personal perspective

- there is a *science of mind* that is neither natural science nor applications technology
  - e.g., Marr's "computational theory" level
- a mind is something more usefully thought of in terms of goals than mechanisms
  - goals can be well thought of as rewards
- reinforcement learning is the beginning of a science of mind
- intrinsic motivation is part of a science of mind

Core Learning Algorithms for Intrinsically Motivated Agents

Rich Sutton
University of Alberta

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

summary

- a *technical breakthrough* leading to a new family of reinforcement learning algorithms
  - temporal-difference (TD) learning with approximations is now straightforward
  - it is now practical to learn in parallel about many predictions at once
- *parallel prediction learning* may be key to understanding mind, brain, and intrinsic motivation

knowledge is *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 a "saying before" of what the sensory signals will be
- all scientific knowledge can be expressed as predictions
- predictions can be about the outcomes of extended courses of behavior (options)
- 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

- predictions are the coin of the mental realm

- thus, we should be focused on machinery for efficiently learning predictions

parallel prediction demons

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

but this promise has been unfulfilled

- 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 now

- now, for the first time, it is practical and straightforward to do massive, in parallel, prediction learning
- with new gradient-based TD algorithms
  - GTD, TDC (NIPS-08, ICML-09, NIPS-09)
  - GQ (submitted)
  - Actor-critic-option algorithms (in prep)

outline

- Perhaps key to mind is being able to make and learn a lot of predictions in parallel, which is what TD learning was 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; including for IM

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 continual stream of *experiential data*

- every time step (say 10ms) we receive a new sensory observation o_t and take an action a_t
- we also update an *agent state representation*, s_t:

$$s_{t+1} = u(s_t, a_t, o_{t+1})$$

- s_t is whatever the agent uses to pick a_t, i.e., there exist probabilities

$$\Pr\left[a_t = a \middle| s_t = s\right]$$

- we place no constraints on s, u, or b

approximation architecture is fixed and linear (for now)

- state-action pairs map in a fixed way to feature vectors

$$s_t \; a_t \quad \rightarrow \quad \mathbf{x}_t \quad = \quad (x_t(1), x_t(2), \ldots, x_t(n)) \quad \in \Re^n$$

- which map linearly to scalar predictions

$$p_t^i = \mathbf{w}_t^i \bullet \mathbf{x}_t = \sum_j w_t^i(j)\, x_t(j) \qquad\qquad \mathbf{w}_t^i \in \Re^n$$

- where the weights, $\mathbf{w}_t^i$, are what is learned
- we consider one prediction, and drop the "*i*"

Watkins's Q(λ)

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

Learns an approximation to the optimal action-value function:

$$p_t = \mathbf{w}_t \bullet \mathbf{x}_t \approx Q^*(s_t, a_t) \in \Re$$

where

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

policy for picking $a_k, k \geq t$

discount rate (e.g., γ=0.9)

Watkins's Q(λ) learning rule

$$\delta_t = r_{t+1} + \max_a \gamma \mathbf{w}_t \cdot \mathbf{x}_{t+1} - \mathbf{w}_t \cdot \mathbf{x}_t$$

TD error, $\delta_t \in \Re$

$$\mathbf{e}_t = \begin{cases} \gamma \lambda \mathbf{e}_t + \mathbf{x}_t & \text{if } a_t = \text{argmax}_a \, \mathbf{w}_t \cdot \mathbf{x}(s_t, a) \\ \mathbf{x}_t & \text{otherwise} \end{cases}$$

eligibility trace, $\mathbf{e}_t \in \Re^n$

trace decay rate, $\lambda \in [0,1]$

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha \delta_t \mathbf{e}_t$$

step size, $\alpha > 0$

- 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,r) semantics

option
π, β, I

- target policy, $\pi : A \times S \to [0,1], \ \sum_a \pi(a \mid s) = 1$ (action set, state set)
- termination probability, $\beta : S \to [0,1]$
- initiation set (set of interest), $I : S \times A \to [0,1]$

option model
r, z

- transient target function, $r : S \to \Re$
- outcome target function, $z : S \to \Re$
- eligibility-trace decay function, $\lambda : S \to [0,1]$

$$p_t \approx E_{\pi,\beta,\lambda}\left[r_{t+1} + r_{t+2} + \cdots + r_T + \begin{matrix} z_T \\ \text{or} \\ \mathbf{w} \bullet \mathbf{x}_T \end{matrix} \;\middle|\; s_t, a_t, T = \text{time of} \begin{matrix} \text{termination by } \beta \\ \text{or} \\ \text{truncation by } \lambda \end{matrix} \right]$$

notational shortcuts

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

$$I_t = I(s_t)$$

$$r_t = r(s_t)$$

$$z_t = z(s_t)$$

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

$$\bar{\mathbf{x}}_t = \sum_a \pi(a \mid s_t)\mathbf{x}(s_t, a) = E\left[\mathbf{x}_t \,\middle|\, a_t \sim \pi\right]$$

GQ learning rule

$$\delta_t = r_{t+1} + \beta_{t+1} z_{t+1} + (1-\beta_{t+1})\mathbf{w}_t \cdot \bar{\mathbf{x}}_{t+1} - \mathbf{w}_t \cdot \mathbf{x}_t$$

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

$$\Delta \mathbf{w}_t = \alpha\left[\delta_t \mathbf{e}_t - (1-\beta_{t+1})(1-\lambda_{t+1})(\mathbf{v}_t \cdot \mathbf{e}_t)\bar{\mathbf{x}}_{t+1}\right]$$

$$\Delta \mathbf{v}_t = \alpha\eta\left[\delta_t \mathbf{e}_t - (\mathbf{v}_t \cdot \mathbf{x}_t)\mathbf{x}_t\right]$$

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

theoretical statement

There exists a scalar Bellman-error objective function

$$J(\mathbf{w}) \quad = \quad \left\| \mathbf{p}_{\mathbf{w}} - \prod T_{\pi}^{\lambda\beta} \mathbf{p}_{\mathbf{w}} \right\|^2_{D_b \cdot I}$$

vector of predictions, one per state

generalized Bellman operator

projection back into the space of representable predictions

such that

$$E_b\left[\Delta \mathbf{w}\right] \quad \propto \quad -\nabla_{\mathbf{w}} J(\mathbf{w})$$

which guarantees convergence to $J(\mathbf{w}) = 0$
(under step-size conditions)

Example uses of GQ

what kind of things might we do with these demons?

- make everything a reward (for some demon)
  - learn an option to achieve it
    - learn a detector for the ability to achieve it
- take hand-coded options, learn about their outcomes
- learn models of options suitable for planning
- guide behavior by the demons' learning progress ("learning feels good")

bump anticipators

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

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

$\beta(s) = 0.1$ (bump in next 10 steps = 0.1 seconds)

$\beta(s) = 0.02$ (bump in next 50 steps = 0.5 seconds)

$\beta(s) = 0.002$ (bump in next 500 steps = 5 seconds)

$$I(s) = 1.0$$

$$r(s) = 0$$

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

$$\lambda(s) = 0.9$$

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

$\pi(a \mid s) = 1$ if $a = \arg\max_{a'} \mathbf{w} \bullet \mathbf{x}(s, a')$, otherwise 0

$\beta(s) = d(s)$ $(+0.01$ if wheels are moving$)$

$I(s) = 1.0$

$r(s) = 0$

$z(s) = d(s)$

$\lambda(s) = 0.95$

Does not need to be run to completion

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, I:

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

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

This form is necessary and sufficient for planning

intrinsic motivation

- imagine *one million prediction demons*, all learning in parallel
  - for various random or cleverly chosen options and target functions
- imagine each can measure its learning progress
- use the *sum-total learning progress* as intrinsic reward to direct the behavior policy
- weed and refine the set of demons, then repeat

conclusions

- *prediction demons* are a powerful language for learning and representing knowledge
- *prediction demons* can learn online, in parallel, and computationally efficiently
- they can certainly be used to learn a lot of stuff about one's world
  - probably more than any previous AI
- and they can be a powerful, lightweight substrate for intrinsic motivation systems

- thank you for your attention