Bayesian Reinforcement Learning Methods

Using Bayesian MDPs and GPTD Methods

Vickie Ye and Alexandr Wang

May 12, 2016

Markov Decision Processes

- System described by a known set of states S and actions A, and unknown reward function $R(s, a)$ and transition function $T(s, a, s') = P(X^{(t+1)} = s' | X^{(t)} = s, Y^{(t)} = a)$.
- We define a quality function

$$Q = \sum_{t=0}^{\infty} \gamma^t R^{(t)},$$

which we approximate for each state-action pair as

$$Q(s, a) = \mathbb{E}[R(s, a)] + \gamma \sum_{s'} T(s, a, s') \mathrm{max}_{a'} Q(s', a').$$

- To estimate Q, we need to estimate T and R.

Estimating T with a Bayesian model

We model our observed transition counts for each (s, a) as

$$\mathbf{m}^{(t)} \sim \mathrm{Mult}(\pi(s, a))$$
$$\pi(s, a) \sim \mathrm{Dirichlet}(\alpha)$$

where

$$\pi(s, a) = (T(s, a, s_0), ..., T(s, a, s_{N-1})),$$

Our posterior is then

$$\pi^{(t)}|D \sim \mathrm{Dirichlet}(\alpha^{(t)}|\mathbf{m}^{(t)}),\ \alpha_i^{(t)} = \alpha_i + m_i^{(t)}$$

Estimating R with a Bayesian model

We model our reward $R(s, a)$ for each state-action pair as

$$\begin{aligned} r(s,a) &\sim \mathcal{N}(\mu, \tau) \\ \mu &\sim \mathcal{N}(\mu_0, c_0\tau) \\ \tau &\sim \mathrm{Ga}(\beta, \rho) \end{aligned}$$

Our posterior is then

$$\tau \sim \mathrm{Ga}\left(\beta + \frac{k}{2}, \rho + \frac{1}{2}\sum_i (r_i - \bar{r})^2 + \frac{kc_0(\bar{r} - \mu_0)^2}{2(n + c_0)}\right),$$

$$\mu \sim \mathcal{N}\left(\frac{k\bar{r} + c_0\mu_0}{k + c_0}, (k + c_0)\tau\right)$$

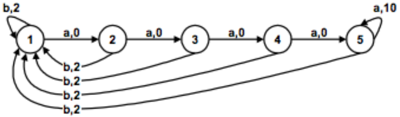


Figure 1: Chain Problem

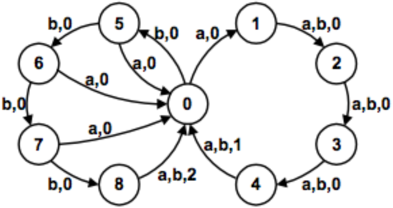


Figure 2: Loop Problem

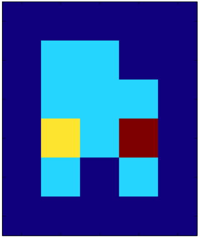


Figure 3: Easy maze

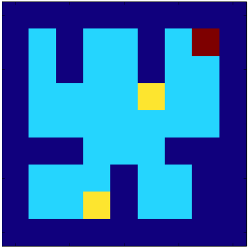
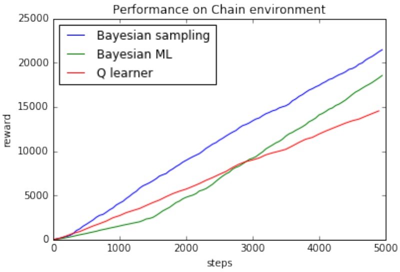
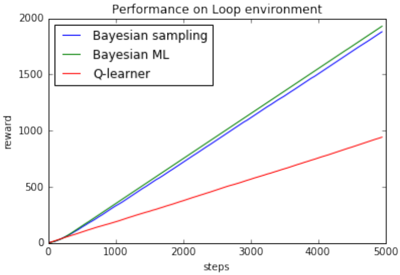


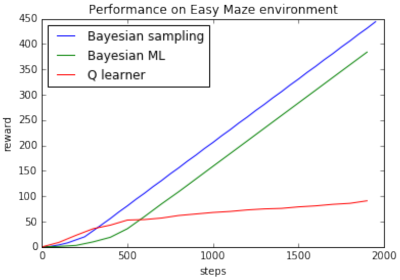
Figure 4: Hard maze

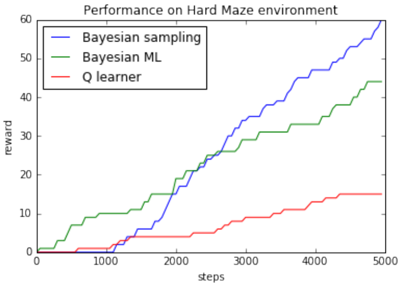
Results on the Chain and Loop problems





Results on the Maze problems





GPSARSA Framework

- In the GPSARSA framework, GPs are used to approximate the quality function Q.
- Implement a kernel $k(x, x')$ for state-action pairs x, x'. It should reflect a similarity notion for the problem at hand.
- Put a GP prior over $Q \sim \mathcal{N}(0, k, (\cdot, \cdot))$ where $\mathbb{E}[Q(x)] = 0$ and $\mathbb{E}[Q(x)Q(x')] = k(x, x')$
- Why use a GP? Get uncertainty estimates for free, and can make decisions in a continuous action space.
- SARSA refers to model where you use the state, action, reward, and the new state and action to update your policies.

More details on GPSARSA

We can formulate the reward model as

$$R(x^{(t)}, x^{(t+1)}) = Q(x^{(t)}) - \gamma Q(x^{(t+1)}) + N(x^{(t)}, x^{(t+1)})$$

where $N(x, x') = \Delta Q(x) - \gamma \Delta Q(x')$, $\Delta Q \sim \mathcal{N}(0, \Sigma)$, with $\Sigma(x, x') = \delta(x - x')\sigma^2(x)$.

We'll omit the results here for brevity, but we can easily formulate the posterior of the quality function Q given the past rewards as

$$Q(x) | R_{t-1} \sim \mathcal{N}\left(v_t(x), p_t(x)\right), \tag{1}$$

where $v_t(x), p_t(x)$ are easily computable given our data. See our paper for more details!

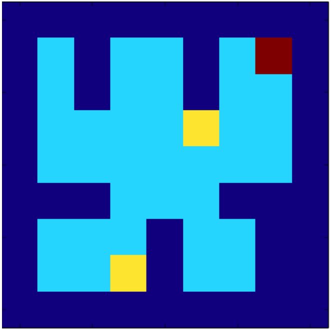
Sparsification

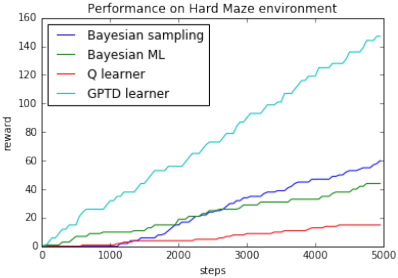
- Re-evaluating the GP for every time step can be very costly. Oftentimes, you'll train your learner for hundreds of thousands of steps, but as we've seen, GPs are $O(n^3)$ computation time.
- Use **sparsification** method to reduce the number of data points we need to store.
- Represent $k(\cdot, \cdot)$ as the inner product of the Hilbert space $\mathcal{H}$, i.e. $k(x, x') = \langle \phi(x), \phi(x') \rangle_{\mathcal{H}}$
- Maintain a dictionary of $\{\tilde{\phi}(x_i)\}$, with which we can approximate $\phi(x_{(i)})$ by linear projection:

$$\phi(x_i) = \sum_j a_{ij} \tilde{\phi}(x_j) + d$$

- Every x we encounter where d is greater than a precision threshold is then added to our dictionary.

Results on the hard Maze problem





Results on the very hard Maze problem