6.231 DYNAMIC PROGRAMMING

LECTURE 6

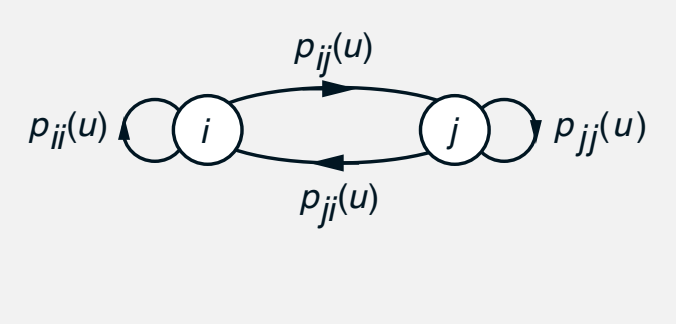
LECTURE OUTLINE

- Review of Q-factors and Bellman equations for Q-factors
- VI and PI for Q-factors
- Q-learning - Combination of VI and sampling
- Q-learning and cost function approximation
- Adaptive dynamic programming
- Approximation in policy space
- Additional topics

REVIEW

DISCOUNTED MDP

- System: Controlled Markov chain with states $i = 1, \ldots, n$ and finite set of controls $u \in U(i)$
- Transition probabilities: $p_{ij}(u)$



- Cost of a policy $\pi = \{\mu_0, \mu_1, \ldots\}$ starting at state i:

$$J_\pi(i) = \lim_{N\to\infty} E\left\{\sum_{k=0}^{N} \alpha^k g\big(i_k, \mu_k(i_k), i_{k+1}\big) \mid i = i_0\right\}$$

with $\alpha \in [0, 1)$

- Shorthand notation for DP mappings

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

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

BELLMAN EQUATIONS FOR Q-FACTORS

- The optimal Q-factors are defined by

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

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

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

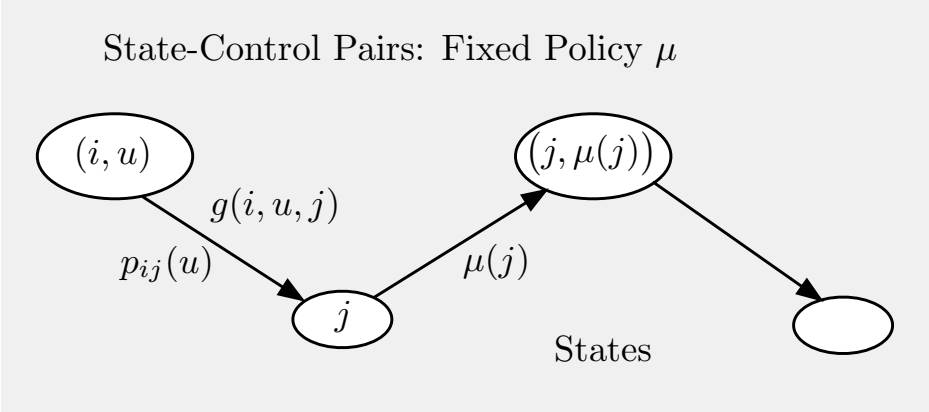
- Equivalently $Q^* = FQ^*$, where

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

- This is Bellman's Eq. for a system whose states are the pairs (i,u)

- Similar mapping F_μ and Bellman equation for a policy μ: $Q_\mu = F_\mu Q_\mu$

BELLMAN EQ FOR Q-FACTORS OF A POLICY



- Q-factors of a policy μ: For all (i,u)

$$Q_\mu(i,u) = \sum_{j=1}^{n} p_{ij}(u)\Big(g(i,u,j) + \alpha Q_\mu\big(j,\mu(j)\big)\Big)$$

Equivalently $Q_\mu = F_\mu Q_\mu$, where

$$(F_\mu Q)(i,u) = \sum_{j=1}^{n} p_{ij}(u)\Big(g(i,u,j) + \alpha Q\big(j,\mu(j)\big)\Big)$$

- This is a linear equation. It can be used for policy evaluation.

- Generally VI and PI can be carried out in terms of Q-factors.

- When done exactly they produce results that are mathematically equivalent to cost-based VI and PI.

WHAT IS GOOD AND BAD ABOUT Q-FACTORS

- All the exact theory and algorithms for costs applies to Q-factors
  - Bellman's equations, contractions, optimality conditions, convergence of VI and PI
- All the approximate theory and algorithms for costs applies to Q-factors
  - Projected equations, sampling and exploration issues, oscillations, aggregation
- A MODEL-FREE (on-line) controller implementation
  - Once we calculate $Q^*(i,u)$ for all (i,u),

$$\mu^*(i) = \arg \min_{u \in U(i)} Q^*(i,u), \qquad \forall\ i$$

  - Similarly, once we calculate a parametric approximation $\tilde{Q}(i,u;r)$ for all (i,u),

$$\tilde{\mu}(i) = \arg \min_{u \in U(i)} \tilde{Q}(i,u;r), \qquad \forall\ i$$

- The main bad thing: Greater dimension and more storage! (It can be used for large-scale problems only through aggregation, or other approximation.)

Q-LEARNING

Q-LEARNING

- In addition to the approximate PI methods adapted for Q-factors, there is an important additional algorithm:
  - Q-learning, a sampled form of VI (a stochastic iterative algorithm).
- Q-learning algorithm (in its classical form):
  - Sampling: Select sequence of pairs (i_k, u_k) [use any probabilistic mechanism for this, but all (i, u) are chosen infinitely often].
  - Iteration: For each k, select j_k according to $p_{i_k j}(u_k)$. Update just $Q(i_k, u_k)$:

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

    Leave unchanged all other Q-factors.
  - Stepsize conditions: $\gamma_k \downarrow 0$
- We move $Q(i,u)$ in the direction of a sample of

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

NOTES AND QUESTIONS ABOUT Q-LEARNING

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

- Model free implementation. We just need a simulator that given (i,u) produces next state j and cost $g(i,u,j)$
- Operates on only one state-control pair at a time. Convenient for simulation, no restrictions on sampling method. (Connection with asynchronous algorithms.)
- Aims to find the (exactly) optimal Q-factors.
- Why does it converge to Q^*?
- Why can't I use a similar algorithm for optimal costs (a sampled version of VI)?
- Important mathematical (fine) point: In the Q-factor version of Bellman's equation the order of expectation and minimization is reversed relative to the cost version of Bellman's equation:

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

CONVERGENCE ASPECTS OF Q-LEARNING

- Q-learning can be shown to converge to true/exact Q-factors (under mild assumptions).
- The proof is sophisticated, based on theories of stochastic approximation and asynchronous algorithms.
- Uses the fact that the Q-learning map F:

$$(FQ)(i,u) = E_j\Big\{g(i,u,j) + \alpha \min_{u'} Q(j,u')\Big\}$$

is a sup-norm contraction.

- Generic stochastic approximation algorithm:
  - Consider generic fixed point problem involving an expectation:

$$x = E_w\big\{f(x,w)\big\}$$

  - Assume $E_w\big\{f(x,w)\big\}$ is a contraction with respect to some norm, so the iteration

$$x_{k+1} = E_w\big\{f(x_k,w)\big\}$$

    converges to the unique fixed point
  - Approximate $E_w\big\{f(x,w)\big\}$ by sampling

STOCH. APPROX. CONVERGENCE IDEAS

- Generate a sequence of samples $\{w_1, w_2, \ldots\}$, and approximate the convergent fixed point iteration $x_{k+1} = E_w\big\{f(x_k, w)\big\}$
- At each iteration k use the approximation

$$x_{k+1} = \frac{1}{k}\sum_{t=1}^{k} f(x_k, w_t) \approx E_w\big\{f(x_k, w)\big\}$$

- A major flaw: it requires, for each k, the computation of $f(x_k, w_t)$ for all values w_t, $t = 1, \ldots, k$.
- This motivates the more convenient iteration

$$x_{k+1} = \frac{1}{k}\sum_{t=1}^{k} f(x_t, w_t), \qquad k = 1, 2, \ldots,$$

that is similar, but requires much less computation; it needs only one value of f per sample w_t.

- By denoting $\gamma_k = 1/k$, it can also be written as

$$x_{k+1} = (1 - \gamma_k)x_k + \gamma_k f(x_k, w_k), \quad k = 1, 2, \ldots$$

- Compare with Q-learning, where the fixed point problem is $Q = FQ$

$$(FQ)(i, u) = E_j\big\{g(i, u, j) + \alpha \min_{u'} Q(j, u')\big\}$$

Q-LEARNING COMBINED WITH OPTIMISTIC PI

- Each Q-learning iteration requires minimization over all controls $u' \in U(j_k)$:

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

- To reduce this overhead we may consider replacing the minimization by a simpler operation using just the "current policy" μ_k

- This suggests an asynchronous sampled version of the optimistic PI algorithm which policy evaluates by

$$Q_{k+1} = F_{\mu^k}^{m_k} Q_k,$$

and policy improves by $\mu^{k+1}(i) \in \arg\min_{u \in U(i)} Q_{k+1}(i,u)$

- This turns out not to work (counterexamples by Williams and Baird, which date to 1993), but a simple modification of the algorithm is valid

- See a series of papers starting with D. Bertsekas and H. Yu, "Q-Learning and Enhanced Policy Iteration in Discounted Dynamic Programming," Math. of OR, Vol. 37, 2012, pp. 66-94

Q-FACTOR APPROXIMATIONS

- We introduce basis function approximation:

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

- We can use approximate policy iteration and LSTD/LSPE for policy evaluation

- Optimistic policy iteration methods are frequently used on a heuristic basis

- An extreme example: Generate trajectory $\{(i_k, u_k) \mid k = 0, 1, \ldots\}$ as follows.

- 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

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

  (3) Update the parameter vector via

$$r_{k+1} = r_k - (\text{LSPE or TD-like correction})$$

- Complex behavior, unclear validity (oscillations, etc). There is solid basis for an important special case: optimal stopping (see text)

BELLMAN EQUATION ERROR APPROACH

- Another model-free approach for approximate evaluation of policy μ: Approximate $Q_\mu(i,u)$ with $\tilde{Q}_\mu(i,u;r_\mu) = \phi(i,u)'r_\mu$, obtained from

$$r_\mu \in \arg\min_r \left\| \Phi r - F_\mu(\Phi r) \right\|_\xi^2$$

where $\|\cdot\|_\xi$ is Euclidean norm, weighted with respect to some distribution ξ.

- Implementation for deterministic problems:

  (1) Generate a large set of sample pairs (i_k, u_k), and corresponding deterministic costs $g(i_k,u_k)$ and transitions $\big(j_k, \mu(j_k)\big)$ (a simulator may be used for this).

  (2) Solve the linear least squares problem:

$$\min_r \sum_{(i_k,u_k)} \Big| \phi(i_k,u_k)'r - \big(g(i_k,u_k) + \alpha\phi\big(j_k,\mu(j_k)\big)'r\big) \Big|^2$$

- For stochastic problems a similar (more complex) least squares approach works. It is closely related to LSTD (but less attractive; see the text).

- Because this approach is model-free, it is often used as the basis for adaptive control of systems with unknown dynamics.

ADAPTIVE CONTROL BASED ON ADP

LINEAR-QUADRATIC PROBLEM

- System: $x_{k+1} = Ax_k + Bu_k, \quad x_k \in \Re^n, u_k \in \Re^m$
- Cost: $\sum_{k=0}^{\infty}(x_k' Q x_k + u_k' R u_k)$, $Q \geq 0$, $R > 0$
- Optimal policy is linear: $\mu^*(x) = Lx$
- The Q-factor of each linear policy μ is quadratic:

$$Q_\mu(x,u) = \begin{pmatrix} x' & u' \end{pmatrix} K_\mu \begin{pmatrix} x \\ u \end{pmatrix} \qquad (*)$$

- We will consider A and B unknown
- We represent Q-factors using as basis functions all the quadratic functions involving state and control components

$$x^i x^j, \qquad u^i u^j, \qquad x^i u^j, \qquad \forall\ i,j$$

These are the "rows" $\phi(x,u)'$ of Φ

- The Q-factor Q_μ of a linear policy μ can be exactly represented within the approximation subspace:

$$Q_\mu(x,u) = \phi(x,u)' r_\mu$$

where r_μ consists of the components of K_μ in $(*)$

PI FOR LINEAR-QUADRATIC PROBLEM

- Policy evaluation: r_μ is found by the Bellman error approach

$$\min_r \sum_{(x_k,u_k)} \Big|\phi(x_k,u_k)'r - \big(x_k'Qx_k + u_k'Ru_k + \phi\big(x_{k+1},\mu(x_{k+1})\big)'r\big)\Big|^2$$

where (x_k, u_k, x_{k+1}) are many samples generated by the system or a simulator of the system.

- Policy improvement:

$$\overline{\mu}(x) \in \arg\min_u \big(\phi(x,u)'r_\mu\big)$$

- Knowledge of A and B is not required

- If the policy evaluation is done exactly, this becomes exact PI, and convergence to an optimal policy can be shown

- The basic idea of this example has been generalized and forms the starting point of the field of adaptive dynamic programming

- This field deals with adaptive control of continuous-space, (possibly nonlinear) dynamic systems, in both discrete and continuous time

APPROXIMATION IN POLICY SPACE

APPROXIMATION IN POLICY SPACE

- We parametrize policies by a vector $r = (r_1, \ldots, r_s)$ (an approximation architecture for policies).
- Each policy $\tilde{\mu}(r) = \{\tilde{\mu}(i; r) \mid i = 1, \ldots, n\}$ defines a cost vector $J_{\tilde{\mu}(r)}$ (a function of r).
- We optimize some measure of $J_{\tilde{\mu}(r)}$ over r.
- For example, use a random search, gradient, or other method to minimize over r

$$\sum_{i=1}^{n} \xi_i J_{\tilde{\mu}(r)}(i),$$

where $\xi_1, \ldots, \xi_n$ are some state-dependent weights.

- An important special case: Introduce cost approximation architecture $V(i; r)$ that defines indirectly the parametrization of the policies

$$\tilde{\mu}(i; r) = \arg \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i, u, j) + \alpha V(j; r)\big), \ \forall\, i$$

- This introduces state features into approximation in policy space.
- A policy approximator is called an actor, while a cost approximator is also called a critic. An actor and a critic may coexist.

APPROXIMATION IN POLICY SPACE METHODS

- Random search methods are straightforward and have scored some impressive successes with challenging problems (e.g., tetris).
  - At a given point/r they generate a random collection of neighboring r. They search within the neighborhood for better points.
  - Many variations (the cross entropy method is one).
  - They are very broadly applicable (to discrete and continuous search spaces).
  - They are idiosynchratic.
- Gradient-type methods (known as policy gradient methods) also have been used extensively.
  - They move along the gradient with respect to r of

$$\sum_{i=1}^{n} \xi_i J_{\tilde{\mu}(r)}(i)$$

  - There are explicit gradient formulas which can be approximated by simulation.
  - Policy gradient methods generally suffer by slow convergence, local minima, and excessive simulation noise.

COMBINATION WITH APPROXIMATE PI

- Another possibility is to try to implement PI within the class of parametrized policies.

- Given a policy/actor $\mu(i;r_k)$, we evaluate it (perhaps approximately) with a critic that produces $\tilde{J}_\mu$, using some policy evaluation method.

- We then consider the policy improvement phase

$$\overline{\mu}(i) \in \arg\min_u \sum_{j=1}^n p_{ij}(u)\big(g(i,u,j)+\alpha\tilde{J}_\mu(j)\big), \quad \forall\ i$$

and do it approximately via parametric optimization

$$\min_r \sum_{i=1}^n \xi_i \sum_{j=1}^n p_{ij}\big(\overline{\mu}(i;r)\big)\Big(g\big(i,\overline{\mu}(i;r),j\big)+\alpha\tilde{J}_\mu(j)\Big)$$

where ξ_i are some weights.

- This can be attempted by a gradient-type method in the space of the parameter vector r.

- Schemes like this been extensively applied to continuous-space deterministic problems.

- Many unresolved theoretical issues, particularly for stochastic problems.

FINAL WORDS

TOPICS THAT WE HAVE NOT COVERED

- Extensions to discounted semi-Markov, stochastic shortest path problems, average cost problems, sequential games ...
- Extensions to continuous-space problems
- Extensions to continuous-time problems
- Adaptive DP - Continuous-time deterministic optimal control. Approximation of cost function derivatives or cost function differences
- Random search methods for approximate policy evaluation or approximation in policy space
- Basis function adaptation (automatic generation of basis functions, optimal selection of basis functions within a parametric class)
- Simulation-based methods for general linear problems, i.e., solution of linear equations, linear least squares, etc - Monte-Carlo linear algebra

CONCLUDING REMARKS

- There is no clear winner among ADP methods
- There is interesting theory in all types of methods (which, however, does not provide ironclad performance guarantees)
- There are major flaws in all methods:
  - Oscillations and exploration issues in approximate PI with projected equations
  - Restrictions on the approximation architecture in approximate PI with aggregation
  - Flakiness of optimization in policy space approximation
- Yet these methods have impressive successes to show with enormously complex problems, for which there is often no alternative methodology
- There are also other competing ADP methods (rollout is simple, often successful, and generally reliable; approximate LP is worth considering)
- Theoretical understanding is important and nontrivial
- Practice is an art and a challenge to our creativity!

THANK YOU