ECE433/COS435 Introduction to RL
Assignment 0: Review of Topics
Spring 2024

Fill me in

Your name here.

Due February 4, 2024

Collaborators

Fill me in

Please fill in the name and netids of your collaborators in this section.

Instructions

You should work alone for this this assignment. Writeups should be typeset in Latex and submitted as PDF. You can work with whatever tool you like for the code, but **please submit the asked-for snippet and answer in the solutions box as part of your writeup. We will only be grading your writeup.** Make sure to still also attach your notebook/code with your submission.

Introduction

This Homework is meant to be a review of standard topics on machine learning, linear algebra, and probability, with a heavy emphasis on topics that will reappear in later stages of this course.

You will also work with PyTorch and Python to build your models, so we want to make sure you are familiar with the specific packages (e.g. `torch.distributions`) that you will be using throughout the course.

Question 1. Probability

In a reinforcement learning setting, we are often interested in settings where an agent interacts with a simple game. We often want to learn the agent's policy π, or its "rule" for making actions. In this problem, you will derive properties of different "rules" over a simple game, more interesting parts of the course.

Suppose we are playing a really simple game with 10 unique boxes, each with their own distinct label from $1, ..., 10$. One of these boxes contains a golden coin that awards the player with a reward of 5, one contains a silver coin that awards the player with a reward of 1, and the rest contain nothing and award the player with a reward of 0 if selected. The player must open a single box, and based on the outcome, their final score is the reward corresponding to the box that they selected. More formally, the player has a policy π and uses it to select an action $a \sim \pi$ corresponding to the box the player picked (e.g. $a = 1$ means the player picked box 1). The player is then given a reward $r(a)$ based on their action.

Question 1.a

Suppose our **agent's policy** is to randomly select one of the labelled boxes, each with equal probability. In other words, $\pi = \mathcal{U}\{1, 10\}$, and the agent selects a box $a \sim \pi$. Conditioned on the fact the the gold coin is in the box labelled 5 and the silver coin is in box labeled 1, what is the expected reward?

Solution

You solution here...

Question 1.b

Given the same assumptions as (1.a), what is the policy of the agent that maximizes the expected reward?

Solution

You solution here...

Question 1.c

What is the entropy of the policy in (1.a) and the entropy of the policy in (1.b)?

Solution

Your solution...

Question 1.d Coding

Suppose instead of boxes, we now deal with a *continuous* set of possible decisions. In this scenario, our agent can choose any number $a \in \mathbb{R}$, and the reward given to the player is defined by

$$\mathcal{R}(x) = \max\Big\{0, 1 - |x - 1|\Big\}$$

In this problem, you will use this reward function to compute the expected return of a policy (to be defined). The key idea is that an expectation can be approximated through sampling (recall Monte Carlo sampling from COS324). In this problem, you will familiarize yourself with the `torch.distributions` library.

See the colab notebook (1.d) for more details. Fill out the indicated code segments below after solving the problem.

Solution

Expected reward is ... **FILL ME**

Fill in your code here:

```
# Your code here...
r = reward(...)
print("Expected reward:", r.mean())
```

Question 1.e Coding

Plot the reward distribution under this policy using `matplotlib.pyplot.hist()` or some other histogram plotting library. Make sure to label your axes for this problem (x-axis is reward, y-axis is probability density). See colab notebook (1.e) for more details.

Hint. *When using* ***`matplotlib.pyplot.hist()`****, remember it needs to be a PDF! (check the documentation.)*

Solution

```
# Your code here...
...
plt.plot()
```

Question 2. MLE

The purpose of this question is to review Maximum Likelihood Estimation (MLE), which appears in many machine learning settings. In previous courses, students may have seen or solved a few questions on computing the Maximum Likelihood Estimator for a set of datapoints, in which case this problem will be review.

A random variable X has a Normal distribution with parameters μ, σ^2 if its probability distribution is uniquely defined as

$$f_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

Question 2.a

Let $\theta = \sigma^2$. For a set of datapoints $X_1, ..., X_n$ sampled i.i.d. from a Normal distribution with unknown parameters μ, θ find the log-likelihood function $L(\mu, \theta; X_1, ..., X_n)$. You answer should be in terms of n, $X_1, ..., X_n$, μ, and θ.

Solution

$$\log L(\mu, \theta; X_1, ..., X_n) = ...$$

Question 2.b

Using your answer in (3.a), solve for the MLE of μ and θ. Solutions should also show that the critical points are maximums e.g. compute derivatives around the critical point and number of critical points argument.

Solution

$$\hat{\mu} = ...$$
$$\hat{\theta} = ...$$

Question 3. Coding

The objective of this question is to familiarize you with the basic mechanics of PyTorch and Python. See **Question 3** on the .ipynb notebook, but you will be coding up gradient descent and building a basic neural network model.

Question 3.a

For this problem and (3.b), see the provided .ipynb notebook for all the questions, and fill out the indicated code segments below after solving each problem.

Solution

```
# Your code here...
def gradient_descent(X, y, weights: np.ndarray, eta: float,
    iterations: int) -> None:
  for i in range(iterations):
    # YOUR CODE HERE!
```

Question 3.b

Rewrite your solution to (a) using the torch library. Your solution must use `loss.backward()` and `weight.grad`.

Solution

```
# Your code here...
def gradient_descent_torch(X, y, weights: torch.tensor,
eta: float, iterations: int) -> None:
  for i in range(iterations):
    # YOUR CODE HERE!
```