12. Final project

This project is concerned with implementing monte carlo policy gradient, also known as REINFORCE, and also the improvement that comes with adding a baseline.

The motivating example we will use to drive the discussion is the episodic option-hedging task that was discussed in lecture, although your code can be written in a way that is more general.

Consider a single European call option, which is a contract to buy 100 shares of a stock, called the underlier, for a pre-set price of K dollars, called the strike price, on a date T called the maturity or expiration date. Assume that we are a hypothetical bank who owns this option and cannot trade the option itself (perhaps there is no market for the option), but the bank can trade the underlier which is assumed to be very liquid. The bank has no view on the option's expected return and just wants to mitigate the risk by hedging.

The underlier (stock) price at time t will be denoted S_t. We take the strike and maturity as fixed, exogenously-given constants. For simplicity, we assume the risk-free rate is zero. The agent we train will learn to hedge this specific option with this strike and maturity.

For European options, the state must contain the current price S_t of the underlying, and the time $\tau := T - t > 0$ still remaining to expiry, as well as our current position of n shares. The state is thus naturally an element of

$$\mathbb{R}_+^2 \times \mathbb{Z} = \{(S, \tau, n) \mid S > 0, \tau > 0, n \in \mathbb{Z}\}.$$

This state space is naturally embedded in $\mathbb{R}^3$, so we will identify our states with 3-dimensional vectors. A state will be denoted by lower-case s, so

$$s_t = (S_t, T - t, n_t).$$

The action space will have 101 elements: each action a_t is a number of shares of the underlier to trade on day t, from 0 to 100, inclusive. If the action space is discrete and not too large, then a common kind of parameterization is to form parameterized numerical preferences $h(s, a, \theta) \in \mathbb{R}$ for each state-action pair (s, a), where θ is a single parameter or parameter vector that is updated during the training process.

Let P_t denote the aggregate profit (or loss, if negative) in dollars on day t from both the price changes in the option position, which cannot be traded, and also the price changes in the underlier. Thus

$$P_t = P_t^{\text{opt}} + P_t^{\text{stock}}$$

For this project, since our goal is to minimize risk (or variance), we take the single-period reward signal to be

$$r_t = -\frac{1}{T} P_t^2$$

Then of course the $G = \sum_{t=1}^{T} r_t$ for an episode is then simply the negative PL variance over the episode.

In this project, for concreteness we shall restrict attention to policies constructed from an exponential soft-max in action preferences:

$$\pi(a \mid s, \boldsymbol{\theta}) := \frac{e^{h(s,a,\boldsymbol{\theta})}}{\sum_{a'} e^{h(s,a',\boldsymbol{\theta})}}$$

Further assume that we are in a discrete-time Black-Scholes world. For concreteness, take $T = 30$ and assume trading decisions in the underlier are made once per day, on days $t = 1, 2, \dots, T$.

According to Black-Scholes theory, the optimal share position to hold in the underlier is

$$100N(d_1), \quad \text{where } d_1 = \frac{\ln \frac{S_t}{K} + \frac{\tau\theta^2}{2}}{\theta\sqrt{\tau}},$$

Note that the strike K is a constant which does not change for the problem, and (S_t, τ) are two parts of the three-dimensional state variable, and θ is an unknown parameter. You should construct your action-preference function $h(s, a, \boldsymbol{\theta})$ to be peaked when a equals the trade which gets you closest to a share position of $100N(d_1)$.

Problem 1. Implement Monte-Carlo Policy-Gradient Control for π_*, or REINFORCE. For completeness, we recall the algorithm:

```
Loop for each episode:
    (1) Generate an episode $s_0, a_0, r_1, \dots$ following policy $\pi(\cdot \mid \cdot, \boldsymbol{\theta})$
    (2) Loop for each timestep $t$ of the episode:
        (a) $G = \sum_{k=t+1}^{T} r_k$
        (b) $\boldsymbol{\theta} = \boldsymbol{\theta} + \alpha G \nabla \ln \pi(a_t \mid s_t, \boldsymbol{\theta})$
```

Do not confuse the state s_t with the underlier price S_t which is one component of the state.

Using this algorithm train an agent over the episodes provided in the training set. Evaluate how quickly your algorithm learns, by plotting the G for each training episode, as in Sutton-Barto Figure 13.1. Evaluate the trained agent performance over the episodes in the test set, by plotting the variance obtained on each test-set episode.

Experiment with various ways of reducing α as the training proceeds. Report the best such method that you find. The plots that you submit of train/test performance should be with respect to the best α-reducing scheme. Also report the value of θ which the final policy is using.

Problem 2. Add a baseline to the REINFORCE algorithm as discussed in class. For completeness, here is the official statement of the algorithm:

```
Loop for each episode:
  (1) Generate an episode $s_0, a_0, r_1, \ldots$ following policy $\pi(\cdot \mid \cdot, \boldsymbol{\theta})$
  (2) Loop for each timestep $t$ of the episode:
      (a) $G = \sum_{k=t+1}^{T} r_k$
      (b) $\delta = G - \hat{v}(s_t, \boldsymbol{w})$
      (c) $\boldsymbol{w} = \boldsymbol{w} + \alpha^{(w)} \delta \, \nabla_{\boldsymbol{w}} \hat{v}(s_t, \boldsymbol{w})$
      (d) $\boldsymbol{\theta} = \boldsymbol{\theta} + \alpha \delta \, \nabla \ln \pi(a_t \mid s_t, \boldsymbol{\theta})$
```

where $\alpha^{(w)}$ is a different learning rate for $\boldsymbol{w}$.

We will consider two possible forms for the baseline function $\hat{v}$: linear and quadratic. For example, for the linear form, $\boldsymbol{w} \in \mathbb{R}^4$ and

$$\hat{v}((S, \tau, n), \boldsymbol{w}) = w_0 + w_1 S + w_2 \tau + w_3 n$$

For the quadratic form, $\boldsymbol{w}$ includes all of the coefficients in a general quadratic function of three variables.

Do all of the same tuning of learning rates and evaluations as in the previous problem over the training episodes and the test episodes. With the best α-reduction scheme that you found, does the baseline cause the agent to learn more quickly? Compare with and without baseline, as in Sutton-Barto Figure 13.2. What is the learned parameter θ at the end of training?