Value Iteration

and extensions

Jan Mrkos

PUI Tutorial
Week 9

Outline

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

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

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

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

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

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 $s \in S$:
    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.

Value Iteration algorithm

Basic algorithm for finding solution of Bellman Equations iteratively:

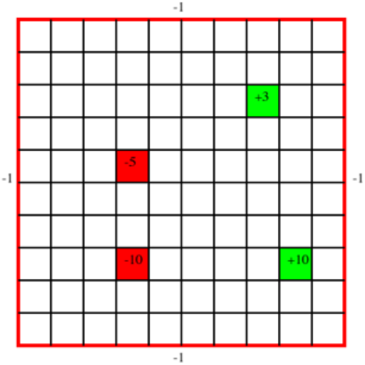
1. initialize V_0 arbitrarily for each state, e.g to 0, set $n = 0$
2. Set $n = n + 1$.
3. Compute Bellman Backup, i.e. for each $s \in S$:
    1. $V_n(s) = \max_{a \in A} \sum_{s' \in S} T(s, a, s')[R(s, a, s') + \gamma V_{n-1}(s')]$
4. GOTO 2.

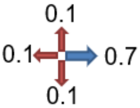
Basic version uses 2 arrays to store state values.

VI example - gridworld

Gridworld domain. Assuming $\gamma = 1$.
Using VI and initializing $\forall s \; V_0(s) = 0$, calculate V_1, V_2, V_3 for the nine states around the +10 tile.
Domain rules:

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





Example from https://artint.info

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 $s \in S$:
    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.

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:

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

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:

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

VI with stopping criterion

Algorithm 1: Value Iteration

```
initialize V_0 arbitrarily for each state, e.g to 0 while Res^V > ϵ do
    pick some state s
    Bellman backup V(s) ← max_{a∈A} ∑_{s'∈S} T(s, a, s')[R(s, a, s') + γV(s')]
    Update residual at s Res^V(s) = |V_old(s) − V_new(s)|
return greedy policy π^V;
```

VI with stopping criterion

Algorithm 2: Value Iteration

initialize V_0 arbitrarily for each state, e.g to 0 **while** $Res^V > \epsilon$ **do**
    pick some state s
    Bellman backup $V(s) \leftarrow \max_{a \in A} \sum_{s' \in S} T(s,a,s')[R(s,a,s') + \gamma V(s')]$
    Update residual at s $Res^V(s) = |V_{\text{old}}(s) - V_{\text{new}}(s)|$
return greedy policy π^V;

Question: What is the greedy policy?

VI with stopping criterion

Algorithm 3: Value Iteration

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

return greedy policy π^V;

Question: What is the greedy 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?

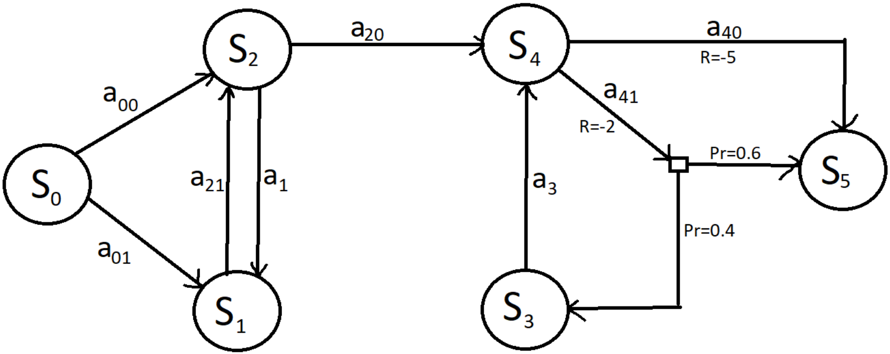
Value iteration properties

- 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

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

Algorithm 4: Asynchronous VI

initialize V_0 arbitrarily for each state, e.g to 0 **while** $Res^V > \epsilon$ **do**
    pick some state s
    Bellman backup $V(s) \leftarrow \max_{a \in A} \sum_{s' \in S} T(s,a,s')[R(s,a,s') + \gamma V(s')]$
    Update residual at s $Res^V(s) = |V_{\text{old}}(s) - V_{\text{new}}(s)|$
return greedy policy π^V;

Asynchronous VI

Algorithm 5: Asynchronous VI

initialize V_0 arbitrarily for each state, e.g to 0 **while** $Res^V > \epsilon$ **do**

- pick some state s
- Bellman backup $V(s) \leftarrow \max_{a \in A} \sum_{s' \in S} T(s,a,s')[R(s,a,s') + \gamma V(s')]$
- Update residual at s $Res^V(s) = |V_{\text{old}}(s) - V_{\text{new}}(s)|$

return greedy policy π^V;

Question: Memory requirements compared to VI?

Asynchronous VI

Algorithm 6: Asynchronous VI

initialize V_0 arbitrarily for each state, e.g to 0 **while** $Res^V > \epsilon$ **do**

  pick some state s

  Bellman backup $V(s) \leftarrow \max_{a \in A} \sum_{s' \in S} T(s,a,s')[R(s,a,s') + \gamma V(s')]$

  Update residual at s $Res^V(s) = |V_{\text{old}}(s) - V_{\text{new}}(s)|$

return greedy policy π^V;

Question: Memory requirements compared to VI?
Question: Convergence condition?

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

Asynchronous VI

Algorithm 7: Asynchronous VI

1 initialize V_0 arbitrarily for each state, e.g to 0 **while** $Res^V > \epsilon$ **do**
2   pick some state s
3   Bellman backup $V(s) \leftarrow \max_{a \in A} \sum_{s' \in S} T(s,a,s')[R(s,a,s') + \gamma V(s')]$
4   Update residual at s $Res^V(s) = |V_{\text{old}}(s) - V_{\text{new}}(s)|$
5 **return** greedy policy π^V;

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 → *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 → *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 → *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

Thank you for participating in the tutorials :-)

Please fill out the feedback form →

https://forms.gle/wyVaeHbXJYEKdUNX7