

Aprendizaje por Refuerzo

Framework Aplicado a robots en RealTime Battle

José Luis Marina
Máster Investigación Informática
Universidad Complutense de Madrid



Obra Creative Commons

Aprendizaje por refuerzo

Problema genérico a resolver:

“Cómo un agente autónomo que siente y reacciona con su entorno, puede aprender a elegir acciones óptimas para la consecución de sus objetivos.”

Machine Learning - Tom Mitchell

Introducción

Aprendizaje por refuerzo

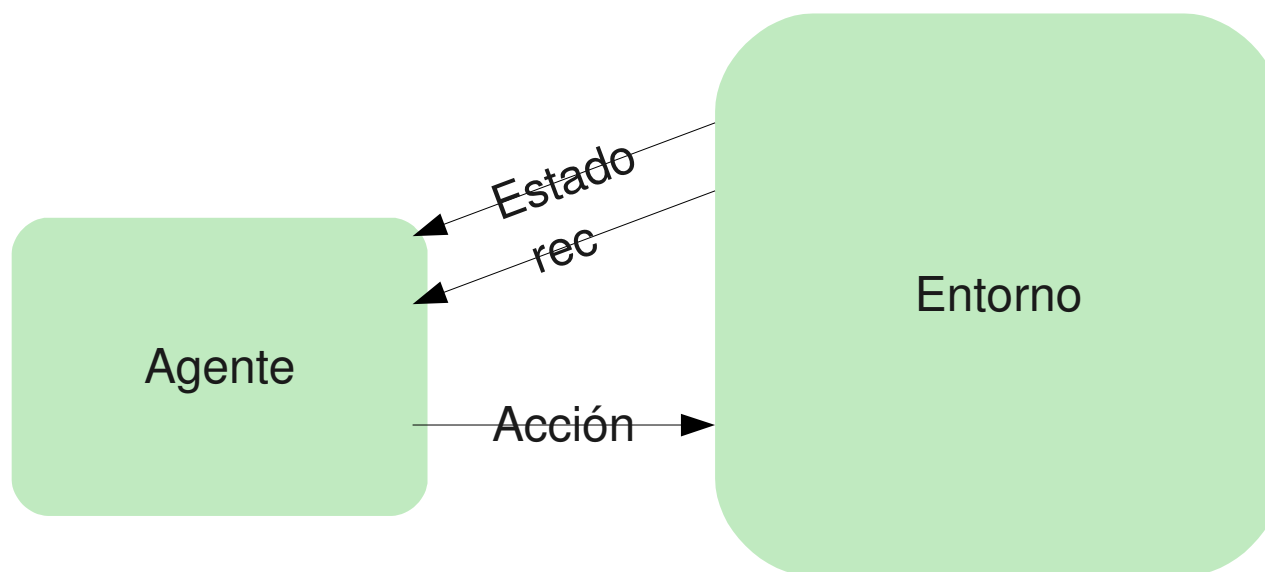
- ✓ Agentes con objetivos concretos y medibles.
- ✓ Conjunto de acciones: $A \{a_0, a_1, \dots\}$
- ✓ Conjunto de sensores del entorno: $E \{e_0, e_1, \dots\}$
- ✓ Conjunto de Estados del entorno: $S \{s_0, s_1, \dots\}$
- ✓ Para cada Acción: Recompensa (o castigo)
- ✓ El agente utiliza la información de los sensores para elegir la acción más adecuada sobre el conjunto de acciones.

Introducción

Aprendizaje por refuerzo

Elegir **política de control** o acciones que maximizen:

$$r_0 + \gamma r_1 + \gamma^2 r_2 + \dots \text{ donde } 0 \leq \gamma < 1$$



Introducción

Aprendizaje por refuerzo

✓ **Proceso de Decisión de Markov (MPD)**

Los estados son observables (esto es mucho decir)

Conjunto finito de estados (ok)

Acciones deterministas

✓ **Todo es más fácil**

Tenemos **S** y **A**, conjuntos finitos y discretos.

En cada t_i observamos un s_i y elegimos una a_i .

$$r_t = r(s_t, a_t) \quad s_{t+1} = \delta(s_t, a_t)$$

El agente no tiene porqué saber ni **r** ni **δ**

Además **r** y **δ** dependen sólo del estado actual.

Introducción

Aprendizaje por refuerzo

✓ Q Learning

$Q(s, a)$ me da el valor de la recompensa de aplicar **a** en el estado **s** más el valor si siguiera la política óptima desde ahora (descontando por γ)

$$Q(s, a) = r(s, a) + \gamma \max_{a'} Q(\delta(s, a), a')$$

Si Q^* es la estimación de Q por el agente.

$$Q^*(s, a) \leftarrow r + \gamma \max_{a'} Q(s', a')$$

Introducción

Aprendizaje por refuerzo

✓ $Q^*(s, a)$

Inicialmente la llenamos con valores al azar o cero.

En cada instante de tiempo:

Observamos el estado s

Elegimos una acción a y la ejecutamos.

Obtenemos la recompensa r y el nuevo estado s'

Actualizamos la entrada de $Q^*(s, a)$

$$Q^*(s, a) \leftarrow r + \gamma \max_{a'} Q(s', a')$$

Si el sistema es un PDM, Q^* converge a Q con el tiempo.

Introducción

Aprendizaje por refuerzo

✓ Fórmula de Sutton y Barto para calcular los valores de la tabla Q:

$$Q(s, a) \leftarrow Q(s, a) + \alpha (R + \gamma \max_{a'} Q(s', a') - Q(s, a))$$

α -> "Step size" (diferencias r y r')

R -> Recompensa

γ -> Factor de descuento en recompensas

Real Time Battle

Entorno en el que varios robots pueden luchar en diferentes modalidades con el objetivo de vencer a los demás.

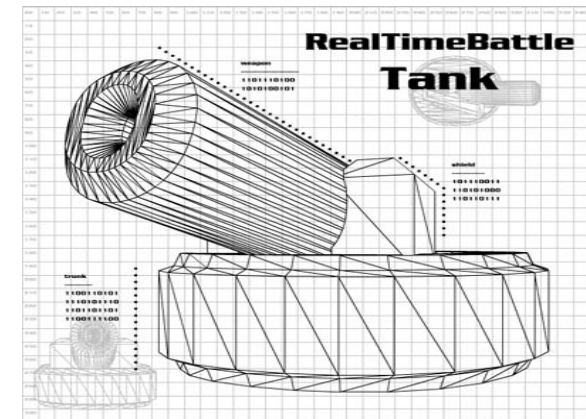
- **Desarrollo en Tiempo Real:** Robots son procesos hijos de RTB.
- **Cualquier Lenguaje de Programación:** Comunicaciones a través de la salida y la entrada estándar.
- **Protocolo de mensajes sencillo.**
- **Hasta 120 robots** si muchas exigencias de HW.
- **Complejidad de entorno e interacciones:** Leyes físicas, gravedad, choques, aceleraciones, etc.

DEMO

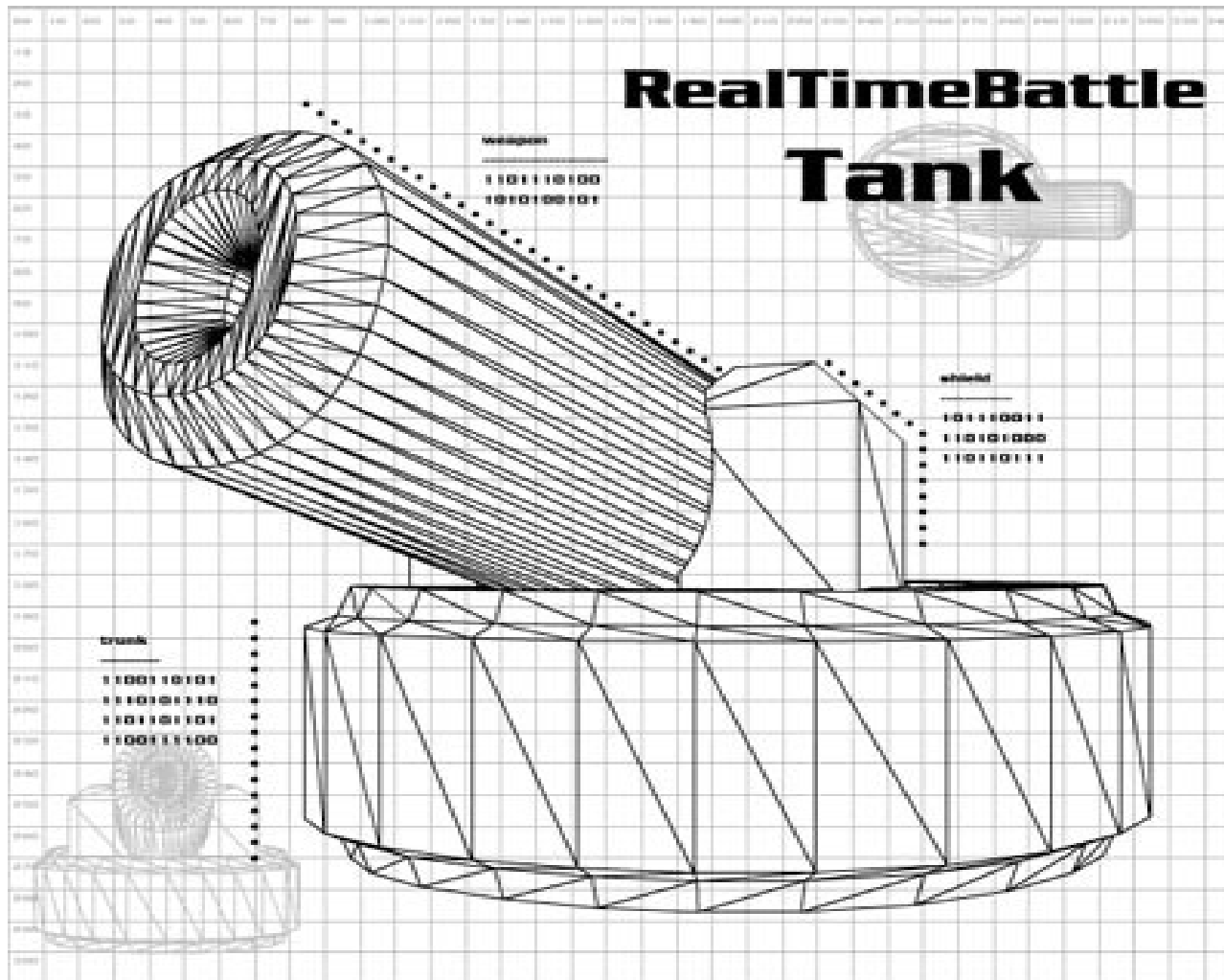
Real Time Battle

Los Robots:

- **Vehículos con ruedas:** Rozamiento en giros, velocidad lineal y angular, pueden derrapar...
 - Acelerar | Frenar | Rotar
- **Radar:** Principal sensor que me avisa de tipo de objeto visto y a qué distancia.
 - Rotar | Barrer entre dos ángulos.
- **Cañón:** Disparos con la energía indicada si tenemos suficiente reserva. Se suman velocidades (vectores)
 - Rotar | Disparar con energía E.



Real Time Battle



Radar :

- Objeto y distancia
- Robot.
- Pared.
- Mina.
- Galleta
- Bala

Otros Sensores:

- Velocidad
- Energía: $si=0$ dead
- Robots restantes.
- Colisiones (objeto)
- Tiempo de juego.
- Impactos

Trabajo

Objetivos :

- **Aplicación de Aprendizaje por refuerzo a entornos complejos.**
 - Comprobar empíricamente:
 - La adaptación a la resolución del problema
 - Completitud de los Estados.
- **Hacer una implementación genérica del algoritmo de función Q basado en la fórmula de Sutton y Barton.**
 - Probarlo en RTB.
 - Prepararlo para otros entornos



Trabajo

Fases :

- **Análisis y estudio del entorno: RTB.**
 - Selección de Sensores.
 - Selección de Acciones.
- **Desarrollo de Clase de uso genérico Q-value**
 - Diseño y desarrollo.
 - Pruebas generales.
- **Resultados en RTB.**
 - Pruebas con variación de parámetros.
 - Conclusiones.



Trabajo

Sensores, estados y Acciones

NUMROBOTS = 0, **8**
 SEE_ROBOT = 1,
 SEE_WALL = 2,
 SEE_COOKIE = 3,
 SEE_MINE = 4,
 SEE_BULLET = 5,
 MY_ENERGY = 6,
 UNDER_FIRE = 7,

SEE_COOKIE	-	SEE_WALL	-	SEE_ROBOT
No		Lejos		Cerca
NUM_ROBOTS		ENERGY		UNDER_FIRE
Poco		Medio		Mucho



x
 LEVEL_1 = 0 **3**
 LEVEL_2 = 1
 LEVEL_3 = 2

$n=3^8 (6561)$
 $s_1 \quad s_2 \quad \dots \quad s_n$

ACELERAR
 GIRAR ROBOT
 GIRAR RADAR Y CAÑÓN
 FRENAR
 DISPARAR

Acciones Básicas

7
 PATROL
 FIRE_AROUND
 BE_QUIET
 GO_FOR_COOKIES
 FIRE_CRAZY
 RAMBOW
 STOP

Acciones Elaboradas

$Q(s, a)$
 $3^8 \times 7$
45927
 elementos

Trabajo

Clase Qtable y Rebote



MAIN:

```
RTB_Rebote    Rebote ("Nombre", "Color");  
Rebote.run();
```

Rebote.run()

```
RTB_QTable Qt;  
Qt.open (num_actions , num_states , init_val  
         "qtable_file", explore_rate , learning);  
  
While (!end_game)  
    get_sensor_values();  
    calculate_state();  
    reward = energy + 400 / (robots_left *  
                             robots_left);  
    action = Qt.next_action(state, reward);  
    execute_action(action);
```

Resultados

Respecto a la idoneidad en la tarea RTB

"Destruir a los demás Robots"

- 100 Torneos de 100 juegos cada uno.
- Con 20 y 3 robots y dos "arenas" (una sencilla)
- Competidor: Seek_and_destroy y Rotate_Fire
- Aprender en un torneo y jugar en otros:
 - Rebote "Aprendiendo"
 - Rebote "Aprendido"
 - Rebote "Acción al Azar"
 - Rebote Monoacción (Rambow y Fire_Crazy)
 - Variaciones de Gamma y Alpha.

Introducción

Cerr Guardar Estadistic: Tot Total de la Secuenc Jueç Rot

Gran TOTAL

Nombre	Posición	Puntos	Juegos	Tiempo de supervivencia
Seek and Destroy(1	2.08	100	72.65
Seek and Destroy(2	2.05	100	72.57
Rebote	3	1.30	100	50.31

Al Azar

Cerr Guardar Estadistic: Tot Total de la Secuenc Jueç Rot

Gran TOTAL

Nombre	Posición	Puntos	Juegos	Tiempo de supervivencia
Seek and Destroy(1	2.00	100	73.40
Seek and Destroy(2	1.90	100	74.00
Rebote	3	1.40	100	59.85

Gamma = 1.0
No Learning

Cerr Guardar Estadistic: Tot Total de la Secuenc Jueç Rot

Gran TOTAL

Nombre	Posición	Puntos	Juegos	Tiempo de supervivencia
Seek and Destroy(1	2.04	100	67.68
Seek and Destroy(2	1.94	100	65.40
Rebote	3	1.61	100	53.27

Gamma = 1.0
Explore = 10%
Learning

Cerr Guardar Estadistic: Tot Total de la Secuenc Jueç Rot

Gran TOTAL

Nombre	Posición	Puntos	Juegos	Tiempo de supervivencia
Seek and Destroy(1	1.97	100	62.93
Seek and Destroy(2	1.83	100	61.06
Rebote	3	1.78	100	59.71

Gamma = 0.8
Explore = 10%

Datos de Resultados

Nombre	Posición	Puntos	Juegos	Tiempo de supervivencia
ReboteRAMBOW	1	2.17	90	57.33
Seek and Destroy(2	1.79	90	56.26
Seek and Destroy(3	1.74	90	54.79

Nombre	Posición	Puntos	Juegos	Tiempo de supervivencia
Seek and Destroy(1	2.14	65	72.50
Seek and Destroy(2	2.03	65	71.49
ReboteCRAZY	3	1.35	65	57.06

- Rebote **No** gana a “Seek_and_Destroy”.
- Es mejor Rebote que el Robot con acciones al Azar.
- Rebote es peor que Rebote_RAMBOW pero mejor que Rebote_CRAZY.
- Mejor valor de Gamma, Alpha y Explore => <0,8 0,2 10%>
- Muchos estados por explorar (sin haberse actualizado)

Datos de Resultados

Experimento	Robot	Posición	Puntos	Tiempo Vivo
Acción al Azar	S&D	1	2,08	72,65
	S & D	2	2,05	72,57
	Rebote	3	1,30	50,31
Acción Rambow	Rebote	1	2,17	57,33
	S & D	2	1,79	56,26,
	S & D	3	1,74	54,79
Acción "Fire Crazy"	S & D	1	2,14	72,50
	S & D	2	2,03	71,49
	Rebote	3	1,35	57,05
Adaptativo	S & D	1	2,04	67,68
	S & D	2	1,94	65,40
	Rebote	3	1,61	53,27
Q-table (sin Learning)	S & D	1	2,00	67,68
	S & D	2	1,90	65,40
	Rebote	3	1,40	59,85
Adaptativo (Gamma = 0,8)	S & D	1	1,97	62,93
	S & D	2	1,83	61,06
	Rebote	3	1,78	59,71

Conclusiones sobre los resultados

¡Más tiempo, más pruebas, más variaciones!

Elección de Acciones:

Compromiso comportamiento simple / elaborado.

Elección de Sensores:

Ok - Trazas de cambio de estado de acuerdo con la realidad.

Pueden ajustarse los límites entre valores.

Pruebas y experimentos:

Mayores tiempos en los torneos.

Incorporar mayores variaciones de los parámetros:

Motor de Qtable: Fácil de usar e incorporar.

Acciones Futuras

RealTime Battle:

- Comparación con acciones simples (derecha, izda, dispara, frena, etc)
- Baterías de pruebas variando parámetros.
- Presentar Rebote-II a concursos de RTB.

Motor Qtable:

- Utilizar en otro Entornos:
 - Operación automática de sistemas monitorizados.
 - Otros torneos de Bots y otros Juegos.
- Reducción de Estados en función de la Matriz Q-table:
 - Estados que no aportan información.
 - Estados que son combinación de otros.
 - Estados inalcanzable.
 - Utilizar la Q-table para entender mejor el Entorno.

Principales Referencias

Machine Learning: C13 – Reinforcement Learning

Tom Mitchell - 1997

RETALIATE: Learning Winning Policies in First-Person Shooter Games

Megan Smith, Stephen Lee-Urban, Héctor Muñoz-Avila – 2007

Learning to be a Bot: Reinforcement Learning in Shooter Games.

Michelle McPartland and Marcus Gallagher – 2008

Sutton, R. S. & Barto, A. G. Reinforcement Learning: An Introduction,

MIT Press, Cambridge, MA - 1998.

Recognizing the Enemy: Combining RL with Strategy Selection using CBR.

B. auslander, S. Lee-Urban, Chad Hogg and H. Muñoz - 2008

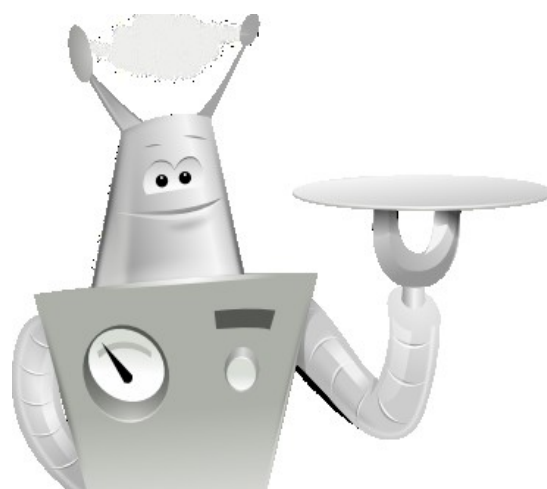
Real Time Battle: Web page and documentation.

<http://realtimebattle.sourceforge.net/>

Imágenes de: <http://www.sxc.hu>

Fin

¿Preguntas?



joseluis.marina@peopleware.es

Aprendizaje por refuerzo

✓ Q Learning

Las acciones deben maximizar:

γ peso de las recompensas de acciones futuras

$$V^\pi(s_t) = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots = \sum_{i=0}^{\infty} \gamma^i r_{t+i}$$

Queremos un algoritmo para calcular:

$$\pi^*(s) = \operatorname{argmax} V^\pi(s), \forall s$$