Reinforcement learning and function approximation

Alexandre Bouchard

June 16, 2004

Contents

1 Problem formulation

- Discrete time
- $\forall$ stage $t \in \{0, 1, 2, \dots\}$ the agent is in some state $i_t \in S$
- It chooses an action $u \in U_{i_t}$
- Assume that $A := \cup_{s \in S} U_s$ and S are finite
- Represent the transition probability from state i to j given that u is selected by $p_{ij}(u)$
- Cost for transition from state i_t to i_{t+1} given that u is selected is denoted by $g(i_t, u, i_{t+1})$

The goal of the agent is to find a *policy*

$$\pi := \big(\mu_k : S \to A\big)_{k=0}^{\infty}$$

that minimizes the expected long-term cost:

$$E\Big[\sum_{t=0}^{\infty} \alpha^t g(i_t, \mu_t(i_t), i_{t+1})\Big]$$

the expectation is taken with respect to the distribution of the Markov chain $(i_0, i_1, \dots)$ characterized by the transition probabilities $p_{i_t, i_{t+1}}(\mu_t(i_t))$ of the problem and $0 < \alpha \leq 1$ is a fixed *discount factor*.

For today we will consider *stationary* policies, that is policies such that:

$$\mu := (\mu, \mu, \dots)$$

Cost-to-go functions:

$$J^{\mu}(i) := \liminf_{N \to \infty} E\Big[\sum_{t=0}^{N} \alpha^t g(i_t, \mu(i_t), i_{t+1}) | i_0 = i\Big]$$

Optimal cost-to-go function:

$$J^{\star}(i) := min_{\pi} J^{\mu}(i)$$

Linear approximators

$$\tilde{J}(i, \vec{r}) := \sum_{k=1}^{K} \vec{r}[k]\phi_k(i)$$

where

$$\vec{r} \in \mathbb{R}^K$$

is a vector of tunable parameters and

$$\phi_k : S \to \mathbb{R}$$

are the *basis functions*.

Suppose we are given a RL problem with S, A, α known, $p(u)$, g unknown initially and that for a fixed policy μ the agent is able to generate trajectories $(i_t : i_t \in S)_{t=0}^{\infty}$ according to the transition probabilities $p_{i_t,i_{t+1}}(\mu(i_t))$.

A possible strategy to solve this problem:

1. Start with some policy μ
2. Evaluate for this fixed policy its cost-to-go vector J^{μ}
3. Apply some improvement to the current policy
4. If the policy can still be improved, go to step 2 (exist only a finite number of policies)

Today, we focus on step 2, namely the evaluation of the cost-to-go vector for a fixed policy.

2 Convergence of approximate $TD(\lambda)$

Proof based on:
Bertsekas, D.P. & Tsitsiklis, J.N. (1996). Neuro-Dynamic Programming.

Theorem 1 *Assumptions:*

- *RL problem as stated before and a fixed stationary policy* μ
- *A linear approximation architecture with bounded basis functions. For convenience, define:*

$$\vec{\phi}(i) := (\phi_1(i), \ldots, \phi_K(i))$$

and also the n *by* K *matrix:*

$$\Phi := \begin{pmatrix} | & & | \\ \phi_1 & \ldots & \phi_K \\ | & & | \end{pmatrix} = \begin{pmatrix} - & \vec{\phi}(1)^T & - \\ & \vdots & \\ - & \vec{\phi}(n)^T & - \end{pmatrix}$$

- *The update rule is:*

$$\vec{r}_{t+1} := \vec{r}_t + \gamma_t d_t \vec{z}_t$$

In the update rule:

$$\vec{r}_{t+1} := \vec{r}_t + \gamma_t d_t \vec{z}_t$$

d_t *is called the* temporal difference

$$d_t := g(i_t, i_{t+1}) + \alpha \tilde{J}(i_{t+1}, \vec{r}_t) - \tilde{J}(i_t, \vec{r}_t)$$

In the update rule:

$$\vec{r}_{t+1} := \vec{r}_t + \gamma_t d_t \vec{z}_t$$

$\vec{z}_t$ *is called the* eligibility vector

$$\vec{z}_t := \sum_{k=0}^{t} (\alpha\lambda)^{t-k} \nabla \tilde{J}(i_t, \vec{r}_t)$$

$$= \sum_{k=0}^{t} (\alpha\lambda)^{t-k} \vec{\phi}(i_k)$$

Note that $\vec{z}_{t+1}$ *can be computed easily by:*

$$\vec{z}_{t+1} := \alpha\lambda\vec{z}_t + \vec{\phi}(i_{t+1})$$

In the update rule:

$$\vec{r}_{t+1} := \vec{r}_t + \gamma_t d_t \vec{z}_t$$

γ_t *is called the* step size sequence. *It is assumed to satisfy:*

$$\sum_{t=0}^{\infty} \gamma_t = \infty$$

$$\sum_{t=0}^{\infty} \gamma_t^2 < \infty$$

Moreover, we assume that:

1. *There exists positive numbers $\vec{\pi}[j]$ such that:*

$$\lim_{t \to 0} P(i_t = j | i_0 = i) = \vec{\pi}[j] \qquad \forall i, j$$

2. *We have $K \leq n$ and Φ has full rank.*

Then, under these assumptions, the sequence $\vec{r}_t$ converges, with probability 1.

Lemma 1 *Let X_t be a Markov process taking values in a set S. Define the maps:*

$$\begin{cases} A : S \to M_n(\mathbb{R}) \\ \vec{b} : S \to \mathbb{R}^n \end{cases}$$

Then the sequence $\vec{r}_t$ defined by:

$$\vec{r}_{t+1} := \vec{r}_t + \gamma_t(A(X_t)\vec{r}_t + \vec{b}(X_t))$$

converges with probability one, provided that:

1. *The sequence of stepsize* γ_t *is deterministic and satisfies* $\sum_{t=0}^{\infty} \gamma_t = \infty$ *and* $\sum_{t=0}^{\infty} \gamma_t^2 < \infty$.
2. *The Markov process* X_t *has an invariant distribution. Let* $E_0\big[\,\cdot\,\big]$ *denotes the expectation with respect to this distribution.*
3. *We have* $A := E_0\Big[A(X_t)\Big]$ *is negative definite.*
4. *There is a* K *such that for all* X_t, $\|A(X_t)\| \leq K$, $\|\vec{b}(X_t)\| \leq K$.
5. *There are* ρ, C *such that:*

$$\|E\Big[A(Xt)|X_0\Big] - A\| \leq C\rho^n \qquad \forall n \geq 0, X_0$$

and

$$\|E\Big[\vec{b}(Xt)|X_0\Big] - \vec{b}\| \leq C\rho^n \qquad \forall n \geq 0, X_0$$

where $\vec{b} := E_0\Big[\vec{b}(X_t)\Big]$.

We now reduce the problem of convergence of $TD(\lambda)$ to the the case of a stochastic iterative algorithm with Markov noise. Let:

$$X_t := (i_t, i_{t+1}, \vec{z}_t)$$

This is indeed a Markov process. Recall that:

$$\vec{z}_{t+1} := \alpha\lambda\vec{z}_t + \vec{\phi}(i_{t+1})$$

Define the maps:

$$\begin{cases} \vec{b}(X_t) := \vec{z}_t g(i_t, i_{t+1}) \\ A(X_t) := \vec{z}_t(\alpha\vec{\phi}(i_{t+1})^T - \vec{\phi}(i_t)^T) \end{cases}$$

Check it works:

$$\begin{aligned} \vec{r}_{t+1} :&= \vec{r}_t + \gamma_t(A(X_t)\vec{r}_t + \vec{b}(X_t)) \\ &= \vec{r}_t + \gamma_t(\left[\vec{z}_t(\alpha\vec{\phi}(i_{t+1})^T - \vec{\phi}(i_t)^T)\right]\vec{r}_t + \left[\vec{z}_t g(i_t, i_{t+1})\right]) \\ &= \vec{r}_t + \gamma_t\vec{z}_t(\alpha\vec{\phi}(i_{t+1})^T\vec{r}_t - \vec{\phi}(i_t)^T\vec{r}_t + g(i_t, i_{t+1})) \\ &= \vec{r}_t + \gamma_t d_t \vec{z}_t \end{aligned}$$

Lemma 2 *Let:*

$$A := E_0\Big[A(X_t)\Big]$$

where $E_0\big[\cdot\big]$ *is the expectation with respect to the stationary distribution. We claim that:*

$$A = \Phi^T D(M - I)\Phi$$

where:

$$M := (1-\lambda)\sum_{m=0}^{\infty}\lambda^m(\alpha P)^{m+1}$$

and

$$D := \begin{pmatrix} \vec{\pi}[1] & & \\ & \ddots & \\ & & \vec{\pi}[n] \end{pmatrix}$$

We define the *weighted quadratic norm* as follow:

$$\|\vec{J}\|_D^2 := \vec{J}^T D \vec{J}$$

where

$$D := \begin{pmatrix} \vec{\pi}[1] & & \\ & \ddots & \\ & & \vec{\pi}[n] \end{pmatrix}$$

3 Sparse Distributed Memories (SDM)

Kanerva, P. (1993). Sparse distributed memory and related models. In M. Hassoun (Ed.), Associative neural memories: Theory and implementation, 50-76. N.Y.: Oxford University Press.

Sutton, R.S. & Barto, A. G. (1998). Reinforcement learning. An introduction. Cambridge, MA: The MIT Press.

Elements of an SDM architecture:

1. Start with some *active locations* $H := \{\vec{h}_k\}$ uniformly distributed in the space. Active locations are points in the state space with an associated *weight* w_k

2. To predict the value of $\vec{x} \in S$, we use the formula:

$$\tilde{J}(\vec{x}) = \frac{\sum_{h_k \in H} \mu(\vec{h}_k, \vec{x}) w_k}{\sum_{h_k \in H} \mu(\vec{h}_k, \vec{x})}$$

   Where μ is a similarity measure. An example of similarity measure: *symmetric triangular functions*

$$\begin{cases} \mu(\vec{h}, \vec{x}) := min_{i=1,\ldots,n} \mu_i(\vec{h}, \vec{x}) \\ \mu_i(\vec{h}, \vec{x}) := \begin{cases} 1 - \frac{|x_i - h_i|}{\beta_i} & if \qquad |x_i - h_i| \leq \beta_i \\ 0 & otherwise \end{cases} \end{cases}$$

3. We also periodically rearrange the distribution of the active locations during the execution of the algorithm.