6.231 DYNAMIC PROGRAMMING

LECTURE 23

LECTURE OUTLINE

- Simulation-based policy and value iteration methods
- λ-Least Squares Policy Evaluation method
- Temporal differences implementation
- Policy evaluation by approximate value iteration
- TD(λ)

POLICY AND VALUE ITERATION BY SIMULATION

- There are many proposals, but we will focus on methods for which there is solid theory:

(a) $\mathbf{Policy\ evaluation}$ methods, to be used in exact or approximate policy iteration.

- Here the policy is fixed.
- As a special case we obtain the rollout method.
- The cost of the policy may be calculated in several different forms: (1) For all states (lookup table representation) or (2) Through an approximation architecture (compact representation) or (3) Through on-line simulation as needed (rollout algorithm).

(b) $\mathbf{Value\ iteration}$ w/ function approximation.

- A big restriction is to find a suitable Euclidean norm for which T is a contraction.
- Such a norm can be found in the case where there is only one policy ($T = T_\mu$).
- Q-Learning is a form of on-line simulation-based value iteration method, but the only available theory applies to the lookup table representation case.

SIMULATION-BASED POLICY EVALUATION

- The policy is fixed and one or more long simulation trajectories are generated.
- The weight vector r of an approximation architecture $\tilde{J}(i,r)$ is adjusted using some kind of "least squares scheme" (off-line, or on-line as the simulation trajectories are generated).
- For on-line methods, a sequence $\{r_t\}$ of parameter vectors is generated.
- There is solid theory only for linear approximation architectures (and under some technical assumptions).
- Typical result: In the limit, as the number of simulation-generated transitions goes to ∞, the sequence of generated parameter vectors converges to a limit that solves a related least-squares approximation problem.
- We will focus on so-called $temporal\ difference\ methods,\ \lambda$-least squares and TD($\lambda$), which may be viewed as on-line simulation-based approximate value iteration methods for policy evaluation.

POLICY EVALUATION BY VALUE ITERATION I

• The remainder of this lecture is based on the paper "Improved Temporal Difference Methods with Function Approximation," by Bertsekas, Borkar, and Nedic at

http://www.mit.edu:8001//people/dimitrib/publ.html

• Let J be the cost function associated with a stationary policy in the discounted context, so J is the unique solution of Bellman's Eq., $J(i) = \sum_{j=1}^{n} p_{ij}\big(g(i,j) + \alpha J(j)\big) \equiv (TJ)(i)$. We assume that the associated Markov chain has steady-state probabilities $\overline{p}(i)$ which are all positive.

• If we use a linear approximation architecture $\tilde{J}(i,r) = \phi(i)'r$, the value iteration

$$J_{t+1}(i) = \sum_{j=1}^{n} p_{ij}\big(g(i,j) + \alpha J_t(j)\big) = (TJ_t)(i)$$

is approximated as $\Phi r_{t+1} \approx T(\Phi r_t)$ in the sense

$$r_{t+1} = \arg\min_{r} \sum_{i=1}^{n} w(i) \left(\phi(i)'r - \sum_{j=1}^{n} p_{ij}\big(g(i,j) + \alpha\phi(j)'r_t\big) \right)^2$$

where the $w(i)$ are some positive weights.

POLICY EVALUATION BY VALUE ITERATION II

- Note that, assuming Φ has full rank, r_{t+1} is uniquely obtained by projecting the value iterate $T(\Phi r_t) = P(g + \alpha \Phi r_t)$ on the range space of the matrix Φ, where the projection is with respect to the norm $\|\cdot\|_D$ given by $\|z\|_D = \sqrt{z'Dz}$, and D is diagonal with the $w(i)$ along the diagonal.

- The iteration converges if the mapping T is a contraction with respect to the norm $\|\cdot\|_D$.
Key fact: This is so if the $w(i)$ are equal to the steady state probabilities $\overline{p}(i)$. The limit is the unique r^* satisfying

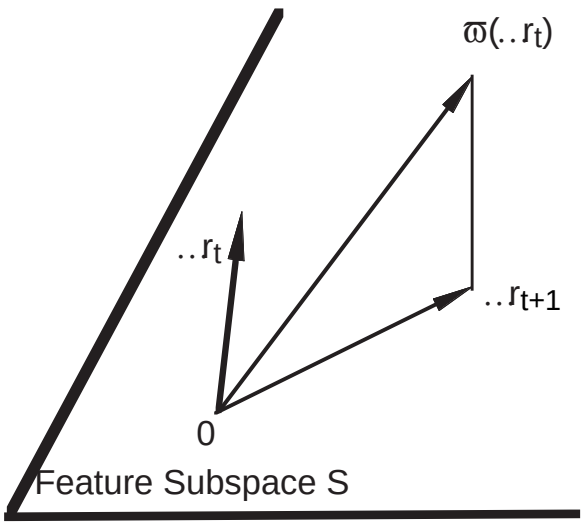
$$r^* = \arg\min_r \sum_{i=1}^{n} w(i) \left(\phi(i)'r - \sum_{j=1}^{n} p_{ij} \left(g(i,j) + \alpha \phi(j)' r^* \right) \right)^2$$

- Simulation-based implementation: Generate an infinitely long trajectory $(i_0, i_1, \ldots)$ using a simulator, and iteratively update r by

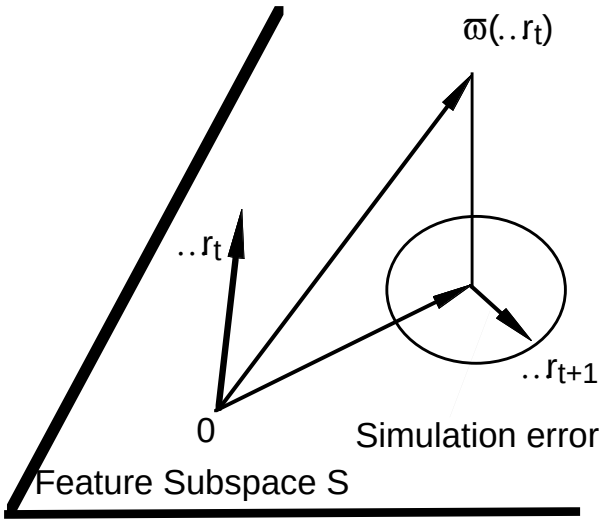
$$r_{t+1} = \arg\min_r \sum_{m=0}^{t} \left(\phi(i_m)'r - g(i_m, i_{m+1}) - \alpha \phi(i_{m+1})' r_t \right)^2$$

This can be shown to converge to the same r^*.

GEOMETRIC INTERPRETATION



Value Iteration with Linear Function Approximation



Simulation-Based Value Iteration with Linear Function Approximation

- The simulation-based implementation yields the (non-simulation) value iterate with linear function approximation [i.e., the projection of $T(\Phi r_t)$] plus stochastic simulation error.

- **Key Convergence Proof Idea:** The simulation error converges to 0 as the simulation trajectory becomes longer. Furthermore, the (non-simulation) value iteration is a convergent linear deterministic algorithm [since it involves a contraction mapping with respect to the weighted norm defined by the steady-state probabilities $\overline{p}(i)$].

USING M-STEP VALUE ITERATION

- For $M \geq 1$, consider the equation

$$J(i) = E\left[\alpha^M J(i_M) + \sum_{k=0}^{M-1} \alpha^k g(i_k, i_{k+1}) \,\middle|\, i_0 = i\right]$$

- This is Bellman's Eq. for a modified problem, involving a Markov chain where each transition corresponds to M transitions of the original, and the cost is calculated using a discount factor α^M and a cost per stage equal to $\sum_{k=0}^{M-1} \alpha^k g(i_k, i_{k+1})$.

- This Bellman equation is also solved uniquely by the same J that solves the ordinary (one-step) Bellman equation $J(i) = E\big[g(i,j) + \alpha J(j)\big]$.

- The corresponding value iteration method is

$$J_{t+1}(i) = E\left[\alpha^M J_t(i_M) + \sum_{k=0}^{M-1} \alpha^k g(i_k, i_{k+1}) \,\middle|\, i_0 = i\right]$$

and can be similarly approximated by simulation.

SIMULATION-BASED M-STEP VALUE ITERATION

- The corresponding simulation-based least-squares implementation is

$$r_{t+1} = \arg\min_r \sum_{m=0}^{t} \left(\phi(i_m)'r - \alpha^M \phi(i_{m+M})' r_t - \sum_{k=0}^{M-1} \alpha^k g(i_{m+k}, i_{m+k+1}) \right)^2$$

- By introducing the temporal differences, defined by

$$d_t(i_k, i_{k+1}) = g(i_k, i_{k+1}) + \alpha\phi(i_{k+1})'r_t - \phi(i_k)'r_t,$$

we can write this iteration as

$$r_{t+1} = \arg\min_r \sum_{m=0}^{t} \left(\phi(i_m)'r - \phi(i_m)'r_t - \sum_{k=m}^{m+M-1} \alpha^{k-m} d_t(i_k, i_{k+1}) \right)^2$$

USING RANDOM STEP VALUE ITERATION

- Consider a version of Bellman's equation where M is random and geometrically distributed with parameter λ, i.e.,

$$\text{Prob}(M = m) = (1-\lambda)\lambda^{m-1}, \qquad m = 1, 2, \ldots$$

- This equation is obtained by multiplying both sides of the M-step Bellman's Eq. with $(1-\lambda)\lambda^{m-1}$, for each m, and adding over m:

$$J(i) = \sum_{m=1}^{\infty} (1-\lambda)\lambda^{m-1} E\left[\alpha^m J(i_m) + \sum_{k=0}^{m-1} \alpha^k g(i_k, i_{k+1}) \,|\, i_0 = i\right.$$

- The corresponding value iteration method is

$$J_{t+1}(i) = \sum_{m=1}^{\infty} (1-\lambda)\lambda^{m-1} E\left[\alpha^m J_t(i_m) + \sum_{k=0}^{m-1} \alpha^k g(i_k, i_{k+1}) \,|\, i_0 = i\right]$$

TEMPORAL DIFFERENCES IMPLEMENTATION

- We can write the random step value iteration as

$$J_{t+1}(i) = J_t(i) + \sum_{k=0}^{\infty} (\alpha\lambda)^k E\Big[g(i_k, i_{k+1}) + \alpha J_t(i_{k+1}) - J_t(i_k) \,|\, i_0 = i\Big]$$

- By using $\phi(i)'r_t$ to approximate J_t, and by replacing $g(i_k, i_{k+1}) + \alpha J_t(i_{k+1}) - J_t(i_k)$ with the temporal differences (TD)

$$d_t(i_k, i_{k+1}) = g(i_k, i_{k+1}) + \alpha\phi(i_{k+1})'r_t - \phi(i_k)'r_t,$$

we obtain the simulation-based least-squares implementation (called λ-least squares policy evaluation method)

$$r_{t+1} = \arg\min_r \sum_{m=0}^{t} \left(\phi(i_m)'r - \phi(i_m)'r_t - \sum_{k=m}^{t} (\alpha\lambda)^{k-m} d_t(i_k, i_{k+1}) \right)^2$$

- Role of the TD: They simplify the formulas.

- Convergence can be shown to an r^* that solves a corresponding least squares problem.

TD(LAMBDA)

- Another method for solving the policy evaluation problem is TD(λ), which uses a parameter $\lambda \in [0,1]$ and generates an infinitely long trajectory $(i_0, i_1, \ldots)$ using a simulator. It iteratively updates r by

$$r_{t+1} = r_t + \gamma_t \left(\sum_{m=0}^{t} (\alpha\lambda)^{t-m} \phi(i_m) \right) d_t(i_t, i_{t+1})$$

where γ_t is a positive stepsize with $\gamma_t \to 0$.

- It can be viewed as a gradient-like method for minimizing the least-squares sum of the preceding λ-least squares method described earlier (see the Bertsekas, Borkar, and Nedic paper).

- For a given value of $\lambda \in [0,1]$, TD(λ) converges to the same limit as the λ-least squares method (under technical assumptions on the choice of γ_t).

- While TD(λ) uses a simpler formula, it tends to be much slower than λ-Least Squares. In practice, it also requires tricky trial and error to settle on good stepsize choices.

TD METHODS: PROPERTIES AND DIFFICULTIES

- As M increases, the M-step Bellman's equation becomes better suited for approximation, because it embodies a longer horizon cost. Thus Φr^* tends to be closer to J when M is large.
- Similarly, Φr^* tends to be closer to J as $\lambda \approx 1$.
- On the other hand, when M or λ is large, the simulation noise inherent in the updates is magnified (more random cost terms are added), and convergence can be very slow. TD(λ) is particularly susceptible to noise, so $\lambda \approx 1$ may be a bad choice. This is less of a problem for the alternative λ-least squares method.
- A serious problem arises when the Markov chain is "slow-mixing," i.e., it takes many transitions for the simulation to reach important parts of the state space. Then if the simulation trajectory is terminated prematurely, the approximation obtained over these parts will be poor. A remedy is to use many long simulation trajectories starting from a set of initial states that adequately covers the state space.