Multi-step learning and Value-based approximation methods

Mark Gluzman

iDDA, CUHK (Shenzhen)

January 28, 2019

Markov Decision Process

MDP is defined as $\mathcal{M} = (\mathcal{X}, \mathcal{A}, P, g)$, where

- $\mathcal{X}$ is a finite state space, $\mathcal{A} = \bigcup_{x \in \mathcal{X}} \mathcal{A}(x)$ is a finite action space.
- P is a state transition probability kernel.

  The system state at the next decision epoch is determined by

$$\mathbb{P}\Big\{x_{k+1} = y \,\Big|\, x_k = x, a_k = a\Big\} = P(x, a, y)$$

  for each $x_k \in \mathcal{X}$, $a_k \in A(x_k) \subset \mathcal{A}$.

- Given $(x_k; a_k)$, a new (random) state x_{k+1} is observed and a (one-step) cost $g(x_k; a_k; x_{k+1})$ is incurred.
- The value function of policy $\mu : \mathcal{X} \to \mathcal{A}$ is

$$J_\mu(x) = \mathbb{E}\Big[\sum_{k=0}^{\infty} \beta^k g\big(x_k, \mu(x_k), x_{k+1}\big) | x_0 = x\Big], \text{ where } 0 < \beta < 1.$$

Bellman operator

- For $J : \mathcal{X} \to \mathbb{R}$ and stationary policy μ

$$
\begin{aligned}
(T_\mu J)(x) &= \mathbb{E}_{y \sim P(\cdot|x,\mu(x))}\Big[g(x,\mu(x),y) + \beta J(j)\Big] \\
&= \sum_{y \in \mathcal{X}} P(x,\mu(x),y)\Big[g(x,\mu(x),y) + \beta J(y)\Big], \quad x \in \mathcal{X}
\end{aligned}
$$

- For $J : \mathcal{X} \to \mathbb{R}$ consider

$$
\begin{aligned}
(TJ)(x) &= \min_{a \in \mathcal{A}(x)} \mathbb{E}_{y \sim P(\cdot|x,a)}\Big[g(x,a,y) + \beta J(y)\Big] \\
&= \min_{a \in \mathcal{A}(x)} \sum_{y \in \mathcal{X}} P(x,a,y)\Big[g(x,a,y) + \beta J(y)\Big], \quad x \in \mathcal{X}
\end{aligned}
$$

Bellman equation

Theorem (Bertsekas, Proposition 1.2.4)

(a) For every stationary policy μ, the associated value function satisfies for all $i \in \mathcal{X}$:

$$J_\mu(x) = \mathbb{E}_{y \sim P(\cdot|x,\mu(x))}\Big[g(x,\mu(x),y) + \beta J_\mu(y)\Big] \quad \textit{or} \quad J_\mu = T_\mu J_\mu$$

(b) J_μ is the unique solution of equation (1).

The main purpose of this talk is to find an efficient way to estimate J_μ.

Policy evaluation: Dynamic programming (P_μ, g_μ are known.)

A fixed point problem $J_\mu = T_\mu J_\mu$ is equivalent to linear system of equations

$$J_\mu = g_\mu + \beta P_\mu J_\mu,$$

where g_μ is a $\mathcal{X}$-vector with entries $g_\mu(x) = \sum_{y \in \mathcal{X}} P(x, \mu(x), y) g(x, \mu(x), y)$ and P_μ is an $\mathcal{X} \times \mathcal{X}$ matrix with entries $P_\mu(x, y) = P(x, \mu(x), y)$.

Monte-Carlo Simulation: P_μ, g_μ are unknown

- Let $c_m(x_0) = g(x_0, x_1) + \beta g(x_1, x_2) ... + \beta^N g(x_N, x_{N+1})$ be the cumulative cost of the mth episode, s.t. x_{N+1} is a terminal state or $\frac{\beta^N}{1-\beta} \max_{x,y \in \mathcal{X}} g(x, y) \approx 0$.
- For all states $x \in \mathcal{X}$ and for all m we have

$$J_\mu(x) = \mathbb{E}[c_m(x)]$$

thus we can estimate $J_\mu(x)$ forming a sample mean:

$$J^K(x) \approx \frac{1}{K} \sum_{m=1}^{K} c_m(x) \tag{1}$$

- Equation (1) can be iteratively calculated

$$J^m(x) = J^{m-1}(x) + \gamma_m \Big(c_m(x) - J^{m-1}(x) \Big), \quad m = 1, .., K$$

where $J^0(x) = 0$ and $\gamma_m = \frac{1}{m}$.

TD(0) learning

- Bellman equation

$$J_\mu(x) = \mathbb{E}\Big[g(x,y) + \beta J_\mu(y)\Big]$$

- Assume that we have an episode $\{x_0, x_1, ..., x_N\}$ and k_m is a mth time when the state x is visited.

$$J_\mu(x) \approx \frac{1}{K} \sum_{m=1}^{K} \Big[g(x_{k_m}, x_{k_m+1}) + \beta J_\mu(x_{k_m+1})\Big]$$

- Update step for TD(0) learning

$$\begin{cases} J^k(x) = J^{k-1}(x_k) + \gamma_m \Big(g(x_k, x_{k+1}) + \beta J^{k-1}(x_{k+1}) - J^{k-1}(x_k)\Big), & \text{if } x = x_k \\ J^k(x) = J^{k-1}(x_k), & \text{if } x \neq x_k \end{cases}$$

N-step TD learning

- N-step Bellman operator:

$$(T_\mu^N J)(x_k) = \mathbb{E}\Big[\sum_{m=0}^{N} \beta^m g(x_{k+m}, x_{k+m+1}) + \beta^{N+1} J(x_{k+N+1})\Big]$$

- N-step Bellman equation:

$$(T_\mu^N J)(x) = J(x), \quad \text{for each } x \in \mathcal{X}$$

- After $k+N$ steps

$$J_\mu^{k+N}(x_k) \approx g_k + \beta g_{k+1} + \ldots + \beta^N g_{k+N} + \beta^{N+1} J_\mu^{k+N-1}(x_{k+N+1})$$

- Update step for N-step TD learning

$$\begin{cases} J_\mu^{k+N+1}(x_k) = J_\mu^{k+N}(x_k) - \gamma_k \Big[J_\mu^{k+N}(x_k) - \sum_{n=0}^{N} \beta^n g_{k+n} - \beta^N J_\mu^{k+N}(x_{k+N+1})\Big] \\ J_\mu^{k+N+1}(y) = J_\mu^{k+N}(y), \quad y \neq x_k \end{cases}$$

λ-weighted multistep Bellman equation

- Consider $N \sim$ Geometric(λ), $0 \leq \lambda < 1$.

  Define an operator:

$$T_\mu^{(\lambda)} = (1-\lambda) \sum_{n=0}^{\infty} \lambda^n T_\mu^{n+1}$$

- The $\lambda-$weighted multistep Bellman equation:

$$J_\mu(x_k) = (1-\lambda)\mathbb{E}\Big[\sum_{n=0}^{\infty} \lambda^n \Big(\sum_{m=0}^{n} \beta^m g(x_{k+m}, x_{k+m+1}) + \beta^{n+1} J_\mu(x_{k+n+1})\Big)\Big]$$

TD(λ) learning

$$
\begin{aligned}
J_\mu(x_k) &= (1-\lambda)\mathbb{E}\Big[\sum_{n=0}^{\infty}\lambda^n\Big(\sum_{m=0}^{n}\beta^m g(x_{k+m},x_{k+m+1})+\beta^{n+1}J_\mu(x_{k+n+1})\Big)\Big] \\
&= \mathbb{E}\Big[(1-\lambda)\sum_{m=0}^{\infty}\beta^m g(x_{k+m},x_{k+m+1})\sum_{n=m}^{\infty}\lambda^n+\sum_{n=0}^{\infty}\beta^{n+1}J_\mu(x_{k+n+1})(\lambda^n-\lambda^{n+1})\Big] \\
&= \mathbb{E}\Big[\sum_{m=0}^{\infty}\beta^m\lambda^m\Big(g(x_{k+m},x_{k+m+1})+\beta\lambda J_\mu(x_{k+m+1})-J_\mu(x_{k+m})\Big)\Big]+J_\mu(x_k) \\
&= \mathbb{E}\Big[\sum_{m=0}^{\infty}\lambda^m\beta^m d_{m+k}\Big]+J_\mu(x_k)
\end{aligned}
$$

Temporal Difference: $d_m = g(x_m,x_{m+1})+\beta\lambda J_\mu(x_{m+1})-J_\mu(x_m)$

TD(λ) learning

- λ-weighted Bellman equation: $J_\mu(x_k) = \mathbb{E}\Big[J_\mu(x_k) + \sum_{m=0}^{\infty} (\lambda\beta)^m d_{m+k}\Big]$
- Online iterations:

$$J(x_k) := J(x_k) + \gamma\Big[J(x_k) + \sum_{m=0}^{\infty} (\lambda\beta)^m d_{m+k} - J(x_k)\Big] = J(x_k) + \gamma \sum_{m=0}^{\infty} (\lambda\beta)^m d_{m+k}$$

- Assume we have a single infinitely long trajectory $(x_0, g_0, x_1, g_1, x_2, ...)$.
- Since we cannot afford to wait until the end of the trajectory we need an on-line version of the algorithm.

TD(λ) learning

Let J^0 is an initial guess. The first two updated are:

- Following the transition (x_0, x_1) :

  $J^1(x_0) = J^0(x_0) + \gamma d_0$

- Following the transition (x_1, x_2) :

  $$\begin{cases} J^2(x_0) = J^1(x_0) + \gamma\lambda\beta d_1 = J^0(x_0) + \gamma d_0 + \gamma\lambda\beta d_1 \\ J^2(x_1) = J^1(x_1) + \gamma d_1 = J^0(x_1) + \gamma d_1 \end{cases}$$

  If $x_0 = x_1$ there are three variants of the TD(λ) algorithm:

- The restart variant: $J^2(x_0) = J^0(x_0) + \gamma d_0 + \gamma d_1$.
- The first-visit variant: $J^2(x_0) = J^0(x_0) + \gamma d_0 + \gamma\lambda\beta d_1$
- The every-visit variant: $J^2(x_0) = J^0(x_0) + \gamma d_0 + \gamma\lambda\beta d_1 + \gamma d_1$.

TD(λ) learning

The update equation for TD(λ) becomes:

$$J^{k+1}(x) = J^k(x) + \gamma_k(x) z_k(x) d_k(x), \text{ for each } x \in \mathcal{X}$$

where $z_{-1} = 0$ and

- The restart variant: $z_k(x) = \begin{cases} 1, & \text{if } x_k = x \\ \beta\lambda z_{k-1}(x), & \text{if } x_k \neq x \end{cases}$
- The first-visit variant: $z_k(x) = \begin{cases} 1, & \text{if } x = x_k \text{ and } x_k \text{ is visited first time} \\ \beta\lambda z_{k-1}(x), & \text{otherwise} \end{cases}$
- The every-visit variant: $z_k(x) = \begin{cases} \beta\lambda z_{k-1}(x) + 1, & \text{if } x_k = x \\ \beta\lambda z_{k-1}(x), & \text{if } x_k \neq x \end{cases}$

Approximate Dynamic Programming

Approximate dynamic programming (neuro-dynamic programming, reinforcement learning):

A principle aim is to address problems with very large number of states in $\mathcal{X}$.

- $|\mathcal{X}|$-dimensional inner product are time-consuming.
- It may impossible to store $|\mathcal{X}|$-vector in a computer memory.

Approximation is value space

$\tilde{J}(x, r)$ is a function of some chosen form and $r = (r_1, ..., r_s)$ is a parameter vector of relatively small dimension s.

Examples:

- Linear form:

$$\tilde{J}(x, r) = \sum_{k=1}^{s} r_k \phi_k(x),$$

where $\phi_k : \mathcal{X} \to \mathbb{R}, \quad k = 1, ..., s$ are known as feature functions.

- Feedforward neural network with a single hidden layer with K neurons:

$$\tilde{J}(x, r) = \sum_{k=1}^{K} r(k)\sigma\Big(\sum_{l=1}^{L} r(k, l)v_l(x)\Big),$$

where state x is encoded as a L -dimensional vector $v(x)$, $\sigma(\cdot)$ is an activation function.

Linear Approximation

- Given a stationary policy μ we want to estimate

$$J_\mu(x) = \mathbb{E}\Big[\sum_{k=0}^{\infty} \beta^k g(x_k, \mu(x_k), x_{k+1}) | x_0 = x\Big]$$

- We approximate $J_\mu(x)$ with a linear architecture

$$\tilde{J}(x, r) = \sum_{k=1}^{s} r_k \phi_k(x) = \phi(x)^T r, \quad x \in \mathcal{X},$$

where $\phi(x)$ is an s-dimensional feature vector.

-

$$\tilde{J}_r = \begin{pmatrix} \phi(x_1)^T r \\ \vdots \\ \phi(x_{|X|})^T r \end{pmatrix} = \Phi r,$$

where Φ is $|\mathcal{X}| \times s$ matrix that has as rows the feature vectors $\phi(x)^T$.

The Projected Equation

- We want to approximate J_μ within space $S = \{\Phi r \mid r \in \mathbb{R}^s\}$.
- The goal is to find $\tilde{J}^*$ w.r.t. $\xi-$weighted norm:

$$\tilde{J}^* = \arg\min_{\tilde{J} \in S} \sum_{x \in \mathcal{X}} |J_\mu(x) - \tilde{J}(x, r)|^2 \xi(x) = \arg\min_{\tilde{J} \in S} ||J_\mu - \tilde{J}_r||_\xi^2$$

- The problem is equivalent to finding $r^* \in \mathbb{R}^s$ s.t.

$$r^* = \arg\min_{r \in \mathbb{R}^s} ||J_\mu - \Phi r||_\xi^2$$

- Let Π denote the projection operation onto S w.r.t the $\xi-$weighted norm.

  Then

$$\Pi J_\mu = \Phi r^*. \tag{2}$$

The Projected Bellman Equation

- We assume

$$J_\mu \approx T_\mu(\Phi r^*) \tag{3}$$

- Combing (3) and (2) we get

$$\Pi T_\mu(\Phi r^*) \approx \Phi r^*$$

- The Projected Bellman equation:

$$\Pi T_\mu(\Phi r) = \Phi r \tag{4}$$

- One can show that ΠT_μ is a contraction operator w.r.t. $||\cdot||_\xi$ norm when ξ is a stationary distribution of the DTMC with transition matrix P_μ.

The Projected Bellman Equation

- By the definition of projection the unique solution of (4) satisfies

$$r^* = \arg\min_{r \in \mathbb{R}^s} ||\Phi r - (g_\mu + \beta P_\mu \Phi r^*)||_\xi^2$$

- By setting to 0 the gradient w.r.t. r we obtain

$$\Phi^T D(\Phi r^* - (g_\mu + \beta P_\mu \Phi r^*)) = 0, \tag{5}$$

where D is the diagonal matrix with ξ along the diagonal.

- Equation (5) can be compactly written as

$$Cr^* = d,$$

where $C = \Phi^T D(I - \beta P_\mu)\Phi$ and $d = \Phi^T D g_\mu$.

TD(0) with linear approximation

- $Cr - d = \Phi^T D(I - \beta P_\mu)r - \Phi^T D g_\mu = \mathbb{E}\Big[\phi(x)\Big(\phi(x)^T r - \beta\phi(y)^T r - g_\mu(x,y)\Big)\Big]$
- Equation $Cr - d = 0$ is equivalent to

$$r = r - \gamma(Cr - d) = r - \gamma\mathbb{E}\Big[\phi(x)\Big(\phi(x)^T r - \beta\phi(y)^T r - g(x,y)\Big)\Big]$$

- Given an episode $(x_0, g_0, x_1, g_1, ..., x_N, g_N)$, the TD(0) iteration is

$$r_{k+1} = r_k - \gamma_k\phi(x_k)\Big(\phi(x_k)^T r_k - \beta\phi(x_{k+1})^T r_k - g_k\Big)$$

Convergence of TD(0) with linear approximation

Theorem (Tsitsiklis, Van Roy, 1997)

Assume that

- *the Markov chain assosiated with policy μ is irreducible and aperiodic*
- *the steady-state variance of transition costs is finite $\mathbb{E}[g^2(x_k, x_{k+!})] < \infty$.*
- *the learning rate is s.t.*

$$\sum_{k=0}^{\infty} \gamma_k = \infty \text{ and } \sum_{k=0}^{\infty} \gamma_k^2 < \infty$$

Then

- *TD(0) converges to r^* that is a unique solution of $\Pi T(\Phi r) = \Phi r$.*
- *r^* satisfies*

$$||\Phi r^* - J_\mu||_\xi \leq \frac{1}{1-\beta}||\Pi J_\mu - J_\mu||_\xi$$

Finite Sample Analyses for TD(0) with Function Approximation

- **Gal Dalal et. al., 2017** found convergence rate under assumption that one can generate iid samples from a steady-state distribution, using recently developed stochastic approximation techniques.
- **Bhandari, Russo, Singal, 2018** found convergence rate for projected TD(0) algorithm (r_k is assumed to be $||r_k|| < R$) using information theoretic techniques.
- **Lei Ying, 2018** found convergence rate for TD(0) algorithm using Stein's Method.

LSTD(0)

- $C = \mathbb{E}\Big[\phi(x)\Big(\phi(x) - \beta\phi(y)\Big)^T\Big]$ and $d = \mathbb{E}[\phi(x)g(x,y)]$
- Based on simulation

$$C_N = \frac{1}{N+1}\sum_{k=0}^{N}\phi(x_k)\Big(\phi(x_k) - \beta\phi(x_{k+1})\Big)^T \text{ and } d_N = \frac{1}{N+1}\sum_{k=0}^{N}\phi(x_k)g(x_k, x_{k+1})$$

- LSTD(0) algorithm: simulate N time-steps according to a policy μ

$$\begin{cases} C_{k+1} = C_k - \frac{1}{k}\Big(\phi(x_k)\Big(\phi(x_k) - \beta\phi(x_{k+1})\Big)^T - C_k\Big) \\ d_{k+1} = d_k - \frac{1}{k}\Big(\phi(x_k)g(x_k, x_{k+1}) - d_k\Big) \end{cases}$$

After the end of simulation:

$$\tilde{r} = C_N^{-1}d_N \qquad J_\mu(x) \approx \phi^T(x)\tilde{r}$$

LSPE(0)

- LSPE method is based on an idea of Projected Value Iteration

$$\Phi r_{k+1} = \Pi T_\mu(\Phi r_k) \tag{6}$$

- Equation (6) is equivalent to $r_{k+1} = r_k - (\Phi^T D \Phi)^{-1}(C r_k - d)$
- a simulation-based implementation:

$$r_{k+1} = r_k - G_k(C_k r_k - d_k),$$

where

$$\begin{cases} C_{k+1} = C_k - \frac{1}{k}\left(\phi(x_k)\left(\phi(x_k) - \beta\phi(x_{k+1})\right)^T - C_k\right) \\ d_{k+1} = d_k - \frac{1}{k}\left(\phi(x_k)g(x_k, x_{k+1}) - d_k\right) \\ G_{k+1} = \left(\frac{1}{k+1}\sum_{t=0}^{k}\phi(x_t)\phi(x_t)^T\right)^{-1} = G_k - \frac{G_k\phi(x_k)\phi^T(x_k)G_k}{1+\phi^T(x_k)G_k\phi(x_k)} \end{cases}$$

Limitations

Limitations:

- No guaranty that TD(λ) with non-linear approximation will converge. Counterexample in Tsitsiklis, Van Roy, 1997.
- Policy improvement may not converge to the near-optimal value function approximation: no guaranty that $\tilde{J}_\mu-$greedy policy is better than μ. Policy oscillation-chattering often occurs (see, Bertsekas, Chapter 6.4.3).

References

- Tsitsiklis, Van Roy (1997) An Analysis of Temporal-Difference Learning with Function Approximation, IEEE Transactions on Automatic Control, 42 (5), pp 674–690
- Bertsekas, D. P. (2012). Dynamic Programming and Optimal Control, Volume 2: Approximate Dynamic Programming, 4th edition. Athena Scientific, Belmont.
- Gal Dalal et. al. (2017) Finite Sample Analyses for TD(0) with Function Approximation, The Thirty-Second AAAI Conference on Artificial Intelligence.
- Bhandari et.al. (2018) A Finite Time Analysis of Temporal Difference Learning With Linear Function Approximation, https://arxiv.org/abs/1806.02450
- Lei Ying (2018) Stein's Method for Big-Data Systems: from Learning Queues to Q-learning, Mathematical Issues in Information Sciences.