TIC

Actividad 1

Introducción

Universidad Tecnológica de Pereira

faceIT

Introducción

En la actividad de Introducción al Aprendizaje por Refuerzo, los estudiantes serán introducidos al concepto fundamental del Aprendizaje por Refuerzo (RL). Se les explicará en detalle los componentes esenciales del RL, incluyendo el agente, el entorno, las acciones, los estados y las recompensas, proporcionando una comprensión profunda de cómo estos elementos interactúan en el proceso de aprendizaje. Además, se resaltará la importancia del Aprendizaje por Refuerzo en la toma de decisiones autónomas, mostrando cómo esta técnica se utiliza en una amplia gama de aplicaciones, desde robótica hasta sistemas de recomendación, para mejorar la capacidad de los agentes de tomar decisiones óptimas en entornos dinámicos y complejos.

Definición de Aprendizaje por Refuerzo

El Aprendizaje por Refuerzo (RL) es un paradigma de aprendizaje automático en el que un agente aprende a tomar decisiones secuenciales para alcanzar un objetivo específico a través de la interacción con un entorno. A diferencia del aprendizaje supervisado, donde el agente recibe ejemplos etiquetados, y del aprendizaje no supervisado, donde el agente encuentra patrones en los datos sin supervisión, en RL el agente aprende a través de la retroalimentación continua que recibe del entorno en forma de recompensas o penalizaciones.

Explicación de los componentes clave:

+ info

Importancia del RL en la toma de decisiones autónomas:

El Aprendizaje por Refuerzo es crucial en la toma de decisiones autónomas debido a su capacidad para aprender de la experiencia y adaptarse a entornos complejos y dinámicos. Permite que los agentes tomen decisiones óptimas en situaciones donde la retroalimentación es escasa, incompleta o tardía. Esto es especialmente importante en aplicaciones como robótica, juegos, sistemas de control, finanzas y atención médica, donde los agentes deben interactuar con entornos inciertos y no deterministas para lograr objetivos específicos. Mediante el aprendizaje por refuerzo, los agentes pueden aprender a tomar decisiones autónomas que maximicen las recompensas a largo plazo, lo que los hace más eficientes y adaptables en diversas situaciones del mundo real.

Recompensas

Son señales de retroalimentación que el agente recibe del entorno después de tomar una acción en un estado determinado. Las recompensas indican al agente si la acción tomada fue beneficiosa o no para alcanzar el objetivo deseado. El objetivo del agente es aprender a maximizar las recompensas acumuladas a lo largo del tiempo.

Estados

Representan la situación actual del entorno en un momento dado. Los estados proporcionan información al agente sobre su posición y las condiciones circundantes, lo que le permite tomar decisiones informadas sobre las acciones a tomar.

Acciones

Son las decisiones que el agente puede tomar en cada paso de tiempo. Las acciones son las acciones disponibles para el agente en un estado determinado y pueden ser discretas o continuas, dependiendo del problema.

Entorno

Es el contexto o el mundo en el que el agente opera. El entorno es dinámico y puede cambiar en respuesta a las acciones del agente. Proporciona retroalimentación al agente en forma de recompensas o penalizaciones según las acciones realizadas.

Agente

Es la entidad que toma decisiones y realiza acciones en un entorno específico. El agente utiliza información del estado actual del entorno para decidir qué acción tomar con el objetivo de maximizar las recompensas a largo plazo.

Aprendizaje por refuerzo

Implementación básica de un agente RL:

```
class AgenteRL:

      def __init__(self, acciones):

      self.acciones = acciones


      def seleccionar_accion(self, estado):

      # Ejemplo de selección aleatoria de acción

      return random.choice(self.acciones)

 # Uso del agente RL

acciones_posibles = ['izquierda', 'derecha', 'arriba', 'abajo']

agente = AgenteRL(acciones_posibles)

estado_actual = [0, 0]  # Estado inicial del entorno

accion_seleccionada = agente.seleccionar_accion(estado_actual)

print("Acción seleccionada por el agente:", accion_seleccionada)
```

Aprendizaje por refuerzo

```python
class EntornoRL:
      def __init__(self, estados):
      self.estados = estados


      def tomar_accion(self, accion):
      # Simulación de la transición de estado
      nuevo_estado = random.choice(self.estados)
      recompensa = random.randint(-10, 10)
      return nuevo_estado, recompensa
# Uso del entorno RL
estados_posibles = ['A', 'B', 'C', 'D']
entorno = EntornoRL(estados_posibles)
accion = 'izquierda'  # Acción seleccionada por el agente
nuevo_estado, recompensa = entorno.tomar_accion(accion)
print("Nuevo estado:", nuevo_estado)
print("Recompensa recibida:", recompensa)
```

Simulación de un entorno RL simple:

Aprendizaje por refuerzo

```
class QLearning:
      def __init__(self, estados, acciones, alpha=0.1, gamma=0.9,
epsilon=0.1):
      self.estados = estados
      self.acciones = acciones
      self.alpha = alpha
      self.gamma = gamma
      self.epsilon = epsilon
      self.q_table = {}

      def actualizar_q_table(self, estado_actual, accion, recompensa,
nuevo_estado):
      if estado_actual not in self.q_table:
            self.q_table[estado_actual] = {a: 0 for a in
self.acciones}
      if nuevo_estado not in self.q_table:
            self.q_table[nuevo_estado] = {a: 0 for a in self.acciones}

      q_actual = self.q_table[estado_actual][accion]
      max_q_nuevo_estado = max(self.q_table[nuevo_estado].values())
      nuevo_q_valor = q_actual + self.alpha * (recompensa + self.gamma
* max_q_nuevo_estado - q_actual)
      self.q_table[estado_actual][accion] = nuevo_q_valor
```

Implementación de un algoritmo de Q-Learning:

Aprendizaje por refuerzo

```
# Uso del algoritmo Q-Learning
estados = ['A', 'B', 'C']
acciones = ['izquierda', 'derecha']
q_learning = QLearning(estados, acciones)

# Simulación de una época de entrenamiento
estado_actual = 'A'
accion = 'izquierda'
nuevo_estado = 'B'
recompensa = 10
q_learning.actualizar_q_table(estado_actual, accion, recompensa,
nuevo_estado)

# Visualización de la tabla Q
print("Tabla Q actualizada:")
print(q_learning.q_table)
```

Implementación de un algoritmo de Q-Learning: