Sistemas de sistemas de recomendación mediante aprendizaje por refuerzo

Deyviss Jesús Oroya Villalta

23 de julio de 2020

Índice general

Introducción **7**

1. Introducción al aprendizaje por refuerzo **9**
1.1. Proceso de decisión de Markov . . . 9
1.2. Programación dinámica . . . 13
1.2.1. Funciones valor estado (v^π) . . . 13
1.2.2. Función valor de estado-accion (q^π) . . . 14
1.2.3. Operadores de Bellman (OB) . . . 15
1.3. Algoritmos de aprendizaje por refuerzo . . . 16

2. Introdución a Sistemas de recomendación **21**
2.1. Filtrado colaborativo . . . 21
2.2. Filtrado basado en contenido . . . 21
2.3. Sistema de basado en un proceso de decición de Markov . . . 22

3. Preprocesado del historial de películas **23**
3.1. Descripción de la base de datos iniciales . . . 23
3.2. Union de base de datos de generos y puntaciones . . . 23
3.3. Normalización de puntación de las película . . . 24
3.4. Componentes principales para generos de películas . . . 24
3.5. Historial de aceptación para cada usuario . . . 25
3.6. Generación sintética del historial de rechazo . . . 25
3.7. Bases de datos finales: Historiales de usuarios . . . 26

4. Sistema de recomendación como proceso de decisión de Markov **29**
4.1. Proceso de decisión de Markov . . . 29
4.2. Plantamiento de los problemas . . . 29
4.3. Metodología . . . 30
4.3.1. Selección de estado inicial . . . 30
4.3.2. Inicialización de la función valor estado-acción . . . 30
4.3.3. Política utilizada . . . 31
4.3.4. Selección de peliculas desde un política dada . . . 31
4.4. Resultados numéricos . . . 31
4.4.1. Políticas de Referencia . . . 31
4.4.2. Caso 1: Usuario sin historial previo . . . 32
4.4.3. Caso 2: Usuario con historial previo . . . 32

5. Conclusiones **33**

Índice de figuras

1.1. Proceso de desición de Markov para el ejemplo (1.1.1) 12
1.2. Comportamiento de la partícula en un potencial 19
1.3. Evolución de la función vlaor por *Value Iteration* 19
1.4. Evolución de la política por *Value Iteration* 20
1.5. Espacio de fases modificado 20

3.1. Analisis PCA en los generos de películas 25
3.2. Diagrama de flujo para el preprocesado de los datos. 27

4.1. Aprendizaje por refuerzo en el sistema de recomendación 30
4.2. Políticas de referencias para un usuario concreto. 32

Introducción

PRESETACION TESIS

Capítulo 1

Introducción al aprendizaje por refuerzo

El aprendizaje por refuerzo es una técnica de aprendizaje automático capaz de resolver problemas de forma dinámica. Esta técnica se define como un proceso iterativo donde existe un agente capaz de tomar decisiones y un sistema dinámico influenciado por este. Debido a la interacción entre el agente y el sistema dinámico, el agente puede controlar el sistema a configuraciones de interés. Estas configuraciones de interés vienen caracterizadas por una función de coste que es máxima cuando la configuración es alcanzada. El aprendizaje por refuerzo utiliza el principio de la programación dinámica para obtener una estrategia de actuación, sin necesidad de conocer la ecuación dinámica del sistema.

A lo largo de este capítulo describiremos formalmente el aprendizaje por refuerzo como un proceso de decisión de Markov, enseñando algunos ejemplos. Luego de esto introduciremos la programación dinámica como método para abordar el problema presentado en el marco de los procesos de decisión de Markov. Por último, mostraremos algunos algoritmos estándar para abordar este tipo de problemas.

1.1. Proceso de decisión de Markov

Un proceso de decisión de Markov es un proceso dinámico en tiempo discreto y probabilístico capaz de modelizar una gran variedad de procesos. Este marco matemático es conocido por lo menos desde la decada de 1950, donde en [Bellman, 1957] se escribía formalmente por primera vez.

A través de este capítulo usaremos la notación introducida en [Sutton and Barto, 2018, cap. 3].

Definición 1.1.1 (Proceso de decisión de Markov) Se define un proceso de decisión de Markov como una tupla de cuatro objetos:

- Una variable aleatoria $S_t \in \mathcal{S}$, que representa el estado en la iteración t.
- Una variable aleatoria $A_t \in \mathcal{A}$, que representa la acción en la iteración t.
- Una variable aleatoria $R_t \in \mathcal{R}$, que representa la recompensa obtenida en la iteración t.
- Una distribución de probabilidad:

$$\mathbb{P}(S_t = s', R_t = r | S_{t-1} = s, A_{t-1} = a) = p(s', r|s, a) \tag{1.1.1}$$

  Donde a la función $p : \mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$ será llamada la **función dinámica** del proceso de decisión de Markov. Esta representa la probabilidad de que estando en el estado s y tomando la acción a, obtengamos la recompensa r y transitemos al estado s'. □

Observación 1.1.1 Aunque el espacio de recompensas puede ser infinito, a lo largo de este capítulo consideraremos que el espacio de recompensas $\mathcal{R}$ es un espacio discreto y finito. Debido a esto siempre podemos sumar en todo el espacio $\mathcal{R}$.

Por otro lado, existe otras distribuciones de probabilidad que pueden ser obtenidas de la función dinámica $p(s', r|s, a)$. Estas distribuciones son las distribución de probabilidad de transición y de recompensa.

La distribución de probabilidad de transición $p(s'|s,a)$ se puede obtener como:

$$p(s'|s,a) = \sum_{r\in\mathcal{R}} p(s',r|s,a) \tag{1.1.2}$$

Mientras que la distribución de probabilidad de recompensa $p(r|s,a)$ se obtiene de la siguiente manera:

$$p(r|s,a) = \sum_{s'\in\mathcal{S}} p(s',r|s,a) \tag{1.1.3}$$

Observación 1.1.2 La distribución de probabilidad de la dinámica $p(s',r|s,a)$, la distribución de probabilidad de transición $p(s'|s,a)$ y la distribución de probabilidad de recompensa $p(r|s,a)$ son funciones distintas aunque utilicemos la misma letra p para referirnos a ellas. Aunque este es un abuso de notación, podemos referirnos a las distintas distribuciones de probabilidad sabiendo que tienen número de argumentos de entrada distintos (véase el cuadro (1.1)).

Distribución de probabilidad	**Función**	**Notación**
de la dinámica	$p : \mathcal{S}\times\mathcal{R}\times\mathcal{S}\times\mathcal{A}\to[0,1]$	$p(s'r\|s,a)$
de transición	$p : \mathcal{S}\times\mathcal{S}\times\mathcal{A}\to[0,1]$	$p(s'\|s,a)$
de recompensa	$p : \mathcal{R}\times\mathcal{S}\times\mathcal{A}\to[0,1]$	$p(r\|s,a)$

Cuadro 1.1: Notación de distribuciones de probilidad.

Por otra parte, podemos obtener la recompensa esperada debido a la acción a y estando en el estado s como una función $r : \mathcal{S}\times\mathcal{A}\to\mathcal{R}$ tal que:

$$r(s,a) = \mathbb{E}[R_t|S_{t-1}=s, A_{t-1}=a] = \sum_{s'\in\mathcal{S}}\sum_{r\in\mathcal{R}} r\ p(s',r|s,a) \tag{1.1.4}$$

Observación 1.1.3 La letra r se utiliza para denotar una variable aleatoria asociada a la recompensa, mientras que utilizamos la notación $r(s,a)$ para referiremos a la esperanza de la variable aleatoria r debido a la acción a y estando en el estado s.

Aunque en la definición formal de un proceso de decisión de Markov es necesario definir la distribución de probabilidad de la dinámica $p(s',r|s,a)$, es habitual definir el proceso de decisión de Markov mediante la distribución de probabilidad de transición $p(s'|s,a)$ y la distribución de probabilidad recompensa $p(r|s,a)$. Además también es posible definir un proceso de decisión de Markov, mediante la recompensa esperada $r(s,a)$ y la distribución de probabilidad de transición $p(s'|s,a)$. Estas tres formas de definir los procesos de decisión de Markov son equivalentes, sin embargo dependiendo de la aplicación una es más conveniente que otra (véase el ejemplo 1.1.1).

Con respecto al comportamiento del agente, se define la regla de decisión y la política para su caracterización. Estos objetos nos indican que acciones tomará el agente o como dependen estas acciones del estado.

Definición 1.1.2 (Regla de decisión) En el momento de tiempo $t\in\mathbb{N}$ una **regla de decisión** es una función $\pi_t : \mathcal{S}\to\mathcal{A}$, que para cada elemento $s\in\mathcal{S}$ asocia $a\in\mathcal{A}$. □

Definición 1.1.3 (Política) Se define como **política** de un proceso de decisión de Markov como una secuencia de reglas de decisión, $\pi = (\pi_1,\dots,\pi_T)$ que se aplican en un proceso de decisión. Podemos definir un espacio de posibles políticas $\Pi = \mathcal{A}\times\mathcal{A}\times\cdots\times\mathcal{A} = \mathcal{A}^T$, tal que $\pi\in\Pi$ □

Entonces dadas esta definiciones previas podemos formular cualquier proceso de decisión de Markov. Con el objetivo ejemplificar las definiciones presentadas anteriormente veremos un ejemplo de proceso de decisión de Markov.

Ejemplo 1.1.1 (Control de una partícula) Sea el espacio de estados $\mathcal{S} = \{s_1, s_2, s_3\}$ y el espacio de acciones $\mathcal{A} = \{\leftarrow\ ,\ =\ ,\ \rightarrow\}$. Además sea la distribución de probabilidad de transición $p(s',r|s,a)$, donde

suponemos que la variable aleatoria recompensa r y la variable aleatoria de estado siguiente s' son independientes. Entonces podemos factorizar la distribución de probabilidad de la dinámica:

$$p(s', r|s, a) = p(r|s, a)p(s'|s, a) \tag{1.1.5}$$

En este caso podemos definir la distribución $p(r|s, a)$ y la distribución $p(s'|s, a)$ por separado. Definiremos estas dos distribuciones de forma determinista. Para ello utilizaremos la función delta de Kronecker δ definida como:

$$\delta(x, y) = \begin{cases} 0 & x \neq y \\ 1 & x = y \end{cases}$$

En primer lugar, definiremos la distribución de probabilidad de transición $p(s'|s, a)$. Para ello asociaremos cada uno de los estados $\{s_1, s_2, s_3\}$ a los vectores unitarios de $\mathbb{R}^3$:

$$\{s_1, s_2, s_3\} \to \{(1, 0, 0), (0, 1, 0), (0, 0, 1)\} \tag{1.1.6}$$

Definimos la función $f_s : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ como:

$$f_s(s, a) = \delta(a, \leftarrow) \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} s + \delta(a, =) \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} s + \delta(a, \to) \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} s \tag{1.1.7}$$

Con ayuda de esta función $f_r(s, a)$ podemos definir la distribución de probabilidad $p(s'|s, a)$ como:

$$p(s'|s, a) = \delta(s', f_s(s, a)) \tag{1.1.8}$$

En la figura (1.1a) se muestra un representación gráfica de la distribución de probabilidad de transición. Allí podemos ver que está definición simplemente representa una partícula que puede moverse en un recta con tres posiciones y que las acciones que podemos tomar representa un movimiento a la derecha, a la izquierda o quedarnos en la posición en la que nos encontramos.

En segundo lugar, definiremos la distribución de probabilidad de recompensa $p(r|s, a)$. Para ello definimos la función $f_r : \mathcal{S} \times \mathcal{A} \to \{-2, -1, 9, 10\}$, como:

$$f_r(s, a) = \begin{cases} 10 - \delta(a, \to) - \delta(a, \leftarrow) & s = s_2 \\ -1 - \delta(a, \to) - \delta(a, \leftarrow) & s \neq s_2 \end{cases} \tag{1.1.9}$$

Entonces podemos definir la distribución de probabilidad $p(r|s, a)$ como:

$$p(r|s, a) = \delta(r, f_r(s, a)) \tag{1.1.10}$$

Suponemos que el hecho de moverse penaliza en una unidad la recompensa. Además que el sistema alcanza el máximo en la posición $s = s_2$

Por otra parte podemos calcular la recompensa esperada para este sistema:

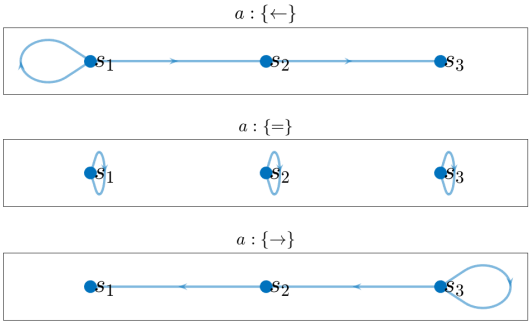
$$r(s, a) = \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} p(s', r|s, a) = \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} p(s'|s, a)p(r|s, a) \tag{1.1.11}$$

Utilizando las ecuaciones (1.1.10) y(1.1.8):

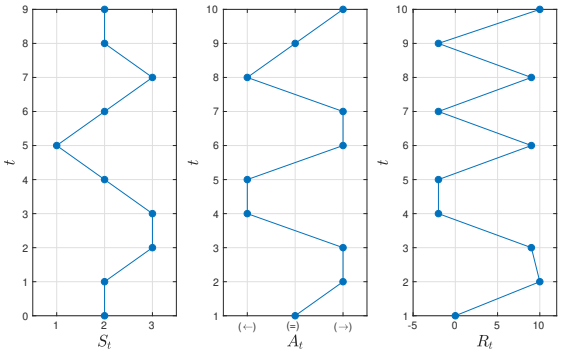
$$r(s, a) = \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} \delta(s', f_s(s, a))\delta(r, f_r(s, a)) = f_r(s, a) \tag{1.1.12}$$

Asi podemos que que $r(s, a) = f_r(s, a)$, por lo que en este ejemplo definir la distribución de probabilidad de recompensa $p(r|s, a)$ o la recompensa esperada $r(s, a)$ son equivalentes.

Para concluir, notemos que con la distribución de probabilidad de transición $p(s'|s, a)$ y la distribución de probabilidad de recompensa $p(r|s, a)$, el proceso de desición de Markov del control de una partícula queda completamente caracterizado.



(a) Proceso de decisión de Markov. Para cada posible acción se puede definir una matriz que a su vez se puede representar como un grafo.



(b) Evolución del sistema bajo una política. De una manera intuitiva, el agente actúa en el sistema como si tuviera un mando derecha-izquierda.

Figura 1.1: Proceso de desición de Markov para el ejemplo (1.1.1)

Los conceptos presentados son suficientes para definir correctamente un proceso de decisión de Markov. A partir de estos se puede definir una función de recompensa que nos determinará que tan cerca estamos a la configuración deseada. Este se llama **retorno esperado**

Definición 1.1.4 Se define el **retorno esperado** a iteración t, como la suma de las recompensas desde la iteración $t+1$ hasta la iteración final T

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{\tau=0}^{T} \gamma^\tau R_{\tau+t+1} \tag{1.1.13}$$

Donde $\gamma \in (0,1)$, es llamado el factor de descuento. □

Observación 1.1.4 El factor de descuento γ impide que el retorno esperado diverja cuando el horizonte temporal es $T = \infty$, siempre que las recompensas estén acotadas en todo el espacio $\mathcal{R}$. Un segundo propósito de este factor es dar más importancia las recompensas obtenidas en un tiempo cercano.

Observación 1.1.5 (Recurencia del retorno esperado) Es importante notar que cuando el horizonte temporal T es infinito el retorno esperado G_t cumple:

$$G_t = R_{t+1} + \gamma G_{t+1} \tag{1.1.14}$$

El aprendizaje por refuerzo tiene como objetivo la maximización el retorno esperado G_t con respecto a la política $\pi \in \Pi$.

Problema 1.1.1 Definimos el **problema de optimización**

$$\max_{\pi\in\Pi} \mathbb{E}_\pi \left[\sum_{\tau=0}^{T} \gamma^\tau R_{\tau+1} \middle| S_0 = s \right] = \max_{\pi\in\Pi} \mathbb{E}_\pi \left[G_0 \middle| S_0 = s \right] \tag{1.1.15}$$

Donde $\mathbb{E}_\pi$ representa la esperanza con respecto a todas las posibles trayectorias en el espacio de estados tomando la política π.

La solución a este problema también tiene un carácter especial, por que lo definiremos.

Definición 1.1.5 Definimos la **política óptima**, π_* como:

$$\pi_* \in \arg\max_{\pi\in\Pi} \mathbb{E}_\pi \left[G_0 | S_0 = s \right] \tag{1.1.16}$$

□

Entonces un proceso de decisión de Markov en síntesis consiste maximizar el retorno esperado G_t con-respecto a la política $\pi \in \Pi$ y sujeto a la distribución de probabilidad de la dinámica $p(s', r|s, a)$.

1.2. Programación dinámica

La programación dinámica es una metodología de optimización, desarrollada en [Bellman, 1954]. Aunque en un principio la programación dinámica fue formulada en sistemas discretos deterministas su generalización a procesos probabilísticos fue realizada poco después de su creación en [Bellman, 1957].

1.2.1. Funciones valor estado (v^π)

Las funciones valor son la base de la programación dinámica, sus leyes de recurencia nos permiten construir algoritmos en el marco del aprendizaje por refuerzo. A continuación las definiremos.

Definición 1.2.1 (Función valor de estado) Definición la **función valor de estado** como una función, $v^\pi : \mathcal{S} \to \mathbb{R}$:

$$v^\pi(s) = \mathbb{E}_\pi\big[\sum_{t=0}^{\infty} \gamma^\tau R_{\tau+1} | S_0 = s\big] = \mathbb{E}_\pi[G_t | S_t = s] \tag{1.2.1}$$

Donde hemos tomado el horizonte temporal, $T = \infty$. □

Observación 1.2.1 En este documento tomaremos la definición 1.2.1 como la definición de función valor de estado, tomando $T = \infty$. Estos problemas son llamados procesos de decisión de Markov con horizonte infinito. Sin embargo, el horizonte temporal T puede ser finito aunque requiere una definición ligeramente distinta. Puede verse más definiciones en [Lazaric, 2013].

Otra pieza fundamental en la programación dinámica es la función valor de estado óptima.

Definición 1.2.2 (Función valor de estado óptima) Definimos la **función valor de estado óptima** como:

$$v_*(s) = \max_{\pi \in \Pi} v^\pi(s) \tag{1.2.2}$$

Esta función tiene una clara interpretación en el proceso de decisión de Markov. Esta es la solución del problema de optimización (1.1.15) para una condición inicial $S_0 = s$. □

Si descomponemos G_t utilizando la ecuación (1.1.14), en la ecuación (1.2.1)

$$v^\pi(s) = \mathbb{E}_\pi[G_t | S_t = s] = \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} | S_t = s] \tag{1.2.3}$$

Deshaciendo la sumatoria de la esperanza en el primer paso, obetenemos:

$$v^\pi(s) = \sum_{s',r} p(s', r | s, \pi(s)) \big[r + \gamma \underbrace{\mathbb{E}_\pi[G_{t+1} | S_t = s']}_{v^\pi(s')}\big] \tag{1.2.4}$$

Y con ello obetenemos la ecuación de Bellman para la función valor de estado.

Colorario 1.2.1 (Ecuación de Bellman) Entonces la función valor de estado $v^\pi(s)$ cumple la siguiente ecuación:

$$v^\pi(s) = \sum_{s',r} p(s', r | s, \pi(s)) \Big[r + \gamma v^\pi(s')\Big] \tag{1.2.5}$$
$$\forall s \in \mathcal{S} \ , \ \forall t \in \{1, \dots, T\}$$

Utilizando las ecuaciones (1.1.4) y (1.1.2) podemos re-escribir la expresión anterior de la siguiente manera:

$$v^\pi(s) = r(s, \pi(s)) + \gamma \sum_{s'} p(s' | s, \pi(s)) v^\pi(s') \tag{1.2.6}$$

Tomando la definición del la función de estado óptima (1.2.2) y la expresión anterior:

$$v_*(s) = \max_{\pi \in \Pi} v^\pi(s) = \max_{\pi \in \Pi} \Big[r(s, \pi(s)) + \gamma \sum_{s'} p(s' | s, \pi(s)) v^\pi(s')\Big] \tag{1.2.7}$$

Podemos obtener la ecuación de Bellman para la función valor de estado óptima.

Colorario 1.2.2 (Ecuación de Bellman óptima) Entonces la función valor de estado óptima $v_*(s)$ cumple la siguiente ecuación:

$$v_*(s) = \max_{a'} \left[r(s, a') + \gamma \sum_{s'} p(s'|s, a') v_*(s') \right] \tag{1.2.8}$$

Esta espresión nos permite recuperar la política óptima tomando el argumento de la maximización anterior:

$$\pi_*(s) = \arg\max_{a'} \left[r(s, a') + \gamma \sum_{s'} p(s'|s, a') v_*(s') \right] \tag{1.2.9}$$

Entonces, podemos obtener las reglas de decisión $\pi_*(s)$ si conocemos la función valor $v_*(s) \quad \forall s \in \mathcal{S}$.

1.2.2. Función valor de estado-accion (q^π)

En la programación dinámica se define otras dos funciones muy útilies y en ocasiones más conveniente que la función valor de estado. Estas son llamadas la función valor estado-acción y la función valor de estado-acción óptima.

Definición 1.2.3 (Función valor de estado-acción) Definimos la función valor de estado-acción como una función, $q^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ tal que:

$$q^\pi(s, a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a] \tag{1.2.10}$$

Esta función es similar a la función valor de estado pero con un grado de libertad adicional a la hora de elegir la primera acción. Las sucesivas acciones vienen dada por la política π. □

Definición 1.2.4 (Función valor de estado-acción óptima) Definimos la función valor de estado-acción óptima como una función, $q_* : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ tal que:

$$q_*(s, a) = \max_{\pi \in \Pi} q^\pi(s, a) \tag{1.2.11}$$

La función de valor estado-acción óptima tiene una interpresación en el proceso de decisión de Markov. Esta es la solución del problema de optimización (1.1.15) para una condición inicial $S_0 = s$ y tomando la primera acción $A_0 = a$. □

Encontraremos la relación entre $v_*(s)$ y $q_*(s)$, con el fin de definir la manera en la que encontramos la política óptima a partir de $q_*(s)$.

Utilizamos la recurencia del retorno esperado G_t

$$q^\pi(s, a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a] = \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} | S_t = s, A_t = a] \tag{1.2.12}$$

luego deshacemos el primer sumatorio de la esperanza E_π:

$$q^\pi(s, a) = \sum_{s', r} p(s', r|s, a) \left[r + \gamma \underbrace{\mathbb{E}_\pi[G_{t+1} | S_t = s']}_{v^\pi(s)} \right] \tag{1.2.13}$$

Entonces obtenemos una relación directa entre la función valor estado $v^\pi(s)$ y la función valor estado-acción $q^\pi(s)$:

$$q^\pi(s, a) = r(s, a) + \gamma \sum_{s'} p(s'|s, a) v^\pi(s') \tag{1.2.14}$$

Ahora utilizamos (1.2.11) en (1.2.14):

$$q_*(s,a) = \max_{\pi \in \Pi} \left[r(s,a) + \gamma \sum_{s'} p(s'|s,a) v^\pi(s') \right] \tag{1.2.15}$$

Dado que el término $r(s,a)$ no depende de la política aplicada podemos extraer de la maximización:

$$q_*(s,a) = r(s,a) + \gamma \sum_{s'} p(s'|s,a) \max_{\pi \in \Pi} v^\pi(s') \tag{1.2.16}$$

Por otra parte, deberemos notar que la maximización la función $v^\pi(s')$ sobre todas las política $\pi \in \Pi$ es la definición de función valor óptima. Por lo que obtenemos una relación entre la función valor de estado óptima $v_*(s)$ y la función valor de estado-acción $q_*(s,a)$:

$$q_*(s,a) = r(s,a) + \gamma \sum_{s'} p(s'|s,a) v_*(s') \tag{1.2.17}$$

El lado derecho de la expresión anterior es exactamente le término que se maximiza en la ecuación de Bellman (1.2.8). Entonces si remplazamos $q_*(s,a)$ en la ecuación de Bellman obtenemos la relación entre $q_*(s)$ y $v_*(s)$:

Colorario 1.2.3 (Relación entre funciones valor óptimas) Podemos obtener $v_*(s)$ a partir de $q_*(s,a)$ de la siguiente manera:

$$v_*(s) = \max_{a' \in \mathcal{A}} q_*(s,a') \tag{1.2.18}$$

Por último, utilizando (1.2.18) y (1.2.17) obtenemos la ecuación de Bellman para la función valor de estado-acción:

Colorario 1.2.4 (Ecuación de Bellman estado-acción óptima)

$$q_*(s,a) = r(s,a) + \gamma \sum_{s'} p(s'|s,a) \max_{a' \in \mathcal{A}} q_*(s',a') \tag{1.2.19}$$

Estas leyes de recurrencia nos permitiran encontrar la función valor mediante la expresión (1.2.18) y mediante esta la política óptima π^*.

1.2.3. Operadores de Bellman (OB)

Los operadores de Bellman son una herramienta necesaria para comprender los algoritmos básicos de aprendizaje por refuerzo. Estos nos proveen teormeas fundamentales que nos aseguran la convergende de los algoritmos. Para ello seguiremos el discruso seguido en [Lazaric, 2013].

Denotaremos con la letra N para representa la cardinalidad del espacio de estados $\mathcal{S}$.

Definición 1.2.5 (OB de estado óptimo) Dado un un espacio vectorial $\mathbb{R}^N$ podemos definir el operador de Bellman $\mathcal{T} : \mathbb{R}^N \to \mathbb{R}^N$ como:

$$\mathcal{T}\mathcal{V}(s) = \max_{a \in \mathcal{A}} \left[r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) \mathcal{V}(s') \right] \tag{1.2.20}$$

□

Es fácil ver que V^π es punto fijo del operador $\mathcal{T}^\pi$, es decir que al aplicar el operador $\mathcal{T}^\pi$ al vector V^π obtenemos otra vez gracias a que V^π cumple la ecuación de Bellman (1.2.6). De la misma manera, V^* es punto fijo del operador $\mathcal{T}$ debido a que cumple la ecuación de Bellman (1.2.8).

Teorema 1.2.1 (Punto Fijo de Banach [Banach, 1922]) Dado un espacio vectorial $\mathcal{E}$ equipado con una norma $||.||$ y un operador contractivo $\mathcal{O} : \mathcal{E} \to \mathcal{E}$. Entonces existe un único punto fijo. Además empezando desde cualquier punto del espacio $v \in \mathcal{E}$ la sucesión:

$$\{v, \mathcal{O}(v)\ ,\ \mathcal{O}(\mathcal{O}(v))\ ,\ \mathcal{O}(\mathcal{O}(\mathcal{O}(v)))\ ,\ ...\} \tag{1.2.21}$$

converge al punto fijo. ■

El $\mathcal{T}_v$ cumple con la contracción en la norma L_∞. Es decir, para todo $\mathcal{V}_1, \mathcal{V}_2 \in \mathbb{R}^N$, si $\mathcal{V}_1 \leq \mathcal{V}_2$

$$||\mathcal{T}_v\mathcal{V}_1 - \mathcal{T}_v\mathcal{V}_2||_\infty \leq \gamma||\mathcal{V}_1 - \mathcal{V}_2||_\infty \tag{1.2.22}$$

Por lo tanto debido al teorema de punto fijo de Banach, empezando en cualquier estimación inicial de la función valor de estado $\mathcal{V}_0$, la aplicación sucesivas del operador de Bellman nos conduce inevitablemente al punto fijo del operador.

De la misma manera se puede definir los operadores de Bellman para la función valor de estado-acción $Q(s, a)$. Para ello denotaremos M como la cardinalidad del espacio de acciones.

Definición 1.2.6 (OB de estado-acción óptimo) Dado un un espacio vectorial $\mathbb{R}^{N\times M}$ podemos definir el operador de Bellman $\mathcal{T}_q : \mathbb{R}^{N\times M} \to \mathbb{R}^{N\times M}$ como:

$$\mathcal{T}_q\mathcal{Q}(s, a) = r(s, a) + \gamma \sum_{s'\in\mathcal{S}} p(s'|s, a) \max_{a'\in\mathcal{A}} \mathcal{Q}(s', a') \tag{1.2.23}$$

Estos dos operadores tienen propiedades análogas a OB que nos permiten decir que mediante la sucesivas aplicaciones del operador $\mathcal{T}_q$ para cualquier vector $\mathcal{Q}(s, a)$ terminaremos encontrando $q^*(s, a)$

1.3. Algoritmos de aprendizaje por refuerzo

Definido ya los operadores de Bellman somos capaces de entender algunos algoritmos de aprendizaje por refuerzo basado en el teorema de punto fijo, además elementos de los métodos de montecarlo. Presentaremos tres alogirmtos: *Value Iteration*, *Q-Iteration* y *Q-learning*.

Value Iteration

Gracias a que el operador de Bellman de estado $\mathcal{T}_v$ cumpla el teorema de punto fijo de Banach da lugar al algoritmo llamado *Value Iteration* [Sutton and Barto, 2018, cap. 4.4]. Este algoritmo requiere aplicar el operador de Bellman para para cada uno de los estados del sistema en cada iteración del algoritmo hasta que la variación de la función valor obtenida sea pequeña con respecto a la iteración anterior (*véase el algoritmo* 1).

Algoritmo 1 *Value Iteration*

procedure VALUE-ITERATION($\mathcal{V}^*, tol$)
  $k \leftarrow 0$
  $\mathcal{V}_k \leftarrow \mathcal{V}^*$
  while $error \leq tol$ **do**
    $k \leftarrow k + 1$
    for $\forall s \in \mathcal{S}$ **do**
      $\mathcal{V}_k(s) \leftarrow \mathcal{T}_v\mathcal{V}_{k-1}(s)$
    end for
    $error \leftarrow ||\mathcal{V}_k - \mathcal{V}_{k-1}||_\infty$
  end while
  $\pi_k^*(s) = \arg\max_{a'\in\mathcal{A}} \left[r(s, a') + \gamma \sum_{s'} p(s'|s, a')v_*(s') \right]$
  return: $[\mathcal{V}_k(s), \pi_k^*(s)]$
end procedure

Q-Iteration

De la misma manera que función valor de estado óptima $v_*(a)$, que la función valor estado-acción óptima $q_*(s, a)$ cumpla el teorema de punto fijo de Banach nos permite realizar el mismo procedimiento utilizado en el algoritmo anterior (*véase el algoritmo* 2).

Algoritmo 2 *Q-Iteration*

procedure Q-ITERATION($\mathcal{Q}^*, tol$)
  $k \leftarrow 0$
  $\mathcal{Q}_k \leftarrow \mathcal{Q}^*$
  while $error \leq tol$ **do**
    $k \leftarrow k + 1$
    for $\forall (s, a) \in \mathcal{S} \times \mathcal{A}$ **do**
      $\mathcal{Q}_k(s, a) \leftarrow \mathcal{T}_q \mathcal{Q}_{k-1}(s, a)$
    end for
    $error \leftarrow ||\mathcal{Q}_k - \mathcal{Q}_{k-1}||_\infty$
  end while
  $\mathcal{V}_k(s) = \max_{a \in \mathcal{A}} Q_k(s, a)$
  $\pi_k^*(s) = \arg\max_{a \in \mathcal{A}} Q_k(s, a)$
  return: $(\mathcal{Q}_k(s, a)\ ,\ \mathcal{V}_k(s)\ ,\ \pi_k^*(s))$
end procedure

Q-learning

El algoritmo de *Q-learning* (*véase algoritmo* 4) pretende encontrar una política óptima a través de un proceso a tiempo real y sin conocimiento de la dinámica del sistema. Este utiliza medidas en el estado siguiente s_{t+1} y la recompensa obtenida r_t debido a la acción a_t que tomemos. Si empezamos en un estado inicial $s_0 = s$ y una estimación inicial de la función valor estado-acción $\mathcal{Q}^*$, podemos elegir la acción a tomar en cada iteración de tiempo mediante esta estimación y medir la recompensa obtenida y el estado al que se ha transitado. Con ayuda de estas medidas podemos mejorar la estimación de la función valor estado-acción.

Para elegir la acción a tomar en cada momento se utiliza una estimación del la función valor de estado-acción $\mathcal{Q}_t$. La acción a tomar se escoge según la siguiente expresión:

$$a_t \leftarrow \arg\max_{a' \in \mathcal{A}} [\mathcal{Q}_t(s_t, a')] \tag{1.3.1}$$

Debido a que la función de valor de estado-acción es aproximada se toman en ciertas iteraciones una acción aleatoria con una pequeña probabilidad ϵ, que evita caer en minimos local.

Luego se puede actualizar la estimación de la función de valor estado acción mediante la expresión:

$$\mathcal{Q}(s_t, a_t) \leftarrow \underbrace{\left[r(s_t, a_t) + \gamma \max_{a' \in \mathcal{A}} \mathcal{Q}(s_{t+1}, a')\right]}_{\approx \mathcal{T}_q \mathcal{Q}(s_t, a_t)} \tag{1.3.2}$$

En principio deberiamos actualizar la función valor de estado-acción mediante la aplicación directa del operador $\mathcal{T}_q$, sin embargo esto requiere que conozcamos la distribución de probabilidad de transisicón $p(s'|s, a)$. Esto se solventa con la evaluación repetida de r y s' para una acción a. Entonces se realizar una media ponderada de los valores obtenidos para $Q(s, a)$. Esto se expresa de forma matemática como:

$$\mathcal{Q}(s_t, a_t) \leftarrow (1 - \alpha)\mathcal{Q}(s_t, a_t) + \alpha \left[r(s_t, a_t) + \gamma \max_{a' \in \mathcal{A}} \mathcal{Q}(s_{t+1}, a')\right] \tag{1.3.3}$$

Esto nos permite obtener una función valor de estado-acción aproximada que va mejorando a tiempo real y que permite actuar bajo una política desde el primer instante de tiempo.

Ejemplo 1.3.1 (Movimiento de una partícula en un potencial) Presentamos un problema de física clásica en el que existe una partícula en un potencial. El potencial elegido tiene la forma presentada en la figura 1.5. Este sistema queda caraterizado mediante la posición x y velocidad v en un instante de tiempo dado. Definiremos el problema dentro de una región acotada:

$$x \in [-6, 6] \quad v \in [-6, 6] \tag{1.3.4}$$

Algoritmo 3 *Q learning*

procedure Q-LEARNING($\mathcal{Q}^*, s_0, tol, \alpha, \epsilon$)
  $k \leftarrow 0$
  $\mathcal{Q} \leftarrow \mathcal{Q}^*$
  while $error \leq tol$ **do**
    $t \leftarrow t + 1$
    if $\epsilon > rand$ **then**
      $a_t \leftarrow$ acción aleatoria
    else
      $a_t \leftarrow \arg\max_{a' \in \mathcal{A}}[\mathcal{Q}_t(s_t, a')]$
    end if
    Actuar con a_t y medir r_t y s_{t+1}
    $\mathcal{Q}(s_t, a_t) \leftarrow (1-\alpha)\mathcal{Q}(s_t, a_t) + \left[r(s_t, a_t) + \gamma \max_{a' \in \mathcal{A}} \mathcal{Q}(s_{t+1}, a')\right]$
  end while
  return $\{a_t\}_{t>0}$
end procedure

Por otra parte, debido a que la formulación presentada esta en tiempo discreto, para resolver este problema necesitamos un esquema numérico que nos permita convertir la variable continua t a una variable discreta. Elegiremos el esquema de euler para obtener el sistema dinámico discreto.

$$\frac{d}{dt}\begin{bmatrix} x(t) \\ v(t) \end{bmatrix} = f(x(t), v(t), a(t)) \tag{1.3.5}$$

$$\underbrace{\frac{d}{dt}\begin{bmatrix} x(t) \\ v(t) \end{bmatrix} = \begin{bmatrix} v(t) \\ -\vec{\nabla}P(x(t)) - \frac{1}{2}x(t) + a(t) \end{bmatrix}}_{Tiempo\ continuo} \implies \underbrace{\begin{bmatrix} x_{t+1} \\ v_{t+1} \end{bmatrix} = \begin{bmatrix} x_t + \Delta t\ v_t \\ v_t + \Delta t\ [-\vec{\nabla}P(x_t) - \frac{1}{2}x_t + a_t] \end{bmatrix}}_{Tiempo\ discreto} \tag{1.3.6}$$

Donde se ha tomando como tamaño de paso $\Delta t = 0.0725$.

Además se ha discretizado el espacio de estados $\mathcal{S}$ y el de acciones $\mathcal{A}$ de la siguiente manera:

- $\mathcal{S} = \underbrace{\{-6.0000, -5.9195, \ldots, 5.9195, 6.0000\}}_{150\ puntos\ en\ x} \times \underbrace{\{-6.0000, -5.9195, \ldots, 5.9195, 6.0000\}}_{150\ puntos\ en\ v}$
- $\mathcal{A} = \{-20, -10, 0, 10, 20\}$

Entonces definimos la recompensa esperada como:

$$r(s, a) = r([x, v], a) = 100 \exp(-(x^2 + v^2)/0.5^2) \tag{1.3.7}$$

Dado en este caso el sistema es determinista, por lo que el operador de Bellman se puede escribir como:

$$\mathcal{T}\mathcal{V}(s) = 100 \exp(-(x^2 + v^2)/0.5^2) + \max_{a \in \mathcal{A}} \left[\mathcal{V}(f(s, a))\right] \tag{1.3.8}$$

Utilizaremos el algoritmo *Q-Iteration* y el algoritmo *Q-learning* para resolver el problema planteado. Además, con el fin de comparar los algoritmos de aprendizaje por refuerzo con metodologías que tienen en cuenta el conocimiento de la dinámica del sistema, resolveremos el siguiente problema de control óptimo:

$$\min_{a \in \mathbb{R}} \int_0^\infty (||s(t)||^2 + ||a(t)||^2) dt \tag{1.3.9}$$
$$sujeto\ a:$$
$$\dot{s} = As(t) + Bu(t)$$

Donde A y B son los jacobiandos de f con respecto al estado $s = [x, v]$ y la acción a, evaluados en $s = [0, 0]$ y $a = 0$. Este control tipo es llamado regulador cuadrático linear (LQR).

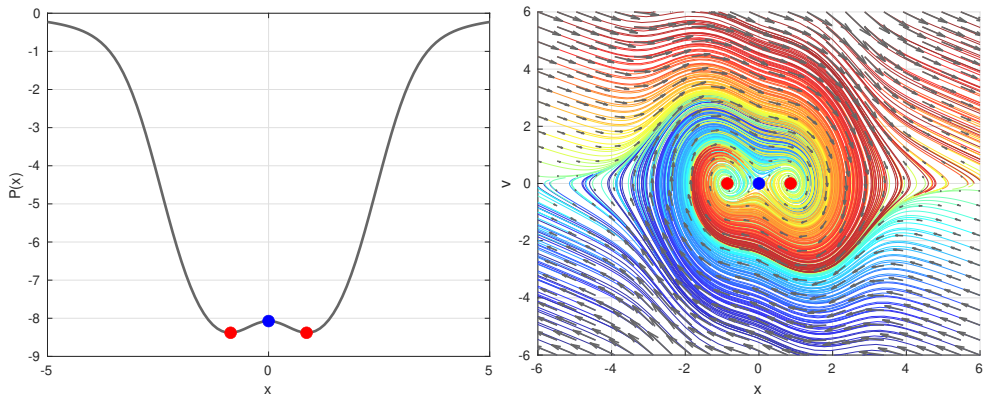


Figura 1.2: A la izquierda se muestra el potencial en el que se mueve la partícula del ejemplo. Además a la derecha se muestra el comportamiento del espacio de fases del sistema. Alli estan dibujadas distintas trayectorias dentro del espacio de fases representadas con distintos colores. Podemos ver como todas las trayectorias acaban en los atractores representados por los puntos rojos, mientras que las trayectorias escapan del repulsor representado por el punto azul.

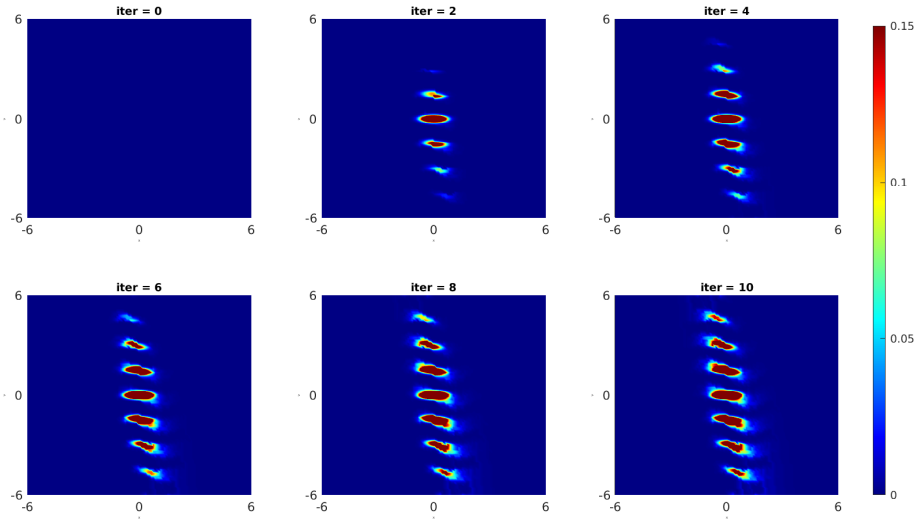


Figura 1.3: Evolución de la función valor $\mathcal{V}_k(x, v)$ en distintas iteraciones del algoritmo *Q Iteration*. Podemos ver la convergencia de función valor de estado.

Por último se presenta en la figura (1.3) y (1.4) se ha representado la evolución de la función valor y la política en distintas iteraciones del algoritmo, respectivamente. Finalmente en la figura (1.5) se compara el espacio de fases libre, el espacio de fases con la política obtenida por los algoritmos *Q-iteration* y *Q-learning* con resperco al espacio de fases obtenida mediante el regulador lineal cuadrático y el espacio de fases del sistema libre.

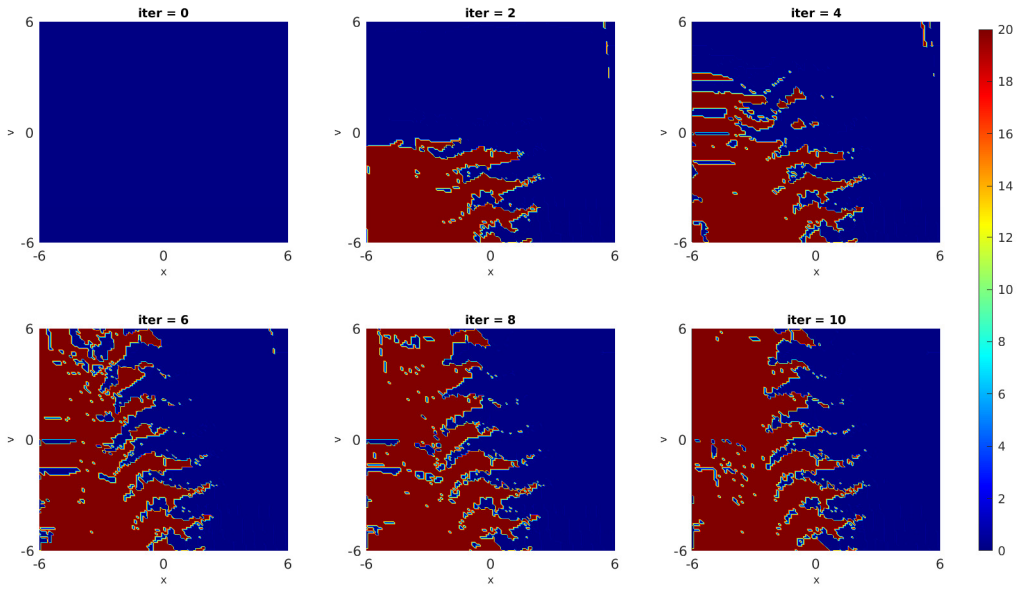
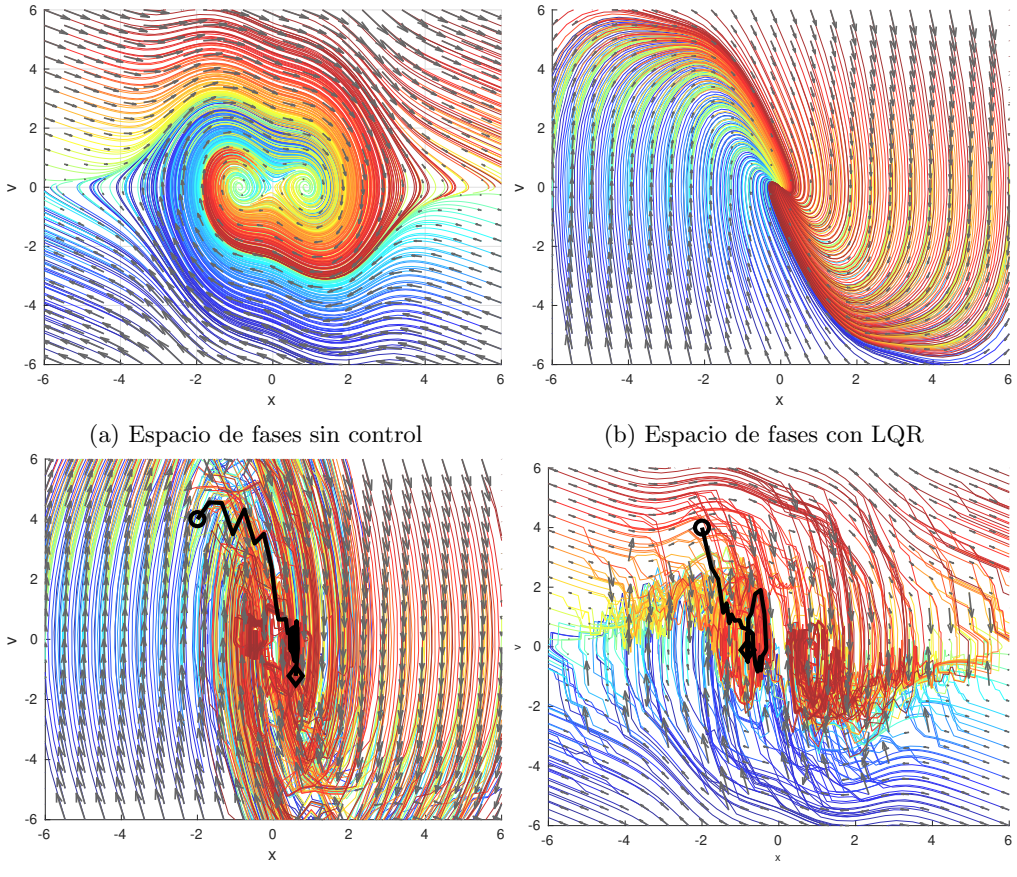


Figura 1.4: Evolución de la política valor $\pi^*(x, v)$ en distintas iteraciones. Dado un punto del espacio de fases (x, v) podemos obtener el valor de la acción.



(a) Espacio de fases sin control

(b) Espacio de fases con LQR

(c) Espacio de fases con *Value Iteration*

(d) Espacio de fases con *Q learning*

Figura 1.5: Comparamos el espacio de fase del sistemas con la política obtenida mediante LQR (figura 1.5b), la política obtenida mediante el algoritmo *Q-Iteration* (figura 1.5d)y la dinámica libre (figura 4.1a).

Capítulo 2

Introdución a Sistemas de recomendación

Un sistema de recomendación es un sistema de filtrado, capaz de predecir puntación que este da a un conjunto de elementos. El objetivo de estos sistemas es el proporcionar elementos con la mayor relevancia.

En este capítulo introduciremos algunos sistemas de recomendación mencionando sus ventajas y desventajas, además de motivar el uso de la modelización de un sistema de recomendación como un proceso oculto de Markov. Luego formulalemos matemáticamente el sistema de recomendación, para además de resolverlo y realizar pruebas experimentales.

Existen varios tipos de sistemas de recomendación, a continuación mencionamos un clasificación representativo de estos.

2.1. Filtrado colaborativo

En los sistemas de filtrado colaborativo [Schafer et al., 2007], las puntaciones predichas de los elementos se calculan mediante la valoración de otros usuarios. Suponiendo que tenemos una base de datos de n usuarios y m elementos a recomendar, los sistemas de recomendación colaborativo almacenan las puntaciones en una matriz $P \in \mathcal{M}_{n\times m}$. De esta forma, un elemento de la matriz Pij, corresponde a la puntación que el usuario i da al elemento j. Deberemos notar que esta matrix P no es completamente conocida sino que tiene muchos elementos desconocidos. De esta manera, el problema de recomendación se traduce a descubrir la totalidad esta matriz P.

- **Ventajas**
  - Interpretación de los resultados.
  - Fácil implementación.
- **Desventajas**
  - Depende de las puntuaciones subjetiva de las personas.
  - Su rendimiento disminuye cuando los datos son dispersos.

2.2. Filtrado basado en contenido

Los sistemas de recomendación basado en contenido [Lops et al., 2011] se utilizan el historial del usuario como base del proceso. Los elementos del conjunto de recomendación tienen asociado un vector de carácteristicas. Por ejemplo, en el contexto de un sistema de recomendación de películas, un elemento puede ser de distinto género o duración, estas características son codificadas en un vector de $\mathbb{R}^d$, donde d es el número de características consideradas. El usuario en estos sistemas elige las películas según sus gustos por lo que va escogiendo un conjunto de vectores que definen un subespacio vectorial. Entonces se considera una buena recomendación al elemento que más cerca este en el subespacio definido por el historial del usuario.

- **Ventajas**

- Independencia de Usuarios. Las recomendaciones solo dependen del perfil de usuario, por lo que se considera un solución personalizada.
- No es necesario que ningún usuario haya evaluado un elemento nuevo.

- **Desventajas**
  - Sobreespecialización. Cuando el historial del usuario es muy grande la solución de este sistema puede llegar a una configuración estacionaria, por lo que podemos llegar a recomendaciones siempre iguales
  - Nuevo usuario. Es necesario que el historial del usuario sea suficientemente representativo para poder realizar recomendaciones fiables.

2.3. Sistema de basado en un proceso de decición de Markov

En [Shani et al., 2005] se describe el procesos de recomendación como un proceso de descisión de Markov. El usuario se considera un sistema dinámico, en donde el estado viene representado por una secuencia de elementos que ha seleccionado. Además las acciones a tomar viene represetnado por la siguiente recomendación. Es decir, se considera que la siguiente recomendación depende de el historial reciente del usuario.

- **Ventajas**
  - Solución dinámica. Un sistema de decomendación basado en un proceso de Markov puede a una solución que depende del estado actual del usuario, dado que este *estado* puede variar en el tiempo la recomendación es dinámico.

- **Desventajas**
  - Gran necesidad de datos. Dado que la estimación se realiza mediante la máxima verosimilitud, esta necesita una gran cantidad de datos-

En [Shani et al., 2005] su validación se realiza a tiempo real mediante la interacción con usuarios reales por lo que no es necesario la simulación del usuario. En este trabajo proponemos una manera de utilizar el historial del usuario para la simulaciíon y además para dar una mejor inicialización al algoritmo de *Q-learning*.

Capítulo 3

Preprocesado del historial de películas

Tomaremos como ejemplo la base de datos *movieslens*[1], publicada en [Harper and Konstan, 2015]. Esta base de datos describe la calificación de 5 estrellas un servicio de recomendación de películas. Contiene 100836 clasificaciones en 9742 películas. Estos datos fueron creados por 610 usuarios entre el 29 de marzo de 1996 y el 24 de septiembre de 2018. Este conjunto de datos se generó el 26 de septiembre de 2018. Todos los usuarios seleccionados habían calificado al menos 20 películas.

A continuación explicaremos los pasos realizados para la obtención del historial de usuario, y luego explicaremos como agregaremos información para que sea posible el analisis de un sistema de recomendación

3.1. Descripción de la base de datos iniciales

Los datos están contenidos en las bases de datos:

Base de datos 3.1.1 (movies.csv) Esta es una base de datos donde se describe enlaza el identificador numérico de la película con el nombre y los generos a los que pertenece. A continuación se describe las variables de esta base de datos:

Variable	Descripción	Tipo	Ejemplo 1	Ejemplo 2
movieId	identificador	numérico	1	3431
title	nombre	string	Toy Story (1995)	Braveheart (1995)
genres	generos	string	Comedy—Horror	Comedy—Drama—Romance

□

Base de datos 3.1.2 (ratings.csv) Esta es una base de datos donde se registra para cada usuario y cada película que ha visto, la puntación que ha dado. A continuación se describe las variables de esta base de datos:

Variable	Descripción	Tipo	Ejemplo 1	Ejemplo 2
userId	identificador de usuario	numérico	311	3431
movieId	identificador de película	numérico	54	23
rating	puntación	numérico	3	5
timestamp	instante de tiempo	numérico	964983815	864913515

□

3.2. Union de base de datos de generos y puntaciones

Dado que queremos ver una recuerencia en el comportamiento de los usuario es muy dificil predecir las películas de manera indivudual. Podemos preguntarnos cuantas veces el usuario va a volver a mirar la misma película. Dado que la respuesta es muy pocas, los datos de entrenamiento para la predicción de la pélicula serán muy pequeñas. Si pensamos en una película como un elemento perteneciente a un espacio vectorial de caráterísticas podemos tener más datos de entrenamiento. Por esta razón, utilizando las bases de datos (3.1.1) y(3.1.2) creamos la base de datos (3.2.1)

[1] <http://grouplens.org/datasets/>

Base de datos 3.2.1 (movies_features.csv) Esta base de datos contiene las películas disponibles para recomendar. Cada registro de la base de datos contiene el identificador y los generos que a los que puede pertener las películas

Variable	Descripción	Tipo	Ejemplo 1	Ejemplo 2
movieId	numérico	Identificador	12	143
title	string	Nombre	"Toy Story (1995)"	"Jumanji (1995)"
(no genres listed)	¿pertenece?	boleano	1	0
Action	¿pertenece?	boleano	1	0
Adventure	¿pertenece?	boleano	1	0
Animation	¿pertenece?	boleano	1	0
...	...	...	...	...

□

3.3. Normalización de puntación de las película

Dado que la puntación de las películas es subjetiva, normalizaremos las puntaciones de cada usuario de manera que la peor película valorada dentro de la base de datos para cada usuario tendrá la puntación -1, mientras que la mejor valorada tendrá el valor de $+1$. De esta forma, evitamos la sobrevaloración o subvaloración de las películas dependiendo del usuario

Base de datos 3.3.1 (ratings_norm.csv) Esta es una base de datos (3.1.2) pero donde la puntación para cada usuario esta normalizada.

Variable	Descripción	Tipo	Ejemplo 1	Ejemplo 2
userId	identificador de usuario	numérico	311	3431
movieId	identificador de película	numérico	54	23
rating	puntación	numérico	-1	1
timestamp	instante de tiempo	numérico	964983815	864913515

□

Observación 3.3.1 En un entorno real esta normalización no es posible debido a que no podemos saber *a priori* cual será la mayor puntación que dará un usuario. En un momento dado tendremos una puntación maximan y otra minima pudiendo escalar las puntaciones. Sin embargo si en alguna iteración este máximo o minimo cambia cambiarán todas las puntaciones relativas, complicando el problema. Aunque en esta tesis no consideramos este paso, deberemos notar que en un entrono real este problema será importante.

3.4. Componentes principales para generos de películas

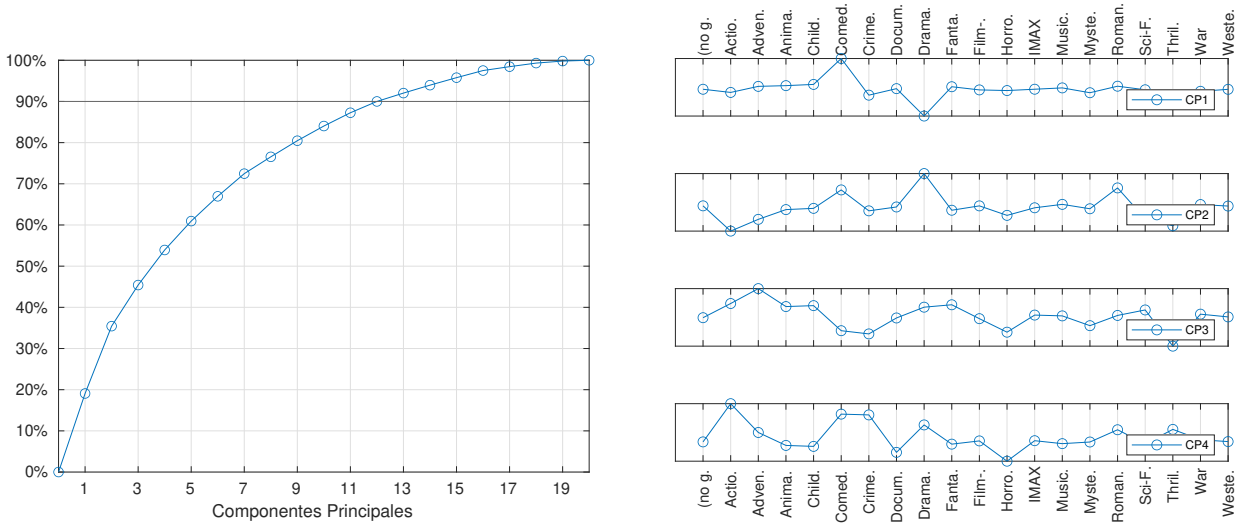
En esta base de datos (3.2.1) tenemos 24 generos de películas. Esta forma de clasificar las películas no tiene por que se la óptima a la hora de carácterizar cada película. Con el fin de reducir la dimensionalidad de la representación de cada película, procederemos al analisis de componentes principales. Esto lo realizaremos con ayuda de la función `pca()` perteneciente a la biblioteca ([MATLAB Machine Learning Toolbox, 2012]).

Nos quedaremos con las primeras componentes principales que nos den una variabilidad explicada del 90 %. En este caso en la figura (3.1a), se puede ver que el número de componentes principales para que esta condición se cumpla es 11.De esta manera podemos tranformar la base de datos (3.2.1) a la base de datos (3.4.1).

Base de datos 3.4.1 (movies_pca.csv) Esta base de datos con las componentes principales

Variable	Descripción	Tipo	Ejemplo 1	Ejemplo 2
movieId	numérico	Identificador	12	143
title	string	Nombre	"Toy Story (1995)"	"Jumanji (1995)"
PC01	Coordenada	numérico	0.1	-0.3
PC02	Coordenada	numérico	0.5	0.1
PC03	Coordenada	numérico	1.0	1.0
...	...	...	...	...
PC11	Coordenada	numérico	4.0	-3.1

□



(a) Variabilidad Explicada

(b) Primeras componentes

Figura 3.1: Analisis PCA en los generos de películas

3.5. Historial de aceptación para cada usuario

Para ello, cruzaremos la base de datos *movies.csv* y *rating.csv*, para crear una nueva base de datos que en sus variables contenga cada uno de los generos disponibles en la totalidad de los datos.

Base de datos 3.5.1 (user_movies_accept.csv) Esta es una base de datos que registra las interacciones de distintos usuarios con las distintas películas, además de informar del vector de caráterísticas de cada película vista.

Variable	Descripción	Tipo	Ejemplo 1	Ejemplo 2
userId	identificador de usuario	numérico	311	3431
rating	puntación	numérico	3	5
timestamp	instante de tiempo	numérico	964983815	864913515
PC01	Coordenada	numérico	0.1	-0.3
PC02	Coordenada	numérico	0.5	0.1
PC03	Coordenada	numérico	1.0	1.0
...	...	...	...	...
PC11	Coordenada	numérico	4.0	-3.1

□

Si consideramos que tenemos d géneros, entonces para cada usuario tenemos una secuencia de vectores $\boldsymbol{x}_t \in [0,1]^d \mid \forall t \in \{1,\dots,T\}$. Por último, si una película pertenece a varios géneros a la vez, el módulo del vector asociado será más grande que una película con menos géneros asociados. Dado que el módulo de un vector puede afectar a los modelos de prodicción normalizaremos todos los vectores $\boldsymbol{x}_d$.

Asi pues, el historial de un usuario se puede entender como la rotación de un vector $\boldsymbol{x}_t$:

$$\{\boldsymbol{x}_t\}_{t\geq 0} \in \mathbb{R}^d \mid ||\boldsymbol{x}_t|| = 1 \ \forall t \leq 0 \tag{3.5.1}$$

3.6. Generación sintética del historial de rechazo

El problema que tenemos en estos datos es que solo tenemos las películas que el usuario ha seleccionado en cada iteración, sin embago si consideramos el proceso de recomendación en cada iteración el usuario se le ha recomendado una serie de peliculas de las cuales solo tenemos constancia la película que ha seleccionado. Es por ello que consideraremos que en cada iteración el sistema de recomendación a ofrecido dos películas al usuario. Una de ella la ha aceptado y la otra la ha rechazado. Para crear el historial de películas rechazadas agregaremos de manera ficticia otra película que tengamos en el historial del usuario pero que tenga menor puntación que la película que ha selecionado en la realidad.

Base de datos 3.6.1 (user_movies_reject.csv) Base de datos con la misma estructura y la mismas instancias que la base de datos (3.5.1), pero esta contiene las películas que el usuario ha rechazado.

Variable	Descripción	Tipo	Ejemplo 1	Ejemplo 2
userId	identificador de usuario	numérico	311	3431
~~rating~~	-	-	-	-
timestamp	instante de tiempo	numérico	964983815	864913515
PC01	Coordenada	numérico	0.1	-0.3
PC02	Coordenada	numérico	0.5	0.1
PC03	Coordenada	numérico	1.0	1.0
...	...	...	...	...
PC11	Coordenada	numérico	4.0	-3.1

□

Observación 3.6.1 Para el historial de aceptación tambien tenemos su correspondiente puntación, sin embargo para el historial de rechazos dado que el usuario no lo ha selecionado, en general no tenemos puntación asociada.

3.7. Bases de datos finales: Historiales de usuarios

En la figura (3.2) mostramos un diagrama de flujo de como hemos creado los historiales de usuario que finalmente se utilizarán para entrenar un proceso de deción de Markov.

Finalmente para cada usuario tenemos: el historial de películas aceptadas y el historial de rechazadas.

- Tenemos la acciones en cada iteración:

$$\{\boldsymbol{a}_t\}_{t\geq 0} = \{\boldsymbol{x}_t^a, \boldsymbol{x}_t^r\}_{t\geq 0} \tag{3.7.1}$$

- Tenemos el estado en cada iteración:

$$\{\boldsymbol{s}_t\}_{t\geq 0} = \{\boldsymbol{x}_t^a\}_{t\geq 0} \tag{3.7.2}$$

Denotaremos al historial de un usuario como

$$\mathcal{H} = \{\boldsymbol{x}_t^r, \boldsymbol{x}_t^a\}_{t\geq 0} \tag{3.7.3}$$

Mostramos como ejemplo la siguiente tabla como historial de aceptación (base de datos 3.5.1):

title	rating	timestamp	Action	Adventure	...
"The Drop (2014)"	2	1.4795e+09	0	0	...
"Django Unchained (2012)"	3.5	1.4795e+09	0.57735	0	...
Ïnterstellar (2014)"	3	1.4795e+09	0	0	...
"The Drop (2014)"	2	1.4795e+09	0	0	...
"The Drop (2014)"	2	1.4795e+09	0	0	...
.Exit Through the Gift Shop (2010)"	3	1.4795e+09	0	0	...
Çollateral (2004)"	3.5	1.4795e+09	0.5	0	...

Además del historial de rechazo (base de datos 3.6.1):

title	rating	timestamp	Action	Adventure	...
"The Drop (2014)"	2	1.4795e+09	0	0	...
"Django Unchained (2012)"	3.5	1.4795e+09	0.57735	0	...
Ïnterstellar (2014)"	3	1.4795e+09	0	0	...
"The Drop (2014)"	2	1.4795e+09	0	0	...
"The Drop (2014)"	2	1.4795e+09	0	0	...
.Exit Through the Gift Shop (2010)"	3	1.4795e+09	0	0	...
Çollateral (2004)"	3.5	1.4795e+09	0.5	0	...

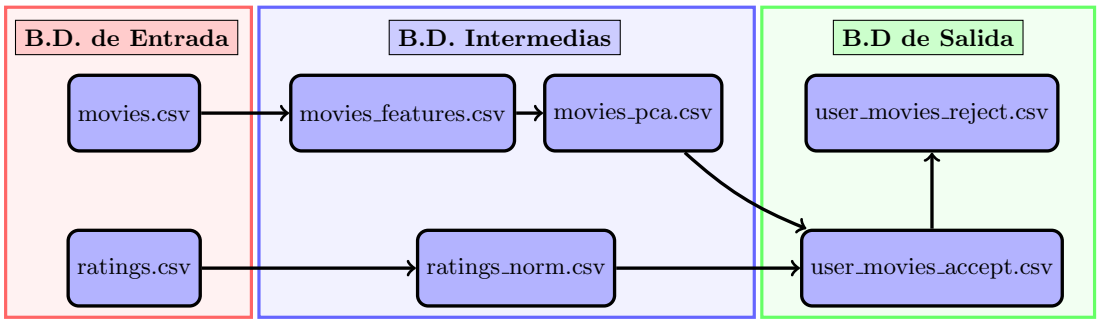


Figura 3.2: Diagrama de flujo para el preprocesado de los datos.
En rojo las base de datos obtenidas de [Harper and Konstan, 2015], en azul las bases de datos creadas como pasos Intermedias y finalmente en verde las bases de datos que se usarán en esta tesis.

Capítulo 4

Sistema de recomendación como proceso de decisión de Markov

En esta sección presentaremos un sistema de recomendación mediante una proceso de decisión de Markov y resolveremos el problema de recomendación mediante la metodología de aprendizaje por refuerzo.

4.1. Proceso de decisión de Markov

Definimos una proceso de decisión de Markov inspirado en [Shani et al., 2005]. Aunque distinto debido a caráteristicas de los datos disponibles.

- **Estado**: Pelicula vista en el instante anterior. Es espacio donde vive este estado es:

$$\mathcal{S} = \{x \in \mathbb{R} \mid ||x|| = 1\} \tag{4.1.1}$$

  Donde d es el número de generos considerados en la base de datos.

- **Acción**: Las dos películas que enseñamos al usuario. Entonces el espacio de acciones es

$$\mathcal{A} = \{\boldsymbol{a} \in \mathcal{S} \times \mathcal{S}\} \tag{4.1.2}$$

- **Recompenza**: Puntación que da a la película selecionada. Representado por $r_t \in [-1, 1]$. Suponemos que este puede depender del estado $\boldsymbol{x}_t$ y de la acción $\boldsymbol{a}_t$, sin embargo no sabemos que forma funcional tiene.

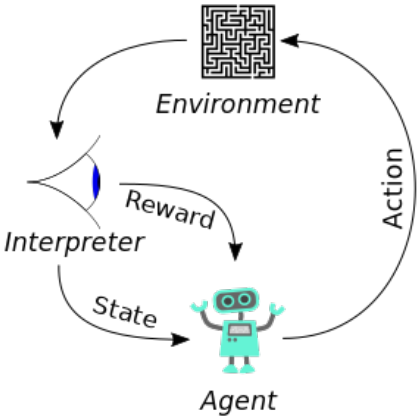
- **Ecuación dinámica**: Tal como se ha creado la base de datos, sugiere que la dinámica del usuario puede escoger la película según la puntación asociada. Sin embargo, en el plantamiento el usuario hasta despues de ver la película no tiene una puntación asociada por lo que la puntación a *priori* que tiene el usuario sobre dos películas es la que determina la eleción. Dado que en la metodología de aprendizaje por refuerzo no es necesario el conocimiento de la dinámica, no es necearia definirla.

Mostramos un esquema de la interación del usuario y el sistema de recomendación en la figura 4.1b

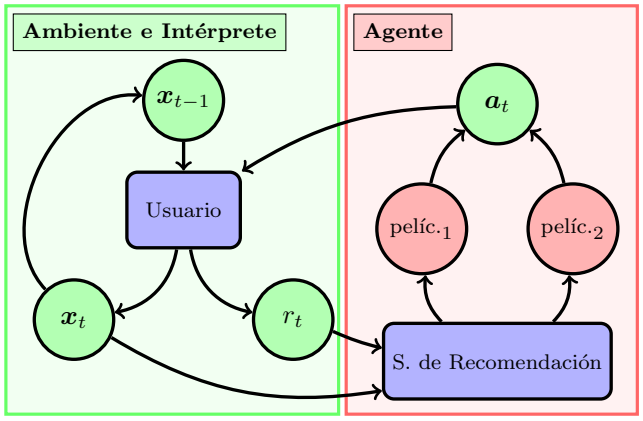
Observación 4.1.1 La acción en cada instante t es un par de vectores $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^d$, sin embargo en la realidad el sistema de recomendación deberá recomendar un título de película y no un vector de caráterísticas. Por esta razón, necesitamos una función que traduzca un vector de caráterísticas $\boldsymbol{x}$ en un titulo de la base de datos. Tomaremos como peliculas correspondiente a $\boldsymbol{x}$ la película más parecidad que tengamos en la base de datos y que no haya visto el usuario.

4.2. Plantamiento de los problemas

En esta tesis buscarémos políticas para realizar buenas recomendaciones de manreda que la puntación acumuladad del usuario sea máxima. Este problema puede ser planteado desde el momento en el que el usuario empieza a usar el sistema de recomendación o cuando tenemos ya historial suficiente del usuario para poder inferir una manera de actuar. En el primero de ellos, dado que no tenemos ninguna información del usuario deberemos utilizar los datos de los usuarios anteriores. Acontinuación plantemos los dos problemas.



(a) Esquema estándar del aprendizaje por refuerzo [Wikipedia, 2017]



(b) Esquema específico de aprendizaje por refuerzo en el sistema de recomendación.

Figura 4.1: Aprendizaje por refuerzo en el sistema de recomendación

Problema 4.2.1 (Sin historial) Suponiendo que tenemos un historial de aceptación $\{\boldsymbol{x}_t^a\}_{t\geq 0}$ y de rechazo $\{\boldsymbol{x}_t^r\}_{t\geq 0}$ del usuario. ¿Cómo podemos obtener una política $\pi : \mathcal{S} \to \mathcal{A}$ que nos maximize la recompensa acumulada?

Problema 4.2.2 (Con historial) Suponiendo que tenemos un base de datos de historial de aceptación $\{\boldsymbol{x}_t^a\}_{t\geq 0}$ y rechazo de varios usuarios $\{\boldsymbol{x}_t^r\}_{t\geq 0}$, pero no tenemos historial del usuario al que se le va a recomendar. ¿Cómo podemos obtener una política $\pi : \mathcal{S} \to \mathcal{A}$, que nos maximize la recompensa acumulada?

4.3. Metodología

En esta sección describiremos los pasos que se ha seguido para abordar los problema planteados en la sección anterior.

4.3.1. Selección de estado inicial

Dado que la condición incial se determina por una primera recomendación que no nos es posible determinar lo tomaremos como aleatorio. Luego buscaremos la política óptima que nos permita encontrar los datos de entrenamiento.

4.3.2. Inicialización de la función valor estado-acción

En cierto momento tenemos un historial de usuario, hacemos *Q-learning* mediante las acciones que tenemos disponibles. En [Fujimoto et al., 2019], se estudia el problema de aprendizaje por refuerzo cuando tenenemos los datos ya dados. Alli se ve que la convergencia del algoritmo *Q-learning* dentro de un conjunto de datos ya dados tambien converge a la política óptima.

Teorema 4.3.1 [Fujimoto et al., 2019]. *La realización de Q-learning mediante el muestreo de un conjunto de datos B converge a la función valor óptimo.*

Entonces

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha\left(r + \gamma \max_{a' s.t (s',a')\in\mathcal{B}} Q(s',a')\right) \tag{4.3.1}$$

La evaluación de la política se realizará mediante el uso de los datos restantes. Para ello se selecionará la acción óptima como la acción que nos mande la política encontrada, proyectado a los datos que tengamos en los datos restantes.

1. **Con historial(Problema 4.2.1)**. Tomaremos como datos de entrenamiento el historial del propio usuario.
2. **Sin historial(Problema 4.2.2)**. Tomaremos como datos de entrenamiento todo el conjunto de usuario suponiendo que todas las interacciones de la base de datos provienen

Algoritmo 4 *Batch Q-learning*

procedure BACTH Q-LEARNING($\mathcal{Q}^*, s_0, tol, \alpha, \epsilon$)
  $k \leftarrow 0$
  $\mathcal{Q}_1 \leftarrow \mathcal{Q}^*$
  while $error \leq tol$ **do**
    $k \leftarrow k + 1$
    Sample (s_t, a_t, s_{t+1}, r_t)
    $\mathcal{Q}_k(s_t, a_t) \leftarrow (1-\alpha)\mathcal{Q}_{k-1}(s_t, a_t) + \left[r_t + \gamma \max_{a' \in \mathcal{A}\ s.t.(s_{t+1}, a') \in \mathcal{B}} \mathcal{Q}_{k-1}(s_{t+1}, a')\right]$
    $error = ||Q_k - Q_{k+1}||^2$
  end while
  return $\{a_t\}_{t>0}$
end procedure

4.3.3. Política utilizada

Utilizaremos la metodología de *Q-learning* inicializando la función valor estado-acción como se ha mencionado en la sección (4.3.2)

4.3.4. Selección de peliculas desde un política dada

Dividiremos la base de datos generada en la sección (3.7). Teniendo en cuenta todos los usuarios ordenaremos temporalmente los datos $\{\boldsymbol{x}_t^r, \boldsymbol{x}_t^a\}$ para todos los usuarios y lo dividiremos por la mitad. De esta forma, existirán algunos usuarios que no hayan interacionado con el sistema de recomendación en toda la base de datos de entrenamiento y otros de los cuales si se tenga historial. De esta manera, podemos probar los problemas (4.2.1) y (4.2.2).

Entonces para un usuario tenemos

1. **Una historial de entrenamiento**: Para cada usuario tenemos:

$$\mathcal{H}^e = \{\boldsymbol{x}_t^a, \boldsymbol{x}_t^r\}_{t\geq 0}^e \tag{4.3.2}$$

   Esta se utilizará para inicializar la función valor estado-acción.

2. **Una historial de prueba**: Para cada usuario tenemos:

$$\mathcal{H}^p = \{\boldsymbol{x}_t^a, \boldsymbol{x}_t^r\}_{t\geq 0}^p \tag{4.3.3}$$

   que se utilizará para la simulación del usuario. Por otra parte, para la simulación de la reacción del usuario ante un política $\pi : \mathcal{S} \to \mathcal{A}$, proyectaremos la acción al conjunto de películas disponibles que no haya visto el usuario hasta el momento. Esta proyección al conjunto de películas disponibles se realiza mediante la metodología de vecinos proximos y utilizando la distancia euclidea del espacio de caráteristicas $\mathcal{S} = \mathbb{R}^d$.

   De esta manera, si $\mathcal{O}$ es el conjunto de películas que el usuario no ha visto, y el sistema nos recomienda una película con vector $\boldsymbol{x} \in \mathbb{R}^d$ entonces deberemos escoger la película $p \in \mathcal{O}$ tal que:

$$\mathrm{p} = \arg\min_{p \in \mathcal{O}} ||x_p - x||^2 \tag{4.3.4}$$

4.4. Resultados numéricos

4.4.1. Políticas de Referencia

Es necesario fijar políticas de referencias que nos permitan valorar si nuestras política de recomendación es efectiva. Por esta razón creasmos una política aleatoria y además dos políticas fíticas que nos darán mejor entendimiento del proceso.

- **Política aleatoria** (π^{ale}): Este política nos dará un referencia de ruido. Nuestro sistema de recomendación deberá ser mejor que dar las recomendaciones de las películas de manera aleatoria.

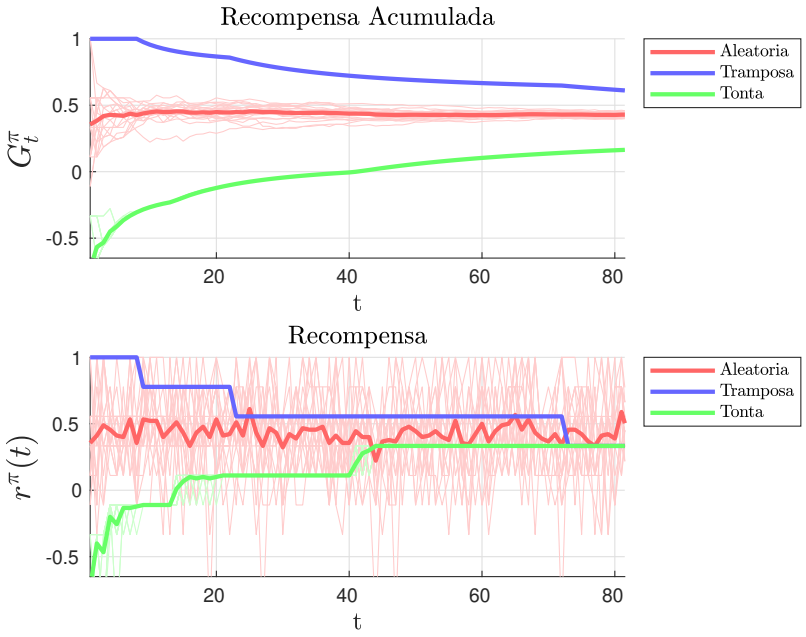


Figura 4.2: Políticas de referencias para un usuario concreto.
Dado el caráter estocástico de las políticas, se dibuja la media de 20 experimentos.

- **Política tramposa** (π^{tram}): Esta política da las mejores recomendaciones posibles. Dado que en la base de datos de prueba tenemos las puntaciones para cada película, la política tramposa mira la película que más puntación tiene y se la ofrece. En la realidad esto es imposible dado que no podemos saber la puntación que tiene una peliculas antes de que el usuario la vea.
- **Política tonta** (π^{ton}): Esta política actua de la misma manera que la política tramposa, sin embargo ofrece las pelícuas con menor puntación. De esta manera, esta política es la peor que podemos realizar.

Utilizaremos la recomensa en el tiempo r_ty la recomensa acumulada G_t^π en el tiempo para comparar políticas. Una comparación entre estas tres políticas de referencia se puede ver en la figura (4.2)

4.4.2. Caso 1: Usuario sin historial previo

4.4.3. Caso 2: Usuario con historial previo

Capítulo 5

Conclusiones

Bibliografía

[Banach, 1922] Banach, S. (1922). Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales. *Fund. math*, 3(1):133–181.

[Bellman, 1954] Bellman, R. (1954). The theory of dynamic programming. Technical report, Rand corp santa monica ca.

[Bellman, 1957] Bellman, R. (1957). A markovian decision process. *Indiana Univ. Math. J.*, 6:679–684.

[Fujimoto et al., 2019] Fujimoto, S., Meger, D., and Precup, D. (2019). Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pages 2052–2062.

[Harper and Konstan, 2015] Harper, F. M. and Konstan, J. A. (2015). The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4).

[Lazaric, 2013] Lazaric, A. (2013). Decision Processes and Dynamic Programming. *Course on Reinforcement Learning*, pages 160–183.

[Lops et al., 2011] Lops, P., De Gemmis, M., and Semeraro, G. (2011). Content-based recommender systems: State of the art and trends. In *Recommender systems handbook*, pages 73–105. Springer.

[MATLAB Machine Learning Toolbox, 2012] MATLAB Machine Learning Toolbox (2012). Matlab pca function (https://www.mathworks.com/help/stats/pca.html).

[Schafer et al., 2007] Schafer, J. B., Frankowski, D., Herlocker, J., and Sen, S. (2007). Collaborative filtering recommender systems. In *The adaptive web*, pages 291–324. Springer.

[Shani et al., 2005] Shani, G., Heckerman, D., and Brafman, R. I. (2005). An mdp-based recommender system. *Journal of Machine Learning Research*, 6(Sep):1265–1295.

[Sutton and Barto, 2018] Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

[Wikipedia, 2017] Wikipedia (2017). Reinforcement learning diagram (https://commons.wikimedia.org/wiki/File:Reinforcement_learning_diagram.svg).