A Guided Tour of Chapter 11: Batch RL: Experience Replay, DQN, LSPI

Ashwin Rao

ICME, Stanford University

Incremental RL makes inefficient use of training data

- Incremental versus Batch RL in the context of fixed finite data
- Let's understand the difference for the simple case of MC Prediction
- Given fixed finite sequence of trace experiences yielding training data:

$$\mathcal{D} = [(S_i, G_i) | 1 \leq i \leq n]$$

- Incremental MC estimates $V(s; \boldsymbol{w})$ using $\nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w})$ for each data pair:

$$\mathcal{L}_{(S_i, G_i)}(\boldsymbol{w}) = \frac{1}{2} \cdot (V(S_i; \boldsymbol{w}) - G_i)^2$$

$$\nabla_{\boldsymbol{w}} \mathcal{L}_{(S_i, G_i)}(\boldsymbol{w}) = (V(S_i; \boldsymbol{w}) - G_i) \cdot \nabla_{\boldsymbol{w}} V(S_i; \boldsymbol{w})$$

$$\Delta \boldsymbol{w} = \alpha \cdot (G_i - V(S_i; \boldsymbol{w})) \cdot \nabla_{\boldsymbol{w}} V(S_i; \boldsymbol{w})$$

- n updates are performed in sequence for $i = 1, 2, \ldots, n$
- Uses update method of FunctionApprox for each data pair (S_i, G_i)
- Incremental RL makes inefficient use of available training data $\mathcal{D}$
- Essentially each data point is "discarded" after being used for update

Batch MC Prediction makes efficient use of training data

- Instead we'd like to estimate the Value Function $V(s; \boldsymbol{w}^*)$ such that

$$w^* = \arg\min_{\boldsymbol{w}} \frac{1}{n} \cdot \sum_{i=1}^{n} \frac{1}{2} \cdot (V(S_i; \boldsymbol{w}) - G_i)^2$$
$$= \arg\min_{\boldsymbol{w}} \mathbb{E}_{(S,G)\sim\mathcal{D}}[\frac{1}{2} \cdot (V(S; \boldsymbol{w}) - G)^2]$$

- This is the solve method of FunctionApprox on training data $\mathcal{D}$
- This approach to RL is known as *Batch RL*
- solve by doing updates with repeated use of available data pairs
- Each update using random data pair $(S, G) \sim \mathcal{D}$

$$\Delta \boldsymbol{w} = \alpha \cdot (G - V(S; \boldsymbol{w})) \cdot \nabla_{\boldsymbol{w}} V(S; \boldsymbol{w})$$

- This will ultimately converge to desired value function $V(s; \boldsymbol{w}^*)$
- Repeated use of available data known as *Experience Replay*
- This makes more efficient use of available training data $\mathcal{D}$

Batch TD Prediction makes efficient use of *Experience*

- In Batch TD Prediction, we have experience $\mathcal{D}$ available as:

$$\mathcal{D} = [(S_i, R_i, S'_i) | 1 \leq i \leq n]$$

- Where (R_i, S'_i) is the pair of reward and next state from a state S_i
- So, Experience $\mathcal{D}$ in the form of finite number of atomic experiences
- This is represented in code as an Iterable [TransitionStep [S]]
- Parameters updated with repeated use of these atomic experiences
- Each update using random data pair $(S, R, S') \sim \mathcal{D}$

$$\Delta \boldsymbol{w} = \alpha \cdot (R + \gamma \cdot V(S'; \boldsymbol{w}) - V(S; \boldsymbol{w})) \cdot \nabla_{\boldsymbol{w}} V(S; \boldsymbol{w})$$

- This is TD Prediction with Experience Replay on Finite Experience $\mathcal{D}$

Batch TD(λ) Prediction

- In Batch TD(λ) Prediction, given finite number of trace experiences

$$\mathcal{D} = [(S_{i,0}, R_{i,1}, S_{i,1}, R_{i,2}, S_{i,2}, \ldots, R_{i,T_i}, S_{i,T_i}) | 1 \leq i \leq n]$$

- Parameters updated with repeated use of these trace experiences
- Randomly pick trace experience (say indexed i) $\sim \mathcal{D}$
- For trace experience i, parameters updated at each time step t:

$$\boldsymbol{E}_t = \gamma\lambda \cdot \boldsymbol{E}_{t-1} + \nabla_{\boldsymbol{w}} V(S_{i,t}; \boldsymbol{w})$$

$$\Delta \boldsymbol{w} = \alpha \cdot (R_{i,t+1} + \gamma \cdot V(S_{i,t+1}; \boldsymbol{w}) - V(S_{i,t}; \boldsymbol{w})) \cdot \boldsymbol{E}_t$$

Key Takeaways from this Chapter

- Batch RL makes efficient use of data
- DQN uses experience replay together with a frozen DNN avoiding pitfalls of time-correlation and semi-gradient
- LSTD is a fast linear-algebra calculation of Batch TD Prediction
- LSPI is an off-policy, experience-replay Control Algorithm using LSTD for Policy Evaluation