EECS 498/598 Deep Learning - Homework 4

March 26th, 2019

Instructions

- This homework is Due April 16th at 11.59pm. Late submission policies apply.
- You will submit a write-up and your code for this homework.

1 [25 points] Deep Q-Network (DQN).

In this question, you will implement DQN algorithm (the detailed algorithm is in the Nature paper `https://web.stanford.edu/class/psych209/Readings/MnihEtAlHassibis15NatureControlDeepRL.pdf`) in `dqn.py` and train an agent to play the CartPole task.

1. Fill the blank in function `select_action` to implement ϵ-greedy action selection method.
2. Fill the blank in function `optimize_model` to train the deep Q network.
3. Fill the blank in class `DQN`.
4. Run the script to train the agent playing CartPole task and report the learning curve of the episode duration. You can change the hyper-parameters and network architecture to make the agent perform well.

2 [25 points] Policy Gradients.

1. We will derive the REINFORCE algorithm in this question. Consider the agent's objective,

$$J(\theta) = \mathbb{E}_{\tau \sim p(\tau;\theta)}[r(\tau)]$$

Show that the gradient of the objective function above can be approximated as following.

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t^i | s_t^i) r(\tau^i)$$

where the outer sum runs over different agent episodes $\tau^1, ..., \tau^N$ and $\tau^i = ((a_1^i, s_1^i), ..., (a_T^i, s_T^i))$ represents a single episode.

2. Let $r(\tau) = \sum_{i=1}^{T} r_t$. In this case, the above gradient estimator becomes

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t^i | s_t^i) \sum_{t'=1}^{T} r_{t'}^i$$

Show that the following is an unbiased estimate of $\nabla_\theta J(\theta)$.

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta (a_t^i | s_t^i) \sum_{t'=t}^{T} r_{t'}^i$$

That is, we can omit the rewards collected in the past while keeping the estimator unbiased. The new estimator has the advantage of having lower variance than the original estimator.

3 [25 points] REINFORCE algorithm.

In this problem, you will implement the REINFORCE algorithm and train an agent to play the CartPole task.

1. Policy network:
   - Fill in the `PolicyNet` class whose forward pass returns a probability of taking one out of two actions. We will implement this by a simple sigmoid where 1 means left action and 0 means right action after sampling.

     Policy: `Linear` (4 - 24) → `ReLU` → `Linear` (24 - 36) → `ReLU` → `Linear` (36 - 1) → `sigmoid`

2. REINFORCE algorithm:
   - Fill in the code inside the `simulate` function to sample actions given the `policy_network`.
   - Fill in the code to compute the discounted rewards used by the policy gradients update and store them in `reward_pool`.
   - Fill in the code for the policy gradients update using the elements in `reward_pool`.
3. Run the script and train the agent to play the CartPole task. Report the learning curve after all training episodes are finished. You should expect the agent to reach a reward of 200.

4 [25 points] Actor-Critic algorithm.

You will be implementing an Actor-Critic algorithm for the Cart-pole environment in this problem.

1. Subclass the `nn.Module` to implement the `Policy` class whose forward pass returns the action distribution and state value for a given input state.

   The actor and critic networks have the following network architectures.

   Actor: `Linear` (4 - 128) → `ReLU` → `Linear` (128 - 2) → `softmax`
   Critic: `Linear` (4 - 128) → `ReLU` → `Linear` (128 - 1)

   where the first linear layer is shared between the two networks.

2. Recall policy gradients,

$$J(\theta) = \mathbb{E}_{\tau \sim p(\tau;\theta)}[r(\tau)]$$

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} \nabla_\theta \log\pi_\theta(a_t^i|s_t^i) A^\pi(s_t^i, a_t^i)$$

where N, T represent the number of agent trajectories and episode length, respectively.

In Actor-Critic, the advantage is estimated using a critic network as

$$A^\pi(s_t^i, a_t^i) = \left(\sum_{t'=t}^{T} \gamma^{t'-t} r(s_{t'}^i, a_{t'}^i)\right) - V_\phi^\pi(s_t^i)$$

Given a single agent episode $\tau = ((s_1, a_1, r_1), ..., (s_T, a_T, r_T))$ as input, implement the `compute_losses` function which computes the `actor_loss` and `critic_loss` as follows.

- `actor_loss`: loss function $\mathcal{L}_\theta$ for the actor network

$$\mathcal{L}_\theta = -\sum_{t=1}^{T} \log\pi_\theta(a_t|s_t) A^\pi(s_t, a_t)$$

- `critic_loss`: loss function $\mathcal{L}_\phi$ for the critic network

$$\mathcal{L}_\phi = \sum_{t=1}^{T} A^\pi(s_t, a_t)^2$$

3. Train the model using the provided optimization code.

4. Report the curve showing average reward as the training progresses.

5 [10 points] Extra credit.

In question 2, show that adding a state dependent baseline does not introduce any bias in the estimator. i.e., Show that the following is an unbiased estimator of the gradient. Adding a baseline can further reduce the variance of the estimator.

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} \nabla_\theta \log\pi_\theta(a_t^i|s_t^i) \left(\left[\sum_{t'=t}^{T} r_{t'}^i\right] - b(s_t^i)\right)$$