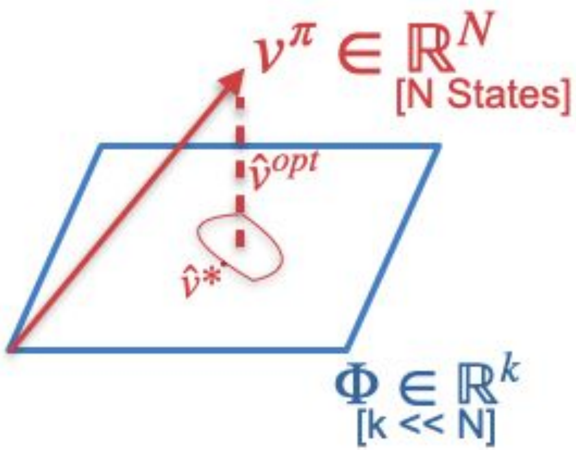
Reinforcement Learning: LSTD, LSTDQ, LSPI, Fitted Q

B. Ravindran

LSTD & LSTDQ



An overview of linear function approximations:

- We look for projection of real value function v^π to the space spanned by feature vectors Φ

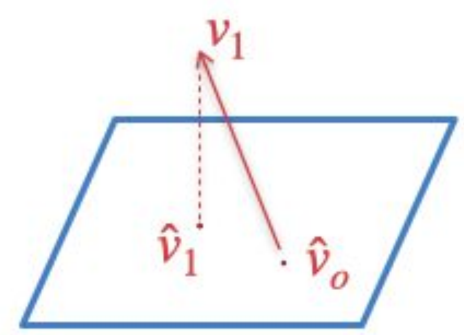
- Given the true v^π values, $\hat{v}^{opt}$ is the projection. Since, we do not know true v^π, we converge at $\hat{v}^*$ following the procedure.

"Why the gap?"

Procedure:

- Start with $\hat{v}_o$
- Apply Update $v_1 = \hat{T}_\pi \hat{v}_0$
- Project back $\hat{v}_1 = \mathbb{P} v_1$
- Repeat till convergence

Final Update (each iteration)



LSTD & LSTDQ

Setting up as a traditional 'Regression Problem':

Traditional Data Representation

Fixed point solution (Desired): $(\mathbb{P}T_\pi)\hat{v}^\pi = \hat{v}^\pi$

$$\hat{v}^\pi = \underbrace{\Phi(\Phi^T\Phi)^{-1}\Phi}_{\text{Projection}}\underbrace{(R^\pi + \gamma P^\pi \hat{v}^\pi)}_{\text{Bellman}}$$

$$\Phi\theta^\pi = \Phi(\Phi^T\Phi)^{-1}\Phi(R^\pi + \gamma P^\pi \Phi\theta^\pi)$$

Therefore, The Least Square solution:

$$\theta^\pi = (\Phi^T(\Phi - \gamma P^\pi\Phi))^{-1}\Phi^T R^\pi$$

Equivalent for Value Function Approximation

Similarly

, $\hat{Q}^\pi = \Phi(\Phi^T\Phi)^{-1}\Phi(R^\pi + \gamma P^\pi \hat{Q}^\pi)$

NOTE:

- We do NOT have the model

- This solution treats every sample as an equal contributor to the error.

- We wish to prioritise some states over others.

Weighted Version:

$$\underbrace{\Phi^T w(\Phi - \gamma P^\pi\Phi)}_{A}\theta^\pi = \underbrace{\Phi^T w R^\pi}_{b}$$

W - Weight function over all states
(eg: steady-state probability distribution)

LSTD & LSTDQ

Weighted Version:

$$\underbrace{\Phi^T w(\Phi - \gamma P^\pi \Phi)}_{A}\theta^\pi = \Phi^T \underbrace{w R^\pi}_{b}$$

We wish to estimate A & B matrices directly from samples rather than, from the model based P & R matrices.

$$A = \Phi^T w(\Phi - \gamma P^\pi \Phi)$$

$$A = \Sigma_s \phi(s) w(s)\ (\Phi(s) - \gamma \Sigma_{s'} P(s, \pi(s), s'))^T$$

$$A = \Sigma_s w(s)\, \Sigma_{s'} P(s, \pi(s), s')\ [\phi(s)\ (\phi(s) - \gamma \phi(s'))^T]$$

$$b = \Phi^T w R^\pi$$

$$b = \Sigma_s \Phi(s) w(s)\ \Sigma_{s'} P(s, \pi(s), s') R(s, \pi(s), s')$$

LSTD & LSTDQ

Given Samples of form:

$$D = \{(s_i, a_i, r_i, s_i^{'}) \,|\, i = 1,2,3,.....L\}$$

[i -> sample index;   L - no. of samples]

Estimate from samples:

$$\tilde{A} = \frac{1}{L}\Sigma_{i=1}^{L}[\phi(s_i)\ (\phi(s_i) - \gamma\phi(s_i^{'}))^T]$$

$$\tilde{B} = \frac{1}{L}\Sigma_{i=1}^{L}\phi(s_i)r_i$$

Action version:

$$\tilde{A} = \frac{1}{L}\Sigma_{i=1}^{L}[\phi(s_i)\ (\phi(s_i) - \gamma\phi(s_i^{'}, \pi(s_i^{'})))^T]$$

$$\tilde{B} = \frac{1}{L}\Sigma_{i=1}^{L}\phi(s_i, a_i)r_i$$

Now, we can solve for
and thereby the value function.

$$\tilde{A}\,.\,\theta^{\pi} = \tilde{b}$$

LSPI
Least Squares Policy Iteration

$$\pi_{t+1}(s) = argmax_a \; \hat{Q}^{\pi_t}(s,a)$$

$$\pi_{t+1} = argmax \; \Phi\hat{\theta}^{\pi_t}$$

- Solve for (some) 'j' no of states
- Generate training data for classifier (finite actions) or regressor (continuous actions)

$$\begin{array}{ll} a_1 & \phi_1(s_1), \phi_2(s_1), \ldots\ldots\ldots, \phi_k(s_1) \\ a_2 & \phi_1(s_2), \phi_2(s_2), \ldots\ldots\ldots, \phi_k(s_2) \\ \vdots & \vdots \\ a_j & \phi_1(s_j), \phi_2(s_j), \ldots\ldots\ldots, \phi_k(s_j) \end{array}$$

Training data

Issues:
Actions and States might be continuous and complex
Maximise each time to pick action?
Solve for all the states?

Sample for 'j' states

$$(s_1, s_2, \ldots\ldots\ldots s_j) \quad -> States$$

$$(a_1, a_2, \ldots\ldots\ldots a_j) \quad -> \pi_{t+1}(s)$$

Fitted-Q Iteration

- Start with few samples. Generate Targets (Q-Learning targets).
  Train(Fit) function approximator (Not necessarily linear) to predict the targets.
- Use function approximator to form policy to generate new samples.
- Repeat.