Mountain Car

Contenido

Configuración

Aplicamos la siguiente configuración:

```
# --- Configuración General ---
ENV_NAME = 'MountainCarContinuous-v0'
VIDEO_FOLDER = Path('videos') # Carpeta para vídeos (nombre diferente para no mezclar)
VIDEO_FOLDER.mkdir(parents=True, exist_ok=True) # Crear carpeta si no existe

# --- Parámetros del Training (Q-Learning) ---
EPISODES = 6000
MAX_STEPS = 900 # Máximos pasos por episodio
LR = 0.15 # Learning Rate (Alpha)
GAMMA = 0.99 # Discount Factor
EPSILON = 1.0 # Epsilon inicial (exploración)
EPSILON_DECAY = 0.9994 # Factor de decaimiento de epsilon
MIN_EPSILON = 0.05 # Epsilon mínimo

# --- Parámetros de Reward Shaping ---
STEP_PENALTY = 0.0 # Castigo por cada paso (además de la recompensa base del entorno)
STRONG_ACTION_REWARD = 0.02 # Premio por usar acciones de mayor magnitud
BOTTOM_POS = -0.5  # Posición del fondo del valle
MAX_PENALTY_BOTTOM = 1.0 # Castigo máximo por estar quieto en el fondo
POS_SENSITIVITY = 60.0 # Sensibilidad del castigo a la posición
VEL_SENSITIVITY = 1100.0 # Sensibilidad del castigo a la velocidad (cerca de 0)

# --- Discretización del Espacio de Estados ---
NUM_BINS = (10, 10) # Número de 'cajas' para (posición, velocidad)

# --- Discretización del Espacio de Acciones ---
# El entorno continuo acepta una fuerza entre -1.0 y 1.0.
# Discretizamos este espacio para poder usar Q-Learning tabular.
ACTIONS = [-0.7, -0.5, -0.3, 0.0, 0.3, 0.5, 0.7] # Acciones discretas
NUM_ACTIONS = len(ACTIONS)

# --- Configuración de Grabación de Vídeos ---
RECORD_START = 0 # Episodio desde el que empezar a considerar grabar
```

```
RECORD_END = EPISODES # Episodio hasta el que considerar grabar
RECORD_EVERY = 500 # Grabar el mejor episodio de cada X episodios

# --- Configuración de la Gráfica ---
PLOT_START = RECORD_START # Episodio inicial para la gráfica
PLOT_END = RECORD_END # Episodio final para la gráfica
AVG_WINDOW = 50 # Ventana para la media móvil en la gráfica
PLOT_NAME = "grafica_recompensas_montaña_notebook"
```

Funciones auxiliares

Creamos cinco funciones:

- preparar_entorno: Crear la instancia del entorno y calcular los límites para la discretización.

- crear_q_table: Inicializar la tabla Q con ceros.

- discretizar_estado: Convertir un estado continuo (posición, velocidad) en índices discretos para la tabla Q.

- elegir_accion: Implementar la política epsilon-greedy para seleccionar la siguiente acción.

- play_n_record: Ejecutar un episodio usando la política aprendida (sin exploración) y grabarlo en vídeo.

```
def preparar_entorno(env_name, num_bins):
    """Crea el entorno y calcula las divisiones para el estado."""
    env = gym.make(env_name, render_mode='rgb_array')
    low = env.observation_space.low
    high = env.observation_space.high
    high_adj = high + 1e-6 # Pequeño ajuste para incluir el límite superior en linspace

    state_bins = []
    for i in range(env.observation_space.shape[0]):
        bins = np.linspace(low[i], high_adj[i], num=num_bins[i] + 1)[1:-1]
        state_bins.append(bins)

    print("Límites del estado:", low, high)
    print("Divisiones para discretizar (bordes internos):")
    for i, b in enumerate(state_bins):
        print(f"  Dimensión {i}: {len(b)+1} cajas ({num_bins[i]} divisiones internas)")
    return env, state_bins

def crear_q_table(num_bins, num_actions):
    """Inicializa la tabla Q con ceros."""
    q_table_size = num_bins + (num_actions,)
    q_table = np.zeros(q_table_size)
    print(f"Tabla Q creada con tamaño: {q_table.shape}")
    return q_table
```

```
def discretizar_estado(estado, state_bins, num_bins_config):
    """Convierte un estado continuo a una tupla de índices (caja_pos, caja_vel)."""
    indices = []
    # Asegurarse de que el estado es un array numpy
    estado_np = np.asarray(estado)
    for i in range(len(estado_np)):
        # digitize nos dice en qué caja cae el valor
        idx = int(np.digitize(estado_np[i], state_bins[i]))
        # Asegurarse de que el índice está dentro de los límites [0, num_bins[i]-1]
        idx_clipped = np.clip(idx, 0, num_bins_config[i] - 1)
        indices.append(idx_clipped)
    return tuple(indices)

def elegir_accion(estado_idx, q_table, epsilon, num_actions):
    """Elige acción: epsilon-greedy."""
    if np.random.random() < epsilon:
        return np.random.randint(0, num_actions) # Explorar
    else:
        return np.argmax(q_table[estado_idx]) # Explotar

def play_n_record(env, q_table, state_bins, actions, num_actions, max_steps, video_folder,
ep, reward, num_bins_config):
    """Juega un episodio usando solo lo aprendido (sin explorar) y lo graba."""

    reward_str = f"reward_{reward:.2f}".replace('.', '_')
    filename = f"mountaincar-mejor-ep{ep}-{reward_str}"
    print(f"--- Grabando episodio {ep} (Recompensa: {reward:.2f})... ---")

    record_env = None # Inicializar por si falla la creación del wrapper
    try:
        # Usamos lambda e_idx: e_idx == 0 para grabar solo el primer episodio que se le pasa
        record_env = RecordVideo(env, str(video_folder), episode_trigger=lambda e_idx: e_idx
== 0, name_prefix=filename)

        # Resetear el entorno envuelto
        obs, info = record_env.reset()

        # Asegúrate de pasar num_bins_config si tu función lo requiere
        estado = discretizar_estado(obs, state_bins, num_bins_config)
        terminado = False
        truncado = False # Para gym >= 0.26
        pasos = 0

        # Bucle del episodio (solo explotación)
        for t in range(max_steps):
            accion_idx = np.argmax(q_table[estado]) # Elegir la mejor acción
            accion_continua = np.array([actions[accion_idx]], dtype=np.float32)

            try:
                # La llamada a step() es la que internamente hace que se capture el frame
                obs_siguiente, rec, terminado, truncado, info = record_env.step(accion_continua)
            except Exception as e:
```

```
            print(f"Error en step grabando ep {ep}, paso {t}: {e}")
            terminado = True # Forzar fin si hay error en step
            break # Salir del bucle de pasos

        # Asegúrate de pasar num_bins_config si tu función lo requiere
        estado_siguiente = discretizar_estado(obs_siguiente, state_bins, num_bins_config)
        estado = estado_siguiente
        pasos = t + 1

        if terminado or truncado:
            break

    print(f"--- Grabación ep {ep} terminada ({pasos} pasos). ---")

except Exception as e:
    # Captura errores durante la inicialización o el bucle de grabación
    print(f"Error al preparar o grabar el vídeo del episodio {ep}: {e}")
    # traceback.print_exc() # Puedes descomentar para obtener un traceback más detallado

finally:
    # Importante cerrar el entorno de grabación para que guarde el vídeo correctamente
    if record_env is not None: # Comprobar si se llegó a crear el wrapper
        try:
            record_env.close()
            print(f"Entorno de grabación para ep {ep} cerrado.")
        except Exception as e:
            print(f"Error cerrando RecordVideo para ep {ep}: {e}")
```

Función principal de entrenamiento

Esta función contiene el bucle principal de Q-Learning. Itera sobre los episodios, y en cada episodio, itera sobre los pasos. En cada paso:

1. Elige una acción (epsilon-greedy).

2. Ejecuta la acción en el entorno.

3. Observa el nuevo estado y la recompensa.

4. Aplica *reward shaping* (modifica la recompensa para guiar mejor el aprendizaje).

5. Actualiza el valor Q correspondiente en la tabla usando la fórmula de Q-Learning.

6. Actualiza el estado.

7. Gestiona la grabación del mejor episodio cada cierto número de episodios.

8. Reduce epsilon.

```
def entrenar(env, q_table, state_bins, actions, num_actions, episodes, max_steps,
          lr, gamma, start_epsilon, eps_decay, min_epsilon,
          step_penalty, strong_action_reward,
          bottom_pos, max_penalty_bottom, pos_sensitivity, vel_sensitivity,
          video_folder, record_start, record_end, record_every, num_bins_config): # Añadido
num_bins_config
   """Bucle principal de Q-learning con grabación periódica del mejor episodio."""

   todas_recompensas = []
   epsilon = start_epsilon
   mejor_recompensa_chunk = -np.inf
   mejor_episodio_chunk = -1
   episodios_en_chunk = 0

   # Crear carpeta de vídeos si no existe (mejor hacerlo fuera, antes de llamar a entrenar, o al
inicio del script)
   # video_folder.mkdir(parents=True, exist_ok=True)

   print(f"\n--- Empezando Entrenamiento ({episodes} episodios) ---")
   # ... (más prints informativos si quieres) ...

   for ep in range(episodes):
      ep_num = ep + 1

      try:
         obs, info = env.reset()
      except Exception as reset_e:
          print(f"Error reseteando entorno en episodio {ep_num}: {reset_e}")
          traceback.print_exc()
          continue # Saltar episodio

      # Asegúrate de pasar num_bins_config a discretizar_estado
      estado = discretizar_estado(obs, state_bins, num_bins_config)
      recompensa_total_episodio = 0.0
      terminado = False
      truncado = False
      pasos = 0

      # Bucle dentro de un episodio (pasos)
      for t in range(max_steps):
         accion_idx = elegir_accion(estado, q_table, epsilon, num_actions)
         accion_continua = np.array([actions[accion_idx]], dtype=np.float32)

         try:
            obs_siguiente, recompensa_base, terminado, truncado, info =
env.step(accion_continua)
            recompensa = float(recompensa_base)
```

```
            estado_siguiente_continuo = obs_siguiente
            # Asegúrate de pasar num_bins_config a discretizar_estado
            estado_siguiente = discretizar_estado(obs_siguiente, state_bins, num_bins_config)

            # --- Reward Shaping ---
            accion_valor = actions[accion_idx]
            posicion = estado_siguiente_continuo[0]
            velocidad = estado_siguiente_continuo[1]
            recompensa += strong_action_reward * abs(accion_valor)
            if not (terminado or truncado):
                recompensa -= step_penalty
            factor_pos = math.exp(-pos_sensitivity * (posicion - bottom_pos)**2)
            factor_vel = math.exp(-vel_sensitivity * velocidad**2)
            castigo_quieto_abajo = max_penalty_bottom * factor_pos * factor_vel
            recompensa -= castigo_quieto_abajo

        except Exception as step_e:
            print(f"Error en step episodio {ep_num}, paso {t}: {step_e}")
            traceback.print_exc()
            terminado = True
            recompensa = 0.0
            estado_siguiente = estado # Mantener estado si falló

        # --- Actualización Q-Learning ---
        recompensa_total_episodio += recompensa

        valor_antiguo = q_table[estado + (accion_idx,)]
        valor_max_siguiente = np.max(q_table[estado_siguiente]) if not (terminado or
truncado) else 0.0
        objetivo = recompensa + gamma * valor_max_siguiente
        valor_nuevo = valor_antiguo + lr * (objetivo - valor_antiguo)
        q_table[estado + (accion_idx,)] = valor_nuevo

        estado = estado_siguiente
        pasos = t + 1

        if terminado or truncado:
            break

    todas_recompensas.append(recompensa_total_episodio)

    # --- Lógica de Grabación (CORREGIDA INDENTACIÓN) ---
    esta_en_rango_grabacion = record_start <= ep_num <= record_end
    if esta_en_rango_grabacion:
        episodios_en_chunk += 1
        if recompensa_total_episodio >= mejor_recompensa_chunk:
            mejor_recompensa_chunk = recompensa_total_episodio
            mejor_episodio_chunk = ep_num

        # Comprobar si grabar al final del chunk o del entrenamiento
        fin_chunk = (episodios_en_chunk >= record_every)
        ultimo_ep_rango = (ep_num == record_end)
```

```
        ultimo_ep_total = (ep_num == episodes)

        # --- ESTE BLOQUE AHORA ESTÁ DENTRO DEL if esta_en_rango_grabacion: ---
        if (fin_chunk or ultimo_ep_rango or ultimo_ep_total) and mejor_episodio_chunk != -
1:
            if record_start <= mejor_episodio_chunk <= record_end:
              # Crear un entorno nuevo para grabar es más seguro
              try:
                 # Asegúrate que ENV_NAME está definido globalmente o pásalo como
argumento
                 env_grabacion = gym.make(ENV_NAME, render_mode='rgb_array')
                 # Asegúrate que play_n_record acepta num_bins_config
                 play_n_record(
                    env=env_grabacion,
                    q_table=q_table,
                    state_bins=state_bins,
                    actions=actions,
                    num_actions=num_actions,
                    max_steps=max_steps,
                    video_folder=video_folder,
                    ep=mejor_episodio_chunk,
                    reward=mejor_recompensa_chunk,
                    num_bins_config=num_bins_config # Pasar num_bins_config
                 )
                 env_grabacion.close()
              except Exception as record_setup_e:
                  print(f"Error al configurar/ejecutar grabación para ep {mejor_episodio_chunk}:
{record_setup_e}")
            # Resetear para el siguiente chunk (también indentado aquí)
            mejor_recompensa_chunk = -np.inf
            mejor_episodio_chunk = -1
            episodios_en_chunk = 0

      # Decaimiento de Epsilon (fuera del if de grabación)
      epsilon = max(min_epsilon, epsilon * eps_decay)

      # Imprimir Progreso (fuera del if de grabación)
      if ep_num % 100 == 0 or ep_num == episodes:
         avg_reward_100 = np.mean(todas_recompensas[-100:]) if len(todas_recompensas)
>= 100 else np.mean(todas_recompensas) if todas_recompensas else 0.0
         print(f"Episodio {ep_num}/{episodes} | "
             f"Recompensa: {recompensa_total_episodio:.2f} | "
             f"Media (últ 100): {avg_reward_100:.2f} | "
             f"Pasos: {pasos} | "
             f"Epsilon: {epsilon:.3f}")

   return todas_recompensas
```

Función para dibujar gráfica

Esta función toma la lista de recompensas por episodio y genera una gráfica mostrando la recompensa de cada episodio y una media móvil para visualizar la tendencia del aprendizaje. Guarda la gráfica como un archivo PNG.

```
def dibujar_grafica(recompensas, start_ep, end_ep, avg_window, filename):
    """Genera y guarda una gráfica de las recompensas por episodio y media móvil."""
    num_episodios = len(recompensas)
    if num_episodios == 0:
        print("No hay recompensas para dibujar.")
        return

    start_idx = max(0, start_ep - 1)
    end_idx = min(num_episodios, end_ep)

    if start_idx >= end_idx:
        print(f"\nRango de episodios para gráfica ({start_ep}-{end_ep}) no válido o sin datos.")
        return

    recompensas_sub = recompensas[start_idx:end_idx]
    episodios_sub = list(range(start_idx + 1, end_idx + 1))

    if not episodios_sub:
         print(f"\nNo hay episodios en el rango ({start_ep}-{end_ep}) para la gráfica.")
         return

    plt.figure(figsize=(12, 6))
    plt.plot(episodios_sub, recompensas_sub, label='Recompensa Episodio', alpha=0.6,
linewidth=1)

    # Calcular media móvil solo sobre el subconjunto visible
    if len(recompensas_sub) >= avg_window:
        media_movil_sub = np.convolve(recompensas_sub, np.ones(avg_window)/avg_window,
mode='valid')
        # Ajustar los episodios para la media móvil (empiezan después de la ventana inicial)
        episodios_media_sub = list(range(start_idx + avg_window, end_idx + 1))
        plt.plot(episodios_media_sub, media_movil_sub,
                 label=f'Media Móvil ({avg_window} ep)', color='red', linewidth=2)
    elif len(recompensas_sub) > 0:
         # Si no hay suficientes datos para la ventana completa, calcular media simple
         media_simple = np.mean(recompensas_sub)
         plt.axhline(media_simple, color='orange', linestyle='--', label=f'Media Total Rango
({media_simple:.2f})')
```

```
        print(f"No hay suficientes datos ({len(recompensas_sub)}) para media móvil de {avg_window} episodios en el rango.")

    plt.xlabel('Episodio')
    plt.ylabel('Recompensa Total Acumulada')
    plt.title(f'Recompensas Q-Learning ({ENV_NAME} - Episodios {start_idx + 1} a {end_idx})')
    plt.legend()
    plt.grid(True, linestyle=':')
    plt.tight_layout()

    output_file = f"{filename}_ep_{start_idx + 1}_to_{end_idx}.png"
    try:
        plt.savefig(output_file)
        print(f"\nGráfica guardada como: '{output_file}'")
        # plt.show() # Mostrar la gráfica en el notebook
    except Exception as e:
        print(f"\nError guardando/mostrando la gráfica '{output_file}': {e}")
    finally:
        plt.close() # Cerrar la figura
```

Ejecución principal

Aquí es donde orquestamos todo el proceso:

1. Preparamos el entorno y obtenemos los `state_bins`.

2. Creamos la tabla Q.

3. Llamamos a la función `entrenar` para iniciar el aprendizaje.

4. Si el entrenamiento produce resultados, llamamos a `dibujar_grafica`.

5. Nos aseguramos de cerrar el entorno al final.

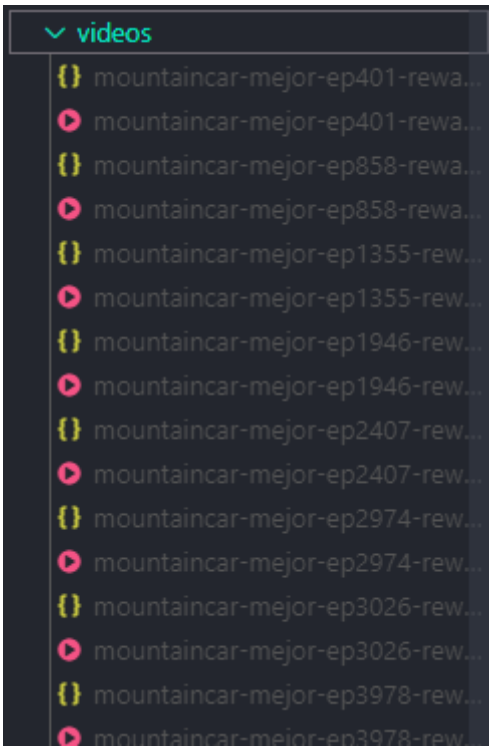
```
                                                try:
    # 1. Preparar entorno y discretización
    print("\n1. Preparando entorno...")
    main_env, state_bins = preparar_entorno(ENV_NAME, NUM_BINS)

    # 2. Crear tabla Q
    print("\n2. Creando tabla Q...")
    q_tabla_inicial = crear_q_table(NUM_BINS, NUM_ACTIONS)

    # 3. Entrenar al agente
    print("\n3. Iniciando entrenamiento...")
    lista_recompensas = entrenar(
        env=main_env,
        q_table=q_tabla_inicial, # Pasamos la tabla inicial
        state_bins=state_bins,
        actions=ACTIONS,
```

```
        num_actions=NUM_ACTIONS,
        episodes=EPISODES,
        max_steps=MAX_STEPS,
        lr=LR,
        gamma=GAMMA,
        start_epsilon=EPSILON,
        eps_decay=EPSILON_DECAY,
        min_epsilon=MIN_EPSILON,
        step_penalty=STEP_PENALTY,
        strong_action_reward=STRONG_ACTION_REWARD,
        bottom_pos=BOTTOM_POS,
        max_penalty_bottom=MAX_PENALTY_BOTTOM,
        pos_sensitivity=POS_SENSITIVITY,
        vel_sensitivity=VEL_SENSITIVITY,
        video_folder=VIDEO_FOLDER,
        record_start=RECORD_START,
        record_end=RECORD_END,
        record_every=RECORD_EVERY,
        num_bins_config=NUM_BINS
    )
    q_tabla_final = q_tabla_inicial # La tabla se modifica in-place
    print("\n--- Entrenamiento Terminado ---")

    # 4. Dibujar gráfica si hay recompensas
    if lista_recompensas:
        print("\n4. Creando Gráfica de Recompensas...")
        dibujar_grafica(
            recompensas=lista_recompensas,
            start_ep=PLOT_START,
            end_ep=PLOT_END,
            avg_window=AVG_WINDOW,
            filename=PLOT_NAME
        )
    else:
        print("\n--- No se generaron recompensas para graficar --- ")

except Exception as e:
    print(f"\n--- ¡ERROR FATAL! El proceso falló: {e} ---")
    traceback.print_exc()
finally:
    # 5. Cerrar el entorno principal
    if main_env is not None:
        try:
            main_env.close()
            print("\nEntorno principal cerrado.")
        except Exception as close_e:
            print(f"Error al cerrar el entorno principal: {close_e}")

    print(f"\nVídeos guardados en: '{VIDEO_FOLDER.resolve()}'")
```

Podemos ver los resultados que da:

```
1. Preparando entorno...
Límites del estado: [-1.2  -0.07] [0.6  0.07]
Divisiones para discretizar (bordes internos):
  Dimensión 0: 10 cajas (10 divisiones internas)
  Dimensión 1: 10 cajas (10 divisiones internas)

2. Creando tabla Q...
Tabla Q creada con tamaño: (10, 10, 7)

3. Iniciando entrenamiento...

--- Empezando Entrenamiento (6000 episodios) ---
Episodio 100/6000 | Recompensa: -566.37 | Media (últ 100): -404.28 | Pasos: 900 | Epsilon: 0.942
Episodio 200/6000 | Recompensa: -127.94 | Media (últ 100): -337.26 | Pasos: 900 | Epsilon: 0.887
Episodio 300/6000 | Recompensa: -375.15 | Media (últ 100): -285.60 | Pasos: 900 | Epsilon: 0.835
Episodio 400/6000 | Recompensa: -154.50 | Media (últ 100): -210.03 | Pasos: 900 | Epsilon: 0.787
--- Grabando episodio 401 (Recompensa: 10.34)... ---
C:\Users\isard\AppData\Roaming\Python\Python311\site-packages\gym\wrappers\record_video.py:75: UserWarning: WARN: Overwriting exist
  logger.warn(
MoviePy - Building video c:\Users\isard\Desktop\bigData\1CEIABDTA - 7RO\Programación de Inteligencia Artificial\TAREA 9 - DQN MOUNT
MoviePy - Writing video c:\Users\isard\Desktop\bigData\1CEIABDTA - 7RO\Programación de Inteligencia Artificial\TAREA 9 - DQN MOUNTA


MoviePy - Done !
```

Los mejores episodios se guardan en la carpeta “videos”:



Se adjuntan dos videos obtenidos con esta red neuronal (mountaincar-episode-301-episode-0.mp4 es de una versión más temprana del mismo script).