Aprendizaje Reforzado

Lección 24

Dr. Pablo Alvarado Moya

CE5506 Introducción al reconocimiento de patrones
Área de Ingeniería en Computadores
Tecnológico de Costa Rica

II Semestre, 2019

Contenido

1. Introducción
2. Procesos de decisión de Markov
3. Iteraciones de valor y política
4. Aprendiendo el modelo para un MDP

Aprendizaje reforzado (1)

Reinforcement learning

- En aprendizaje no-supervisado intentamos de descubrir estructuras en datos sin ningún tipo de información adicional, más que el dato.
- En aprendizaje supervisado contamos con la *respuesta correcta* $y^{(i)}$ para cada dato de entrada $\underline{\mathbf{x}}^{(i)}$.
- En problemas de control y de toma de decisiones secuenciales, es dificil o imposible encontrar supervisión explícita $y^{(i)}$.
  - Ejemplo: enseñando a caminar un robot, no necesariamente sabemos cómo debemos mover cada articulación para que se desplace “bien”.

Aprendizaje reforzado (2)

Reinforcement learning

- En **aprendizaje reforzado** brindaremos una **función de recompensa**, que indica al agente de aprendizaje, si está haciendo las cosas bien o mal.
  - Ejemplo: el robot recibirá una recompensa positiva si avanza o negativa si retrocede o se cae.
- El aprendizaje reforzado es complejo, porque los procesos evolucionan en el tiempo, y recompensa no es inmediata (por ejemplo, ganar o perder en un juego es la recompensa final, consecuencia de muchas decisiones parciales anteriores)
- Decisiones cercanas al final, no necesariamente son las que conducen al resultado final (p. ej. frenar abrubtamente no es necesariamente la *causa* de un accidente, sino el *efecto* de intentar evitarlo).

Aplicaciones

- El aprendizaje reforzado se ha aplicado exitosamente en
  - vuelo autónomo de drones (helicópteros, cuadracópteros, botes, etc.)
  - locomoción robótica con extremidades
  - enrutamiento en redes celulares
  - selección de estrategias de mercado
  - control de fábricas
  - indexación eficiente de páginas web
  - aprendizaje de juegos (Atari clásico, AlphaZero, Dota2, etc.)

Mercado

- Hay un fuerte interés en el mercado en aprendizaje reforzado.
- Área de fuerte investigación e impresionante progreso.
- Se considera la entrada a *inteligencia artifical general*
- DeepMind de Google lo usó para aprender a jugar *Go* y Atari.
- OpenAI de Elon Musk lo usó para aprender a jugar *Dota 2*

Procesos de decisión de Markov

- Los **procesos de decisión de Markov** (MDP) proveen el formalismo en el que se presentan usualmente los problemas de aprendizaje reforzado (AR).
- Un MDP es una tupla $\langle \mathcal{S}, \mathcal{A}, \{P_{sa}\}, \gamma, R \rangle$, con
  - S el conjunto de **estados** $S = \{s_1, s_2 \ldots\}$.
    (Ejemplo: número de posiciones posibles del agente)
  - A el conjunto de **acciones** $A = \{a_1, a_2 \ldots\}$
    (Ejemplo: número de direcciones indicadas al agente)
  - P_{sa} las probabilidades de **transición**, $\sum_{s'} P_{sa}(s') = 1$, $P_{sa} \geq 0$
    - Para cada estado $s \in \mathcal{S}$ y acción $a \in \mathcal{A}$, P_{sa} es una distribución sobre el espacio de estados.
    - P_{sa} da la distribución sobre los estados a los que se hará una transición si tomamos la acción a en el estado s.
  - $\gamma \in [0,1)$ el **factor de degradación** (*discount factor*)
  - $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ es la **función de recompensa** (*reward funct.*)
    (usamos también dependencia exclusiva del estado $R : \mathcal{S} \to \mathbb{R}$)

Ejemplo MDP (1)

- Vamos a usar este ejemplo de Russel y Norvig:

(1,3)	(2,3)	(3,3)	$+1$ (4,3)
(1,2)		(3,2)	-1 (4,2)
Inicio (1,1)	(2,1)	(3,1)	(4,1)

- Tenemos 11 estados $s = (i,j) \in \mathcal{S}$ y
- 4 acciones $\mathcal{A} = \{N,S,E,O\}$

Ejemplo MDP (2)

(1,3)	(2,3)	(3,3)	+1 (4,3)
(1,2)		(3,2)	−1 (4,2)
Inicio (1,1)	(2,1)	(3,1)	(4,1)

- Modelamos error de robot en desplazarse con las probabilidades de transición.

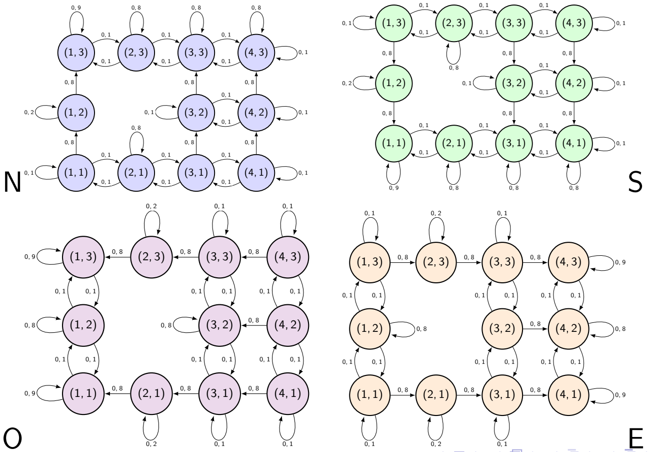
$$P_{(3,1)N}\left((3{,}2)\right) = 0{,}8 \qquad P_{(3,1)N}\left((4{,}1)\right) = 0{,}1$$
$$P_{(3,1)N}\left((2{,}1)\right) = 0{,}1 \qquad P_{(3,1)N}\left((3{,}3)\right) = 0$$

$$P_{(3,1)O}\left((2{,}1)\right) = 0{,}8 \qquad P_{(3,1)O}\left((3{,}2)\right) = 0{,}1$$
$$P_{(3,1)O}\left((3{,}1)\right) = 0{,}1 \qquad P_{(3,1)O}\left((4{,}1)\right) = 0$$

- Nótese que hay total dependencia de la acción.

Ejemplo MDP (3)

- Las probabilidades de transición $P_{sa}(s)$ para cada acción $a \in \{N,S,E,O\}$ se pueden representar con grafos:



Ejemplo MDP (4)

- Las recompensas serán $R\left((4{,}3)\right) = 1$, $R\left((4{,}2)\right) = -1$, y $R\left(s\right) = -0{,}02$ para los otros estados.

Dinámica de un MDP

- La dinámica de un MDP se resume así:
  1. Iniciamos en un estado $s_{(0)}$
  2. Elegimos una acción $a_{(0)} \in \mathcal{A}$ a tomar en el MDP
  3. Acorde a $a_{(0)}$ saltamos aleatoriamente a un siguiente estado $s_{(1)} \sim P_{s_{(0)}a_{(0)}}$; sea $t = 1$
  4. Elegimos otra acción $a_{(t)}$.
  5. Acorde a $a_{(t)}$ seleccionamos siguiente estado $s_{(t+1)} \sim P_{s_{(t)}a_{(t)}}$.
  6. $t \leftarrow t + 1$; repetir desde 4
- Este proceso se representa usualmente como:

$$s_{(0)} \xrightarrow{a_{(0)}} s_{(1)} \xrightarrow{a_{(1)}} s_{(2)} \xrightarrow{a_{(2)}} s_{(3)} \cdots$$

- El **saldo** (*payoff*) luego de una secuencia de pasos es

$$R(s_{(0)},a_{(0)}) + \gamma R(s_{(1)},a_{(1)}) + \gamma^2 R(s_{(2)},a_{(2)}) + \cdots$$

Simplificación del saldo

- Para las derivaciones que siguen asumiremos que la recompensa solo depende del estado.
- El **saldo** en este caso será

$$R(s_{(0)}) + \gamma R(s_{(1)}) + \gamma^2 R(s_{(2)}) + \cdots$$

- La extensión de los métodos a recompensas de estado-acción $\gamma^i R(s_{(i)}, a_{(i)})$ no presenta dificultades adicionales.

Objetivo del aprendizaje reforzado

- En el aprendizaje reforzado queremos **elegir secuencia de acciones** $(a_{(0)}, a_{(1)}, \ldots)$ que **maximicen** el valor esperado del saldo total:

$$\mathrm{E}[R(s_{(0)}) + \gamma R(s_{(1)}) + \gamma^2 R(s_{(2)}) + \cdots]$$

- En el tiempo t, la recompensa se degrada por el factor γ^t.
- Para maximizar la esperanza necesitamos recompensas positivas lo antes posible y posponer recompensas negativas lo más posible.

Políticas

- Una **política** es una función $\pi : \mathcal{S} \to \mathcal{A}$ que mapea de estados a acciones.
- **Ejecutamos** una política π si, estando en el estado s tomamos la acción $a = \pi(s)$.
- Ejemplo de política para caso de estudio:

$\rightarrow$	$\rightarrow$	$\rightarrow$	$+1$
$\uparrow$		$\uparrow$	-1
$\uparrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$

Función de valor

- La **función de valor** para una política π es $V^\pi(s) : \mathcal{S} \to \mathbb{R}$

$$V^\pi(s) = \mathrm{E}\left[R(s_{(0)}) + \gamma R(s_{(1)}) + \gamma^2 R(s_{(2)}) + \cdots \middle| s_{(0)} = s, \pi\right]$$

Función de valor

$V^\pi(s)$ es entonces el valor esperado del saldo total (la suma de recompensas degradadas), iniciando en estado s y tomando acciones acordes a π.

- (Aunque π **no** es variable aleatoria, en literatura es costumbre usar $|\pi$ para denotar la consideración de política π)

Ejemplo de función de valor

$\rightarrow$	$\rightarrow$	$\rightarrow$	$+1$
$\downarrow$		$\rightarrow$	-1
$\rightarrow$	$\rightarrow$	$\uparrow$	$\uparrow$

Política $\pi(s)$

0.39	0.59	0.70	$+1$
-0.53		-0.74	-1
-0.57	-0.63	-0.69	-0.88

Valores $V^{\pi}(s)$

Ecuaciones de Bellman (1)

- Dada una política fija π, la función de valor $V^\pi(s)$ satisface las **ecuaciones de Bellman**:

$$\begin{aligned}
V^\pi(s) &= \mathrm{E}\left[R(s_{(0)}) + \gamma R(s_{(1)}) + \gamma^2 R(s_{(2)}) + \cdots \middle| s_{(0)} = s, \pi\right] \\
&= \mathrm{E}\left[R(s_{(0)}) + \gamma \underbrace{\left(R(s_{(1)}) + \gamma R(s_{(2)}) + \cdots\right)}_{V^\pi(s_{(1)})} \middle| s_{(0)} = s, \pi \right] \\
&= R(s) + \gamma \sum_{s' \in S} P_{s\pi(s)}(s') V^\pi(s') \\
&= R(s) + \gamma\, \mathrm{E}[V^\pi(s')]
\end{aligned}$$

Ecuaciones de Bellman (2)

- Entonces, el valor esperado de la suma de recompensas degradadas $V^\pi(s)$ iniciando en s tiene dos términos:
  1. La **recompensa inmediata** $R(s)$ producida por iniciar en s.
  2. El segundo término puede reescribirse observando que

$$\sum_{s' \in S} P_{s\pi(s)}(s') V^\pi(s') = \mathrm{E}_{s' \sim P_{s\pi(s)}} [V^\pi(s')]$$

  que es la suma de recompensas degradadas esperada, obtenida *después* del primer paso en el MDP.

Encontrando la función de valor (1)

- Las ecuaciones de Bellman se usan para encontrar V^π eficientemente.
- En un MDP de estados finitos ($|\mathcal{S}| < \infty$) podemos plantear una ecuación $V^\pi(s)$ para cada estado s.
- Por ejemplo:

$$V^\pi((3,1)) = R((3,1)) + \gamma[\underbrace{P_{(3,1)N}((3,2))}_{0,8} V^\pi((3,2)) + \underbrace{P_{(3,1)N}((4,1))}_{0,1} V^\pi((4,1)) + \underbrace{P_{(3,1)N}((2,1))}_{0,1} V^\pi((2,1))]$$

- Esto da un conjunto de $|\mathcal{S}|$ ecuaciones lineales con $|\mathcal{S}|$ variables (los valores $V^\pi(s)$ en cada estado).

Encontrando la función de valor (2)

- Ejemplo:

$$\begin{bmatrix} 0.91 & -0.72 & 0 & 0 & -0.09 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.82 & -0.72 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -0.09 & 1 & -0.09 & 0 & -0.72 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -0.09 & 0.91 & 0 & 0 & -0.72 & 0 & 0 & 0 & 0 \\ -0.72 & 0 & 0 & 0 & 0.82 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -0.09 & 0 & 0 & 1 & -0.72 & 0 & 0 & -0.09 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.09 & 0 & 0 & 0.91 & -0.72 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.82 & -0.72 & 0 \\ 0 & 0 & 0 & 0 & 0 & -0.09 & 0 & 0 & 0 & 0.91 & -0.72 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} V^\pi((1,1)) \\ V^\pi((2,1)) \\ V^\pi((3,1)) \\ V^\pi((4,1)) \\ V^\pi((1,2)) \\ V^\pi((3,2)) \\ V^\pi((4,2)) \\ V^\pi((1,3)) \\ V^\pi((2,3)) \\ V^\pi((3,3)) \\ V^\pi((4,3)) \end{bmatrix} = \begin{bmatrix} -0.02 \\ -0.02 \\ -0.02 \\ -0.02 \\ -0.02 \\ -0.02 \\ -1 \\ -0.02 \\ -0.02 \\ -0.02 \\ +1 \end{bmatrix}$$

- Usamos cualquiera de los métodos estándar de resolución de ecuaciones (QR, LU, SVD, etc.)

Función de valor óptima

- La **función de valor óptima** se define como:

$$V^*(s) = \max_{\pi} V^{\pi}(s)$$

- Esto corresponde al mejor saldo total que se puede obtener con cualquier política.
- También hay una versión de las ecuaciones de Bellman para la función de valor óptima:

$$V^*(s) = R(s) + \max_{a \in \mathcal{A}} \gamma \sum_{s' \in \mathcal{S}} P_{sa}(s') V^*(s')$$

1. El primer término es la recompensa inmediata
2. El segundo término es el máximo saldo total esperado sobre todas las acciones a

Política óptima

- Podemos entonces definir la política

$$\pi^*(s) = \arg\max_{a\in\mathcal{A}} \sum_{s'\in\mathcal{S}} P_{sa}(s')V^*(s')$$

- Esta política $\pi^*(s)$ corresponde a aquella que maximiza $V^*(s)$
- Para todo estado s y toda política π, se cumple

$$V^*(s) = V^{\pi^*}(s) \geq V^{\pi}(s)$$

- Nótese que $\pi^*(s)$ produce el máximo valor para **todos** los estados.
- Esto implica que la misma política $\pi^*(s)$ se usa independientemente del estado inicial.

Solución de MDP

- Nótese que con 4 acciones y 11 estados en nuestro ejemplo, el número posible de políticas es $4^{11} \approx 4{,}19 \times 10^6$, y por tanto no es factible hacer la búsqueda exhaustiva de π^*, donde para cada política hay que resolver el sistema de ecuaciones.
- Esto empeora exponencialmente con el número de estados y acciones.
- En principio, si conociéramos V^* sería trivial encontrar π^*, pues estaría dada directamente por:

$$\pi^*(s) = \arg\max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} P_{sa}(s') V^*(s')$$

- Necesitamos entonces un algoritmo para encontrar V^*, y luego con la ecuación anterior encontramos π^*.

Iteración de valor

- El algoritmo de **iteración de valor** se plantea como:

for $s \in \mathcal{S}$ **do**
    $V(s) := 0$
end for
repeat
    for $s \in \mathcal{S}$ **do**
        Actualice $V(s) := R(s) + \max_{a\in\mathcal{A}} \gamma \sum_{s'} P_{sa}(s')V(s')$
    end for
until (convergencia)

- La iteración de valor aplica repetidamente las ecuaciones de Bellman

Iteración de valor

- Hay dos formas de hacer las actualizaciones de la línea 6:

$$V(s) := R(s) + \max_{a\in\mathcal{A}} \gamma \sum_{s'} P_{sa}(s')V(s')$$

1. Actualización **sincrónica**: calcula primero **todos** los nuevos valores $V(s)$ antes de sobreescribirlos.
2. Actualización **asincrónica**: los estados se visitan en algún orden y $V(s)$ se actualiza de una vez.

- Se ha demostrado que ambas estrategias convergen a $V^*(s)$.
- Con $V^*(s)$ encontramos π^* con

$$\pi^*(s) = \arg\max_{a\in\mathcal{A}} \sum_{s'\in\mathcal{S}} P_{sa}(s')V^*(s')$$

Iteración de política

- Otro algoritmo disponible para encontrar π^* es la **iteración de política**:

  1: Inicialice π aleatoriamente
  2: **repeat**
  3: $\quad V := V^\pi$
  4: $\quad$ **for** $s \in \mathcal{S}$ **do**
  5: $\qquad \pi(s) = \arg\max_{a\in\mathcal{A}} \sum_{s'} P_{sa}(s')V(s')$
  6: $\quad$ **end for**
  7: **until** (convergencia)

- El ciclo interno repetidamente calcula la función de valor para la política actual y luego actualiza la política usando la función de valor actual.
- Después de un número finito de iteraciones V converge a V^* y π converge a π^*

Iteraciones de valor y política en MDP

- Ambos algoritmos de iteración de política y de valor son estándar en la solución de MDP.
- Ninguno de los dos se considera mejor que el otro.
- Para MDP pequeños se usa con más frecuencia iteración de política
- Para MDP con espacios de estado grandes ($|\mathcal{S}| > 1000$) se evita solucionar en la iteración un sistema grande de ecuaciones lineales usando la iteración de valor.
- Por la última razón, es más frecuente encontrar la iteración de valor.

Aprendiendo el modelo para un MDP

- Hasta ahora hemos supuesto que conocemos las probabilidades de transición y las recompensas.
- En la mayoría de problemas reales no tenemos esa información, y debemos estimarla de los datos.
- Vamos a suponer que $\mathcal{S}$, $\mathcal{A}$ y γ son conocidos.
- Supongamos que tenemos varias secuencias disponibles:

$$s_{(0)}^{(1)} \xrightarrow{a_{(0)}^{(1)}} s_{(1)}^{(1)} \xrightarrow{a_{(1)}^{(1)}} s_{(2)}^{(1)} \xrightarrow{a_{(2)}^{(1)}} s_{(3)}^{(1)} \cdots$$

$$s_{(0)}^{(2)} \xrightarrow{a_{(0)}^{(2)}} s_{(1)}^{(2)} \xrightarrow{a_{(1)}^{(2)}} s_{(2)}^{(2)} \xrightarrow{a_{(2)}^{(2)}} s_{(3)}^{(2)} \cdots$$

- La notación $s_{(t)}^{(j)}$ indica el estado en el paso t del j-ésimo experimento.
- Los experimentos se repiten ya sea por un número finito (grande) de pasos, o hasta que el experimento termine.

Estimación de máxima verosimilitud

- Con la experiencia acumulada se derivan los estimados de máxima verosimilitud para las probabilidades de transición:

$$P_{sa}(s') = \frac{\#\text{ veces tomamos } a \text{ en } s \text{ y llegamos a } s'}{\#\text{ veces tomamos } a \text{ en } s}$$

- Si lo anterior da $0/0$ por falta de observaciones, entonces usamos $1/|S|$
- Conforme observemos más experimentos, si contamos con variables para numerador y denominador es fácil actualizar el modelo en-línea.

Estrategia para acumular experiencia

1: **repeat**
2: Tome acciones según política π para acumular experiencia
3: Actualice estimaciones de $P_{sa}(s')$
4: Resuelva con iteración de valor para obtener V
5: Actualice $\pi(s) = \arg\max_{a\in\mathcal{A}} \sum_{s'\in\mathcal{S}} P_{sa}(s')V(s')$
6: **until** (convergencia)

- El paso 4 es eficiente puesto que podemos usar el V de la iteración anterior como inicialización (en vez de $V(s) = 0$), lo que permite rápida convergencia.

Resumen

1. Introducción
2. Procesos de decisión de Markov
3. Iteraciones de valor y política
4. Aprendiendo el modelo para un MDP

Este documento ha sido elaborado con software libre incluyendo LaTeX, Beamer, GNUPlot, GNU/Octave, XFig, Inkscape, GNU-Make y Subversion en GNU/Linux