6.7950 Fall 2022: - Recitation 9 Handout

1 Concentration inequalities

When studying bandit problem during lecture, we stumbled upon certain concentration inequalities that helped us analyze the finite-sample performance of certain algorithms. For this part of the recitation we are going to study these inequalities a bit more.

1.1 Concept

For many problems we may have a set of random variables for which we only have partial information. In case of the multi-armed bandit problems we studied, the reward for each action was either 0 or 1 according to some Bernoulli distributions with unknown parameters. In a variation of the problem, these rewards could instead have come from a different distribution. In a recommender system example, the rewards could be scaled by the user's rating of a movie: if the user rated the movie badly then a lower reward could be applicable, in which case a more generic distribution on the range $[0, 1]$ could appear. In contrast, for many problems associated with physical systems we may not have good upper and lower bounds for certain variables, but we instead may have an estimate of their variance.

In that context, where different problems have different sets of random variables with only partial information about them, concentration inequalities are very helpful. They provide a mathematical tool to study each problem regardless of their underlying distributions, so long as they satisfy the conditions required by each inequality. In general, the choice of concentration bound is determined by what type of information is available and the more it is assumed the tighter these bounds are. Note that sometimes these bounds can be vacuous, e.g., for a choice of parameters they may state that some probability is not greater than 1, which is trivially true.

1.2 Markov's inequality

One of the simplest and most fundamental of such inequality is attributed to Markov is as follows. For a random variable X and a real number a satisfying the conditions

- $X \geq 0$
- $a > 0$

Then the following inequality holds

$$P(X \geq a) \leq \frac{\mathbb{E}(X)}{a}$$

Question: Prove the Markov inequality for a continuous distribution X with pdf f_X. *Hint:* start with $\mathbb{E}(X)$

Solution:

$$\begin{aligned}
\mathbb{E}(X) &= \int_0^\infty x f_X(x)dx \\
&= \int_0^a x f_X(x)dx + \int_a^\infty x f_X(x)dx \\
&\geq \int_a^\infty x f_X(x)dx \\
&\geq \int_a^\infty a f_X(x)dx \\
&= aP(X \geq a)
\end{aligned}$$

Thus, rearranging this inequality we get Markov's inequality

1.3 Equivalent expressions

You may notice that depending on the context some bounds are written equivalently in different manners. For example, Markov's inequality can be rewritten in terms of some quantity $\delta = \frac{\mathbb{E}(X)}{a}$ as

$$\begin{gathered}
P(X \geq a) \leq \frac{\mathbb{E}(X)}{a} = \delta \\
P\left(X \geq \frac{\mathbb{E}(X)}{\delta}\right) \leq \delta \\
P\left(\mathbb{E}(X) \leq X\delta\right) \leq \delta \\
P\left(\mathbb{E}(X) > X\delta\right) \geq 1-\delta
\end{gathered}$$

You might also see differences between a sum of variables and a mean of variables. For instance, in lecture we have seen the following form of Hoeffding's inequality

$$P\left(\left|\frac{1}{n}\sum_{t=1}^n X_t - \mu\right| > (b-a)\sqrt{\frac{\log\frac{2}{\delta}}{2n}}\right) \leq \delta$$

for n r.v.s $X_t \in [a,b]$ with $\mathbb{E}[X_t] = \mu$. But we can instead look at the sum $S_n = \sum_{i=t}^n X_t$ as variables and $E_n = \mathbb{E}[S_n]$ as mean of the sum to rewrite it as

$$P\left(|S_n - E_n| > (b-a)\sqrt{\frac{n\log\frac{2}{\delta}}{2}}\right) \leq \delta$$

Question: For a scalar t, find $F(t)$ by rewriting Hoeffding's inequality in the form

$$P\left(|S_n - E_n| \leq t\right) \leq F(t)$$

Solution: We start by finding the expression for δ as a function of t considering that t is given by

$$t = (b-a)\sqrt{\frac{n \log \frac{2}{\delta}}{2}}$$
$$\therefore \frac{2t^2}{n(b-a)^2} = \log \frac{2}{\delta}$$
$$\therefore \delta = 2\exp\left(-\frac{2t^2}{n(b-a)^2}\right)$$

which allows us to rewrite the inequality as

$$P\left(|S_n - E_n| > t\right) \leq 2\exp\left(-\frac{2t^2}{n(b-a)^2}\right)$$

Finally, instead of a bound on how much S_n is far from E_n, we can equivalently consider the chance that it's near E_n as

$$P\left(|S_n - E_n| \leq t\right) \geq 1 - 2\exp\left(-\frac{2t^2}{n(b-a)^2}\right)$$

Note that one could also use the name ”tail bound” instead of ”concentration bound”, but as we've just seen, they are complementary.

2 Study problems

Here's a few problems to help you review some potentially tricky parts of what we have seen so far. They are not intended to mimic what you will find in the exam, but rather to probe your understanding of some concepts

2.1 Definitions and high-level understanding

1. What is the the relationship between a Markov Process (MP), a Markov Reward Process (MRP) and a Markov Decision Process (MDP)
2. In what type of problems is dynamic programming more effective in general? What type of analysis tool often emerges in these settings?
3. If you were to rename reinforcement learning to explicitly characterize it as an extension of dynamic programming, how would you do that?

4. At some point during lecture, we ignored stochastic policies, but later on started using them again. Why did both of these transitions happen?
5. Why was the DP theory developed using state value functions V, but when discussing RL itself we started focusing on state-action value functions Q?

Solution:

1. A Markov Process is a stochastic process satisfying the Markov property where the states evolve through time according to a function that does not depend on the history of the states given the current state. A MRP is an extension of Markov processes that includes a reward signal, while a MDP is a MRP that also includes the selection of actions
2. In general, DP is most effective in optimization problems whose optimal solution depends on the solution of smaller subproblems, typically at previous time steps. This structure is a natural setting for the application of finite induction analysis, as it similarly boils down to propagating the validity of some property from some previous time to the current
3. A possible name would be ”sampled approximate dynamic programming”, since the absence of a model suggest the use of sampling, while the presence of large state-spaces often requires some sort of approximation in practical problems.
4. When studying DP, we could greatly simplify the problems we were studying (well-conditioned infinite horizon MDPs) by reducing the class of policies considered to be stationary and deterministic, as there was no loss in generality and expressive power in doing so. However, when studying policy space methods in RL, we found that deterministic policies are hard to be smoothly update during optimization (which can create instabilities), so switching back to stochastic policies gave us a tool to achieve more controlled changes to policies.
5. Using state-action value functions Q in RL was more natural because it's straightforward to obtain greedy actions based Q when one does not have a transition model: it's only necessary to find $a^* = \arg\max_a Q(s, a)$. In contrast, when using V the greedy action is given by $a^* = \arg\max_a \mathbb{E}_{s'}[r(s, a) + \gamma V(s')]$, which requires the ability to compute an expectation over the next state s', something that is not a problem in the DP setting and instead can be advantageous since there are many more state-action pairs than states.

2.2 General questions on DP

1. Identify the issue(s) and improve the technical accuracy of the following sentence about DP: ”Value iteration converges to the optimal value function, while policy iteration converges to the optimal policy”
2. For a MDP with action space A, state space S, compute the number of possible deterministic policies when
   - The horizon is infinite with a discount factor $\gamma < 1$
   - The horizon is finite and ends at T

3. How many possible values for V^π one could potentially find during policy iteration and how many possible values for V can one potentially find during value iteration?
4. Characterize the transition matrix that appears in one MDP when taking policy π in terms of the transition probabilities of the underlying Markov process.
5. In policy iteration for DP, we can evaluate a policy in different ways. Explain and compare two ways that do not involve sampling
6. What are some of the key properties of the Bellman operator?

Solution:

1. The optimal policy is not unique, so it would be more precise to say that policy iteration converges to **one** optimal policy.
2. In the infinite horizon case, a policy must take one of $|A|$ possible actions for each of the $|S|$ states, so there are $|A|^{|S|}$ possible policies. In the finite horizon case, the same logic applies, but for T sets of states, resulting in $\left(|A|^{|S|}\right)^T = |A|^{T|S|}$ possible policies.
3. Since each V^π is associated to one policy π, then the number of possible functions V^π is equal to the number of possible policies. For value iteration, however, we can provide any appropriately sized scalar function V as an initial guess to value iteration
4. We can write the transition matrix for a given action a by describing each of its elements P^a_{ij} as

$$P^a_{ij} = P(x_i|a, x_j)$$

   where each $P(x_i|a, x_j)$ is the probability of state moving from state x_j to x_i given that actions a was taken. We can then consider the transition matrix resulting from following policy π as

$$P^\pi_{ij} = \sum_a \pi(a|x_j)P^a_{ij} = \sum_a \pi(a|x_j)P(x_i|a, x_j)$$

5. One can directly compute V^π by solving a linear system of equations involving the transition probability matrix: $V^\pi = (I - \gamma P^\pi)^{-1} r^\pi$ or one could iteratively apply the Bellman operator $\mathcal{T}^\pi$. Thought the direct method reaches the correct answer in finite time, it's more computationally demanding as it needs to invert a large matrix and it might be more efficient to just apply a few iterations of the iterative method to get a sufficiently good approximation of V^π.
6. The Bellman operators T (both the optimal one $\mathcal{T}$ and the one associated with policy π, that is, $\mathcal{T}^\pi$) satisfy the following properties
   - T is γ-contractive in the L_∞ norm
   - T is element-wise monotonte: if $W_1 \leq W_2$ elemntwise, then $TW_1 \leq TW_2$ holds elementwise
   - For some scalar cm shifting the argument of T by c elementwise also shifts the output elementwise by γc: $T(W + cI_N) = TW + \gamma c I_N$

In the special of $\mathcal{T}^\pi$ (and not the optimal one), then $\mathcal{T}^\pi$ is also γ contractive in the d_π - weighted L_2 norm for the stationary distribution d_π associated with π if such stationary distribution exists and has only positive elements.

2.3 General question on RL

1. Intuitively, why do we usually take learning rates satisfying the Robbins Monro conditions
2. Discuss the following sentence: ”linear function approximations in RL are very limited in what they can approximate, so they are often not used in practice”
3. Why is the classic TD(0) method considered a pseudo-gradient method?
4. What problem are both the asynchronous part of A3C and the experience replay buffer are trying to solve?
5. Why was the performance difference lemma used? What information does it tells us?
6. Provide an everyday life situation where using an upper confidence bound -like approach might be reasonable for an individual
7. What was the motivation for using actor-critic methods instead of simple policy gradients?
8. During lectures, we sampled both actions and states because we don’t have model for the MDP. In the same circumstances, how can we partially avoid sampling? (Note: this might not be very obvious from what we focused in lecture)

Solution:

1. We want learning rates that are large enough that they will move the object we want to optimize to one optimal set of parameters, but that will also eventually converge to a value and discard the noisiness of the updates
2. For starters, linear functions can become as expressive as full tabular methods (as you’ve seen during HW4), so limited representation power is not an issue. This full expressive power may not be feasible when the state space is large, thus approximations are necessary. Despite that, in many situations linear function approximations are well suited, provided that there good enough features, so they can be said to commonly used in practice. The situations where they may be more lacking are when we don’t have a good set of features, thus often requiring the uses of learned features, as can be seen in the case of neural networks, for example.
3. In the classic TD(0) method, the update is not actually the gradient of any quantity, though it can be perceived as resulting from the gradient of the temporal error if you ignore that some of the terms will change when updating the approximation V_θ. Namely, in the error signal $\epsilon = V_\theta(s_t) - r(s_t, a_t) - \gamma V_\theta(s_{t+1})$, only the first terms $V_\theta(s_t)$ is considered to be changed by modifying θ.

4. Both methods (among other things) try to mitigate the fact that when sampling a trajectory in a MDP, the sequence of states and actions are not independent, but are actually highly correlated since they come from a sequential process. This is an issue for much of the theory of stochastic gradient descent and other learning methods, which typically perform better when the samples are not correlated.

5. The performance difference lemma allows us to compute the benefit (in terms of expected value function) of following a policy π', without having to actually sample from that policy: we only needs to know or estimate the advantage A^π of following π

6. One can, for example, consider the problem of selecting a restaurant visit in social occasions. On one hand, it's desirable to go to the best place (according to criteria that may involve price, food quality, distance) every time, but in order to acquire more information about a specific location it's necessary to visit it. This follows a similar dilemma as the exploration-exploitation one seen for the multi-armed bandit and analogously it makes sense to focus on selecting restaurants that have a high change of being of high quality.

7. The actor-critic update provided a way to mitigate the high variance suffered by a simple policy gradient method like REINFORCE

8. Though we don't have the transition model for the states, we quite often have the probabilities of selecting each action (since choosing the policies is part of the optimization problem). So one could use that information to reduce the sampling noise on actions and that's exactly what happens in certain methods, like the expected SARSA. Compare both of their temporal differences

$$\text{SARSA - } \delta_t = r_t(s_t, a_t) + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)$$
$$\text{Expected SARSA - } \delta_t = r_t(s_t, a_t) + \mathbb{E}_{a \sim \pi(\cdot|s_{t+1})}\left[\gamma Q(s_{t+1}, a)\right] - Q(s_t, a_t)$$

2.4 Nailing the difference between Q-learning and SARSA

1. The name SARSA comes from certain quantities that needed to be known before an update is taken. What are these quantities?
2. How would you name Q-learning using the same logic used to name SARSA?
3. How do you manage exploration in SARSA in comparison to Q-learning and how does that affect how each method converges?
4. Can you in principle have a completely random exploration policy in Q-learning while still converging to the optimal value? What are other practical problems associated with such choice?

Solution:

1. They are named for the the state, action and reward sampled at time t (that is, s_t, a_t and r_t), the resulting state s_{t+1} and the next action a_{t+1} that the policy will take.

2. Since the Q-learning update does not require the next action a_{t+1} (rather, it uses a greedy action w.r.t. Q for the update equation, which may be different from a_{t+1}, the next action taken by the exploration/behavior policy), one could call it SARS, though that name does not have positive connotation.

3. In SARSA, exploration is part of the single policy that is both being optimized and used to generate samples, while for Q-learning it comes from a behavior policy (which is decoupled from the greedy policy generated from Q). Thus, Q-learning will converge to the optimal Q^* as long as the behavior policy is exploratory enough and the learning rates are chosen appropriately. In comparison, SARSA needs its policy to be both optimal and exploratory (which may be conflicting goals), so convergence to the optimal only happens if the policy becomes greedy with respect to Q in the limit of $t \to \infty$. For example, if one parameterizes SARSA with an ϵ-greedy policy, it would be necessary to have $\epsilon \to 0$.

4. Yes, the behavior policy in Q-learning could be uniformly random, as long as all states are visited infinitely often, which one would expect given that no sequence of actions has zero probability. However, the resulting sampling of state-action pairs could be extremely inefficient in some problems as the exploration is not guided by the information that is gathered and certain state-action pairs might have extremely low probability of being sampled, even if they may provide high rewards.