MS&E338 Reinforcement Learning Lecture 18 - May 30, 2018

Deep Exploration with Value Generalization

Lecturer: Ben Van Roy *Scribe: Melody Guan, Bar Light, Xavier Warnes*

The goal of this lecture is to combine ideas from the first half of the course with ideas from the second part of the course. In particular, we want to apply deep exploration in a non-tabular reinforcement learning (RL) setting.

1 LSVI

We now recall the algorithm least-squares value iteration (LSVI) studied in the first part of the course.

Algorithm 1: LSVI
Buffer = {}
For $l = 1, 2, 3, ..$
  Fit Q_θ to buffered data (see below).
  Apply greedy policy and observe trajectory $(s_0, a_0, r_1, \ldots, s_\tau)$
  Append transitions $(s_t, a_t, r_{t+1}, s_{t+1})$ to buffer.

Recall that Q_θ can be a fancy neural network.

LSVI adapts the value iteration algorithm to an RL setting with generalization. When fitting Q_θ, we have a buffer of data $D = \{(s_t, a_t, r_t, s'_t)\}$ and we update Q_θ in a least square fashion: we initialize θ, and for $h = 0, ..., H-1$ we iterate

$$\theta \leftarrow \underset{\overline{\theta}}{\text{argmin}} \sum_{(s,a,r,s') \in D} (r + \max_{a' \in A} Q_\theta(s', a') - Q_{\overline{\theta}}(s, a))^2, \tag{1}$$

where H is the planning horizon. We then update Q_θ to be Q_{θ_H} where θ_H is the θ of the last iteration. Since LSVI needs to fit a Q-function for each episode, it can be computationally cumbersome and is not very practical for large-scale problems. This can be fixed with incremental alternatives, such as temporal differences (TD) methods. In a TD method, we sample only a mini-batch D' from the buffer D and update θ using a step size γ and a discount factor α:

$$\theta \leftarrow \theta - \gamma \nabla_{\overline{\theta}} \left(\sum_{(s,a,r,s') \in D'} (r + \alpha \max_{a' \in A} Q_\theta(s', a') - Q_{\overline{\theta}}(s, a))^2 \right). \tag{2}$$

The discount factor reflects how much we value future rewards and it sort of plays the role of the planning horizon H above. The problem with Algorithm 1 is inefficient exploration. In Algorithm 1, instead of greedy algorithm we can also use ϵ-greedy or Boltzmann exploration. We saw in the last few classes (recall the mouse example) that these algorithms do not explore well. This is the reason that we introduced UCRL and PSRL. We now review PSRL.

2 RLSVI

We now review the algorithm posterior sampling for reinforcement learning (PSRL).

Algorithm 2: PSRL

Start with a prior over MDPs (R, P).
For $l = 1, 2, 3, ..$
  Sample an MDP $(\hat{R}, \hat{P})$.
  Compute the optimal policy $\hat{\pi}$ for the MDP $(\hat{R}, \hat{P})$.
  Apply $\hat{\pi}$ over the episode $(s_0, a_0, r_1, \ldots, s_\tau)$.
  Compute the posterior distribution.

In the above algorithm, we need to solve for an optimal policy of a sampled MDP during each episode, which can be time consuming for large scale problems. One workaround is to learn Q functions instead of the underlying MDPs. Instead of having a prior on MDPs (R, P) we can equivalently have a prior over Q functions. Since every MDP (R, P) induces a Q function with the same optimal policy function (and every Q function induces a class of MDPs with the same optimal policy function), having a prior over Q functions is equivalent to having a prior over MDPs (R, P).

In the spirit of Thompson sampling (TS) algorithms, randomized least-squares value iteration (RLSVI) introduces uncertainty to LSVI by approximately sampling Q functions.

Algorithm 3: RLSVI

Buffer = {}
For $l = 1, 2, 3, ..$
  Approximately sample Q_θ from posterior.
  Apply greedy policy $(s_0, a_0, r_1, \ldots, s_\tau)$
  Append transitions $(s_t, a_t, r_{t+1}, s_{t+1})$ to buffer.

RLSVI does not explicitly maintain and update belief distributions as a standard Bayesian method would. Here is how RLSVI approximately samples Q_θ from a posterior. Assume that the prior over θ is $N(0, \sigma_0^2 I)$.

Initialize θ.

Sample $\widetilde{\theta}$ from the prior $N(0, \sigma_0^2 I)$.

For each $(s_t, a_t, r_{t+1}, s_{t+1})$ in the buffer D, we sample $\tilde{w}_t \sim N(0, \sigma^2)$ and let $\tilde{r}_{t+1} = r_{t+1} + \tilde{w}_t$. Let $\tilde{D}$ denote the perturbed buffer.

For $h = 0, 1, \ldots H-1$ we iterate

$$\theta \leftarrow \arg\min_{\bar{\theta}} \frac{1}{\sigma^2} \sum_{(s,a,\tilde{r},s') \in \tilde{D}} (\tilde{r} + max_{a\prime \in A} Q_\theta(s\prime, a\prime) - Q_{\bar{\theta}}(s,a))^2 + \frac{1}{\sigma_0^2} ||\bar{\theta} - \tilde{\theta}||_2^2 \tag{3}$$

Note that this is identical to LSVI execpt that we use a perturbed version of the data when we optimize. Why is this similar to Bayesian updating? the sampling of Q_θ is similar to the Kalman filter (see Section 2.1 for more details).

Computational results for RLSVI looks good. For example, RLSVI using a neural network with two hidden layers solves the cart-pole swing-up example discussed in class efficiently.

In the current setting, RLSVI is too slow, so we need an incremental version for RLSVI (as in Section 1). A key feature of the RLSVI is that it calls for an estimate trained on randomly perturbed data at the beginning of each epsiode $l = 1, 2, ...$ This feature causes auto-correlation between samples in an incremental version of RLSVI. A solution for this problem is ensemble sampling. Ensemble sampling learn $k = 1, ..., K$ models in parallel. We have k copies $(\widetilde{\theta}_1, ..., \widetilde{\theta}_K)$. At the beginning of each episode l we sample $\widetilde{\theta}_k \sim Unif\{1, ..., K\}$. The ensemble sampling approach is a powerful concept since it elimnates a lot of auto-correlation. There is more room for future research to improve the ensemble sampling approach.

2.1 Kalman filter

We now discuss why the perturbation technique used in RLSVI makes sense. We start with a prior $\theta \sim N(0, \sigma_0^2 I)$ and then update based on observed data. For simplicity we consider the scalar case. The observed data is given by $y_1 = \theta x_1 + w_1$,...,$y_N = \theta x_N + w_N$ where w_n is iid $N(0, \sigma^2)$. The posterior of θ is normal. Define

$$\theta^* = argmin_\theta \frac{1}{\sigma^2} \sum_{n=1}^{N} (y_n + \widehat{w}_n - \theta x_n)^2 + \frac{1}{\sigma_0^2} (\theta - \widehat{\theta})^2 \tag{4}$$

where $\widehat{\theta} \sim N(0, \sigma_0^2)$ and $\widehat{w}_n \sim N(0, \sigma^2)$. Then θ^* is a sample from the posterior. To see this note that since the posterior of θ is normal, it is enough to check that θ^* has the same expected value and the same variance as the posterior of θ. Using simple calculus, we can set the derivative to 0 in the last expression and obtain:

$$\theta^* = \frac{\frac{1}{\sigma^2} \sum_{n=1}^{N} x_n (y_n + \hat{w}_n) + \hat{\theta}/\sigma_0^2}{\frac{1}{\sigma^2} \sum_{n=1}^{N} x_n^2 + \frac{1}{\sigma_0^2}}$$

It is well known that the conditional expectation of θ is given by

$$\mathbb{E}(\theta | D) = \left(\frac{1}{\sigma^2} \sum x_n^2 + \frac{1}{\sigma_0^2} \right)^{-1} \left(\frac{1}{\sigma^2} \sum x_n y_n \right). \tag{5}$$

We conclude that

$$\mathbb{E}(\theta | D) = \mathbb{E}(\theta^*). \tag{6}$$

Similarly, we can show that $Var(\theta | D) = Var(\theta^*)$. We conclude that θ^* is a sample from the posterior of θ. The same analysis can be extended to the multi-dimensional case. This analysis suggegsts that for the linear setting, training a least-squares solution on perturbed versions of the data is equivalent to conjugate Bayesian posterior. This is the main idea behind the algorithm RLSVI where optimization happens over randomized versions of the underlying data and prior (Equation (3)).