Sistemas Inteligentes

Aprendizaje por refuerzo

Francisco Martinez(frammnm@gmail.com / 794893@unizar.es)

Introduccion

El documento a continuación expresa el proceso de implementación de ***Value Iteration*** y ***QLearning*** en el contexto de un ambiente cuadriculado donde se deben superar distintos objetivos en cuanto a comportamientos y alcance de soluciones en el espacio de exploración. Se habla del trasfondo teórico detrás del algoritmo como también se comentarán los análisis o propiedades necesarias en los parámetros del agente para converger a los comportamientos deseados. Finalmente se realizará una síntesis de los resultados obtenidos de cada algoritmo y su coherencia.

Marco teorico

En estas dos prácticas se trabajan soluciones derivadas de los procesos de decisión de markov que tienen como objetivo el conseguir una política que dado un estado cualquiera del espacio de exploración que se maneja esta pueda devolver la acción a tomar, la forma en que esta política puede ser calculada posee variantes y específicamente manejaremos Value Iteration y Q Learning.

- **Value Iteration :** Este algoritmo realiza el cálculo de política a través de un vector de valores para cada uno de los distintos estados del problema, el concepto de valor representa la evaluación de qué tan probable es pasar a ese estado, la recompensa asociada a tomar dicha acción y un descuento(parametrizado) que nos permite expresar si valoramos más recompensas futuras o presentes. La política busca maximizar la ganancia producida por la solución por lo cual las acciones a tomar dentro de la política serán decidas por las que nos guíen a los estados del mayor valor posible.

$$V_{i+1}(s) := \max_a \left\{ \sum_{s'} P_a(s, s') \left(R_a(s, s') + \gamma V_i(s') \right) \right\}$$

- **QLearning :** Este es un algoritmo posee la característica de que no posee un modelo asociado, el agente a través de ensayo y error con el sistema converge en la formulación

de una política eficiente. El proceso de decisión es sumamente similar al de Value Iteration explicado previamente pero ya que posee los mismos conceptos de recompensa, y descuento para acciones lejanas en el futuro pero con la variación de que también se valoran las recompensas pasadas entre recientes ocurrieron en el tiempo, un promedio de estos valores viejos y futuros en combinación de un parámetro épsilon que ayuda al agente a tomar decisiones aleatorias para que pueda explorar todo el espacio de búsqueda del problema.

$$Q^{new}(s_t, a_t) \leftarrow (1-\alpha) \cdot \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\left(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} \right)}^{\text{learned value}}$$

Ejercicios de análisis

A continuación se realizará una explicación de los distintos parámetros que fueron decididos para la resolución de los distintos desafíos o comportamientos requeridos:

```
def question2():
      answerDiscount = 0.9
      answerNoise = 0.0
      return answerDiscount, answerNoise
```

2) En esta pregunta se tiene como desafío hacer que el agente cruzara de un extremo a otro a través de un camino rodeado de recompensas negativas, para lograrlo se colocó un descuento bajo(cercano a 1) para poder apreciar la recompensa que se encuentra lejos en el tiempo además de eliminar el ruido al momento de tomar acciones para que el agente no cayera en los extremos de recompensa negativa por aleatoriedad.

```
def question3a():
      answerDiscount = 0.2
      answerNoise = 0.0
      answerLivingReward = 0.2
      return answerDiscount, answerNoise, answerLivingReward
```

3a) El requerimiento de preferir una salida cercana se manipula a través del descuento donde colocando un valor alto(cercano a 0) preferimos recompensas más inmediatas, pero con baja recompensa mientras se encuentre vivo para que así tome el camino más inmediato(que seria el del acantilado). En este caso el ruido puede ser bajo ya que esta es el comportamiento mas facil de adoptar sin requerir exploracion.

```
def question3b():
      answerDiscount = 0.3
      answerNoise = 0.3
      answerLivingReward = 0.2
      return answerDiscount, answerNoise, answerLivingReward
```

3b) Preferir una salida cercana nuevamente se resuelve con la misma decisión para el parámetro de descuento pero ahora sí es vital que exista una recompensa mientras está vivo como también un poco de ruido para que el la política refleje la exploración de caminos necesaria para llegar a este comportamiento.

```
def question3c():
      answerDiscount = 0.9
      answerNoise = 0.0
      answerLivingReward = 0.1
      return answerDiscount, answerNoise, answerLivingReward
```

3c) Ahora para preferir la salida más distante ponemos bajo descuento para valorar la recompensa lejana, poco ruido para evitar exploración, y poca recompensa mientras viva para que no tome desviaciones en su recorrido.

```
def question3d():
      answerDiscount = 0.9
      answerNoise = 0.2
      answerLivingReward = 0.3
      return answerDiscount, answerNoise, answerLivingReward
```

3d) Para lograr el comportamiento de salida distante repetimos el descuento bajo pero aumentamos el ruido para permitir más exploración como también la recompensa mientras se esté vivo lo ayuda a considerar caminos más largos.

```
def question3e():
      answerDiscount = 1.0
      answerNoise = 0.1
      answerLivingReward = 1.0
      return answerDiscount, answerNoise, answerLivingReward
```

3e) Para lograr que un episodio no termine ya que nunca se desea llegar a una salida lo necesario es aumentar la recompensa mientras se esté vivo para que la recompensa de las salidas sea despreciable por lo cual nunca preferirá una casilla de salida.

```
def question8():
      answerEpsilon = 0.5
      answerLearningRate = 0.5
      # return answerEpsilon, answerLearningRate
      return 'NOT POSSIBLE'
```

8) En esta pregunta nos piden resolver el problema de la pregunta dos nuevamente pero con Qlearning, pero no existe forma de manipular los parámetros para que solo en 50 iteraciones encuentra la respuesta, la cantidad de entrenamiento es muy baja.

Resultados

Los resultados fueron obtenidos de forma exitosa por la herramienta de auto calificación que existe en los archivos del proyecto, en particular los resultados de QLearning obtenidos resuelven 100% de los casos por lo que se interpreta que los algoritmos representan de forma correcta las distintas estrategias. Se considera que los resultados son coherentes con la hipótesis teórica presentada originalmente y en particular se aprecia la consistencia de los distintos parámetros de los algoritmos para alcanzar comportamientos distintos.