6.231 Dynamic Programming and Stochastic Control
Fall 2008

6.231 DYNAMIC PROGRAMMING

LECTURE 23

LECTURE OUTLINE

- Review of indirect policy evaluation methods
- Multistep methods, LSPE(λ)
- LSTD(λ)
- Q-learning
- Q-learning with linear function approximation
- Q-learning for optimal stopping problems

REVIEW: PROJECTED BELLMAN EQUATION

- For a fixed policy μ to be evaluated, consider the corresponding mapping T:

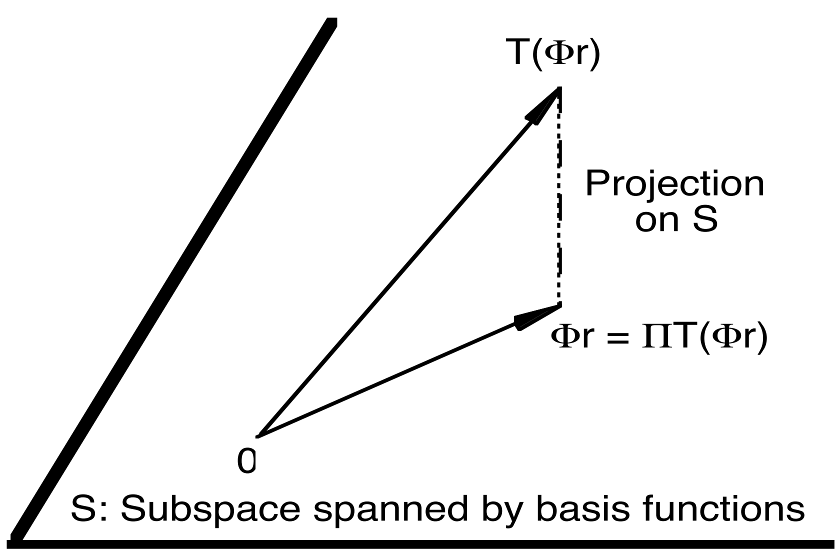
$$(TJ)(i) = \sum_{i=1}^{n} p_{ij}\big(g(i,j)+\alpha J(j)\big), \qquad i = 1,\ldots,n,$$

or more compactly,

$$TJ = g + \alpha PJ$$

- The solution J_μ of Bellman's equation $J = TJ$ is approximated by the solution of

$$\Phi r = \Pi T(\Phi r)$$



Indirect method: Solving a projected form of Bellman's equation

PVI/LSPE

- **Key Result:** ΠT is contraction of modulus α with respect to the weighted Euclidean norm $\|\cdot\|_\xi$, where $\xi = (\xi_1, \ldots, \xi_n)$ is the steady-state probability vector. The unique fixed point Φr^* of ΠT satisfies

$$\|J_\mu - \Phi r^*\|_\xi \leq \frac{1}{\sqrt{1-\alpha^2}} \|J_\mu - \Pi J_\mu\|_\xi$$

- **Projected Value Iteration (PVI):** $\Phi r_{k+1} = \Pi T(\Phi r_k)$, which can be written as

$$r_{k+1} = \arg\min_{r \in \Re^s} \left\|\Phi r - T(\Phi r_k)\right\|_\xi^2$$

or equivalently

$$r_{k+1} = \arg\min_{r \in \Re^s} \sum_{i=1}^n \xi_i \left(\phi(i)'r - \sum_{j=1}^n p_{ij}\Big(g(i,j) + \alpha\phi(j)'r_k\Big)\right)^2$$

- **LSPE (simulation-based approximation):** We generate an infinite trajectory $(i_0, i_1, \ldots)$ and update r_k after transition (i_k, i_{k+1})

$$r_{k+1} = \arg\min_{r \in \Re^s} \sum_{t=0}^k \Big(\phi(i_t)'r - g(i_t, i_{t+1}) - \alpha\phi(i_{t+1})'r_k\Big)^2$$

JUSTIFICATION OF PVI/LSPE CONNECTION

- By writing the necessary optimality conditions for the least squares minimization, PVI can be written as

$$\left(\sum_{i=1}^{n} \xi_i\,\phi(i)\phi(i)'\right) r_{k+1} = \left(\sum_{i=1}^{n} \xi_i\,\phi(i) \sum_{j=1}^{n} p_{ij}\big(g(i,j) + \alpha\phi(j)'r_k\big)\right)$$

- Similarly, by writing the necessary optimality conditions for the least squares minimization, LSPE can be written as

$$\left(\sum_{t=0}^{k} \phi(i_t)\phi(i_t)'\right) r_{k+1} = \left(\sum_{t=0}^{k} \phi(i_t)\big(g(i_t, i_{t+1}) + \alpha\phi(i_{t+1})'r_k\big)\right)$$

- So LSPE is just PVI with the two expected values approximated by simulation-based averages.

- Convergence follows by the law of large numbers.

- The bottleneck in rate of convergence is the law of large of numbers/simulation error (PVI is a contraction with modulus α, and converges fast relative to simulation).

LEAST SQUARES TEMP. DIFFERENCES (LSTD)

- Taking the limit in PVI, we see that the projected equation, $\Phi r^* = \Pi T(\Phi r^*)$, can be written as $Ar^* + b = 0$, where

$$A = \sum_{i=1}^{n} \xi_i \, \phi(i) \left(\alpha \sum_{j=1}^{n} p_{ij} \phi(j) - \phi(i) \right)'$$

$$b = \sum_{i=1}^{n} \xi_i \, \phi(i) \sum_{j=1}^{n} p_{ij} g(i, j)$$

- A, b are expected values that can be approximated by simulation: $A_k \approx A$, $b_k \approx b$, where

$$A_k = \frac{1}{k+1} \sum_{t=0}^{k} \phi(i_t) \big(\alpha \phi(i_{t+1}) - \phi(i_t) \big)'$$

$$b_k = \frac{1}{k+1} \sum_{t=0}^{k} \phi(i_t) g(i_t, i_{t+1})$$

- **LSTD method:** Approximates r^* as

$$r^* \approx \hat{r}_k = -A_k^{-1} b_k$$

- Conceptually very simple ... but less suitable for optimistic policy iteration (hard to transfer info from one policy evaluation to the next).

- Can be shown that convergence rate is the same for LSPE/LSTD (for large k, $\|r_k - \hat{r}_k\| << \|r_k - r^*\|$).

MULTISTEP METHODS

- Introduce a multistep version of Bellman's equation $J = T^{(\lambda)}J$, where for $\lambda \in [0,1)$,

$$T^{(\lambda)} = (1-\lambda)\sum_{t=0}^{\infty} \lambda^t T^{t+1}$$

- Note that T^t is a contraction with modulus α^t, with respect to the weighted Euclidean norm $\|\cdot\|_\xi$, where ξ is the steady-state probability vector of the Markov chain.

- From this it follows that $T^{(\lambda)}$ is a contraction with modulus

$$\alpha_\lambda = (1-\lambda)\sum_{t=0}^{\infty} \alpha^{t+1}\lambda^t = \frac{\alpha(1-\lambda)}{1-\alpha\lambda}$$

- T^t and $T^{(\lambda)}$ have the same fixed point J_μ and

$$\|J_\mu - \Phi r_\lambda^*\|_\xi \leq \frac{1}{\sqrt{1-\alpha_\lambda^2}}\,\|J_\mu - \Pi J_\mu\|_\xi$$

where Φr_λ^* is the fixed point of $\Pi T^{(\lambda)}$.

- The fixed point Φr_λ^* depends on λ.

- Note that $\alpha_\lambda \downarrow 0$ as $\lambda \uparrow 1$, so error bound improves as $\lambda \uparrow 1$.

PVI(λ)

$$\Phi r_{k+1} = \Pi T^{(\lambda)}(\Phi r_k) = \Pi\left((1-\lambda)\sum_{t=0}^{\infty}\lambda^t T^{t+1}(\Phi r_k)\right)$$

or

$$r_{k+1} = \arg\min_{r\in\Re^s}\left\|\Phi r - T^{(\lambda)}(\Phi r_k)\right\|_\xi^2$$

- Using algebra and the relation

$$(T^{t+1}J)(i) = E\left\{\alpha^{t+1}J(i_{t+1}) + \sum_{k=0}^{t}\alpha^k g(i_k, i_{k+1}) \;\Big|\; i_0 = i\right\}$$

we can write PVI(λ) as

$$r_{k+1} = \arg\min_{r\in\Re^s}\sum_{i=1}^{n}\xi_i\Bigg(\phi(i)'r - \phi(i)'r_k - \sum_{t=0}^{\infty}(\alpha\lambda)^t E\Big\{d_k(i_t, i_{t+1}) \mid i_0 = i\Big\}\Bigg)^2$$

where

$$d_k(i_t, i_{t+1}) = g(i_t, i_{t+1}) + \alpha\phi(i_{t+1})'r_k - \phi(i_t)'r_k,$$

are the, so called, temporal differences (TD) - they are the errors in satisfying Bellman's equation.

LSPE(λ)

- Replacing the expected values defining PVI(λ) by simulation-based estimates we obtain LSPE(λ).
- It has the form

$$r_{k+1} = \arg\min_{r \in \Re^s} \sum_{t=0}^{k} \left(\phi(i_t)' r - \phi(i_t)' r_k - \sum_{m=t}^{k} (\alpha\lambda)^{m-t} d_k(i_m, i_{m+1}) \right)^2$$

where $(i_0, i_1, \ldots)$ is an infinitely long trajectory generated by simulation.

- Can be implemented with convenient incremental update formulas (see the text).
- Note the λ-tradeoff:
  - As $\lambda \uparrow 1$, the accuracy of the solution Φr_λ^* improves - the error bound to $\|J_\mu - \Phi r_\lambda^*\|_\xi$ improves.
  - As $\lambda \uparrow 1$, the "simulation noise" in the LSPE(λ) iteration (2nd summation term) increases, so longer simulation trajectories are needed for LSPE(λ) to approximate well PVI(λ).

Q-LEARNING I

- Q-learning has two motivations:
  - Dealing with multiple policies simultaneously
  - Using a model-free approach [no need to know $p_{ij}(u)$ explicitly, only to simulate them]
- The Q-factors are defined by

$$Q^*(i,u) = \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha J^*(j)\big), \quad \forall\ (i,u)$$

- In view of $J^* = TJ^*$, we have $J^*(i) = \min_{u \in U(i)} Q^*(i,u)$ so the Q factors solve the equation

$$Q^*(i,u) = \sum_{j=1}^{n} p_{ij}(u)\left(g(i,u,j) + \alpha \min_{u' \in U(j)} Q^*(j,u')\right), \ \forall\ (i,u)$$

- $Q(i,u)$ can be shown to be the unique solution of this equation. **Reason:** This is Bellman's equation for a system whose states are the original states $1,\ldots,n$, together with all the pairs (i,u).
- Value iteration:

$$Q(i,u) := \sum_{j=1}^{n} p_{ij}(u)\left(g(i,u,j) + \alpha \min_{u' \in U(j)} Q(j,u')\right), \ \forall\ (i,u)$$

Q-LEARNING II

- Use any probabilistic mechanism to select sequence of pairs (i_k, u_k) [all pairs (i, u) are chosen infinitely often], and for each k, select j_k according to $p_{i_k j}(u_k)$.

- At each k, Q-learning algorithm updates $Q(i_k, u_k)$ according to

$$Q(i_k, u_k) := \big(1 - \gamma_k(i_k, u_k)\big) Q(i_k, u_k) + \gamma_k(i_k, u_k) \left(g(i_k, u_k, j_k) + \alpha \min_{u' \in U(j_k)} Q(j_k, u') \right)$$

- Stepsize $\gamma_k(i_k, u_k)$ must converge to 0 at proper rate (e.g., like $1/k$).

- **Important mathematical point:** In the Q-factor version of Bellman's equation the order of expectation and minimization is reversed relatively to the ordinary cost version of Bellman's equation:

$$J^*(i) = \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u) \big(g(i, u, j) + \alpha J^*(j) \big)$$

- Q-learning can be shown to converge to true/exact Q-factors (a sophisticated proof).

- **Major drawback:** The large number of pairs (i, u) - no function approximation is used.

Q-FACTOR APROXIMATIONS

- Introduce basis function approximation for Q-factors:

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

- We cannot use LSPE/LSTD because the Q-factor Bellman equation involves minimization/multiple controls.
- An optimistic version of LSPE(0) is possible:
- Generate an infinitely long sequence $\{(i_k, u_k) \mid k = 0, 1, \ldots\}$.
- At iteration k, given r_k and state/control (i_k, u_k):
  (1) Simulate next transition (i_k, i_{k+1}) using the transition probabilities $p_{i_k j}(u_k)$.
  (2) Generate control u_{k+1} from the minimization

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

  (3) Update the parameter vector via

$$r_{k+1} = \arg \min_{r \in \Re^s} \sum_{t=0}^{k} \Big(\phi(i_t, u_t)' r - g(i_t, u_t, i_{t+1}) - \alpha \phi(i_{t+1}, u_{t+1})' r_k \Big)^2$$

Q-LEARNING FOR OPTIMAL STOPPING

- Not much is known about convergence of optimistic LSPE(0).
- Major difficulty is that the projected Bellman equation for Q-factors may not be a contraction, and may have multiple solutions or no solution.
- There is one important case, **optimal stopping**, where this difficulty does not occur.
- Given a Markov chain with states $\{1, \ldots, n\}$, and transition probabilities p_{ij}. We assume that the states form a single recurrent class, with steady-state distribution vector $\xi = (\xi_1, \ldots, \xi_n)$.
- At the current state i, we have two options:
  - Stop and incur a cost $c(i)$, or
  - Continue and incur a cost $g(i, j)$, where j is the next state.
- Q-factor for the continue action:

$$Q(i) = \sum_{j=1}^{n} p_{ij} \Big(g(i,j) + \alpha \min \big\{ c(j), Q(j) \big\} \Big) \triangleq (FQ)(i)$$

- **Major fact:** F is a contraction of modulus α with respect to norm $\| \cdot \|_{\xi}$.

LSPE FOR OPTIMAL STOPPING

- Introduce Q-factor approximation

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

- PVI for Q-factors:

$$\Phi r_{k+1} = \Pi F(\Phi r_k)$$

- LSPE

$$r_{k+1} = \left(\sum_{t=0}^{k} \phi(i_t)\phi(i_t)' \right)^{-1} \sum_{t=0}^{k} \phi(i_t)\Big(g(i_t, i_{t+1}) + \alpha \min\big\{ c(i_{t+1}), \phi(i_{t+1})' r_k \big\} \Big)$$

- Simpler version: Replace the term $\phi(i_{t+1})' r_k$ by $\phi(i_{t+1})' r_t$. The algorithm still converges to the unique fixed point of ΠF (see H. Yu and D. P. Bertsekas, “A Least Squares Q-Learning Algorithm for Optimal Stopping Problems”).