**Homework 04**

**AA203: Optimal and learning-based Control**

Ricardo Luiz Moreira Paschoeto

rp304154@stanford.edu

HW02, 03 and 04 was done in colaboration with my group partner Srikanth.

**Problem 1:** Deep reinforcement learning

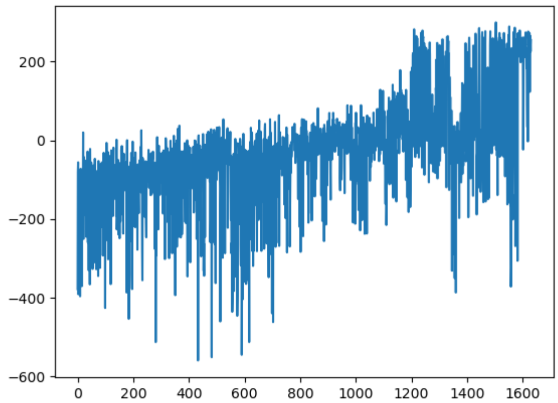


*Figura 1 - episodic_rewards x episodes*

Python code:

```
    def forward(self, x):
        """
        forward of both actor and critic
        """
        # TODO map input to
        # mean of action distribution
        # variance of action distribution (pass this thro
ugh a non-negative function)
        # state value

        x = F.relu(self.affine1(x))
        x = F.relu(self.affine2(x))

        a_mean = self.action_mean(x)
        a_var = torch.exp(self.action_var(x))
        s_values = self.value_head(x)

        return 0.5*a_mean, 0.5*a_var, s_values
```

```python
def select_action(state):

    state = torch.from_numpy(state).float()
    mu, sigma, state_value = model(state)

    # create a normal distribution over the continuous action space
    m = Normal(loc=mu,scale=sigma)

    # and sample an action using the distribution
    action = m.sample()

    # save to action buffer
    model.saved_actions.append(SavedAction(m.log_prob(action), state_value))

    # the action to take (left or right)
    return action.data.numpy()
```

```python
def finish_episode():
    """
    Training code. Calculates actor and critic loss and performs backprop.
    """
    R = 0
    saved_actions = model.saved_actions
    policy_losses = [] # list to save actor (policy) loss
    value_losses = [] # list to save critic (value) loss
    returns = [] # list to save the true values

    # calculate the true value using rewards returned from the environment
    for r in model.rewards[::-1]:
        # TODO compute the value at state x
        # via the reward and the discounted tail reward
        R = r + args.gamma * R
        returns.insert(0, R)
```

```
    returns = torch.tensor(returns)
    returns = (returns - returns.mean()) / (returns.std()
 + eps)

    # whiten the returns
    returns = torch.tensor(returns).float()
    returns = (returns - returns.mean()) / (returns.std()
 + eps)

    for (log_prob, value), R in zip(saved_actions, return
s):
        # TODO compute the advantage via subtracting off
value

        adv = R - value.item()

        # TODO calculate actor (policy) loss, from log_pr
ob (saved in select action)
        # and from advantage
        policy_losses.append(-log_prob * adv)


        # append this to policy_losses

        # TODO calculate critic (value) loss
        value_losses.append(F.mse_loss(value, R))

    # reset gradients
    optimizer.zero_grad()

    # sum up all the values of policy_losses and value_lo
sses
    loss = torch.stack(policy_losses).sum() + torch.stack
(value_losses).sum()

    # perform backprop
    loss.backward()
    optimizer.step()
```

```
# reset rewards and action buffer
del model.rewards[:]
del model.saved_actions[:]
```

**Problem 2:** Extremal curves

$$J = \int_0^1 \left(\frac{1}{2}x(\dot{t})^2 + 5x(t)x(\dot{t}) + x(t)^2 + 5x(t)\right)dt$$

Where:

$$g\big(x(t), x(\dot{t}), t\big) = \frac{1}{2}x(\dot{t})^2 + 5x(t)x(\dot{t}) + x(t)^2 + 5x(t).$$

Applying Euler Equation,

$$g_x(x, \dot{x}, t) - \frac{d}{dt}g_{\dot{x}}(x, \dot{x}, t) = 0\ (I)$$

Where:

$$g_x = \frac{\partial}{\partial x}g(x, \dot{x}, t),$$

$$g_{\dot{x}} = \frac{\partial}{\partial \dot{x}}\ g(x, \dot{x}, t).$$

Computing:

$$g_x = 5x(\dot{t}) + 2x(t) + 5\ and\ g_{\dot{x}} = x(\dot{t}) + 5x(t).$$

Replacing in (I):

$$5x(\dot{t}) + 2x(t) + 5 - \frac{d}{dt}\big[x(\dot{t}) + 5x(t)\big] = 0$$

$$x(\ddot{t}) - 2x(t) - 5 = 0,$$

$$x(\ddot{t}) - 2x(t) = 5\ (II)$$

Homogenous Equation:

$$x(\ddot{t}) - 2x(t) = 0,$$

$$r^2 - 2r = 0 => r = 0\ or\ r = 2.$$

Then:

$$x_h(t) = C_1 + C_2e^{2t}(III)$$

Particular Solution:

$$x_p(t) = B,$$

$$\ddot{x}_p(t) = 0,$$

Replacing in (II):

$$0 - 2B = 5 => B = -\frac{5}{2}.$$

Overall Solution:

$$x(t) = C_1 + C_2e^{2t} - \frac{5}{2}.$$

Using conditions:

$$x^*(0) = 1 \ and \ x^*(1) = 3.$$

$$x^*(0) = C_1 + C_2 - \frac{5}{2} = 1,$$

$$x^*(1) = C_1 + C_2 e^2 - \frac{5}{2} = 3.$$

Solving the system we have:

$$C_1 = \frac{7e^2 - 11}{2(e^2 - 1)} \ and \ C_2 = \frac{2}{(e^2 - 1)}$$

Then the extremal curve:

$$x(t)^* = \frac{e^2 - 3}{e^2 - 1} + \frac{2e^{2t}}{e^2 - 1}$$

**Problem 3:** Minimum control effort

$$\dot{x(t)} = -2x(t) + u(t), x(0) = 2 \; and \; x(1) = 0.$$

$$J = \int_0^1 u(t)^2 dt.$$

Hamiltonian:

$$H(x(t), u(t), p(t), t) = u(t)^2 + p(t)^T\big(-2x(t) + u(t)\big)$$

Necessary Optimality:

$$\dot{x(t)}^* = \frac{\partial}{\partial p} H(x(t), u(t), p(t), t), (I)$$

$$\dot{p}^*(t) = -\frac{\partial}{\partial x} H(x(t), u(t), p(t), t), (II)$$

$$\frac{\partial}{\partial u} H(x(t), u(t), p(t), t) = 0, (III)$$

Computing:

From (I):

$$2u(t) + p(t) = 0 => u(t) = -\frac{1}{2}p(t), (IV)$$

From (II):

$$\dot{x(t)}^* = -2x(t) - \frac{1}{2}p(t)$$

From (III):

$$\dot{p}^*(t) = -2 => p(t) = -2\int dt => p(t) = -2t + C.$$

Using boundary condition, $t_f \; free \; and \; x\big(t_f\big) = x_f$:

$$H(x(t), u(t), p(t), t) = u(t)^2 + p(t)^T\big(-2x(t) + u(t)\big) = 0.$$

Computing:

$$u(t)[-u(t) + 4x(t)] = 0 => x(t) = \frac{u(t)}{4}$$

Replacing in (I):

$$\dot{x(t)}^* = -2x(t) + u(t) = -2\left(\frac{u(t)}{4}\right) + u(t) = \frac{u(t)}{2}, (V)$$

Where:

$$u(t) = -\frac{1}{2}p(t) = t - \frac{C}{2}$$

Integrating (V):

$$x(t)^* = x(0) + \frac{1}{2}\int_0^t (\sigma - \frac{C}{2})d\sigma$$

$$x(t)^* = 2 + \frac{t^2}{4} - \frac{1}{2}Ct$$

Using $x(1) = 0$,

$$0 = 2 + \frac{1}{4} - \frac{C}{2} => C = \frac{9}{2}$$

Finally:

$$u(t)^* = t - \frac{9}{4}t$$

$$x(t)^* = \frac{t^2}{4} - \frac{9}{4}t + 2.$$

**Problem 4:** Zermelo's ship

$$\dot{x}(t) = v\cos\theta(t) + \omega\big(y(t)\big)$$
$$\dot{y}(t) = v\sin\theta(t)$$

a)

$$\omega\big(y(t)\big) = \frac{v}{h}y(t), h > 0$$

$$H(x(t), u(t), p(t), t) = [p(t)_1 \quad p(t)_2]\begin{bmatrix} v\cos\theta(t) + \frac{v}{h}y(t) \\ v\sin\theta(t) \end{bmatrix}$$

$$H(x(t), u(t), p(t), t) = p(t)_1\left[v\cos\theta(t) + \frac{v}{h}y(t)\right] + p(t)_2 v\sin\theta(t)$$

Using Necessary Optimality Conditions:

$$-\frac{\partial}{\partial x(t)}H = \dot{p(t)}_1^* = 0$$

$$\frac{\partial}{\partial y(t)}H = \dot{p(t)}_2^* = -\frac{v}{h}p(t)_1$$

$$\frac{\partial}{\partial \theta(t)}H = -p(t)_1 v\sin\theta(t) + p(t)_2 v\cos\theta(t) = 0, (I)$$

Then:

$$p(t)_1^* = C_1$$

$$p(t)_2^* = -\frac{v}{h}C_1 t + C_2$$

From (I):

$$\tan\theta^*(t) = \frac{p(t)_2^*}{p(t)_1^*} = \frac{-\frac{v}{h}C_1 t + C_2}{C_1}$$

$$\tan\theta^*(t) = \frac{C_2}{C_1} - \frac{v}{h}t$$

Where $\frac{C_2}{C_1} = \alpha$:

$$\tan\theta^*(t) = \alpha - \frac{v}{h}t$$

b) $\omega\big(y(t)\big) \equiv \beta$.

$$H(x(t), u(t), p(t), t) = p(t)_1[v\cos\theta(t) + \beta] + p(t)_2 v\sin\theta(t)$$

$$-\frac{\partial}{\partial x(t)}H = \dot{p(t)}_1^* = 0$$

$$\frac{\partial}{\partial y(t)} H = p\dot{(t)}_2^* = 0$$

$$\tan \theta(t) = \frac{p(t)_2}{p(t)_1} = \frac{C_1}{C_2}$$

From state equation y(t):

$$y(t_f)^* = y_0 + v \int_{t_0}^{t_1^*} \sin \sigma \, d\sigma$$

$$0 = y_0 + vC_1[t_1^* - t_0]$$

$$t_1^* - t_0 = \frac{y_0}{vC_1}$$

$$C_1 = \sin \theta(t)$$