Prediction and Control by Dynamic Programing

CS60077: Reinforcement Learning

Abir Das

IIT Kharagpur

Sep 21, 22 and 28, 2020

Agenda

§ Understand how to evaluate policies using dynamic programing based methods

§ Understand policy iteration and value iteration algorithms for control of MDPs

§ Existence and convergence of solutions obtained by the above methods

Resources

- Reinforcement Learning by David Silver [Link]
- Reinforcement Learning by Balaraman Ravindran [Link]
- SB: Chapter 4

Dynamic Programing

"*Life can only be understood going backwards, but it must be lived going forwards.*"
- S. Kierkegaard, Danish Philosopher.

The first line of the famous book by Dimitri P Bertsekas.

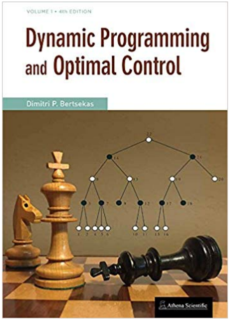


Image taken from: *amazon.com*

Dynamic Programing

- Dynamic Programing [DP] in this course, refer to a collection of algorithms that can be used to compute optimal policies given a perfect model of the environment in a MDP.
- Limited utility due to the 'perfect model' assumption and due to computational expense.
- But still are important as they provide essential foundation for many of the subsequent methods.
- Many of the methods can be viewed as attempts to achieve much the same effect as DP with less computation and without perfect model assumption of the environment.
- The key idea in DP is to use the value functions and Bellman equations to organize and structure the search for good policies.

Dynamic Programing

§ Dynamic Programing addresses a bigger problem by breaking it down as subproblems and then

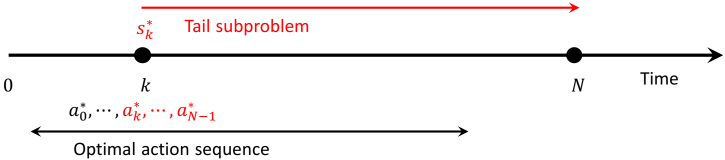
- Solving the subproblems
- Combining solutions to subproblems

Dynamic Programing

§ Dynamic Programing addresses a bigger problem by breaking it down as subproblems and then

- Solving the subproblems
- Combining solutions to subproblems

§ Dynamic Programing is based on the principle of optimality.



Principle of Optimality

Let $\{a_0^*, a_1^*, \cdots, a_{(N-1)}^*\}$ be an optimal action sequence with a corresponding state sequence $\{s_1^*, s_2^*, \cdots, s_N^*\}$. Consider the tail subproblem that starts at s_k^* at time k and maximizes the 'reward to go' from k to N over $\{a_k, \cdots, a_{(N-1)}\}$, then the tail optimal action sequence $\{a_k^*, \cdots, a_{(N-1)}^*\}$ is optimal for the tail subproblem.

Requirements for Dynamic Programing

- Optimal substructure *i.e.*, principle of optimality applies.
- Overlapping subproblems, *i.e.*, subproblems recur many times and solutions to these subproblems can be cached and reused.
- MDPs satisfy both through Bellman equations and value functions.
- Dynamic programming is used to solve many other problems, *e.g.*, Scheduling algorithms, Graph algorithms (e.g. shortest path algorithms), Bioinformatics etc.

Planning by Dynamic Programing

§ Planning by dynamic programing assumes full knowledge of the MDP

§ For *prediction/evaluation*

- Input: MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$ and policy π
- Output: Value function v_π

Planning by Dynamic Programing

- Planning by dynamic programing assumes full knowledge of the MDP
- For *prediction/evaluation*
  - Input: MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$ and policy π
  - Output: Value function v_π
- For *control*
  - Input: MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$
  - Output: Optimal value function v_* and optimal policy π_*

Iterative Policy Evaluation

- Problem: Policy evaluation: Compute the state-value function v_π for an arbitrary policy π.
- Solution strategy: Iterative application of Bellman expectation equation.
- Recall the Bellman expectation equation.

$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left\{ r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) v_\pi(s') \right\} \tag{1}$$

- Consider a sequence of approximate value functions $v^{(0)}, v^{(1)}, v^{(2)}, \cdots$ each mapping $\mathcal{S}^+$ to $\mathbb{R}$. Each successive approximation is obtained by using eqn. (1) as an update rule.

$$v^{(k+1)}(s) \leftarrow \sum_{a \in \mathcal{A}} \pi(a|s) \left\{ r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) v^{(k)}(s') \right\}$$

Iterative Policy Evaluation

$$v^{(k+1)}(s) \leftarrow \sum_{a \in \mathcal{A}} \pi(a|s) \left\{ r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) v^{(k)}(s') \right\}$$

- In code, this can be implemented by using two arrays - one for the old values $v^{(k)}(s)$ and the other for the new values $v^{(k+1)}(s)$. Here, new values of $v^{(k+1)}(s)$ are computed one by one from the old values $v^{(k)}(s)$ without changing the old values.
- Another way is to use one array and update the values 'in place', *i.e.*, each new value immediately overwriting the old one.
- Both these converges to the true value v_π and the 'in place' algorithm usually converges faster.

Iterative Policy Evaluation

Iterative Policy Evaluation, for estimating $V \approx v_\pi$

Input: π, the policy to be evaluated
Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(\text{terminal}) = 0$
Loop:
  $\Delta \leftarrow 0$
  Loop for each $s \in \mathcal{S}$:
    $v \leftarrow V(s)$
    $V(s) \leftarrow \sum_{a \in \mathcal{A}} \pi(a|s) \left\{ r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a)V(s') \right\}$
    $\Delta \leftarrow \max \left(\Delta, |v - V(s)| \right)$
until $\Delta < \theta$

Evaluating a Random Policy in the Small Gridworld

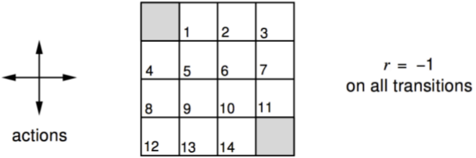


Figure credit: *[SB] chapter 4*

- Undiscounted episodic MDP $(\lambda = 1)$
- Non-terminal states are $\mathcal{S} = \{1, 2, \cdots, 14\}$
- Two terminal states (shown as shaded squares)
- 4 possible actions in each state, $\mathcal{A} = \{up, down, right, left\}$
- Deterministic state transitions
- Actions leading out of the grid leave state unchanged
- Reward is -1 until the terminal state is reached
- Agent follows uniform random policy $\pi(n|.) = \pi(s|.) = \pi(e|.) = \pi(w|.)$

Evaluating a Random Policy in the Small Gridworld

v_k for the Random Policy

$k = 0$

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

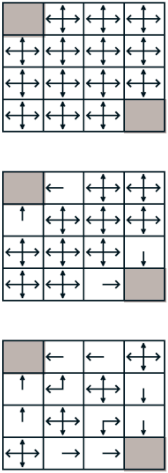
$k = 1$

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0

$k = 2$

0.0	-1.7	-2.0	-2.0
-1.7	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.7
-2.0	-2.0	-1.7	0.0

Greedy Policy w.r.t. v_k



random policy

Figure credit: *[SB] chapter 4*

Evaluating a Random Policy in the Small Gridworld

$k = 3$

0.0	-2.4	-2.9	-3.0
-2.4	-2.9	-3.0	-2.9
-2.9	-3.0	-2.9	-2.4
-3.0	-2.9	-2.4	0.0

$k = 10$

0.0	-6.1	-8.4	-9.0
-6.1	-7.7	-8.4	-8.4
-8.4	-8.4	-7.7	-6.1
-9.0	-8.4	-6.1	0.0

$k = \infty$

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0

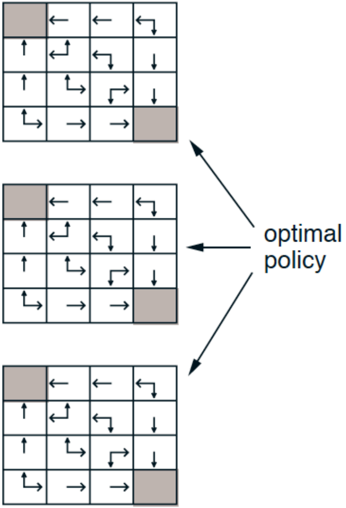


Figure credit: *[SB] chapter 4*

Improving a Policy: Policy Iteration

- Given a policy π
  - Evaluate the policy

$$v_\pi \doteq v^{(k+1)}(s) \leftarrow \sum_{a\in\mathcal{A}} \pi(a|s) \left\{ r(s,a) + \gamma \sum_{s'\in\mathcal{S}} p(s'|s,a) v^{(k)}(s') \right\}$$

  - Improve the policy by acting greedily with respect to v_π

$$\pi' = \text{greedy}(v_\pi)$$

    being greedy means choosing the action that will land the agent into best state *i.e.*, $\pi'(s) \doteq \arg\max_{a\in\mathcal{A}} q_\pi(s,a) =$
    $\arg\max_{a\in\mathcal{A}} \left\{ r(s,a) + \gamma \sum_{s'\in\mathcal{S}} p(s'|s,a) v_\pi(s') \right\}$
- In Small Gridworld improved policy was optimal $\pi' = \pi_*$
- In general, need more iterations of improvement/evaluation
- But this process of policy iteration always converges to π_*

Improving a Policy: Policy Iteration

Given a policy π

§ Evaluate the policy

$$v_\pi \doteq v^{(k+1)}(s) \leftarrow \sum_{a\in\mathcal{A}} \pi(a|s)\left\{r(s,a) + \gamma \sum_{s'\in\mathcal{S}} p(s'|s,a)v^{(k)}(s')\right\}$$

$$= \underbrace{\sum_{a\in\mathcal{A}} \pi(a|s)r(s,a)}_{r_\pi(s)} + \gamma \sum_{s'\in\mathcal{S}} \underbrace{\sum_{a\in\mathcal{A}} \pi(a|s)p(s'|s,a)}_{p_\pi(s'|s)} v^{(k)}(s')$$

$$= r_\pi(s) + \gamma \sum_{s'\in\mathcal{S}} p_\pi(s'|s)v^{(k)}(s')$$

- $r_\pi(s)$ = one step expected reward for following policy π at state s.
- $p_\pi(s'|s)$ = one step transition probability under policy π.

§ Improve the policy by acting greedily with respect to v_π

$$\pi'(s) = \arg\max_{a\in\mathcal{A}} \left\{r(s,a) + \gamma \sum_{s'\in\mathcal{S}} p(s'|s,a)v_\pi(s')\right\}$$

Policy Iteration

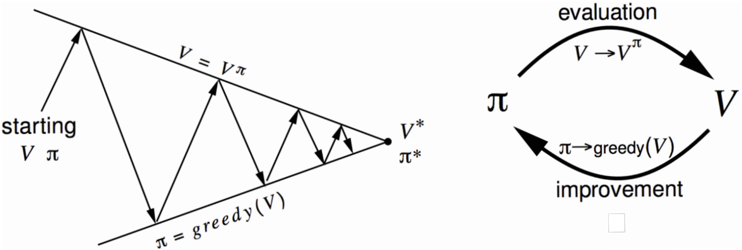


Figure credit: *[David Silver: DeepMind]*

- Policy Evaluation: Estimate v_π by iterative policy evaluation.
- Policy Improvement: Generate $\pi' \geq \pi$ by greedy policy improvement.

Policy Iteration

Algorithm 1: Policy iteration

```
1 initialization: Select π^0, n ← 0;
2 do
3     (Policy Evaluation) v_(π^{n+1}) ← r_(π^n) + γ P_{π^n} v_(π^n) ; // componentwise
4     (Policy Improvement)
        π^{n+1}(s) ∈ argmax_{a∈A} [r(s,a) + γ Σ_{s'∈S} p(s'|s,a) v_(π^{n+1})(s')] ∀s ∈ S;
5     n ← n + 1;
6 while π^{n+1} ≠ π^n;
7 Declare π* = π^n
```

1 initialization: Select π^0, $n \leftarrow 0$;

2 **do**

3 **(Policy Evaluation)** $v_{(\pi^{n+1})} \leftarrow r_{(\pi^n)} + \gamma \mathcal{P}_{\pi^n} v_{(\pi^n)}$; // componentwise

4 **(Policy Improvement)**

$$\pi^{n+1}(s) \in \arg\max_{a \in \mathcal{A}} [r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) v_{(\pi^{n+1})}(s')] \ \forall s \in \mathcal{S};$$

5 $n \leftarrow n + 1$;

6 **while** $\pi^{n+1} \neq \pi^n$;

7 Declare $\pi^* = \pi^n$

- why in step (4), $\in$ is used?
- Note the terminating condition.

Policy Iteration

- At each step of policy iteration, the policy improves *i.e.*, the value function for a policy at a later iteration is greater than or equal to the value function for a policy at an earlier step.
- This comes from the *policy improvement theorem* which (informally) is - Let π^n be some stationary policy and let π^{n+1} be greedy w.r.t. $v_{(\pi^n)}$, then $v_{(\pi^{n+1})} \geq v_{(\pi^n)}$, *i.e.*, π^{n+1} is an improvement upon π^n.

$$
\begin{aligned}
r_{\pi^{n+1}} + \gamma P_{\pi^{n+1}} v_{(\pi^n)} &\geq r_{\pi^n} + \gamma P_{\pi^n} v_{(\pi^n)} \\
&= v_{(\pi^n)} \text{ [Bellman eqn.]} \\
\implies r_{\pi^{n+1}} &\geq (I - \gamma P_{\pi^{n+1}}) v_{(\pi^n)} \\
\implies (I - \gamma P_{\pi^{n+1}})^{-1} r_{\pi^{n+1}} &\geq v_{(\pi^n)} \\
\implies v_{\pi^{n+1}} &\geq v_{(\pi^n)}
\end{aligned} \tag{2}
$$

- The first step: π^{n+1} is obtained by maximizing $r_\pi + \gamma P_\pi v_{(\pi^n)}$ over all π's. So, $r_{\pi^{n+1}} + \gamma P_{\pi^{n+1}} v_{(\pi^n)}$ will be better than any other π in $r_\pi + \gamma P_\pi v_{(\pi^n)}$. That 'any other π' happens to be π^n.

Policy Iteration: Example ([SB])

§ Jack manages two locations of a car rental company. At any location if car is available, he rents it out and gets \$10. To ensure that cars are available, Jack can move cars between the two locations overnight, at a cost of \$2 per car.

§ Cars are returned and requested randomly according to Poisson distribution. Probability that n cars are rented or returned is $\frac{\lambda^n}{n!}e^{-\lambda}$.

- 1st location - λ: average requests $=$ 3, average returns $=$ 3
- 2nd location - λ: average requests $=$ 4, average returns $=$ 2

§ there can be no more than 20 cars at each location and a maximum of 5 cars can be moved from one location to the other

Policy Iteration: Example - MDP Formulation

- **State**: number of cars at each location at the end of the day (between 0 and 20).
- **Actions**: number of cars moved overnight from one location to other (max 5).
- **Reward**: \$10 per car rented (if available) and -\$2 per car moved.
- **Transition probability**: The Poisson distribution defined in the last slide.
- **Discount factor**: γ is assumed to be 0.9.

Policy Iteration: Example

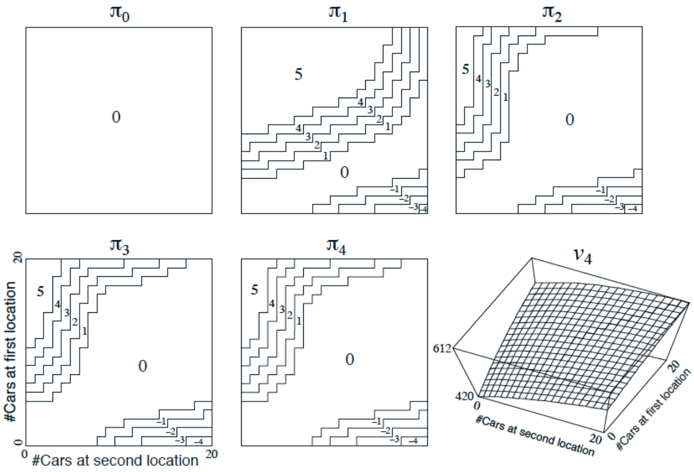


Figure credit: *[SB - Chapter 4]*

Figure: The sequence of policies found by policy iteration on Jack's car rental problem, and the final state-value function

Policy Iteration: Disadvantages

- § Policy iteration involves the policy evaluation step first and this itself requires a few iterations to get the exact value of v_π in limit.
- § The question is - must we wait for exact convegence to v_π? Or can we stop short of that?
- § The small gridworld example showed that there is no change of the greedy policy after the first three iterations.
- § So the question is - is there such a number of iterations such that after that the greedy policy does not change?

Value Iteration

§ A related question is - what about the extreme case of 1 iteration of policy evaluation and then greedy policy improvement? If we repeat this cycle, does it find the optimal policy at least in limit?

Value Iteration

- A related question is - what about the extreme case of 1 iteration of policy evaluation and then greedy policy improvement? If we repeat this cycle, does it find the optimal policy at least in limit?
- The good news is that - yes the gurantee is there and we will soon prove that. However, first let us modify the policy iteration algorithm to this extreme case. This is known as 'value iteration' strategy.

Value Iteration

- What policy iteration does: iterate over

$$v_\pi \doteq v^{(k+1)}(s) \leftarrow \sum_{a \in \mathcal{A}} \pi(a|s) \left\{ r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) v^{(k)}(s') \right\}$$

- And then

$$\pi'(s) = \arg\max_{a \in \mathcal{A}} \left\{ r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) v_\pi(s') \right\}$$

Value Iteration

- What policy iteration does: iterate over

$$v_\pi \doteq v^{(k+1)}(s) \leftarrow \sum_{a\in\mathcal{A}} \pi(a|s) \left\{ r(s,a) + \gamma \sum_{s'\in\mathcal{S}} p(s'|s,a) v^{(k)}(s') \right\}$$

- And then

$$\pi'(s) = \arg\max_{a\in\mathcal{A}} \left\{ r(s,a) + \gamma \sum_{s'\in\mathcal{S}} p(s'|s,a) v_\pi(s') \right\}$$

- What value iteration does: evaluate $\forall a \in \mathcal{A}$

$$r(s,a) + \gamma \sum_{s'\in\mathcal{S}} p(s'|s,a) v^{(k)}(s')$$

- And then take max over it

$$v^{(k+1)}(s) = \max_{a\in\mathcal{A}} \left\{ r(s,a) + \gamma \sum_{s'\in\mathcal{S}} p(s'|s,a) v^{(k)}(s') \right\}$$ Where have we seen it?

Value Iteration

Algorithm 2: Value iteration

initialization: $v \leftarrow v^0 \in \mathcal{V}$, pick an $\epsilon > 0$, $n \leftarrow 0$;
while $||v^{n+1} - v^n|| > \epsilon\frac{1-\gamma}{2\gamma}$ **do**
  foreach $s \in \mathcal{S}$ **do**
    $v^{n+1}(s) \leftarrow \max\limits_a \left\{ r(s,a) + \gamma \sum\limits_{s'} p(s'/s,a) v^n(s') \right\}$
  end
  $n \leftarrow n + 1$;
end
foreach $s \in \mathcal{S}$ **do**
  /* Note the use of $\pi(s)$. It mens deterministic policy */
  $\pi(s) \leftarrow \arg\max\limits_a \left\{ r(s,a) + \gamma \sum\limits_{s'} p(s'/s,a) v^n(s') \right\}$; // n has already been incremented by 1
end

§ Take a note of the stopping criterion

Summary of Exact DP Algorithms for Planning

Problem	**Bellman Equation**	**Algorithm**
Prediction	Bellman Expectation Equation	Iterative Policy Evaluation
Control	Bellman Expectation Equation + Greedy Policy Improvement	Policy Iteration
Control	Bellman Optimality Equation	Value Iteration

Norms

Definition

Given a vector space $\mathcal{V} \subseteq \mathbb{R}^d$, a function $f : \mathcal{V} \rightarrow \mathbb{R}^+$ is a norm (denoted as $||.||$) if and only if

- $||v|| \geq 0 \quad \forall v \in \mathcal{V}$
- $||v|| = 0$ if and only if $v = 0$
- $||\alpha v|| = |\alpha| \, ||v||$, $\forall \alpha \in \mathbb{R}$ and $\forall v \in \mathcal{V}$
- **Triangle inequality:** $||u + v|| \leq ||u|| + ||v|| \quad u, v \in \mathcal{V}$

Different types of Norms

§ L_p norm:

$$||v||_p = \left(\sum_{i=1}^{d} |v_i|^p\right)^{\frac{1}{p}}$$

§ L_0 norm:

$$||v||_0 = \text{Number of non-zero elements in } v$$

§ L_∞ norm:

$$||v||_\infty = \max_{1 \leq i \leq d} |v_i|$$

Cauchy Sequence, Completeness

Definition

A sequence of vectors $v_1, v_2, v_3, \dots \in \mathcal{V}$ (with subscripts $n \in \mathbb{N}$) is called a **Cauchy sequence** if for any positive real $\epsilon > 0, \exists\ N \in \mathbb{Z}^+$ such that $\forall m, n > N,\ ||v_m - v_n|| < \epsilon$.

- Basically, for any real positive ϵ, an element can be found in the sequence, beyond which any two elements of the sequence will be within ϵ of each other.
- In other words, the elements of the sequence comes closer and closer to each other - *i.e.*, the sequence converges.

Cauchy Sequence, Completeness

Definition

A sequence of vectors $v_1, v_2, v_3, \cdots \in \mathcal{V}$ (with subscripts $n \in \mathbb{N}$) is called a **Cauchy sequence** if for any positive real $\epsilon > 0, \exists\ N \in \mathbb{Z}^+$ such that $\forall m, n > N,\ ||v_m - v_n|| < \epsilon$.

- Basically, for any real positive ϵ, an element can be found in the sequence, beyond which any two elements of the sequence will be within ϵ of each other.
- In other words, the elements of the sequence comes closer and closer to each other - *i.e.*, the sequence converges.

Definition

A vector space $\mathcal{V}$ equipped with a norm $||.||$ is **complete** if **every** Cauchy sequence converges in that norm to a point **in the space**. To pay tribute to Stefan Banach, the great Polish mathematician, such a space is also called the **Banach space**.

Contraction Mapping, Fixed Point

Definition

An operator $\mathcal{T} : \mathcal{V} \rightarrow \mathcal{V}$ is **L-Lipschitz** if for any $u, v \in \mathcal{V}$

$$||\mathcal{T}u - \mathcal{T}v|| \leq L||u - v||$$

- If $L \leq 1$, then $\mathcal{T}$ is called a **non-expansion**, while if $0 \leq L < 1$, then $\mathcal{T}$ is called a **contraction**.

Contraction Mapping, Fixed Point

Definition

An operator $\mathcal{T}:\mathcal{V}\rightarrow\mathcal{V}$ is **L-Lipschitz** if for any $u, v \in \mathcal{V}$

$$||\mathcal{T}u - \mathcal{T}v|| \leq L||u - v||$$

§ If $L \leq 1$, then $\mathcal{T}$ is called a **non-expansion**, while if $0 \leq L < 1$, then $\mathcal{T}$ is called a **contraction**.

Definition

Let v is a vector in the vector space $\mathcal{V}$ and $\mathcal{T}$ is an operator $\mathcal{T}:\mathcal{V}\rightarrow\mathcal{V}$. Then v is called a **fixed point** of the operator $\mathcal{T}$, if $\mathcal{T}v = v$.

Banach Fixed Point Theorem

Theorem

Suppose $\mathcal{V}$ is a Banach space and $T : \mathcal{V} \to \mathcal{V}$ is a contraction mapping, then,
- $\exists$ an unique v^* in $\mathcal{V}$ s.t. $Tv^* = v^*$ and
- for arbitrary v^0 in $\mathcal{V}$, the sequence $\{v^n\}$ defined by $v^{n+1} = Tv^n = T^{n+1}v^0$, converges to v^*.

The above theorem tells that

§ T has fixed point, **an unique fixed point**.

§ For **arbitrary** starting point if we keep repeatedly applying T on that starting point, then we will converge to v^*.

Banach Fixed Point Theorem - Proof (1)

§ Let v^n and v^{m+n} be two values of v obtained after the n^{th} and the $(n+m)^{th}$ iteration.

$$
\begin{aligned}
||v^{m+n} - v^n|| &\leq \sum_{k=0}^{m-1} ||v^{n+k+1} - v^{n+k}|| \quad \text{[Triangle inequality]} \\
&= \sum_{k=0}^{m-1} ||T^{n+k}v^1 - T^{n+k}v^0|| \leq \sum_{k=0}^{m-1} \lambda||T^{n+k-1}v^1 - T^{n+k-1}v^0|| \\
&\leq \sum_{k=0}^{m-1} \lambda^{n+k}||v^1 - v^0|| \quad \text{[Repeated use of contraction]} \\
&= ||v^1 - v^0|| \sum_{k=0}^{m-1} \lambda^{n+k} \\
&= \frac{\lambda^n(1-\lambda^m)}{1-\lambda}||v^1 - v^0|| \qquad (3)
\end{aligned}
$$

Banach Fixed Point Theorem - Proof (2)

- As m and $n \to \infty$ and as $\lambda < 1$, the norm of difference between v^{m+n} and v^n becomes less and less.
- That means the sequence $\{v^n\}$ is Cauchy.
- And since $\mathcal{V}$ is a Banach space and since every Cauchy sequence converges to a point in that Banach space, therefore the Cauchy sequence $\{v^n\}$ also converges to a point in $\mathcal{V}$.

Banach Fixed Point Theorem - Proof (2)

- As m and $n \to \infty$ and as $\lambda < 1$, the norm of difference between v^{m+n} and v^n becomes less and less.
- That means the sequence $\{v^n\}$ is Cauchy.
- And since $\mathcal{V}$ is a Banach space and since every Cauchy sequence converges to a point in that Banach space, therefore the Cauchy sequence $\{v^n\}$ also converges to a point in $\mathcal{V}$.
- What we have proved till now is that the sequence $\{v^n\}$ will reach a converging point in the same space.
- Lets say that the converging point is v^*.
- What we will try to prove next is that v^* is a fixed point and then we will try to prove that v^* is an **unique** fixed point.

Banach Fixed Point Theorem - Proof (3)

§ Let us try to see what we get as the norm of the difference between v^* and Tv^*.

Banach Fixed Point Theorem - Proof (3)

- Let us try to see what we get as the norm of the difference between v^* and Tv^*.
- In the first line below we apply triangle inequality where v^n is the value of v at the n^{th} iteration.

$$\begin{aligned} ||Tv^* - v^*|| &\leq ||Tv^* - v^n|| + ||v^n - v^*|| \\ &= ||Tv^* - Tv^{n-1}|| + ||v^n - v^*|| \\ &\leq \lambda||v^* - v^{n-1}|| + ||v^n - v^*|| \text{ [Contraction property]} \end{aligned} \tag{4}$$

Banach Fixed Point Theorem - Proof (3)

- Let us try to see what we get as the norm of the difference between v^* and Tv^*.
- In the first line below we apply triangle inequality where v^n is the value of v at the n^{th} iteration.

$$\begin{aligned} ||Tv^* - v^*|| &\leq ||Tv^* - v^n|| + ||v^n - v^*|| \\ &= ||Tv^* - Tv^{n-1}|| + ||v^n - v^*|| \\ &\leq \lambda||v^* - v^{n-1}|| + ||v^n - v^*|| \text{ [Contraction property]} \end{aligned} \tag{4}$$

- Since $\{v^n\}$ is Cauchy and v^* is the convergence point, both the terms in the above equation will tend to 0 as $n \to \infty$.
- So, as $n \to \infty, ||Tv^* - v^*|| \to 0$. That means in limit $Tv^* = v^*$. So, it is proved that v^* is a fixed point.

Banach Fixed Point Theorem - Proof (4)

§ Now we will show the uniqueness, *i.e.*, v^* is unique.

Banach Fixed Point Theorem - Proof (4)

- Now we will show the uniqueness, *i.e.*, v^* is unique.
- Let u^* and v^* be two fixed points of the space. From the contraction property, we can write $||Tu^* - Tv^*|| \leq \lambda ||u^* - v^*||$.
- But, since u^* and v^* are fixed points, $Tu^* = u^*$ and $Tv^* = v^*$.

Banach Fixed Point Theorem - Proof (4)

§ Now we will show the uniqueness, *i.e.*, v^* is unique.

§ Let u^* and v^* be two fixed points of the space. From the contraction property, we can write $||Tu^* - Tv^*|| \leq \lambda||u^* - v^*||$.

§ But, since u^* and v^* are fixed points, $Tu^* = u^*$ and $Tv^* = v^*$.

§ That means $||u^* - v^*|| \leq \lambda||u^* - v^*||$ which can not be true for $\lambda < 1$ unless $v^* = u^*$.

§ So, it is proved that v^* is an unique fixed point.

Existence and Uniqueness of Bellman Equations

§ Now, we will start talking about the existance and uniqueness of the solution to Bellman expecation equations and the Bellman optimality equations.

§ In case of a finite MDP the value function v can be thought of as a vector in a $|\mathcal{S}|$ dimensional vector space $\mathcal{V}$.

§ Whenever, we will use norm $||.||$ in this space we will mean the max norm, unless otherwise specified.

Existence and Uniqueness of Bellman Equations

§ Previously, we have seen

- $r_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) r(s,a)$, one step expected reward for following policy π at state s.
- $p_\pi(s'|s) = \sum_{a \in \mathcal{A}} \pi(a|s) p(s'|s,a)$, one step transition probability under policy π.

§ Using these notations, the Bellman expectation equation becomes,

$$
\begin{aligned}
v_\pi(s) &= \sum_{a \in \mathcal{A}} \pi(a|s) \left\{ r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) v_\pi(s') \right\} \\
&= r_\pi(s) + \gamma \sum_{s' \in \mathcal{S}} p_\pi(s'|s) v_\pi(s')
\end{aligned}
$$

Existence and Uniqueness of Bellman Equations

§ $v_\pi(s) = r_\pi(s) + \gamma \sum_{s' \in \mathcal{S}} p_\pi(s'|s) v_\pi(s')$

Existence and Uniqueness of Bellman Equations

- $v_\pi(s) = r_\pi(s) + \gamma \sum_{s' \in \mathcal{S}} p_\pi(s'|s) v_\pi(s')$
- Refresher from earlier lectures

$$\begin{bmatrix} v(s_1) \\ v(s_2) \\ \vdots \\ v(s_n) \end{bmatrix} = \begin{bmatrix} r(s_1) \\ r(s_2) \\ \vdots \\ r(s_n) \end{bmatrix} + \gamma \begin{bmatrix} \mathcal{P}_{11} & \mathcal{P}_{12} & \cdots & \mathcal{P}_{1n} \\ \mathcal{P}_{21} & \mathcal{P}_{22} & \cdots & \mathcal{P}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{P}_{n1} & \mathcal{P}_{n2} & \cdots & \mathcal{P}_{nn} \end{bmatrix} \begin{bmatrix} v(s_1) \\ v(s_2) \\ \vdots \\ v(s_n) \end{bmatrix}$$

- $v_\pi = r_\pi + \gamma P_\pi v_\pi$
- r_π is a $|\mathcal{S}|$ dimensional vector while P_π is a $|\mathcal{S}| \times |\mathcal{S}|$ dimensional matrix.
- For all $s', p_\pi(s'|s)$ is one row (s^{th} row) of the P_π matrix. Similarly, $v_\pi(s')$'s are the value functions for all states *i.e.*, in the vectorized notation, this is a vector v_π.

Existence and Uniqueness of Bellman Equations

- $v_\pi = r_\pi + \gamma P_\pi v_\pi$
- We are, now, going to define a linear operator.

$$L_\pi : \mathcal{V} \to \mathcal{V} \text{ such that}$$
$$L_\pi v \equiv r_\pi + \gamma P_\pi v \;\forall v \in \mathcal{V}, \quad [\mathcal{V} \text{ as defined in slide (37)}] \tag{5}$$

- So using this operator notation, we can write the Bellman expectation equation as the following,

$$L_\pi v_\pi = v_\pi \tag{6}$$

- So far we have proved the Banach Fixed Point Theorem. Now we will try to show that L_π is a contraction.
- We will hold the proof of $\mathcal{V}$ being a Banach space for later.

Existence and Uniqueness of Bellman Equations

§ Let u and v be in $\mathcal{V}$. So,

$$\begin{aligned} L_\pi u(s) &= r_\pi(s) + \gamma \sum_{s'} p_\pi(s'|s)u(s') \\ L_\pi v(s) &= r_\pi(s) + \gamma \sum_{s'} p_\pi(s'|s)v(s') \end{aligned} \tag{7}$$

§ **One important note:** $L_\pi u(s)$ or $L_\pi v(s)$ does not mean L_π applied on $u(s)$ or $v(s)$. It means the s^{th} component of the vector $L_\pi u$ or $L_\pi v$

Existence and Uniqueness of Bellman Equations

§ Let us consider the case, $L_\pi v(s) > L_\pi u(s)$. Then,

$$
\begin{aligned}
0 &\leq L_\pi v(s) - L_\pi u(s) \\
&= \gamma \sum_{s'} p_\pi(s'|s)\{v(s') - u(s')\} \\
&\leq \gamma ||v-u|| \sum_{s'} p_\pi(s'|s) \\
&\text{[Why is this?]} \\
&= \gamma ||v-u|| \ \ \text{[Since } \sum_{s'} p_\pi(s'|s) = 1] \qquad (8)
\end{aligned}
$$

§ Similarly, when $L_\pi u(s) > L_\pi v(s)$, we can show that,

$$
0 \leq L_\pi u(s) - L_\pi v(s) \leq \gamma ||u-v|| = \gamma ||v-u|| \ \text{[Since } ||u-v|| = ||v-u||] \qquad (9)
$$

Existence and Uniqueness of Bellman Equations

§ Putting the two equations 8 and 9 together, we can get that

$$|L_\pi v(s) - L_\pi u(s)| \leq \gamma ||v - u|| \; \forall s \in \mathcal{S} \tag{10}$$

§ Pointwise or componentwise the difference is being drawn closer by a factor of γ, so the maximum of the difference will also have come down.

$$||L_\pi v - L_\pi u|| \leq \gamma ||v - u|| \tag{11}$$

§ So, that means that L_π is a contraction.

Existence and Uniqueness of Bellman Equations

§ Another proof of contraction property of Bellman expectation operator.

$$
\begin{aligned}
||L_\pi v - L_\pi u||_\infty &= \left|\left| r_\pi(s) + \gamma \sum_{s' \in \mathcal{S}} p_\pi(s'|s) v(s') - r_\pi(s) - \gamma \sum_{s' \in \mathcal{S}} p_\pi(s'|s) u(s') \right|\right|_\infty \\
&= \gamma \left|\left| \sum_{s' \in \mathcal{S}} p_\pi(s'|s) \left\{ v(s') - u(s') \right\} \right|\right|_\infty \\
&= \gamma \max_{s \in \mathcal{S}} \left| \sum_{s' \in \mathcal{S}} p_\pi(s'|s) \left\{ v(s') - u(s') \right\} \right| \\
&\leq \gamma \max_{s \in \mathcal{S}} \sum_{s' \in \mathcal{S}} p_\pi(s'|s) \left| \left\{ v(s') - u(s') \right\} \right| \\
&\leq \gamma \max_{s \in \mathcal{S}} \sum_{s' \in \mathcal{S}} p_\pi(s'|s) \, ||\{v - u\}||_\infty
\end{aligned}
$$

[Absolute value each element $\leq$ max norm of a vector]

$$
= \gamma \, ||\{\{v - u\}||_\infty \cancelto{1}{\sum_{s' \in \mathcal{S}} p_\pi(s'|s)} = \gamma \, ||\{v - u\}||_\infty
$$

Existence and Uniqueness of Bellman Equations

§ Next we have to move on to the Bellman optimality equation's convergence proof.

§ Bellman optimality equation is given by

$$v_*(s) = \max_{a \in \mathcal{A}} \left\{ r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) v_*(s') \right\} \tag{12}$$

§ Let us define the Bellman optimality operator,

$$L : \mathcal{V} \rightarrow \mathcal{V} \text{ such that}$$

$$(Lv)(s) \equiv \max_{a \in \mathcal{A}} \left\{ r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) v(s') \right\} \forall v \in \mathcal{V} \tag{13}$$

§ To declutter notation, we will use $Lv(s)$ to denote $(Lv)(s)$.

§ Then Bellman optimality equation becomes

$$v_* = Lv_* \quad \backslash\backslash\texttt{Componentwise} \tag{14}$$

Existence and Uniqueness of Bellman Equations

§ Now we will prove that L is contraction by taking the same route as we took for L_π.

§ Let u and v be in $\mathcal{V}$. Let us also assume, first, that $Lv(s) \geq Lu(s)$. Then we can write,

$$\begin{aligned}
0 &\leq Lv(s) - Lu(s) \\
&= \Big\{ r(s, a_s^*) + \gamma \sum_{s'} p(s'/s, a_s^*) v(s') \Big\} - \Big\{ r(s, (a')_s^*) + \gamma \sum_{s'} p(s'/s, (a')_s^*) u(s') \Big\} \\
&\leq \Big\{ r(s, a_s^*) + \gamma \sum_{s'} p(s'/s, a_s^*) v(s') \Big\} - \Big\{ r(s, a_s^*) + \gamma \sum_{s'} p(s'/s, a_s^*) u(s') \Big\}
\end{aligned}$$

[why?? Note what has changed!] (15)

Existence and Uniqueness of Bellman Equations

§ Now we will prove that L is contraction by taking the same route as we took for L_π.

§ Let u and v be in $\mathcal{V}$. Let us also assume, first, that $Lv(s) \geq Lu(s)$. Then we can write,

$$
\begin{aligned}
0 &\leq Lv(s) - Lu(s) \\
&= \Big\{ r(s, a_s^*) + \gamma \sum_{s'} p(s'/s, a_s^*) v(s') \Big\} - \Big\{ r(s, (a')_s^*) + \gamma \sum_{s'} p(s'/s, (a')_s^*) u(s') \Big\} \\
&\leq \Big\{ r(s, a_s^*) + \gamma \sum_{s'} p(s'/s, a_s^*) v(s') \Big\} - \Big\{ r(s, a_s^*) + \gamma \sum_{s'} p(s'/s, a_s^*) u(s') \Big\}
\end{aligned}
$$

[why?? Note what has changed!] (15)

§ The two actions a_s^* and $(a')_s^*$ maximize the value functions v and u respectively at state s. So replacing $(a')_s^*$ with a_s^*, in the second part reduces the value of the second part.

Existence and Uniqueness of Bellman Equations

$$
\begin{aligned}
0 &\le Lv(s) - Lu(s) \\
&\le \Big\{r(s, a_s^*) + \gamma \sum_{s'} p(s'/s, a_s^*) v(s')\Big\} - \Big\{r(s, a_s^*) + \gamma \sum_{s'} p(s'/s, a_s^*) u(s')\Big\} \\
&= \gamma \sum_{s'} p(s'/s, a_s^*)[v(s') - u(s')] \\
&\le \gamma \sum_{s'} p(s'/s, a_s^*) ||v - u|| \;\; \text{[Use of max norm similar to } L_\pi\text{]} \\
&= \gamma ||v - u|| \;\; \text{[Since } \sum_{s'} p(s'/s, a_s^*) = 1\text{]}
\end{aligned} \tag{16}
$$

Similarly, for the second case $Lu(s) \ge Lv(s)$, we can write,

$$
0 \le Lu(s) - Lv(s) \le \gamma ||v - u|| \tag{17}
$$

Combining eqns. (16) and (17), $|Lv(s) - Lu(s)| \le \gamma ||v - u|| \; \forall s \in \mathcal{S}$ which again from definition of max norm leads to $||Lv - Lu|| \le \gamma ||v - u||$

Value Iteration Theorem

Theorem (Value Iteration Theorem(ref. S P. Singh and R C. Yee, 1993))

Let $v^0 \in \mathcal{V}, \epsilon > 0$, *sequence* $\{v^n\}$ *is obtained from* $v^{n+1} = Lv^n$, *Then*

I. v^n *converges in norm to* v^*.

II. $\exists$ *a finite* N *at which the condition* $||v^{n+1} - v^n|| < \epsilon\frac{1-\gamma}{2\gamma}$ *is met* $\forall n > N$.

III. $\pi(s)$ *(obtained by* $\arg\max_a \left\{r(s,a) + \gamma \sum_{s'} p(s'/s,a)v^{n+1}(s')\right\}$ $\forall s \in \mathcal{S}$*) is* ϵ *optimal.*

IV. $||v^{n+1} - v^*|| \leq \frac{\epsilon}{2}$ *when the condition* $||v^{n+1} - v^n|| < \epsilon\frac{1-\gamma}{2\gamma}$ *holds.*

Value Iteration Theorem

Theorem (Value Iteration Theorem(ref. S P. Singh and R C. Yee, 1993))

Let $v^0 \in \mathcal{V}, \epsilon > 0$, *sequence* $\{v^n\}$ *is obtained from* $v^{n+1} = Lv^n$, *Then*

I. v^n *converges in norm to* v^*.

II. $\exists$ *a finite* N *at which the condition* $||v^{n+1} - v^n|| < \epsilon\frac{1-\gamma}{2\gamma}$ *is met* $\forall n > N$.

III. $\pi(s)$ *(obtained by* $\arg\max_a \left\{r(s,a) + \gamma \sum_{s'} p(s'/s,a)v^{n+1}(s')\right\} \forall s \in \mathcal{S}$*) is* ϵ *optimal.*

IV. $||v^{n+1} - v^*|| \leq \frac{\epsilon}{2}$ *when the condition* $||v^{n+1} - v^n|| < \epsilon\frac{1-\gamma}{2\gamma}$ *holds.*

§ statement *III* means $||v_\pi - v^*|| \leq \epsilon$. And statement *IV* tells that $||v^{n+1} - v^*|| \leq \frac{\epsilon}{2}$. Are they redundant?

Value Iteration Theorem

Theorem (Value Iteration Theorem(ref. S P. Singh and R C. Yee, 1993))

Let $v^0 \in \mathcal{V}, \epsilon > 0$, sequence $\{v^n\}$ is obtained from $v^{n+1} = Lv^n$, Then

I. *v^n converges in norm to v^*.*

II. *$\exists$ a finite N at which the condition $||v^{n+1} - v^n|| < \epsilon\frac{1-\gamma}{2\gamma}$ is met $\forall n > N$.*

III. *$\pi(s)$ (obtained by $\arg\max\limits_a \left\{r(s,a) + \gamma\sum\limits_{s'} p(s'/s,a)v^{n+1}(s')\right\} \forall s \in \mathcal{S}$) is ϵ optimal.*

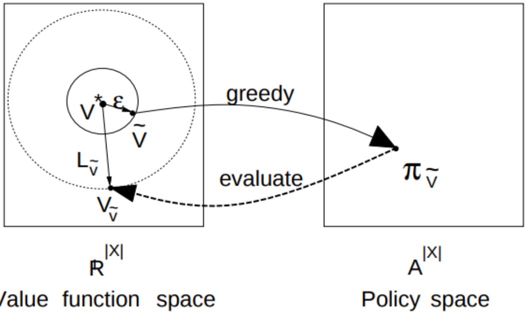
IV. *$||v^{n+1} - v^*|| \leq \frac{\epsilon}{2}$ when the condition $||v^{n+1} - v^n|| < \epsilon\frac{1-\gamma}{2\gamma}$ holds.*

§ statement *III* means $||v_\pi - v^*|| \leq \epsilon$. And statement *IV* tells that $||v^{n+1} - v^*|| \leq \frac{\epsilon}{2}$. Are they redundant?

§ No! Think about what is v_π and what is v^{n+1}.

Value Iteration Theorem

- Though the figure is related to policy iteration, remember the figure in slide (17).



- Figure credit: *[Singh and Yee, 1993]*

- Equality occurs if and only if value function given by the value iteration algorithm is equal to the optimal policy.
- What *III* is telling is that v_π is ϵ optimal and what *IV* is telling is that v^{n+1} is $\frac{\epsilon}{2}$ optimal given condition in *II*.

Proof

§ **Proof**: Suppose, for some n, *II* is met *i.e.*, $||v^{n+1} - v^n|| < \epsilon\frac{1-\gamma}{2\gamma}$ and $\pi(s)$ obtained by *III*. Now, by triangle inequality,

$$||v_\pi - v^*|| \leq ||v_\pi - v^{n+1}|| + ||v^{n+1} - v^*|| \tag{18}$$

Proof

§ **Proof**: Suppose, for some n, *II* is met *i.e.*, $||v^{n+1} - v^n|| < \epsilon\frac{1-\gamma}{2\gamma}$ and $\pi(s)$ obtained by *III*. Now, by triangle inequality,

$$||v_\pi - v^*|| \leq ||v_\pi - v^{n+1}|| + ||v^{n+1} - v^*|| \tag{18}$$

§ Now we have seen L_π to be such that

$$\begin{aligned} L_\pi v(s) &= r_\pi(s) + \gamma \sum_{s'} p_\pi(s'|s)v(s') \\ &= \sum_{a \in \mathcal{A}} \pi(a|s)\left\{r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a)v(s')\right\} \end{aligned} \tag{19}$$

Proof

§ **Proof**: Suppose, for some n, *II* is met *i.e.*, $||v^{n+1} - v^n|| < \epsilon\frac{1-\gamma}{2\gamma}$ and $\pi(s)$ obtained by *III*. Now, by triangle inequality,

$$||v_\pi - v^*|| \leq ||v_\pi - v^{n+1}|| + ||v^{n+1} - v^*|| \tag{18}$$

§ Now we have seen L_π to be such that

$$\begin{aligned} L_\pi v(s) &= r_\pi(s) + \gamma \sum_{s'} p_\pi(s'|s)v(s') \\ &= \sum_{a\in\mathcal{A}} \pi(a|s) \left\{ r(s,a) + \gamma \sum_{s'\in\mathcal{S}} p(s'|s,a)v(s') \right\} \end{aligned} \tag{19}$$

§ Let us apply L_π on v^{n+1} and remember that π is deterministic policy. So,

$$L_\pi v^{n+1}(s) = r(s,\pi(s)) + \gamma \sum_{s'} p(s'/s,\pi(s))v^{n+1}(s') \tag{20}$$

Proof

§ Now we have seen L to be such that

$$Lv(s) \equiv \max_{a \in \mathcal{A}} \left\{ r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a)v(s') \right\} \forall v \in \mathcal{V} \tag{21}$$

Proof

§ Now we have seen L to be such that

$$Lv(s) \equiv \max_{a \in \mathcal{A}} \left\{ r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a)v(s') \right\} \forall v \in \mathcal{V} \tag{21}$$

§ So, similarly, let us apply L on v^{n+1}. So,

$$Lv^{n+1}(s) = \max_{a \in \mathcal{A}} \left\{ r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a)v^{n+1}(s') \right\} \tag{22}$$

Proof

- Repeating eqn. (20) and (22)

$$L_\pi v^{n+1}(s) = r(s, \pi(s)) + \gamma \sum_{s'} p(s'/s, \pi(s)) v^{n+1}(s') \tag{23}$$

-

$$Lv^{n+1}(s) = \max_{a \in \mathcal{A}} \left\{ r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) v^{n+1}(s') \right\} \tag{24}$$

Proof

§ Repeating eqn. (20) and (22)

$$L_\pi v^{n+1}(s) = r(s, \pi(s)) + \gamma \sum_{s'} p(s'/s, \pi(s)) v^{n+1}(s') \tag{23}$$

§

$$Lv^{n+1}(s) = \max_{a \in \mathcal{A}} \left\{ r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) v^{n+1}(s') \right\} \tag{24}$$

§ Now, because π was chosen such that π maximizes the argument inside the $\max\{.\}$ operator, so whether we apply L_π on v^{n+1} or L on v^{n+1}, they are the same, *i.e.*, $Lv^{n+1} = L_\pi v^{n+1}$.

Proof

§ Now let us take the first term in eqn. (18) and proceed.

$$\begin{aligned}
||v_\pi - v^{n+1}|| &= ||L_\pi v_\pi - v^{n+1}|| \text{ [By eqn. 6 - fixed point]} \\
&\leq ||L_\pi v_\pi - Lv^{n+1}|| + ||Lv^{n+1} - v^{n+1}|| \text{ [Triangle inequality]} \\
&= ||L_\pi v_\pi - L_\pi v^{n+1}|| + ||Lv^{n+1} - Lv^{n}|| \\
&\text{[1. Using previous slide 2. } v^{n+1} = Lv^n] \\
&\leq \gamma||v_\pi - v^{n+1}|| + \gamma||v^{n+1} - v^{n}|| \text{ [Contraction mappings]} \\
\implies ||v_\pi - v^{n+1}|| &\leq \frac{\gamma}{1-\gamma}||v^{n+1} - v^n|| \\
&\leq \frac{\gamma}{1-\gamma}\epsilon\frac{1-\gamma}{2\gamma} \text{ [By statement } II \text{ of the theorem]} \\
&= \frac{\epsilon}{2}
\end{aligned} \tag{25}$$

Proof

§ Now let us take the second term in eqn. (18) and proceed.

$$\begin{aligned}
||v^{n+1} - v^*|| &\leq \sum_{k=0}^{\infty} ||v^{n+k+2} - v^{n+k+1}|| \text{ [Triangle inequality repeatedly]} \\
&= \sum_{k=0}^{\infty} ||L^{k+1}v^{n+1} - L^{k+1}v^{n}|| \text{ [From iterative application of } L\text{]} \\
&\leq \sum_{k=0}^{\infty} \gamma^{k+1}||v^{n+1} - v^{n}|| \text{ [}L\text{ is a contraction mapping]} \\
&= \frac{\gamma}{1-\gamma}||v^{n+1} - v^{n}|| \text{ [G.P. sum]} \\
&\leq \frac{\gamma}{1-\gamma}\epsilon\frac{1-\gamma}{2\gamma} \text{ [By statement } \mathit{II} \text{ of the theorem]} \\
&= \frac{\epsilon}{2}
\end{aligned} \tag{26}$$

§ this is also the proof of statement *IV* of the theorem

Proof

Now putting eqn. 25 and eqn. 26 in eqn. 18, we get,

$$||v_\pi - v^*|| \leq \frac{\epsilon}{2} + \frac{\epsilon}{2} = \epsilon \tag{27}$$

So, statement *III* is proved.

Asynchronous Dynamic Programing

- Major drawback of DP methods is that they involve operations over entire state set.
- the game of backgammon has over 10^{20} states. Even if we could perform the value iteration update on a million states per second, it would take over a thousand years to complete a single sweep.

foreach $s \in \mathcal{S}$ **do**

$$v^{n+1}(s) \leftarrow \max_a \left\{ r(s\ \ a) + \gamma \sum_{s'} p(s'/s\ \ a) v^n(s') \right\}$$

end

- Inplace dynamic programing uses one single array to do the update

foreach $s \in \mathcal{S}$ **do**

$$v(s) \leftarrow \max_a \left\{ r(s\ \ a) + \gamma \sum_{s'} p(s'/s\ \ a) v(s') \right\}$$

end

- For convergence, the order of update does not matter as long as all states are picked at least a few times.

Asynchronous Dynamic Programing

- Real Time Dynamic Programing (RTDP): The main idea is to reduce computation again but by not choosing randomly the states.
- In an MDP there may be many states which occur very rarely *i.e.*, they are seldom visited. So there is no point in putting more effort in trying to discover the true value of these states. The agent might not visit it at all.
- Pick an initial state and run a policy/agent from that state. Then employ DP update only on these states.
- This makes changes to the value function estimate. Get the policy from it and sample a trajectory again and do updates along the trajectory.
- Why is it called Real Time?
- Many ideas from RTDP will be used in full RL problem.