Aprendizaje Reforzado

Integrantes:
Huertas Villegas Cesar
Urias Vega Juan Daniel
Zavala Roman Irvin Eduardo

Inteligencia Artificial
Grupo 561

Aprendizaje Reforzado

El aprendizaje por refuerzo o aprendizaje reforzado es un área del aprendizaje automático inspirada en la psicología conductista, cuya ocupación es determinar qué acciones debe escoger un agente de software en un entorno dado con el fin de maximizar alguna noción de "recompensa" o premio acumulado.

- Aprendizaje reforzado por modelos
- Aprendizaje reforzado libre de modelos.

Recompensas

Las recompensas son entradas sensoriales que se le proporciona al agente por el entorno que lo rodea, estas pueden ser buenas o malas.

Donde una recompensa buena es cuando se consigue un objetivo determinado, y una recompensa mala es cuando se consigue un objetivo pero este puede causar consecuencias negativas al agente.

Ejemplo:

Si un agente quiere aprender a volar un helicóptero una recompensa buena sería aterrizar una vez despegue y donde recompensas negativas puede ser estrellarse, tambalearse o desviarse de la ruta.

Procesos de decisiones de Markov

Los Procesos de decisiones de Markov son una solución a los problemas del Aprendizaje Reforzado, el cual describe formalmente el ambiente donde se encuentra un agente, donde el ambiente es completamente observable.

Propiedad de Markov

La propiedad de Markov nos muestra que el futuro es independiente del pasado, dado el presente, lo cual se expresa en la siguiente fórmula:

$$\mathbb{P}\left[S_{t+1} \middle| S_t\right] = \mathbb{P}\left[S_{t+1} \middle| S_1, \ldots, S_t\right]$$

El cual nos dice que el estado actual (St) contiene toda la información relevante de los estados anteriores

Matriz de Transición de Estados

Esta se encarga de mostrar la probabilidad de transición de un estado S a un estado S’

Procesos de recompensas de Markov

Los procesos de recompensa de Markov se pueden ver como procesos de Markov normales con valores que juzgan que tan positivo es estar en un estado.

$$R_s = \mathbb{E}\left[R_{t+1} \middle| S_t = s\right]$$

- S, es una lista de estados a la cual puede pertenecer.
- P, es una matriz de transición.
- R, es la recompensa inmediata del estado donde nos encontrados.
- Y, es el valor de descuento que va entre 0 y 1

Retorno

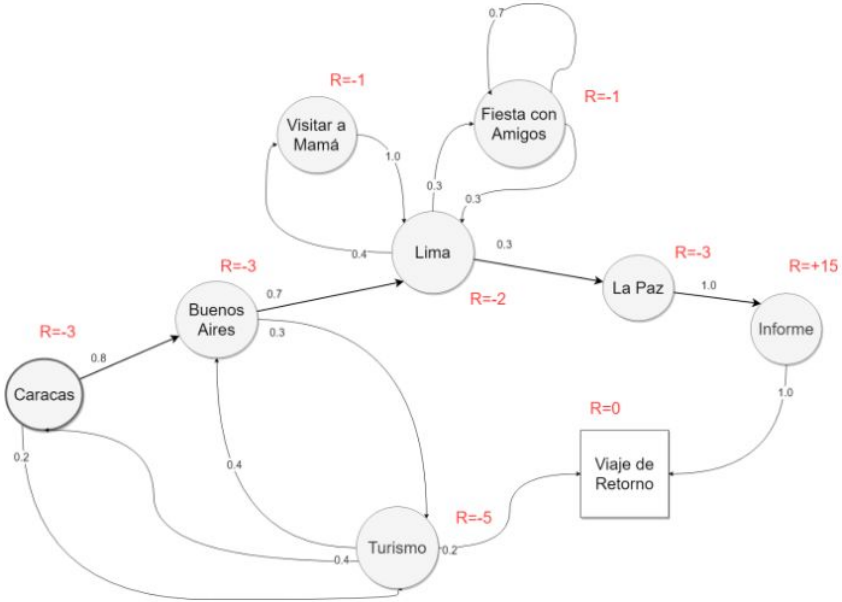
Es la recompensa total multiplicado por el descuento en cada paso del tiempo.

$$G_t = R_{t+1} + \gamma R_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Ejemplo

	Ruta
1	CARACAS, TURISMO, VIAJE DE RETORNO
2	CARACAS, BUENOS AIRES, LIMA, FIESTA CON AMIGOS, FIESTA CON AMIGOS, LIMA, LA PAZ, INFORME, VIAJE DE RETORNO
3	CARACAS, BUENOS AIRES, LIMA, VISITAR A MAMÁ, LIMA, LA PAZ, INFORME, VIAJE DE RETORNO
4	CARACAS, BUENOS AIRES, LIMA, LA PAZ, INFORME, VIAJE DE RETORNO

Retorno
v1 = −3 −5×1/3 +0*1/9 =−4.66
v2 = −3 −3×1/3 -2*1/9 -1*1/27-1*1/81-2*1/243*-3*1/729+15*1/2187+0*1/6561= -4.26
v3 = −3 −3×1/3 -2*1/9 -1*1/27-2*1/81-3*1/243*+15*1/729+15*1/2187= −4.27
v4 = −3 −3×1/3 -2*1/9 -3*1/27+15*1/81+0*1/243=−4.14



Aprendizaje reforzado pasivo

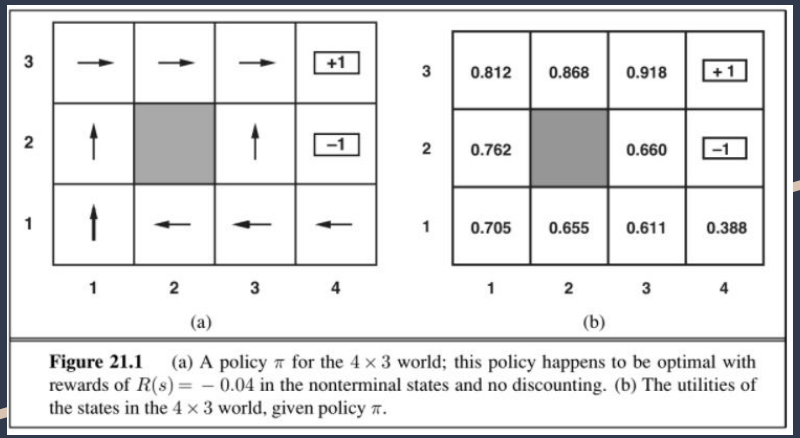
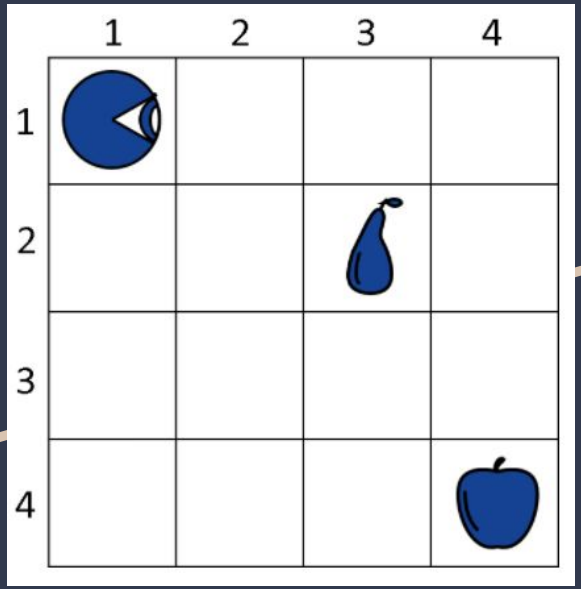


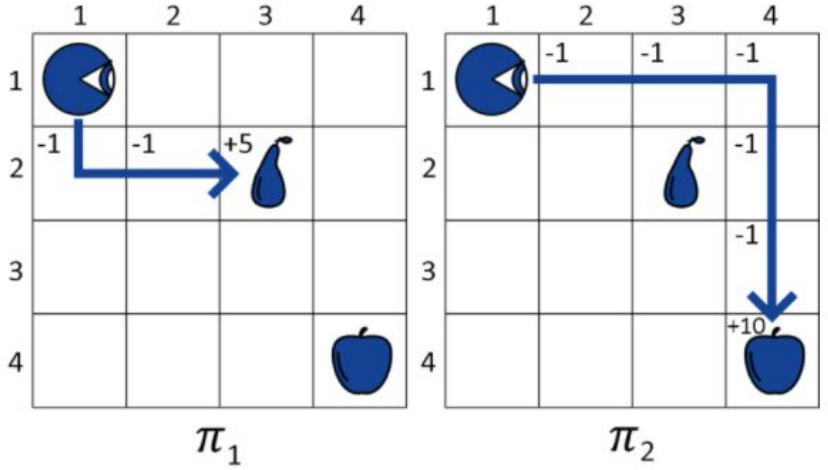
Figure 21.1 (a) A policy π for the 4×3 world; this policy happens to be optimal with rewards of $R(s) = -0.04$ in the nonterminal states and no discounting. (b) The utilities of the states in the 4×3 world, given policy π.

En este aprendizaje la característica principal es que el agente aprende de lo observado pero no tiene ninguna decisión ya que todo está fijado con anterioridad y no va a cambiar en el entrenamiento. Lo único que va a hacer es evaluar las secuencias.

$$(1,1)_{\text{-.04}} \rightsquigarrow (1,2)_{\text{-.04}} \rightsquigarrow (1,3)_{\text{-.04}} \rightsquigarrow (1,2)_{\text{-.04}} \rightsquigarrow (1,3)_{\text{-.04}} \rightsquigarrow (2,3)_{\text{-.04}} \rightsquigarrow (3,3)_{\text{-.04}} \rightsquigarrow (4,3)_{\text{+1}}$$
$$(1,1)_{\text{-.04}} \rightsquigarrow (1,2)_{\text{-.04}} \rightsquigarrow (1,3)_{\text{-.04}} \rightsquigarrow (2,3)_{\text{-.04}} \rightsquigarrow (3,3)_{\text{-.04}} \rightsquigarrow (3,2)_{\text{-.04}} \rightsquigarrow (3,3)_{\text{-.04}} \rightsquigarrow (4,3)_{\text{+1}}$$
$$(1,1)_{\text{-.04}} \rightsquigarrow (2,1)_{\text{-.04}} \rightsquigarrow (3,1)_{\text{-.04}} \rightsquigarrow (3,2)_{\text{-.04}} \rightsquigarrow (4,2)_{\text{-1}} .$$

Una visión del mismo ejemplo pero más sencilla :)





Obtenido de:

https://www.baeldung.com/cs/ml-policy-reinforcement-learning

Estimación de utilidad directa

La suma de las recompensas desde un estado hasta el estado final se le llama “reward on go”. Al final de cada secuencia, este algoritmo calcula los “reward on go” y los actualiza, se espera que al realizarlo infinitas veces este valor converja.

Al no explotar el hecho de que los estados son dependientes y el uso de la ecuación de Bellman, esta estimación converge muy lentamente.

$$U_\pi(s) = R(s) + \gamma \sum_{s'} T(s, \pi(s), s') U_\pi(s')$$

Note the dependence on neighboring states

Suppose we observe the following trial:

$$(1,1)_{-0.04} \rightarrow (1,2)_{-0.04} \rightarrow (1,3)_{-0.04} \rightarrow (1,2)_{-0.04} \rightarrow (1,3)_{-0.04} \rightarrow (2,3)_{-0.04} \rightarrow (3,3)_{-0.04} \rightarrow (4,3)_{+1}$$

The total reward starting at (1,1) is 0.72. We call this a sample of the observed-reward-to-go for (1,1).

Programación dinámica adaptativa

$$U_\pi(s) = R(s) + \gamma \sum_{s'} T(s, \pi(s), s') U_\pi(s')$$

Este método soluciona la mayor falla del anterior, también aprende el modelo de transición y la función de recompensa, al final se soluciona el proceso de decisión de Markov con programación dinámica. Las nuevas funciones son lineales.

La función de recompensa es facil de obtener ya que cada que se visita un estado nuevo, la recompensa se guarda en R(s).

El modelo de transiciones T(s, π(s), s') se obtiene observando que tan seguido se obtiene s' cada que se está en s y se hace la acción π(s).

Algoritmo y rendimiento ADP

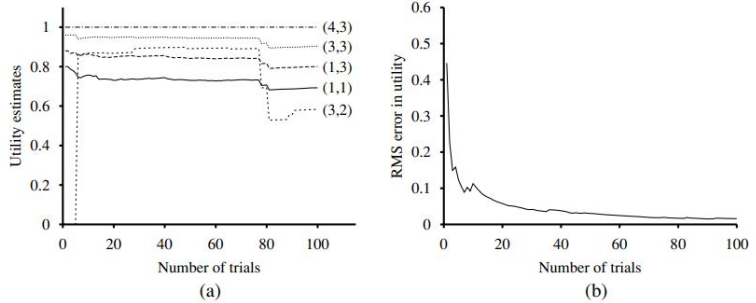


Figure 21.3 The passive ADP learning curves for the 4×3 world, given the optimal policy shown in Figure 21.1. (a) The utility estimates for a selected subset of states, as a function of the number of trials. Notice the large changes occurring around the 78th trial—this is the first time that the agent falls into the -1 terminal state at (4,2). (b) The root-mean-square error (see Appendix A) in the estimate for $U(1,1)$, averaged over 20 runs of 100 trials each.

function PASSIVE-ADP-AGENT($percept$) **returns** an action
  inputs: $percept$, a percept indicating the current state s' and reward signal r'
  persistent: π, a fixed policy
    mdp, an MDP with model P, rewards R, discount γ
    U, a table of utilities, initially empty
    N_{sa}, a table of frequencies for state–action pairs, initially zero
    $N_{s'|sa}$, a table of outcome frequencies given state–action pairs, initially zero
    s, a, the previous state and action, initially null

  if s' is new **then** $U[s'] \leftarrow r'$; $R[s'] \leftarrow r'$
  if s is not null **then**
    increment $N_{sa}[s,a]$ and $N_{s'|sa}[s',s,a]$
    for each t such that $N_{s'|sa}[t,s,a]$ is nonzero **do**
      $P(t \mid s,a) \leftarrow N_{s'|sa}[t,s,a] \,/\, N_{sa}[s,a]$
  $U \leftarrow$ POLICY-EVALUATION(π, U, mdp)
  if s'.TERMINAL? **then** $s,a \leftarrow$ null **else** $s,a \leftarrow s', \pi[s']$
  return a

Update reward function

Update transition model

Figure 21.2 A passive reinforcement learning agent based on adaptive dynamic programming. The POLICY-EVALUATION function solves the fixed-policy Bellman equations, as described on page 657.

Al resolver un sistema de ecuaciones lineales O(n3) puede llegar a ser muy complejo en aplicaciones como el Backgammon.

Aprendizaje diferencial-temporal

$$U_\pi(s) = U_\pi(s) + \alpha(R(s) + \gamma U_\pi(s') - U_\pi(s))$$

En vez de calcular la utilidad con exactitud se puede aproximar y tener un método menos costoso computacionalmente.

$$U_\pi(s) = R(s) + \gamma \sum_{s'} T(s, \pi(s), s') U_\pi(s')$$

- Instead of doing this sum over all successors, only adjust the utility of the state based on the successor observed in the trial.
- It does not estimate the transition model – model free

Para lograr esto se agrega un learning rate para converger, tardará más que el ADP pero es menos costoso computacionalmente.

Algoritmo y rendimiento

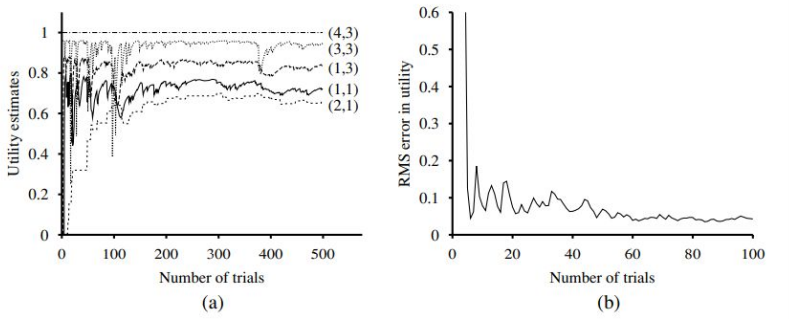


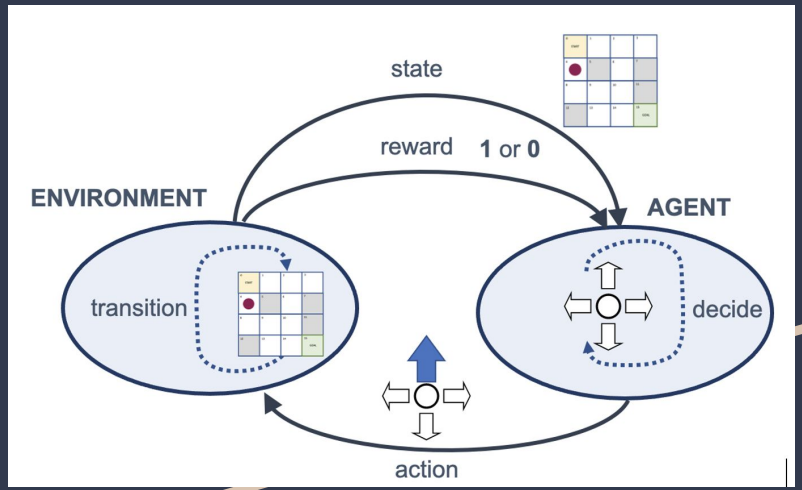
Figure 21.5 The TD learning curves for the 4×3 world. (a) The utility estimates for a selected subset of states, as a function of the number of trials. (b) The root-mean-square error in the estimate for $U(1,1)$, averaged over 20 runs of 500 trials each. Only the first 100 trials are shown to enable comparison with Figure 21.3.

function PASSIVE-TD-AGENT(*percept*) **returns** an action
  inputs: *percept*, a percept indicating the current state s' and reward signal r'
  persistent: π, a fixed policy
    U, a table of utilities, initially empty
    N_s, a table of frequencies for states, initially zero
    s, a, r, the previous state, action, and reward, initially null

  if s' is new **then** $U[s'] \leftarrow r'$
  if s is not null **then**
    increment $N_s[s]$
    $U[s] \leftarrow U[s] + \alpha(N_s[s])(r + \gamma\, U[s'] - U[s])$
  if s'.TERMINAL? **then** $s, a, r \leftarrow$ null **else** $s, a, r \leftarrow s', \pi[s'], r'$
  return a

Figure 21.4 A passive reinforcement learning agent that learns utility estimates using temporal differences. The step-size function $\alpha(n)$ is chosen to ensure convergence, as described in the text.

Aprendizaje reforzado activo



A diferencia de un agente de aprendizaje pasivo el cual tiene una política fija la cual determinan su comportamiento, un agente de aprendizaje activo debe decidir qué acciones tomar.

Se puede modificar el agente anterior para manejar este nuevo modelo de aprendizaje. Para poder realizar esto es necesario utilizar un modelo completo con probabilidades de resultado para todas las acciones en lugar de solo el modelo para la política fija.

$$U(s) = R(s) + \gamma \max_a \sum_{s'} P(s' \mid s, a) U(s') \, .$$

Para este ejemplo se utilizara la ecuación de Bellman para utilidades. Donde la utilidad de un estado es la recompensa inmediata por ese estado más la utilidad descontada esperada del próximo estado, asumiendo que el agente escoge la opción más óptima.

Rutas

(2,1), (1,1), (1,2), (1,3), (2, 3), (3, 3), +1

(1,1), (1,2), (1,3), (2, 3), (3, 3), +1

(3,1), (3,2), (3,3), +1

Ruta optima: (3,1), (3,2), (3,2), +1

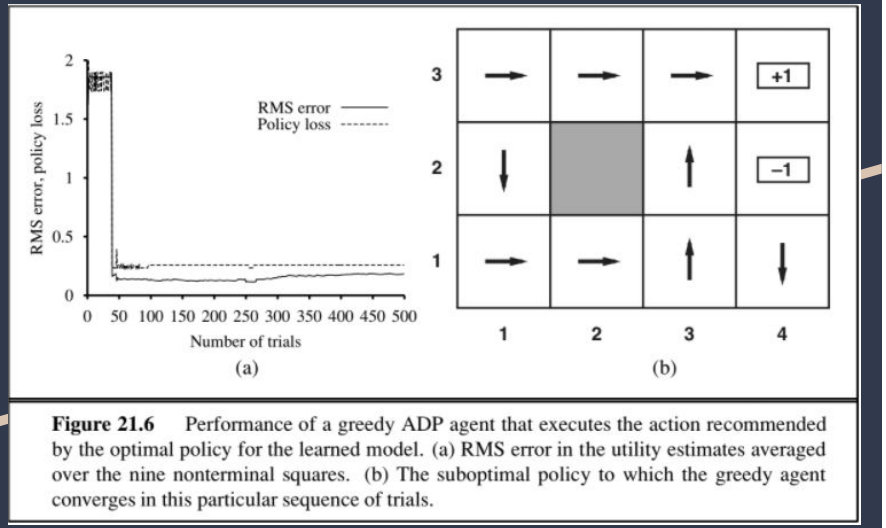


Figure 21.6 Performance of a greedy ADP agent that executes the action recommended by the optimal policy for the learned model. (a) RMS error in the utility estimates averaged over the nine nonterminal squares. (b) The suboptimal policy to which the greedy agent converges in this particular sequence of trials.

Una vez modificado dicho agente, solo nos queda trabajar con las utilidades para extraer una acción óptima mediante la anticipación de un paso. De esta manera tendremos una política óptima ya disponible que ejecute la acción más óptima.

De esta manera tendremos un agente que decidirá sus propios pasos con el fin de buscar una ruta óptima como se puede apreciar en la figura.

El agente encuentra la ruta que le lleva al +1 en la Iteración 39 a través de la ruta inferior. Después de repetir la ruta con ligeras variaciones, a partir de la iteración 276 el agente se apega a dicha ruta. Jamás encontrando la ruta más óptima. Un fenómeno al cual se le conoce como **Agente Codicioso.**

Ventajas

- El agente intenta encontrar una política óptima (o al menos buena) explorando diferentes acciones en el mundo.
- Maximiza el rendimiento.
- Ayuda a lograr resultados a largo plazo que son muy difíciles de lograr por métodos convencionales.
- Es un modelo de aprendizaje muy similar al que manejamos los humanos.

Desventajas

- El agente no siempre encuentra la politica mas optima y hay posibilidades de que se convierta en un agente codicioso.
- No suele ser útil para resolver problemas simples.
- Necesita de muchos datos y cálculos.
- El utilizarse demasiado puede conducir a una sobrecarga de estados que puede afectar los resultados.

Bibliografía

Russell J., Norvin P. (2010). *Artificial Intelligence A Modern approach* (3rd ed.). Pearson.

Oregon State University. *Reinforcement Learning.* https://web.engr.oregonstate.edu/~xfern/classes/cs434/slides/RL-1.pdf

Bhatt S. (2018). *Explaining Rei Learning: Active vs Passive. https://www.kdnuggets.com/2018/06/explaining-reinforcement-learning-active-passive.html*