## 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
- Approximation in policy space

### DISCOUNTED MDP

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

$$p_{ij}(u) \underbrace{p_{ij}(u)}_{i} \underbrace{p_{jj}(u)}_{j} p_{jj}(u)$$

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

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

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

• Shorthand notation for DP mappings

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

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

## THE TWO MAIN ALGORITHMS: VI AND PI

• Value iteration: For any  $J \in \Re^n$ 

$$J^*(i) = \lim_{k \to \infty} (T^k J)(i), \qquad \forall \ i = 1, \dots, n$$

• Policy iteration: Given  $\mu^k$ - Policy evaluation: Find  $J_{\mu^k}$  by solving

$$J_{\mu^{k}}(i) = \sum_{j=1}^{n} p_{ij} \left( \mu^{k}(i) \right) \left( g \left( i, \mu^{k}(i), j \right) + \alpha J_{\mu^{k}}(j) \right), \quad i = 1, \dots, n$$

or 
$$J_{\mu^k} = T_{\mu^k} J_{\mu^k}$$

- Policy improvement: Let  $\mu^{k+1}$  be such that

$$\mu^{k+1}(i) \in \arg\min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u) (g(i, u, j) + \alpha J_{\mu^k}(j)), \quad \forall i$$

or  $T_{\mu^{k+1}}J_{\mu^k} = TJ_{\mu^k}$ 

• We discussed approximate versions of VI and PI using projection and aggregation

• We focused so far on cost functions and approximation. We now consider Q-factors.

## **BELLMAN EQUATIONS FOR** *Q***-FACTORS**

• The optimal *Q*-factors are defined by

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

• 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  $F_{\mu}$  and Bellman equation for a policy  $\mu$ :  $Q_{\mu} = F_{\mu}Q_{\mu}$ 



#### SUMMARY OF BELLMAN EQS FOR Q-FACTORS



• Optimal Q-factors: For all (i, u)

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

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

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

• Q-factors of a policy  $\mu$ : For all (i, u)

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

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

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

## 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),

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

• The main bad thing: Greater dimension and more storage! (Can be used for large-scale problems through aggregation, or other cost function approximation.)

## **Q-LEARNING**

• In addition to the approximate PI methods adapted for Q-factors, there is an important additional algorithm:

- Q-learning, which can be viewed as a sampled form of VI
- Q-learning algorithm (in its classical form):
  - Sampling: Select sequence of pairs  $(i_k, u_k)$ , and for each k, select  $j_k$  according to  $p_{i_k j}(u_k)$ . (Use any probabilistic mechanism for this, but all pairs (i, u) are chosen infinitely often.)
  - Iteration: Update just  $Q(i_k, u_k)$ :

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

All other Q(i, u),  $(i, u) \neq (i_k, u_k)$  are left unchanged.

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

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

• Model free implementation. 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.

- Aims to find the (exactly) optimal Q-factors.
- Why does it converge to  $Q^*$ ?

• Why can't I do the same thing for optimal costs?

• 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:

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

## **CONVERGENCE ASPECTS OF Q-LEARNING**

- Q-learning can be shown to converge to true/exact Q-factors (under mild assumptions)
- Proof is sophisticated, using theories of stochastic approximation and asynchronous algorithms
- Uses the fact that the Q-learning map F is a sup-norm contraction and has the form

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

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

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

- Assume  $E_w\{f(x,w)\}$  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  $E_w\{f(x,w)\}$  by sampling

#### STOCH. APPROXIMATION CONVERGENCE IDEAS

• For each k, obtain samples  $\{w_1, \ldots, w_k\}$  and use the approximation

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

• This iteration approximates the convergent fixed point iteration  $x_{k+1} = E_w \{f(x_k, w)\}$ 

• A major flaw: it requires, for each k, the computation of  $f(x_k, w_t)$  for **all** values  $w_t, t = 1, ..., k$ .

• This motivates the more convenient iteration

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

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

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

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

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

## *Q***-FACTOR APROXIMATIONS**

• We introduce basis function approximation:

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

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

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

• Example: Generate trajectory  $\{(i_k, u_k) \mid k = 0, 1, \ldots\}$ .

- 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

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

(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)

## **APPROXIMATION IN POLICY SPACE**

• We parameterize policies by a vector  $r = (r_1, \ldots, r_s)$  (an approximation architecture for policies)

• Each policy  $\tilde{\mu}(r) = \{\tilde{\mu}(i;r) \mid i = 1, ..., n\}$ 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

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

where  $(p_1, \ldots, p_n)$  is some probability distribution over the states.

• An important special case: introduce cost approximation architecture V(i, r) that defines indirectly the parameterization of the policies

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

• Brings in features to approximation in policy space

## **APPROXIMATION IN POLICY SPACE METHODS**

• Random search methods are straightforward and have scored some impressive successes with challenging problems (e.g., tetris)

• Gradient-type methods (known as policy gradient methods) also have been worked on extensively.

• They move along the gradient with respect to r of

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

• There are explicit gradient formulas which have been approximated by simulation

• Policy gradient methods generally suffer by slow convergence, local minima, and excessive simulation noise

# FINAL ACCOUNTING: COMPARISONS

• 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 no alternative methodology

• There are also other competing ADP methods (rollout is simple, often successful, and generally reliable)

• Theoretical understanding is important and nontrivial

• Practice is an art and a challenge to our creativity!