Reinforcement Learning

Bellman Equations

aka Dynamic Programming

Akka Zemmari

Bellman Equations

Solving an MDP

Prediction

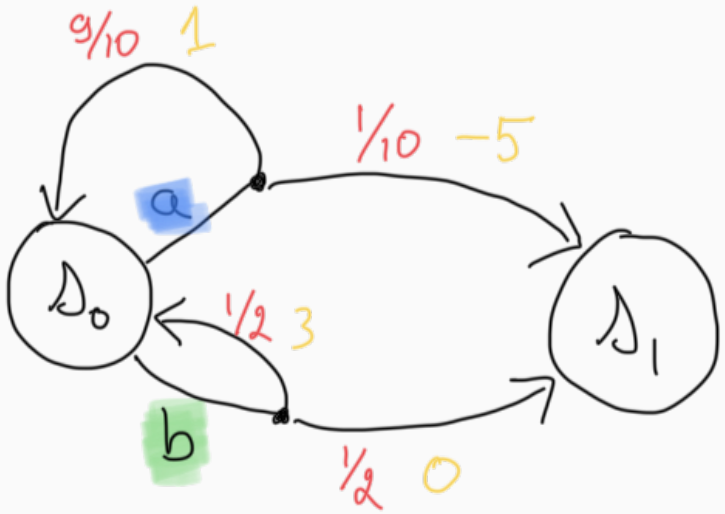
- **Estimate**: $v_\pi(s)$ or $q_\pi(s, a)$ for a given policy π
  Also called: Policy Evaluation
- **Key Question**:
  $>$ Given my strategy, what is my expected return?

Control

- **Estimate**: $\pi_*(s)$ or $q_*(s, a)$
  Goal: Find the Optimal Policy
- **Key Question**:
  $>$ What is the optimal way to behave? For example, what is the best treatment?

Bellman Equations - An Example

Evaluate the value of two different policies for this simple MDP.



$\pi_0(S_0) = a$

$\pi_1(S_0) = b$

- Blackboard.

Bellman Equations

Bellman Equations for Deterministic Policies

- **State-Value Function**:

$$V^{\pi}(s) = \sum_{s' \in S, r \in R} p(s', r|s, \pi(s)) \left[r + \gamma V^{\pi}(s') \right]$$

- **Action-Value Function**:

$$Q^{\pi}(s, a) = \sum_{s' \in S, r \in R} p(s', r|s, a) \left[r + \gamma Q^{\pi}(s', a') \right]$$

Richard E. Bellman (1920-1984)

The equations provide recursive relationships for evaluating a policy.

Bellman Equations

Bellman Equations for Deterministic Policies

- **State-Value Function**:

$$V^{\pi}(s) = \sum_{s' \in S, r \in R} p(s', r|s, \pi(s)) \left[r + \gamma V^{\pi}(s') \right]$$

- **Action-Value Function**:

$$Q^{\pi}(s, a) = \sum_{s' \in S, r \in R} p(s', r|s, a) \left[r + \gamma Q^{\pi}(s', a') \right]$$

Richard E. Bellman (1920-1984)

The equations provide recursive relationships for evaluating a policy.

Question. What if the policy is probabilistic? $\pi : S \times A \to [0, 1]$

Optimal Bellman Equations

Same but for V^* and Q^*.

Optimal Bellman Equations

- **State-Value Function**:

$$\forall s \in S, V^*(s) = \max_a \sum_{s' \in S, r \in R} p(s', r|s, a) \left[r + \gamma V^*(s') \right]$$

- **Action-Value Function**:

$$\forall s \in S, a \in A, Q^*(s, a) = \sum_{s' \in S, r \in R} p(s', r|s, a) \left[r + \gamma V^*(s') \right]$$

Again, the equations provide recursive relationships for finding the optimal policy, but this time the system of equations is non-linear.

Unique Solution

Theorem

V^* is the unique solution to the following system of equations:

$$\forall s \in S, \quad V(s) = \max_a \sum_{s' \in S, r \in R} p(s', r|s, a) \left[r + \gamma V(s') \right]$$

All(most) algorithms for solving MDPs are based on Bellman equations in some way.

Using Bellman Equations to Estimate the Value of a Policy

First Idea: Solve the System of Equations

To estimate the value of a policy, solve the Bellman equations for each state:

$$\begin{cases} V^{\pi}(s_0) = \sum_{s',r} p(s', r \mid s_0, \pi(s_0)) \left[r + \gamma V^{\pi}(s') \right] \\ V^{\pi}(s_1) = \sum_{s',r} p(s', r \mid s_1, \pi(s_1)) \left[r + \gamma V^{\pi}(s') \right] \\ \quad \vdots \\ V^{\pi}(s_N) = \sum_{s',r} p(s', r \mid s_N, \pi(s_N)) \left[r + \gamma V^{\pi}(s') \right] \end{cases}$$

This represents a system of linear equations.

Matrix Form

Let's rewrite the system of equations in matrix form.

$$V^{\pi}(s) = \sum_{s',r} p(s', r \mid s, \pi(s)) \left[r + \gamma V^{\pi}(s') \right]$$

can be written as:

$$V^{\pi}(s) = r(s, \pi(s)) + \gamma \sum_{s'} p(s' \mid s, \pi(s)) V^{\pi}(s')$$

where $r(s, \pi(s)) = \mathbb{E}[R_t \mid S_t = s, A_t = \pi(s)]$ is the expected reward for taking action $\pi(s)$ in state s.

Matrix Form

Thus we get

$$\begin{cases} V^\pi(s_0) = r(s_0, \pi(s_0)) + \gamma \sum_{s'} p(s' \mid s_0, \pi(s_0)) V^\pi(s') \\ V^\pi(s_1) = r(s_1, \pi(s_1)) + \gamma \sum_{s'} p(s' \mid s_1, \pi(s_1)) V^\pi(s') \\ \quad \vdots \\ V^\pi(s_N) = r(s_N, \pi(s_N)) + \gamma \sum_{s'} p(s' \mid s_N, \pi(s_N)) V^\pi(s') \end{cases}$$

Let $\mathbf{V}^\pi$ and $\mathbf{R}^\pi$ be the vectors of values and rewards, and $\mathbf{P}^\pi$ the transition matrix. Then the system of equations can be written as:

Matrix Representation: $\mathbf{V}^\pi = \mathbf{R}^\pi + \gamma \mathbf{P}^\pi \mathbf{V}^\pi$

Matrix Form

Thus we get

$$\begin{cases} V^\pi(s_0) = r(s_0, \pi(s_0)) + \gamma \sum_{s'} p(s' \mid s_0, \pi(s_0)) V^\pi(s') \\ V^\pi(s_1) = r(s_1, \pi(s_1)) + \gamma \sum_{s'} p(s' \mid s_1, \pi(s_1)) V^\pi(s') \\ \quad \vdots \\ V^\pi(s_N) = r(s_N, \pi(s_N)) + \gamma \sum_{s'} p(s' \mid s_N, \pi(s_N)) V^\pi(s') \end{cases}$$

Let $\mathbf{V}^\pi$ and $\mathbf{R}^\pi$ be the vectors of values and rewards, and $\mathbf{P}^\pi$ the transition matrix. Then the system of equations can be written as:

Matrix Representation: $\mathbf{V}^\pi = \mathbf{R}^\pi + \gamma \mathbf{P}^\pi \mathbf{V}^\pi$

Solution: $\mathbf{V}^\pi = (\mathbf{I} - \gamma \mathbf{P}^\pi)^{-1} \mathbf{R}^\pi$

Computational Complexity

$$\mathbf{V}^{\pi} = (\mathbf{I} - \gamma \mathbf{P}^{\pi})^{-1} \mathbf{R}^{\pi}$$

- Size of the matrix: $|S| \times |S|$
- Complexity of matrix inversion:
  - $O(|S|^3)$ using Gauss-Jordan elimination
  - $O(|S|^{2.807})$ using Strassen's algorithm
  - $O(|S|^{2.376})$ using Coppersmith-Winograd algorithm

Computational Complexity

$$\mathbf{V}^\pi = (\mathbf{I} - \gamma \mathbf{P}^\pi)^{-1} \mathbf{R}^\pi$$

- Size of the matrix: $|S| \times |S|$
- Complexity of matrix inversion:
  - $O(|S|^3)$ using Gauss-Jordan elimination
  - $O(|S|^{2.807})$ using Strassen's algorithm
  - $O(|S|^{2.376})$ using Coppersmith-Winograd algorithm

Problems

- This is too slow for large MDPs.
- This idea cannot be used for the optimal policy because the system of equations is non-linear :-(

Computational Complexity

$$\mathbf{V}^{\pi} = (\mathbf{I} - \gamma \mathbf{P}^{\pi})^{-1} \mathbf{R}^{\pi}$$

- Size of the matrix: $|S| \times |S|$
- Complexity of matrix inversion:
  - $O(|S|^3)$ using Gauss-Jordan elimination
  - $O(|S|^{2.807})$ using Strassen's algorithm
  - $O(|S|^{2.376})$ using Coppersmith-Winograd algorithm

Problems

- This is too slow for large MDPs.
- This idea cannot be used for the optimal policy because the system of equations is non-linear :-(

We need to find a **faster** and **more general** algorithm.

Iterative Methods

Fixed-Point computation

We have a function $f : \mathbb{R}^n \to \mathbb{R}^n$ and we want to find a fixed point: a point x^* such that $f(x^*) = x^*$.

α-contraction

A function is an α-contraction iff

$$\forall x, y \in \mathbb{R}^n, \quad \|f(x) - f(y)\| \leq \alpha \|x - y\|$$

for some $\alpha \in [0, 1)$.

The killer theorem

Banach Theorem

If f is an α-contraction, then

- there exists a unique fixed point x^*
- the sequence $x_{k+1} = f(x_k)$ converges to x^* for any initial point x_0
- it converges exponentially fast: $\|x^* - x_k\| \leq \frac{\alpha^k}{1-\alpha}\|x_1 - x_0\|$

We can use this to solve the Bellman equations

Three main algorithms derived from Banach's theorem:

- **Policy Evaluation.** Find the value $\mathbf{V}^{\pi}$ of a policy π.
- **Value Iteration.** Find the optimal value function $\mathbf{V}^{*}$.
- **Policy Iteration.** Find the optimal policy π^{*}.

Policy Evaluation - Finding the Value V^π of a Policy π

Given a policy π, we define the Bellman operator $T^\pi : \mathbb{R}^{|S|} \to \mathbb{R}^{|S|}$ as:

$$(T^\pi(\mathbf{V}))_s = \sum_{s',r} p(s', r \mid s, \pi(s)) \left[r + \gamma \mathbf{V}(s')\right]$$

Finding the value of a policy is equivalent to finding the fixed point of T^π, i.e. $\mathbf{V}^\pi = T^\pi(\mathbf{V}^\pi)$.

It is possible to prove that T^π is a α-contraction, therefore we can apply Banach's theorem to find the fixed point.

Policy Evaluation - Finding the Value V^{π} of a Policy π

Policy Evaluation Algorithm

1. Initialize $\mathbf{V}_0$ randomly
2. Repeat until convergence:
    - For each state s, update $V_{k+1}(s) = \sum_{s',r} p(s', r \mid s, \pi(s)) [r + \gamma V_k(s')]$

Policy Evaluation Algorithm

1. Initialize $\mathbf{V}_0$ randomly
2. Repeat **until convergence**:
    - For each state s, update
      $V_{k+1}(s) = \sum_{s',r} p(s', r \mid s, \pi(s)) \left[r + \gamma V_k(s') \right]$

What does **until convergence** mean?

- The difference between two consecutive values is smaller than a threshold: $\| V^{\pi}_{k+1} - V^{\pi}_{k} \| < \epsilon$
- There is a fixed number of iterations: "repeat N times"
- ...

Value Iteration

This is very similar to Policy Iteration. We define the optimal Bellman operator $T^* : \mathbb{R}^{|S|} \to \mathbb{R}^{|S|}$ as:

$$(T^*(\mathbf{V}))_s = \max_a \sum_{s',r} p(s', r \mid s, a) \left[r + \gamma \mathbf{V}(s') \right]$$

Value Iteration Algorithm

1. Initialize $\mathbf{V}_0$ randomly
2. Repeat **until convergence**:
    - For each state s, update $V_{k+1}(s) = \max_a \sum_{s',r} p(s', r \mid s, a) \left[r + \gamma V_k(s') \right]$

Optimal Policy

Once we have the optimal value function $\mathbf{V}^*$, we can find the optimal policy π^* by taking the greedy policy with respect to $\mathbf{V}^*$:

$$\pi^*(s) = \arg\max_a \sum_{s',r} p(s', r \mid s, a) \left[r + \gamma \mathbf{V}^*(s') \right]$$

We can also apply directly Value Iteration to find the optimal action-value function $\mathbf{Q}^*$, in which case we get the optimal policy directly using:

$$\pi^*(s) = \arg\max_a \mathbf{Q}^*(s, a)$$

Optimal Policy

Once we have the optimal value function $\mathbf{V}^*$, we can find the optimal policy π^* by taking the greedy policy with respect to $\mathbf{V}^*$:

$$\pi^*(s) = \arg\max_a \sum_{s',r} p(s', r \mid s, a) \left[r + \gamma \mathbf{V}^*(s') \right]$$

We can also apply directly Value Iteration to find the optimal action-value function $\mathbf{Q}^*$, in which case we get the optimal policy directly using:

$$\pi^*(s) = \arg\max_a \mathbf{Q}^*(s, a)$$

Question. Why is it not the best idea to use the optimal action-value function $\mathbf{Q}^*$ to find the optimal policy?

Policy Iteration

Value Iteration: $V_0 \to V_1 \to V_2 \to \ldots \to V^*$

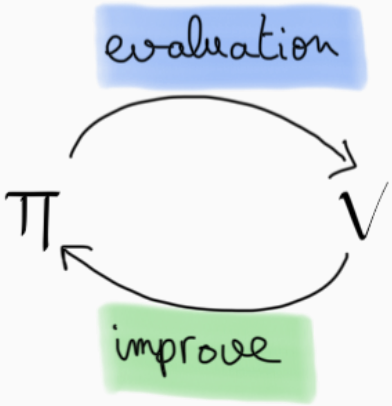
Policy Iteration: $\pi_0 \to \pi_1 \to \pi_2 \to \ldots \to \pi^*$

–> strategy improvement.

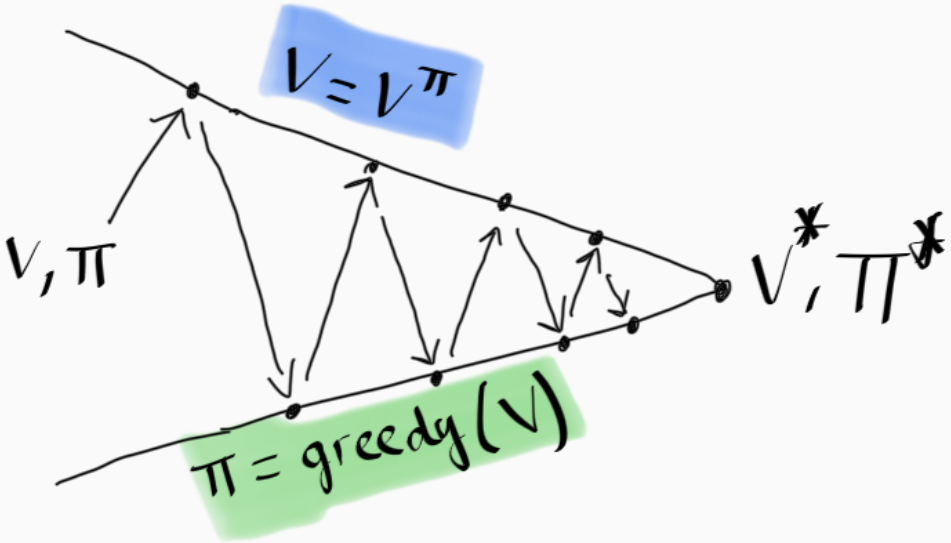
It works in two steps that are repeated until convergence:

$$\pi_k \to V^{\pi_k} \to \pi_{k+1}$$

- **Evaluation.** $\pi_k \to V^{\pi_k}$
  - Compute the value function V^{π_k} for the policy π_k.
  - How? Use policy evaluation.
- **Improvement.** $V^{\pi_k} \to \pi_{k+1}$
  - Define a new policy π_{k+1} that is greedy with respect to Q^{π_k}.
  - $\pi_{k+1}(s) = \arg\max_a Q^{\pi_k}(s, a)$



Policy Iteration's loop



Policy Iteration's convergence

Value Iteration vs Policy Iteration

- **Value Iteration** is simpler and often faster.
- **Value Iteration**. Convergence is only asymptotic.
- **Policy Iteration** is more stable and can be more efficient in some cases.
- **Policy Iteration** is guaranteed to converge to the optimal policy in a finite number of steps, while **Value Iteration** converges to the optimal value function but not necessarily in a finite number of steps.
- In **Policy Iteration** we know when to stop: when the policy does not change anymore, it means we have found the optimal policy.
- **Policy Iteration**. More expensive per iteration because it requires policy evaluation.

Summary

- **Bellman Equations** provide recursive relationships for the value of a policy or the optimal value of an MDP.

$$V^{\pi}(s) = \sum_{s',r} p(s', r \mid s, \pi(s)) \left[r + \gamma V^{\pi}(s') \right]$$

$$V^{*}(s) = \max_{a} \sum_{s',r} p(s', r \mid s, a) \left[r + \gamma V^{*}(s') \right]$$

- **Iterative Methods** are used to solve the Bellman equations. Based on fixed-point computation and Banach's theorem.

$$X_{k+1} = f(X_k)$$

- **Policy Evaluation**, **Value Iteration**, and **Policy Iteration** are powerful examples of iterative methods applied to MDPs.