A User's Guide to GQ,
a Universal-Predictive-Knowledge Learning Algorithm

Rich Sutton

with thanks to
Hamid Maei, Csaba Szepesvari, Doina Precup, Shalabh Bhatnagar, David Silver, Michael Delp, and Eric Weiwiora

perhaps intelligence is just:

- knowing a lot of stuff
- being able to apply it to meet your goals

and perhaps knowledge is just *predictions*

- of what will happen
- of what you could cause to happen
  - at various time scales
  - conditional on actions or courses of action

predictions can be more powerful than you think

- not just one-step predictions!
- predictions can be about the outcomes of extended courses of behavior
- all scientific knowledge can be expressed as predictions
- all the little things you know can be well thought of as prediction

much of mind is about prediction

- *perception* and *state representation* can be thought of as making predictions
- *models the world* and *cause and effect* can be thought of in terms of predictions
- *planning* can be thought of as composing predictions to anticipate possible futures, and then choosing among them
- *learning value functions* (and thus much of conventional RL) is learning predictions

perhaps...

- predictions are the coin of the mental realm

- thus, we should focus on machinery for efficiently learning predictions

parallel predictions

- we should be able to make *and learn* lots of predictions at once, in parallel
  - as in prediction *demons*
- *every* demon should be able to learn on *every* step
- this has always been the promise of off-policy temporal-difference learning

this promise has been unfulfilled, until now

- there has been no practical algorithm for parallel prediction learning
  - previous methods were too complex (LSTD, iLSTD), restricted to table lookup (Q-learning), not parallel (Monte Carlo, Sarsa), too slow (importance sampling), or had weak approximators (averaging)
- until this year, at UofA (GTD, TDC, GQ)

- now, for the first time, it is straightforward to do massive, in parallel, prediction learning
- with GQ

Outline

- Perhaps intelligence is just knowing a lot, which is making lots of predictions, which is what TD learning is made for, and which we can now, finally, do, with GQ
- Setting (online prediction learning)
  - Watkins's linear Q(λ)
- GQ inputs and outputs
- The GQ algorithm
- Example uses of GQ

Setting

Real-time, incremental

- all predictions are *made* and *learned* on every time step
  - 100 times a second
  - constant-time computation per step
  - constant memory

a continuous stream of *experiential data*

- every time step, we see a new state, action, and observation, s_t, a_t, o_{t+1}, …
  - s_t is the *agent state*—whatever the agent uses, updated by $s_{t+1} = u(s_t, a_t, o_{t+1})$
  - a_t is from a fixed (for now) known behavior policy, $b(s,a) = \Pr\left[a_t = a \middle| s_t = s\right]$
  - o_{t+1} is from the world

Approximation architecture is fixed and linear (for now)

$y^i_t(s,a) = \theta^T_{it}\varphi(s,a)$ = the *i*th prediction

θ_{it} = vector of parameters for the *i*th prediction at time *t*

$\varphi(s,a)$ = vector of features representing s, a

all predictions will learn in independently, in parallel,

thus we can drop the *i* and just talk about y_t and θ_t

Watkins's Q(λ)

Watkins's Q(λ) semantics (what is learned)

learns

$$y_t(s,a) = \theta_t^T \varphi(s,a) \approx Q^*(s,a)$$

where

$$Q^*(s,a) = \max_\pi E\left[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots \,\middle|\, s_t = s, a_t = a, \pi \right]$$

Watkins's Q(λ) learning rule

$$\Delta\theta_t = \alpha\delta_t \mathrm{e}_t$$

(scalar) $$\delta_t = r_{t+1} + \max_a \gamma\,\theta_t^T\varphi(s_{t+1}, a) - \theta_t^T\varphi(s_t, a_t)$$

(vector) $$\mathrm{e}_t = \begin{cases} \gamma\lambda \mathrm{e}_t + \varphi(s_t, a_t) & \text{if } a_t = \operatorname{argmax}_a \theta_t^T\varphi(s_t, a) \\ \varphi(s_t, a_t) & \textit{otherwise} \end{cases}$$

- unsound (may diverge)
- traces frequently cut
- off-policy - learns the optimal policy and values *independent of the behavior policy*

Watkins's Q(λ) provides a hint of the massive generalizations that are possible

- you could learn simultaneously about multiple, different
  - reward functions
  - discount rates
  - non-greedy target policies
  - state-dependent terminations
- you could predicted outcomes as well as rewards

GQ

GQ(λ,π,β,z,ζ) semantics

- eligibility-trace decay function, $\lambda : S \rightarrow [0,1]$
- target policy, $\pi : S \times A \rightarrow [0,1], \sum_a \pi(s,a) = 1, \forall s \in S$
- termination probability, $\beta : S \rightarrow [0,1]$
- outcome target function, $z : S \rightarrow \Re$
- transient target function, $\zeta : S \rightarrow \Re$

$$y_t(s,a) = E\left[\zeta_{t+1} + \zeta_{t+2} + \cdots + \zeta_T + z_T \,\middle|\, s_t = s, a_t = a, \pi, \beta\right]$$

(MC form)

Notational shortcuts

$$\lambda_t = \lambda(s_t)$$

$$\beta_t = \beta(s_t)$$

$$z_t = z(s_t)$$

$$\zeta_t = \zeta(s_t)$$

$$\bar{\varphi}_t = \bar{\varphi}(s_t)$$

$$\bar{\varphi}(s) = \sum_a \pi(s,a)\varphi(s,a) = E\left[\varphi(s,a) \middle| a \sim \pi, s\right]$$

GQ learning rule

$$\delta_t = \zeta_{t+1} + \beta_{t+1} z_{t+1} + (1 - \beta_{t+1}) \mathbf{v}_t^T \bar{\mathbf{x}}_{t+1} - \mathbf{v}_t^T \mathbf{x}(s_t, a_t)$$

$$\mathbf{e}_t = \mathbf{x}(s_t, a_t) + (1 - \beta_t) \lambda_t \frac{\pi(s_t, a_t)}{b(s_t, a_t)} \mathbf{e}_{t-1}$$

$$\Delta \mathbf{v}_t = \alpha_1 \left[\delta_t \mathbf{e}_t - (1 - \beta_{t+1})(1 - \lambda_{t+1})(\mathbf{u}_t^T \mathbf{e}_t) \bar{\mathbf{x}}_{t+1} \right]$$

$$\Delta \mathbf{u}_t = \alpha_2 \left[\delta_t \mathbf{e}_t - (\mathbf{u}_t^T \mathbf{x}(s_t, a_t)) \mathbf{x}(s_t, a_t) \right]$$

- everything is O(#features)
- everything is well-defined and readily available
- similarities to expected Sarsa, with (1-β) in place of γ

GQ learning code

```
void GQlearn(x,xnext,λ,β,z,ζ,ρ) {/* all except xs are scalar */
    static double w[n],v[n],e[n];
    double α=0.0001,η=1.0,dotux,dotue;
    δ = ζ + β*z + (1-β)*dot(v,xnext) - dot(v,x);
    for (i=0;i<n;i++) e[i] = ρ*e[i] + x[i];
    dotue = dot(u,e);
    dotux = dot(u,x);
    for (i=0;i<n;i++) {
        v[i] += α * (δ*e[i] - (1-β)*(1-λ)*dotue*xnext[i]);
        u[i] += α*η * (δ*e[i] - dotux*x[i]);
        e[i] *= (1-β) * λ;
    }
}
```

Theoretical statements

There exists a scalar objective function

$$J(\theta) \quad = \quad \left\| y_\theta - \Pi T_\pi^{\lambda\beta} y_\theta \right\|_{D_b}^2$$

such that

$$E\left[\Delta\theta \middle| b\right] \quad \propto \quad \nabla_\theta \, J(\theta)$$

which guarantees convergence to $J(\theta) = 0$
(under step-size conditions)

Example uses of GQ

Bump anticipator

Continuously predict imminent bumps at various time scales under the behavior policy

$$\pi(s,a) = b(s,a)$$

$$z(s) = \left\| accelerometer(s) \right\|$$

$$\zeta(s) = 0$$

$$\beta(s) = 0.1 \quad \text{(bump in next 10 steps = 0.1 seconds)}$$

$$\beta(s) = 0.02 \quad \text{(bump in next 50 steps = 0.5 seconds)}$$

$$\beta(s) = 0.002 \quad \text{(bump in next 500 steps = 5 seconds)}$$

$$\lambda(s) = 0.9$$

three different predictions

These 3 predictions could then be added to the agent state, used to make decisions

“Near something” detector

Can I get any IR sensor to give a sustained high reading without moving very much?

Add detector $d(s)$ for high IR reading for 0.5 seconds

$$z(s) = d(s)$$

$$\zeta(s) = 0$$

$$\pi(s,a) = 1 \text{ if } a = \arg\max_{a'} y_t(s,a') \text{ otherwise } 0$$

$$\beta(s) = d(s) \; (+\,0.01 \text{ if wheels are moving})$$

$$\lambda(s) = 0.95$$

Does not need to be run to completion

Does my wall-following procedure keep me near something?

Use "near something" detector $n(s)$ learned in previous example

$z(s) = n(s)$

$\zeta(s) = 0$

$\pi =$ hand-coded wall-following policy

$\beta(s) = 0.002$ (terminates after about 5 seconds)

$\lambda(s) = 0.9$

How far am I from a corner? or from the battery charger?

Make a corner detector $c(s)$ (two separated IRs high at the same time) or charger detector $c(s)$ (power trickling into battery)

$$z(s) = 0$$

$$\zeta(s) = -1$$

$$\pi(s, a) = 1 \text{ if } a = \arg\max_{a'} y(s, a') \text{ otherwise } 0$$

$$\beta(s) = c(s)$$

$$\lambda(s) = 0.95$$

Other possibilities

- Is there a ball (concave object) present?
- Am I stuck? (what would happen to wheel motion when torque is applied after/for several time steps?)
- Can I get the rattle to sound? How?
- target policies with constant actions may be useful

Learning a model of an option

For planning we need *models* of possible courses of action (e.g., wall following)

Each such *option model* is a bunch of predictions for the option's π, β:

Predictions whose outcome targets are the elements of the agent state: $z^i(s) = s(i), \zeta^i(s) = 0$

One more prediction whose transient target is $\zeta(s) = r(s) - \bar{r}$, the deviation of the current reward from the long-term average reward

This form is necessary and sufficient for planning

Conclusions

- GQ uses a powerful language for predictions and knowledge
  - that still challenges our intuitions
- GQ can learn online, in parallel, and computationally efficiently
- It can certainly be used to learn a lot of stuff about one's world
  - probably more than any previous AI
  - what will we do with all the knowledge?