1 Refresher: DP

2 Shortest Path Problem: DP Backward Recursion

The backward recursive equations are

$$f_i = \min_j [c_{ij} + f_j],\ i < j \tag{1}$$

Or, If stage t is specified then

$$f_t(i) = \min_j [c_{ij} + f_{t+1}(j)],\ i < j \tag{2}$$

3 Shortest Path Problem: Forward Calculation (Reaching)

$$f_j = \min_i [c_{ij} + f_i],\ i < j \tag{3}$$

Or, if stage t is specified then

$$f_{t+1}(j) = \min_i [c_{ij} + f_t(i)],\ i < j \tag{4}$$

4 DP Elements

To solve any sequential decision making problem with DP one must identify and define the following elements. The main elements of the DP recursive equation are

1. Stage t: For finite horizon problems it is usually time t (but not always true). For infinite horizon problems its always time.

2. State i or j and in general S: Usually its the AVAILABLE resource that need to be allocated at stage t. However, there are problems where the state is the inventory at hand, or the price of an asset in an asset acquisition problem (e.g. stock market, oil prices), and so on. The next state of a system depends on the current state and the action taken in the current state.

3. Action or decision x or a or k: The action (x_t or a_t) taken in (state i at stage t or S_t) that moves the system to state (j at stage $t+1$ or S_{t+1}) under the influence of a exogenous process W. This means S_{t+1} is a function of (S_t, x_t, W_{t+1}).

4. Exogenous process W: Its either deterministic or stochastic (uncertainty). For example, in inventory control problems it is the demand.

5. Contribution function (reward or cost): C or c or r and depending on the type of problem you will use $C(i, a, j)$ or $C(S, x)$ or $c_{i,j}$ and so on. This is the immediate contribution of an action taken in a particular state. Also known as the one-step cost or reward function.

6. Value function of a state f or V or R: The long-run value $f_t(S)$ of being in a state S at stage t. This is used in making decisions in state S.

7. Transition probability $p(i, a, j)$ or $P(i, a, j)$ or $p_{ij}(k)$: This is only for stochastic DP where it denotes the probability of transitioning from state i to state j under action a.

8. Objective function: This is the max or min operator that acts on the value function.

5 Deterministic DP-Finite Horizon-Modified version with action x

$$f_t(i) = \min_x [c(i,x) + f_{t+1}(j)],\ i < j \tag{5}$$

6 Incorporating time value of money

$$f_t(i) = \min_x [c(i,x) + \beta f_{t+1}(j)],\ i < j \tag{6}$$

where $0 < \beta < 1$.

7 Deterministic DP-Infinite Horizon

$$f_t = \min_k [\alpha^k f_{t+1} + R_k] \tag{7}$$

7.1 Steps in value iteration

$$f^{b+1} = \min_k [\alpha^k f^b + R_k] \tag{8}$$

1. Set b=0.
2. Choose f^0 to be any number.
3. Find f^1, f^2, $\cdots$
4. Terminate when $|f^{b+1} - f^b| < \epsilon$, where ϵ is a very small number (such as 0.01).

8 Probabilistic or Stochastic DP-Finite Horizon

a) The next state is certain but the reward/cost obtained in the current state is stochastic,

$$f_t(i) = \min_x [Expected\ Cost\ c(i,x) + \sum_j f_{t+1}(j)], \tag{9}$$

where

$$c(i,x) = \sum_j p(i,x,j)c(i,x,j) \tag{10}$$

b) The next state is uncertain and the reward/cost obtained in the current state is stochastic. In general

$$f_t(i) = \min_x [Expected\ Cost\ c(i,x) + \sum_j p(i,x,j) f_{t+1}(j)], \tag{11}$$

c) There are problems where $c(i,x)$ is fixed (not an expected value) and the next state is uncertain.

$$f_t(i) = \min_x [c(i,x) + \sum_j p(i,x,j) f_{t+1}(j)], \tag{12}$$

d) There are also problems where $c(i,x)$ does not exist. The next state is obviously uncertain.

$$f_t(i) = \min_x [\sum_j p(i,x,j) f_{t+1}(j)], \tag{13}$$

Stochastic DP (finite horizon) is also represented as decision trees.

9 Incorporating time value of money

$$f_t(i) = \min_x [Expected\ Cost\ c(i,x) + \beta \sum_j p(i,x,j) f_{t+1}(j)], \tag{14}$$

where $0 < \beta < 1$.

10 Probabilistic or Stochastic DP-Infinite Horizon

10.1 Bellman's optimality Equation for average cost/reward for finite state but infinite horizon

Define $V_i^n(R)$ = Total cost of operating the system for n steps starting in state i and following policy R. R is a vector of all actions that corresponds to each state.

Let M denote the total number of states. Since every state is reachable infinitely often, the notion of iteration number n is introduced and stage (which is often time) index t is dropped.

The long run **average** expected cost **per unit time** following policy R (as n tends to infinity) is $g(R)$

$$g(R) = C_{ik} - V_i(R) + \sum_{j=0}^{M} P_{ij}(k) V_j(R) \quad \forall i \tag{15}$$

This is called the Bellman's optimality equation for long run average cost/reward for a system that moves from state i to state j under action k and a transition probability $P_{ij}(k)$. The above Bellman's equation is independent of n and g(R), $V_i(R)$ stabilizes as $n \to \infty$. The Bellman's equation for n-step transition to reach the stabilized g(R) and $V_i(R)$ is as follows

$$g(R) = C_{ik} - V_i^n(R) + \sum_{j=0}^{M} P_{ij}(k) V_j^{n-1}(R), \quad \forall i \tag{16}$$

Note that there are M+1 equations (state i=0,1,..,M) M+2 unknowns which are

$$g(R), V_0(R), .., V_M(R) \tag{17}$$

To solve the Bellman's equation to get the optimal decision in each state, we assume

$$V_M(R) = 0 \tag{18}$$

(similar to backward recursion but M^{th} state is not an end state. Remember, each state is reachable infinitely often). So, we have M+1 unknowns. We are performing a forward algorithm since this is an infinite horizon problem.

11 Solution to Bellman's equation

There are 2 ways to solve: Value Iteration, Policy Iteration.

11.1 Policy Iteration

Step 1: Value Determination. We have (M+1) equations and (M+2) variables.
Assume an arbitrary policy and set iteration index n=1. We are solving this as $n \to \infty$.

$$R^1 = (R_0^1, ..., R_M^1) \tag{19}$$

Let

$$V_M(R) = 0 \tag{20}$$

Solve (M+1) equations and (M+1) variables in this following Bellman's equation

$$V_i(R^n) = C_{ik} - g(R^n) + \sum_{j=0}^{M} P_{ij}(k) V_j(R^n), \ \forall i \tag{21}$$

Step2: Policy Determination New Policy

$$R^{n+1} = (R_i^{n+1})_{\forall i} = arg \min_k [C_{ik} - V_i(R^n) + \sum_{j=0}^{M} P_{ij}(k) V_j(R^n)], \ \forall i \tag{22}$$

If

$$R^n \neq R^{n+1} \tag{23}$$

then set $n = n + 1$ and goto step 1, else stop. g(R) gives the long run average cost/reward. R^* is the optimal policy.

11.2 Example: Solving MDP with Policy Iteration

See hand out

11.3 Value Iteration

Policy iteration gets complicated as the number of system states grow. Calculations are very tedious and solving simultaneous equations is very cumbersome. So we use Value Iteration.
Step 1:

$$V^0 = (V_0^0, V_1^0, V_2^0, \cdots, V_M^0) = 0 \tag{24}$$

Set n=0 and ϵ, which is a small number
Step 2: Find new values of V_i^{n+1}

$$V_i^n = \min_k [C_{ik} - g + \sum_{j=0}^{M} P_{ij}(k) V_j^{n-1}], \ \forall i \tag{25}$$

Since g is not known you can assign any V_i^{n-1} value to g and use the same value within an iteration, and the value of the same i from $n - 1^{th}$ iteration between iterations.

Step 3: Policy Determination

$$R^{n+1} = (R_i^{n+1})_{\forall i} = arg\min_k [C_{ik} - V_i^{n-1} + \sum_{j=0}^{M} P_{ij}(k) V_j^{n-1}] \tag{26}$$

Step 4: If span of

$$|V^n - V^{n-1}| < \epsilon \tag{27}$$

then STOP. $R^n = R^*$ which is the optimal Policy. Else, set n=n+1 and goto step 2.

11.4 Example for value iteration average cost/reward

See Excel worksheet.

11.5 Summary

Bellman's equation for average cost/reward for finite state but infinite horizon (policy iteration)

$$V_i(R) = C_{ik} - g(R) + \sum_{j=0}^{M} P_{ij}(k) V_j(R), \quad \forall i \tag{28}$$

Where $V_i(R)$ is the total expected cost of starting in state i and following through n steps with policy R (R is a vector of k values, one k for each i). i, j are states, k - actions.

$g(R)$ - Long Run average cost per unit time of following policy R. C_{ik} is the immediate cost of action k in state i. $P_{ij}(k)$ is the transition probability from state i to state j following policy R (or action k in state i) .

12 Bellman's equation for discounted cost

Let $V_i^n(R)$ be the expected total discounted cost starting in state i and evolving over n steps and following policy R.

$$V_i(R) = C_{ik} + \beta \sum_{j=0}^{M} P_{ij}(k) V_j(R) \quad \forall i \tag{29}$$

C_{ik} - Cost for first observed period under R and $\beta \sum_{j=0}^{M} P_{ij}(k) V_j^{n-1}(R)$ is the expected total discounted cost by evolving over n-1 steps. β is the discount factor $0 < \beta < 1$ (time value of money).

So we have M+1 equations and M+1 unknown variables.

12.1 Policy Iteration for discounted cost criteria

We are solving this as $n \to \infty$. Step 1: Value determination. Set n=1, For an arbitrarily chosen policy

$$R_1 = (R_0^1, R_1^1, R_2^1, \cdots, R_M^1) \tag{30}$$

$$V_i(R^n) = C_{ik} + \beta \sum_{j=0}^{M} P_{ij}(k) V_j(R^n) \quad \forall i \tag{31}$$

Solve M+1 simultaneous equations.
Step 2: Policy determination

$$R^{n+1} = arg\min_k [C_{ik} + \beta \sum_{j=0}^{M} P_{ij}(k) V_j(R^n)] \tag{32}$$

Step 3: If $R^{n+1} = R^n$ stop, else n=n+1 and goto step 1

12.2 Example for policy iteration -Discounted cost criteria

See hand out.

12.3 Value Iteration for discounted cost/reward

Step 1: Set n=0 Choose $V_i^0 = 0, \ \forall i$
Step 2: Evaluate

$$V_i^n = \min_k [C_{ik} + \beta \sum_{j=0}^{M} P_{ij}(k) V_j^{n-1}] \tag{33}$$

Step 3: Check

$$|V_i^n - V_i^{n-1}| < \epsilon \quad \forall i \tag{34}$$

If True - STOP and get policy by using

$$R^n = arg\min_k [C_{ik} + \beta \sum_{j=0}^{M} P_{ij}(k) V_j^{n-1}] \tag{35}$$

Else increment n=n+1 goto and step 2.

13 SMDP: Average cost criteria

For Markov process, $t_{ij}(k)$ is no longer equal. Cannot use value iteration because $g(R)$ cannot be any arbitrary V_i. Solve using policy iteration

$$V_i(R) = C_{ik} - g(R) t_{ik} + \sum_{j=0}^{M} P_{ij}(k) V_j(R), \ \forall i \tag{36}$$

$$t_{ik} = \sum_{j=0}^{M} P_{ij}(k) t_{ij}(k) \tag{37}$$

$$c_{ik} = \sum_{j=0}^{M} P_{ij}(k) c_{ij}(k) \tag{38}$$

14 SMDP: Discounted cost criteria if $t_{ij}(k)$ is exponentially distributed

Solve using policy iteration

$$V_i(R) = C_{ik} + \sum_{j=0}^{M} e^{-\gamma t_{ij}(k)} P_{ij}(k) V_j(R) \quad \forall i \tag{39}$$

Solve using value iteration

$$V_i^n = \min_k [C_{ik} + \sum_{j=0}^{M} e^{-\gamma t_{ij}(k)} P_{ij}(k) V_j^{n-1}] \quad \forall i \tag{40}$$

Match $e^{-\gamma t_{ij}(k)}$ to $\beta^{t_{ij}(k)}$ to determine γ. Value iteration can be used.

15 Approximate DP

ADP (learning-based MDP) is needed in place of DP (MDP) for sequential decision making problems if the following occurs

1. Transition probabilities are not known
2. Number of system states is large (remember the difference between high dimensional and large system state)

15.1 Issues due the above

1. Since transition probabilities are not known (curse of modeling), a simulation framework is needed to generate the Markov jumps. **Solution:** Use a learning-version of Bellman's equation, which will only yield near-optimal solutions under certain conditions. This will cause convergence to a band but not to a point like the g(R) as in MDP. This will cause the results to be approximate.
2. Value functions cannot be stored due to high computational storage requirement for the large number of states (curse of dimensionality). Also, reading and writing into a large matrix of states and its value function values is not computationally feasible. Therefore, value functions must be estimated. **Solution:** value function approximation with or without state space aggregation. This will cause the results to be approximate.
3. Synchronous update as in MDP where the values of all states are updated in every iteration is not possible due to high computational time to update the values because of the large number of states: **Solution:** Asynchronous update of only one state in each iteration, which introduces the notion of sampling one realization of the Markov jump. This will cause slower convergence and will need millions of iterations.
4. Since values of states are no longer stored, the actions are not tractable. A scheme to find the best actions must be devised. **Solution:** Create an argmin or argmax equation.

5. Since ADP is an unsupervised learning scheme, exploration, learning and learnt phases are required to obtain the near-optimal solutions.

To resolve the above issues and to apply DP to large scale problems, take OR 774 advanced DP.