Evita Obstáculos

Robótica Adaptativa

Mario Romero Courel

USC

1. Introducción:

Para este trabajo, he decidido desarrollar, mediante una DDPG, el comportamiento de evitar obstáculos en el Turtlebot. Este trabajo ha supuesto muchos retos, como el manejo de los datos del LIDAR, el espacio de acciones y el ajuste de la propia red.

2. Entorno:

El espacio de observación define las características del estado del entorno que el agente puede percibir. En nuestro caso, utilizamos datos del LIDAR (Light Detection and Ranging) para representar la información del entorno. El LIDAR proporciona mediciones de distancia en 360 grados alrededor del robot. Normalizamos estos datos para que tengan una media de cero y una desviación estándar de uno. De esta manera siempre que esté lejos de un objeto devolverá un array de 1 y conforme se vaya acercando este número se reducirá

El espacio de acción describe las acciones que el agente puede tomar. En nuestro entorno, el agente puede tomar velocidad lineales y angulares. Utilizamos un espacio continuo para representar estas acciones, con valores entre -1.0 y 1.0.

La función de recompensa guía el aprendizaje del agente. En nuestro caso, no se proporciona una función de recompensa específica en el código, pero en un entorno real, diseñaríamos una función que motive al agente a lograr ciertos objetivos (por ejemplo, evitar colisiones o alcanzar una ubicación específica).

Transiciones:

Las transiciones representan cómo el estado del entorno cambia cuando el agente toma una acción. En nuestro código, actualizamos los datos del LIDAR en la función scan_callback. El agente debe aprender a mapear las observaciones del LIDAR a acciones que maximicen la recompensa esperada.

Ademas, la condición de finalización es que el mínimo valor en los datos del LIDAR sea inferior a 0.5. Esto podría representar una colisión inminente o una situación peligrosa.

3. RED DDPG:

Esta sección describe la implementación de un agente Deep Deterministic Policy Gradient (DDPG). DDPG es un algoritmo de aprendizaje por refuerzo diseñado para espacios de acción continuos. Combina las ideas de Deep Q-Networks (DQNs) con gradientes de políticas deterministas para una exploración y aprendizaje eficientes.

El agente DDPG consta de varios componentes clave:

Red Actor: Esta red neuronal toma el estado como entrada y genera una acción determinista. Representa la política del agente para seleccionar acciones en un estado dado.

Red Crític: Esta red neuronal toma tanto el estado como la acción como entrada y genera un valor Q, que representa la recompensa futura esperada por tomar esa acción en ese estado.

Redes Objetivo: Se utilizan dos redes adicionales, un actor objetivo y un crítico objetivo, para mejorar la estabilidad del aprendizaje. Estas redes se actualizan periódicamente con los pesos de los modelos de actor y crítico principales.

Memoria: El agente almacena experiencias pasadas (estado, acción, recompensa, siguiente estado, hecho) en un búfer de reproducción para su entrenamiento.

El agente opera en un bucle episódico, interactuando con el entorno y aprendiendo a través de los siguientes pasos:

Primero observa el estado actual y selecciona una acción. Utiliza una estrategia de exploración-explotación (e-greedy), donde con cierta probabilidad (epsilon), toma una acción aleatoria para la exploración. En caso contrario, elige la acción predicha por el modelo de actor. Posteriormente almacena la tupla (s,a,r,s´) en la memoria

Posteriormente el agente calcula el valor Q objetivo utilizando la recompensa y el valor Q del siguiente estado de la red crítica objetivo. Luego, el modelo crítico se entrena para minimizar la diferencia entre el valor Q predicho y el valor Q objetivo.

El modelo de actor se entrena para maximizar el valor Q obtenido por el modelo crítico para el estado actual y las acciones que propone.

Los pesos de las redes de actor y crítico objetivo se actualizan periódicamente con los pesos de las redes de actor y crítico principales.

La tasa de exploración (epsilon) se reduce gradualmente con el tiempo para animar al agente a explotar la política aprendida.

Este proceso permite al agente aprender la política óptima para maximizar las recompensas a largo plazo a través de ensayo y error, utilizando el búfer de reproducción para un aprendizaje eficiente y las redes objetivo para un entrenamiento estable.