6.4210/6.4212
Robotic Manipulation

Midterm 2 Practice

Additional practice on material from Problem Sets 5–7.

6.4210/2
Midterm 2 Practice

1. (15 points) **Control**

   (a) Consider a single joint under torque control connected to a link under the influence of gravity, with equations of motion given by

   $$I\ddot{q} + b\dot{q} + \tau_g(q) = \tau,$$

   where $I \in \mathbb{R}$ is the rotational inertia, $b \in \mathbb{R}$ is viscous joint friction, $\tau_g(q) \in \mathbb{R}$ is the gravitational torque experienced by the joint, and $\tau \in \mathbb{R}$ is the input torque. Consider using a P-controller with gravity compensation:

   $$\tau = K_p(q_d - q) + \tau_g(q).$$

   For setpoint control, given a step input in $q_d = 1$ and assuming $I > 0$, $K_p > 0$, $0 < b < 2\sqrt{IK_p}$, exact gravity compensation, and initial rest at $q = 0$, describe qualitatively (i) any overshoot or oscillations and (ii) the steady-state error.

   Overshoot and oscillations

   Steady-state error

   (b) How would adding a derivative term to this controller help? Describe the impact on (i) any overshoot or oscillations and (ii) the steady-state error.

   Overshoot and oscillations

   Steady-state error

   (c) Our robot is holding a door handle and we want the robot to open the door, but we only have a noisy estimate of the location of the hinge axis. How should we control the motion of the robot? Consider the following options:

   1. Position control
   2. Hybrid force/position control
   3. Stiffness control in joint coordinates
   4. Stiffness control in end-effector coordinates

   Pick one method and briefly justify your answer and also describe how you would pick the parameters (paths, stiffnesses, etc.). Your answer does not need to include mathematical detail.

   Choice of controller

Reason for the choice

Path and parameters

2. (15 points) **Picking up an unexpected payload**

A vertical actuator lifts a moving mass m. With z positive upward, its dynamics and controller are

$$m\ddot{z} = u - mg, \qquad u = \hat{m}g + k_p(z_d - z) - k_d\dot{z},$$

where $m, \hat{m}, g, k_p, k_d > 0$ and z_d is constant. The mass used for gravity compensation is $\hat{m}$; the actual moving mass is m. The actuator command u is an upward force. Assume no contact with the environment in parts (a) and (b), and ignore actuator force limits.

(a) (5 points) With $e = z_d - z$, derive the closed-loop differential equation for e. Compare $m = \hat{m}$ with $m > \hat{m}$: when is the equation homogeneous, and when do we expect a nonzero steady-state error? For $m > \hat{m}$, does the actuator settle above or below its target? Explain qualitatively; an expression for the steady-state error is not required.

Error dynamics

When is the equation homogeneous?

When is the steady-state error nonzero?

Settling position for the heavier payload

(b) (5 points) Initially $m = \hat{m}$. The actuator picks up a known additional mass $\Delta m > 0$. Give a change to the feedforward term that restores zero steady-state error without integral feedback. Would increasing k_d alone remove the offset? Explain.

Feedforward change

Effect of increasing k_d alone

(c) (5 points) A colleague instead adds integral feedback,

$$u_I(t) = k_i \int_0^t (z_d - z(s))\, ds, \qquad k_i > 0.$$

A rigid overhead obstruction holds the actuator at a fixed $z_b < z_d$. While the actuator remains blocked, does the integral command grow, decrease, or remain constant? What happens to the contact force, and why can this behavior be undesirable? A qualitative explanation is sufficient.

Integral command

Contact force

Why the behavior is undesirable

3. (15 points) **Imitation error**

A robot moves along a line according to $x_{n+1} = x_n + u_n$. The expert chooses $u = \pi_E(x) = -x/2$. The training demonstrations densely cover $D = [-3/8, 3/8]$ and contain no states outside it. At every state, the learner's action is $1/4$ larger than the expert's, so $\pi_L(x) = -x/2 + 1/4$. Both rollouts start at $x_0 = 0$.

(a) (5 points) What are x_1, x_2, x_3 for the expert? What are they for the learner? At which step does the learner first leave D?

Expert states (x_1, x_2, x_3)

Learner states (x_1, x_2, x_3)

First step outside D

(b) (5 points) What state does the learner approach? Does it ever return to D?

Limiting state

Return to D

(c) (5 points) Describe one DAgger iteration that supplies useful data at $x = 7/16$. What label is recorded there? Why does simply training longer on the original demonstrations not supply that information?

One DAgger iteration

Expert label at $x = 7/16$

Training longer on the original data

4. (15 points) **Policy gradient warm-up**

A robot makes one scalar command a and receives reward $r(a) = -(a-1)^2$. Its policy samples $a \sim \mathcal{N}(\mu, 1)$; only the mean μ is learned. The policy density is

$$\pi_\mu(a) = \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{(a-\mu)^2}{2}\right].$$

We maximize expected reward with REINFORCE, without a baseline. For a batch of N independent trials collected with the current policy, the update is

$$\mu_{\text{new}} = \mu + \alpha \widehat{g}, \qquad \widehat{g} = \frac{1}{N} \sum_{i=1}^{N} r(a_i) \frac{\partial}{\partial \mu} \log \pi_\mu(a_i).$$

The sampled commands and rewards are held fixed when computing this derivative.

(a) (4 points) Derive $\partial \log \pi_\mu(a)/\partial \mu$.

(b) (5 points) At $\mu = 0$, two recorded trials have commands $a_1 = -1$ and $a_2 = 2$. Compute their rewards, the batch gradient estimate $\widehat{g}$, and the new mean for $\alpha = 1/4$.

Reward for $a_1 = -1$

Reward for $a_2 = 2$

Batch gradient estimate

Updated mean

(c) (6 points) Find the exact expected reward $J(\mu) = \mathbb{E}[r(a)]$ and its derivative at $\mu = 0$. You may use $\mathbb{E}[a] = \mu$ and $\mathbb{E}[a^2] = \mu^2 + 1$. Does the gradient estimate in (b) equal this derivative? Does any difference establish that the implementation is incorrect? Explain the distinction between a sampled gradient and its expectation.

Expected reward

Derivative at $\mu = 0$

Comparison with the batch estimate

Does the difference establish an implementation error?

5. (15 points) **Q-learning, policy evaluation, and replay**

A discounted task has states S, T and a terminal state G. The discount is $\gamma = 1/2$. The dynamics and current estimates are

State	Action	Reward	Next state	Current Q
S	exit	2	G	2
S	explore	0	S or T, each with probability 1/2	1
T	finish	8	G	4
T	wait	1	T	6

There are no further rewards after reaching G. Each new trial starts at S. The estimates need not be correct. Each part starts from the original table, and the learning rate is $\alpha = 1/2$.

(a) (5 points) The learner records $(S, \text{explore}, 0, T)$. What is its Q-learning target, and what is the updated value of $Q(S, \text{explore})$? Separately, what is the target for a recorded transition $(T, \text{finish}, 8, G)$?

Target for $(S, \textit{explore}, 0, T)$

Updated $Q(S, \textit{explore})$

Target for $(T, \textit{finish}, 8, G)$

(b) (5 points) A second learner evaluates a fixed policy π, choosing actions uniformly at S, and choosing finish with probability 3/4 and wait with probability 1/4 at T. Its estimates Q^π initially equal the table above. For $(S, \text{explore}, 0, T)$, compute its target using the expected next action under π, and its updated $Q^\pi(S, \text{explore})$. Why does its target differ from Q-learning's?

Fixed-policy target

Updated $Q^\pi(S, \textit{explore})$

Why the targets differ

(c) (5 points) Another learner starts with the original table, acts greedily from S, and has a replay buffer containing only $(S, \text{exit}, 2, G)$ transitions. Can collecting more trials with that greedy policy

and replaying this buffer guarantee learning the optimal policy? What change to data collection would address the problem?

Can greedy collection and replay guarantee an optimal policy?

Change to data collection

6. (20 points) **Returns and PPO**

We're using PPO to train a robot to put a block in a bin. A trial ends when the block lands in the bin or falls off the table. Sometimes we stop recording after a fixed number of steps while the block is still on the table. We want to use those trials for training too. How should we account for rewards the robot could still earn after we stop recording?

Suppose we record T steps. At step t, the robot takes an action in state s_t, earns reward r_t, and reaches s_{t+1}. Before the action, our critic predicted how much discounted reward the robot would earn from s_t onward. We call that prediction $V(s_t)$.

We'll work backward through the recorded rewards to estimate G_t, the return from step t onward, using discount factor γ. If the task ended, there are no more rewards. If we stopped recording early, we use the critic's prediction at s_T to account for the rest of the trial. PPO uses the difference $A_t = G_t - V(s_t)$, the *advantage*, to tell whether the outcome was better or worse than the critic predicted. We use these advantages directly ($\lambda = 1$, no normalization).

(a) (5 points) In the function below, `rewards[t]` stores r_t and `values[t]` stores the critic's prediction $V(s_t)$. The two arrays each have T entries. We pass $V(s_T)$ separately as `last_value`. The flag `terminated` is true if the trial ended in success or failure and false if we stopped recording early. What Python expression goes in each blank? We use NumPy arrays and import NumPy as `np`.

```
def targets(rewards, values, last_value, terminated, gamma):
    G = _________________________________  # (1)
    returns = np.zeros(len(rewards))
    for t in range(len(rewards) - 1, -1, -1):
        G = _____________________________  # (2)
        returns[t] = G
    advantages = ________________________  # (3)
    return returns, advantages
```

Expression (1)

Expression (2)

Expression (3)

(b) (5 points) Suppose we recorded two steps and saw rewards 1 and 2, in that order. The critic predicted $V(s_0) = 3$, $V(s_1) = 6$, and $V(s_2) = 6$. With $\gamma = 1/2$, what returns and advantages does the function give if we stopped recording early? What does it give if the task ended at s_2?

Cutoff return targets (G_0, G_1)

Cutoff advantages (A_0, A_1)

Terminal return targets (G_0, G_1)

Terminal advantages (A_0, A_1)

(c) (5 points) Now we'll use the advantages from the trial in (b) where we stopped recording early. Since we collected the data, the policy has changed. In s_0, the recorded action is now $3/2$ times as likely as it was when we collected it. In s_1, the recorded action is half as likely. These probability ratios are $\rho_0 = 3/2$ and $\rho_1 = 1/2$. With $\epsilon = 1/5$, compute the negative PPO surrogate loss

$$L = -\frac{1}{2}\sum_{i=0}^{1} \min(\rho_i A_i, \text{clip}(\rho_i, 1-\epsilon, 1+\epsilon)A_i)\,.$$

Explain the clipping choice for each sign of advantage. Does clipping guarantee that both probability ratios remain within the interval?

Negative PPO surrogate loss

Clipping choice for the positive advantage

Clipping choice for the negative advantage

Are the ratios guaranteed to stay in the interval?

(d) (5 points) An implementation sets the continuation value to zero whenever collection stops, including a cutoff. How much does that change G_0 in (b), and why is the change inappropriate for the continuing task? Separately, if evaluation reports the discounted reward actually observed in this two-transition prefix, what should it report? Should it add the critic's continuation estimate?

Change in G_0 *at a cutoff*

Why the change is inappropriate

Observed discounted prefix reward

Should the report include the critic estimate?

7. (15 points) **Pushmi-Pullyu**

We want a robot to slide a block along a straight track until its front edge lines up with a mark. A camera tells us how far the block is from the mark. To move it, we send the robot a push command of length ℓ, between 0 and 4 cm. Before trying to reach the mark, we try two pushes and measure how far the block actually moves.

Push command ℓ (cm)	2	4
Distance the block moved d (cm)	1/2	3/2

The 2 cm command moved the block only half a centimeter. If the block is now 1 cm short of the mark, what command should we send? We'll use the two measurements to fit a straight line, $\hat{d}(\ell) = \alpha\ell + \beta$, that predicts how far a given push will move the block. Then we can work backward from the distance we want it to move.

(a) (5 points) What slope α and intercept β fit the two measurements? Suppose the block is e centimeters short of the mark. According to the model, which command in $[0, 4]$ will bring it closest to the mark in one push? Give the command as a function of e, then evaluate it for $e = 1$ and $e = 3$. In the second case, can the block reach the mark in one push?

Fitted slope α

Fitted intercept β

Calibrated command as a function of e

Command for $e = 1$

Command for $e = 3$

Can one push cover 3 cm?

(b) (5 points) A teammate suggests a simpler approach. If the block is e cm short, send a push of length $\min(4, e)$, look at the camera again, and apply the same rule to the remaining distance. The idea is that we can correct a push that falls short on the next try.

Suppose the block actually responds to a command according to

$$d(\ell) = \max(0, \ell/2 - 1/2).$$

Starting 1 cm short of the mark, what will the robot do on its first push and on later attempts? Does measuring the position again correct the teammate's assumption about how far a push moves the block? What happens if we use the command from part (a) instead?

Successive pushes with the teammate's rule

Effect of replanning on model error

Result with the calibrated command

(c) (5 points) The block is again 1 cm short of the mark. This time we'll use our fitted model to compare two possible pushes before the robot moves. The candidates are $\ell^{(1)} = 2$ cm and $\ell^{(2)} = 4$ cm. We score each one by the squared distance we predict will remain between the block and the mark, $S_k = (\hat{d}(\ell^{(k)}) - 1)^2$. The planner gives more weight to a push with a smaller score and sends the weighted average command,

$$w_k = \frac{e^{-S_k/\lambda}}{e^{-S_1/\lambda} + e^{-S_2/\lambda}}, \qquad \bar{\ell} = w_1\ell^{(1)} + w_2\ell^{(2)}, \qquad \lambda > 0.$$

What score and weight does each candidate receive? What command does the planner send, and what score does the model predict for it? Would averaging two feasible, low-cost plans always give another feasible, low-cost plan? Explain using a counterexample or a constraint that averaging can violate.

Candidate scores (S_1, S_2)

Weights (w_1, w_2)

Updated command

Predicted score of the updated command

Does averaging always preserve feasibility and low cost? Explain.

8. (16 points) **A method to our madness**

Each cell describes a separate robot learning problem. Which methods have both the training data they need and a policy that can produce the requested behavior? Write the letters of *all* that apply, or a dash if none does. Use each method as listed. Discrete actions are named commands such as *left* and *right*. Continuous actions are real-valued commands.

A Deep Q-learning with one output per discrete action
B REINFORCE with a Gaussian policy
C REINFORCE with a categorical policy
D Soft actor-critic with a continuous-action policy
E Behavior cloning by least-squares regression directly on the action
F Behavior cloning by flow matching on continuous actions
G Q-learning with a recurrent network and one output per discrete action
H Behavior cloning by recurrent least-squares regression on the action
I Behavior cloning with an MLP classifier trained with softmax loss
J Behavior cloning with a recurrent layer before the softmax classifier

With a **simulator**, we can run the policy we're training and observe its rewards. With **play data**, we have a fixed collection of non-expert trajectories with a reward for every transition. In both cases we want a policy with high return. Assume the play data cover the states and actions we need, and that we can replay whole trajectories in order.

With **expert demonstrations**, the robot has only recorded trajectories without rewards. The goal is to reproduce the expert's behavior. For a single strategy, the expert consistently chooses the same action in a given state. For a mixture, the expert sometimes goes left around an obstacle and sometimes right, even given the same history. The policy must be able to produce both alternatives.

In the **full** column, the robot observes the state. Use a feedforward network for these problems. In the **partial** column, two trials can have identical recent observations but require different actions. A cue early in the trial tells the robot which situation it is in, so it must remember that cue. A feedforward network sees only the current observation. For B and C, use a feedforward policy in the full column and a recurrent policy in the partial column. All other networks are feedforward unless listed as recurrent.

		Observability	
Data source	**Action space**	**Full**	**Partial**
Simulator	Discrete		
	Continuous		
Expert, single strategy	Discrete		
	Continuous		
Expert, mixture of strategies	Discrete		
	Continuous		
Reward-labeled play data	Discrete		
	Continuous		

9. (15 points) **Some actions are more equal than others**

A robot starts in the same state on every trial. It samples one command $a \sim \mathcal{N}(\mu, \sigma^2)$ and the trial ends. Every command earns one point, with one extra point if $0 \le a \le 1$. We want the robot to learn both where to center its commands and how much to vary them.

We start with $\theta = (\mu, \ell) = (0, 0)$, where $\ell = \log \sigma^2$. Given N recorded commands a_i and their rewards r_i, we use REINFORCE with no baseline. With step size $\alpha > 0$, the update is

$$\widehat{g} = \frac{1}{N} \sum_{i=1}^{N} r_i \nabla_\theta \log \pi_\theta(a_i), \qquad \theta_{\text{new}} = \theta + \alpha \widehat{g}.$$

(a) (2 points) What are the initial mean and standard deviation? If we increase only ℓ, does the distribution shift, spread out, or both?

Initial mean

Initial standard deviation

Effect of increasing only ℓ

(b) (3 points) For a recorded action a, the log density is

$$\log \pi_\theta(a) = -\tfrac{1}{2}\big(\log(2\pi) + \ell + (a - \mu)^2 e^{-\ell}\big).$$

Holding a fixed, what is the derivative with respect to ℓ in terms of a, μ, and ℓ? At the initial parameters, if we observe a single trial with $a = 0.5$, would a REINFORCE step increase or decrease the variance?

Derivative with respect to ℓ

Variance change for $a = 0.5$

(c) (5 points) For this calculation, use a batch with ten trials at $a = 0.5$ with reward 2, ten at $a = -1$ with reward 1, and ten at $a = 2$ with reward 1. At the initial parameters, the derivative of the log density with respect to μ is a. What are the new parameters after one batch update?

Updated mean

Updated log variance

(d) (2 points) Now we hold $\mu = 0.5$ fixed. Suppose we start with $\sigma = 1$, sample $a = 0.5$, and receive reward 2. After one REINFORCE step of size α on ℓ, what is the new σ?

(e) (2 points) Again we start with $\mu = 0.5$ and $\sigma = 1$. Suppose we sample $a = \epsilon$, where $0 < \epsilon < 0.5$, and receive reward 2. What is the new σ after one REINFORCE step of size α on ℓ?

(f) (1 point) With $\mu = 0.5$ fixed, what must σ approach for the expected reward to approach 2? What action does the limiting policy take?

Limiting standard deviation

Action of the limiting policy

10. (12 points) **Pushing the point**

(a) (2 points) In PS6 we cloned a teacher that reliably pushed a block to its goal. In what way was the teacher sub-optimal?

Behavior of the teacher

(b) (4 points) In PS6 the cloned policy chose which face of the block to push. To choose how far to push, we sampled a number z from a Gaussian and converted it to a push distance. PPO learned one parameter, θ. It added θg to the cloned Gaussian mean, so a larger θ favored longer pushes wherever $g > 0$. The notebook set $g = 0$ when the remaining distance to the goal along the push direction was at most 7 cm.

Suppose we recorded a push with 3 cm left to go along that direction. After a PPO update increases θ, we evaluate the same face and the same z in the same state. Why is the likelihood of that action under the updated policy divided by its likelihood under the old policy exactly 1?

Reason for the ratio near the goal

Now suppose the block has 12 cm left to go along the push direction, so $g = 1$. For the chosen face, we test the action whose z equals the old Gaussian mean. After the same increase in θ, is the likelihood ratio for that action above or below 1? Why?

Likelihood ratio at the old mean

(c) (2 points) In PS7 the MPPI planner looked two pushes ahead. Once it chose a plan, how many of those pushes did the robot execute in the simulator before it observed the new state and planned again?

Pushes executed before replanning

(d) (4 points) In the PS7 evaluation, the MPPI planner took slightly more pushes on average than the calibrated controller, yet finished sooner in simulated time. Suppose we change `rollout_cost` to charge a fixed amount per push in place of the charge for predicted execution time. If two plans reach the same goal in the same number of valid pushes, what distinction can the original cost make that the modified cost loses?

What the two costs can distinguish

How could the finger's position at the start of a push make one of those plans faster?

Example involving the finger



End of practice problems.