Code Assignment: **Find me if you can!**
ECE 595: Reinforcement Learning

September 27, 2023

In this code assignment (in Python!), you will be implementing a bandit algorithm with an application in Positioning, Navigation, and Timing (PNT) systems. The task involves simulating how a target can select three anchor nodes simultaneously to improve its position estimation accuracy using a bandit algorithm. The accuracy of the position estimation will be determined using the trilateration technique. Trilateration is a method used to determine the position of a target based on the time differences of arrival (TDOA) of signals from multiple reference points, known as anchor nodes. By measuring the pseudoranges, of the target from each selected anchor node, the target's position can be estimated by intersecting spheres centered at the anchor nodes.

Find me if you can!

Implement a bandit algorithm to simulate simultaneous anchor node selection by one target in a PNT system. The algorithm should select three anchor nodes at each time step and calculate the accuracy (reward) of position estimation using the trilateration technique.

Step 1: Initialize the problem parameters.

For this problem, five anchor nodes are considered for the target to select from. The total number of time-steps or rounds for the simulation is also provided in the starter code. The anchor node positions and an exemplary target position are given to have reproducible results. Also, to reduce the stochasticity in the results, an initial estimate of the target's position is given which will be used to predict the final estimated position. In addition, two ϵ values are considered to evaluate the performance of the simple Bandit algorithm for $\epsilon = 0.01$ and $\epsilon = 0.3$.

Step 2: Implement the Bandit Algorithm.

Suppose, (x_a, y_a, z_a) represents the given position of each anchor node and (x_t, y_t, z_t) represents the given position of the target. So, determine the real ranges $d_{a,t}$ between the target t and each anchor node a. The pseudoranges $\hat{d}_{a,t}$ are the distances of the target from each anchor node including some noise due to inherent uncertainties and errors that can occur in real-world distance measurements. For convenience of simulation, the code for measuring the pseudoranges of the target from an anchor node by adding small random variations to the actual range measurements is already given.

Let, the initial position estimate of the target be represented by $\hat{\mathbf{p}}^{(ite)=0} = (\hat{x}_t^{(ite)=0}, \hat{y}_t^{(ite)=0}, \hat{z}_t^{(ite)=0})$ where ite represents the time-step or round. At each time step, choose three anchor nodes (this is your action A) based on the simple bandit algorithm. The selected anchor nodes and their positions along with the target's estimated position and the measured pseudoranges of the selected anchor nodes are then used to determine the Jacobian $J^{(ite)}$ (using Eq. 1). The Jacobian matrix $J^{(ite)}$ is a 3×3 matrix of the functions of the pseudoranges created by setting the functions equal to 0, e.g., $\hat{d}_{a,t} = \sqrt{(x_a - \hat{x}_t^{(ite)})^2 + (y_a - \hat{y}_t^{(ite)})^2 + (z_a - \hat{z}_t^{(ite)})^2} \leftrightarrow f(\hat{x}_t^{(ite)}, \hat{y}_t^{(ite)}, \hat{z}_t^{(ite)}) = \sqrt{(x_a - \hat{x}_t^{(ite)})^2 + (y_a - \hat{y}_t^{(ite)})^2 + (z_a - \hat{z}_t^{(ite)})^2} - \hat{d}_{a,t}$ (you will have 3 functions like the f, i.e., f, g, h because you choose 3 anchor nodes).

$$J^{(ite)} = \begin{bmatrix} \frac{\partial f(\hat{\mathbf{p}}^{(ite)})}{\partial x_t} & \frac{\partial f(\hat{\mathbf{p}}^{(ite)})}{\partial y_t} & \frac{\partial f(\hat{\mathbf{p}}^{(ite)})}{\partial z_t} \\ \frac{\partial g(\hat{\mathbf{p}}^{(ite)})}{\partial x_t} & \frac{\partial g(\hat{\mathbf{p}}^{(ite)})}{\partial y_t} & \frac{\partial g(\hat{\mathbf{p}}^{(ite)})}{\partial z_t} \\ \frac{\partial h(\hat{\mathbf{p}}^{(ite)})}{\partial x_t} & \frac{\partial h(\hat{\mathbf{p}}^{(ite)})}{\partial y_t} & \frac{\partial h(\hat{\mathbf{p}}^{(ite)})}{\partial z_t} \end{bmatrix} \tag{1}$$

Afterwards, calculate the GDOP value $GDOP(A)$ for choosing the action A, and measure the reward $R(A)$ from the selected anchor nodes. The reward is defined as $R(A) = \frac{\sqrt{\frac{10}{3}}}{GDOP(A)}$, where $GDOP(A)$ is defined as the Geometric Dilution of Precision (GDOP) and $GDOP(A) = \sqrt{\sum_{\forall i} G(i,i)}$, $G = (J^{(ite)^T} \cdot J^{(ite)})^{-1}$, where $J^{(ite)^T}$ is the transpose of the Jacobian matrix $J^{(ite)}$.

Given the $R(A)$, follow the steps of the simple bandit algorithm and update $N(A)$ and $Q(A)$.

Before selecting the new action, you need to update the position estimate of the target as follows: $\hat{\mathbf{p}}^{(ite+1)} = (\hat{x}_t^{(ite)} - \Delta\hat{x}_t, \hat{y}_t^{(ite)} - \Delta\hat{y}_t, \hat{z}_t^{(ite)} - \Delta\hat{z}_t)$, where $(\Delta\hat{x}_t, \Delta\hat{y}_t, \Delta\hat{z}_t) = (J^{(ite)^T} \cdot J^{(ite)})^{-1} \cdot J^{(ite)^T} \cdot \mathcal{RES}^{(ite)}$ where $\mathcal{RES}^{(ite)}$ is the residual matrix.

$$\mathcal{RES}^{(ite)} = \begin{bmatrix} f(\hat{\mathbf{p}}^{(ite)}) \\ g(\hat{\mathbf{p}}^{(ite)}) \\ h(\hat{\mathbf{p}}^{(ite)}) \end{bmatrix} \tag{2}$$

Step 3: Plot and analyze the results.

Run the algorithm for the specified number of rounds. You will have two sets of the following results by taking ϵ to be 0.01 and 0.3.

Record and display:
1. GDOP versus steps of the simple bandit algorithm
2. Reward versus steps of the simple bandit algorithm
3. Euclidean distance between the real (given in Step 1) and estimated position of the target versus steps of the simple bandit algorithm

Did you find me?!!!