REINFORCEMENT LEARNING

Vivek Borkar

IIT BOMBAY

Decision, Games and Evolution @ ICTS

March 10-21, 2025

Learning paradigms:

1. Supervised learning: based on precise feedback on performance (e.g., gradient information in neural network training, think of a 'teacher')

2. Unsupervised learning: no feedback, 'self-organization' driven by data (e.g., clustering, think of learning to skip rope or ride a skateboard)

3. Reinforcement learning: based on 'evaluative' feedback (e.g., reward or cost, think of your parents or spouse)

Long history, since late 1800's (Thorndike, models of animal behavior), later in psychology in early-mid 20th century (Bush and Mosteller)

Classical engineering approaches to adaptation are model based, e.g.:

Self-tuning control: For a parametric model, estimate the parameters and plug them into the current controller in lieu of the true parameter (Separation of estimation and control).

Model Reference Adaptive Control: Feed control input to the plant and a parametric model and use the error signal from the outputs to tweak parameters.

Alternative 'direct adaptive control' by Tsypkin school in Russia, also K. S. Narendra (US) and M. A. L. Thathachar (India), based on 'learning automata' is closer to RL in spirit.

Major inputs from cognitive scientists in initial phases, continued till date. The Schultz experiments and Dayan-Montague explanation of the data was a major fillip.

RL was resurrected in late 20th century because of computational feasibility in view of vastly increased computing power and emerging applications in robotics etc.

Loosely speaking: RL = iterative schemes for MDPs + stochastic approximation to make it 'data driven' (replaces conditional average/average by incremental iterates for averaging)

Straddles between MCMC and numerical schemes for 'policy evaluation': lower variance at cost of per stage computation.

Q-learning

Consider the discounted value function

$$V(i) = \min E\left[\sum_{m=0}^{\infty} \alpha^m k(X_m, Z_m)\right]$$

where the minimum is over all admissible controls. It satisfies the dynamic programming equation

$$V(i) = \min_u [k(i,u) + \alpha \sum_j p(j|i,u)V(j)] \ \forall i.$$

This is an equation of the form $V = F(V)$ for a suitably defined $F(\cdot) : \mathcal{R}^{|S|} \mapsto \mathcal{R}^{|S|}$.

$F(\cdot)$ is nonlinear and involves a conditional expectation with respect to $p(\cdot|i,u)$ *inside* the nonlinear operator '$\min_u(\cdot)$'.

This presents a problem if we want a data-driven algorithm to solve it using stochastic approximation that leverages the averaging property of stochastic approximation.

For example, we have the value iteration algorithm $V_{n+1} = F(V_n)$, i.e.,

$$V_{n+1}(i) = \min_u (k(i,u) + \alpha \sum_j p(j|i,u) V_n(j)) \ \forall i.$$

To replace the right hand side with an incremental scheme based on stochastic approximation to do the averaging runs into trouble because of the minimization outside the averaging operation : average of a minimum is not the same as minimum of the average.

One way out is to consider the so called Q-values defined by

$$Q(i,u) = k(i,u) + \alpha \sum_j p(j|i,u) V(j).$$

This is nothing but the object being minimized on the right hand side of the DP equation $V(i) = \min_a Q(i,a)$. Therefore

$$Q(i,u) = k(i,u) + \alpha \sum_j p(j|i,u) \min_a Q(j,a).$$

This is similar to the DP equation above and again is of the form $Q = G(Q)$ for a suitably defined $G(\cdot)$. Now the 'min' is inside the (conditional) averaging.

To be precise, for $x := [x(i,u)] \in \mathcal{R}^{|S|\times|U|}$, $G(x) = [G_{iu}]$ is defined by:

$$G_{iu}(x) = k(i,u) + \alpha \sum_j p(j|i,u) \min_a x(j,a) \ \forall i,u,$$

so that the above can be written as a fixed point equation $Q = G(Q)$.

In fact, the 'Q-value iteration' $Q_{n+1} = G(Q_n)$, i.e.,

$$Q_{n+1}(i,u) = k(i,u) + \alpha \sum_j p(j|i,u) \min_a Q_n(j,a), \ \forall i,u \quad (1)$$

will converge to the desired Q. This can be proved exactly as for value iteration.

Note also that a minimizer of $Q(i, \cdot)$ yields an optimal choice of control in state i **without** requiring prior knowledge of $p(\cdot|\cdot, \cdot)$.

But we have increased the dimensionality of the iterates from $|S|$ to $|S| \times |U|$. However, the minimization is now inside the conditional average. Thus using the 'stochastic approximation for finding fixed points of functions', one can propose a stochastic approximation scheme as follows.

Let $a(n) > 0$ be a stepsize sequence satisfying
$\sum_n a(n) = \infty,\ \sum_n a(n)^2 < \infty$.

The (i,u)th component of the iteration then is

$$\begin{aligned} & Q_{n+1}(i,u) \\ &= (1-a(n))Q_n(i,u) + a(n)(k(i,u) + \\ &\quad \alpha \min_a Q_n(\xi_{iu}(n+1),a)) \\ &= Q_n(i,u) + a(n)(k(i,u) + \alpha \min_a Q_n(\xi_{n+1}(i,u),a) \\ &\quad - Q_n(i,u)), \end{aligned} \tag{2}$$

where $\{\xi_n(i,u)\}$ are independent S-valued random variables such that

$$P(\xi_{n+1}(i,u) = j) = p(j|i,u) \ \forall \ i,u,n.$$

In words, we have replaced the conditional average with respect to $p(\cdot|i,u)$ by an evauation at a simulated random variable generated according to $p(\cdot|i,u)$, and then replaced the resulting candidate update by a convex combination of it with the previous iterate $Q_n(i,u)$ with weights $a(n), 1-a(n)$ respectively.

The above iteration can be written as

$$Q_{n+1} = Q_n + a(n)(G(Q_n) - Q_n + M_{n+1}),$$

where $M_{n+1} = [M_{n+1}(i,u)]$ defined by

$$M_{n+1}(i,u) := \alpha(\min_a Q_n(\xi_{n+1}(i,u),a) - \sum_j p(j|i,u) \min_a Q_n(j,a))$$

is a martingale difference sequence.

Thus it is a stochastic approximation algorithm whose o.d.e. limit is

$$\dot{q}(t) = G(q(t)) - q(t). \tag{3}$$

We next prove that $G : \mathcal{R}^{|S|\times|U|} \mapsto \mathcal{R}^{|S|\times|U|}$ is a contraction with respect to the 'max norm' $\|x\|_\infty := max_{i,u}|x(i,u)|$. We have, for $x, y \in \mathcal{R}^{|S|\times|U|}$,

$$\begin{aligned} |G_{iu}(x) - G_{iu}(y)| &= |k(i,u) + \alpha \sum_j p(j|i,u) \min_a x(j,a) - \\ &\quad k(i,u) - \alpha \sum_j p(j|i,u) \min_a y(j,a)| \\ &= \alpha |\sum_j p(j|i,u)(\min_a x(j,a) - \min_a y(j,a))| \\ &\leq \alpha \sum_j p(j|i,u) |\min_a x(j,a) - \min_a y(j,a))| \end{aligned}$$

$$
\begin{aligned}
&\leq \alpha \sum_j p(j|i,u) \max_k |\min_a x(k,a) - \min_a y(k,a))| \\
&= \alpha \max_k |\min_a x(k,a) - \min_a y(k,a))| \\
&\leq \alpha \max_k \max_a |x(k,a) - y(k,a))| \\
&\qquad \text{(Check this)} \\
&= \alpha \|x - y\|_\infty .
\end{aligned}
$$

Taking maximum over i, u on the left hand side, we get

$$\|G(x) - G(y)\|_\infty \leq \alpha \|x - y\|_\infty,$$

i.e., G is a contraction with respect to the norm $\|\cdot\|_\infty$.

By the Banach contraction mapping theorem, G has a unque fixed point Q, i.e., a unique Q satisfying $Q = G(Q)$. But this is precisely the Q-value we are looking for. From the theory of stochastic approximation algorithms for fixed point computation done in class, we have $q(t) \to Q$ for $q(\cdot)$ satisfying (3) and therefore $Q_n \to Q$ a.s. This, of course, requires that we first check that

$$\sup_n \|Q_n\| < \infty.$$

This is easy because the each iterate is a convex combination of the previous one with a uniformly bounded quantity.

$\{\xi_n(i,u)\}$ can be generated by a simulator or sampled from accumulated data. Thus suppose we have a large data base of actual observed transitions (i,u,j), $i,j \in S, u \in U$. If we sample one with first two values $=$ the prescribed i and u, then the third value will be j with probability $\approx p(j|i,u)$.

In practice, we usually have a single run $\{(X_n,Z_n)\}$ of a controlled Markov chain with state process $\{X_n\}$ and control process $\{Z_n\}$. Thus at each time n, you observe a *single* pair (X_n,Z_n) and therefore learn only about the (X_n,Z_n)th component of Q_n. That is, you can update only one component at a time.

Thus the algorithm becomes

$$Q_{n+1}(i,u) = Q_n(i,u) + a(n)I\{X_n = i, Z_n = u\}(k(i,u) + \alpha \min_a Q_n(X_{n+1}, a) - Q_n(i,u)), \quad (4)$$

where $I\{X_n = i, Z_n = u\} = 1$ if $X_n = i$ and $Z_n = u$, and 0 otherwise. This is the Q-learning algorithm of Watkins.

Note that (4) is simply the asynchronous version of (2) and will have the o.d.e. limit

$$\dot{q}(t) = \Lambda(t)(G(q(t)) - q(t)).$$

Here $\Lambda(t)$ is a diagonal matrix with nonnegative entries on its diagonal, which are strictly positive if we have

$$\liminf_{n\uparrow\infty} \frac{\sum_{m=0}^{n} I\{X_m = i, Z_m = u\}}{n} > 0 \qquad (5)$$

with probability one for all i, u. This means that there has to be enough 'exploration' of all state-control pairs.

In the 'off-policy' case, i.e., when we are running a simulator or using past data to train the algorithm, we can ensure this relatively easily, because the choice of $\{Z_n\}$ is flexible. But in on-policy mode, i.e., when a real system is being controlled while learning, this is not automatic.

The natural choice for sake of getting closer to optimum would be to use $Z_n = Z_n^* :=$ the minimizer of $Q_n(X_n, \cdot)$, that being the current guess for the optimal choice. But this does not ensure (5). Hence one often takes $Z_n = Z_n^*$ with probability $1 - \epsilon$ and a random (say, uniformly distributed) $u \in U$ with probability ϵ for some $\epsilon > 0$. SARSA is one such algorithm.

Schemes which slowly decrease this ϵ to zero have also been proposed. How to explore best remains an active research area.

This level of simplicity is not always available, e.g., in training autonomous vehicles, where some actions may be disastrous and cannot be tried even once. 'Safe' reinforcement learning tries to address this issue. Another issue is delayed reward, such as reward obtained only at the end of the task e.g. in robotics.

The above is what one might call 'vanilla' Q-learning. I have ignored a major issue in Q-learning, viz., the curse of dimensionality. For many practical problems, $|S|$ is too large and $|S| \times |U|$ is even worse. The common fix people use is function approximation, i.e., to approximate $Q(i, u)$ by a parametrized function family.

Let this family of functions be $f_\beta(i,u)$ where $\beta \in \mathcal{R}^d$ with only moderately large d, and then write a learning algorithm that tries to find the optimal β.

Some popular schemes are linear function approximation, i.e., linear combination of prescribed basis functions with the weights constituting the β vector, neural networks including the 'deep' ones, etc. These tweaks are usually not provably correct, in fact counterexamples exist. Nevertheless, they often work well, as the success of Q-learning in game playing programs shows.

Post-decision state formalism

We described how the presence of a nonlinearity outside a conditional average makes it difficult to come up with a stochastic approximation version of value iteration and one has to resort to Q-values to work around the problem, thereby blowing up the dimension of the iterates. There are, however, some special situations where one can work with states alone instead of state-control pairs, thereby retaining the original dimension of value iteration.

This is the so called post-decision state formalism. In many situations, the action of control and noise can be separated in the sense that the Markov chain dynamics can be explicitly written as

$$X_{n+1} = f(g(X_n, Z_n), \zeta_{n+1}),$$

where f, g are known maps and $\{\zeta_n\}$ are i.i.d. with a known distribution.

An example is a discrete queue with i.i.d. arrivals $\{\zeta_n\}$ and controlled departures $\{Z_n\}$ so that the dynamics is

$$X_{n+1} = X_n - Z_n + \zeta_{n+1}.$$

Here $Z_n \le X_n$ because the queue length puts a natural upper bound on the number of departures. It is more convenient to leave Z_n unconstrained and replace the above by

$$X_{n+1} = X_n - X_n \wedge Z_n + \zeta_{n+1}.$$

This fits the above framework with $g(x,z) = x - x \wedge z$ and $f(y,k) = y + k$.

For simplicity, we shall take ζ_n to be non-negative integer valued with

$$P(\zeta_n = k) = \mu(k),\ k \geq 0,$$

so that $\mu(k) \geq 0\ \forall k$ and $\sum_k \mu(k) = 1$. Then $Y_n = g(X_n, Z_n)$ is called the post-state and satisfies

$$Y_{n+1} = g(f(Y_n, \zeta_{n+1}), Z_{n+1}).$$

Equivalently, we have the coupled updates

$$Y_n = g(X_n, Z_n),\ X_{n+1} = f(Y_n, \zeta_{n+1}).$$

It is easy to see that $\{Y_n\}$ as well as the alternating sequence

$$X_0, Y_0, X_1, Y_1, \cdots, X_n, Y_n, \cdots$$

are controlled Markov chains in their own right. We call them the Y-chain and the XY-chain resp., and call $\{X_n\}$ the X-chain.

We begin with the XY-chain and instead of using a single notation $V(\cdot)$ for the value function, we use $V(\cdot)$ for the value function when the argument is from the X-chain and $W(\cdot)$ when the argument is from the Y-chain. Then the dynamic programming principle yields

$$\begin{aligned} W(y) &= \sum_k \mu(k) V(f(y,,k)), \\ V(x) &= \min_u (k(x,u) + \alpha W(g(x,u)). \end{aligned}$$

Combining both, we get the dynamic programming equation for the Y-chain as:

$$W(y) = \sum_k \mu(k) \left[\min_u \left(k(f(y,k),u) + \alpha W(g(f(y,k),u)) \right) \right]. \tag{6}$$

Note that the expectation is now with respect to μ and it is outside the minimization. Thus we can write the stochastic approximation version as

$$\begin{aligned} W_{n+1}(i) \;=\; & W_n(i) + a(n) I\{Y_n = i\}(\min_u k(f(i, \zeta_{n+1}), u) \\ & + \alpha W(g(f(i, \zeta_{n+1}), u)) - W_n(i)). \end{aligned}$$

The dimension is now only $|S|$. This is possible when the explicit model is known and is simulated off-line, because we can then generate $\{Y_n\}$ as easily as $\{X_n\}$.

In on-line scenario, this will require that Y_n, ζ_n can be tapped. For example, for the queuing example, we shall need the arrivals and departures observed, not just the queue length.

Actor-critic algorithms

Actor-critic algorithms follow a philosophy similar to the policy iteration.

Given the current guess $v_n(\cdot)$ for the optimal stationary policy, first solve a linear system for the fixed policy $v_n(\cdot)$ (policy evaluation step). This is amenable to a stochastic approximation based policy evaluation scheme.

Then update this guess to $v_{n+1}(\cdot)$ by a suitable stochastic approximation scheme for minimization.

For a fixed stationary policy $v(\cdot)$, the policy evaluation can be done by solving the linear equation

$$V(i) = k(i, v(i)) + \alpha \sum_j p(j|i, v(i))V(j), \ \ i \in S, \qquad (7)$$

solvable by the linear value iteration

$$V_{n+1}(i) = k(i, v(i)) + \alpha \sum_j p(j|i, v(i))V_n(j), \ \ i \in S.$$

The stochastic approximation version of this is

$$V_{n+1}(i) \ = \ V_n(i) + a(n)I\{X_n = i\}(k(i, v(i)) + \alpha V_n(X_{n+1}) - V_n(i)). \qquad (8)$$

But we need to do this for $v(\cdot) = v_n(\cdot)$, which itself is changing with time. Hence in order to get the same effect as the above iteration with $v(\cdot)$ replaced by $v_n(\cdot)$, we invoke two time scale stochastic approximation.

That is, we change $v_n(\cdot)$ on a slower time scale so as to perform the minimization operation implicit in the second component of a policy iteration step. Then $v_n(\cdot)$ is 'quasi-static' for the above iteration, i.e., can be treated as $\approx$ constant for purposes of analysis.

This, however, calls for a further tweak. A stationary policy is a map $S \mapsto U$ where both S, U are discrete. It is much more convenient to write stochastic minimization schemes for continuous variables. So we consider randomized stationary policies instead, which subsume stationary policies. Such a policy is a map $\phi : i \in S \mapsto \phi(i, \cdot) \in \mathcal{U} :=$ the simplex of probability vectors on U, defined as

$$\mathcal{U} := \{\nu \in \mathcal{R}^{|U|} : \nu(a) \geq 0 \ \forall \ a \in U, \ \sum_{a \in U} \nu(a) = 1\},$$

with the interpretation: $\phi(i, a) :=$ the probability of choosing control a in state i.

Then the corresponding counterpart of (7) is

$$V_\phi(i) = \sum_a \phi(i,u)k(i,u) + \alpha \sum_j \sum_a \phi(i,u)p(j|i,u)V_\phi(j),\ i \in S. \tag{9}$$

We update this randomized stationary policy through iterates $\phi_n(\cdot,u), n \geq 0$, on a slower time scale. That is,

$$P(Z_n = u|X_m, m \leq n; Z_m, m < n) =$$
$$P(Z_n = u|X_n) = \phi_n(X_n, u). \tag{10}$$

We replace (8) by its variant

$$V_{n+1}(i) = V_n(i) + a(\nu(i,n))I\{X_n = i\}(k(i,Z_n) + \alpha V_n(X_{n+1}) - V_n(i)). \tag{11}$$

Here Z_n satisfying (10) and $\nu(i,n) := \sum_{m=0}^{n} I\{X_m = i\}$. The stepsize schedule $\{a(n)\}$ satisfies the Robbins-Monro conditions as usual. There are several alternatives for updating ϕ_n. We shall consider a simple parametric one wherein we take ϕ_n to be of the form

$$\phi_n(i,u) = \frac{e^{\beta_n(i,u)}}{\sum_b e^{\beta_n(i,b)}},$$

where $\{\beta_n(i,u)\}$ are parameters we update on a slower time scale. We restrict $\beta(i,u)$ to an interval $[-\beta_0, \beta_0]$ for a prescribed $\beta_0 >> 0$ for all i,u. This ensures that $\phi_n(i,u) \geq \epsilon > 0 \ \forall \ i,u$, for some small $\epsilon > 0$. This in turn ensures sufficient exploration.

Let $\{b(n)\}$ be another stepsize sequence satisfying the Robbins-Monro conditions such that $b(n) = o(a(n))$. Consider the following iteration for each fixed $i \in S$ and all $a \in U$:

$$\beta_{n+1}(i,u) = \Gamma(\beta_n(i,u) + b(\nu'(i,u,n))I\{X_n = i, Z_n = u\} \times [V_n(i) - k(i,u) - \alpha V_n(X_{n+1})]), \quad (12)$$

for $\nu'(i,u,n) := \sum_{m=0}^{n} I\{X_m = i, Z_m = u\}$, with $\Gamma(\cdot) :=$ the projection to $[-\beta_0, \beta_0]$.

Returning to the analysis of the algorithm, we ignore for the time being the additional technicalities caused by the projection operator $\Gamma(\cdot)$.

Using the two time scale logic, we then have

$$V_n(i) - V_{\phi_n}(i) \to 0 \ \forall i \tag{13}$$

In turn, the limiting o.d.e. for (12) becomes

$$\dot{\beta}_t(i,u) = -K_{iu}(\phi_t) := V_{\phi_t}(i) - k(i,u) - \alpha \sum_b p(j|i,u) V_{\phi_t}(j) \tag{14}$$

where

$$\phi_t(i,u) := \frac{e^{\beta_t(i,u)}}{\sum_b e^{\beta_t(i,b)}} \ .$$

Then $\phi_t(i,\cdot)$ satisfies

$$\begin{aligned} \dot{\phi}_t(i,u) &= \frac{e^{\beta_t(i,u)}\dot{\beta}_t(i,u)}{\sum_b e^{\beta_t(i,b)}} - \frac{e^{\beta_t(i,u)} \sum_b \dot{\beta}_t(i,b) e^{\beta_t(i,b)}}{(\sum_b e^{\beta_t(i,b)})^2} \\ &= \phi_t(i,u)(-K_{iu}(\phi_t) - \sum_b \phi_t(j,b)(-K_{ib}(\phi_t)) \\ &= G_{iu}(\phi_t). \end{aligned} \tag{15}$$

This is recognized as a special case of the replicator dynamics. (In fact, because it remains in the probability simplex, the effect of $\Gamma(\cdot)$ becomes irrelevant in the o.d.e. limit, i.e., the o.d.e. limits for the projected and unprojected algorithms turn out to be identical.) Consider the Liapunov function

$$\phi \mapsto \Phi(\phi) := \sum_i V_\phi(i).$$

Then, letting $D_G V_\phi := \left[D_G V_\phi(1), \cdots, D_G V_\phi(s)\right]^T$, we have

$$\begin{aligned} \dot{\Phi}(\phi_t) &= \sum_i \langle D_G V_{\phi_t}, G(\phi_t) \rangle \\ &\leq -\sum_{i,a} K_{ia}(\phi_t) G_{ia}(\phi_t) \leq 0. \end{aligned}$$

Since the sum on the right equals

$$\sum_i \left(\sum_u \phi_t(i,u) K_{iu}(\phi_t)^2 - (\sum_u \phi_t(i,u) K_{iu}(\phi_t))^2 \right),$$

it is zero only when $\phi_t(i,\cdot)$ is concentrated on u for which $K_{iu}(\phi)$ takes identical values.

On the other hand, from (15), it is clear that only the optimal stationary strategies will be stable equilibria. Otherwise a suitable small perturbation can move the trajectory away from the equilibrium (check this). Invoking 'with probability 1 avoidance of unstable equilibria due to noise', we conclude that the algorithm will lead to optimal stationary policies with probability 1.

This is where our ignoring $\Gamma(\cdot)$ *will* make a difference. Note that $\phi(i, \cdot)$ corresponding to stationary strategies are on the boundary of $\mathcal{U}$.

But we have excluded the boundary of $\mathcal{U}$ by using $\Gamma(\cdot)$. What one can conclude with some additional work is that the scheme will converge to the set of nearly optimal randomized stationary policies which are in proximity of some optimal stationary policies.

Remarks :

1. Other minimization schemes in place of (12) are possible.

2. Since (11) is 'nearly linear' (because $v_n(\cdot)$ changes slowly and can be treated as approximately static), one legitimately can use linear function approximation schemes such as TD(λ).

TD(0) (for Temporal Difference)

Next we study the algorithm TD(0), a special (and the simplest) case of a parametric family of algorithms known as TD(λ), parametrized by a parameter $\lambda \in [0, 1]$.

This algorithm is for *policy evaluation*, i.e., for learning the performance of a fixed policy, not for optimizing performance over policies. Thus we fix a stationary policy a priori, rendering $\{X_n\}$ a time-homogeneous Markov chain. Given this fact, we suppress the control altogether in our notation and work with an uncontrolled Markov chain $\{X_n\}$ with transition probabilities $p(\cdot|\cdot)$.

Assume that this chain is irreducible with the unique stationary distribution $\pi = [\pi(1), \cdots, \pi(s)], s = |S|$.

Let $D :=$ the $s \times s$ diagonal matrix whose ith diagonal entry is $\pi(i)$. The 'DP' equation is

$$V(i) = k(i) + \alpha \sum_j p(j|i) V(j), \ \ i \in S,$$

written as a vector equation

$$V = k + \alpha P V$$

for $k = [k(1), \cdots, k(s)]^T$ and $P = [[p(j|i)]]_{i,j \in S} \in \mathcal{R}^{s \times s}$.

The idea is to approximate V by a linear combination of prescribed linearly independent basis functions

$$\phi_i : S \mapsto R, 1 \leq i \leq M, \text{ with } s >> M \geq 1.$$

Thus $V(i) \approx \sum_{m=1}^{M} r_m \phi_m(i)$, i.e., $V \approx \Phi r$ where $r = [r_1, \cdots, r_M]^T$ and Φ is an $s \times M$ matrix whose ith column is ϕ_i. Since $\{\phi_i\}$ are linearly independent, Φ is full rank.

Plugging this approximation into the dynamic programming equation above leads to

$$\Phi r \approx k + \alpha P \Phi r. \tag{16}$$

But there is no reason why the RHS $\subset$ Range(Φ). This suggests the fixed point equation

$$\Phi r = \Pi(k + \alpha P \Phi r) := F(\Phi r), \tag{17}$$

where Π denotes the projection to Range(Φ) with respect to a suitable norm, and $F(x) := \Pi(k + \alpha P x)$. It turns out to be convenient to use projection with respect to the weighted norm

$$\|x\|_D := (\sum_i \pi(i)|x_i|^2)^{1/2}.$$

Then it can be verified that (check this)

$$\Pi x := \Phi(\text{argmin}_y \|x - \Phi y\|_D^2) = \Phi(\Phi^T D \Phi)^{-1} \Phi^T D x. \tag{18}$$

The invertibility of $\Phi^T D \Phi$ is guaranteed by the fact that Φ is full rank.

Note also that

$$\begin{aligned} \|Px\|_D^2 &= \sum_i \pi(i) (\sum_j p(j|i) x_j)^2 \\ &\leq \sum_i \pi(i) \sum_j p(j|i) x_j^2 \\ &\qquad \text{(by Jensen's inequality)} \\ &= \sum_j \pi(j) x_j^2 = \|x\|_D^2. \end{aligned}$$

By this and the fact $\|\Pi x\|_D \leq \|x\|_D$ (because Π is a $\|\cdot\|_D$-projection), it follows that

$$\begin{aligned}
\|F(x) - F(x')\|_D &= \|\Pi(k + \alpha P x) - \Pi(k + \alpha P x')\|_D \\
&\leq \|k + \alpha P x - k - \alpha P x'\|_D \\
&= \alpha \|P(x - x')\|_D \\
&\leq \alpha \|x - x'\|_D,
\end{aligned}$$

so F is a $\|\cdot\|_D$-contraction and has a unique fixed point x^*. Since $x^* \in \text{Range}(\Phi)$, there exists r^* such that $x^* = \Phi r^*$. This r^* is unique because the $\{\phi_i\}$ are linearly independent. That is, r^* is the unique solution to (17).

We now derive an iterative scheme for updating $\{r_n\}$, the successive guesses for r^*. Let $\varphi(i)^T \in \mathcal{R}^M$ for $i \in S$ denote the ith row of Φ. Define the *temporal difference* at time n as

$$d_n := k(X_n) + \alpha\varphi(X_{n+1})^T r_n - \varphi(X_n)^T r_n.$$

Note that its conditional expectation given $X_n =$ (say) i is precisely the discrepancy between the right and left hand sides of the ith approximate equation in (16).

Let $\{a(n)\}$ be stepsizes satisfying the Robbins-Monro conditions.

The TD(0) algorithm is

$$\begin{aligned} r_{n+1} &= r_n - a(n)\varphi(X_n)d_n \\ &= r_n - a(n)\varphi(X_n)(k(X_n) + \\ &\quad \alpha\varphi(X_{n+1})^T r_n - \varphi(X_n)^T r_n). \end{aligned} \tag{19}$$

If we freeze $r_n = r$, the stationary expectation of the ith component of the correction term on the right (i.e., the expression multiplying $a(n)$) turns out to be

$$g(r) := \sum_i \pi(i)\varphi(i)(k(i) + \alpha \sum_j p(j|i)\varphi(j)r - \varphi(i)r).$$

The map $g(\cdot) = [g_1(\cdot), \cdots, g_M(\cdot)]^T : \mathcal{R}^M \mapsto \mathcal{R}^M$ can be written as

$$g(r) = \Phi^T D(k + \alpha P \Phi r - \Phi r). \tag{20}$$

To motivate this, note that it is of the form $g(r) = -\nabla^r \Psi(r, r')|_{r'=r}$, where

$$\Psi(r, r') := \frac{1}{2} \|\Phi r - (k + \alpha P \Phi r')\|_D^2$$

and ∇^r is the gradient with respect to the first argument r.

Compare this with (17)-(18). Thus TD(0) is 'almost' a gradient descent for $\Psi(r, r^*)$, which is what we aim to minimize, but with the unknown r^* replaced by its current guess r_n at time n.

Using our analysis of 'stochastic approximation with Markov noise', the limiting o.d.e. can be written as

$$\dot{r}(t) = g(r(t)). \tag{21}$$

Let $\langle x, y \rangle_D := x^T D y$ for $x, y \in \mathcal{R}^s$.

Consider the Liapunov function $V(x) := \frac{1}{2}\|r - r^*\|^2$. Then

$$\frac{d}{dt} V(r(t)) \leq 0 \text{ and } = 0$$

only at r^*.

This proves that r^* is the unique globally asymptotically stable equilibrium for (21) and therefore $r_n \to r^*$ with probability 1.

In order to justify the above we need to first establish the stability of iterates, i.e., $sup_n \|r_n\| < \infty$ with probability 1. Again, the stability test covered in class applies: Define $g^\infty(r) := \lim_{c\uparrow\infty} \frac{g(cr)}{c}$, which turns out to be the same as $g(r)$ with the term k replaced by the zero vector. The o.d.e. $\dot{r}(t) = g^\infty(r(t))$ is then seen to have the origin as the unique asymptotically stable equilibrium, from which the above claim follows by the aforementioned test.

The more general scheme TD(λ) has an additional parameter $\lambda \in [0,1]$ and the update rule

$$r_{n+1} = r_n + a(n) z_n d_n,$$

where the 'eligibility vectors' $z_n := \sum_{m=0}^{n} (\alpha\lambda)^{n-m} \varphi(X_m)$ are given recursively by

$$z_{n+1} = \alpha\lambda z_n + \varphi(X_n),\ n \geq 0.$$

The analysis is messier for $\lambda \neq 0$. The intuition is that the eligibility vectors also give weight to past observations. This give an extra parameter λ to play with. TD(λ), $\lambda > 0$, are computationally more intensive for $\lambda > 0$.

Certain strengths of $TD(\lambda)$ are worth note:

1. Since we update only the weights $\{r_n\}$ and the state appears only as the argument of the 'features' $\varphi(\cdot)$, it generalizes easily to general state spaces, except that the math gets slightly more abstract (e.g., summation with weights $\pi(\cdot)$ will get replaced by integration with respect to the stationary probability distribution π).

2. If it is a partially observed Markov chain, we can choose $\varphi(\cdot)$ that depend only on the observables with no change in the theory. Thus in principle it is already applicable to partially observed Markov chains.

3. While by itself it only evaluates a fixed policy, TD(λ) can serve as a component of two time scale learning algorithms, such as the actor-critic algorithm.

The reason linear function approximation cannot be 'legitimately' plugged directly into fully nonlinear schemes such as Q-learning is that it is not provably convergent, in fact counterexamples exist. It is well behaved with respect to the $\|\cdot\|_D$ norm, not the $\|\cdot\|_\infty$ norm that dominates the dynamic programming world.

LSPE(0) (for Least Squares Policy Evaluation)

The LSPE(λ) is another family of policy evaluation schemes like TD(λ), parametrized by a parameter $\lambda \geq 0$. It often gives better convergence at the expense of more computation. We shall restrict to the easier case of $\lambda = 0$ and follow the same notation as for TD(0).

'LSPE' stands for Least Squares Policy Evaluation.

Consider the infinite horizon discounted cost. The idea is to solve for the linear parametrization $V \approx \Phi r$, with Φ having full rank, the projected Bellman equation $\Phi r = \Pi(k + \alpha P \Phi r)$ by minimizing the square error

$$\|\Phi r - \Pi(k + \alpha P \Phi r)\|_D^2.$$

Minimizing this in an iterative fashion leads to

$$\begin{aligned} r_{n+1} &= \text{argmin} \|\Phi r - \Pi(k + \alpha P \Phi r_n)\|_D^2 \\ &= \text{argmin} \|\Phi r - (k + \alpha P \Phi r_n)\|_D^2 \\ &= B^{-1}(\hat{A} r_n + b). \end{aligned}$$

Here

$$\begin{aligned} B &:= \sum_i \pi(i)\varphi(i)\varphi(i)^T = \Phi^T D\Phi, \\ \widehat{A} &:= \alpha \sum_i \pi(i)\varphi(i) \sum_j p(j|i)\varphi(j)^T = \alpha\Phi^T DP\Phi, \\ b &:= \sum_i \pi(i)\varphi(i)k(i) = \Phi^T Dk. \end{aligned}$$

Let $A = \widehat{A} - \Phi^T D\Phi$. An incremental version of the above recursion would be

$$r_{n+1} = (1 - a(n))r_n + a(n)B^{-1}(\widehat{A}r_n + b)$$

$$= r_n + a(n)B^{-1}(Ar_n + b).$$

The variant based on a single run of the Markov chain then becomes

$$r_{n+1} = r_n + a(n) B_n^{-1} (A_n r_n + b_n),$$

where A_n, B_n, b_n are running estimates of A, B, b. given by

$$\begin{aligned} B_n &:= \frac{1}{n+1} \sum_{m=0}^{n} \varphi(X_m) \varphi(X_m)^T, \\ A_n &:= \frac{1}{n+1} \sum_{m=0}^{n} \varphi(X_m)(\alpha \varphi(X_{m+1}) - \varphi(X_m)), \\ b_n &:= \frac{1}{n+1} \varphi(X_m) k(X_m). \end{aligned}$$

These are calculated recursively by

$$\begin{aligned} B_n &= B_{n-1} + \frac{1}{n}(\varphi(X_{n-1})\varphi(X_{n-1})^T - B_{n-1}), \\ A_n &= A_{n-1} + \frac{1}{n}(\varphi(X_{n-1})(\alpha\varphi(X_n) - \varphi(X_{n-1}))^T - A_{n-1}), \\ b_n &= b_{n-1} + \frac{1}{n}(\varphi(X_{n-1})k(X_{n-1}) - b_{n-1}). \end{aligned}$$

One also sets $B_0 = \delta I + \varphi(X_0)\varphi(X_0)^T$, instead of just $\varphi(X_0)\varphi(X_0)^T$, for some $\delta > 0$, so that B_0 and therefore B_n are invertible for all $n \geq 0$. This makes negligible difference in the long run because of the averaging.

B_n^{-1} can also be calculated recursively by the Sherman-Morrison formula

$$(C + uv^T)^{-1} = C^{-1} - \frac{C^{-1}uv^TC^{-1}}{1 + v^TC^{-1}u}$$

for $C \in R^{d\times d}, u, v \in \mathcal{R}^d$.

The limiting o.d.e. is

$$\dot{r}(t) = B^{-1}(Ar(t) + b).$$

The equilibrium r^* of this o.d.e. will satisfy

$$B^{-1}(Ar^* + b) = \theta$$

where $\theta :=$ the zero vector. Using the definitions of A, B, b, this can be written as

$$(\Phi^T D \Phi)^{-1}((\alpha \Phi^T D P \Phi)r^* - \Phi^T D \Phi r^* + \Phi^T D k) = \theta.$$

Left-multiplying by Φ,

$$\alpha\Phi(\Phi^T D\Phi)^{-1}\Phi^T DP\Phi r^* - \Phi r^* + \Phi(\Phi^T D\Phi)^{-1}\Phi^T Dk = \theta,$$

that is,

$$\Phi r^* = \Pi(\alpha P\Phi r^* + k)$$

as desired.

It remains to prove that r^* is indeed an asymptotically stable equilibrium of this o.d.e.

Since it is an affine o.d.e., it suffices to check that the eigenvalues of $B^{-1}A$ are in the left half complex plane, which can be verified. Hence r^* is the globally asymptotically stable euilibrium of the o.d.e.

The scaling limit of this o.d.e. is $\dot{r} = B^{-1}Ar$, which likewise has θ as its globally asymptotically stable equilibrium.

This implies that the $\{r_n\}$ remain bounded with probability one by the stability test done in class, which, coupled with the earlier observation, implies that $r_n \to r^*$ with probability one.

Policy gradient methods

We consider here a vanilla version of policy gradient methods. This is a very important class of RL algorithms and many schemes that have attracted attention in recent times are rooted in this.

The central idea is to consider a parametrized controller (e.g., a deep neural network) and do a stochastic gradient descent over the parameter. The key step then is to get an expression for this gradient.

We analyze this in the context of average cost. Consider the parametrized Poisson equation with parameter $\theta \in \mathcal{R}^\ell$, given by

$$V_\theta(i) = k_\theta(i) - \beta_\theta + \sum_j p_\theta(j|i) V_\theta(j), \ i \in S. \tag{22}$$

Here the transition probabilities and the running cost depend on a parametrized control parammetrized by θ, i.e.,

$$p_\theta(j|i) = \sum_u p(j|i,u) q_\theta(u|i), \ k_\theta(i) = \sum_u q_\theta(u|i) k(i,u), \tag{23}$$

where we have considered a parametrized family q_θ of stationary randomized policies.

This explicit representation, however, is not necessary for deriving the expression for $\nabla^\theta \beta_\theta$. (Recall that β_θ is the average cost under θ.) The main point here is that the θ-dependence comes through the transition probabilities and the running cost function, through their dependence on a parametrized control.

Let $\pi_\theta :=$ the unique stationary distribution under θ, assuming irreducibility. Differentiating both sides of (22),

$$\nabla^\theta V_\theta(i) = \nabla^\theta k_\theta(i) - \nabla^\theta \beta_\theta + \sum_j \nabla^\theta p_\theta(j|i) V_\theta(j) + \sum_j p_\theta(j|i) \nabla^\theta V_\theta(j).$$

Multiplying both sides by $\pi_\theta(i)$, summing over i, and using the facts that

$$\pi_\theta(i) = \sum_j \pi_\theta(j) p_\theta(i|j) \text{ and } \nabla^\theta \log q_\theta(u|i) = \frac{\nabla^\theta q_\theta(u|i)}{q_\theta(u|i)} \; \forall i, u,$$

the terms involving $\nabla^\theta V_\theta(\cdot)$ cancel out and we have

$$\begin{aligned}
\nabla^\theta \beta_\theta &= \sum_i \pi_\theta(i) \nabla^\theta k_\theta(i) + \sum_i \pi_\theta(i) \sum_j \nabla^\theta p_\theta(j|i) V_\theta(j) \\
&= \sum_i \pi_\theta(i) \nabla^\theta k_\theta(i) + \sum_i \pi_\theta(i) \sum_j \sum_u \nabla^\theta q_\theta(u|i) p(j|i,u) V_\theta(j) \\
&= \sum_i \pi_\theta(i) \nabla^\theta k_\theta(i) + \\
&\quad \sum_i \pi_\theta(i) \sum_j \sum_u q_\theta(u|i) p(j|i,u) \nabla^\theta \log(q_\theta(u|i)) V_\theta(j).
\end{aligned}$$

Consider the controlled Markov chain $(X_n, Z_n), n \geq 0$, with $Z_n \approx q_\theta(\cdot|X_n)$. Then a stochastic gradient descent scheme for minimizing β_θ is given by

$$\begin{aligned} \theta(n+1) &= \theta(n) - a(n)[\nabla^\theta k_\theta(X_n) + \\ &\qquad \nabla^\theta(\log q_\theta(Z_n|X_n))V_\theta(X_{n+1})]_{\theta=\theta(n)} \end{aligned} \tag{24}$$

where $\{a(n)\}$ satisfy the usual conditions.

Using the theory of 'stochastic approximation with Markov noise', we see that the iteration will track the o.d.e.

$$\dot{\theta}(t) = -\nabla^{\theta} \beta_{\theta(t)}, \tag{25}$$

as desired. Of course, (24) presupposes that $V_{\theta(n)}(\cdot)$ is available. This can be computed concurrently on a faster time scale by a stochastic approximation version of policy evaluation as in actor-critic algorithms, or, writing $V_\theta(i) = \min_a Q_\theta(i,a)$, by Q-learning on a faster time scale, etc.

Bellman error methods

Here the idea is to approximate the value function $V(\cdot)$ by a function $F(\cdot;\theta)$ from a family of functions parametrized by θ. We first consider the policy evaluation case. The 'Bellman error' then is

$$\mathcal{E}(\theta) := \frac{1}{2}\|F(\cdot;\theta) - (k + \alpha P F(\cdot;\theta))\|_D^2.$$

Then the objective is to minimize this over θ. Note that the Bellman error is zero for the value function.

The gradient of $\mathcal{E}(\cdot)$ is

$$\begin{aligned}\nabla\mathcal{E}(\theta) &= \sum_i \pi(i)(k(i) + \alpha\sum_j p(j|i)F(j;\theta) - F(i;\theta)) \\ &\quad \times(\alpha\sum_j p(j|i)\nabla F(j;\theta) - \nabla F(i;\theta)).\end{aligned}$$

The stochastic gradient descent then becomes

$$\begin{aligned}\theta_{n+1} &= \theta_n - a(n)(k(X_n) + \alpha F(X_{n+1};\theta_n) - F(X_n;\theta_n)) \\ &\quad \times(\alpha\nabla F(\tilde{X}_{n+1};\theta_n) - \nabla F(X_n;\theta_n)).\end{aligned}$$

Here $\tilde{X}_{n+1}$ is a simulated random variable according to $p(\cdot|i)$, conditionally independent of X_{n+1} and all random variables realized toll n, given $X_n = i$.

This is the '*double sampling*' idea due to Baird. It leads to the correct gradient dynamics of the limiting o.d.e.

Using X_{n+1} in place of $\tilde{X}_{n+1}$ leads to a conditional expectation of products in place of a product of conditional expectations, which is wrong. However, this does not matter for deterinistic dynamics.

As with all SGD, this ensures a.s. convergence to a local minimum. In practice, double sampling may be awkward in some simulation environments.

This algorithm can be considered as a special case of a family of algorithms

$$\begin{aligned}\theta_{n+1} &= \theta_n - a(n)I\{X_n = i\}(k(i) + \alpha F(X_{n+1};\theta_n) - F(i;\theta_n)) \\ &\quad \times(\lambda\alpha\nabla F(\tilde{X}_{n+1};\theta_n) - \nabla F(X_n;\theta_n)),\end{aligned}$$

where $\lambda > 0$ is a tunable parameter. $\lambda = 0$ with linear function approximation leads to $TD(0)$.

Going beyond policy evaluation, one can also use this formalism for optimization. Combining with Q-learning, let $Q(i, u; \theta)$ denote the parametrized Q-value.

The algorithm then is

$$\begin{aligned}\theta_{n+1} \;=\; & \theta_n - a(n)(k(X_n, Z_n) + \alpha \max_v Q(X_{n+1}, v; \theta_n) \\ & - \; Q(X_n, Z_n; \theta_n)) \times \\ & (\lambda\alpha \nabla Q(\tilde{X}_{n+1}, Z'_n; \theta_n) - \nabla F(X_n, Z_n; \theta_n)),\end{aligned}$$

where we use $Z'_n \in$ Argmax $(Q(X_{n+1}, \cdot; \theta_n))$ by Danskin's theorem. This can be analyzed using nonsmooth analysis.

For $\lambda = 1$, this is a proper gradient-like scheme. For $\lambda = 0$ with the parametrized family given by a deep neural network, this is known as Deep Q-Network (DQN) reinforcement learning. In this, we can view the Bellman error (for a stationary randomized policy $\varphi(\cdot|i), i \in S$) as

$$\bar{\mathcal{E}}(\theta, \theta') := \frac{1}{2} \sum_i \pi(i) \varphi(u|i) \times$$
$$(Q(i,u;\theta) - (k(i,u) + \alpha p(j|i,u) \max_v Q(j,v;\theta'))^2.$$

The partial gradient w.r.t. θ is

$$\begin{aligned} \nabla_\theta \bar{\mathcal{E}}(\theta, \theta') \;=\; & -\sum_i \pi(i) \varphi(u|i) (k(i) + \alpha \sum_j p(j|i) \max_v Q(j,v;\theta') \\ & -\; Q(i,u;\theta)) \nabla_\theta Q(i,u;\theta)). \end{aligned}$$

The DQN algorithm is

$$\begin{aligned}\theta_{n+1} \;=\; \theta_n + a(n)(k(X_n, Z_n) + \alpha \max_v Q(X_{n+1}, v; \theta'_n) \\ - \; Q(X_n, Z_n; \theta_n))\nabla Q(X_n, Z_n; \theta_n)),\end{aligned}$$

Here θ'_n is the 'target'. It is updated periodically and set equal to θ_n.

Another version which replaces $\nabla^\theta \max_v Q(X_{n+1}, v; \theta'_n)$ by $\nabla^\theta Q(X_{n+1}, v; \theta'_n)|_{v \in Argmax(Q(X_{n+1}, \cdot; \theta_n))}$ instead of $\nabla^\theta Q(X_{n+1}, v; \theta'_n)|_{v \in Argmax(Q(X_{n+1}, \cdot; \theta'_n))}$ is called double DQN and has better performance.

Both schemes do not have theoretical guarantees and can fail.

Experience replay: Here the right hand side of DQN scheme is averaged over a randomly selected batch of past triplets $(X_m, Z_m, X_{m+1}), m \leq n$, at time n. Thus it can be implemented with a single run of a simulation with a buffer. Some advantages are:

1. It reduces variance.

2. It helps reduce effect of anomalous transitions.

3. It avoids the problems caused by overfitting to current data.

4. Re-use of data leads to data efficiency.

5. It is better suited for delayed rewards.

6. In case of Bellman error: If the averaging is done for only one of the product terms and over triplets $(X_m, Z_m, X_{m+1}), m \leq n$, for which $X_m = X_n, Z_m = Z_n$, then we get the same effect as double sampling.

Zap Q-learning (Devraj, Meyn, et al)

Combines aspects of TD(λ)/DQN and LSPE(λ). Let $\{a(n)\}, \{b(n)\}$ satisfy the Robbins-Monro conditions with $b(n) = o(a(n))$ and $A_0 =$ a nonsimgular matrix.

$$
\begin{aligned}
\upsilon_n(i) &:= \text{argmin}(Q(i, \cdot; \theta_n)), \\
\theta_{n+1} &= \theta_n - b(n) A_{n+1}^{-1} \zeta_n (k(X_n, Z_n) + \alpha Q(X_{n+1}, \upsilon_n(X_{n+1})) \\
&\quad - Q(X_n, Z_n; \theta_n)), \\
A_{n+1} &= A_n + a(n)(\zeta_n [\alpha \phi(X_{n+1}, \upsilon_n(X_{n+1}) - \phi(X_n, Z_n)]^T \\
&\quad - A_n), \\
\zeta_{n+1} &= \lambda \alpha \zeta_n + \phi(X_{n+1}, \upsilon_n(X_{n+1}).
\end{aligned}
$$