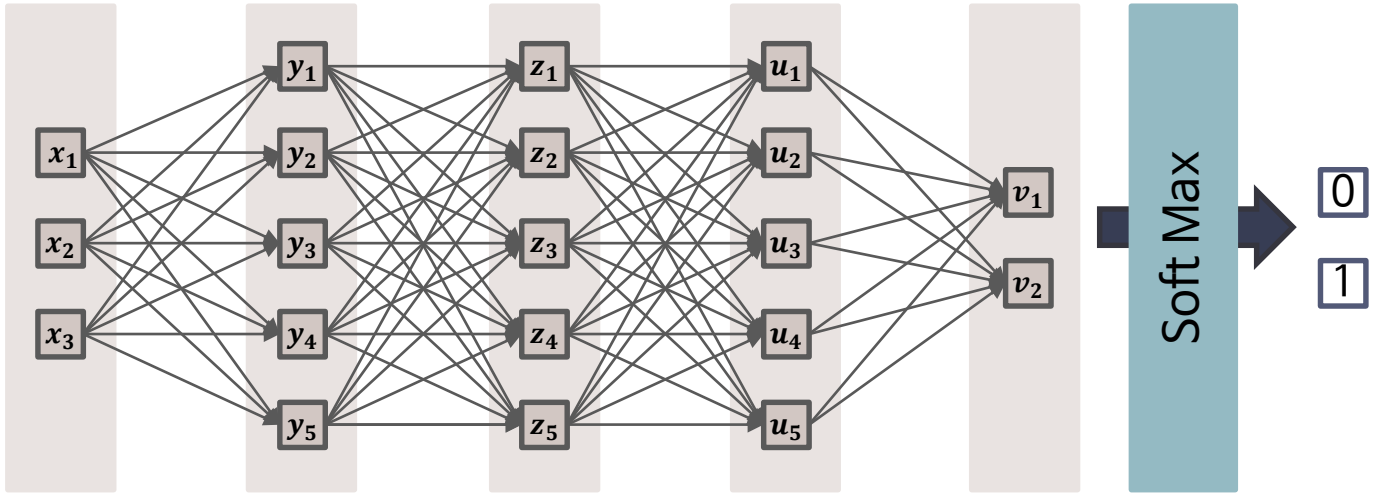
Step 3.
모든 $i \in S$에 대해

$\hat{d}_{k+1}(i) = \underset{a \in A(i)}{argmax}\, Q_{\hat{d}_k}(i,a)$ 을 통해 최적 policy 도출,

(가능하다면 $\hat{d}_{k+1} = \hat{d}_k$을 만족하도록 도출)

Step 4.
$\hat{d}_{k+1} = \hat{d}_k$ 이라면 종료,
아니라면 $k = k + 1$ 로 변경 후 step 2로 이동.

Deep Neural Network



$$y_i = f\left(\sum_{j=1}^{3} x_j w_{ji}\right) \quad z_k = f\left(\sum_{i=1}^{5} y_i w_{ik}\right) \quad u_l = f\left(\sum_{k=1}^{5} z_k w_{kl}\right) \quad v_m = f\left(\sum_{l=1}^{2} u_l w_{lm}\right)$$

$$SoftMax = \frac{e^{v_m}}{\sum_{m=1}^{2} e^{v_m}}$$

최종 결과가 주어진 training set을 가장 잘 설명할 수 있는 weight 학습 [3]

확률적 의사결정 모델 구성요소 (1/2)

- 상태(State) [4]
  - $S = (t, P, R, H)$
  - $P = (n_{I,}\, n_D)$ n_I : 현장 내 긴급환자 수, n_D : 현장 내 응급환자 수
  - $R = (R_a)_{a \in A}$: 자원 상태 벡터(Resource state vector)

    R_a : a 특성를 가지는 구급차의 수 (A : 특성 집합)
  - $a = (a_1, a_2, a_3, a_4)$

    a_1 : 구급차의 상태, a_2 : 이송 환자 중증도, a_3 : 구급차의 현 상태의 시작 시점, a_4 : 할당 응급실
  - $H = (h_1, \ldots, h_M)$ h_j : j 응급실의 상태 정보

    $h_j = (r_j, y_j, \delta_j)$ r_j : 긴급환자 수, y_j : 응급환자 수, δ_j : 응급실의 서비스 시작 시점

- 의사결정 시점
  - 초기 시점과 구급차가 현장에 도착한 시점

- 행동 (Action)

  - $X(s) = \{x_{ph}(s) : p \in \{I, D\}, h \in \{1, \ldots, M\}\}$

확률적 의사결정 모델 구성요소 (2/2)

- 상태전이 [4]
  - $S' = S^M(S, X, W(S, X))$
  - $W(S, X)$: 시뮬레이션을 통해 얻어진 표본경로를 따라 상태 전이

- 보상 함수 (Reward Function)
  - 병원에 대기 중이던 환자가 서비스를 받기 시작하는 시점의 생존율
  - $R(s,x) = \begin{cases} \beta_{jI} \times f_I(t) & if\ \delta_j = t\ \ and\ \ r_j > 0 \\ \beta_{jD} \times f_D(t) & if\ \delta_j = t\ \ and\ \ r_j = 0\ \ and\ \ y_j > 0 \end{cases}$
  - β_{jk} : 응급실 j의 k 등급환자 치료 역량
  - $f(t)$: t 시점의 환자 생존율 함수

응급실 서비스 시작 시점이 현 시점이고 대기에 긴급환자가 있는 경우

응급실 서비스 시작 시점이 현 시점이고 대기에 응급환자만 있는 경우

- 목적 함수 (Objective function)
  - $V(s) = \max_{x \in X}[R(s,x) + E(V(s')|s,x)]$

실험 시나리오

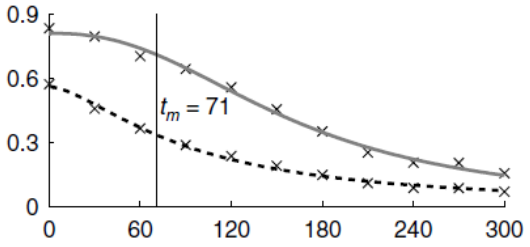
- 실험 시나리오
  - 12명의 부상자 발생 (긴급환자 : 3명, 응급환자 : 9명)
  - 3차 병원 2곳, 2차 병원 2곳 (2차 병원에서 긴급환자 치료 시 생존율 80% 감소 가정)
  - 4대의 구급차가 환자 이송

- 실험 변수 설정
  - 병원 관련 변수 (Travel time = 실제 거리/(60km/h))

Hospital	Hospital classification	Travel time (min)	Service time (min)
1	Tertiary	~Lognormal(26, 10.4)	~Exp(20)
2	Tertiary	~Lognormal(34, 13.6)	~Exp(20)
3	Secondary	~Lognormal(26, 10.4)	~Exp(30)
4	Secondary	~Lognormal(30, 12)	~Exp(30)

  - 환자 관련 변수 ($f_i(t) = \frac{\beta_{0,i}}{(t/\beta_{1,i})^{\beta_{2,i}+1}}$)

긴급환자			응급환자		
$\beta_{0,I}$	$\beta_{1,I}$	$\beta_{2,I}$	$\beta_{0,D}$	$\beta_{1,D}$	$\beta_{2,D}$
0.56	91	1.58	0.81	160	2.41



29

시뮬레이션 기반의 근사적 동적 계획법 [5]

단계 0. 초기화

단계 0.1 모든 상태 s에 대해 $\bar{V}^0(s)$ 초기화

단계 0.2 초기 상태 ${s_0}^1$ 선정

단계 0.3 $n = 1$ 로 설정

단계 1. 이산 사건 시뮬레이션 환경 초기화

단계 2. 조건 ($n_I > 0$ or $n_D > 0$ 또는 병원 내 환자 수 > 0 또는 이송 중인 환자 수 > 0) 만족 시 수행

단계 2.1 개발 (exploitation) 과정 선택 시 ($\geq e^{-\delta \times n}$), 하단의 식 해결

$$\hat{v}^n = \max_{x \in X} \left(R(s^n, x^n) + E\left(\bar{V}^{n-1}(s'^n) \middle| s^n, x^n \right) \right)$$

x^n 는 최대화 문제 해결을 통해 얻어진 x 의 값

탐사 (exploration) 과정 선택 시 ($< e^{-\delta \times n}$), 현 상태에서 선택 가능한 의사 결정 중 임의로 선택 후 $\hat{v}^n$ 계산

단계 2.2 하단의 식을 이용하여 $\bar{V}^n(s)$ 갱신

$$\bar{V}^n(s) = \left(1 - \alpha_{s,\, n-1}\right) \bar{V}^{n-1}(s) + \alpha_{s,\, n-1} \hat{v}^n$$

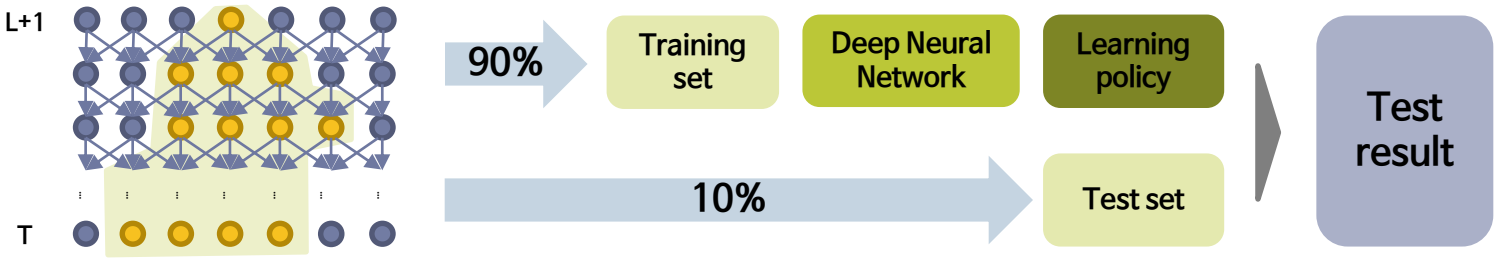
단계 2.3 이산 사건 시뮬레이션을 통해 다음 상태 추출 ($\leq t + 1$)

$$s' = S^M\left(s, x, w(s, x)\right)$$

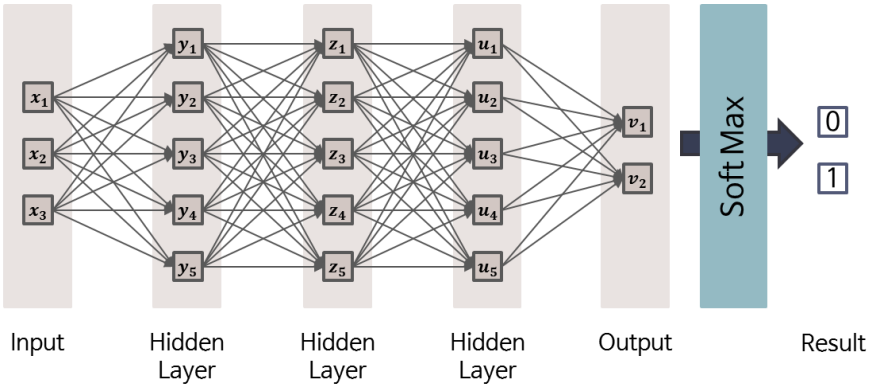
단계 3. $n = n + 1$ 으로 설정. $n < N$ 인 경우, 단계 1로 이동.

DNN의 매개 변수(Parameter)에 따른 성능 비교 (1/2)

- DNN 성능 실험 과정



- 매개 변수
  - 원 모델의 초기 노드 : L = 0 (사고 발생 직후 ($t = 0$), 긴급 환자 3명, 응급 환자 9명)
  - 부분 모델의 초기 노드 : L = 10 (사고 발생 16분 후 ($t = 16$), 긴급 환자 1명, 응급 환자 6명)
  - Max epoch : Training set 전체를 몇 번 학습할 것인가
  - Number of layers
  - Number of nodes
  - Variance of initial weights



DNN의 매개 변수(Parameter)에 따른 성능 비교 (2/2)

- 실험 결과
  - 초기 weight 값의 분산(variance)이 작은 경우 성능이 좋지 않음
  - Layer를 구성하는 노드의 수가 많으면 training set 결과는 좋지만 test set 결과는 좋지 않음
    → Input layer의 노드 수가 35개로 필요 이상의 노드로 학습 시 과대 적합(overfitting) 효과

Parameter	1	2	3	4	5	6
Max epoch	5	5	5	5	20	5
Num. of layer	3	3	3	3	3	4
Num. of nodes	30 30 50	30 30 50	30 30 50	60 60 100	30 30 50	30 30 50 50
Variance of initial weights	0.05	0.1	0.2	0.2	0.2	0.2
%-misclassified (training)	11.41	11.40	11.39	11.31	11.40	11.39
%-misclassified (testing)	11.48	11.46	11.46	12.25	11.47	11.48
p-value (Training)	0.06	<0.01	-	0.08	<0.01	<0.01
p-value (Testing)	0.07	<0.01	-	0.21	0.08	0.06

Reference

[1] W. B. Powell. Approximate Dynamic Programming: Solving the curses of dimensionality. John Wiley &Sons, 2007.

[2] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A survey of monte carlo tree search methods. IEEE Transactions on Computational Intelligence and AI in Games, 4(1):1-43, 2012.

[3] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. Nature, 529(7587):484-489, 2016.

[4] M. L. Puterman. Markov decision processes: discrete stochastic dynamic programming. John Wiley & Sons, 2014.

[5] 신교홍, 이태식, 근사적 동적계획법을 이용한 다중손상사고 환자 이송 우선순위와 이송병원 결정, 2015 춘계공동학술대회, April, 2015.