UNDERSTANDING BELLMAN EQUATIONS

Shuo Liu
Computer Science
Northeastern University
shuo.liu2@northeastern.edu

ABSTRACT

This article covers Bellman Equation, its RL specializations, and generalizations.

1 BELLMAN EQUATION AND OPERATOR

1.1 DEFINITION

The Bellman Equation and optimal Bellman Equation for V-values are, Sutton & Barto (2018),[1]

$$\begin{aligned} V^\pi(s) &\doteq \mathbb{E}_{a\sim\pi(\cdot|s)}\left[Q^\pi(s,a)\right], \\ &= \mathbb{E}_{a\sim\pi(\cdot|s)}\left[R(s,a)+\gamma\mathbb{E}_{s'\sim P(\cdot|s,a)}\left[V^\pi(s')\right]\right], \\ V^*(s) &\doteq \max_a\left[Q^*(s,a)\right], \\ &= \max_a\left[R(s,a)+\gamma\mathbb{E}_{s'\sim P(\cdot|s,a)}\left[V^*(s')\right]\right]. \end{aligned} \tag{1}$$

and the Bellman Equation and optimal Bellman Equation for Q-values are,

$$\begin{aligned} Q^\pi(s,a) &\doteq R(s,a)+\gamma\mathbb{E}_{s'\sim P(\cdot|s,a)}\left[V^\pi(s')\right], \\ &= R(s,a)+\gamma\mathbb{E}_{s'\sim P(\cdot|s,a)}\left[\mathbb{E}_{a'\sim\pi(a'|s')}Q^\pi(s',a')\right]. \\ Q^*(s,a) &\doteq R(s,a)+\gamma\mathbb{E}_{s'\sim P(\cdot|s,a)}\left[V^*(s')\right], \\ &= R(s,a)+\gamma\mathbb{E}_{s'\sim P(\cdot|s,a)}\left[\max_{a'}Q^*(s',a')\right]. \end{aligned} \tag{2}$$

where $V^\pi(s)$ and $Q^\pi(s,a)$ are value representations following policy π, e.g., vectors and functions.

$$\tilde{\pi}(s) \doteq \operatorname*{argmax}_a Q^\pi(s,a). \tag{3}$$

Bellman Equations establish relations between states and succeeding states, which can be applied as updating rules for value prediction. A succinct representation is to define the Bellman Equation as a unary mathematical operator. The V-value Bellman and optimal Bellman Operators are,

$$\begin{aligned} (\mathcal{T}^\pi\circ V^\pi)(s) &\doteq \mathbb{E}_{a\sim\pi(\cdot|s)}\left[R(s,a)+\gamma\mathbb{E}_{s'\sim P(\cdot|s,a)}\left[V^\pi(s')\right]\right], \\ (\mathcal{T}^*\circ V^\pi)(s) &\doteq \max_a\left[R(s,a)+\gamma\mathbb{E}_{s'\sim P(\cdot|s,a)}\left[V^\pi(s')\right]\right]. \end{aligned} \tag{4}$$

The Bellman and optimal Bellman Operators $\mathcal{T}^\pi$ for Q-values are,

$$\begin{aligned} (\mathcal{T}^\pi\circ Q^\pi)(s,a) &\doteq R(s,a)+\gamma\mathbb{E}_{s'\sim P(\cdot|s,a)}\left[\mathbb{E}_{a'\sim\pi(a'|s')}Q^\pi(s',a')\right], \\ (\mathcal{T}^*\circ Q^\pi)(s,a) &\doteq R(s,a)+\gamma\mathbb{E}_{s'\sim P(\cdot|s,a)}\left[\max_{a'}Q^\pi(s',a')\right]. \end{aligned} \tag{5}$$

Curse of Dimension Why do we mostly use MDP (where the future evolution is independent of its history) and hence Bellman Equations to model RL problems? Bellman (1957) coined the "curse of dimension", which describes the exponential increase in the state space size as dimensionality grows, making calculations extremely complex. Breaking this curse often requires altering the problem or its constraints, though complete solutions are not always achievable.

For convenience, we use Q-value as the representative in the following parts of this article.

[1]The Bellman Equations shown in the main article are stochastic, the deterministic one can be derived as, $V(s)=\max_a\{R(s,a)+\gamma V(s')\}$, where $s'\leftarrow T(s,a)$ is a deterministic succeeding state of s following a.

1.2 IMPORTANT PROPERTIES

Proposition 1 (γ-contraction)**.** *Given any $Q,\ Q' \mapsto \mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|}$, Bellman Operators are γ-contraction Operators in L^∞ norm,*

$$\begin{aligned} \|\mathcal{T}^\pi \circ Q - \mathcal{T}^\pi \circ Q'\|_\infty &\leqslant \gamma \|Q - Q'\|_\infty, \\ \textit{and } \|\mathcal{T}^* \circ Q - \mathcal{T}^* \circ Q'\|_\infty &\leqslant \gamma \|Q - Q'\|_\infty. \end{aligned} \tag{6}$$

Corollary 1 (Fixed-point Iteration)**.** *For any $Q^0 \mapsto \mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|}$, after $k \to \infty$ iterations of Bellman transformation, $Q^{\pi,\infty} \doteq \lim_{k\to\infty}(\mathcal{T}^\pi)^k \circ Q^0$, or $Q^{*,\infty} \doteq \lim_{k\to\infty}(\mathcal{T}^*)^k \circ Q^0$, according to Banach's Fixed Point Theorem,*

$$\begin{gathered} Q^{\pi,\infty} = Q^{*,\infty} = Q^*, \\ \textit{which } \textbf{\textit{uniquely}} \textit{ satisfies } \mathcal{T}^\pi \circ Q^* = Q^*, \textit{ or } \mathcal{T}^* \circ Q^* = Q^*. \end{gathered} \tag{7}$$

Theorem 1 (Fundamental theorem)**.** *Any memoryless policy that is greedy to Q^* (**deterministically** maximizes) is optimal Szepesvári (2010),*

$$\tilde{\pi}^* \doteq \operatorname*{argmax}_a Q^* = \pi^*. \tag{8}$$

Proposition 2 (Monotone)**.** *Bellman Operators are monotonic. For any Q-values $Q, Q' \mapsto \mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|}$,*

$$\begin{aligned} (Q \leqslant Q') &\Leftrightarrow (\mathcal{T}^\pi \circ Q \leqslant \mathcal{T}^\pi \circ Q'), \\ (Q \leqslant Q') &\Leftrightarrow (\mathcal{T}^* \circ Q \leqslant \mathcal{T}^* \circ Q'). \end{aligned} \tag{9}$$

2 BELLMAN BACKUP FOR PLANNING

2.1 DYNAMIC PROGRAMMING

According to the Fundamental Theorem, we can find π^* efficiently once having access to Q^*, without the need to find the policy whose Q-function **dominates** the others' brute-force-ly. To avoid the Bellman Curse of Dimensionality, we can apply Dynamic Programming (DP) methods to solve MDPs by keeping track of Q-values during calculations, thanks to Bellman recursions.

Value Iteration Value iteration (so-called backward induction) involves iteratively applying $\mathcal{T}^*$ to arbitrarily initialized values Q^0 until convergence. According to Corollary 1 and Theorem 1, value iteration converges to Q^* as $k \to \infty$, then an optimal policy π^* can be derived by greedifying Q^*.

Policy Iteration Policy iteration starts with an arbitrary policy π^0 and values Q^0. In each iterative step k, $Q^{\pi,k}$ is calculated by applying Bellman Operator $\mathcal{T}^{\pi,k}$ that follows current policy π^k to $Q^{\pi,k-1}$ from the last iteration, and then π^{k+1} is derived from greedifying $Q^{\pi,k}$. This process is repeated until convergence, and policy iteration can produce optimal policy after sufficient iterations.

3 BELLMAN RESIDUAL FOR LEARNING

3.1 TD-LEARNING WITH LOOK-UP TABLE

When the transition model is unavailable (model-free), we can use the residuals (RHS minus LHS) of the Bellman Equations as learning objective,

$$\begin{aligned} (\mathcal{B}^\pi \circ Q)(s,a) &\doteq r + \gamma Q(s', \pi(s')) - Q(s,a), \\ (\mathcal{B}^* \circ Q)(s,a) &\doteq r + \gamma \max_{a'} Q(s', a') - Q(s,a). \end{aligned} \tag{10}$$

Assuming that our sampling and parameter updating roughly follow the true state distribution $\mu(s)$, the expectation of Bellman residual will be closed to zero at the optima. This approach is often called temporal difference (TD) learning.

TD-learning In TD-learning with learning rate α, the update rule for Q-values is,

$$Q(s,a) \leftarrow Q(s,a) + \alpha(\mathcal{B}^\pi \circ Q)(s,a). \tag{11}$$

According to Stochastic Approximation Theorem, let k be the visitation times of state-action pair, and learning rates $0 \leqslant \alpha^k < 1$ satisfies $\forall(s,a)$, $\sum_{k=1}^{\infty} \alpha^k(s,a) = \infty, \sum_{k=1}^{\infty}[\alpha^k(s,a)]^2 < \infty$. Following TD-learning updates, $Q^{\pi,k}(s,a)$ converges to $Q^*(s,a)$ as $k \to \infty$ (Jaakkola et al. (1994)).

Q-learning In Q-learning that relies on optimal Bellman Equation, the Q-value update is,

$$Q(s,a) \leftarrow Q(s,a) + \alpha(\mathcal{B}^* \circ Q)(s,a). \tag{12}$$

According to Stochastic Approximation Theorem, let k be the visitation times of state-action pair, and learning rates $0 \leqslant \alpha^k < 1$ satisfies $\forall(s,a)$, $\sum_{k=1}^{\infty} \alpha^k(s,a) = \infty, \sum_{k=1}^{\infty}[\alpha^k(s,a)]^2 < \infty$. Following Q-learning updates, $Q^{*,k}(s,a)$ converges to $Q^*(s,a)$ as $k \to \infty$ (Watkins & Dayan (1992)). The deep version of Q-learning algorithm, Deep Q-Network (DQN), is shown in Appendix.

However, the nice property of convergence only holds in the tabular case and cannot be extended to a function approximation as discussed later.

3.2 TD-LEARNING WITH FUNCTION APPROXIMATION

To introduce generalization to the value function, we represent the approximated Q-value in a parameterized functional form. Our goal is to minimize the mean squared value error,

$$\mathcal{L}(\theta) = \frac{1}{2}\sum_{s\in\mathcal{S}} \mu(s)\Big[Q^{\text{target}} - Q_\theta(s,a)\Big]^2, \tag{13}$$

where Q^{target} is the ground truth and Q_θ is the prediction. Just like TD-learning, the Bellman residual can be applied for the value function approximation.

Semi gradient for Bellman Residual Similar to stochastic gradient methods with unbiased target estimators, if we use the Bellman Equation to get target Q-value Q^{target}, but here we just ignore its potential gradient change, the gradient ascent for Bellman residual is,

$$\begin{aligned}\Delta_{\text{semi}}\theta &= -\frac{1}{2}\alpha\nabla_\theta\Big[Q^{\text{target}} - Q_\theta(s,a)\Big]^2 \\ &= \alpha\Big[Q^{\text{target}} - Q_\theta(s,a)\Big]\nabla_\theta Q_\theta(s,a), \text{ where } Q^{\text{target}} = r + \gamma Q_{\theta}(s',a')\end{aligned} \tag{14}$$

Since we neglects a part of the gradient of Q^{target}, it is called semi gradient for Bellman residual (θ in red). Though semi-gradient methods are fast and simple, they could have divergence issue, e.g., Baird's counter-example (the star problem).

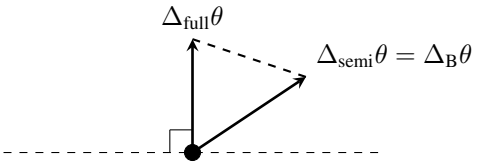
Full Gradient for Bellman Residual The full Bellman residual gradient should include all gradient components, including the gradient of the target estimation,

$$\begin{aligned}\Delta_{\text{full}}\theta &= -\frac{1}{2}\alpha\nabla_\theta\Big[r + \gamma Q_\theta(s',a') - Q_\theta(s,a)\Big]^2 \\ &= -\alpha\Big[r + \gamma Q_\theta(s',a') - Q_\theta(s,a)\Big]\Big[\gamma\nabla_\theta Q_\theta(s',a') - \nabla_\theta Q_\theta(s,a)\Big].\end{aligned} \tag{15}$$

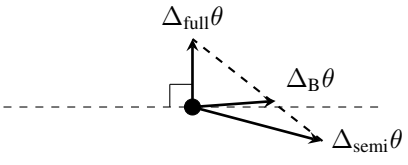
If the approximation system is general enough and the value functions are continuous, the full Bellman residual gradient is guaranteed to converge to the optima. However, this is at the sacrifice of learning speed, as illustrated by the hall problem.

Hybrid Gradient for Bellman Residual In contrast to Figure 1a where Δ_{semi} boosts Δ_{full}, Figure 1b represents the case where the semi gradient may diverge. Baird (1995) combined these 2 methods: to keep stable, Δ_{B} should stay in the same direction as Δ_{full} (above the perpendicular axis); meanwhile, Δ_{B} should stay as close as possible to Δ_{semi} to increase learning speed.

$$\begin{aligned}\Delta_{\text{B}}\theta &= (1-\omega)\cdot\Delta_{\text{semi}}\theta + \omega\cdot\Delta_{\text{full}}\theta, \\ &= -\alpha\Big[r + \gamma Q_\theta(s',a') - Q_\theta(s,a)\Big]\Big[\omega\gamma\nabla_\theta Q_\theta(s',a') - \nabla_\theta Q_\theta(s,a)\Big], \\ &\text{s.t., } \Delta_{\text{B}}\theta\cdot\Delta_{\text{full}}\theta \geqslant 0 \Leftrightarrow \omega \geqslant \frac{\Delta_{\text{semi}}\theta\cdot\Delta_{\text{full}}\theta}{\Delta_{\text{semi}}\theta\cdot\Delta_{\text{full}}\theta - \Delta_{\text{full}}\theta\cdot\Delta_{\text{full}}\theta}.\end{aligned} \tag{16}$$



(a) Gradient ascent by semi-gradient.



(b) Gradient descent by semi-gradient.

Figure 1: Epoch-wise gradient vectors for Bellman residual gradients.

4 Bellman Generalizations

4.1 Soft Bellman Equation

The Soft Bellman Equation incorporates an entropy regularization term, encouraging exploration by penalizing overly deterministic policies. It is useful where balancing exploration and exploitation is critical, such as environments with sparse rewards or high uncertainty.

$$Q(s,a) = R(s,a) + \gamma \mathbb{E}_{a' \sim \pi(\cdot|s'), s' \sim P(\cdot|s,a)} \Big[Q(s',a') - \lambda \log \pi(\cdot|s') \Big], \tag{17}$$

where λ controls the weight of the entropy term $\mathcal{H}(\cdot|s)$.

4.2 Continuous-Time Bellman Equation

Hamilton-Jacobi-Bellman (HJB) Equation is the continuous-time analogue of the Bellman Equation and used in problems involving continuous state and action spaces. Given system dynamics $f(s,a)$,

$$\frac{\partial Q(s,a)}{\partial t} + \max_a \Big[R(s,a) + \nabla_s Q(s,a)^T f(s,a) \Big] = 0. \tag{18}$$

4.3 Distributional Bellman Equation

The Distributional Bellman Equation models the return as a distribution $\mathcal{Q}$, which is particularly useful in risk-sensitive scenarios with specific demands of quantifying the variability of returns,

$$\mathcal{Q}(s,a) \overset{\text{D}}{=} R(s,a) + \gamma \mathcal{Q}(s',a'), \tag{19}$$

where $\mathcal{Q}$ is distributional expected returns, and $\overset{\text{D}}{=}$ denotes equility in distribution.

4.4 Multi-Agent Bellman Equation

The Multi-Agent (MA) Bellman Equation generalizes the Bellman Equation to cooperative MA systems, where multiple agents collaborate to achieve a common goal.

Joint Bellman Equation The Joint MA Bellman Equation models the group of agents as an entity, considering their collective actions $\mathbf{a} = (a_1, \ldots, a_N)$ and expected returns Q,

$$Q(s,\mathbf{a}) = R(s,\mathbf{a}) + \gamma \mathbb{E}_{\mathbf{a}' \sim \pi(\cdot|s'), s' \sim P(\cdot|s,\mathbf{a})} \Big[Q(s',\mathbf{a}') \Big]. \tag{20}$$

Individual Bellman Equation The Individual Bellman Equation computes Q_i for each agent, assuming that agents' execution and rewards are independent (but observing a shared reward $R(s,\mathbf{a})$),

$$Q_i(s,a_i) = R(s,\mathbf{a}) + \gamma \mathbb{E}_{a_i' \sim \pi_i(\cdot|s'), s' \sim P(\cdot|s,a_i)} \Big[Q_i(s',a_i') \Big]. \tag{21}$$

4.5 Multi-Task Bellman Equation

The Generalized Bellman Equation extends the Bellman Equation to include additional objectives. It is particularly useful in multi-task learning, constrained RL or hierarchical RL with additional objectives $\phi(s,a)$ or constraints $\phi(s,a) \leftarrow \lambda c(s,a)$ alongside the primary goal,

$$Q(s,a) = R(s,a) + \phi(s,a) + \gamma \mathbb{E}_{a' \sim \pi(\cdot|s'), s' \sim P(\cdot|s,a)} \Big[Q(s',a') \Big]. \tag{22}$$

REFERENCES

Leemon C. Baird. Residual algorithms: Reinforcement learning with function approximation. In *Machine Learning Proceedings 1995*, pp. 30–37. Elsevier, 1995.

Richard Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

Thomas Jaakkola, Michael I. Jordan, and Satinder P. Singh. On the convergence of stochastic iterative dynamic programming algorithms. *Neural Computation*, 6(6):1185–1201, 1994. doi: 10.1162/neco.1994.6.6.1185.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2nd edition, 2018.

Csaba Szepesvári. *Algorithms for Reinforcement Learning*, volume 4 of *Synthesis Lectures on Artificial Intelligence and Machine Learning*. Morgan & Claypool Publishers, 2010. ISBN 9781608454921. doi: 10.2200/S00268ED1V01Y201005AIM009.

Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3–4):279–292, 1992. doi: 10.1007/BF00992698.

A DEEP Q-NETWORK

Deep Q-Network (DQN) approximates the Q-function with a target network to stabilize training.

Algorithm 1 Deep Q-learning with Experience Replay

Initialize: parameter ϕ for Q-network Q_ϕ, target Q-network $\phi' \leftarrow \phi$, replay memory $\mathcal{D}$
for each episode **do**
  Choose s_0 from $\mathcal{S}$ as initial state
  for $t = 0$ to $T - 1$ **do**
    Take action a_t according to policy derived by Q (e.g. ϵ-greedy)
    Observe reward r_{t+1} and next state s_{t+1}
    Store transition $(s_t, a_t, r_{t+1}, s_{t+1})$ into $\mathcal{D}$
    for each mini-batch of N transitions $\{s_i, a_i, r_{i+1}, s_{i+1}\}$ **do**
      $\delta_i \leftarrow \underbrace{r_{i+1} + \gamma \max_{a'} Q_{\phi'}(s_{i+1}, a')}_{\text{target Q prediction}} - Q_\phi(s_i, a_i)$
      $\phi \leftarrow \phi + \frac{1}{N} \sum_i \alpha \delta_i \nabla_\phi Q_\phi(s_i, a_i)$
    end for
    $\phi' \leftarrow \tau \phi' + (1 - \tau)\phi$
  end for
end for
return value function Q