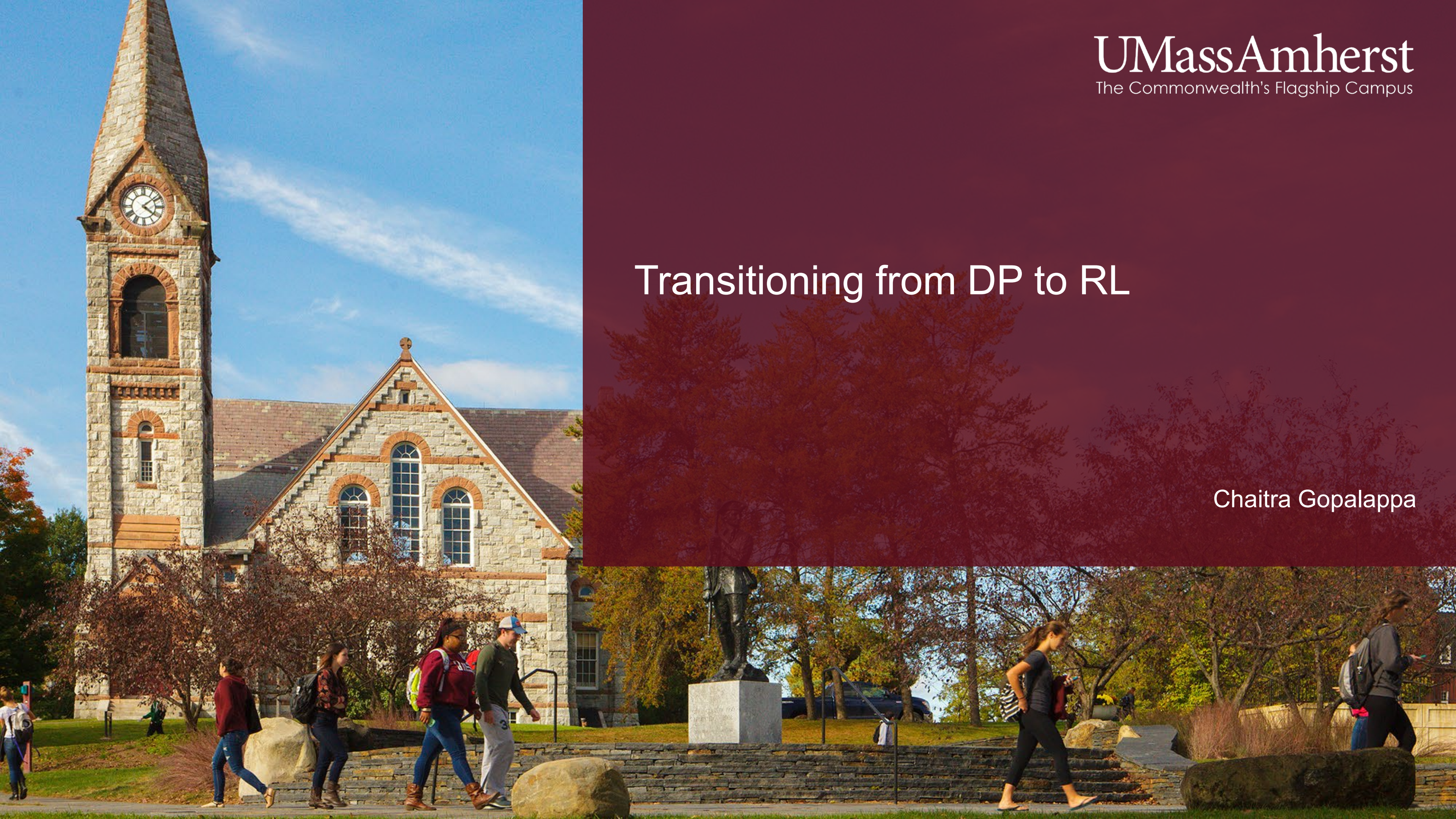
UMassAmherst
The Commonwealth's Flagship Campus



Transitioning from DP to RL

Chaitra Gopalappa

References

- Sutton and Barto, Second edition
  - Chapter 2, Section 2.4 and 2.5
  - Chapter 3
  - Chapter 6, Section 6.4 and 6.5

- A factory is interested in determining whether to order parts (raw material) or not based on inventory at the end of each week. Below are the details of this factory
  - N products manufactured;
    - Demand for each i~ Poisson(λ_i per week)
    - Maximum backorder = B_i
    - Revenue per product = r_i
  - Manufacturing these products requires a total of M number of parts
    - Maximum inventory capacity for each part j = k_j
    - Total inventory capacity for all parts = K
    - Shipping and ordering costs for each part = S_j
    - Delivery times from supplier for each part ~ Normal(μ_j, σ_j)
  - Not all parts are needed for all products
  - Parts assembled in multiple steps
    - Assembly time for step k of product i ~ Normal($\bar{\mu}_{ik}, \bar{\sigma}_{ik}$)

Challenges with DP: #1

- Curse of modeling
  - Generating TPM and TRewardM
  - Transitions are a function of multiple random variables
  - Difficult to estimate complex pdfs
- Solution: maximum likelihood estimation through historical data or simulation generated data
  - $P(i, a, j) \approx \frac{w(i,a,j)}{V(i,a)}$
  - $V(i, a)$= number of time action a taken when system is in state i
  - $w(i, a, j)$ = number of times transition was to state j if action a taken when system was in state i
- If MLE feasible use DP: as DP guarantees optimality
  - Example: in factory problem in slide 1, N=2, M=2 (when problem is small)

Challenges with DP: #2

- In factory problem in slide 1, if N=tens or hundreds, M=hundreds or thousands?
- Curse of dimensionality
  - Large number of states
  - Estimation issue: # of samples needed to estimate TPM and TRM will be large
- Solution:
  - Reinforcement learning: method of performing DP within a simulator

Solution algorithms to solve 'discrete time' MDP

- DP policy iteration
- DP value iteration

Dynamic programming (model-based)

- Model building algorithms: Build TPM and TRM using simulation and then apply DP
- Q-learning
- DQL: deep learning + RL
- Policy gradient

Reinforcement learning (model-free space)

Theory behind RL: Transitioning DP to RL (discounted total reward)

1. Bellman optimality equation to derive Q-factors
    1. Q(s,a): Action-value functions (recollect we used state-value functions V(s) in DP)
2. Q-factor version of Bellman optimality equation
3. Q-factor value iteration – still DP domain- but updating Q-factors instead of value function
4. Robbins- Monro algorithm – to estimate mean of RV from sample
5. Using Robbins-Monro to update Q-factors- leads to 'model-free' algorithm
6. Finally, leads to Q-learning value iteration algorithm

Each step will be discussed next

Recollect: DP Bellman equations

Bellman equation for a "fixed" (deterministic) policy π

$$V(s) = \sum_{s'} p(s, \pi(s), s')[r + \lambda V(s')]$$

State-value function

- $r = r(s, \pi(s), s')$; r here is deterministic
- Note: if there is randomness in value of r we rewrite the value function as
  - $V(s) = \sum_{s',r} [p(s, \pi(s), s', r)[r + \lambda V(s')]]$

Bellman "optimality" equation

$$v^*(s) = max_{a \in A} \sum_{s'} [p(s, a, s') \cdot (r(s, a, s') + \lambda v^*(s'))]$$

- v^* is the optimal value function for the MDP. r here is deterministic
- Note: if there is randomness in value of r we rewrite
  - $v^*(s) = max_{a \in A} \sum_{s',r} [p(s', r \mid s, a) \cdot (r(s, a, s') + \lambda v^*(s'))]$
- Furthermore, it is the only function satisfying the property

Value function

$$v^*(s) = max_{a\in A} \sum_{s'} p(s,a,s')[r(s,a,s') + \lambda\, v^*(s')]$$

State-value function

Q-factor version of Bellman optimality

(Chapter 3, Sutton and Barto)

- $Q(s,a) = \sum_{s'} p(s,a,s')[r(s,a,s') + \lambda\, v^*(s')]$
- $v^*(s) = max_{a\in A}\ Q(s,a)$
- Combining the two:
- $Q(s,a) = \sum_{s'} p(s,a,s')\left[r(s,a,s') + \lambda \max_{a'} Q(s',a')\right]$

Action-value function

Q-factor version of value iteration (still in DP)

1. Initialize
   1. $Q(s,a) \in \mathbb{R}$, arbritrarily $\forall (s,a) \in (\mathcal{S}, A)$;
   2. Set $V(s) = max_{a\in A}\, Q(s,a), \forall$ s $\in S$
   3. *set tolerance* θ *(=1e-6)*

2. For each (s,a) pair compute Q-factor:

$$Q(s,a) = \sum_{s'} p(s,a,s')\left[r(s,a,s') + \lambda \max_{a'} Q(s',a')\right]$$

3. For each s compute value functin

$$v \leftarrow V(s)$$

$$V(s) = max_{a\in A}\ Q(s,a)$$

If $(|v - V(s)|) < \theta$ *goto step 4, else goto step 2*

4. Find corresponding optimal policy
   - $\pi(s) = argmax_a Q(s,a)$

Except for this additional step to calculate Q-values, everything else is the same as in value iteration

Robbins-Monro algorithm

UMassAmherst

- Suppose X is a random variable, with x_i is the i^{th} independent sample of X
- Then, $\mathbb{E}[X] = \lim_{k \to \infty} \frac{\sum_{i=1}^{k} x_i}{k}$
- Robbins-Monro algorithm utilizes above to allow for incremental updating **(Sutton and Barto, Section 2.4 and 2.5):**
  - Let $X_k = \frac{\sum_{i=1}^{k} x_i}{k}$ then $X_{k+1} = \frac{\sum_{i=1}^{k+1} x_i}{k+1} = \frac{\sum_{i=1}^{k} x_i + x_{k+1}}{k+1}$
  - $X_{k+1} = \frac{(kX_k + X_k - X_k + x_{k+1})}{k+1} = \frac{X_k(k+1) - X_k + x_{k+1}}{k+1}$
  - $X_{k+1} = \frac{X_k(k+1)}{k+1} - \frac{X_k}{k+1} + \frac{x_{k+1}}{k+1} = X_k - \frac{X_k}{k+1} + \frac{x_{k+1}}{k+1}$
  - $X_{k+1} = X_k + \frac{1}{k+1}[x_{k+1} - X_k] \sim X_k + \alpha[x_{k+1} - X_k]$
  - $NewEstimate \leftarrow OldEstimate + StepSize\ [Sample\ - OldEstimate]$; StepSize =$\alpha = \frac{1}{k+1}$

Robbins-Monro to Q-factor updating

R-M: $X_{k+1} = X_k + \alpha[x_{k+1} - X_k]$
$X_{k+1} = X_k + \alpha[sample - X_k]$

- $Q(s,a) = \sum_{s'} p(s,a,s') \left[r(s,a,s') + \lambda \max_{a'} Q(s',a')\right]$
- $\sim Q(s,a) = \mathbb{E}\left[r(s,a,s') + \lambda \max_{a'} Q(s',a')\right]$
- *Applying Robbins-Monro*
  - $Q(s,a)$ *is random variable, then* $\left[r(s,a,s') + \lambda \max_{a'} Q(s',a')\right]$ *is a sample of that random variable*
  - $Q(s,a) = Q(s,a) + \alpha[sample - Q(s,a)]$
  - $Q(s,a) = Q(s,a) + \alpha\left[r(s,a,s') + \lambda \max_{a'} Q(s',a') - Q(s,a)\right]$
  - $Q_{k+1}(S_t,A_t) = Q_k(S_t,A_t) + \alpha\left[R_{t+1} + \lambda \max_{a'} Q(S_{t+1},a') - Q_k(S_t,A_t)\right]$
  - $Q(S_t,A_t) = Q(S_t,A_t) + \alpha\left[R_{t+1} + \lambda \max_{a'} Q(S_{t+1},a') - Q(S_t,A_t)\right]$

Robbins-Monro to Q-factor updating

R-M: $X_{k+1} = X_k + \alpha[x_{k+1} - X_k]$
$X_{k+1} = X_k + \alpha[sample - X_k]$

- $Q(s,a) = \sum_{s'} p(s,a,s')\left[r(s,a,s') + \lambda \max_{a'} Q(s',a')\right]$
- $\sim Q(s,a) = \mathbb{E}\left[r(s,a,s') + \lambda \max_{a'} Q(s',a')\right] \sim \mathbb{E}[sample]$
- *Applying Robbins-Monro*
  - $Q(s,a)$ *is random variable, then* $\left[r(s,a,s') + \lambda \max_{a'} Q(s',a')\right]$ *is a sample of that random variable*
  - $Q(s,a) = Q(s,a) + \alpha[sample - Q(s,a)]$
  - $Q(s,a) = Q(s,a) + \alpha\left[r(s,a,s') + \lambda \max_{a'} Q(s',a') - Q(s,a)\right]$
  - $Q(S_t,A_t) = Q(S_t,A_t) + \alpha\left[R_{t+1} + \lambda \max_{a} Q(S_{t+1},a) - Q(S_t,A_t)\right]$

Notice there is no p (no TPM)=> model free!

- $NewEstimate \leftarrow OldEstimate + StepSize\ [Target\ (or\ sample) - OldEstimate]$; $\alpha = \frac{1}{k+1}$
  - Note: *sample* is more of a *target* here given the dynamics
  - $[Target - OldEstimate]$ is the error in the estimate, which can be reduced by taking a step toward the target.
  - If we set $\alpha = \frac{1}{k+1}$ it is direct averaging of samples, however, this can be altered
  - Generally, convergence with probability 1 if following conditions are satisfied,
    - $\sum_{k=1}^{\infty} \alpha_k = \infty\ and\ \sum_{k=1}^{\infty} \alpha_k^2 < \infty$
  - Or, specific to action selection, suppose $\alpha_n(a)$ is step-size after n^{th} selection of action a
    - $\sum_{n=1}^{\infty} \alpha_n(a) = \infty\ and\ \sum_{n=1}^{\infty} \alpha_n^2(a) < \infty$

Recollect these are the same as stochastic approximation convergence properties when we did gradient descent; In fact, 'incremental' stochastic gradient descent is sometimes attributed to R-M

UMassAmherst

The Commonwealth's Flagship Campus