VI extensions

6. května 2019

B4M36PUI/BE4M36PUI — Planning for Artificial Intelligence

Outline

- Review of MDP concepts
- Value Iteration algorithm
- VI extensions

Review of last tutorial

Value function of a policy

Look at the following definition of a value function of a policy for inifnite-horizon MDP. It contains multiple mistakes, correct them on a piece of paper:

Def: Value function of a policy for infinite-horizon MDP

Assume infinite horizon MDP with $\gamma \in [0, 100]$. Then let Value function of a policy π for every state $s \in S$ be defined as

$$V^{\pi}(s) = \sum_{s' \in S} R(s, \pi(s), s') R(s, \pi(s), s') + \gamma \pi(s')$$

Value function of a policy

Look at the following definition of a value function of a policy for inifnite-horizon MDP. It contains multiple mistakes, correct them on a piece of paper:

Def: Value function of a policy for infinite-horizon MDP

Assume infinite horizon MDP with $\gamma \in [0, 100]$. Then let Value function of a policy π for every state $s \in S$ be defined as

$$V^{\pi}(s) = \sum_{s' \in S} R(s, \pi(s), s') R(s, \pi(s), s') + \gamma \pi(s')$$

$$\gamma \in [0, 1)$$

$$V^{\pi}(s) = \sum_{s' \in S} T(s, \pi(s), s')[R(s, \pi(s), s') + \gamma V^{\pi}(s')]$$

Value function of a policy

Look at the following definition of a value function of a policy for inifnite-horizon MDP. It contains multiple mistakes, correct them on a piece of paper:

Def: Value function of a policy for infinite-horizon MDP

Assume infinite horizon MDP with $\gamma \in [0, 100]$. Then let Value function of a policy π for every state $s \in S$ be defined as

$$V^{\pi}(s) = \sum_{s' \in S} R(s, \pi(s), s') R(s, \pi(s), s') + \gamma \pi(s')$$

$$\gamma \in [0, 1)$$

$$V^{\pi}(s) = \sum_{s' \in S} T(s, \pi(s), s')[R(s, \pi(s), s') + \gamma V^{\pi}(s')]$$

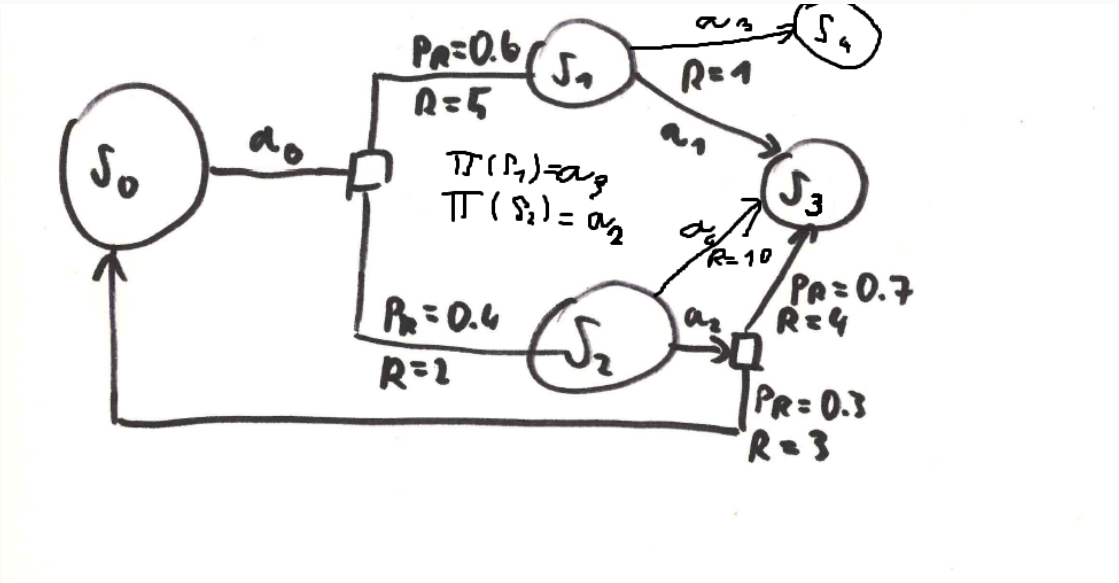
Question: Difference to def. of an optimal value function?

Bellman Equations

Write down equations for finding a value function of a policy π. How would you solve these equations?

- T : $T(S_0, a_0, S_1) = 0.6$
  $T(S_0, a_0, S_2) = 0.4$
  $T(S_1, a_1, S_3) = 1$
  $T(S_2, a_2, S_3) = 0.7$
  $T(S_2, a_2, S_0) = 0.3$
- R : $R(S_0, a_0, S_1) = 5$
  $R(S_0, a_0, S_2) = 2$
  $R(S_1, a_1, S_3) = 1$
  $R(S_2, a_2, S_3) = 4$
  $R(S_2, a_2, S_0) = 3$
  $R(S_1, a_3, S_4) = 4$
  $R(S_2, a_4, S_3) = 10$

- S: S_0, S_1, S_2, S_3, S_4
- A: a_0, a_1, a_2, a_3, a_4)



Value Iteration

value Iteration algorithm

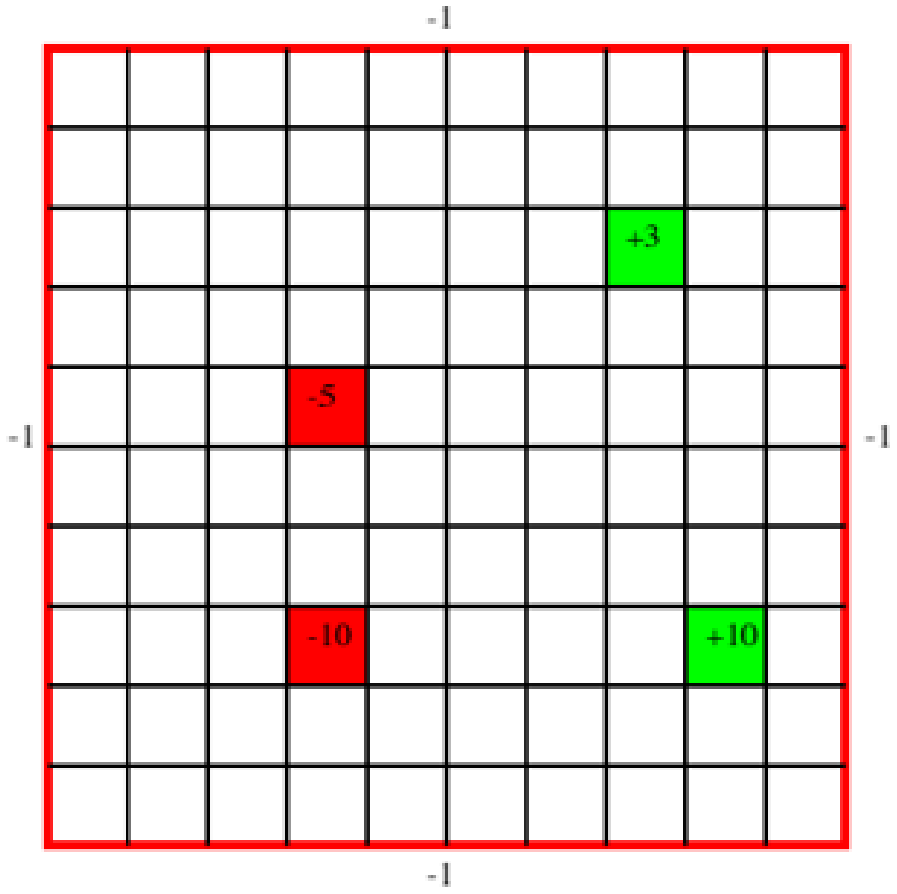
Basic algorithm for finding solution of Bellman Equations iteratively.

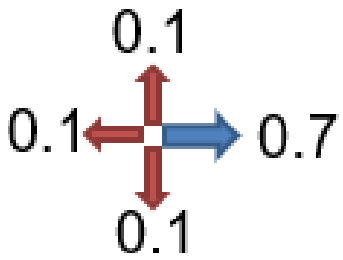
1. initialize V_0 arbitrarily for each state, e.g to 0, set $n = 0$
2. Set $n = n + 1$.
3. Compute Bellman Backup, i.e. for each $s \in S$:
   3.1 $V_n(s) = \max_{a \in A} \sum_{s' \in S} T(s, a, s')[R(s, a, s') + \gamma V_{n-1}(s')]$
4. GOTO 2.

VI example

Robot Emil-like domain.
Assuming $\gamma = 1$.
Using VI, initializing
$\forall s \; V_0(s) = 0$, calculate
V_1, V_2, V_3 for the nine states
around $+10$ tile.

- Moving into edges gives -1 reward
- Moving onto marked tiles gives corresponding reward





From http://artint.info/html/ArtInt_224.html#gridworld-ex

value Iteration algorithm

Basic algorithm for finding solution of Bellman Equations iteratively.

1. initialize V_0 arbitrarily for each state, e.g to 0, set $n = 0$
2. Set $n = n + 1$.
3. Compute Bellman Backup, i.e. for each $s \in S$:
   3.1 $V_n(s) = \max_{a \in A} \sum_{s' \in S} T(s, a, s')[R(s, a, s') + \gamma V_{n-1}(s')]$
4. GOTO 2.

Question: Does it converge? How fast? When do we stop?

Residual - definition

Def: Residual

Residual of value function V_n from V_{n+1} at state $s \in S$ is defined by:

$$Res^{V_n}(s) = |V_n(s) - V_{n+1}(s)|$$

Residual - definition

Def: Residual

Residual of value function V_n from V_{n+1} at state $s \in S$ is defined by:

$$Res^{V_n}(s) = |V_n(s) - V_{n+1}(s)|$$

Residual of value function V from V' is given by:

$$Res^{V_n} = ||V_n - V_{n+1}||_\infty = \max_s |V_n(s) - V_{n+1}(s)|$$

VI stopping criterion

Stopping criterion: When residual of consecutive value functions is below low value of ϵ:

$$||V_n - V_{n+1}|| < \epsilon$$

However, this does not imply ϵ distance of value of greedy policy from optimal value function.

👁 for proof, see: [Singh, Yee: An Upper Bound on the Loss from Approximate Optimal-Value Functions, p.229]

VI stopping criterion

Stopping criterion: When residual of consecutive value functions is below low value of ϵ:

$$||V_n - V_{n+1}|| < \epsilon$$

However, this does not imply ϵ distance of value of greedy policy from optimal value function.

Theorems for general MDP exist of form:

$$V_n, V^* \text{as above} \implies \forall s \; |V_n(s) - V^*(s)| < \epsilon(\text{Some MDP dependent term})$$

for proof, see: [Singh, Yee: An Upper Bound on the Loss from Approximate Optimal-Value Functions, p.229]

VI stopping criterion

Stopping criterion: When residual of consecutive value functions is below low value of ϵ:

$$||V_n - V_{n+1}|| < \epsilon$$

However, this does not imply ϵ distance of value of greedy policy from optimal value function.

Theorems for general MDP exist of form:

$$V_n, V^* \text{as above} \implies \forall s \; |V_n(s) - V^*(s)| < \epsilon(\text{Some MDP dependent term})$$

In case of discounted ($\gamma < 1$) infinite-horizon MDPs:

$$V_n, V^* \text{as above} \implies \forall s \; |V_n(s) - V^*(s)| < 2\frac{\epsilon\gamma}{1-\gamma}$$

for proof, see: [Singh, Yee: An Upper Bound on the Loss from Approximate Optimal-Value Functions, p.229]

VI with stopping criterion

1. initialize V_0 arbitrarily for each state, e.g to 0, set $n = 0$
2. Set $n = n + 1$.
3. Compute Bellman Backup, i.e. for each $s \in S$:
   - 3.1 $V_n(s) = \max_{a \in A} \sum_{s' \in S} T(s, a, s')[R(s, a, s') + \gamma V_{n-1}(s')]$
   - 3.2 Calculate residual $Res = \max_{s \in S} |V_n(s) - V_{n-1}(s)|$
4. if $res > \epsilon$ GOTO 2. else TERMINATE

VI with stopping criterion

1. initialize V_0 arbitrarily for each state, e.g to 0, set $n = 0$
2. Set $n = n + 1$.
3. Compute Bellman Backup, i.e. for each $s \in S$:
   3.1 $V_n(s) = \max_{a \in A} \sum_{s' \in S} T(s, a, s')[R(s, a, s') + \gamma V_{n-1}(s')]$
   3.2 Calculate residual $Res = \max_{s \in S} |V_n(s) - V_{n-1}(s)|$
4. if $res > \epsilon$ GOTO 2. else TERMINATE

Question: What is the policy?

VI with stopping criterion

1. initialize V_0 arbitrarily for each state, e.g to 0, set $n = 0$
2. Set $n = n + 1$.
3. Compute Bellman Backup, i.e. for each $s \in S$:
   3.1 $V_n(s) = \max_{a \in A} \sum_{s' \in S} T(s, a, s')[R(s, a, s') + \gamma V_{n-1}(s')]$
   3.2 Calculate residual $Res = \max_{s \in S} |V_n(s) - V_{n-1}(s)|$
4. if $res > \epsilon$ GOTO 2. else TERMINATE

Question: What is the policy?

- *Greedy policy* π_n^V is the policy given as argmax of V_n.

Value iteration properties

- Convergence: VI converges from any initialization (unlike PI)
- Termination: when residual is "small"

Value iteration properties

- Convergence: VI converges from any initialization (unlike PI)
- Termination: when residual is "small"

Question: What are the memory requirements of VI?

Value iteration properties

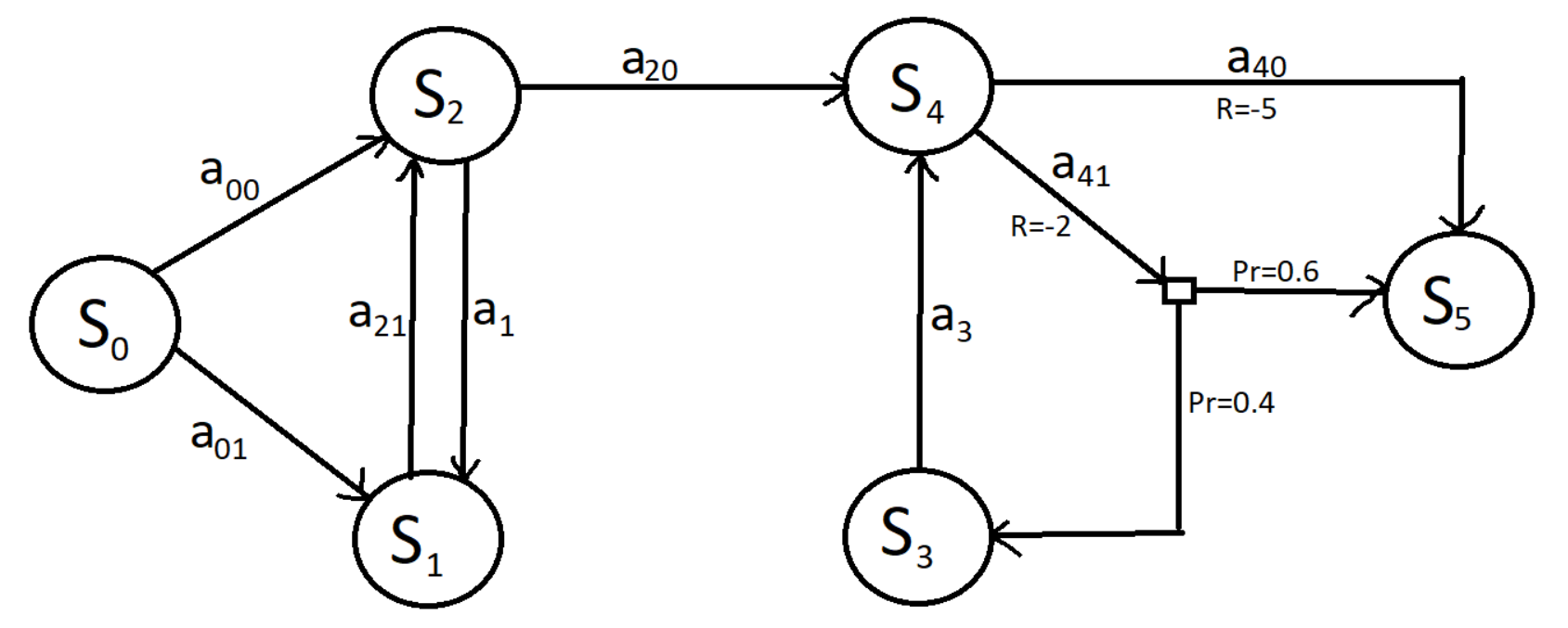
- Convergence: VI converges from any initialization (unlike PI)
- Termination: when residual is "small"

Question: What are the memory requirements of VI?

- Value of each state needs to be stored twice

VI extensions

Another beautiful MDP example



- All undeclared rewards are -1

Task: Initialize VI with negative distance to S_5 and calculate first 3 iterations of VI, with state ordering S_0 to S_5

Asynchronous VI

1. initialize V_0 arbitrarily for each state, e.g to 0
2. While $Res^V > \epsilon$, do:
   - 2.1 pick some state s
   - 2.2 Bellman backup $V(s) \leftarrow \max_{a \in A} \sum_{s' \in S} T(s, a, s')[R(s, a, s') + \gamma V(s')]$
   - 2.3 Update residual at s $Res^V(s) = |V_{\text{old}}(s) - V_{\text{new}}(s)|$

Asynchronous VI

1. initialize V_0 arbitrarily for each state, e.g to 0
2. While $Res^V > \epsilon$, do:
   - 2.1 pick some state s
   - 2.2 Bellman backup $V(s) \leftarrow \max_{a \in A} \sum_{s' \in S} T(s, a, s')[R(s, a, s') + \gamma V(s')]$
   - 2.3 Update residual at s $Res^V(s) = |V_{\text{old}}(s) - V_{\text{new}}(s)|$

Question: Memory requirements compared to VI?

Asynchronous VI

1. initialize V_0 arbitrarily for each state, e.g to 0
2. While $Res^V > \epsilon$, do:
   - 2.1 pick some state s
   - 2.2 Bellman backup $V(s) \leftarrow \max_{a \in A} \sum_{s' \in S} T(s, a, s')[R(s, a, s') + \gamma V(s')]$
   - 2.3 Update residual at s $Res^V(s) = |V_{\text{old}}(s) - V_{\text{new}}(s)|$

Question: Memory requirements compared to VI?

Question: Convergence condition?

- Asymptotic as VI under condition that every state visited ∞ often.

Asynchronous VI

1. initialize V_0 arbitrarily for each state, e.g to 0
2. While $Res^V > \epsilon$, do:
   - 2.1 pick some state s
   - 2.2 Bellman backup $V(s) \leftarrow \max_{a \in A} \sum_{s' \in S} T(s, a, s')[R(s, a, s') + \gamma V(s')]$
   - 2.3 Update residual at s $Res^V(s) = |V_{\text{old}}(s) - V_{\text{new}}(s)|$

Question: Memory requirements compared to VI?

Question: Convergence condition?

- Asymptotic as VI under condition that every state visited ∞ often.

Question: How to pick s in 2.1?

- Simplest is *Gauss-Seidel VI*, that is run AVI over all states iteratively

Prioritized VI

How else can we pick states to update? (Ans: by ordering them in clever ways)

- Build priority queue of states to update $\rightarrow$ *Prioritized Sweeping VI*. Update states in the order of the queue. Priority function:

$$\text{priority}_{PS}(s) \leftarrow \max\{\text{priority}_{PS}(s), \max_{a \in A}\{T(s,a,s')Res^{V}(s')\}\}$$

Prioritized VI

How else can we pick states to update? (Ans: by ordering them in clever ways)

- Build priority queue of states to update $\rightarrow$ *Prioritized Sweeping VI*. Update states in the order of the queue. Priority function:

$$\text{priority}_{PS}(s) \leftarrow \max\{\text{priority}_{PS}(s), \max_{a \in A}\{T(s,a,s')Res^{V}(s')\}\}$$

EXAMPLE ON BOARD

Prioritized VI

How else can we pick states to update? (Ans: by ordering them in clever ways)

- Build priority queue of states to update $\rightarrow$ *Prioritized Sweeping VI*. Update states in the order of the queue. Priority function:

$$\text{priority}_{PS}(s) \leftarrow \max\{\text{priority}_{PS}(s), \max_{a \in A}\{T(s,a,s')Res^{V}(s')\}\}$$

EXAMPLE ON BOARD

Convergence?

Prioritized VI

How else can we pick states to update? (Ans: by ordering them in clever ways)

- Build priority queue of states to update $\rightarrow$ *Prioritized Sweeping VI*. Update states in the order of the queue. Priority function:

$$\text{priority}_{PS}(s) \leftarrow \max\{\text{priority}_{PS}(s), \max_{a \in A}\{T(s,a,s')Res^{V}(s')\}\}$$

EXAMPLE ON BOARD

Convergence?

- If all states start with non-zero priority

Prioritized VI

How else can we pick states to update? (Ans: by ordering them in clever ways)

- Build priority queue of states to update $\rightarrow$ *Prioritized Sweeping VI*. Update states in the order of the queue. Priority function:

$$\text{priority}_{PS}(s) \leftarrow \max\{\text{priority}_{PS}(s), \max_{a \in A}\{T(s,a,s')Res^{V}(s')\}\}$$

EXAMPLE ON BOARD

Convergence?

- If all states start with non-zero priority
- OR If you interleave regular VI sweeps with Prioritized VI

Partitioned VI

Subdivide state space into partitions. Pre-solve parts independently.

- Example is *Topological VI*

Partitioned VI

Subdivide state space into partitions. Pre-solve parts independently.

- Example is *Topological VI*

Topological VI:

1. Find acyclic paritioning (by finding strongly connected components in the graph)
2. Run VI in each partition to convergence backward from terminal states

Partitioned VI

Subdivide state space into partitions. Pre-solve parts independently.

- Example is *Topological VI*

Topological VI:

1. Find acyclic paritioning (by finding strongly connected components in the graph)
2. Run VI in each partition to convergence backward from terminal states

---

Question: Why acyclic partitioning?

Partitioned VI

Subdivide state space into partitions. Pre-solve parts independently.

- Example is *Topological VI*

Topological VI:

1. Find acyclic paritioning (by finding strongly connected components in the graph)
2. Run VI in each partition to convergence backward from terminal states

---

Question: Why acyclic partitioning?

EXAMPLE ON BOARD