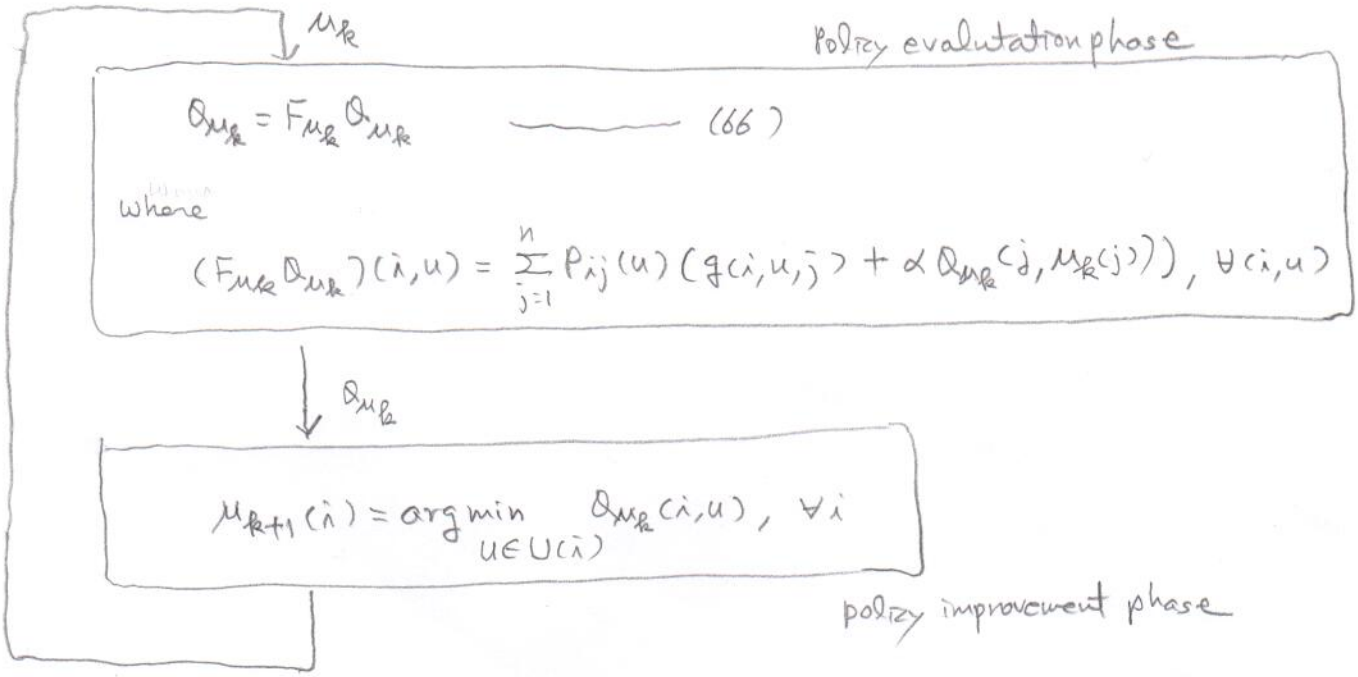
Q-Learning and PI



- Optimistic PI

$$Q_{\mu^k} = F_{\mu^k}^{m_k} Q_{\mu^{k-1}}$$ instead of solving (66) exactly

Simulation-based Extreme Optimistic PI

- Known as SARSA (State-Action-Reward-State-Action)
- Use a single sample between policy updates
= On-policy learning

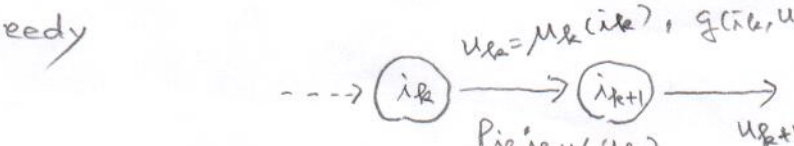
At the start of iteration k, we have the current $Q_{\mu^{k-1}}$, μ^k, we are at some state i_k, and we have chosen a control u_k. Then,

(1) Simulate the next transition (i_k, i_{k+1}) using the transition probabilities $p_{i_k j}(u_k)$

(2) Generate the control u_{k+1} from the minimization

$$u_{k+1} = \mu^k(i_{k+1}) = \arg\min_{u \in U(i_{k+1})} Q_{\mu^{k-1}}(i_{k+1}, u)$$

Note that in some schemes, u_{k+1} is chosen with a small probability to be a random element of $U(i_{k+1})$ in order to enhance exploration, e.g.) ε-greedy



(3) Update Q-factor via single-sample approximation of extreme optimistic PI, i.e., $m_k = 1$

$$Q_{\mu_k}(i_k, u_k) = F_{\mu_k} Q_{\mu_{k-1}}(i_k, u_k)$$

$$\Rightarrow Q_{\mu_k}(i_k, u_k) = (1-\gamma_k) Q_{\mu_{k-1}}(i_k, u_k) + \gamma_k \left(g(i_k, u_k, i_{k+1}) + \alpha Q_{\mu_{k-1}}(i_{k+1}, u_{k+1}) \right) \quad \text{——— (67)}$$

on-policy

$$\Rightarrow Q_{\mu_k}(i_k, u_k) = Q_{\mu_{k-1}}(i_k, u_k) + \gamma_k \underbrace{\left(g(i_k, u_k, i_{k+1}) + \alpha Q_{\mu_{k-1}}(i_{k+1}, u_{k+1}) - Q_{\mu_{k-1}}(i_k, u_k) \right)}_{\text{TD}} \quad \text{——— (68)}$$

Comparison of SARSA in (67) with Q-learning in (65).

- Both SARSA and Q-learning algorithms are model-free method.
- SARSA is on-policy TD learning and Q-learning is off-policy TD learning.
- SARSA is an optimistic PI method and Q-learning is a VI method.
- Q-learning outperforms SARSA, in particular, due to the off-policy learning capability.
- The convergence of Q-learning can be proven using the theory of asynchronous distributed stochastic approximation whereas the convergence properties of SARSA are unclear.
- Both SARSA and Q-learning can be easily generalized to multistep algorithms, i.e., SARSA(λ) and Q-learning(λ), to reduce bias.
- Both SARSA and Q-learning can use linear approx. architecture or state aggregation to overcome the curse of the excessive number of Q-factors / state-control pairs (i, u).

Q-Factor Approximation and Projected Equations

- Introduce a linear parametric architecture for Q

$$\tilde{Q}_\mu(i,u,r) = \phi(i,u)'r$$

or

$$\tilde{Q}_\mu(i,u,r) = \phi(i)'r_u, \quad \forall u \in U(i)$$

- The projected Bellman equation for a policy μ

$$\Phi r = \Pi F_\mu(\Phi r)$$

where

$$F_\mu(\Phi r)(i,u) = \sum_{j=1}^{n} p_{ij}(u)\left(g(i,u,j) + \alpha(\Phi r)(j,\mu(j))\right) \qquad (69)$$

- Simulation-based Implementation

$$\sum_{t=0}^{k} \phi(i_t,u_t)\, q_{k,t} = 0 \qquad \text{"LSTD(0)"}$$

$$r_{k+1} = r_k - \left(\sum_{t=0}^{k} \phi(i_t,u_t)\phi(i_t,u_t)'\right)^{-1} \sum_{t=0}^{k} \phi(i_t,u_t)\, q_{k,t} \qquad \text{"LSPE(0)"}$$

$$r_{k+1} = r_k - \gamma_k \phi(i_k,u_k)\, q_{k,k} \qquad \text{"TD(0)"}$$

where $q_{k,t}$ are the corresponding TD

$$q_{k,t} = \phi(i_t,u_t)'r_k - \alpha\phi(i_{t+1},u_{t+1})'r_k - g(i_t,u_t,i_{t+1}).$$

SARSA for approximate Q

At the start of iteration k, we have the current r_k and μ_k, we are at some state i_k, and we have chosen a control u_k. Then,

(1) Simulate the next transition (i_k, i_{k+1}) using the transition probabilities $p_{i_k j}(u_k)$

(2) Generate the control u_{k+1} from the minimization

$$u_{k+1} = \mu_k(i_{k+1}) = \arg\min_{u \in U(i_{k+1})} \tilde{Q}_{\mu_{k-1}}(i_{k+1},u)$$

Note that in some schemes, u_{k+1} is chosen with a small probability to be a random element of $U(i_{k+1})$ in order to enhance exploration, e.g.) ϵ-greedy

(3) Update parameter vector via TD(ϕ)

$$r_{k+1} = r_k - \gamma_k \phi(i_k, u_k) g_{k,k}$$

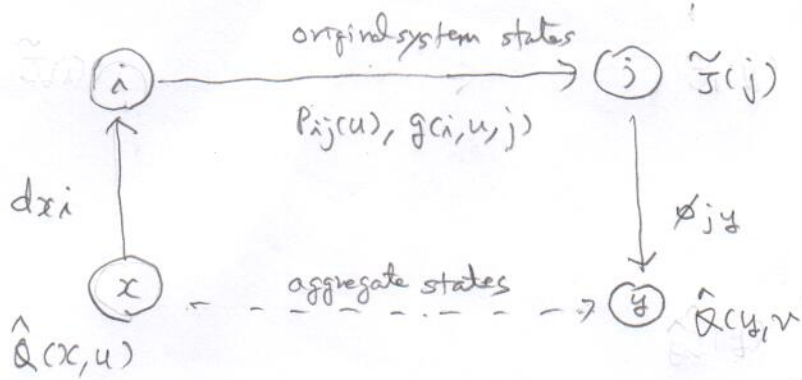
$$\Rightarrow \quad r_{k+1} = r_k + \gamma_k \phi(i_k, u_k) \Big(g(i_k, u_k, i_{k+1}) + \alpha \phi(i_{k+1}, u_{k+1})' r_k - \phi(i_k, u_k)' r_k \Big) \qquad \text{---- (10)}$$

- Note that approximate SARSA in (10) can be viewed as an single-sample approximation of extreme PI ($m_k = 1$) for the projected Bellman equation in (69).
- Compare SARSA in (68) and approximate SARSA in (90).

Q-Learning and Aggregation

- Assume that the control constraint set $U(i)$ is independent of the state i and is denoted by U.
- The aggregate Bellman equation for $\hat{Q}(x,u)$, $x \in A$, $u \in U$

$$\hat{Q}(x,u) = \hat{g}(x,u) + \alpha \sum_{y \in A} \hat{P}_{xy}(u) \min_{v \in U} \hat{Q}(y,v) \qquad \text{---- (11)}$$



$$= \underbrace{\sum_{i=1}^{n} d_{xi} \sum_{j=1}^{n} P_{ij}(u)\, g(i,u,j)}_{\hat{g}(x,u)} + \alpha \sum_{y \in A} \overbrace{\left(\sum_{i=1}^{n} d_{xi} \sum_{j=1}^{n} P_{ij}(u) \phi_{jy} \right)}^{\hat{P}_{xy}(u)} \min_{v \in U} \hat{Q}(y,v)$$

$$= \sum_{i=1}^{n} d_{xi} \sum_{j=1}^{n} P_{ij}(u) \left(g(i,u,j) + \alpha \sum_{y \in A} \phi_{jy} \min_{v \in U} \hat{Q}(y,v) \right)$$

- VI method for $\hat{Q}$

$$\hat{Q}_{k+1} = F\hat{Q}_k \qquad \text{——— (12)}$$

where

$$(F\hat{Q}_k)(x,u) = \sum_{i=1}^{n} d_{xi} \sum_{j=1}^{n} p_{ij}(u)\Big(g(i,u,j) + \alpha \sum_{y \in A} \phi_{jy} \min_{v \in U} \hat{Q}_k(y,v)\Big), \quad \forall (x,u)$$

- One can show that F is a maximum-norm contraction with modulus α so that the VI algorithm converges to $\hat{Q}^*$ from every starting point $\hat{Q}_0$.

- Simulation-based VI method for $\hat{Q}$

Generate an infinite sequence of state-control pairs $\{(x_k, u_k)\} \subset A \times U$. Given the pair (x_k, u_k), generate an original system state i_k according to the disaggregation probabilities $d_{x_k i}$, and then next state j_k according to the probabilities $p_{i_k j}(u_k)$. Finally, generate an aggregate state y_k according to the aggregation probabilities $\phi_{j_k y}$.

For each $(x_k, u_k, i_k, j_k, y_k)$ generated, update the Q-factor of (x_k, u_k) as follows:

$$\hat{Q}_{k+1}(x,u) = (1-\gamma_k)\hat{Q}_k(x,u) + \gamma_k (F_k \hat{Q}_k)(x,u), \quad \forall (x,u)$$

——— (13)

where

$$(F_k \hat{Q}_k)(x,u) = \begin{cases} g(i_k, u_k, j_k) + \alpha \min_{v \in U} \hat{Q}_k(y_k, v) & \text{if } (x,u) = (x_k, u_k) \\ \hat{Q}_k(x,u) & \text{if } (x,u) \neq (x_k, u_k) \end{cases}$$

- Note that $(F_k \hat{Q}_k)(i_k, u_k)$ in (13) is a single sample approximation of the expected values defining $(F\hat{Q}_k)(i_k, u_k)$ in (12).

- After solving the Q-factors $\hat{Q}$ for the aggregate system, the Q-factors $\tilde{Q}$ for the original system are approximated by

$$\tilde{Q}(j,v) = \sum_{y \in A} \phi_{jy} \hat{Q}(y,v) \qquad \text{——— (14)}$$

Then, the optmal cost-to-go function $\tilde{J}(j)$ of the orignal system is approxmated by

$$\tilde{J}(j) = \min_{v \in U} \tilde{Q}(j, v).$$