Basic of POMDPs

Bio-Medical Computing Laboratory

TAEHEUM CHO

20th Aug, 2018

INDEX

Who is this Markov guy?

- Andrey Andreyevich Markov (1856-1922)
- Russian mathematician
- Known for his work in stochastic processes
  - Later known as Markov Chains

Markov process (MP)

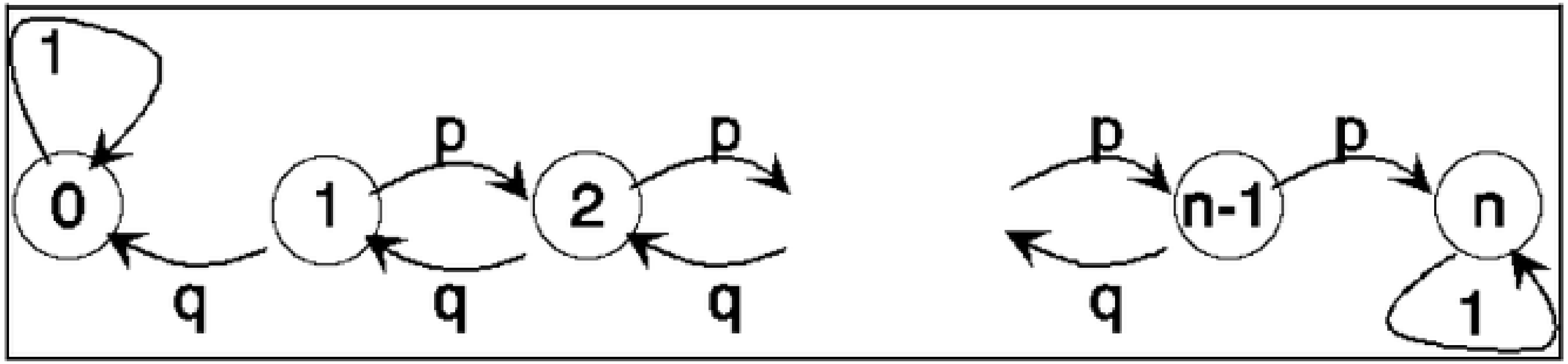
- Markov published his first paper on ‘Markov processes’ in 1906.
  - "Rasprostranenie zakona bol'shih chisel na velichiny, zavisyaschie drug ot druga".
- there is no definitive agreement in the literature on the use of terms that signify Markov processes.
  - Usually the term "Markov process" is reserved for a process with a discrete set of times, i.e. a discrete-time Markov chain (DTMC).
- A Markov chain is "a stochastic model that follows Markov properties.
- Markov property
  - predictions for the future of the process is based solely on its present state
  - conditional on the present state of the system, its future and past states are independent.

Markov process (MP)

- An example of Markov chain
  - If you have eaten cheese today, tomorrow will eat lettuce and grapes at the same probability.
  - If you have eaten the grapes today, tomorrow will eat 1/10 of the grapes, 4/10 of the cheese, and 5/10 of the lettuce.
  - If you have eaten lettuce today, eat grapes 4/10 and cheese 6/10.
  - tomorrow's food is not be influenced by yesterday's food!

Markov process (MP)

- An example of modeling an environment using the Markov process
  - gambler's ruin problem



n = The amount of money you can have
Pj = The probability of getting n from j

Markov process (MP)

$P_0=0;\ \ P_n=1; \qquad P_1=pP_2;\ \ P_{n-1}=p+qP_{n-2}; \qquad \therefore\ P_j=pP_{j+1}+qP_{j-1}\ \ (1<j<n-1)$

$\mathit{since}\ \ p+q=1,\ \ (p+q)P_j=pP_{j+1}+qP_{j-1}\ \ \mathit{or}\ \ p(P_{j+1}-P_j)=q(p_j-P_{j-1})\ \ \therefore\ (P_{j+1}-P_j)=\rho(P_j-P_{j-1})\ \ [\rho=q/p]$

$$P_2 - P_1 = \rho(P_1 - P_0) = \rho P_1 \qquad P_2 = P_1(1+\rho).$$

$$P_3 - P_2 = \rho(P_2 - P_1) = \rho^2 P_1 \qquad P_3 = P_2 + \rho^2 P_1 = P_1(1+\rho+\rho^2).$$

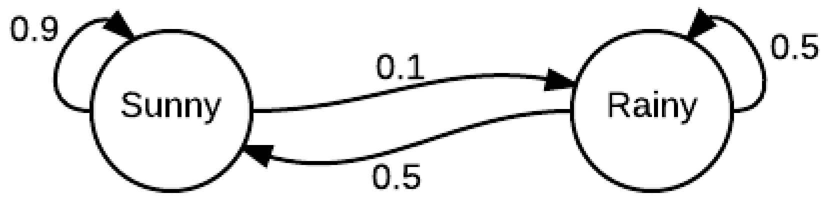
$$P_4 - P_3 = \rho(P_3 - P_2) = \rho^3 P_1 \qquad \text{for } j \geq 1,\ P_j = P_1(1+\rho+\rho^2+\rho^3+\ldots+\rho^{j-1}).$$

$$\vdots \qquad P_j = \left(\frac{1-\rho^j}{1-\rho^n}\right); \quad \rho = \frac{q}{p} \neq 1$$

$$P_n - P_{n-1} = \rho(P_{n-1} - P_{n-2}) = \rho^{n-1} P_1 \qquad P_j = \frac{j}{n}; \quad \rho = \frac{q}{p} = 1$$

Markov process (MP)

- The components of Markov process
  - X = A set of finite state space
  - P_{ij} = transition probability (from state i to state j)



$MP\ (X, P)$

- The transition of each state occurs at discrete time.
- The time when staying in any arbitrary state belonging to the state set X is defined as a step.
- The transition probability to the next state j from state i is always the same regardless of which state it has visited before.

$$p_{ij} = p(X_{n+1} = j \mid X_n = i)$$

Markov decision processes (MDP)

- The components of MDP
  - S is a finite set of states
  - A is a finite set of action
  - P is the probability that action a in state s at time t will lead state s' at time $t+1$
  - R is the immediate reward received after transitioning from state s to state s', due to action a
  - Γ is the discount factor

Markov decision processes (MDP)

- MDP: discrete time stochastic control process

  MP (Markov Process) : $p(s'|s)$
  MDP (Markov Decision Process) : $p(s'|s,a)$

- Reward

  $R(x, a)$

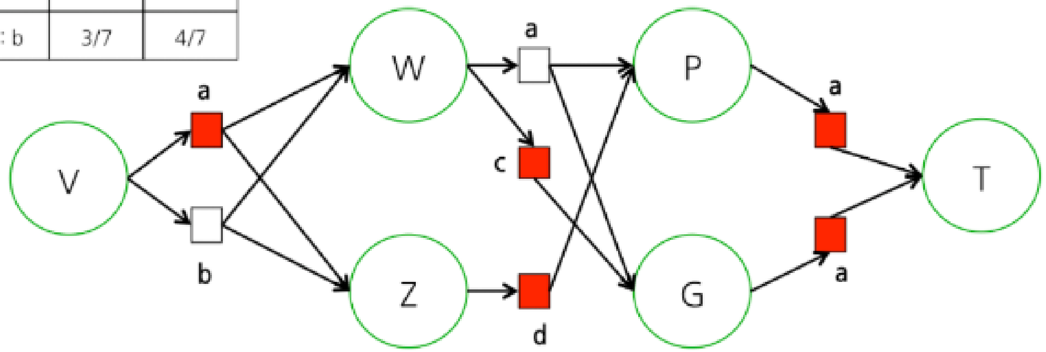
- Policy: 'probability function for action' mapped to state

  $\pi = \{\pi_t \mid \pi_t ; X \rightarrow A(X), t = \{0, 1, \ldots, H-1\}\}$

Markov decision processes (MDP)

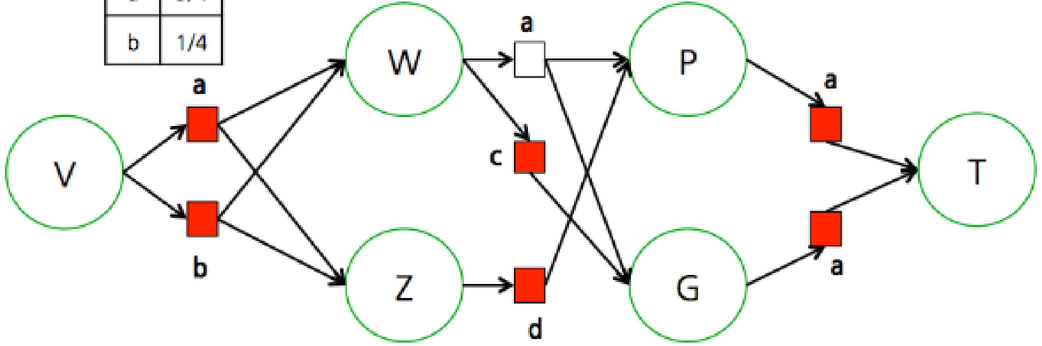
- Policy
  - Deterministic policy
  - Stochastic policy

Node V	p(w|v)	p(z|v)
action : a	1/4	3/4
action : b	3/7	4/7



a	3/4
b	1/4

a	0
c	1



$$\pi = \{\pi_0(v) = \{a, b\}, \pi_1(w) = c, \pi_1(z) = d, \pi_2(P) = a, \pi_2(G) = a\}$$

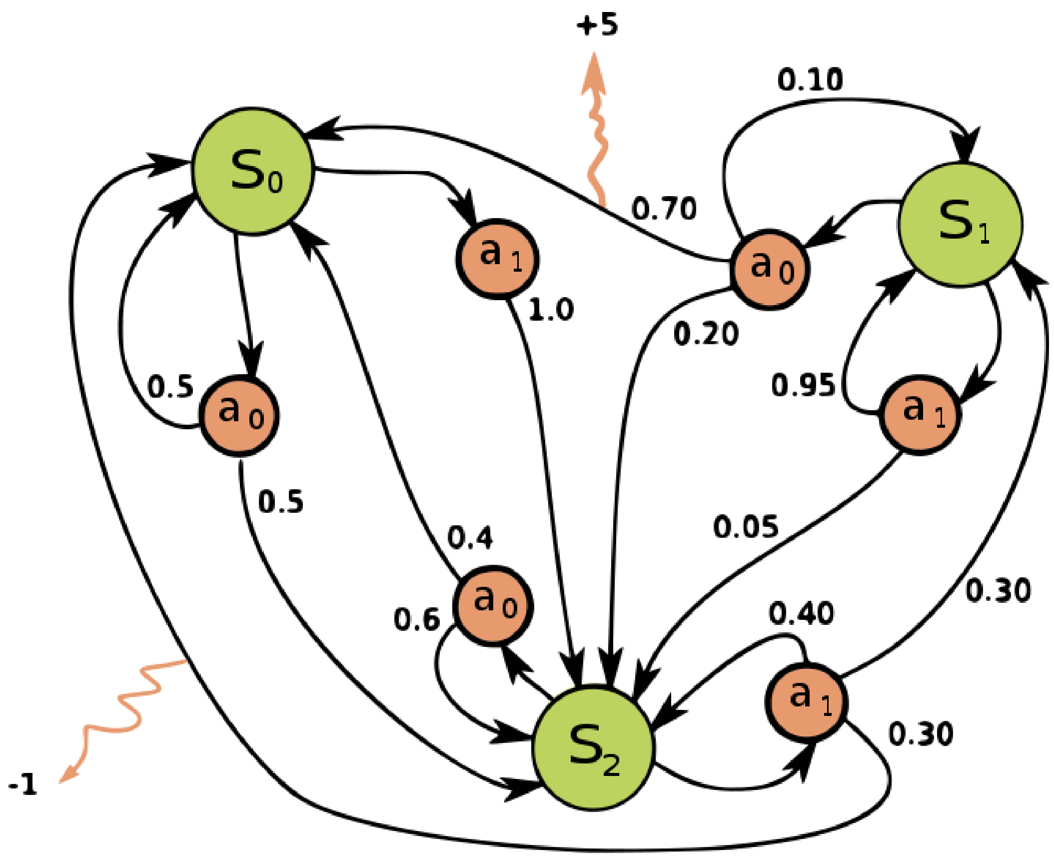
Markov decision processes (MDP)

- Vaule function
  - The total sum of the reward obtained after selecting any policy and proceeding with the process
  - What we want is to find an optimal policy that maximizes reward.

$$V_H^{\pi^*}(x) = \max_\pi V_H^\pi(x)$$

  - transition probability carried out by the policy π may be different each time it is repeated. Because, transition is represented by probability function even if the same action in selected.
    - Use the expectation value as an evaluation function (value function)

Markov decision processes (MDP)



partially observable Markov decision process (POMDP)

- The components of POMDP

S is a set of states,

A is a set of actions,

T is a set of conditional transition probabilities between states,

$R : S \times A \to \mathbb{R}$ is the reward function.

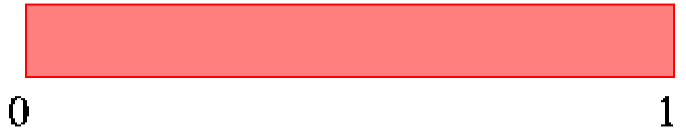
Ω is a set of observations,

O is a set of conditional observation probabilities, and

$\gamma \in [0, 1]$ is the discount factor.

partially observable Markov decision process (POMDP)

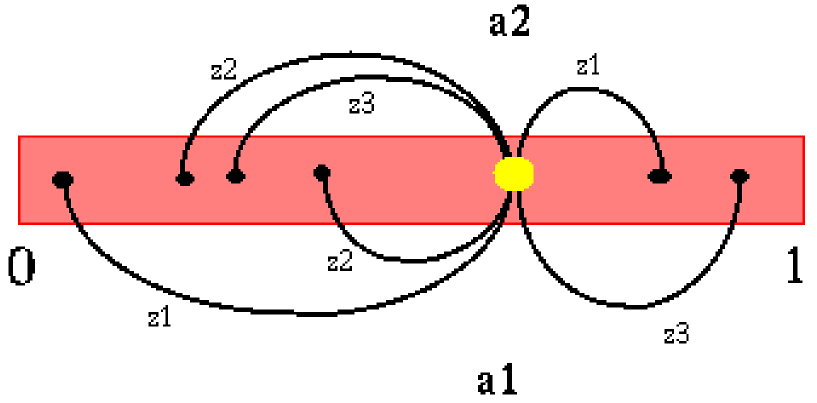
- in POMDPs our problem is to find a mapping from probability distributions (over states) to actions.
- We will refer to a probability distribution over states as a **belief state** and the entire probability space as the **belief space**.
  - a belief state is a probability distribution, the sum of all probabilities must sum to 1
  - The belief space is labeled with a 0 on the left and a 1 on the right.
  - This is the probability we are in state s1.



1D belief space for a 2 state POMDP

partially observable Markov decision process (POMDP)

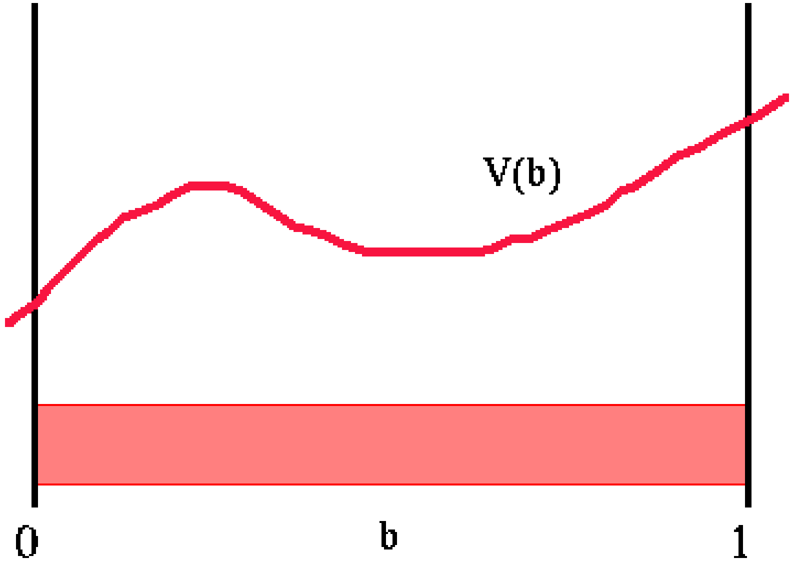
- Assume we start with a particular belief state b and we take action a1 and receive observation z1 after taking that action.
- The figure below shows this process graphically for a POMDP with two states (s1 and s2), two actions (a1 and a2) and three observations (z1, z2 and z3).
- Since observations are probabilistic, each resulting belief state has a probability associated with it.
  - if we take an action and get an observation, then we know with certainty what our next belief state is.



1D belief space for a 2 state POMDP

partially observable Markov decision process (POMDP)

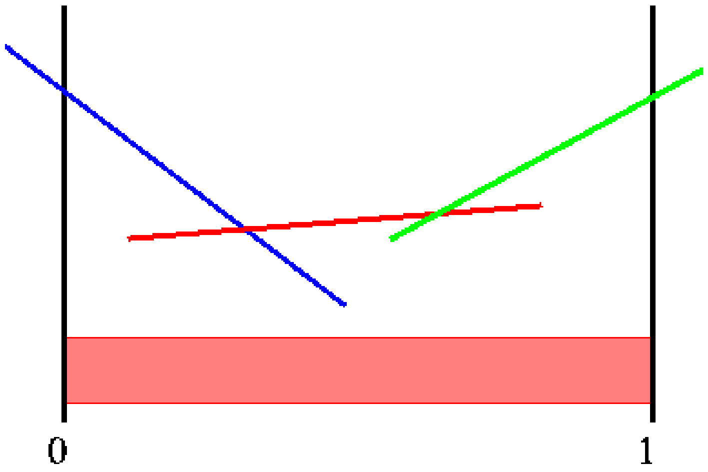
- the next belief state depends only on the current belief state (and the current action and observation).
- In fact, we can convert a discrete POMDP problem into a continuous space CO-MDP problem where the continuous space is the belief space.
  - The figure below shows a sample value function over belief space.



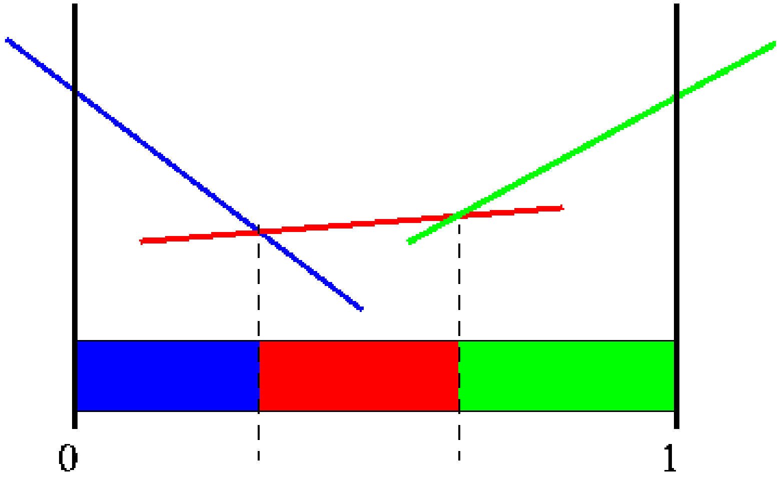
Value function over belief space

partially observable Markov decision process (POMDP)

- The key insight is that the finite horizon value function is piecewise linear and convex (PWLC) for every horizon length.
- This means that for each iteration of value iteration, we only need to find a finite number of linear segments that make up the value function.
- The vertical axis is the value, while the horizontal axis is the belief state.
- These linear segments will completely specify the value function (over belief space) that we desire.



Sample PWLC value function



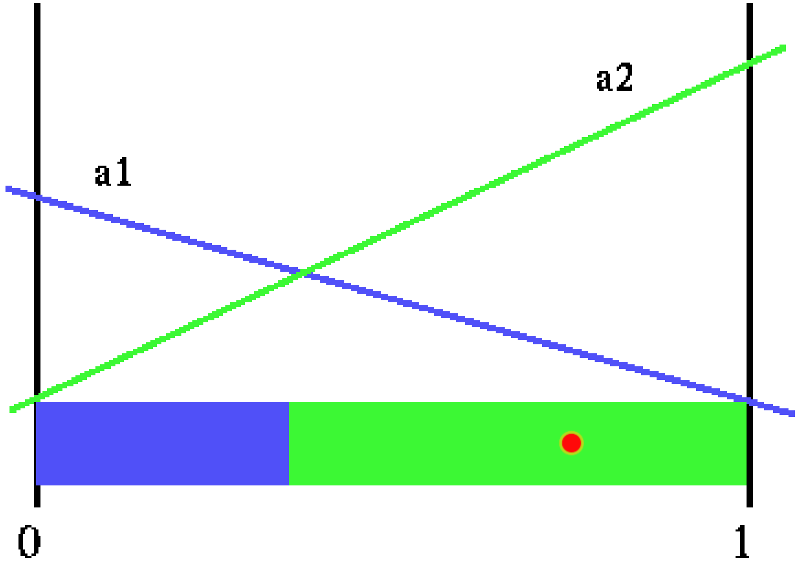
Sample PWLC function and its partition of belief space

partially observable Markov decision process (POMDP)

- Unfortunately, the continuous space causes us further problems.
  - for continuous state CO-MDPs it is impossible to enumerate all possible states

partially observable Markov decision process (POMDP)

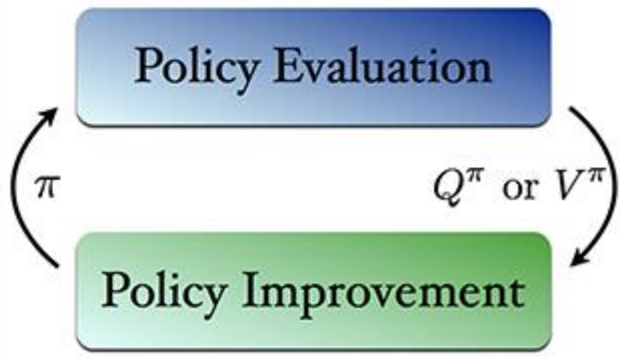
- belief state is [0.75 0.25]
- the value of doing action a1 in this belief state is 0.75 x 0 + 0.25 x 1 = 0.25. Similarly, action a2 has value 0.75 x 1.5 + 0.25 x 0 = 1.125.



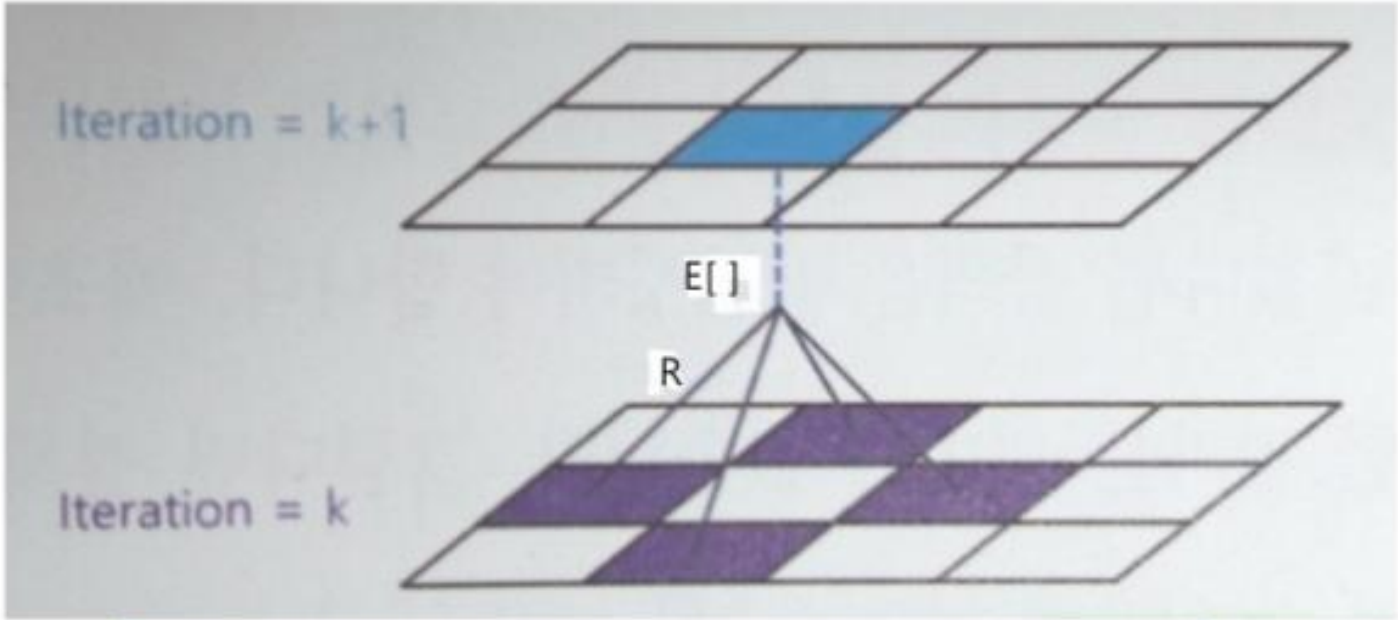
Horizon 1 value function

POMDP value iteration

- policy iteration – dynamic programming
- policy: information of how do an agent act in all the state.
  - stochastic
  - deterministic
- Therefore, Goal is that finding some policy which can obtain highest reward
- To obtain the optimal policy, we need to use policy evaluation and policy improvement.



POMDP value iteration



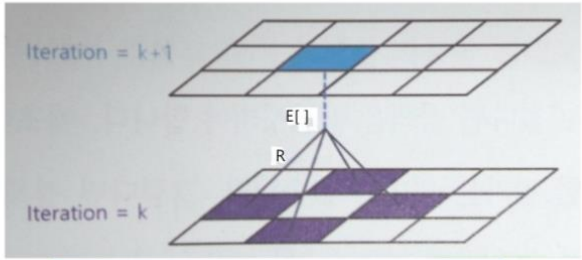
[파이썬과 케라스로 배우는 강화학습] p.71 그림 3.8

$$v_\pi(s) = \mathbb{E}_\pi\left[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s\right]$$

Bellman expectation equation

POMDP value iteration

- process of policy evaluation
  - 1. Fetch a value functions $v_k(s')$ which is stored in the next state s' that can be obtained from the current state s.
  - 2. Multiply $v_k(s')$ by the discount factor γ and adds reward R_s^a for the action going to that state.
  - 3. Multiply policy value $\pi(a|s)$.
  - 4. Repeat for all possible actions and add all the values.
  - 5. Store that value (k+1)th into state s location.
  - 6. Repeat avobe all process.



[파이썬과 케라스로 배우는 강화학습] p.71 그림 3.8

Bellman expectation equation:

$$V_\pi(s) = \sum_{s'} P(s'|s;\pi(s))[R(s,\pi(s),s') + \gamma V_\pi(s')], \forall s$$

POMDP value iteration

- policy improvement
  - policy improvement is that updating policy based on policy evaluation.
  - Just choose the biggest one!

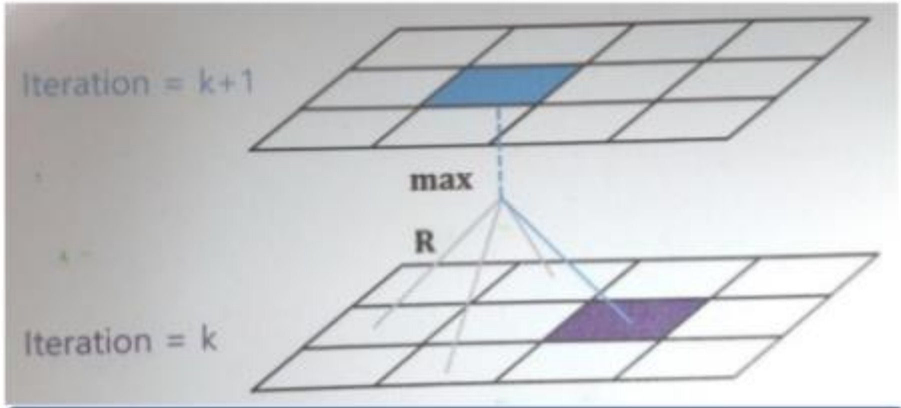
$$\pi'(s) = \arg\max_{a} Q^{\pi}(s,a)$$

POMDP value iteration

- Value iteration is that updating value function assuming optimal policy.

$$v_*(s) = \max_a E[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a].$$

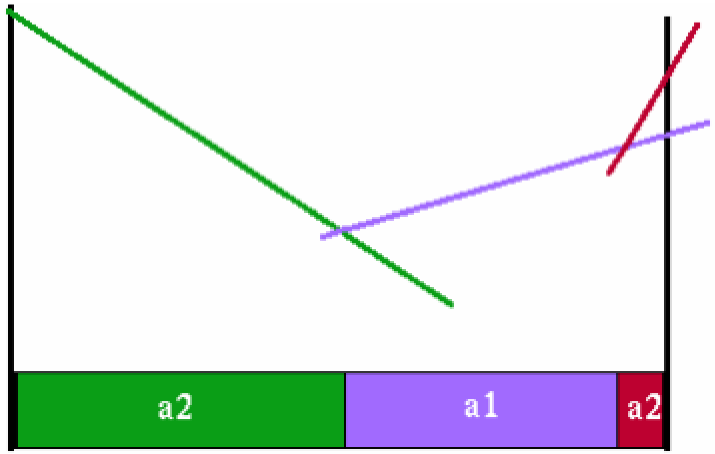
- The difference from policy iteration is that it updates only the highest value, not the whole state.



[파이썬과 케라스로 배우는 강화학습] p.95 그림 3.18

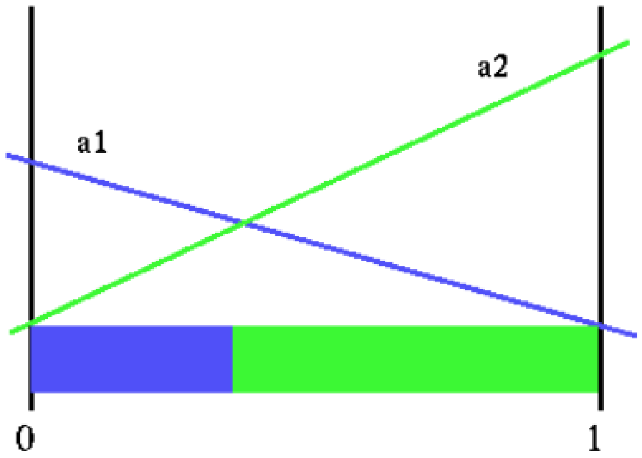
POMDP value iteration

- Value Iteration for POMDPs
  - The value function of POMDPs can be represented as max of linear segments
  - This is piecewise-linear-convex (PWLC)
    - State is known at edges of belief space
    - Can always do better with more knowledge of state



POMDP value iteration

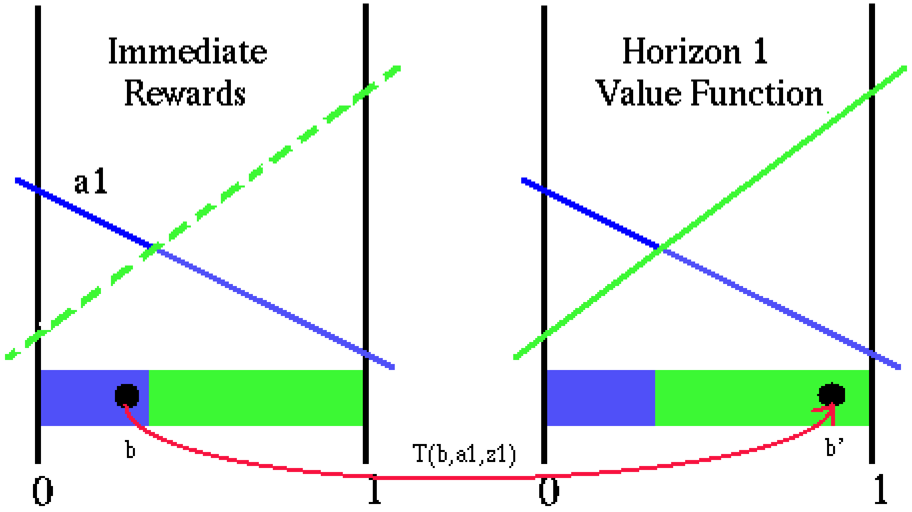
- Example POMDP for value iteration
  - Two states: s1, s2
  - Two actions: a1, a2
  - Three observations: z1, z2, z3
  - Positive rewards in both states: R(s1) = 1.0, R(s2) = 1.5



Horizon 1 value function

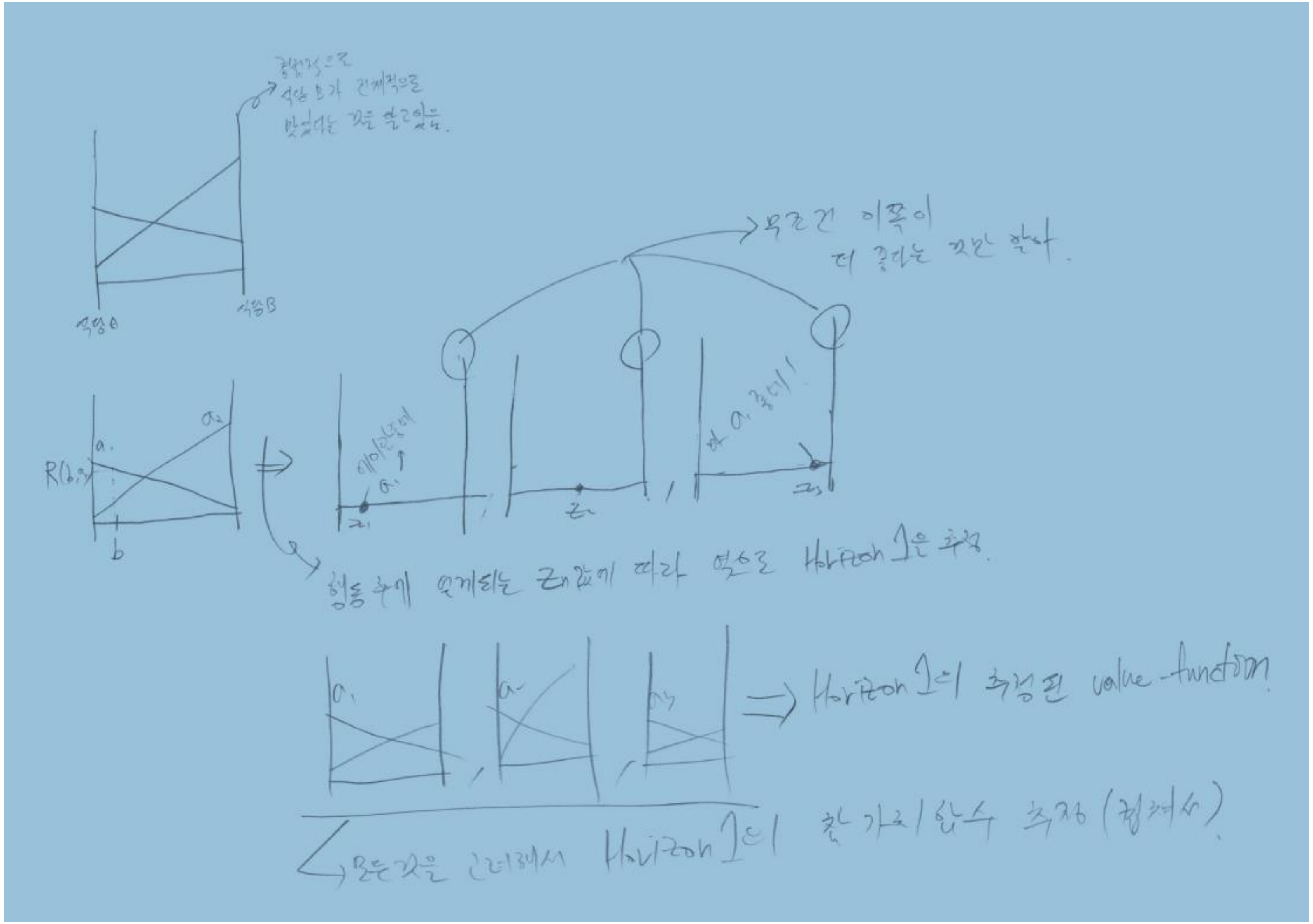
POMDP value iteration

- Horizon 1 value function
  - Calculate immediate rewards for each action in belief space



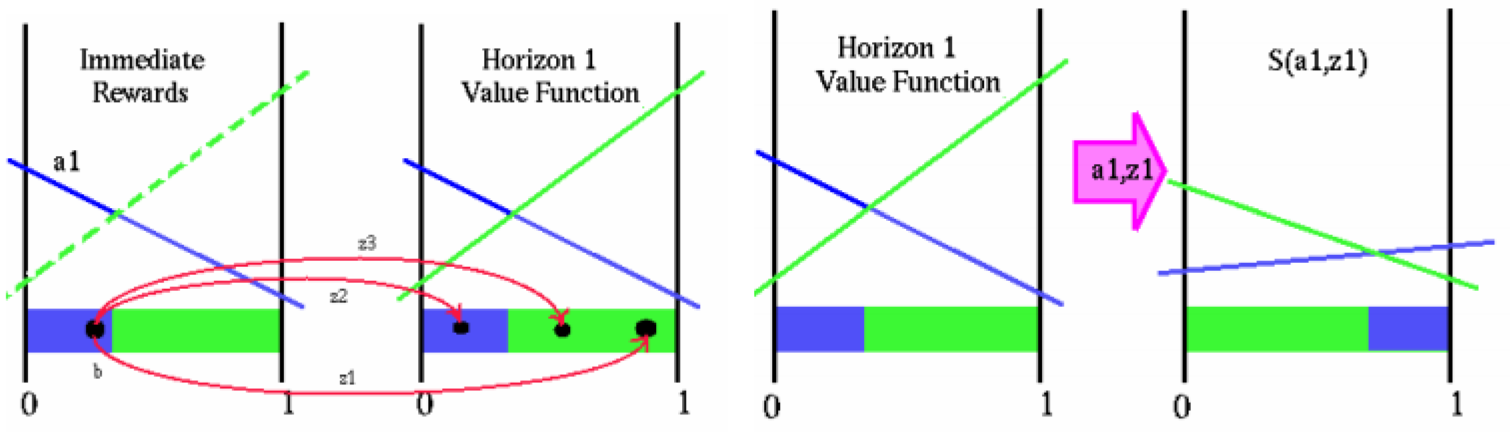
Value of a fixed action and observation

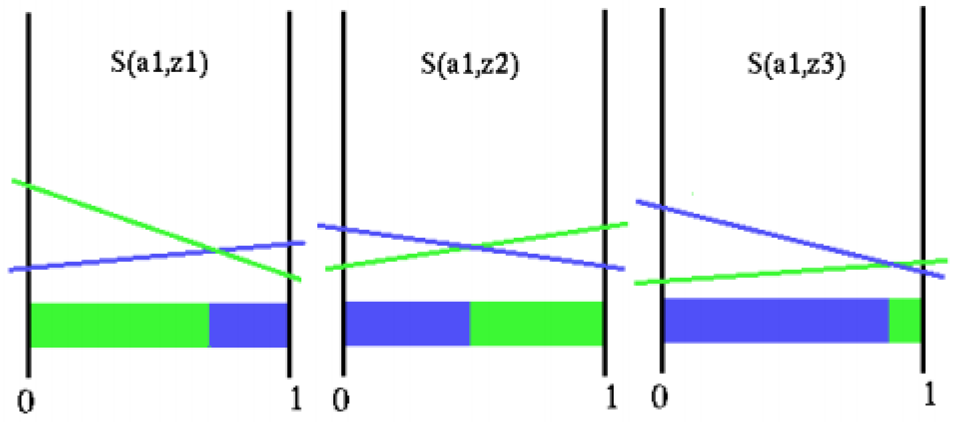
POMDP value iteration



POMDP value iteration

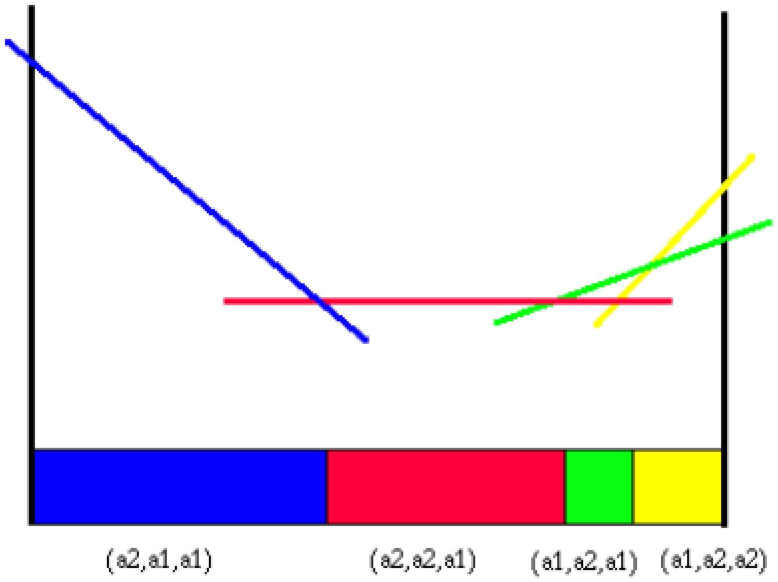
- Need to transform value function with observations



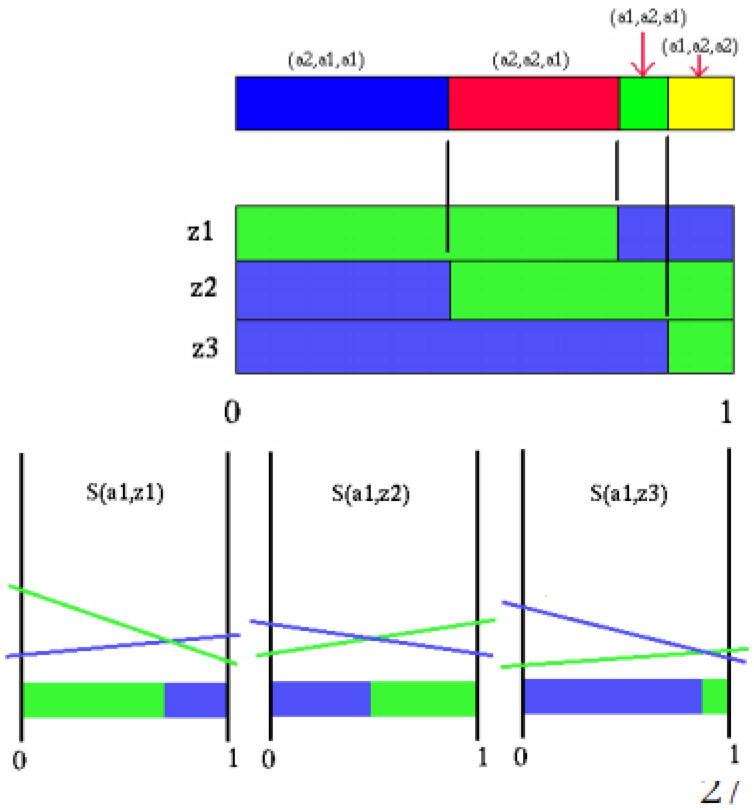


POMDP value iteration

- Each action from horizon 1 yields new vectors from the transformed space

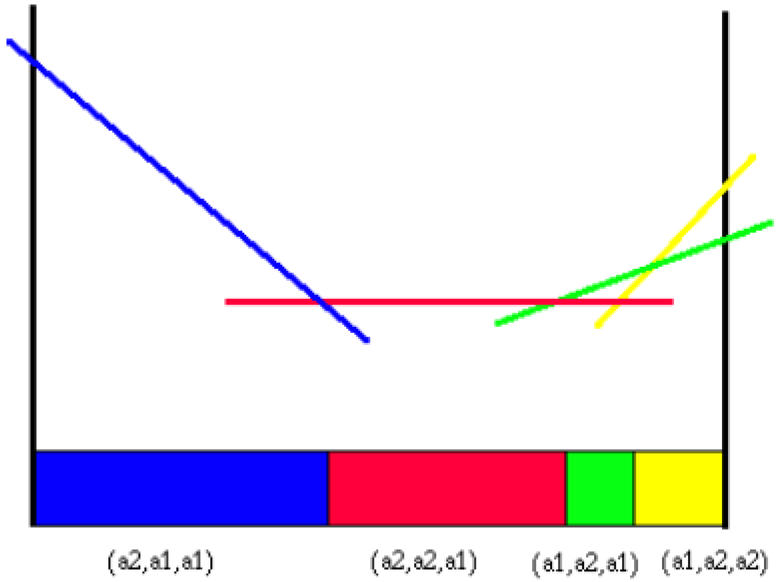


Value function and partition for taking action a1 in step 1

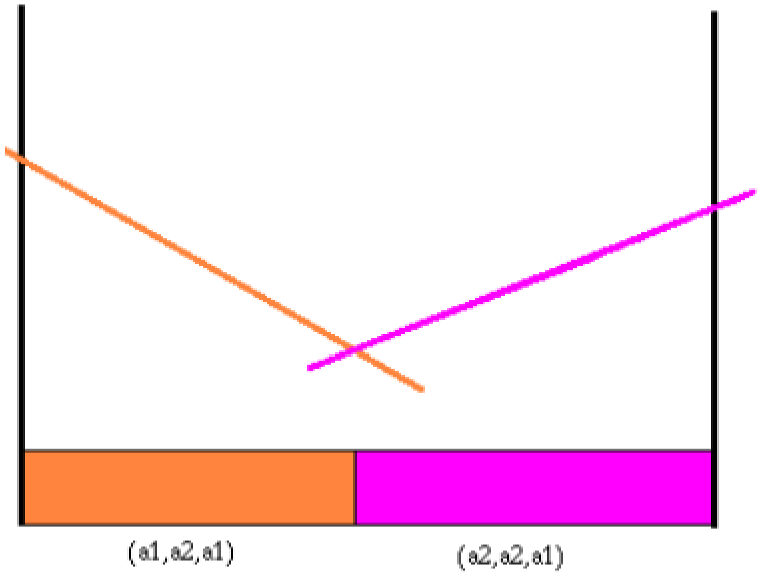


POMDP value iteration

- Each action from horizon 1 yields new vectors from the transformed space



Value function and partition for taking action a1 in step 1

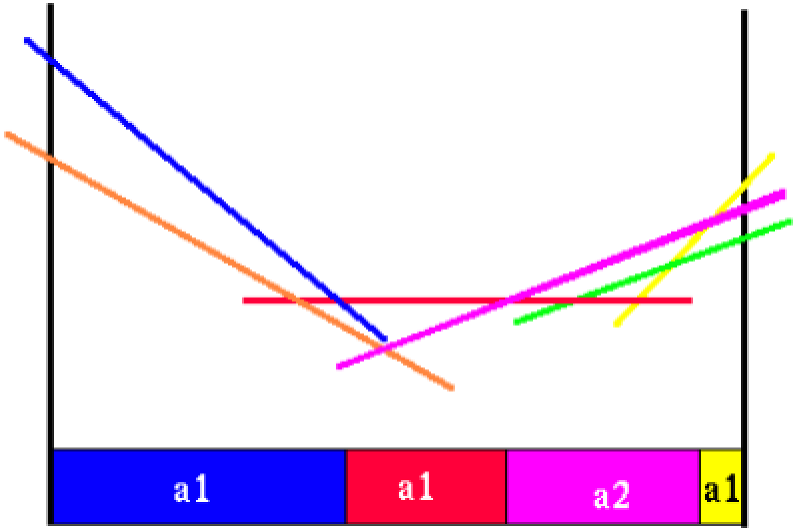


Value function and partition for taking action a2 in step 1

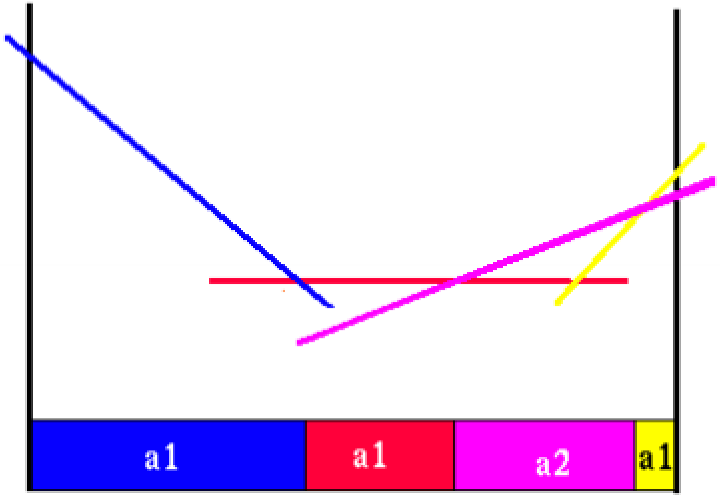
28

POMDP value iteration

- Combine vectors to yield horizon 2 value function (can also prune dominated vectors)

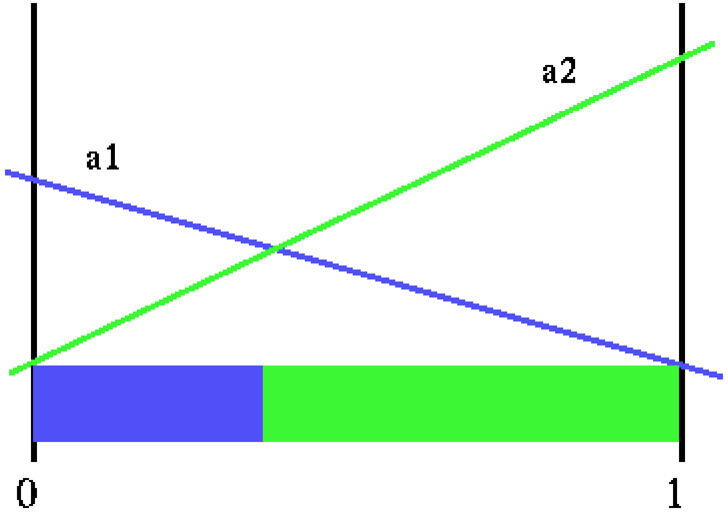


Combined a1 and a2 value functions

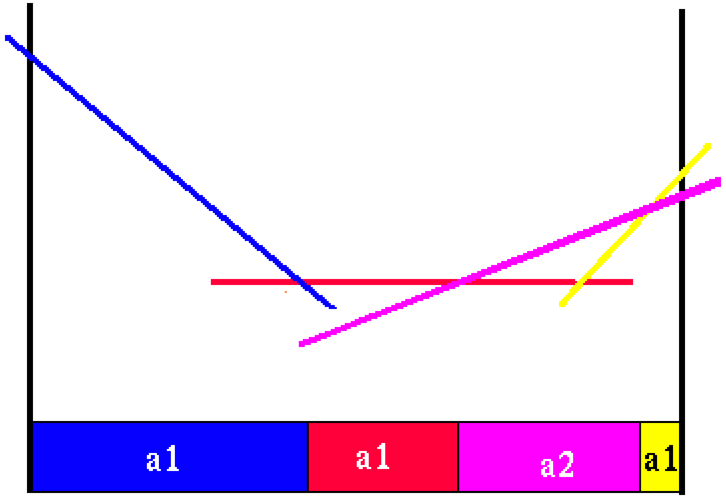


Horizon 2 value function with pruning

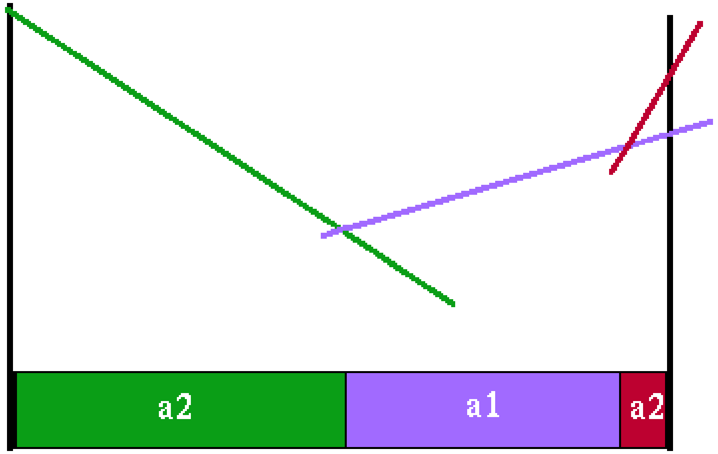
POMDP value iteration



Horizon 1 value function



Value function for horizon 2



Value function for horizon 3

POMDP value iteration

- Bellman optimality equation for POMDPs

$$V^*(b) = \max_{a \in A}\left[\sum_{s \in S} b(s) R(s,a) + \gamma \sum_{o \in O} \Pr(o \mid b,a) V^*(b_o^a)\right]$$

$$=> \; V^*(b) = \max_a (R(b,a) + \gamma \Sigma\, Pr(z|b,a) \cdot V_{t-1}(b'))$$