GQ(λ) Quick Reference Guide

Adam White and Richard S. Sutton

July 7, 2014

This document should serve as a quick reference for the linear GQ(λ) off-policy learning algorithm. We refer the reader to Maei and Sutton (2010) for a more detailed explanation of the intuition behind the algorithm and convergence proofs. If you questions or concerns about the content in this document or the attached java code please email adam.white@ualberta.ca.

1 Requirements and Setting

For each use of GQ(λ) you need to provide the following three *question functions*. (In the following $\mathcal{S}$ and $\mathcal{A}$ denote the sets of states and actions.)

- $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$; target policy to be learned. Incidently, if π is chosen as the greedy policy with respect to the learned value function, then the algorithm will implement a generalization of the Greedy-GQ algorithm (Maei, Szepesvari, Bhatnagar & Sutton 2010).
- $\gamma : \mathcal{S} \rightarrow [0, 1]$; termination function ($\gamma(s) = 1 - \beta(s)$ in GQ paper)
- $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$; reward function

In the earlier publications there was also specified a fourth question function, the terminal reward function $z : \mathcal{S} \rightarrow \mathbb{R}$ used to specify a final reward at termination. Since that time it has been recognized that this functionality can be included in the reward function, making use of the termination function (Modayil, White & Sutton 2014). For example, if one wanted only a terminal reward function $z(s)$ upon termination in state s, one would use a reward function of $r(s, a, s') = (1 - \gamma(s'))z(s')$. This completes the specification of the predictive question that you are seeking to answer using the GQ(λ) algorithm.

The specific approximate answer found by GQ(λ) will depend upon the following four *answers functions* (these also must be provided):

- $b : \mathcal{S} \times \mathcal{A} \to [0, 1]$; behavior policy
- $I : \mathcal{S} \times \mathcal{A} \to [0, 1]$; interest function (can set to 1 for all state-action pairs or indicate selected state-action pairs to be best approximated)
- $\boldsymbol{\phi} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^n$; feature-vector function
- $\lambda : \mathcal{S} \to [0, 1]$; bootstrapping or eligibility-trace decay-rate function

The following data structures are internal to GQ:

- $\boldsymbol{\theta} \in \mathbb{R}^n$; the learned weights of the linear approximation: $Q^\pi(s, a) = \boldsymbol{\theta}^\top \boldsymbol{\phi}(s, a) = \sum_{i=1}^n \boldsymbol{\theta}_i \boldsymbol{\phi}_i(s, a)$
- $\boldsymbol{w} \in \mathbb{R}^n$; secondary set of learned weights
- $\boldsymbol{e} \in \mathbb{R}^n$; eligibility trace vector

Parameters internal to GQ:

- α; step-size parameter for learning $\boldsymbol{\theta}$
- $\eta \in [0, 1]$; relative step-size parameter for learning $\boldsymbol{w}$ ($\alpha\eta$)

2 Algorithm Specification

We can now specify GQ(λ). Let $\boldsymbol{w}$ and $\boldsymbol{e}$ be initialized to zero and $\boldsymbol{\theta}$ be initialized arbitrarily. Let the subscript t denote the current time step. Let ρ_t denote the "importance sampling" ratio:

$$\rho_t = \frac{\pi(S_t, A_t)}{b(S_t, A_t)}, \tag{1}$$

where $S_t \in \mathcal{S}$ and $A_t \in \mathcal{A}$ are the state and action occuring on time step t. Let $\bar{\boldsymbol{\phi}}_t$ denote the expected next feature vector:

$$\bar{\boldsymbol{\phi}}_t = \sum_{a \in \mathcal{A}} \pi(S_t, a) \boldsymbol{\phi}(S_t, a) \tag{2}$$

The following equations fully specify GQ(λ):

$$\delta_t = r(S_t, A_t, S_{t+1}) + \gamma(S_{t+1}) \boldsymbol{\theta}_t^\top \bar{\boldsymbol{\phi}}_{t+1} - \boldsymbol{\theta}_t^\top \boldsymbol{\phi}(S_t, A_t) \tag{3}$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \left[\delta_t \boldsymbol{e}_t - \gamma(S_{t+1})(1 - \lambda(S_{t+1}))(\boldsymbol{w}_t^\top \boldsymbol{e}_t) \bar{\boldsymbol{\phi}}_{t+1} \right] \tag{4}$$

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \alpha\eta [\delta_t \boldsymbol{e}_t - (\boldsymbol{w}_t^\top \boldsymbol{\phi}(S_t, A_t)) \boldsymbol{\phi}(S_t, A_t)] \tag{5}$$

$$\boldsymbol{e}_t = I(S_t) \boldsymbol{\phi}(S_t, A_t) + \gamma(S_t) \lambda(S_t) \rho_t \boldsymbol{e}_{t-1} \tag{6}$$

3 Pseudocode

The following pseudocode characterizes the algorithm and its use.

Initialize $\boldsymbol{\theta}$ arbitrarily and $\boldsymbol{w} = 0$
Repeat (for each episode):
    Initialize $\boldsymbol{e} = 0$
    $S \leftarrow$ initial state of episode
    Repeat (for each step of episode):
        $A \leftarrow$ action selected by policy b in state S
        Take action A, observe next state, S'
        $\bar{\boldsymbol{\phi}} \leftarrow 0$
        For all $a \in \mathcal{A}(s)$:
            $\bar{\boldsymbol{\phi}} \leftarrow \bar{\boldsymbol{\phi}} + \pi(S', a)\boldsymbol{\phi}(S', a)$
        $\rho = \frac{\pi(S,A)}{b(S,A)}$
        GQlearn$(\boldsymbol{\phi}(S, A), \bar{\boldsymbol{\phi}}, \lambda(S'), \gamma(S'), r(S, A, S'), \rho, I(S))$
        $S \leftarrow S'$
    until S' is terminal

GQLearn$(\boldsymbol{\phi}, \bar{\boldsymbol{\phi}}, \lambda, \gamma, R, \rho, I)$
    $\delta \leftarrow R + \gamma\boldsymbol{\theta}^\top\bar{\boldsymbol{\phi}} - \boldsymbol{\theta}^\top\boldsymbol{\phi}$
    $\boldsymbol{e} \leftarrow \rho\boldsymbol{e} + I\boldsymbol{\phi}$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha(\delta\boldsymbol{e} - \gamma(1-\lambda)(\boldsymbol{w}^\top\boldsymbol{e})\bar{\boldsymbol{\phi}})$
    $\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha\eta(\delta\boldsymbol{e} - (\boldsymbol{w}^\top\boldsymbol{\phi})\boldsymbol{\phi})$
    $\boldsymbol{e} \leftarrow \gamma\lambda\boldsymbol{e}$

4 Code

The files `GQlambda.java` and `GQlambda.cpp` contain implementations of the GQlearn function described in the pseudocode. We have excluded optimizations (e.g., binary features or efficient trace implementation) to ensure the code is simple and easy to understand. We leave it to the reader to provide environment code for interfacing to GQ(λ) (e.g., using RL-Glue).

5 References

Maei, H. R., Szepesvari, Cs., Bhatnagar, S., Sutton, R. S. (2010). Toward Off-Policy Learning Control with Function Approximation. In *Proceedings of the 27th International Conference on Machine Learning*, Haifa, Israel.

Maei, H. R. and Sutton, R. S. (2010). GQ(λ): A general gradient algorithm for temporal-difference prediction learning with eligibility traces. In *Proceedings of the Third Conference on Artificial General Intelligence 1*: 91–96.

Modayil, J., White, A., Sutton, R. S. (2014). Multi-timescale nexting in a reinforcement learning robot. *Adaptive Behavior 22*(2):146–160.

Sutton, R. S., Barto, A. G. (1998). Reinforcement Learning: An Introduction. MIT Press.