UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

MÁSTER UNIVERSITARIO EN INGENIERÍA DE TELECOMUNICACIÓN

TRABAJO FIN DE MÁSTER

ESTUDIO Y DESARROLLO DE ALGORITMOS DE APRENDIZAJE POR REFUERZO A PARTIR DE LA TEORÍA DE OPTIMIZACIÓN DUAL

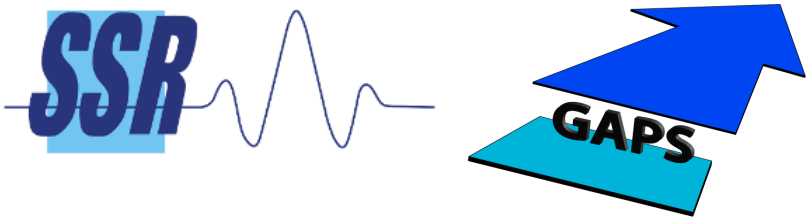
DANIEL M. GARCÍA-OCAÑA HERNÁNDEZ

2017

ESTUDIO Y DESARROLLO DE ALGORITMOS DE APRENDIZAJE POR REFUERZO A PARTIR DE LA TEORÍA DE OPTIMIZACIÓN DUAL

Daniel M. García-Ocaña Hernández

Tutor: D. Santiago Zazo Bello



Grupo de Aplicaciones de Procesado de Señales

Departamento de Señales, Sistemas y Radiocomunicaciones

Escuela Técnica Superior de Ingenieros de Telecomunicación

Universidad Politécnica de Madrid

Junio 2017

TRABAJO FIN DE MÁSTER

Título:	Estudio y desarrollo de algoritmos de aprendizaje por refuerzo a partir de la teoría de optimización dual
Autor:	D. Daniel M. García-Ocaña Hernández
Tutor:	D. Santiago Zazo Bello
Departamento:	Señales, Sistemas y Radiocomunicaciones

MIEMBROS DEL TRIBUNAL

Presidente:	D. .....
Vocal:	D. .....
Secretario:	D. .....
Suplente:	D. .....

FECHA DE LECTURA: _____

CALIFICACIÓN: _____

Resumen

Uno de los problemas más importantes en el campo de aprendizaje automático es la toma de decisiones para conseguir un objetivo. Este tipo de problemas se suele denominar aprendizaje secuencial, y uno de los marcos de trabajo más prometedores es el denominado aprendizaje por refuerzo. Históricamente, los algoritmos de aprendizaje por refuerzo han sido motivados principalmente estableciendo conexiones con la teoría de control óptimo.

En la actualidad existen diversos algoritmos de aprendizaje por refuerzo que implementan programación dinámica estimando la función valor del problema de forma estocástica, es decir, a partir de muestras y sin conocimiento previo de los datos del problema. Aunque estos algoritmos son completamente válidos, resulta interesante buscar nuevas interpretaciones que den lugar a nuevos algoritmos capaces de mejorar el desempeño de los ya existentes.

El objetivo de este Trabajo de Fin de Máster será el estudio y desarrollo de nuevos algoritmos de aprendizaje por refuerzo desde el punto de vista de la teoría de optimización. Para ello, se llevará a cabo una revisión de los fundamentos sobre los que se construye el campo del aprendizaje por refuerzo: los procesos de decisión de Markov y las ecuaciones de Bellman. Seguidamente, se hará una breve presentación de los conceptos fundamentales relativos a optimización convexa y teoría de la dualidad, y finalmente, se estudiará el estado del arte en lo que a los algoritmos de aprendizaje por refuerzo se refiere.

Una vez completada la fase de revisión de estas teorías, se investigará la conexión entre el problema de control óptimo y su formulación como un problema de programación lineal, con la intención de desarrollar nuevos algoritmos de optimización estocástica que resuelvan el problema de aprendizaje por refuerzo. De este modo, se derivará el problema dual asociado a dicho programa lineal, y será precisamente a partir de esta formulación dual de la que se deriven importantes propiedades que darán pie a tomar un nuevo enfoque diferente de lo estudiado hasta ahora: la búsqueda en el espacio de políticas a través de la variable dual.

Por último, se formalizarán dos algoritmos que empleen el método primal-dual desarrollado: uno para la resolución de problemas de pequeña escala, y su correspondiente extensión a problemas de gran escala, y se realizarán diversas pruebas de convergencia y comparación de resultados con algunas de las técnicas más empleadas en la actualidad, con el objetivo de evaluar la utilidad e idoneidad del algoritmo formulado.

Palabras clave

Control óptimo, programación dinámica, aprendizaje por refuerzo, proceso de decisión de Markov, teoría de control, gradiente en la política, dualidad, aprendizaje máquina, aprendizaje secuencial, ecuación de Bellman.

Abstract

One of the most important problems in machine learning is the task of decision making in order to achieve a target. This problem is usually referred to as sequential learning, and one of the most promising working frameworks is the so called reinforcement learning. Historically, reinforcement learning algorithms have been driven by establishing connections with optimal control theory.

Currently, there exist several reinforcement learning algorithms implementing dynamic programming by estimating the value function of the problem in an stochastic manner, i.e., from samples and without knowledge of problem's environment. Even though these algorithms are completely valid, it seems interesting to seek for new interpretations which can lead to new algorithms able to improve the performance of existing ones.

The aim of this master's thesis is the study and development of new reinforcement learning algorithms from the optimization theory approach. With that goal in mind, a review of reinforcement learning basis will be initially done, covering Markov decision processes and Bellman equations. Following, fundamentals of convex optimization and duality will be introduced and finally, it will be studied the state of the art, in terms of reinforcement learning algorithms.

Once this theoretical review had been completed, it will be investigated the connection between optimal control problem and its formulation as a linear program, with the intention of developing new stochastic optimization algorithms which solves reinforcement learning problem. After having this linear program form, the dual problem associated with this optimization problem will be formulated, and it will be from the dual form, from which important properties will be obtained. These properties will be the basis of a new approach, different from that currently studied in this field: the search in the policy space through dual variable.

Finally, two algorithms using this primal-dual method will be formalized: one oriented to solve small scale problems, and its extension to big scale problems. In order to evaluate the performance of this novel algorithms, a batch of typical reinforcement learning problems will be used, and the results will be compared with those obtained when solving the same problems with some of the most used algorithms nowadays.

Keywords

Optimal control, dynamic programming, reinforcement learning, Markov decision process, control theory, policy gradient, duality, machine learning, sequential learning, Bellman equation.

Agradecimientos

En primer lugar quiero agradecer a Santiago la oportunidad que me ha dado de introducirme en este campo de la inteligencia artificial que ha despertado tanto interés en mi, y a Sergio todo el tiempo que ha dedicado a ayudarme a resolver los problemas que me iban surgiendo. Sin vosotros no habría sido posible llevar a cabo este gran trabajo que hemos realizado.

De igual manera, quiero darle las gracias a varias personas que han estado siempre presente, en las duras y en las maduras, tanto estos dos años de máster como todos los años anteriores desde que les conocí en el instituto: la gente de Le Groupè. Aunque nos hayamos podido distanciar más o menos en estos dos años, siempre que he necesitado un respiro, quejarme o celebrar algo bueno que me haya podido pasar habéis estado ahí. Diego, Romo, Gabri, Miguel, Héctor, Iván, Jony: gracias por estar ahí. Incluso muy a mi pesar, tengo que darle las gracias a Sergio, que con nuestros más y nuestros menos, se ha portado bien cuando lo he necesitado.

Igual de importante es de agradecer el apoyo de mis queridos «vesinos»: Puri, Antonio, y especialmente Nacho y Joaquín. Gi, Juako, habéis sufrido y disfrutado conmigo prácticamente toda mi vida. Desde luego que muchas veces no habría sacado las ganas de seguir si no fuese por el apoyo de gente como vosotros, que ha confiado en que era capaz de hacerlo y que valía para esto. Muchas gracias.

Por último pero no menos importante, agradecer el apoyo continuo de mi familia, de mis padres y mi hermano. Vosotros os habéis llevado la peor parte de la historia: aguantar mis frustraciones en casa. Después de estos 6 años, quiero pediros perdón por las manías que me habéis tenido que aguantar y daros las gracias por el apoyo incondicional que me habéis ofrecido. Gracias por haberme dado todo el cariño que teníais dentro y todo vuestro tiempo.

Daniel M. García-Ocaña Hernández.
2017.

Índice general

Índice de figuras

Índice de tablas

Glosario

- **DP:** programación dinámica (*Dynamic Programming*)
- **RL:** aprendizaje por refuerzo (*Reinforcement Learning*)
- **MP:** proceso de Markov (*Markov Process*)
- **MDP:** proceso de decisión de Markov (*Markov Decision Process*)
- **PI:** mejora de la política (*Policy Improvement*)
- **PE:** evaluación de la política (*Policy Evaluation*)
- **TD:** diferencia temporal (*Temporal Difference*)
- **MC:** Monte-Carlo
- **LP:** programación lineal (*Linear Program*)
- **BDA-MB:** Bellman-ascenso dual basado en modelo (*Bellman-Dual Ascent - Model Based*)
- **BDA-MF:** Bellman-ascenso dual libre de modelo (*Bellman-Dual Ascent - Model Free*)
- **BDALA-MF:** Bellman-ascenso dual con aproximación lineal libre de modelo (*Bellman-Dual Ascent with Linear Approximation - Model Free*)

Capítulo 1

Introducción

1.1. Motivación

Uno de los problemas que más se ha planteado en los últimos años ha sido cómo un agente puede aprender a predecir y controlar la respuesta de su entorno. Esta disciplina se conoce comúnmente como *control óptimo estocástico*. Cuando el modelo del entorno es conocido, se dice que entonces el agente puede *planificar* su secuencia de decisiones óptimas. La programación dinámica (*Dynamic Programming*, DP) es una herramienta muy útil para resolver este tipo de problemas de planificación. Sin embargo, hay muchos casos en los cuales el modelo es desconocido. En estas situaciones, más que planificar, se dice que el agente tiene que *aprender* de la interacción con el entorno. De este modo, el agente aspirará a predecir cómo reaccionará el entorno a cada acción que se tome en cada posible estado y en consecuencia descubrir qué acciones dan lugar a la respuesta más favorable. Dicho «descubrimiento» se realizará mediante la prueba de diferentes acciones en las mismas situaciones, repetidas veces.

Este último problema de aprendizaje es lo que se conoce como aprendizaje por refuerzo (*Reinforcement Learning*, RL) ya que el agente tiende a reforzar aquellas acciones que producen la respuesta más favorable. En el caso general, el entorno cambiará de estado en función del estado actual en que se encuentre y la acción escogida por el agente. Por tanto, la acción del agente no sólo afectará a la respuesta inmediata de la interacción actual, sino que también tendrá cierta repercusión en todas las respuestas futuras. Un enfoque clásico para abordar el problema de aprendizaje por refuerzo es a través de los procesos de decisión de Markov (*Markov Decision Process*, MDP), pues permiten modelar de manera sencilla la interacción entre el agente y su entorno.

Con el objetivo de poder resolver el problema de RL, se dará por hecho que el agente es capaz de percibir el estado del entorno y de tomar acciones para alterarlo, obteniendo como resultado alguna recompensa. Además, las transiciones de un estado del entorno a

otro estarán gobernadas por alguna regla decisoria. De este modo, podemos definir los siguientes elementos principales del problema de RL:

1. los observables vistos por el agente, O_t, que indican el estado del entorno, S_t, el cual pertenece a algún conjunto de estados $\mathcal{S}$;
2. las acciones tomadas por el agente, A_t, pertenecientes a algún conjunto de acciones $\mathcal{A}$;
3. y las recompensas proporcionadas por el entorno, $R_t \in \mathbb{R}$.

Los conjuntos de estados y acciones podrán ser, bien discretos o bien subconjuntos del espacio de vectores reales. La recompensa R_t es una señal escalar de realimentación que indica cómo de bien se ha comportado el agente en el instante t. El objetivo del agente será por tanto maximizar la recompensa total acumulada a largo plazo, asumiendo la llamada «hipótesis de la recompensa»: todos los objetivos que se quieran alcanzar en un problema deberán poder ser descritos en términos de la maximización de la recompensa total acumulada esperada. Por ejemplo, si el objetivo del agente es escapar de un laberinto en el menor tiempo posible, entonces el agente podría obtener una recompensa negativa en cada instante t de tiempo y un recompensa nula sólo cuando encuentra la salida.

En la figura 1.1 se muestra, de manera esquemática, la interacción entre el agente y el entorno.

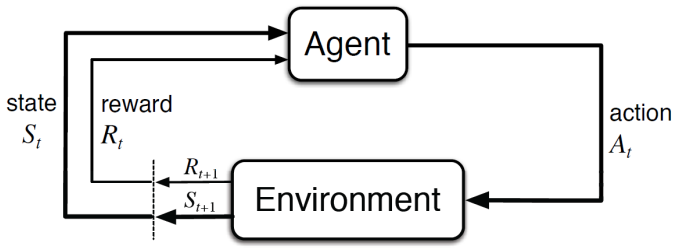


Figura 1.1: Interacción agente-entorno en aprendizaje por refuerzo [1].

La recompensa total acumulada que se esperaría obtener empezando desde un estado concreto del entorno es lo que en terminología de teoría de control y MDPs se conoce como función valor de estados. Por tanto, para maximizar la recompensa total acumulada a largo plazo, el agente deberá buscará transitar a aquellos estados cuya función valor sea mayor. El conjunto de decisiones que lleven al agente a actuar de manera óptima en cada uno de los estados del entorno de nuestro problema será en consecuencia la política de comportamiento (o de control) óptima.

Cualquier algoritmo que sea capaz de aprender a partir la interacción con el entorno con el fin de alcanzar un objetivo determinado, podrá ser considerado un método de aprendizaje por refuerzo.

En base a todo lo anterior, parece natural pensar que hasta ahora, el enfoque predominante en los algoritmos de RL haya sido el de interactuar con el entorno a fin de estimar la función valor asociada a cada estado del conjunto de estados $\mathcal{S}$ de nuestro problema, para posteriormente extraer la política que maximiza la recompensa total acumulada. Es decir, de alguna manera, estos algoritmos hacen una búsqueda en el espacio de las funciones valor.

La motivación principal de este trabajo de fin de máster es el estudio de un enfoque alternativo que permita encontrar la solución al problema de control planteado explorando directamente el espacio de políticas definido por el entorno desconocido, a partir de la interacción con el mismo. Con ello, se buscará reducir el tiempo necesario por el agente para aprender la política de comportamiento óptima, o lo que es lo mismo, el tiempo que tardamos en alcanzar la recompensa máxima. Este tipo de métodos que permiten estimar la política se conocen en la literatura como métodos de gradiente en la política (*policy gradient methods*), y hasta el momento han sido poco desarrollados.

1.2. Objetivo y estructura del documento

El objetivo de este trabajo será la formulación de un algoritmo de aprendizaje por refuerzo novedoso que base su comportamiento en la exploración del espacio de políticas a través de la interacción con el entorno. Para llegar a dicha formulación, se abordará el problema de aprendizaje por refuerzo desde el punto de vista de la teoría de optimización, más concretamente haciendo uso de la dualidad de Lagrange. Primero se estudiará la problemática en que el espacio de acciones es pequeño, para posteriormente, extender estas ideas a problemas en los que el espacio de acciones es muy grande o incluso continuo.

De acuerdo a estos objetivos, el documento se estructura de la siguiente manera:

- Capítulo 1: Introducción. Presenta la motivación para la realización de este trabajo así como el objetivo que se busca conseguir.

- Capítulo 2: Estado del arte. Se da una visión general de la situación actual en lo que a algoritmos de aprendizaje por refuerzo se refiere.

- Capítulo 3: Contexto del problema. En este capítulo se pone en contexto el problema que se desea resolver, definiendo cada una de las partes que lo componen. Una vez conocida la problemática, se hace una clasificación general de los métodos existentes para resolver dicho problema.

Los capítulos que van del 4 al 8 constituyen los fundamentos teóricos en los que se basa este trabajo:

- Capítulo 4: Procesos de decisión de Markov. Este capítulo introduce los procesos de decisión de Markov, elemento fundamental que permitirá modelar el problema de aprendizaje por refuerzo. Se presentan además las ecuaciones de Bellman que como se verá, jugarán un papel fundamental.

- Capítulo 5: Resolución de las ecuaciones de Bellman en problemas de pequeña escala. En este capítulo se detallan dos técnicas ampliamente usadas en la resolución de las ecuaciones de Bellman cuando los problemas tienen baja dimensionalidad: programación dinámica y diferencias temporales. Ambas técnicas serán de vital importancia en el desarrollo del nuevo algoritmo que se plantea en este documento.

- Capítulo 6: Resolución de las ecuaciones de Bellman en problemas de gran escala. En este capítulo se aborda la resolución de las ecuaciones de Bellman cuando la dimensionalidad del problema es mayor, mediante la aproximación de funciones, dando lugar a las ecuaciones de Bellman proyectadas. Se presenta una versión del método de las diferencias temporales detallado en el capítulo 5 adaptado a la nueva situación, que cobrará especial relevancia cuando se extienda el algoritmo desarrollado a problemas de gran escala.

- Capítulo 7: Optimización convexa. Se hace un breve resumen de las principales herramientas de optimización convexa empleadas para derivar el nuevo algoritmo.

- Capítulo 8: Dualidad. Como continuación del capítulo anterior, se presenta la teoría de la dualidad, la cual será el núcleo del algoritmo obtenido en este trabajo.

Los capítulos que van del 9 al 11 recogen la derivación y formulación del algoritmo novel desarrollado, tanto desde el punto de vista teórico como de implementación. Para cada una de las implementaciones realizadas, una para problemas de pequeña escala y otra para problemas de gran escala, se evaluará el algoritmo mediante la resolución de una serie de problemas típicos en el campo de aprendizaje por refuerzo y se compararán dichos resultados con los obtenidos al emplear algunos de los los algoritmos que constituyen el estado del arte actual.

- Capítulo 9: Ecuaciones de Bellman y teoría dual. A partir de las ideas expuestas en los capítulos 4–8 se desarrolla un nuevo método para resolver las ecuaciones de Bellman haciendo uso de la teoría dual.

- Capítulo 10: Desarrollo de un algoritmo primal-dual novel. Se presenta un algoritmo para la resolución de problemas de aprendizaje por refuerzo basado en el nuevo método expuesto en el capítulo 9. Este capítulo se centra únicamente en problemas de pequeña escala.

- Capítulo 11: Extensión del algoritmo BDA-MF a problemas de gran escala. En este capítulo se extiende el método del capítulo 9 y el algoritmo desarrollado en el capítulo 10 al caso en que el problema adquiere dimensionalidad mayor en los conjuntos de estados y/o acciones.
- Capítulo 12: Conclusiones y líneas de trabajo futuro.
- Anexos: Recogen las demostraciones de los teoremas, proposiciones y lemas empleados en el capítulo 9 para la formulación del nuevo método de resolución de las ecuaciones de Bellman a partir de la teoría dual.
- Referencias.

Capítulo 2

Estado del arte

El aprendizaje por refuerzo es una rama del aprendizaje máquina o *machine learning* en la que se lleva trabajando desde hace años debido al interés que suscita su aplicación en situaciones cotidianas de la vida real. No obstante, en los últimos años ha ganado más fuerza como consecuencia de los avances que se han producido en cuanto a capacidad computacional y a que el mercado está demandado soluciones que utilicen esta tecnología (ej.: robots que puedan atender las necesidades de los clientes o coches autónomos capaces de conducir solos). En consecuencia, se ha puesto mayor interés en desarrollar algoritmos sólidos que permitan resolver problemas de esta naturaleza, y han surgido una serie de métodos que se han convertido en los más empleados a día de hoy. A grandes rasgos, los algoritmos existentes se pueden dividir en dos grupos según la dimensión que tenga el problema a tratar:

1. Orientados a problemas con un conjunto pequeño y discreto de estados: cuando se tienen estas condiciones, la dimensionalidad es aún razonable y se puede disponer de una tabla que almacene las funciones valor de cada estado del conjunto de estados $\mathcal{S}$ y la política obtenida. Los algoritmos más extendidos para resolver este tipo de problemas de RL son SARSA, Q-learning y doble Q-learning. Por supuesto, existen variantes de estos tres algoritmos que introducen ligeras mejoras orientadas a la reducción de varianza en la estimación o a la corrección del sesgo. No obstante, su comportamiento es similar al de sus versiones primigenias.

2. Orientados a problemas con un conjunto grande o continuo de estados: cuando el problema adquiere una dimensionalidad mayor, deja de ser factible la idea de almacenar las funciones valor y la política en una tabla, pues consumirían toda la memoria disponible; en consecuencia, los algoritmos anteriores dejan de tener validez. Es por ello que se hace necesario emplear aproximaciones que generalicen la función valor para subconjuntos de estados cercanos entre sí, y permitan por tanto

reducir la dimensionalidad del problema. En función del tipo de aproximación que se emplee, podremos encontrar las siguientes soluciones:

a) Aproximación lineal: los algoritmos más empleados cuando se utilizan aproximaciones lineales de la función valor son *Least-Squares Policy Iteration* (LSPI), *Gradient Temporal Difference* (GTD) y GTD versión 2 (GTD2).

b) Aproximación no lineal mediante redes neuronales: de reciente aparición debido al resurgimiento de las redes neuronales, los algoritmos existentes cuando se emplea este tipo de aproximaciones son Deep Q-networks (DQN) y Neural Fitted Q-iteration (NFQ).

Todos los algoritmos anteriores entran dentro de la categoría de los llamados métodos basados en la función valor (*value-based methods*). En lo que respecta a los algoritmos basados en el gradiente de la política, aún no existe mucha literatura escrita más allá de los métodos actor-crítico, los cuales únicamente se emplean cuando se usa aproximación de funciones, y algunas publicaciones presentando ideas teóricas al respecto [3][4]. Por tanto, queda aún mucho terreno que explotar en lo que a los métodos basados en el gradiente de la política se refiere, y será lo que se pretenda conseguir con este trabajo.

Dado que por limitaciones de tiempo resultará imposible contrastar el algoritmo que se desarrolle con todos los citados anteriormente, los resultados obtenidos se enfrentarán únicamente a aquellos que se han considerado más representativos y que son más empleados a día de hoy: SARSA, Q-learning y LSPI.

Capítulo 3

Contexto del problema

3.1. El problema de decisión secuencial

Tal y como se presentó en la Introducción, uno de los problemas que más se ha planteado en los últimos años ha sido cómo un agente puede aprender a predecir y controlar la respuesta de su entorno cuando se desconocen las reglas que lo modelan. Generalmente, este tipo de problemas se resuelven mediante lo que se conoce como aprendizaje por refuerzo (RL): el agente aprende a través de la interacción con el entorno, descubriendo qué acciones dan lugar a la respuesta más favorable y reforzándolas. Un enfoque clásico para abordar el problema de aprendizaje por refuerzo es a través de los procesos de decisión de Markov (MDP), pues permiten modelar de manera sencilla la interacción entre el agente y su entorno.

Tras estas consideraciones, podremos afirmar que cualquier algoritmo que sea capaz de aprender a partir la interacción con el entorno con el fin de alcanzar un objetivo determinado, podrá ser considerado un método de aprendizaje por refuerzo.

Todas estas nociones dan aún una idea un tanto difusa sobre lo que es RL. A continuación, se presentan algunos conceptos que caracterizan el problema de aprendizaje por refuerzo de manera más detallada:

- **Objetivo:** típicamente se define como un estado del entorno al que desea llegar el agente, de manera que se tomarán una serie de acciones que provocarán transiciones de un estado a otro hasta que el entorno alcance el estado objetivo deseado. Otra interpretación puede darse desde el punto de vista de la señal de recompensa: el objetivo del agente será maximizar la recompensa total acumulada a largo plazo. Se puede deducir que ambas definiciones deberán ser consistentes, es decir, el agente deberá tomar aquellas acciones que lleven al entorno al estado deseado, de manera que se obtenga la mayor recompensa acumulada posible.
- **Secuencial:** el aprendizaje por refuerzo es un problema de decisión secuencial.

Las acciones del agente determinarán el siguiente estado al que se transite y la recompensa obtenida. Por tanto, el agente aprende a partir de datos correlados.

- **Señal de recompensa:** se trata de la realimentación que el agente recibe del entorno. Permite hacerse una idea respecto a si el agente se está acercando al objetivo.

- **Realimentación atrasada:** las acciones que se toman influyen tanto en la recompensa actual como en la futura. A veces, la recompensa obtenida en un instante de tiempo t no es muy significativa en dicho instante, pero si podría serlo su influencia en recompensas futuras. De este modo, habrá ocasiones en las que el agente tenga que sacrificar la recompensa instantánea obtenida en aras de obtener una recompensa mayor a largo plazo. Siguiendo con el ejemplo de escapar del laberinto que se mencionó en la Introducción, a veces el agente deberá escoger desvíos del camino más corto con intención de saber que no tiene que ir por ellos. De esta manera, se evita que en un futuro se decida tomar ese camino peor, y por tanto se garantiza la máxima recompensa a largo plazo.

- **Exploración vs explotación:** el dilema entre exploración y explotación es una de las cuestiones fundamentales de RL. Cuando el agente se enfrenta a una situación determinada, puede optar por tomar la mejor decisión en base a la información de que dispone (esto es, explotación). Puesto que se considera que las transiciones de estados y la señal de recompensa son estocásticas, una decisión alternativa podría dar lugar a un resultado mejor, peor o igual, pero en cualquier caso, el agente podría recabar más información y aumentar su conocimiento sobre la respuesta del entorno (esto es, exploración), En otras palabras, el agente puede decidir tomar las acciones que ya se comprobó en el pasado que eran efectivas (explotación) o tratar de descubrir acciones que podrían dar lugar a resultados mejores (exploración). En la práctica, los algoritmos de RL deben encontrar un grado de compromiso entre el nivel de exploración y de explotación.

En base a todos estos conceptos, se puede detallar de manera más precisa la labor que llevará a cabo un algoritmo de aprendizaje por refuerzo: dado un entorno desconocido, el algoritmo de aprendizaje por refuerzo tratará de deducir una serie de reglas de actuación –política de comportamiento– que permitan encontrar la secuencia de acciones que dirijan al agente al estado objetivo, maximizando la recompensa total obtenida a largo plazo. Para lograrlo, deberá existir un cierto balance entre la exploración y la explotación realizadas de manera que se pueda (a) explorar todo el entorno y aprender su comportamiento en base a la interacción con él, a la par que (b) explotemos el aprendizaje realizado y podamos alcanzar el objetivo deseado. Se dice que el algoritmo ha convergido cuando

se ha encontrado la política de comportamiento óptima.

Uno de los problemas a los que se enfrenta el campo del aprendizaje por refuerzo en la actualidad es el de encontrar la política óptima en un tiempo razonable; es decir, se busca un aprendizaje rápido y efectivo. Si bien es cierto que los algoritmos que existen hasta la fecha presentan un buen desempeño, se piensa que aún puede mejorarse más la velocidad de convergencia, bien mediante modificaciones de los algoritmos ya existentes o a través de nuevos enfoques poco estudiados hasta el momento.

3.2. Conceptos fundamentales en aprendizaje secuencial

3.2.1. Agente y entorno

De aquí en adelante, se van a considerar problemas de aprendizaje secuencial donde, en cada instante de tiempo t, el agente toma una acción A_t, percibe la observación O_t y obtiene una recompensa escalar R_t. Desde un punto de vista complementario, el del entorno, la acción será un parámetro de entrada mientras que la observación y la recompensa serán parámetros de salida.

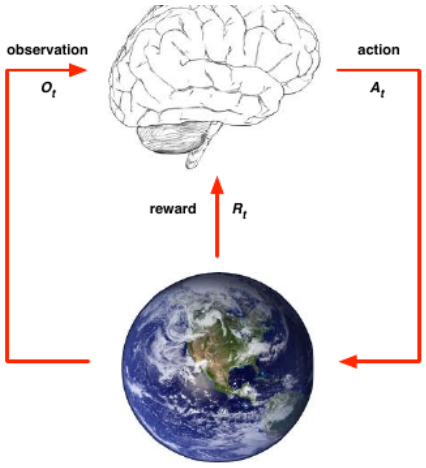


Figura 3.1: Agente y entorno

La *historia* en el instante t, a la que nos referiremos por H_t, se constituye por la secuencia de observaciones, acciones y recompensas obtenidas hasta ese instante t, es decir:

$$H_t = \{O_1 = o_1, R_1 = r_1, A_1 = a_1, \cdots, A_{t-1} = a_{t-1}, O_t = o_t, R_t = r_t\}$$

Obsérvese que se usarán letras mayúsculas para referirse a las variables aleatorias y letras minúsculas para referirse a realizaciones concretas de esas variables.

Emplearemos el término *estado* del entorno en el instante t, al cual nos referiremos

por S_t, como la parte de la historia H_t que determina (en un sentido estocástico) la observación y la recompensa en el instante $t+1$, es decir, O_{t+1} y R_{t+1}. De manera más formal, el estado será una función de la historia: $S_t = f(H_t)$.

De aquí en adelante, se va a suponer que los entornos considerados son totalmente observables, de manera que $O_t = S_t$. De hecho, cuando se dice que el entorno puede modelarse como un MDP, implícitamente se está asumiendo que el agente puede observar totalmente el estado (es decir, un MDP asume que $O_t = S_t$). Una propiedad de los MDP que será de gran importancia es que satisfacen la *propiedad de Markov* para las transiciones de estados, lo cual significa que la transición del estado actual al siguiente sólo dependerá del estado y acción actual, en lugar de depender de toda la historia. Es decir:

$$\mathbb{P}(S_{t+1} \mid H_t) = \mathbb{P}(S_{t+1} \mid S_t, A_t) \tag{3.1}$$

Para los subsecuentes capítulos, resultará conveniente clasificar los problemas en base al tamaño de sus conjuntos de estados y acciones. Cuando estos conjuntos sean pequeños y discretos, se hablará de *problemas de pequeña escala*, y dichos problemas se caracterizarán por el hecho de que la mayoría de las funciones involucradas pueden ser representadas a través de tablas. Sin embargo, esta representación tabular resultará computacionalmente inabordable cuando los conjuntos de estados y acciones sean muy grandes o continuos. En este caso, se hablará de *problemas de gran escala* y la representación eficiente de estos espacios grandes y continuos se convertirá en un problema. Para solucionarlo, se recurrirá a las aproximaciones, tal y como veremos en el capítulo 6.

3.2.2. Política, recompensa y función valor

Además del agente y el entorno, se pueden identificar otros tres elementos más en cualquier problema de aprendizaje por refuerzo: las políticas, la señal de recompensa y las funciones valor.

Política

Una *política* π dictamina cómo se comporta el agente, y se define como una función de los estados. Las políticas pueden ser deterministas o estocásticas.

1. Si la política es determinista, se define formalmente como una aplicación de los estados a las acciones, tal que la acción tomada en cualquier estado vendrá dada de manera determinista como la salida de la política. Por ejemplo, si nos encontramos en el estado $S_t = s$, la acción tomada por el agente en el instante t estará definida por $a = \pi(s)$. El espacio de políticas determinista se denota por Π^{MD}, siendo MD las siglas de *Markovian Deterministic*. El término «Markoviano» hace referencia a que el problema cumple la propiedad de Markov anteriormente citada.

2. Si la política es estocástica, entonces la política es una distribución de probabilidad condicionada, tal que la probabilidad de tomar la acción a supuesto que estamos en el estado $S_t = s$ viene dada por $\pi(a \mid s) = \mathbb{P}(A_t = a \mid S_t = s)$. El espacio de políticas estocásticas se denota por Π^{MR}, siendo MR las siglas de *Markovian Randomized*.

El espacio de políticas general, incluyendo tanto aquellas deterministas como las estocásticas, será referido como Π.

Por último, se dice que una política es estacionaria cuando es independiente del instante temporal en el que estemos o del tiempo transcurrido.

Recompensa

Como ya se ha comentado anteriormente, la señal de *recompensa*, R_t, se encarga de definir el objetivo en un problema de RL. En cada instante de tiempo, el entorno manda al agente una señal escalar. De este modo, el único objetivo del agente será maximizar la recompensa obtenida a largo plazo.

Función valor

Esta idea de la recompensa a largo plazo se define de manera formal como *función valor*. La función valor de estados es una función de los estados del entorno y la política del agente, y comúnmente se refiere a ella como $v^{\pi}(s)$. Representa la recompensa total que el agente podría esperar acumular en el futuro, cuando el entorno está en el estado s y el agente se comporta de acuerdo a la política π. Por tanto, dado que el agente tiene por objetivo encontrar la política que maximiza la recompensa a largo plazo –es decir, la función valor–, el proceso de aprendizaje debería estar guiado por la función valor más que por las recompensas instantáneas R_t. Sin embargo, mientras que las recompensas son recibidas directamente del entorno, las funciones valor deben ser estimadas a partir de H_t (es decir, la historia de estados, acciones y recompensas), y esta estimación debe ser actualizada durante todo el ciclo de vida del problema. Por ello, muchos de los algoritmos más eficientes de RL incluyen algún tipo de método para estimar la función valor de manera eficiente. El interés por estimar la función valor, tal y como se verá en el capítulo 4, será debido a que una vez la función valor es conocida, podremos mejorar la política disponible. De hecho, cuando se resuelven problemas de aprendizaje por refuerzo, habitualmente se diferencian dos fases del problema:

- Predicción: se refiere al problema de estimar la función valor $v^{\pi}(s)$ para una política π y todo estado posible s.

- Control: se refiere a la tarea de aprender la política óptima que maximiza la función valor.

Para referirnos a la política óptima y a la función valor óptima emplearemos la notación π^* y v^* respectivamente.

Como se verá en los siguientes capítulos, las tareas de predicción y control suelen intercalarse entre sí, y cada algoritmo emplea un método diferente para resolver cada una de ellas.

3.3. Taxonomía de los métodos empleados en RL

El reto principal al que se enfrentan tanto la programación dinámica como el aprendizaje por refuerzo es por tanto encontrar la política óptima que maximiza el rendimiento a largo plazo, medido a través de la función valor. La diferencia fundamental entre programación dinámica y aprendizaje por refuerzo es que la primera de ellas asume que el modelo es conocido –entendiendo por modelo la probabilidad de transición entre estados, así como la distribución de recompensas para cualquier posible transición–, mientras que la última considera que el modelo se desconoce y aprende a partir de la interacción con el entorno (*basado en muestras* o *basado en trayectoria*).

A continuación, se presenta una clasificación de los algoritmos de DP y RL en función del enfoque algorítmico que sigan. Esta clasificación será seguida en el resto de capítulos:

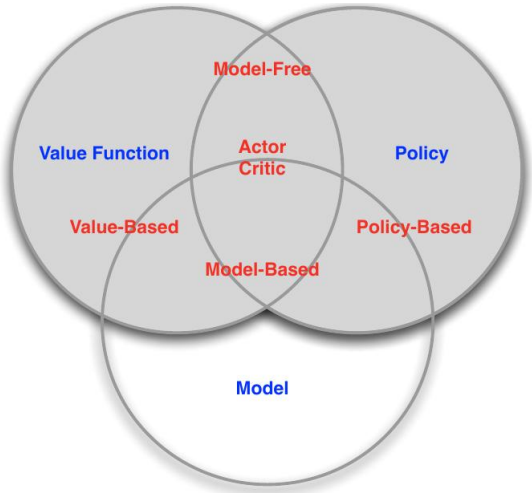


Figura 3.2: Clasificación de los métodos de RL.

1. *Value iteration*: El agente busca la función valor óptima. A continuación, esta función valor es empleada para calcular la política óptima.

2. *Policy iteration*: El agente escoge una política y calcula su función valor. Después,

emplea esa función valor para mejorar la política. El proceso se repite de manera iterativa hasta que se converge a la política óptima.

3. *Policy search*: El agente emplea técnicas de optimización para buscar la política óptima directamente sobre el espacio de políticas.

Este tipo de clasificación no es excluyente (por ejemplo, *value iteration* puede ser considerado un caso extremo de *policy iteration*). Además, muchos algoritmos combinan las ideas de más de una de estas clases.

Desde una perspectiva de implementación, se puede diferenciar también entre algoritmos *online* y *offline*:

1. Algoritmos *offline*: El agente aprende a partir de datos recopilados de antemano, antes de que el algoritmo se ejecute. La mayor adversidad a la que se enfrentan estos algoritmos, es que los datos disponibles no sean muy representativos de la política óptima, dando lugar a soluciones de mucha varianza.

2. Algoritmos *online*: el agente aprende al mismo tiempo que captura los datos.

Una vez se tiene claro el contexto del problema de aprendizaje por refuerzo, se puede situar mejor el objetivo de este trabajo. Lo que se pretende es formular un nuevo algoritmo *online* de aprendizaje por refuerzo que permita encontrar la política óptima de alguna manera no muy explorada hasta el momento, de forma que dé pie a nuevas líneas de estudio e investigación. Por ello, se decidió tratar de desarrollar un método de tipo *policy search*, pues actualmente apenas se han explotado. En el capítulo 9 se presentará la solución planteada.

Los cuatro capítulos siguientes asentarán la base teórica que será empleada en el capítulo 9 para desarrollar el algoritmo propuesto.

Capítulo 4

Procesos de decisión de Markov

4.1. Proceso de Markov

Un proceso de Markov (Markov Process, MP) es el modelo más sencillo que existe para representar una señal que no sea blanca, pues la probabilidad de cualquier muestra depende únicamente del valor de la muestra anterior. Se define como una tupla $\langle \mathcal{S}, \mathcal{P} \rangle$ donde $\mathcal{S}$ es el conjunto de estados y $\mathcal{P}$ es el *kernel* de la probabilidad de transición de estados. Por simplicidad, se asumirá que $\mathcal{S}$ es finito (lo cual implica que es contable) y no vacío, que las transiciones de estados ocurren en instantes de tiempo discretos (la extensión a conjuntos de estados finitos y tiempo continuo es posible, pero se escapa del alcance de este trabajo) y que el *kernel* de la probabilidad de transición de estados es estacionario. El *kernel* de la probabilidad de transición de estados asigna a cada estado $s \in \mathcal{S}$ una medida de probabilidad sobre $\mathcal{S}$. Supóngase que S_t hace referencia al estado en el instante t, donde, de nuevo, se usarán letras mayúsculas para referirse a las variables aleatorias y letras minúsculas para referirse a las realizaciones de la variable aleatoria. Dado que se está asumiendo un conjunto de estados finito, el número de posibles transiciones desde cualquier estado $s \in \mathcal{S}$ a cualquier otro estado $s' \in \mathcal{S}$ será $|\mathcal{S}|^2$, donde $|\cdot|$ denota la cardinalidad del conjunto. Por tanto, se podrá expresar de manera conveniente el *kernel* de la probabilidad de transición en forma matricial. Más concretamente, se tendrá que $\mathcal{P}$ es una matriz de dimensión $|\mathcal{S}| \times |\mathcal{S}|$ cuyos elementos serán:

$$\mathcal{P}_{ss'} \triangleq \mathbb{P}\left(S_{t+1} = s' | S_t = s\right), \quad s, s' \in \mathcal{S} \tag{4.1}$$

donde $\mathcal{P}_{ss'}$ hace referencia al elemento de la fila s y columna s'. Destacar que $\mathcal{P}$ es una matriz estocástica ya que sus filas son vectores de probabilidad, es decir:

$$\mathcal{P}_{ss'} \geq 0, \quad \forall s, s' \in \mathcal{S} \tag{4.2}$$

$$\sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'} = 1 \tag{4.3}$$

Se dirá que el proceso es de Markov ya que la probabilidad de transición dependerá únicamente del estado actual y no de la historia de los estados anteriores al instante t actual. En otras palabras, la transición estará influenciada por el estado en el que nos encontramos, no por cómo llegamos a él:

$$\begin{aligned} &\mathbb{P}(S_{t+1} = s_{t+1} \,|S_t = s_t, S_{t-1} = s_{t-1}, S_{t-2} = s_{t-2}, \ldots, S_0 = s_0) \\ &\quad = \mathbb{P}\left(S_{t+1} = s_{t+1} | S_t = s_t\right) \end{aligned} \tag{4.4}$$

A continuación se enunciarán algunas definiciones y propiedades que presentan este tipo de procesos estocásticos, también conocidos como *cadenas de Markov*.

Definición 4.1. Sea $\mathcal{P}_{ss'}^{(n)} \triangleq \mathbb{P}(S_{t+n} = s' | S_t = s)$ la probabilidad de alcanzar el estado $s' \in \mathcal{S}$ en n pasos cuando estamos en el estado s.

1. Se dice que el estado s' es alcanzable desde el estado s, si existe algún $n > 1$ tal que $\mathcal{P}_{ss'}^{(n)} > 0$. Una cadena de Markov se dice que es irreducible si cualquier estado s' es alcanzable desde cualquier otro estado s, para todo $s' \in \mathcal{S}$.

2. Se dice que una cadena de Markov es homogénea en el tiempo si la matriz de transición $\mathcal{P}$ se mantiene invariante después de cada paso.

3. Se dice que el estado s es aperiódico si existe un n tal que para todo $n' > n$ se tiene que $\mathcal{P}_{ss}^{(n)} > 0$. Una cadena de Markov se dice que es aperiódica si todos sus estados son aperiódicos.

Definición 4.2. Sea $d(t) = (d_s(t))_{s \in \mathcal{S}}$ el vector de probabilidades de visita de estados en el instante t, con componentes $d_s(t) \triangleq \mathbb{P}(S_t = s)$

1. El vector de probabilidades de visita de estados evoluciona de acuerdo a la ecuación:

$$d(t)^T = d(t-1)^T \mathcal{P} \tag{4.5}$$

2. Este tipo de vector se conoce como *vector de probabilidad*.

$$d_s(t) \geq 0 \quad \text{para todo } s \in \mathcal{S} \tag{4.6}$$

$$\sum_{s \in \mathcal{S}} d_s(t) = 1 \tag{4.7}$$

El interés real, para nuestro caso, se encuentra en el comportamiento asintótico de la cadena de Markov. Más adelante se verá que para que podamos aprender a partir de la interacción, el entorno deberá comportarse de manera consistente a lo largo del tiempo. Esto significa que las probabilidades de transición de estado deberán mantenerse constantes (este hecho ha sido implícitamente asumido cuando se ha escrito $\mathcal{P}$ de manera independiente a t, y es conocido como homogeneidad en el tiempo, tal y como se describió en la definición 4.1). Bajo esta asunción, deberá existir una distribución de visita de estados estacionaria para la cadena de Markov (es decir, la distribución de visita de estados tenderá a una distribución límite), la cual deberá ser también independiente de la distribución inicial. El aprendizaje en entornos no estacionarios se escapa del alcance de este trabajo.

Proposición 4.1. *Para una cadena de Markov homogénea en el tiempo, se tiene:*

$$\begin{aligned} d(t)^T &= d(t-1)^T\mathcal{P} \\ &= d(t-2)^T\mathcal{P}\mathcal{P} = d(t-2)^T\mathcal{P}^2 \\ &\vdots \\ &= d(0)^T\mathcal{P}^t \end{aligned} \tag{4.8}$$

A continuación, se introduce de manera formal el concepto de distribución estacionaria.

Definición 4.3. Se dice que una distribución d es una distribución estacionaria para la cadena de Markov si:

$$d^T = d^T\mathcal{P} \tag{4.9}$$

Esta distribución puede ser interpretada como la distribución de visita de estados, es decir, como la proporción media de tiempo que la cadena pasa en cada estado.

Definición 4.4. Se dice que una distribución d es la distribución límite de la cadena de Markov si para toda distribución inicial se cumple:

$$d = \lim_{t\to\infty} d(t) \tag{4.10}$$

Aunque esta distribución límite no tiene por qué existir necesariamente, si existe, entonces es única. Además, si d es una distribución límite, de acuerdo a la definición 4.3 se deduce que es también la distribución estacionaria de la cadena de Markov. Recalcar que $\lim_{t\to\infty} d(t) = \lim_{t\to\infty} d(t-1)$. En consecuencia se tiene que:

$$d = \lim_{t\to\infty} d(t) = \lim_{t\to\infty} d(t-1)\mathcal{P} = d\mathcal{P} \tag{4.11}$$

El siguiente teorema detalla las condiciones para la existencia de d.

Teorema 4.1. *Una cadena de Markov aperiódica, irreducible y homogénea en el tiempo, tiene una distribución límite de estados sí y solo sí la matriz de transición tiene un único autovalor real igual a 1, y ningún autovalor complejo de magnitud igual a 1.*

A partir de (4.11) y del teorema 4.1, se deduce que el vector d de distribución límite será el autovector normalizado asociado al autovalor 1 de $\mathcal{P}^T$.

4.2. Proceso de decisión de Markov

4.2.1. Definición

Los procesos de decisión de Markov (*Markov Decision Process*, MDP) pueden entenderse como una extensión de los MP en la cual la probabilidad de transición de estados depende de una variable aleatoria adicional, la acción, y donde cada transición tiene asociada otra variable aleatoria, la recompensa. De manera más formal, un MDP se define como la tupla $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$ donde $\mathcal{S}$ es el conjunto de estados, $\mathcal{A}$ es el conjunto de acciones, $\mathcal{P}$ es el *kernel* de probabilidad de transición y $\mathcal{R}$ es la función de recompensa. Una vez más, por simplicidad, se asumirá que tanto $\mathcal{S}$ como $\mathcal{A}$ son finitos y no vacíos.

A diferencia de los MP, el *kernel* de probabilidad de transición de estados ahora depende de la acción tomada por el agente, y asigna a cada par estado-acción $(s, a) \in \mathcal{S} \times \mathcal{A}$ una medida de probabilidad sobre $\mathcal{S}$ que será la probabilidad de ir a otro estado $s' \in \mathcal{S}$. Supongamos que S_t y A_t hacen referencia al estado y la acción en el instante t, respectivamente. Dado que estamos asumiendo conjuntos finitos de estados y acciones, el número total de posibles transiciones desde cualquier par estado-acción $(s, a) \in \mathcal{S} \times \mathcal{A}$ a cualquier otro estado $s' \in \mathcal{S}$ será $|\mathcal{S}|^2|\mathcal{A}|$. Por tanto, de manera similar a los MP, se podrá expresar el *kernel* de probabilidad en forma matricial. En concreto, se puede expresar $\mathcal{P}$ para un MDP como una matriz de dimensión $|\mathcal{S}||\mathcal{A}| \times |\mathcal{S}|$ o como un tensor $|\mathcal{S}| \times |\mathcal{S}| \times |\mathcal{A}|$. En cualquier caso, los elementos de cada representación vienen dados por:

$$\mathcal{P}^a_{ss'} \triangleq \mathbb{P}\left(S_{t+1} = s' | S_t = s, A_t = a\right) \tag{4.12}$$

Para la representación matricial de $\mathcal{P}$, cada una de las filas se refiere a los pares estado acción (s, a) y cada columna hace referencia al estado futuro s':

$$\mathcal{P} \triangleq (\mathcal{P}^a_{ss'})_{(s,a)\in\mathcal{S}\times\mathcal{A}, s'\in\mathcal{S}}$$

Llámese R_{t+1} a la recompensa aleatoria obtenida en el instante $t + 1$. Entonces se puede definir la función de recompensa $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ como el valor esperado de la

recompensa aleatoria para cualquier transición desde cualquier par estado-acción:

$$\mathcal{R}_s^a \triangleq \mathcal{R}(s,a) \triangleq \mathbb{E}\left[R_{t+1}|S_t = s, A_t = a\right] \tag{4.13}$$

Abusando ligeramente de la notación, será conveniente definir el vector de recompensas como un vector de longitud $|\mathcal{S}||\mathcal{A}|$ que contiene la función recompensa para cada posible par estado-acción:

$$\mathcal{R} \triangleq \left(\mathcal{R}_s^a\right)_{s\in\mathcal{S},a\in\mathcal{A}}$$

Como ya se introdujo en el capítulo anterior, el agente toma las acciones siguiendo una política. De manera formal, se define una política $\pi \in \Pi$, donde Π es algún conjunto de distribuciones de probabilidad sobre $\mathcal{A}$, como la distribución sobre las acciones dados los estados:

$$\pi\left(a|s\right) \triangleq \mathbb{P}\left(A_t = a|S_t = s\right) \tag{4.14}$$

Cabe destacar que una política define completamente el comportamiento de un agente. Se asumirá que las políticas son estacionarias (es decir, $A_t \sim \pi(\cdot|S_t)$, $\forall t > 0$). De este modo, si el comportamiento del agente cambia a lo largo del tiempo (porque aprende de qué manera controlar el entorno, por ejemplo), consideraremos que es el agente quien cambia su política, en lugar de decir que es la política la que cambia.

Definición 4.5. Una política determinista es un caso particular de la aplicación $\pi : \mathcal{S} \to \mathcal{A}$, tal que al escribir $\pi\left(s\right) = a$ nos referiremos a que $\pi(a|s) = 1$ (es decir, $\mathbb{P}\left(A_t = a|S_t = s\right) = 1$).

Tras estos conceptos, podremos definir $\mathcal{P}_{ss'}^{\pi}$ y $\mathcal{R}_s^{\pi}$ como la probabilidad de transición promedio y la función recompensa promedio, respectivamente, cuando las acciones se eligen de acuerdo a la política π:

$$\mathcal{P}_{ss'}^{\pi} = \sum_{a\in\mathcal{A}} \pi\left(a|s\right)\mathcal{P}_{ss'}^{a} \tag{4.15}$$

$$\mathcal{R}_s^{\pi} = \sum_{a\in\mathcal{A}} \pi\left(a|s\right)\mathcal{R}_s^{a} \tag{4.16}$$

Resultará conveniente agrupar todos los elementos de (4.15) para todos los pares de transición de estado $(s, s') \in \mathcal{S}\times\mathcal{S}$ en una matriz $\mathcal{P}^{\pi}$ de dimensión $|\mathcal{S}| \times |\mathcal{S}|$, y juntar las recompensas (4.16) para todo $s \in \mathcal{S}$ en un vector de recompensas $\mathcal{R}^{\pi}$ de longitud $|\mathcal{S}|$:

$$\mathcal{P}^{\pi} \triangleq \left(\mathcal{P}_{ss'}^{\pi}\right)_{s,s'\in\mathcal{S}} \tag{4.17}$$

$$\mathcal{R}^{\pi} \triangleq \left(\mathcal{R}_s^{\pi}\right)_{s\in\mathcal{S}} \tag{4.18}$$

En el resto de este documento, siempre que se hable de aprender a predecir o controlar

el MDP a partir de muestras, sin conocer $\mathcal{P}$ o $\mathcal{R}$, se asumirá la siguiente condición:

Suposición 1. *Para cualquier MDP y cualquier política estacionaria, la cadena de Markov inducida asociada a la transición estado-acción-estado-recompensa satisfará el teorema 4.1, de manera que tendrá una distribución límite que será la distribución de visita de estados única estacionaria.*

De la suposición 1 se puede intuir que, teniendo una distribución de visita de estados estacionaria y siendo capaces de visitar todos los estados, el agente podrá obtener estimaciones consistentes a partir de las muestras.

Sin embargo, existen muchos problemas que se modelan de manera natural con un estado *terminal* donde terminan. La probabilidad de quedarse en dicho estado es uno (por este motivo los estados terminales también se conocen como estados *absorbentes*). Estos MDP con estados terminales, típicamente se conocen como episódicos, y la trayectoria desde el estado inicial al terminal recibe el nombre de episodio. Parece claro que la suposición 1 no se satisface para este tipo de MDP episódicos. Dado que, con el objetivo de poder aprender a partir de las muestras con los métodos que se propongan en el siguiente capítulo, necesitaremos que el MDP satisfaga la suposición 1, será necesario hacer un pequeño cambio en la formulación del problema: cada vez que la trayectoria alcance el estado terminal, reiniciaremos el episodio (se reiniciará a cero la recompensa acumulada, y la trayectoria al estado inicial). Se obtendrán así episodios infinitos, formados por episodios de longitud finita. De este modo, la suposición 1 se deberá cumplir para todo MDP en el cual la transición al estado terminal haya sido transformada en el reinicio al estado inicial.

4.2.2. Funciones valor y ecuaciones de Bellman

El objetivo en un MDP es encontrar la política que maximiza el retorno esperado a largo plazo. De manera formal, se define el *retorno* obtenido desde el instante t como la recompensa acumulada descontada:

$$G_t \triangleq R_{t+1} + \gamma R_{t+2} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{4.19}$$

donde $0 < \gamma < 1$ es el llamado factor de *descuento*.

Como puede apreciarse en (4.19), el valor de la recompensa pasados $k+1$ instantes sufrirá un factor de escala de reducción de γ^k. Esto es así para evitar la divergencia de (4.19) (dado que la suma tiene infinitos términos), pero también tiene otro significado, y es el de determinar el valor actual de las recompensas futuras: una recompensa obtenida $k+1$ instantes de tiempo más tarde valdrá γ^k veces lo que valdría si la hubiésemos

recibido inmediatamente. Cuando $\gamma \approx 0$, se dice que el retorno es «miope», mientras que si $\gamma \approx 1$, se dirá que es «hipermétrope».

Cabe destacar que el retorno G_t es una variable aleatoria que depende de la trayectoria específica seguida por el agente. Se define a continuación la *función valor de estados* $v^\pi : \mathcal{S} \to \mathbb{R}$ como el retorno esperado sobre todas las posibles trayectorias cuando el agente comienza en un estado y a continuación sigue la política π:

$$v^\pi(s) \triangleq \mathbb{E}_{\pi,\mathcal{P}}\left[G_t | S_0 = s, A_{t+k} \sim \pi\right], \quad k = 0, \ldots, \infty \tag{4.20}$$

donde $\mathbb{E}_{\pi,\mathcal{P}}[\cdot]$ denota el valor esperado sobre la distribución de la política y sobre la distribución de transiciones de estado. También se va a definir la función valor de estados-acciones $q^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ como el retorno esperado sobre todas las posibles trayectorias cuando el agente comienza en un estado, toma una acción cualquiera y a continuación sigue la política π:

$$q^\pi(s,a) \triangleq \mathbb{E}_{\pi,\mathcal{P}}\left[G_t | S_t = s, A_t = a, A_{t+k} \sim \pi\right], \quad k = 1, \ldots, \infty \tag{4.21}$$

Las funciones valor satisfacen una relación recursiva conocida como ecuación de Bellman. Para la función valor de estados, tenemos para $k = 0, \ldots, \infty$:

$$\begin{aligned} v^\pi(s) &= \mathbb{E}_{\pi,\mathcal{P}}\left[G_t | S_t = s, A_t \sim \pi\right] \\ &= \mathbb{E}_{\pi,\mathcal{P}}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots | S_t = s, A_{t+k} \sim \pi\right] \\ &= \mathbb{E}_{\pi,\mathcal{P}}\left[R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \ldots) | S_t = s, A_{t+k} \sim \pi\right] \\ &= \mathbb{E}_{\pi,\mathcal{P}}\left[R_{t+1} + \gamma G_{t+1} | S_t = s, A_{t+k} \sim \pi\right] \\ &= \mathbb{E}_{\pi,\mathcal{P}}\left[R_{t+1} + \gamma v^\pi(S_{t+1}) | S_t = s, A_{t+k} \sim \pi\right] \end{aligned} \tag{4.22}$$

De manera similar, para la función valor de estados-acciones se define, para $k = 1, \ldots, \infty$:

$$\begin{aligned} q^\pi(s,a) &= \mathbb{E}_{\pi,\mathcal{P}}\left[G_t | S_t = s, A_t = a, A_{t+k} \sim \pi\right] \\ &= \mathbb{E}_{\pi,\mathcal{P}}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots | S_t = s, A_t = a, A_{t+k} \sim \pi\right] \\ &= \mathbb{E}_{\pi,\mathcal{P}}\left[R_{t+1} + \gamma G_{t+1} | S_t = s, A_t = a, A_{t+k} \sim \pi\right] \\ &= \mathbb{E}_{\pi,\mathcal{P}}\left[R_{t+1} + \gamma q^\pi(S_{t+1}, A_{t+1}) | S_t = s, A_t = a, A_{t+k} \sim \pi\right] \end{aligned} \tag{4.23}$$

En otras palabras, las funciones valor de estados y de estados-acciones se pueden descomponer en la recompensa instantánea más el valor descontado del estado o del par estado-acción sucesor, respectivamente.

La relación existente entre las funciones valor de estados y de estados-acciones es la

siguiente:

$$v^{\pi}(s) = \sum_{a \in \mathcal{A}} \pi(a|s)\, q^{\pi}(s,a) \tag{4.24}$$

$$q^{\pi}(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v^{\pi}(s') \tag{4.25}$$

donde (4.24) se obtiene a partir de (4.21) al considerar que $A_t \sim \pi$ (es decir, que la acción actual se escoge de acuerdo a la política π); y (4.25) se obtiene a partir de (4.23) teniendo en cuenta (4.13) y la siguiente relación para $k = 1, \ldots, \infty$:

$$\mathbb{E}_{\pi,\mathcal{P}}\left[G_{t+1}|S_t = s, A_t = a, A_{t+k} \sim \pi\right] = \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v^{\pi}(s') \tag{4.26}$$

De manera visual, estas relaciones se pueden explicar a través del diagrama mostrado en la figura 4.1.

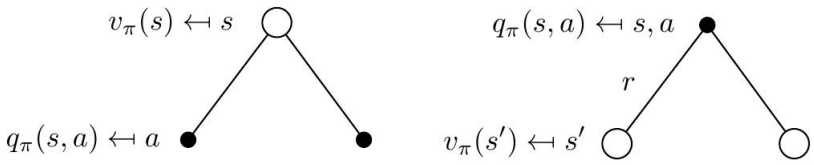


Figura 4.1: Relación existente entre $v_\pi(s)$ y $q_\pi(s,a)$.

Además, a partir de (4.24) y (4.25), se llega a:

$$v^{\pi}(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v^{\pi}(s') \right) \tag{4.27}$$

$$q^{\pi}(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \sum_{a' \in \mathcal{A}} \pi(a'|s')\, q^{\pi}(s',a') \tag{4.28}$$

De nuevo, se puede obtener una representación visual, mostrada en la figura 4.2.

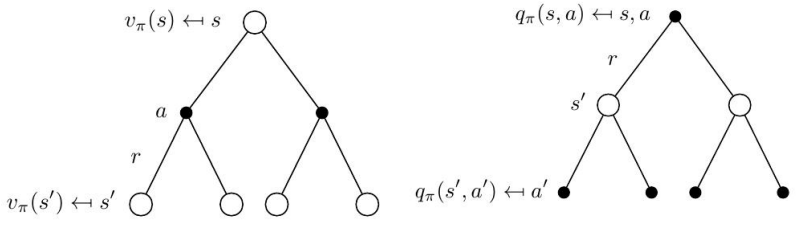


Figura 4.2: Ecuaciones recursivas para $v_\pi(s)$ y $q_\pi(s,a)$.

Resulta interesante notar que las funciones valor para cada estado (o estado acción)

forman un sistema lineal de ecuaciones. Por tanto, conocidos $\mathcal{R}$ y $\mathcal{P}$ se podrá obtener v^π y q^π para cada estado y cada par estado-acción, respectivamente, resolviendo un sistema lineal de ecuaciones. Esto se puede ver de manera más clara reescribiendo las ecuaciones en forma vectorial. Abusando ligeramente de la notación, se definirán los vectores formados por la función valor de estados y estados-acciones para cada estado y cada par estado-acción, respectivamente, de la siguiente manera:

$$v^\pi \triangleq (v^\pi(s))_{s\in\mathcal{S}} \tag{4.29}$$

$$q^\pi \triangleq (q^\pi(s,a))_{s\in\mathcal{S},a\in\mathcal{A}} \tag{4.30}$$

De manera adicional, se expresará la política $\pi \in \Pi$ como la matriz $\mathit{\Pi}$ de dimensión $|\mathcal{S}| \times |\mathcal{S}||\mathcal{A}|$ (como puede apreciarse, se usará la letra griega Π mayúscula para referirse al espacio de políticas, y $\mathit{\Pi}$ mayúscula y en cursiva para referirse a la política en forma matricial).

Por tanto, teniendo en cuenta (4.17) y (4.18) se podrá expresar (4.22)–(4.23), para todos los estados y todos los pares estado-acción, en forma vectorial de la siguiente manera:

$$v^\pi = \mathcal{R}^\pi + \gamma\mathcal{P}^\pi v^\pi \tag{4.31}$$

$$q^\pi = \mathcal{R} + \gamma\mathcal{P}\mathit{\Pi} q^\pi \tag{4.32}$$

haciendo especial hincapié en que (4.31) tiene en cuenta el vector de recompensas promedio $\mathcal{R}^\pi$ y la matriz de transición $\mathcal{P}^\pi$ inducida por π, mientras que (4.32) considera los términos $\mathcal{R}$ y $\mathcal{P}$ de manera independiente a la política.

Las ecuaciones (4.31)–(4.32) permiten ver de manera clara que las ecuaciones de Bellman forman un sistema lineal de ecuaciones. Además, a partir de esta formulación compacta resulta directo resolver las funciones valor en forma cerrada:

$$v^\pi = (I - \gamma\mathcal{P}^\pi)^{-1}\mathcal{R}^\pi \tag{4.33}$$

$$q^\pi = (I - \gamma\mathcal{P}\mathit{\Pi})^{-1}\mathcal{R} \tag{4.34}$$

La siguiente proposición manifiesta la existencia de una solución única a estos sistemas de ecuaciones.

Proposición 4.2. *Bajo el cumplimiento de la suposición 1, existen soluciones únicas v^π y q^π para* (4.33) *y* (4.34)*, respectivamente, y son funciones valor que satisfacen* (4.27) *y* (4.28)*.*

De acuerdo a la notación anterior, finalmente podrá expresarse (4.24)–(4.25) en forma

vectorial de la siguiente manera:

$$v^{\pi} = \Pi q^{\pi} \tag{4.35}$$

$$q^{\pi} = \mathcal{R} + \gamma \mathcal{P} v^{\pi} \tag{4.36}$$

Del mismo modo, a partir de la matriz de la política se podrán reformular las ecuaciones (4.17)–(4.18) de la siguiente manera:

$$\mathcal{P}^{\pi} = \Pi \mathcal{P}$$

$$\mathcal{R}^{\pi} = \Pi \mathcal{R}$$

Hasta el momento, únicamente se ha considerado el problema de *evaluación de la política*, es decir, se ha presentado la manera de obtener las funciones valor para una política π dada. Ahora, se va a pasar a considerar el problema de *control* en el cual el agente tratará de buscar la política con la que lograr el mejor desempeño posible. Puesto que el desempeño de una política es evaluado a partir de su función valor, se dirá que el agente se comporta de manera óptima –de ahí el nombre de política óptima– cuando consigue la función valor máxima –la función valor óptima–. La política óptima y la función valor óptima se definen como se muestra a continuación.

Definición 4.6. La función valor de estados óptima v^* será la función valor de estados máxima sobre todas las políticas posibles:

$$v^*(s) \triangleq \max_{\pi \in \Pi} v^{\pi}(s), \quad \forall s \in \mathcal{S} \tag{4.37}$$

donde Π denota el conjunto de políticas. La función valor de estados-acciones óptima $q^*(s,a)$ será la función valor de estados-acciones máxima sobre todas las políticas posibles:

$$q^*(s,a) \triangleq \max_{\pi \in \Pi} q^{\pi}(s,a), \quad \forall (s,a) \in \mathcal{S} \times \mathcal{A} \tag{4.38}$$

De manera intuitiva, se puede definir la política óptima como aquella que lleva a conseguir la función valor óptima. Esta idea se enuncia más formalmente a través del siguiente teorema.

Teorema 4.2. *Definiendo un orden parcial sobre las políticas tal que:*

$$\pi \geq \pi' \quad si \quad v^{\pi}(s) \geq v^{\pi'}, \forall s \tag{4.39}$$

Entonces, para cualquier MDP:

1. Existe una política óptima π^* que es mejor o igual que el resto de políticas del espacio de políticas, es decir: $\exists \pi^* \in \Pi \ : \ \pi^* \geq \pi, \ \forall \pi \in \Pi$.

2. Toda política óptima logra la función valor de estados y de estados-acciones óptima, es decir: $v^{\pi^*}(s) = v^*(s)$ y $q^{\pi^*}(s,a) = q^*(s,a)$.

Resulta además que la existencia de una política óptima determinista está garantizada por el siguiente lema.

Lema 4.1. *Existe siempre una política óptima determinista para cualquier MDP que satisfaga la suposición 1 [5].*

La derivación y demostración de este lema se tratará más en detalle en el capítulo 9.

El enfoque basado en la búsqueda de la política óptima mediante el cálculo de la función valor para toda política posible, y las definiciones (4.37)–(4.38) es lo que típicamente se conoce como *policy search* . Puesto que el número de posibles políticas deterministas crece de manera exponencial con el tamaño de los conjuntos de estados y acciones, parece claro que este método de búsqueda directa puede resultar inviable cuando $\mathcal{S}$ y $\mathcal{A}$ son muy grandes. Para tratar de solucionar este problema de escalabilidad, en la práctica se emplean algunas aproximaciones que restringen el espacio de búsqueda a espacios de políticas más pequeños.

Otro enfoque consiste en encontrar primero la función valor óptima y después extraer la política óptima a partir de v^*. La teoría que subyace a esta idea se formula en el siguiente lema.

Lema 4.2. *La política óptima puede ser encontrada mediante la maximización de la acción a sobre la función valor $q^*(s,a)$ óptima, es decir:*

$$\pi^*(a \mid s) = \begin{cases} 1 & \textit{if} \quad a = \underset{a \in \mathcal{A}}{\arg\max} q^*(s,a) \\ 0 & \text{resto} \end{cases} \tag{4.40}$$

En los capítulos 5 y 6 se presentarán diferentes algoritmos para encontrar v^* y q^*, y a continuación derivar la política óptima a partir de ellos. Pero antes de pasar a explicarlos introduciremos el concepto de *greediness* o «avaricia».

Definición 4.7. Se dirá que una política es *greedy* para algún estado $s \in \mathcal{S}$ con respecto a alguna función valor $v : \mathcal{S} \to \mathbb{R}$ o $q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, si dicha política maximiza la función valor:

$$\begin{aligned} \pi(s) &= \arg\max_{a \in \mathcal{A}} q(s,a) \\ &= \arg\max_{a \in \mathcal{A}} \left[\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v(s') \right] \end{aligned} \tag{4.41}$$

Resulta sencillo observar que la política óptima será *greedy* con respecto a la función valor óptima. Este resultado se recoge de manera formal en el siguiente lema.

Lema 4.3. *La política óptima π^* es greedy para todo $s \in \mathcal{S}$ con respecto a la función valor óptima $v^* : \mathcal{S} \to \mathbb{R}$ o $q^* : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$:*

$$\begin{aligned} \pi^*(s) &= \arg\max_{a \in \mathcal{A}} q^*(s,a) \\ &= \arg\max_{a \in \mathcal{A}} \left[\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v^*(s') \right] \end{aligned} \tag{4.42}$$

Las ecuaciones de Bellman (4.24)–(4.28), definidas anteriormente para políticas generales, pueden ser particularizadas para políticas óptimas deterministas, dando como resultado las siguientes relaciones:

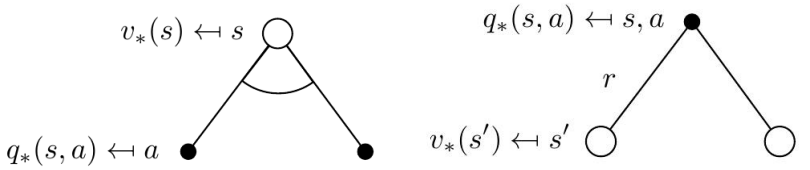
$$v^*(s) = \max_{a \in \mathcal{A}} q^*(s,a) \tag{4.43}$$

$$q^*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v^*(s') \tag{4.44}$$

$$v^*(s) = \max_{a \in \mathcal{A}} \left[\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v^*(s') \right] \tag{4.45}$$

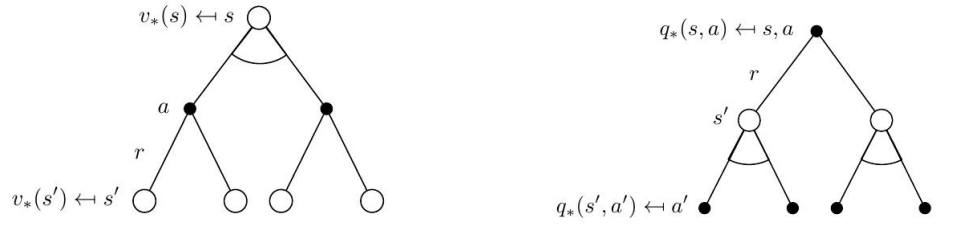
$$q^*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \max_{a' \in \mathcal{A}} q^*(s',a') \tag{4.46}$$

Las figuras 4.3 y 4.4 ilustran, a través de diagramas, estas dependencias.



$$v_*(s) = \max_a q_*(s,a) \qquad q_*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_*(s')$$

Figura 4.3: Relación existente entre $v^*(s)$ y $q^*(s,a)$.



$$v_*(s) = \max_a \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_*(s') \qquad q_*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \max_{a'} q_*(s',a')$$

Figura 4.4: Ecuaciones de punto fijo para $v^*(s)$ y $q^*(s,a)$.

Capítulo 5

Resolución de las ecuaciones de Bellman en problemas de pequeña escala

Tal y como se detalló en el capítulo 3, los problemas de aprendizaje por refuerzo se van a clasificar en dos grandes grupos en función del tamaño de sus conjuntos de estados y acciones. Atendiendo a este criterio, se pueden encontrar soluciones basadas en tablas, apropiadas para problemas de pequeña escala, y soluciones basadas en aproximaciones de la función valor, que permitirán abordar los problemas de gran dimensionalidad.

En este capítulo se van a tratar dos de las técnicas más usadas cuando los conjuntos de estados y acciones de nuestro problema son pequeños y discretos: programación dinámica (*Dynamic Programming*, DP) y diferencias temporales (*Temporal Difference*, TD).

Antes de comenzar la explicación, recalcar que ambos métodos permitirán resolver el problema de control que se plantee. No obstante, DP se emplea cuando el modelo es conocido y por tanto el agente puede planificar las decisiones de antemano. Por el contrario, TD es un método de aprendizaje por refuerzo puro en el sentido de que se da por desconocido el modelo del entorno, y por tanto el agente debe aprender la política de comportamiento óptima a partir de la interacción con el mismo. Dado que será este último caso el de mayor interés de cara al objetivo de este trabajo –desarrollo de nuevos algoritmos de aprendizaje por refuerzo–, se presentarán dos de los algoritmos más empleados en la actualidad que hacen uso de TD: SARSA y Q-learning. Precisamente por ser los más extendidos hasta la fecha, serán estos dos algoritmos con los que se compare el algoritmo novel que se desarrolle en el capítulo 10.

5.1. Programación dinámica

El término *programación dinámica* hace referencia a un conjunto de métodos iterativos que permiten resolver las ecuaciones de Bellman. La palabra «programación» es un sinónimo matemático estándar de optimización, y «dinámica» hace referencia al hecho de que estamos optimizando una función a lo largo de un horizonte temporal. DP ha sido aplicado a multitud de problemas prácticos que comparten una propiedad fundamental definida como el *principio de optimalidad*, que en palabras de Richard Bellman es explicado de la siguiente manera:

> Una política óptima tiene la propiedad de que, cualesquiera que sean el estado inicial y la decisión inicial, las decisiones restantes deberán constituir una política óptima con respecto al estado resultante de la primera decisión (ver Bellman, 1957, Cap. III.3).

En otras palabras, el principio de optimalidad establece que el problema puede ser descompuesto en subproblemas y que la solución óptima del problema puede ser también descompuesta, de manera que se puede extraer la solución óptima a partir de los subproblemas. Esto es exactamente lo que ocurre en, por ejemplo, la ecuación de Bellman (4.45), donde la función valor óptima para el estado s se descompone en la acción que proporciona la recompensa máxima esperada y la función valor óptima para el siguiente estado esperado. Ejemplos de problemas prácticos que satisfacen el principio de optimalidad son: planificación, problemas de camino más corto, problemas de control óptimo, etc.

5.1.1. Operador de Bellman

Se van a presentar ahora dos aplicaciones que jugarán un papel teórico importante en control óptimo. Para cualquier función $v : \mathcal{S} \to \mathbb{R}$ consideraremos el lado derecho de las ecuaciones de Bellman (4.22) y (4.27) como una aplicación, conocida como *operador de Bellman* para la política π, y para cualquier $s \in \mathcal{S}$ se definirá de la siguiente manera:

$$\begin{aligned}(T_\pi v)(s) &\triangleq \mathbb{E}_{\pi,\mathcal{P}}\left[R_{t+1} + \gamma v^\pi(S_{t+1}) | S_t = s\right] \\ &= \sum_{a\in\mathcal{A}} \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{ss'}^a v^\pi(s')\right)\end{aligned} \tag{5.1}$$

Se puede observar que (5.1) dependerá de la política π que usemos para promediar las acciones y las transiciones de estado. Para el caso concreto de la política óptima, se define el *operador de Bellman óptimo*, T, de nuevo para una función $v : \mathcal{S} \to \mathbb{R}$, como el

lado derecho de la ecuación de Bellman óptima (4.45), para cualquier $s \in \mathcal{S}$:

$$\begin{aligned}(Tv)(s) &\triangleq \max_{a\in\mathcal{A}} \mathbb{E}_{\mathcal{P}}\left[R_{t+1} + \gamma v(S_{t+1}) \,|S_t = s\right] \\ &= \max_{a\in\mathcal{A}} \left[\mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{ss'}^a v(s')\right] \end{aligned} \tag{5.2}$$

Abusando ligeramente de la notación, se puede definir también el mismo operador para cualquier función valor de estados-acciones $q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ a partir de (4.23)–(4.28) , (4.38) y (4.46), para cualquier par $(s, a) \in \mathcal{S} \times \mathcal{A}$:

$$\begin{aligned}(T_\pi q)(s,a) &\triangleq \mathbb{E}_{\pi,\mathcal{P}}\left[R_{t+1} + \gamma q^\pi(S_{t+1}, A_{t+1}) \,|S_t = s, A_t = a\right] \\ &= \mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{ss'}^a \sum_{a'\in\mathcal{A}} \pi(a'|s')\, q^\pi(s', a') \end{aligned} \tag{5.3}$$

$$\begin{aligned}(Tq)(s,a) &\triangleq \max_{a\in\mathcal{A}} \mathbb{E}_{\mathcal{P}}\left[R_{t+1} + \gamma q(S_{t+1}, A_{t+1}) \,|S_t = s, A_t = a\right] \\ &= \mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{ss'}^a \max_{a'\in\mathcal{A}} q(s', a') \end{aligned} \tag{5.4}$$

Saber si nos referimos al operador de Bellman de las funciones valor de estados o de estados-acciones debería quedar claro a partir del contexto.

El operador de Bellman en forma vectorial para una política π dada será definido como:

$$T_\pi v \triangleq \left((T_\pi v)(s)\right)_{s\in\mathcal{S}} = \mathcal{R}^\pi + \gamma \mathcal{P}^\pi v \tag{5.5}$$

$$T_\pi q \triangleq \left((T_\pi q)(s,a)\right)_{(s,a)\in\mathcal{S}\times\mathcal{A}} = \mathcal{R} + \gamma \mathcal{P} \Pi q \tag{5.6}$$

De manera similar, el operador de Bellman óptimo expresado en forma vectorial será:

$$Tv \triangleq \left((Tv)(s)\right)_{s\in\mathcal{S}} \tag{5.7}$$

$$Tq \triangleq \left((Tq)(s,a)\right)_{(s,a)\in\mathcal{S}\times\mathcal{A}} \tag{5.8}$$

De este modo se podrán reescribir las ecuaciones de Bellman, en una forma muy compacta, de la siguiente manera:

$$v^\pi = T_\pi v^\pi \tag{5.9}$$

$$q^\pi = T_\pi q^\pi \tag{5.10}$$

$$v^* = Tv^* \tag{5.11}$$

$$q^* = Tq^* \tag{5.12}$$

Todos estos operadores de Bellman satisfacen una propiedad muy importante conocida

como *aplicación contractiva*. Antes de demostrar que cumplen dicha propiedad, se definirá el concepto de aplicación contractiva.

Definición 5.1. Una aplicación $F : \mathcal{X} \to \mathcal{X}$ se dice que es una aplicación contractiva o una contracción si existe una constante c, con $0 \leq c < 1$, tal que:

$$\|F(v) - F(v')\| \leq c\|v - v'\| \tag{5.13}$$

para todo $v, v' \in \mathcal{X}$, donde $\mathcal{X}$ es algún espacio con una métrica $\|\cdot\|$.

Con el fin de probar que los operadores de Bellman son contracciones, deberemos definir antes la norma infinito.

Definición 5.2. La norma infinito (o norma máxima), denotada como ∞-norm, entre dos vectores valor, $v \in \mathbb{R}^{|\mathcal{S}|}$ y $v' \in \mathbb{R}^{|\mathcal{S}|}$, vendrá dada por la mayor diferencia entre cada una de sus componentes:

$$\|v - v'\|_\infty = \max_{s \in \mathcal{S}} |v(s) - v'(s)| \tag{5.14}$$

También será necesaria la siguiente suposición en relación con la recompensa.

Suposición 2. *La recompensa instantánea esperada obtenida después de transitar de s a s' cuando se toma la acción a está acotada, es decir, existe algún escalar B tal que:*

$$|R_t| \leq B < \infty, \quad \forall (s, a, s') \in \mathcal{S} \times \mathcal{A} \times \mathcal{S} \tag{5.15}$$

Tras estas consideraciones, se pasa a continuación a demostrar que los operadores de Bellman son contracciones.

Proposición 5.1. *Supongamos $0 < \gamma < 1$ y que la suposición 2 se cumple. Entonces, los operadores de Bellman, T_π y T, son contracciones:*

$$\|T_\pi v - T_\pi v'\|_\infty \leq \gamma \|v - v'\|_\infty, \qquad \forall v, v\,' \in \mathbb{R}^{|\mathcal{S}|} \tag{5.16}$$

$$\|Tv - Tv'\|_\infty \leq \gamma \|v - v'\|_\infty, \qquad \forall v, v' \in \mathbb{R}^{|\mathcal{S}|} \tag{5.17}$$

Demostración. La demostración es similar para T_π y para T, con lo cual sólo se demostrará el cumplimiento para la primera de ellas. Lo primero de todo, destacar que, puesto que $\mathcal{P}^\pi$ es una matrix estocástica, todos sus elementos son menores o iguales que 1. En consecuencia se cumple:

$$\|\mathcal{P}^\pi v\|_\infty \leq \|v\|_\infty \tag{5.18}$$

Ahora se usará esta relación para la definición del operador:

$$\begin{aligned}\|T_\pi(v) - T_\pi(v')\|_\infty &= \|\mathcal{R}^\pi + \gamma\mathcal{P}^\pi v - (\mathcal{R}^\pi + \gamma\mathcal{P}^\pi v')\|_\infty \\ &= \gamma\|\mathcal{P}^\pi(v - v')\|_\infty \\ &\leq \gamma\|v - v'\|_\infty \end{aligned} \tag{5.19}$$

□

Las contracciones tienen propiedades muy útiles e interesantes. Por ello, el haber demostrado que los operadores de Bellman son contracciones supone una gran ventaja. En concreto, podemos aplicar el teorema de las aplicaciones contractivas o teorema del punto fijo de Banach de la siguiente manera:

Teorema 5.1. *Lo siguiente se cumple para el operador de Bellman, para alguna política π:*

1. Existe una única solución v^π a la ecuación de punto fijo $T_\pi v = v$.
2. Para cualquier $v_0 \in \mathbb{R}^{|\mathcal{S}|}$, la iteración de punto fijo dada por $v_{t+1} = T_\pi(v_t)$ converge a v^π a medida que $t \to \infty$.
3. Las iteraciones v_t satisfacen las siguientes cotas de error:

$$\|v_t - v^\pi\| \leq \frac{\gamma^t}{1-\gamma}\|v_1 - v_0\| \tag{5.20}$$

$$\|v_t - v^\pi\| \leq \frac{\gamma}{1-\gamma}\|v_t - v_{t-1}\| \tag{5.21}$$

Teorema 5.2. *Lo siguiente se cumple para el operador de Bellman óptimo:*

1. Existe una única solución v^* a la ecuación de punto fijo $Tv = v$.
2. Para cualquier $v_0 \in \mathbb{R}^{|\mathcal{S}|}$, la iteración de punto fijo dada por $v_{t+1} = T(v_t)$ converge a v^* a medida que $t \to \infty$.
3. Las iteraciones v_t satisfacen las siguientes cotas de error:

$$\|v_t - v^*\| \leq \frac{\gamma^t}{1-\gamma}\|v_1 - v_0\| \tag{5.22}$$

$$\|v_t - v^*\| \leq \frac{\gamma}{1-\gamma}\|v_t - v_{t-1}\| \tag{5.23}$$

En resumen, los teoremas 5.1 y 5.2 enuncian que si aplicamos de manera recursiva el operador de Bellman (óptimo) a cualquier función $v : \mathcal{S} \to \mathbb{R}$, de manera asintótica

obtendremos la función valor de estados (óptima). Esto da pie a pensar en el uso de métodos iterativos de cara a encontrar las funciones valor de estados v^π (para una política π dada) o para encontrar la función valor de estados óptima v^*. Las mismas conclusiones pueden extraerse para la función valor de estados-acciones, q^π y q^*. En la siguiente sección emplearemos estas ideas para presentar dos algoritmos: *value iteration* (VI) y *policy iteration* (PI), los cuales son conocidos como métodos de programación dinámica.

5.1.2. Métodos de programación dinámica

Cuando se trata con MDPs, se consideran dos problemas diferentes:

1. Predicción: consiste en encontrar la función valor de estados o de estados-acciones para una política dada. Este problema se conoce típicamente como evaluación de la política (*policy evaluation*) ya que la función valor nos dirá cómo de buena es una política (en términos de recompensa acumulada esperada)

2. Control: consiste en encontrar la política óptima; es decir, la política que lleva a la función valor óptima.

Los métodos de programación dinámica son directamente derivados de las propiedades de convergencia de los operadores de Bellman, presentadas en el teorema 5.1.

Predicción

Para el problema de predicción, se van a considerar las siguientes iteraciones derivadas del teorema 5.1:

$$v_{t+1} = T_\pi\left(v_t\right) = \mathcal{R}^\pi + \gamma \mathcal{P}^\pi v_t \tag{5.24}$$

$$q_{t+1} = T_\pi\left(q_t\right) = \mathcal{R} + \gamma \mathcal{P} \Pi\, q_t \tag{5.25}$$

Estas recursiones dan pie al algoritmo llamado *policy evaluation* (PE), el cual converge asintóticamente a las funciones valor v^π o q^π para una política π dada. En la práctica, para el problema de predicción, la función valor de estados v^π es todo lo que necesitamos conocer para poder evaluar cómo de buena es una política, de manera que q^π no suele ser calculado. En el problema de control la situación es diferente y existen algunas variaciones algorítmicas prácticas tanto para la función valor de estados como de estados-acciones. Además, tal y como se verá en la sección 5.2, la función valor de estados-acciones será generalmente preferida para aprender la política óptima.

La figura 5.1 ilustra el proceso por el cual se actualiza a $v_{t+1}\left(s\right)$ a partir de $v_t\left(s\right)$.

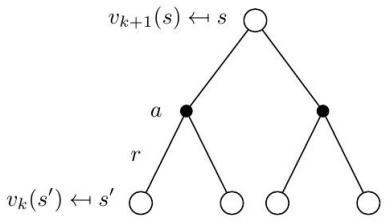


Figura 5.1: Esquema empleado para la actualización de la función valor en PE.

Control

El objetivo del problema de control es obtener la política óptima (es decir, la política con mayor función valor asociada). De cara a mejorar una política π dada para obtener otra política π' tal que $v^{\pi'} \geq v^{\pi}$, el agente puede considerar si debería cambiar la política en un determinado estado por otra acción $a \neq \pi(s)$. Una manera de comprobar si la política resultante tras este cambio ha mejorado la función valor, es evaluar el resultado de seleccionar a en el estado s, y después seguir $\pi(s)$, lo cual es equivalente a calcular la función valor de estados-acciones, o simplemente la «función q».

$$q^{\pi}(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v^{\pi}(s') \tag{5.26}$$

Una forma intuitiva de encontrar una política que mejore $q^{\pi}(s,a)$ es la política *greedy*, $\pi'(s)$, dada por (4.41), tal que:

$$\pi'(s) = \arg \max_{a \in \mathcal{A}} q^{\pi}(s,a) \tag{5.27}$$

El teorema 5.3, conocido como el teorema de mejora de la política *(policy improvement theorem)* formaliza estas ideas.

Teorema 5.3. *Sean $\pi'(s)$ y $\pi(s)$ cualquier par de políticas deterministas que satisfacen la siguiente desigualdad para todo $s \in \mathcal{S}$*

$$q^{\pi}(s, \pi'(s)) \geq v^{\pi}(s), \quad \forall s \in \mathcal{S} \tag{5.28}$$

Entonces, se tiene que:

$$v^{\pi'}(s) \geq v^{\pi}(s) \tag{5.29}$$

Es decir, el teorema 5.3 establece que reemplazando $\pi(s)$ por la política *greedy* $\pi'(s)$ dada por (5.27), la función valor resultante $v^{\pi'}$es mejor o igual que la función valor original v^{π}. El algoritmo consistente en la iteración de estas dos fases de evaluación de la política y mejora de la política se conoce como *policy iteration* (PI), y típicamente es representado

como una secuencia de políticas que mejoran (*improvement*, I) de manera monótona a partir de funciones valor evaluadas (*evaluation*, E)

$$\pi_0(s) \xrightarrow{E} v^{\pi_0}(s) \xrightarrow{I} \pi_1(s) \xrightarrow{E} v^{\pi_1}(s) \xrightarrow{E} \cdots \xrightarrow{I} \pi^*(s) \xrightarrow{E} v^*(s) \tag{5.30}$$

$$\pi_0(s) \xrightarrow{E} q^{\pi_0}(s,a) \xrightarrow{I} \pi_1(s) \xrightarrow{E} q^{\pi_1}(s,a) \xrightarrow{E} \cdots \xrightarrow{I} \pi^*(s) \xrightarrow{E} q^*(s,a) \tag{5.31}$$

Un enfoque alternativo consiste en aplicar directamente el operador de Bellman óptimo a la función valor de manera recursiva, tal y como se derivó en el teorema (5.2). Puesto que el operador óptimo de Bellman es una contracción, este procedimiento, conocido como *value iteration* (VI) también converge a la función valor óptima. A continuación, ya se puede extraer la política óptima como aquella que es *greedy* respecto a la función valor óptima para cada estado.

Curiosamente, se puede encontrar una cierta relación entre VI y PI de la siguiente manera: como ya se ha mencionado, en cada instante PI lleva a cabo la evaluación de la política, de manera que se tiene que esperar hasta que haya convergido la función valor de la política actual. Sin embargo, en lugar de esperar a la convergencia, podríamos tolerar cierto error y truncar la etapa de evaluación de la política en un punto determinado, de manera que a continuación se llevase a cabo el paso de mejora de la política sobre la aproximación truncada de la función valor. Ahora consideremos el caso extremo en el cual sólo se realiza una iteración del bucle de evaluación de la política y justo después se toma la política *greedy*. Este caso extremo es equivalente a aplicar el operador de Bellman óptimo. En otras palabras, podemos ver VI como un caso extremo de PI en el cual el paso de evaluación de la política es aproximado con una única iteración. Como un caso intermedio, se puede considerar más de una iteración de la etapa de evaluación de la política antes de pasar a mejorarla. Este enfoque es lo que se conoce como policy iteration generalizado (*generalized policy iteration*, GPI), y presenta PI y VI como casos extremo particulares. La figura 5.2 muestra de manera esquemática este proceso. A continuación se mostrarán las etapas de PI y VI en mayor detalle. Estos algoritmos de programación dinámica constituyen la base de los algoritmos de aprendizaje que se estudiarán en la sección 5.2

Policy iteration

Policy iteration evalúa las políticas por medio de sus funciones valor asociadas, y a su vez usa esas funciones valor para encontrar nuevas políticas. El algoritmo empieza con una política arbitraria π_0. En cada iteración t se determina la función valor de la política actual, v^{π_t} o q^{π_t}, por medio del algoritmo PE, que lo que hace es aplicar de manera recursiva (5.24) o (5.25). Una vez la fase de PE se ha completado, se mejora la política

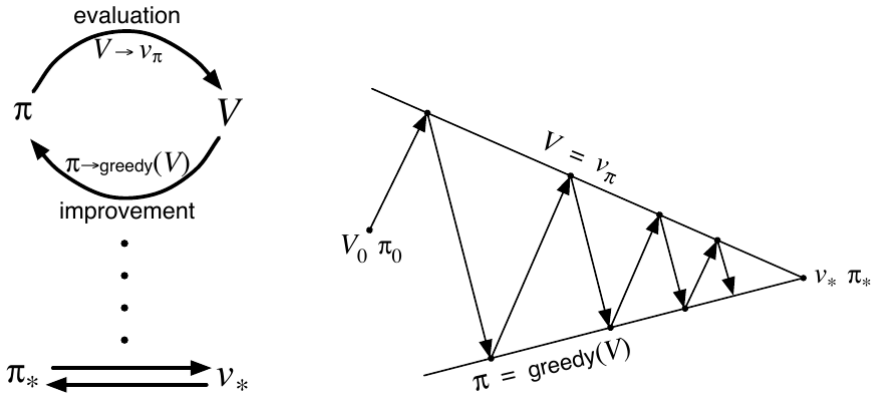


Figura 5.2: Algoritmo policy iteration generalizado (GPI).

tomando la política *greedy* dada por (5.27):

$$\pi_{t+1}(s) \in \arg\max_{a\in\mathcal{A}} q^{\pi_t}(s,a), \quad \forall s \in \mathcal{S} \tag{5.32}$$

Value iteration

Value iteration comienza con una función valor inicial, v_0 o q_0, y a continuación emplea las siguientes recursiones del operador de Bellman, derivadas del teorema 5.2:

$$v_{t+1} = Tv_t \tag{5.33}$$

$$q_{t+1} = Tq_t \tag{5.34}$$

que en su versión desarrollada para todo estado y todo par estado-acción son:

$$v_{t+1}(s) = \max_{a\in\mathcal{A}} \left[\mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{ss'}^a v_t(s')\right]$$

$$q_{t+1}(s,a) = \mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{ss'}^a \max_{a'\in\mathcal{A}} q_t(s',a')$$

Debido a la propiedad de contracción del operador de Bellman óptimo enunciada en el teorema (5.2), podemos garantizar la convergencia de este algoritmo a las funciones valor óptimas, $v^*(s)$ o $q^*(s,a)$. Se recuerda que cualquier política que es *greedy* respecto a la función valor óptima será óptima. Esta conclusión se expresa de manera formal, empleando la notación del operador de Bellman, en la proposición 5.2.

Proposición 5.2. *Una política estacionaria $\pi^* : \mathcal{S} \to \mathcal{A}$ es óptima sí y solo sí $\pi^*(s)$ lleva a obtener la función valor óptima* (4.45) *para cada $s \in \mathcal{S}$. Esto se puede expresar*

de manera compacta de la siguiente manera:

$$T_{\pi^*} v^* = T v^* \tag{5.35}$$

De manera equivalente, se puede decir que π^* es greedy con respecto a la función valor óptima:

$$\pi^*(s) = \arg\max_{a \in \mathcal{A}} \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v^*(s') \right) = \arg\max_{a \in \mathcal{A}} \; q^*(s, a), \quad \forall s \in \mathcal{S} \tag{5.36}$$

Por tanto, una vez que se ha encontrado q^*, se puede obtener fácilmente la política óptima calculando la política *greedy* con respecto a q^*.

5.2. Diferencias temporales

Ahora se van a pasar a considerar problemas en los que la distribución de probabilidad de transiciones de estados y/o la distribución de recompensas son desconocidas. Esta configuración es la que típicamente se conoce como *libre de modelo*, en el sentido de que el agente aprende sin tener cualquier modelo previo del entorno.

Al igual que en el caso basado en modelo, donde se presentó la programación dinámica como método principal para resolver el problema de control óptimo, distinguiremos dos subproblemas: predicción y control.

La idea ahora será resolver las ecuaciones de Bellman empleando estimaciones de la función valor. Esta idea será muy efectiva y como veremos, será la base del método de las diferencias temporales (*temporal difference*, TD).

5.2.1. Predicción

El problema de predicción pretende ahora estimar la función valor de una política dada, para un MDP. Para el problema de planificación de DP, se vio que policy evaluation era capaz de predecir la función valor tomando el valor esperado de las recompensas futuras. Este proceso se representa en la figura 5.3, donde la zona roja indica que de cara a predecir la función valor en S_t, se hace el promedio de las recompensas sobre todas las posibles transiciones de estados S_{t+1}.

Cuando no se tiene acceso a las distribuciones de transición de estados o de las recompensas, no se puede calcular la recompensa esperada exacta. Afortunadamente sí se podrán usar métodos para estimar la recompensa esperada. De entre todos los existentes, vamos a centrarnos únicamente en TD por ser la base de diversos algoritmos de RL empleados a día de hoy.

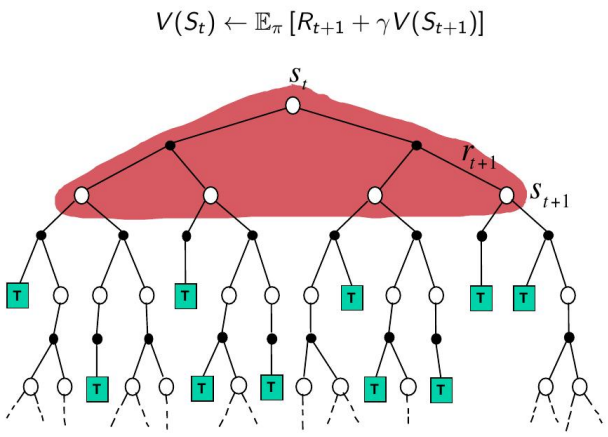


Figura 5.3: Representación del funcionamiento de la programación dinámica.

La manera inicial que se propone para estimar la función valor se conoce como método de Monte-Carlo (MC):

$$v\left(S_t\right) \leftarrow v\left(S_t\right) + \alpha_t \left(G^\pi\left(S_t\right) - v\left(S_t\right)\right) \tag{5.37}$$

Recordando a partir de (4.19) y (4.20), la función valor para alguna política π es el retorno esperado (es decir, la recompensa acumulada) sobre todas las posibles trayectorias inducidas por π. Por comodidad en el seguimiento de los desarrollos, se presenta a continuación de nuevo la definición de retorno:

$$G^\pi\left(S_t\right) \triangleq R_{t+1} + \gamma R_{t+2} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{5.38}$$

donde se ha introducido una notación más específica: $G^\pi\left(S_t\right) \triangleq G_t$.

No obstante, el método MC es bien sabido que presenta mucha varianza, de manera que se requiere de una cantidad elevada de muestras para poder reducirla, y no siempre se puede disponer de ellas. Para solucionar este problema, aparece el método de las diferencias temporales, TD, y lo que propone es estimar también el retorno como la suma de la recompensa instantánea y la estimación de la función valor en el estado futuro:

$$\text{MC}: \quad v\left(S_t\right) \leftarrow v\left(S_t\right) + \alpha_t \left(G^\pi\left(S_t\right) - v\left(S_t\right)\right) \tag{5.39}$$

$$\text{TD}: \quad v\left(S_t\right) \leftarrow v\left(S_t\right) + \alpha_t \left(R_{t+1} + \gamma v\left(S_{t+1}\right) - v\left(S_t\right)\right) \tag{5.40}$$

donde α_t hace referencia al tamaño del paso o tasa de aprendizaje, perteneciente a una

secuencia tal que se cumple la siguiente propiedad:

$$\sum_{t=0}^{\infty} \alpha_t = \infty \quad \text{y} \quad \sum_{t=0}^{\infty} \alpha_t^2 < \infty \tag{5.41}$$

Por tanto, el retorno (5.38) se puede reescribir como:

$$G^{\pi}(S_t) = R_{t+1} + \gamma v^{\pi}(S_{t+1}) \tag{5.42}$$

TD estima v^{π} en (5.42) aproximando $G^{\pi}(S_t)$ con otra aproximación, $\widehat{G}^{\pi}(S_t)$, dada por:

$$\widehat{G}^{\pi}(S_t) \triangleq R_{t+1} + \gamma v(S_{t+1}) \approx G^{\pi}(S_t) \tag{5.43}$$

En otras palabras, TD remplaza la función valor verdadera v^{π} por su estimación actual v. Esta idea de actualizar una estimación a partir de otra es lo que se conoce como *bootstrapping* en la literatura de RL y aparece ilustrada en la figura 5.4.

Comúnmente, se emplea la siguiente terminología:

- El retorno de *bootstrapping*, $\widehat{G}^{\pi}(S_t)$, es conocido como *TD target.*
- El llamado *TD error* se define como:

$$\delta_t \triangleq R_{t+1} + \gamma v(S_{t+1}) - v(S_t) \tag{5.44}$$

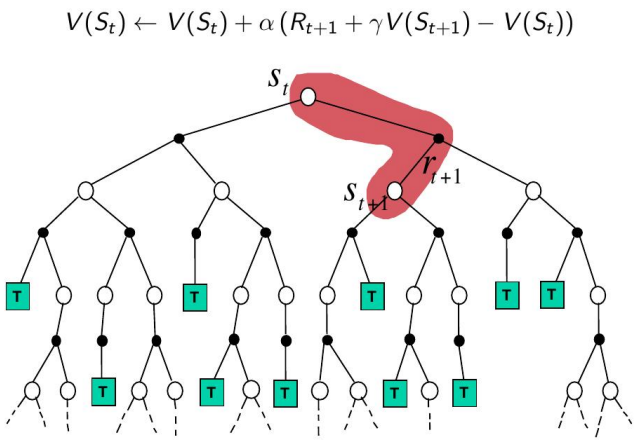


Figura 5.4: Diagrama de *temporal difference.*

De acuerdo a todos estos conceptos, se podrá obtener una recursión de TD para las funciones valor de estados-acciones siguiendo el mismo razonamiento que para las funciones valor de estados. Primero, consideremos el retorno de estados-acciones definido

por:

$$G^{\pi}\left(S_t, A_t\right) = R_{t+1} + \gamma q^{\pi}\left(S_{t+1}, A_{t+1}\right) \tag{5.45}$$

donde $A_{t+1} \sim \pi$. TD aproxima $G^{\pi}\left(S_t, A_t\right)$ con otro término dado por:

$$\widehat{G}^{\pi}\left(S_t, A_t\right) \triangleq R_{t+1} + \gamma q\left(S_{t+1}, A_{t+1}\right), \quad A_{t+1} \sim \pi \tag{5.46}$$

tal que:

$$\widehat{G}^{\pi}\left(S_t, A_t\right) \approx G^{\pi}\left(S_t, A_t\right)$$

De este modo, se obtiene:

$$q\left(S_t, A_t\right) \leftarrow q\left(S_t, , A_t\right) + \alpha_t\left(R_{t+1} + \gamma q\left(S_{t+1}, A_{t+1}\right) - q\left(S_t, , A_t\right)\right) \tag{5.47}$$

donde de nuevo, α_t hace referencia al tamaño del paso o tasa de aprendizaje, perteneciente a una secuencia que satisface (5.41).

El pseudocódigo del algoritmo TD para funciones valor de estados y de estados-acciones viene dado por los algoritmos 5.1 y 5.2, respectivamente. Como puede apreciarse, se ha considerado un valor de α constante para todos los instantes t.

Algoritmo 5.1 TD para las funciones valor de estados.

Entrada: π, la política a ser evaluada.
Salida: v, la estimación de la función valor v^{π}.

```
Inicializar v(s) arbitrariamente (e.g., v(s) = 0), para todo s ∈ S
repetir(para cada episodio)
    Inicializar s
    repetir(para cada paso en el episodio)
        Elegir la acción a ∼ π(·|s)
        Tomar la acción a y observar r, s'
        v(s) ← v(s) + α(r + γv(s') − v(s))
        s ← s'
    hasta que s sea terminal
hasta que no podamos correr más episodios
devolver v
```

Resulta importante destacar que las predicciones por TD suelen presentar mucha menos varianza que las predicciones por MC. Esta reducción de varianza tiene el coste adherido de introducir un término de sesgo. Además de esta diferencia, TD generalmente converge más rápido que MC.

A continuación se extenderá el método TD con el objetivo de hacer predicciones para resolver el problema de control, y se presentarán dos algoritmos de control basados en TD: Q-learning y SARSA.

Algoritmo 5.2 TD para las funciones valor de estados-acciones.

Entrada: π, la política a ser evaluada.
Salida: q, la estimación de la función valor q^{π}.

Inicializar $q(s,a)$ arbitrariamente (e.g., $q(s,a)=0$), para todo $(s,a) \in \mathcal{S} \times \mathcal{A}$
repetir(para cada episodio)
    Inicializar s, a
    repetir(para cada paso en el episodio)
        Tomar la acción a y observar r, s'
        Elegir la acción $a' \sim \pi(\cdot|s)$
        $q(s,a) \leftarrow q(s,a) + \alpha(r + \gamma q(s',a') - q(s,a))$
        $s \leftarrow s'$
        $a \leftarrow a'$
    hasta que s sea terminal
hasta que no podamos correr más episodios
devolver q

5.2.2. Control

Como es lógico pensar, para llevar a cabo un control efectivo, será necesario explorar todos los pares estado-acción para obtener así estimaciones más precisas de las funciones valor, y por tanto ser capaces de escoger de manera apropiada las acciones que conducirán a las mayores estimaciones de las funciones valor. La necesidad de establecer un compromiso entre exploración y explotación es de vital importancia en RL. La idea fundamental a tener en mente es que se está estimando, a partir de muestras, la distribución del retorno condicionado a los pares estado-acción. Por tanto, lo que se pretende conseguir es explotar las acciones que han resultado en mayores estimaciones de la función valor hasta el momento, pero al mismo tiempo, explorar otras acciones por si estas pudieran provocar incluso mayores recompensas.

Diseñar políticas de comportamiento que aseguren un compromiso óptimo de exploración vs. explotación es un área de investigación activa; no obstante, se aparta de la temática de este trabajo. En lugar de ello, se va a presentar una política de comportamiento heurística que ha resultado obtener buenos resultados en la práctica: la política $\epsilon-$greedy. La política $\epsilon-$greedy es la siguiente:

$$\pi_{\epsilon}(a \mid s) \triangleq \begin{cases} 1-\epsilon+\epsilon/|\mathcal{A}|, & \text{si } a = \arg\max_{a' \in \mathcal{A}} \; q(s,a') \\ \epsilon/|\mathcal{A}|, & a \in \mathcal{A} \end{cases} \tag{5.48}$$

donde $\epsilon \in (0,1)$ es el parámetro de exploración, tal que cuanto más grande sea ϵ, mayor exploración y menor explotación se producirán.

A continuación se procede a presentar dos algoritmos muy conocidos en RL: SARSA y Q-learning, los cuales son aproximaciones de los algoritmos PI y VI, respectivamente,

basadas en TD.

SARSA

SARSA es una aproximación del algoritmo PI. Dado que TD aprende a partir de tuplas de muestras de la forma $(s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1})$, el nombre SARSA viene de unir las iniciales de cada nombre en la tupla de datos empleada por el algoritmo: estado (*state*), acción, recompensa, (siguiente) estado, (siguiente) acción. SARSA está basado en tres ideas clave:

La primera de ellas consiste en reemplazar q^π por su estimación por *bootstrapping*, para la política actual π_ϵ, de manera que la actualización de la estimación de la función valor de estados-acciones se convierte en:

$$q\left(s_t, a_t\right) \leftarrow q\left(s_t, a_t\right) + \alpha_t \left(R_{t+1} + \gamma q\left(s_{t+1}, a_{t+1}\right) - q\left(s_t, a_t\right)\right) \tag{5.49}$$

donde $a_{t+1} \sim \pi_\epsilon\left(\cdot|s_{t+1}\right)$ es la acción tomada de la política actual, y α_t es el tamaño del paso o tasa de aprendizaje perteneciente a alguna secuencia que satisfaga (5.41). La figura 5.5 ilustra la actualización para las iteraciones de TD (5.49).

La segunda idea es llevar a cabo el paso de mejora de la política tomando la política $\epsilon-$greedy respecto a la estimación actual de la función q. Resaltar que el *TD target* es el retorno por *bootstrapping* con respecto a la política ϵ-greedy, es decir, $\widehat{G}^{\pi_\epsilon}\left(S_t\right)$. Por tanto, dado que la política de comportamiento empleada para obtener las tuplas de datos y la política objetivo son la misma (e igual a la política ϵ-greedy denotada por π_ϵ), se dice que SARSA es un algoritmo de aprendizaje *on-policy*.

La tercera idea es realizar el paso de mejora de la política sin haber esperado a la convergencia de la estimación (5.49) de la función valor obtenida por *bootstrapping*. En lugar de ello, la implementación más común de SARSA consiste en realizar una única iteración de TD (5.49) y a continuación llevar a cabo la mejora de la política.

El esquema iterativo de SARSA aparece ilustrado en la figura 5.6. El pseudocódigo de SARSA se puede encontrar también en el algoritmo 5.3.

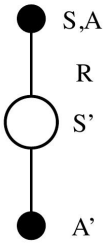


Figura 5.5: Diagrama de actualización del algoritmo SARSA

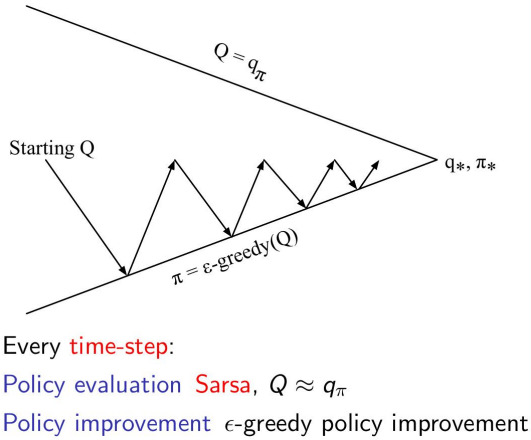


Figura 5.6: Algoritmo SARSA para control on-policy.

Algoritmo 5.3 Algoritmo SARSA con política objetivo ϵ-greedy, para MDPs episódicos.

Entrada: Parámetro de exploración ϵ y tasa de aprendizaje α.
Salida: π, la política óptima $\pi^\star$ aproximada.

Inicializar $q(s,a)$ arbitrariamente para todo $(s,a) \in \mathcal{S} \times \mathcal{A}$
Inicializar $q(s,\cdot) = 0$ para todos los estados terminales
repetir(para cada episodio)
    Inicializar s
    Escoger la acción $a \sim \pi_\epsilon(\cdot|s)$ usando la política ϵ-greedy (5.48) a partir de la q actual
    repetir(para cada paso en el episodio)
        Tomar la acción a y observar r, s'
        Escoger la acción $a' \sim \pi_\epsilon(\cdot|s')$ usando (5.48) a partir de la q actual
        $q(s,a) \leftarrow q(s,a) + \alpha(r + \gamma q(s',a') - q(s,a))$
        $s \leftarrow s'$, $a \leftarrow a'$
    hasta que s sea terminal
hasta que no podamos correr más episodios
para todo $s \in \mathcal{S}$ **hacer**
    $\pi(s) \leftarrow \arg\max_{a \in \mathcal{A}} q(s,a)$
devolver π

Q-learning

El algoritmo Q-learning es una aproximación del algoritmo VI, y está basado en dos simples ideas:

La primera de ellas consiste en remplazar q^π con su estimación por *bootstrapping* en el operador de Bellman óptimo, de manera que la actualización de la estimación de la función valor de estados-acciones se convierte en:

$$q\left(s_t, a_t\right) \leftarrow q\left(s_t, a_t\right) + \alpha_t \left(R_{t+1} + \gamma \max_{a \in \mathcal{A}} \; q\left(s_{t+1}, a\right) - q\left(s_t, a_t\right) \right) \tag{5.50}$$

donde, como de costumbre, α_t es el tamaño del paso o tasa de aprendizaje perteneciente a alguna secuencia que satisfaga (5.41). Esta actualización es diferente de la de SARSA en el sentido de que aquí estamos aproximando el operador de Bellman *óptimo* en lugar del operador de Bellman para la política actual. Desde una perspectiva de TD, recalcar que el término $R_{t+1} + \gamma \max_{a \in \mathcal{A}} \; q\left(s_{t+1}, a\right)$ es la aproximación por *bootstrapping* del retorno obtenido al seguir la política *greedy*. Este es un punto fundamental de Q-learning: la función valor se estima para la política *greedy*. Con el objetivo de ver este matiz de una forma más clara, digamos que π_{greedy} denota la política *greedy* tal que:

$$\pi_{\text{greedy}}\left(a|s\right) = \begin{cases} 1, & a = \arg\max_{a' \in \mathcal{A}} \; q\left(s, a'\right) \\ 0, & \text{resto} \end{cases} \tag{5.51}$$

Entonces, de acuerdo a (5.45), el retorno obtenido con la política *greedy* estará dado por:

$$G^{\pi_{\text{greedy}}}\left(s_t\right) = R_{t+1} + \gamma \max_{a \in \mathcal{A}} \; q^{\pi_{\text{greedy}}}\left(s_{t+1}, a\right) \tag{5.52}$$

Por tanto, Q-learning emplea la siguiente aproximación:

$$\widehat{G}^{\pi_{\text{greedy}}}\left(s_t\right) = R_{t+1} + \gamma \max_{a \in \mathcal{A}} \; q\left(s_{t+1}, a\right) \approx G^{\pi_{\text{greedy}}}\left(s_t\right) \tag{5.53}$$

La figura 5.7 ilustra el diagrama de actualización para la iteración (5.50).

La segunda idea consiste en emplear una política de comportamiento exploratoria tal como ϵ-greedy, dada por (5.48), con el objetivo de asegurar que todos los pares estado-acción sean evaluados. Puesto que la política objetivo es *greedy*, es diferente de la política de comportamiento ϵ-greedy. Por tanto, se dirá que Q-learning es un algoritmo *off-policy*.

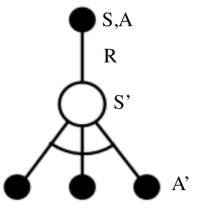


Figura 5.7: Diagrama de actualización para el algoritmo Q-learning.

El pseudocódigo de Q-learning se puede encontrar en el algoritmo 5.4.

Algoritmo 5.4 Algoritmo Q-learning con política de comportamiento ϵ-greedy, para MDPs episódicos.

Entrada: Parámetro de exploración ϵ y tasa de aprendizaje α.
Salida: π, la política óptima $\pi^\star$ aproximada.

Inicializar $q(s,a)$ arbitrariamente para todo $(s,a) \in \mathcal{S} \times \mathcal{A}$
Inicializar $q(s,\cdot) = 0$ para todos los estados terminales
repetir(para cada episodio)
    Inicializar s
    repetir(para cada paso en el episodio)
        Escoger la acción $a \sim \pi_\epsilon(\cdot|s)$ usando (5.48) a partir de la q actual
        Tomar la acción a y observar r, s'
        $q(s,a) \leftarrow q(s,a) + \alpha(r + \gamma \,\text{máx}_{a' \in \mathcal{A}}\; q(s',a') - q(s,a))$
        $s \leftarrow s'$
    hasta que s sea terminal
hasta que no podamos correr más episodios
para all $s \in \mathcal{S}$ **hacer**
    $\pi(s) \leftarrow \arg\,\text{máx}_{a \in \mathcal{A}}\; q(s,a)$
devolver π

Capítulo 6

Resolución de las ecuaciones de Bellman en problemas de gran escala

En este capítulo se va a abordar la resolución de las ecuaciones de Bellman cuando el problema adquiere mayor dimensionalidad y se desconoce el modelo del entorno.

En el capítulo anterior, se ha considerado un modelo muy conveniente en el cual los espacios de políticas y acciones eran de un tamaño moderado y discretos. En tal escenario, las funciones valor podían ser representadas a través de tablas donde cada estado tenía una entrada $v(s)$ o cada par estado-acción tenía una entrada $q(s, a)$. Esta asunción simplifica las cuestiones de implementación pero no es muy realista en muchos problemas reales. Un ejemplo típico donde es impracticable este enfoque es el Backgammon, un juego que presenta 10^{20} estados. Otro ejemplo podría ser el control de un helicóptero: el espacio de estados es continuo.

Para que el aprendizaje por refuerzo sea capaz de solucionar este tipo de problemas, será necesario hacer algún tipo de aproximación puesto que la solución exacta, en general no podrá ser calculada. De este modo, los algoritmos presentados en el capítulo anterior no podrán ser aplicados en su forma original y habrá que derivar una versión basada en aproximaciones que funcione para el caso en que desconocemos el modelo del entorno. Entrando más en detalles, se va a considerar que más que tener acceso al estado del entorno, el agente tiene acceso a un conjunto de características (*features*) del estado.

En este capítulo únicamente abordaremos aproximaciones lineales, donde la función valor para algún estado será aproximada como una combinación lineal del vector de características para dicho estado.

Supongamos que θ y ω hacen referencia a los vectores de parámetros para las aproximaciones de las funciones valor de estados y de estados-acciones respectivamente, de

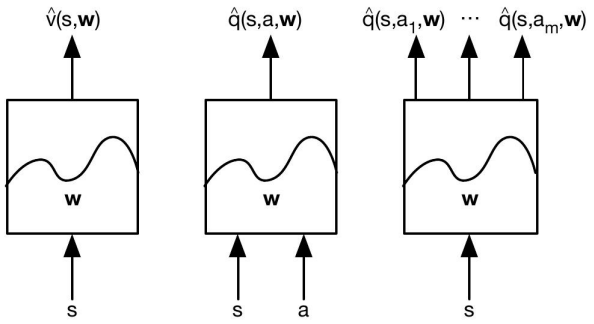


Figura 6.1: Tipos de aproximaciones de las funciones valor.

manera que se tiene:

$$v^{\pi}(s) \approx v^{\pi}(s, \omega) \tag{6.1}$$

$$q^{\pi}(s, a) \approx q^{\pi}(s, a, \theta) \tag{6.2}$$

Ahora el problema de hacer predicciones (es decir, de estimar el vector de funciones valor v^{π}) resulta equivalente a buscar el vector de parámetros $\omega^{\star}$ o $\theta^{\star}$ que es óptimo en cierto sentido.

Definamos $\bar{\phi} : \mathcal{S} \to \mathbb{R}^N$ como una aplicación de los estados a las características, tal que $\bar{\phi}(s)$ proporciona el vector de características de longitud N que representa el estado s. Las componentes de este vector estarán dadas por un conjunto de funciones base, $\phi_1, \ldots, \phi_N : \mathcal{S} \to \mathbb{R}$:

$$\bar{\phi}(s) = \begin{bmatrix} \bar{\phi}_1(s) \\ \vdots \\ \bar{\phi}_N(s) \end{bmatrix} \tag{6.3}$$

De entre muchos tipos de parametrizaciones, la aproximación lineal de la forma:

$$v^{\pi}(s, \omega) = \bar{\phi}(s)^T \omega \tag{6.4}$$

ha sido extensamente estudiada en la literatura y es prometedora, principalmente porque permite obtener soluciones con un bajo coste computacional. Además, si se es capaz de escoger la aplicación de las características de manera cuidadosa, tal que estas capturen correctamente la estructura de las recompensas y de las transiciones de estados, entonces el modelo de aproximación lineal dará, en general, buenos resultados.

Asumamos un conjunto finito de acciones por el momento: $\mathcal{A} \triangleq \{a_1, \cdots, a_A\}$. Para la aproximación de la función valor de estados-acciones, las funciones base serán denotadas por $\phi_1, \phi_2, \cdots, \phi_M : \mathcal{S} \times \mathcal{A} \to \Re$, las cuales se agrupan en el vector de características de

estados-acciones de la siguiente manera:

$$\phi\left(s,a\right)=\left[\begin{array}{c}\phi_1\left(s,a\right)\\ \vdots\\ \phi_M\left(s,a\right)\end{array}\right] \tag{6.5}$$

Resulta estándar obtener $\phi(s,a)$ a partir de $\bar{\phi}(s)$ empleando N características para cada posible acción, de manera que $M=NA$; estableciendo las N características correspondientes a la acción a igual a $\bar{\phi}(s)$; y las otras $(N-1)A$ características, que se corresponden con las acciones no elegidas, igual a cero:

$$\phi\left(s,a_m\right)^T=\left[\underbrace{0,\ldots,0}_{a_1},\ldots,\underbrace{\bar{\phi}_1\left(s\right),\ldots,\bar{\phi}_N\left(s\right)}_{a_m},\ldots,\underbrace{0,\ldots,0}_{a_A}\right] \tag{6.6}$$

De esta manera, la función $q(s,a)$ puede ser linealmente aproximada por el vector de parámetros $\theta\in\mathbb{R}^M$ como:

$$q\left(s,a_m,\theta\right)=\phi\left(s,a_m\right)^T\theta \tag{6.7}$$

6.1. Familias de funciones base

Las funciones base empleadas para representar el espacio pueden ser arbitrariamente complejas, incluso cuando la aproximación paramétrica es tan simple como la aproximación lineal. A continuación se presentarán los esquemas de funciones base más comunes: agregación de estados o discretización, base polinómica y funciones base radiales. Resulta importante destacar que escoger el conjunto de funciones base correcto puede ser crítico a la hora de alcanzar buenas aproximaciones.

6.1.1. Agregación de estados o discretización

Para la agregación de estados, el espacio de estados se particiona en N subconjuntos separados, siendo $\mathcal{S}_n$ el subconjunto n-ésimo de dicha partición, para $n=1,\cdots,N$. Para una acción dada, la aproximación asigna el mismo valor de $v(s)$ para todos los estados en $\mathcal{S}_n$. Esto se corresponde con un vector de características dependiente del estado, que toma valores binarios (0 o 1):

$$\bar{\phi}_n\left(s\right)=\begin{cases}1 & \text{si } s\in\mathcal{S}_n\\ 0 & \text{resto}\end{cases} \tag{6.8}$$

En este caso, el vector de características de estados-acciones (6.6) puede ser escrito

de manera compacta:

$$\phi_n(s, a_j) = \begin{cases} 1 & \text{si } s \in \mathcal{S}_n \text{ y } a = a_j \\ 0 & \text{resto} \end{cases} \tag{6.9}$$

El problema principal de este enfoque es que (1) el número de estados agregados crece de manera exponencial con la dimensión del conjunto de estados, $\mathcal{S}$, y (2) la variación entre dos estados consecutivos podría ser muy abrupta.

6.1.2. Base polinómica

Consideremos ahora que el espacio de estados es continuo, de manera que cada estado es descrito por un vector de S variables de estado: $s = (s_i)_{i=1}^{S}$. La base polinómica de orden j vendrá dada por:

$$\bar{\phi}_n(s) = \prod_{i=1}^{j} s_i^{c_{n,i}} \tag{6.10}$$

donde $c_{n,i}$ es un entero entre 0 y j.

6.1.3. Funciones base radiales Gaussianas (RBFs)

De nuevo, considérese que el espacio de estados es continuo y que el vector de estado está compuesto por S variables de estado. Una función base radial (*Radial Basis Function*, RBF) es una función que toma valores reales, cuyo valor depende únicamente en la distancia entre la variable de estado y algún punto de referencia c_n. Por simplicidad, se considerarán RBF de tipo Gaussiano, las cuales se definen de la siguiente manera:

$$\bar{\phi}_n(s) = \frac{1}{\sqrt{(2\pi)^S |B_n|}} e^{-\frac{1}{2}(s-c_n)^T B_n^{-1}(s-c_n)} \tag{6.11}$$

donde c_n y B_n son la media y la covarianza de la n-ésima función base, respectivamente. La figura 6.2 ilustra esta idea.

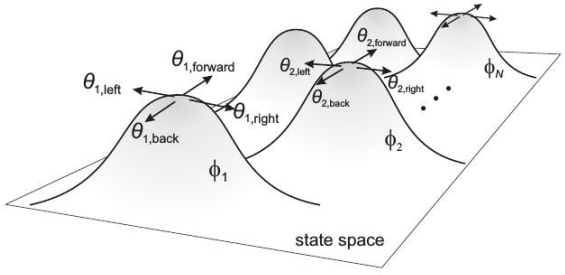


Figura 6.2: Aproximación de funciones empleando RBFs.

6.2. Ecuación de Bellman proyectada

Supongamos que se tiene la matriz Φ de dimensión $S \times N$, cuyas columnas son los vectores $\bar{\phi}_n$ de funciones base:

$$\overline{\Phi} \triangleq \begin{bmatrix} \bar{\phi}(1)^T \\ \vdots \\ \bar{\phi}(S)^T \end{bmatrix} = \begin{bmatrix} \bar{\phi}_1(1) & \cdots & \bar{\phi}_N(1) \\ \vdots & \cdots & \vdots \\ \bar{\phi}_1(S) & \cdots & \bar{\phi}_N(S) \end{bmatrix} \tag{6.12}$$

Entonces, la aproximación lineal (6.4) se puede expresar de manera vectorial, más compacta, como:

$$v^\pi = \overline{\Phi}\omega \tag{6.13}$$

De cara a resolver para ω, se puede remplazar v^π por su aproximación (6.13) en (4.31), de manera que se obtiene un sistema lineal de ecuaciones conocido como ecuación de Bellman aproximada:

$$\overline{\Phi}\omega = \mathcal{R}^\pi + \gamma \mathcal{P}^\pi \overline{\Phi}\omega \tag{6.14}$$

Siguiendo el mismo procedimiento para las funciones valor de estados-acciones, se define Φ como la siguiente matriz de dimensión $SA \times M$:

$$\Phi \triangleq \begin{bmatrix} \phi(1,1)^T \\ \vdots \\ \phi(1,A)^T \\ \vdots \\ \phi(S,1)^T \\ \vdots \\ \phi(S,A)^T \end{bmatrix} = \begin{bmatrix} \phi_1(1,1) & \cdots & \phi_M(1,1) \\ \vdots & \cdots & \vdots \\ \phi_1(1,A)^\top & \cdots & \phi_M(1,1) \\ \vdots & \cdots & \vdots \\ \phi_1(S,1)^\top & \cdots & \phi_M(S,1)^\top \\ \vdots & \cdots & \vdots \\ \phi_1(S,A)^\top & \cdots & \phi_M(S,A)^\top \end{bmatrix} \tag{6.15}$$

y se aproxima la ecuación de Bellman para estados-acciones (4.32) como:

$$\Phi\theta = \mathcal{R} + \gamma \mathcal{P} \Pi \Phi\theta \tag{6.16}$$

Se supondrá que las características constituyen un conjunto de funciones base linealmente independientes que representan de manera efectiva los estados, y que $\overline{\Phi}$ y Φ son de rango completo por construcción. Sin embargo, las ecuaciones de punto fijo (6.14) y (6.16) podrían no tener una solución ω o θ ya que el lado derecho no tiene por qué caer en el rango de $\overline{\Phi}$ y Φ respectivamente. De cara a solucionar este problema, se va a proyectar el lado derecho de (6.14)–(6.16) (es decir, el operador de Bellman) sobre el rango de la correspondiente matriz de características

Suponiendo que $\overline{\mathbb{X}}$ y $\mathbb{X}$ denotan el espacio del rango de las matrices de características $\overline{\Phi}$ y Φ, se define el operador proyección con respecto a una métrica $\|\cdot\|_D$ como:

$$\overline{\Xi}v \triangleq \arg\min_{x' \in \overline{\mathbb{X}}} \|v - x'\|_D^2 \tag{6.17}$$

$$\Xi q \triangleq \arg\min_{x' \in \mathbb{X}} \|q - x'\|_D^2 \tag{6.18}$$

donde $\overline{\Xi}$ y Ξ son los operadores proyección en $\overline{\mathbb{X}}$ y $\mathbb{X}$ respectivamente. Supondremos que $\overline{D}$ y D son matrices diagonales que contienen la probabilidad estacionaria límite de visita de estados y de estados acciones asociada a los estados y estados-acciones de la cadena de Markov; es decir, $\overline{D} = \text{diag}\,[d^\pi]$ donde d^π viene dada por (4.11). Por tanto, las matrices $\overline{\Xi}$ y Ξ vendrán dadas por:

$$\overline{\Xi} = \overline{\Phi}\left(\overline{\Phi}^\top \overline{D\Phi}\right)^{-1} \overline{\Phi}^T \overline{D} \tag{6.19}$$

$$\Xi = \Phi\left(\Phi^\top D\Phi\right)^{-1} \Phi^T D \tag{6.20}$$

En consecuencia, se definirá la ecuación de Bellman proyectada (*projected Bellman equation*, PBE) para las funciones valor de estados y estados-acciones de la siguiente manera:

$$\overline{\Phi}\omega = \overline{\Xi}\left(\mathcal{R}^\pi + \gamma \mathcal{P}^\pi \overline{\Phi}\omega\right) = \overline{\Xi} T\left(\overline{\Phi}\omega\right) \tag{6.21}$$

$$\Phi\theta\Xi\left(\mathcal{R} + \gamma \mathcal{P}\Pi\Phi\theta\right) = \Xi T\left(\Phi\theta\right) \tag{6.22}$$

6.2.1. Predicción

El problema de predicción consiste ahora en evaluar la política aproximada de una determinada política. Dado que la matriz de características o funciones base es dada por el entorno y es por tanto parte de la descripción de nuestro problema, la evaluación de la política consistirá en encontrar el parámetro ω o θ óptimo que resuelve la PBE. Se expone a continuación la manera de resolver este problema para el caso de la función valor de estados-acciones (6.22).

Desarrollando (6.20) en (6.22), se obtiene:

$$\Phi\theta = \Phi\left(\Phi^T D\Phi\right)^{-1} \Phi^T D\left(\mathcal{R} + \gamma \mathcal{P}\Pi\Phi\theta\right) \tag{6.23}$$

Multiplicando a ambos lados por $\left(\Phi^T D\Phi\right)^{-1} \Phi^T D$ y reorganizando el resultado, se llega a que el parámetro óptimo será:

$$\theta^* = \left(\Phi^T D\Phi - \gamma \Phi^T D\mathcal{P}\Pi\Phi\right)^{-1} \Phi^T D\mathcal{R} \tag{6.24}$$

Si se definen los siguientes términos:

$$\Gamma \triangleq \Phi^T D \Phi \tag{6.25}$$

$$\Lambda \triangleq \Phi^T D \mathcal{P} \Pi \Phi \tag{6.26}$$

$$z \triangleq \Phi^T D \mathcal{R} \tag{6.27}$$

se podrá reescribir (6.24) de la siguiente manera:

$$\theta^* = (\Gamma - \gamma \Lambda)^{-1} z \tag{6.28}$$

Cuando conozcamos el modelo del sistema, esta expresión nos permitirá obtener el parámetro óptimo de forma exacta. No obstante, cuando no conozcamos el modelo, habrá que aproximar los términos (6.25)–(6.27) a partir de muestras y finalmente resolver (6.28) con los términos aproximados. Un método que implementa esta solución basada en muestras es *Least-Squares Temporal-Difference* (LSTD), descrito en el algoritmo 6.1. Este algoritmo, primero estima los términos (6.25)–(6.27), y cuando tiene una estimación sólida (pasadas n muestras), calcula (6.28).

Algoritmo 6.1 Algoritmo LSTD para la evaluación de la política sobre las funciones valor de estados-acciones.

Entrada: π, la política a ser evaluada. Funciones base ϕ
Salida: θ, el parámetro estimado para la aproximación lineal de q^π.

Inicializar $\Gamma = 0_{M \times M}$, $\Lambda = 0_{M \times M}$, $z = 0_M$
repetir(para cada episodio)
    Inicializar s, a y observar $\phi(s, a)$
    repetir(para cada paso en el episodio)
        Tomar la acción a y observar $r, s', \phi(s')$
        Escoger la acción $a' \sim \pi(\cdot|s)$ y construir $\phi(s', a')$
        $\Gamma \leftarrow \Gamma + \phi(s, a)\phi(s, a)^\top$
        $\Lambda \leftarrow \Lambda + \phi(s, a)\phi(s', a')^\top$
        $z \leftarrow z + \phi(s, a)r$
        $s \leftarrow s'$
        $a \leftarrow a'$
    hasta que s sea terminal
hasta que no podamos correr más episodios
Calcular: $\theta = (\Gamma - \gamma \Lambda)^{-1} z$
devolver $\hat{q}_\theta = \Phi\theta$

6.2.2. Control

Para la etapa de control, se seguirá un enfoque similar al de *policy iteration* (PI), que como se vio en el capítulo anterior consiste en (1) evaluar la política actual y (2)

mejorar la política. Esta manera de resolver el problema de control recibe el nombre de *Least-Squares Policy Iteration* (LSPI).

Para la evaluación de la política, LSPI emplea LSTD de manera que se obtiene θ^* través de (6.28). A continuación mejora la política evaluada mediante la elección de la acción[1] *greedy* que maximiza la función valor de estados-acciones que se aproximó linealmente:

$$\pi(s) \in \arg\max_{a \in \mathcal{A}} \phi(s,a)^\top \theta^*, \quad \forall s \in \mathcal{S} \tag{6.29}$$

El conjunto de muestras $(\phi(s_t, a_t), r_{t+1}, \phi(s_{t+1}, a_{t+1}))$ que se empleen para el aprendizaje será actualizado entre cada iteración. Con el objetivo de garantizar un compromiso aceptable de exploración vs. explotación, se usará una política ϵ-greedy para tomar estas muestras:

$$\pi_\epsilon(a \mid s) \triangleq \begin{cases} 1 - \epsilon + \epsilon/|\mathcal{A}| & \text{si } a = \arg\max_{a' \in \mathcal{A}} \phi(s,a')^\top \theta \\ \epsilon/|\mathcal{A}| & \text{a} \in \mathcal{A} \end{cases} \tag{6.30}$$

Por último, para asegurar que LSPI aprende rápido, la mejora de la política podrá realizarse una vez cada pocas transiciones de la fase de evaluación de la política, antes de tener una evaluación muy precisa de la política actual. Este número de transiciones entre actualizaciones de la política será K.

El pseudocódigo de LSPI se muestra en el algoritmo 6.2.

[1] Se está asumiendo un conjunto de acciones finito, de manera que la maximización sobre el conjunto de acciones puede llevarse a cabo de manera relativamente eficiente.

Algoritmo 6.2 LSPI.

Entrada: Parámetro de exploración ϵ.
Salida: π, la política óptima $\pi^\star$ aproximada.

Inicializar $\Gamma = 0_{M\times M}$, $\Lambda = 0_{M\times M}$, $z = 0_M$, $t = 0$
Inicializar θ aleatoriamente
repetir(para cada episodio)
    Inicializar s, a y determinar $\phi(s, a)$
    repetir(para cada paso en el episodio)
        Tomar la acción $a \sim \pi_\epsilon(\cdot|s)$ empleando (6.30) y observar r, s'
        Escoger la acción $a' \sim \pi_\epsilon(\cdot|s')$ empleando (6.30) y determinar $\phi(s', a')$
        $\Gamma \leftarrow \Gamma + \phi(s, a)\phi(s, a)^\top$
        $\Lambda \leftarrow \Lambda + \phi(s, a)\phi(s', a')^\top$
        $z \leftarrow z + \phi(s, a)r$
        $s \leftarrow s'$, $a \leftarrow a'$
        $t \leftarrow t + 1$
        si $((\mathrm{mod}(t, K) == 0)$ **entonces** (mejora de la política implícita)
            Calcular: $\theta = (\Gamma - \gamma\Lambda)^{-1} z$
    hasta que s sea terminal
hasta que no podamos correr más episodios
Calcular: $\theta = (\Gamma - \gamma\Lambda)^{-1} z$
para todo $s \in \mathcal{S}$ **hacer**
    $\pi(s) \leftarrow \arg\max_{a\in\mathcal{A}} \phi(s, a)^\top\theta$
devolver π

Capítulo 7

Optimización convexa

A continuación se presenta una de las herramientas que, junto a la teoría dual que se contará en el capítulo 8, será la base del desarrollo del nuevo algoritmo de aprendizaje por refuerzo propuesto en el capítulo 9.

7.1. Formulación del problema de optimización

7.1.1. Terminología básica

Se usará la notación:

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0, \quad i = 1, \ldots, m \\ & h_i(x) = 0, \quad i = 1, \ldots, p \end{array} \tag{7.1}$$

para describir el problema de encontrar un valor x que minimice $f_0(x)$ entre todos los x que satisfagan las condiciones $f_i(x) \leq 0$, $i = 1, \ldots, m$ y $h_i(x) = 0$, $i = 1, \ldots, p$. Se llamará a $x \in \mathbb{R}^n$ la *variable de optimización* y a la función $f_0 : \mathbb{R}^n \to \mathbb{R}$ la *función objetivo* o *función de coste*. Las desigualdades $f_i(x) \leq 0$ son llamadas *restricciones de desigualdad*, y las correspondientes funciones $f_i : \mathbb{R}^n \to \mathbb{R}$, *funciones de restricción de desigualdad*. Las ecuaciones $h_i(x) = 0$ reciben el nombre de *restricciones de igualdad*, y las funciones $h_i : \mathbb{R}^n \to \mathbb{R}$ son las *funciones de restricción de igualdad*.

El conjunto de puntos para el cual la función objetivo y las funciones de restricciones están definidas:

$$\mathcal{D} = \bigcap_{i=0}^{m} \mathbf{dom} f_i \cap \bigcap_{i=0}^{p} \mathbf{dom} h_i$$

se conoce como *dominio* del problema de optimización (7.1). Un punto $x \in \mathcal{D}$ es *factible* o una *solución factible* si satisface las restricciones $f_i(x) \leq 0$, $i = 1, \ldots, m$ y $h_i(x) =$

0, $i = 1, \ldots, p$. El problema (7.1) se dice que es factible si existe al menos un punto factible. Al conjunto de todos los puntos factibles se le llama *conjunto factible* o *conjunto de restricción* y se denota por $\mathcal{F}$.

El *valor óptimo* $p^\star$del problema (7.1) se define como:

$$p^\star = \text{ínf}\,\{f_0(x) \mid f_i(x) \leq 0,\, i = 1, \ldots, m,\; h_i(x) = 0,\, i = 1, \ldots, p\}$$

y se permite que tome valores de $\pm\infty$. Si el problema es no factible, tendremos $p^\star = \infty$ (siguiendo la convención estándar de que el ínfimo de un conjunto vacío es ∞). Si $p^\star = -\infty$ se dice que el problema (7.1) es *no acotado por abajo*.

7.1.2. Forma estándar

Se dice que el problema de optimización (7.1) es un problema en *forma estándar*. En la formulación del problema estándar se adopta la convención de que el lado derecho de las restricciones de igualdad y desigualdad es cero.

Problemas equivalentes

Se dice que dos problemas son equivalentes si mediante la resolución de uno, tenemos también la solución del otro, y viceversa [2]. Como un ejemplo simple, consideremos el problema:

$$\begin{array}{rl} \text{minimize} & \widetilde{f}(x) = \alpha_0 f_0(x) \\ \text{subject to} & \widetilde{f}_i(x) = \alpha_i f_i(x) \leq 0, \quad i = 1, \ldots, m \\ & \widetilde{h}_i(x) = \beta_i h_i(x) = 0, \quad i = 1, \ldots, p \end{array} \tag{7.2}$$

donde $\alpha_i > 0$, $i = 0, \ldots, m$ y $\beta_i \neq 0$, $i = 0, \ldots, p$. Este problema es obtenido a partir del problema (7.1) mediante un escalado de todas las funciones, siendo consecuentes con el signo de cada una y el del factor de escala. Como consecuencia, el conjunto factible del problema (7.2) y del problema original (7.1) son idénticos. Un punto es óptimo en el problema (7.1) sí y solo sí es óptimo en el problema escalado (7.2), de manera que los dos problemas son equivalentes. Sin embargo no son el mismo problema ya que las funciones objetivo y de restricción son diferentes.

7.2. Optimización convexa

Un *problema de optimización convexa* es uno de la forma:

$$\begin{aligned} \text{minimize} \quad & f_0(x) \\ \text{subject to} \quad & f_i(x) \leq 0, \quad i = 1, \ldots, m \\ & a_i^T x = b_i, \quad i = 1, \ldots, p \end{aligned} \tag{7.3}$$

donde $f_0, \ldots, f_m$ son funciones convexas. Comparando (7.3) con el problema general en forma estándar (7.1), se aprecia que el problema convexo tiene tres requisitos adicionales:

- la función objetivo debe ser convexa,
- las funciones de restricción de desigualdad deben ser convexas,
- las funciones de restricción de igualdad $h_i(x) = a_i^T x - b_i$ deben ser afines.

7.2.1. Problemas de optimización lineales

Cuando la función objetivo y las restricciones son todas afines, el problema se conoce como *programa lineal* (*linear program*, LP). Un programa lineal general tiene la siguiente forma:

$$\begin{aligned} \text{minimize} \quad & c^T x + d \\ \text{subject to} \quad & Gx \leq h \\ & Ax = b \end{aligned} \tag{7.4}$$

donde $G \in \mathbb{R}^{m \times n}$ y $A \in \mathbb{R}^{p \times n}$. Los programas lineales son, por tanto, problemas de optimización convexos. La interpretación geométrica de los programas lineales aparece ilustrada en la figura 7.1. El conjunto factible del LP (7.4) es un poliedro $\mathcal{P}$; el problema consiste en minimizar la función afín $c^T x + d$ (o de manera equivalente, la función $c^T x$) sobre $\mathcal{P}$.

Para este tipo de problemas, se define el concepto de solución básica [6] de la siguiente manera.

Definición 7.1. Considérese un poliedro $\mathcal{P}$ definido por unas restricciones lineales de igualdad y desigualdad, y asumamos que $x^\star$ es un elemento perteneciente a $\mathbb{R}^n$.

1. El vector $x^\star$ es una *solución básica* si:
   a) Todas las restricciones de igualdad están activas;
   b) De entre las restricciones que estén activas en $x^\star$, hay n de ellas que son linealmente independientes.
2. Si $x^\star$ es una solución básica que satisface todas las restricciones, decimos que se trata de una *solución básica factible*.

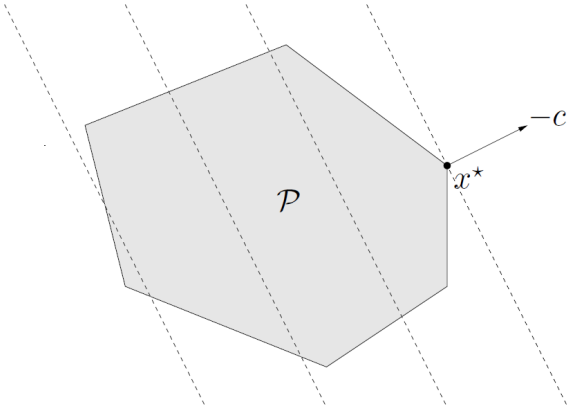


Figura 7.1: Interpretación geométrica de un LP. El conjunto factible $\mathcal{P}$, el cual es un poliedro, aparece sombreado. El punto $x^\star$ es óptimo: es el punto en $\mathcal{P}$ más lejano en la dirección $-c$ [2].

De la definición 7.1 (2) puede deducirse que una solución básica factible de un programa lineal no deberá poder expresarse como la combinación convexa de cualquier otra solución factible del programa lineal.

Una propiedad de las soluciones básicas factibles que será de gran importancia, pero que no se va a demostrar en este documento, es que cuando el programa lineal tiene m restricciones de igualdad, cualquier solución básica factible podrá tener como mucho m componentes positivas. Como veremos, en el capítulo 9 esta propiedad tomará especial relevancia.

Programación lineal en forma de desigualdad

Se define el *programa lineal en forma de desigualdad* como un LP que carece de restricciones de igualdad, típicamente escrito como:

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & Ax \leq b \end{aligned} \tag{7.5}$$

7.3. Optimización vectorial

Se define un *problema de optimización vectorial* de manera general como:

$$\begin{aligned} \text{minimize (con respecto al cono } K) \quad & f_0(x) \\ \text{subject to} \quad & f_i(x) \leq 0, \quad i = 1, \ldots, m \\ & h_i(x) = 0, \quad i = 1, \ldots, p \end{aligned} \tag{7.6}$$

Aquí $x \in \mathbb{R}^n$ es la variable de optimización, $K \subseteq \mathbb{R}^q$ es un cono convexo, $f_0 : \mathbb{R}^n \to \mathbb{R}^q$ es la función objetivo, $f_i : \mathbb{R}^n \to \mathbb{R}$ son las funciones de restricción de desigualdad, y $h_i : \mathbb{R}^n \to \mathbb{R}$ son las funciones de restricción de igualdad. La única diferencia entre este problema y el problema de optimización estándar (7.1) es que ahora, la función objetivo toma valores de $\mathbb{R}^q$, y las especificaciones del problema incluyen el cono convexo K, que se emplea para comparar valores. En el contexto de optimización vectorial, al problema de optimización estándar (7.1) a veces se le llama *problema de optimización escalar*.

7.3.1. Puntos óptimos

Considérese el conjunto de valores objetivo de los puntos factibles:

$$\mathcal{O} = \{f_0(x) \mid \exists x \in \mathcal{D},\ f_i(x) \leq 0,\ i = 1, \ldots, m,\ h_i(x) = 0,\ i = 1, \ldots, p\} \subseteq \mathbb{R}^q$$

al cual se le conoce como *valores objetivo alcanzables*. Si este conjunto tiene un elemento mínimo, es decir, existe un punto factible x tal que $f_0(x) \leq_K f_0(y)$ para todo y factible, entonces x es óptimo para el problema (7.6), y nos referimos a $f_0(x)$ como el *valor óptimo* del problema.

De manera más formal, se define que un punto $x^\star$ es optimo sí y solo sí es factible y además:

$$\mathcal{O} \subseteq f_0(x^\star) + K$$

7.3.2. Puntos óptimos de Pareto

Cuando el conjunto de valores objetivo alcanzables no tiene un elemento mínimo, el problema no tiene un punto óptimo.Para estos casos se define el concepto de elementos *minimales* del conjunto de valores alcanzables, los cuales jugarán un rol muy importante. Se dirá que un punto x factible es *Pareto óptimo* (o *eficiente*) si $f_0(x)$ es un elemento minimal del conjunto de valores alcanzables $\mathcal{O}$. En consecuencia, $f_0(x)$ es un *valor óptimo de Pareto* para el problema de optimización (7.6). De este modo, un punto x es Pareto óptimo si es factible y, para todo y factible, $f_0(y) \leq_K f_0(x)$ implica $f_0(y) = f_0(x)$. En otras palabras: cualquier punto factible y que es mejor o igual que x (es decir $f_0(y) \leq_K f_0(x)$) tiene exactamente el mismo valor objetivo que x.

De manera formal, un punto x es Pareto óptimo si y solo sí es factible y además:

$$(f_0(x) - K) \cap \mathcal{O} = \{f_0(x)\}$$

Un problema de optimización vectorial puede tener varios valores óptimos de Pareto

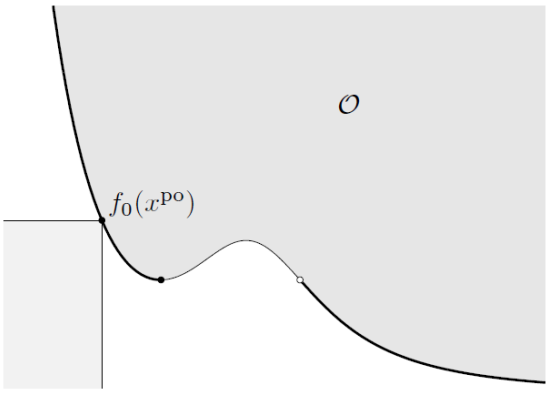


Figura 7.2: El conjunto de valores alcanzables $\mathcal{O}$ para un problema de optimización vectorial con valores objetivo en $\mathbb{R}^2$, con cono $K = \mathbb{R}^2_{++}$, se muestra sombreado. Este problema no tiene un punto o valor óptimo, pero sí tiene un conjunto de puntos óptimos de Pareto, cuyos valores correspondientes aparecen marcados en la curva negra en la parte baja de la izquierda de la frontera de $\mathcal{O}$. El punto marcado como $f_0(x^{po})$ es un valor óptimo de Pareto, y x^{po} es un punto óptimo de Pareto. La región marcada más suave corresponde a $f_0(x^{po}) - K$, y compone el conjunto de todos los puntos $z \in \mathbb{R}^2$ correspondientes a valores objetivos mejores que (o iguales a) $f_0(x^{po})$ [2].

(y varios puntos). El conjunto de valores óptimos de Pareto, denotado por P, satisface:

$$P \subseteq \mathcal{O} \cap \mathbf{bd}\mathcal{O}$$

siendo **bd** la frontera o *boundary*. Es decir, todo valor óptimo de Pareto es un valor objetivo alcanzable que cae en la frontera del conjunto de valores objetivos alcanzables. Todos estos conceptos se ilustran en la figura 7.2.

7.3.3. Escalarización

La escalarización es una técnica estándar para encontrar puntos Pareto óptimos en problemas de optimización vectorial. Supóngase un vector λ tal que $\lambda >_{K^*} 0$, es decir, un vector que sea positivo en la desigualdad dual generalizada. Ahora consideremos el problema de optimización escalar:

$$\begin{array}{ll} \text{minimize} & \lambda^T f_0(x) \\ \text{subject to} & f_i(x) \leq 0, \quad i = 1, \ldots, m \\ & h_i(x) = 0, \quad i = 1, \ldots, p \end{array} \tag{7.7}$$

y definamos x como un punto óptimo. Entonces x es Pareto optimo para el problema de optimización vectorial (7.6)[2].

Mediante la técnica de escalarización, podemos encontrar puntos Pareto óptimo para cualquier problema de optimización vectorial mediante la resolución de un problema de optimización escalar ordinario (7.7). La única condición es que el vector λ, a veces llamado vector de pesos, debe satisfacer $\lambda >_{K^*} 0$.

7.3.4. Optimización multiobjetivo

Cuando el problema de optimización vectorial involucra al cono $K = \mathbb{R}^q_+$, se le denomina problema de optimización *multicriterio* o *multiobjetivo*. Las componentes de f_0, digamos $F_1, \ldots, F_q$, pueden ser interpretadas como q objetivos escalares diferentes, cada uno de los cuales querremos minimizar. Nos referiremos a F_1 como el objetivo i-ésimo del problema. Un problema de optimización multiobjetivo es convexo si $f_1, \ldots, f_q$ son convexos, $h_1, \ldots, h_p$ son afines, y los objetivos $F_1, \ldots, F_q$ son convexos.

En un problema multiobjetivo, un punto óptimo $x^\star$ satisface:

$$F_i(x^\star) \leq F_i(y), \qquad i = 1, \ldots, q$$

para todo y factible. En otras palabras, $x^\star$ es simultáneamente óptimo para cada uno de los problemas escalares:

$$\begin{array}{rll} \text{minimize} & F_j(x) & \\ \text{subject to} & f_i(x) \leq 0, & i = 1, \ldots, m \\ & h_i(x) = 0, & i = 1, \ldots, p \end{array} \tag{7.8}$$

para $j = 1, \ldots, q$. Cuando únicamente hay un punto óptimo, se dice que los objetivos son *no competitivos*, pues no se deberá llegar a ningún compromiso entre los objetivos; cada objetivo es tan pequeño como podría ser, incluso si los demás fuesen ignorados.

Un punto óptimo de Pareto x^{po} satisface lo siguiente: si y es factible y $F_i(y) \leq F_i(x^{po})$ para $i = 1, \ldots, q$, entonces $F_i(x^{po}) = F_i(y)$, $i = 1, \ldots, q$. Esto se puede reformular como: un punto es Pareto óptimo sí y solo sí es factible y no hay otro punto factible mejor.

Capítulo 8

Dualidad

8.1. Función dual de Lagrange

8.1.1. Lagrangiano

Un problema de optimización en forma estándar viene dado por:

$$\begin{array}{rll} \text{minimize} & f_0(x) & \\ \text{subject to} & f_i(x) \leq 0, & i = 1, \ldots, m \\ & h_i(x) = 0, & i = 1, \ldots, p \end{array} \tag{8.1}$$

con variable $x \in \mathbb{R}^n$. Denotemos por $p^\star$ el valor óptimo. Como punto de partida, no se va a asumir que el problema (8.1) sea convexo.

La idea básica de la dualidad de Lagrange es tener en cuenta las restricciones de (8.1) extendiendo la función objetivo con la suma ponderada de las funciones de restricción. Se definirá el *Lagrangiano* $\mathcal{L} : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ asociado al problema (8.1) como:

$$\mathcal{L}(x, \lambda, \nu) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x)$$

con $\mathbf{dom}\, \mathcal{L} = \mathcal{D} \times \mathbb{R}^m \times \mathbb{R}^p$. Los pesos λ_i recibirán el nombre de *multiplicadores de Lagrange* asociados con la i-ésima restricción de desigualdad $f_i(x) \leq 0$; de manera similar, los pesos ν_i serán los multiplicadores de Lagrange asociados a la i-ésima restricción de igualdad $h_i(x) = 0$. Los vectores λ y ν son llamados en la literatura como *variables duales* o *vectores de multiplicadores de Lagrange* asociados con el problema (8.1).

8.1.2. Función dual de Lagrange

Se define la *función dual de Lagrange* (o simplemente *función dual*) $g : \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ como el mínimo valor del Lagrangiano sobre x: para $\lambda \in \mathbb{R}^m$, $\nu \in \mathbb{R}^p$,

$$g(\lambda, \nu) = \inf_{x \in \mathcal{D}} \mathcal{L}(x, \lambda, \nu) = \inf_{x \in \mathcal{D}} \left(f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x) \right) \tag{8.2}$$

Cuando el Lagrangiano no está acotado por abajo en x, la función dual toma valor $-\infty$. Dado que la función dual consiste en el ínfimo a trozos de una familia de funciones afines en (λ, ν), es cóncava, incluso cuando el problema (8.1) es no convexo.

Una propiedad muy importante de la función dual [2] es que será una cota inferior del valor óptimo $p^\star$; es decir, para cualquier $\lambda \geq 0$ y cualquier ν se tiene:

$$g(\lambda, \nu) \leq p^\star$$

8.2. Problema dual de Lagrange

Para cada par (λ, ν) con $\lambda \geq 0$, la función dual de Lagrange proporciona una cota inferior del valor óptimo $p^\star$ del problema de optimización (8.1). Por tanto, el problema ahora será encontrar la mejor cota inferior que se puede obtener a partir del Lagrangiano. Esta idea da pie al siguiente problema de optimización:

$$\begin{aligned} \text{maximize} \quad & g(\lambda, \nu) \\ \text{subject to} \quad & \lambda \geq 0 \end{aligned} \tag{8.3}$$

El problema (8.3) se conoce como el *problema dual de Lagrange* asociado al problema (8.1). En este contexto, el problema original (8.1) a veces es llamado *problema primal*. Los vectores $(\lambda^\star, \nu^\star)$ se denominarán *óptimos duales* o *multiplicadores de Lagrange óptimos* si son óptimos para el problema (8.3).

Puesto que el objetivo a maximizar ahora es cóncavo y la restricción es convexa, el problema dual de Lagrange (8.3) será un problema de optimización convexa. Este será el caso tanto si el problema primal (8.1) es convexo como si no.

8.2.1. Dualidad débil y fuerte

El valor óptimo del problema dual de Lagrange, al que denotaremos $d^\star$, es, por definición, la mejor cota inferior de $p^\star$ que se podrá obtener a partir de la función dual de Lagrange. Más concretamente, tendremos esta simple pero importante desigualdad:

$$d^\star \leq p^\star$$

que se cumplirá siempre incluso si el problema original es no convexo. Esta propiedad se conoce como *dualidad débil*. La diferencia $p^\star - d^\star$ es conocida como brecha de dualidad del problema original, pues determina la brecha entre el valor óptimo del problema primal y la mejor cota inferior de ella que se podrá obtener a partir de la función dual de Lagrange.

Si se cumple la igualdad:

$$d^\star = p^\star$$

es decir, si la brecha de dualidad es cero, entonces diremos que se existe dualidad fuerte. Esto significa que la mejor cota que podrá ser obtenida a partir de la función dual de Lagrange es igual al valor óptimo del problema primal.

8.2.2. Cualificación de las restricciones de Slater

La propiedad de dualidad fuerte no siempre se cumple. No obstante, si el problema primal es convexo, casi siempre existirá dualidad fuerte.

Se pueden encontrar una serie de condiciones sobre el problema, más allá de la convexidad, bajo las cuales existirá dualidad fuerte. Estas condiciones son conocidas como *cualificaciones de las restricciones*. Entre ellas, una de las más conocidas es la *restricción de Slater*, que dice lo siguiente: dado un problema primal convexo como el mostrado en (7.3), debe existir un punto $x \in \mathbf{relint}\,\mathcal{D}$, siendo **relint** el interior relativo o *relative interior*, tal que:

$$f_i(x) < 0, \qquad i = 1, \ldots, m, \qquad Ax = b \tag{8.4}$$

El teorema de Slater establece, por tanto, que existirá dualidad fuerte si el problema es convexo y se cumple la restricción de Slater (8.4).

La restricción de Slater se puede refinar un poco más cuando las funciones de restricción de desigualdad son afines. Si las primeras k funciones de restricción son afines, entonces existe dualidad fuerte suponiendo que la siguiente condición débil se cumpla: existe un punto $x \in \mathbf{relint}\,\mathcal{D}$ tal que:

$$f_i(x) \leq 0, \qquad i = 1, \ldots, k, \qquad f_i(x) < 0, \qquad i = k+1, \ldots, m, \qquad Ax = b \tag{8.5}$$

En otras palabras, las desigualdades afines no tienen por qué satisfacerse con estricta desigualdad.

La restricción de Slater (y su versión refinada (8.5)) no solo implica dualidad fuerte en problemas convexos. Además implica que el valor óptimo dual es obtenido cuando $d^\star > -\infty$; es decir, existe un dual factible $(\lambda^\star, \nu^\star)$ con $g(\lambda^\star, \nu^\star) = d^\star = p^\star$.

8.3. Dualidad de Lagrange como un punto de silla

8.3.1. Caracterización max-min de la dualidad débil y fuerte

Es posible expresar los problemas de optimización primal y dual de una forma más simétrica. Para comenzar con este enfoque, y suponiendo únicamente la existencia de restricciones de desigualdad, se partirá de:

$$\begin{aligned}\sup_{\lambda\geq 0}\mathcal{L}(x,\lambda) &= \sup_{\lambda\geq 0}\left(f_0(x)+\sum_{i=1}^{m}\lambda_i f_i(x)\right)\\ &= \begin{cases} f_0(x) & f_i(x)\leq 0\\ \infty & \text{resto}\end{cases}\end{aligned}$$

Esto significa que se puede expresar el valor óptimo del problema primal como:

$$p^\star = \inf_x \sup_{\lambda\geq 0}\mathcal{L}(x,\lambda) \tag{8.6}$$

Por la definición de la función dual, además se tiene:

$$d^\star = \sup_{\lambda\geq 0}\inf_x \mathcal{L}(x,\lambda) \tag{8.7}$$

La condición de dualidad débil podrá ser expresada por tanto como la siguiente desigualdad:

$$\sup_{\lambda\geq 0}\inf_x \mathcal{L}(x,\lambda) \leq \inf_x \sup_{\lambda\geq 0}\mathcal{L}(x,\lambda) \tag{8.8}$$

y la dualidad fuerte como la igualdad:

$$\sup_{\lambda\geq 0}\inf_x \mathcal{L}(x,\lambda) = \inf_x \sup_{\lambda\geq 0}\mathcal{L}(x,\lambda)$$

La desigualdad (8.8) no depende de ninguna propiedad de $\mathcal{L}$. De manera general se tiene:

$$\sup_{z\in Z}\inf_{\omega\in W} f(w,z) \leq \inf_{\omega\in W}\sup_{z\in Z} f(w,z) \tag{8.9}$$

para cualquier $f : \mathbb{R}^n\times\mathbb{R}^m \to \mathbb{R}$. Esta desigualdad general recibe el nombre de *desigualdad máx-mín*. Cuando se cumple la igualdad, es decir:

$$\sup_{z\in Z}\inf_{\omega\in W} f(w,z) = \inf_{\omega\in W}\sup_{z\in Z} f(w,z) \tag{8.10}$$

se dice que f (y W y Z) satisface la *propiedad máx-mín fuerte* o propiedad del *punto de silla*. Como cabía esperar, la propiedad máx-mín fuerte únicamente se satisface en

determinados casos. Uno de ellos es cuando $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ es el Lagrangiano de un problema para el cual existe dualidad fuerte.

8.3.2. Interpretación del punto de silla

Retomando la formulación del problema máx-mín para el Lagrangiano (8.8), se puede apreciar que si $x^\star$ y $\lambda^\star$ son puntos primal y dual óptimos para un problema en el cual existe dualidad fuerte, entonces forman un punto de silla para el Lagrangiano. La relación contraria es también cierta: si (x, λ) son un punto de silla del Lagrangiano, entonces x es el primal óptimo, λ es el dual óptimo, y la brecha de dualidad es cero.

8.3.3. Métodos de búsqueda del punto de silla

Como puede deducirse de lo anterior, si el problema primal cumple la condición de dualidad fuerte, resolver el problema de punto de silla del Lagrangiano significará encontrar la solución óptima de los problemas primal y dual. Por tanto, cualquier método que permita encontrar el punto de silla del Lagrangiano permitirá resolver un problema de optimización con restricciones, en el cual exista dualidad fuerte. A continuación se presentan tres posibles métodos para encontrar el punto de silla, en los cuales se va a suponer siempre que el problema primal cumple la propiedad de dualidad fuerte.

Arrow-Hurwicz

De acuerdo a (8.6) y (8.7), el problema a resolver se puede descomponer en dos problemas, uno de minimización y otro de maximización, que están fuertemente acoplados. Lo que Arrow y Hurwicz proponen en [7] es alternan una etapa de descenso por gradiente en la variable primal con otra de ascenso por gradiente en la variable dual. De nuevo, suponiendo únicamente restricciones de desigualdad se tiene:

$$\begin{aligned} x_{t+1} &\coloneqq x_t - \alpha_x \nabla_x \mathcal{L}(x_t, \lambda_t) \\ \lambda_{t+1} &\coloneqq [\lambda_t + \alpha_\lambda \nabla_\lambda \mathcal{L}(x_{t+1}, \lambda_t)]_+ \end{aligned}$$

donde α es el tamaño del paso o tasa de aprendizaje y se define $[\quad]_+$ como la proyección[1] $u_+ = \max\{\lambda, 0\}$, para cada componente del vector λ, sobre el conjunto de números reales no negativos $\mathbb{R}_+$. De este modo, repitiendo el proceso de manera iterativa se conseguirá converger al punto de silla.

Tal y como se detalla en [7], cuando el Lagrangiano es lineal tanto en la variable primal como en la dual, este método no converge y se mantiene en un estado oscilante,

[1] Esta proyección garantiza la no negatividad de los multiplicadores de Lagrange de las restricciones de desigualdad. Cuando se aplique el método Arrow-Hurwicz con restricciones de igualdad, no será necesario proyectar.

con lo cual habría que buscar una alternativa. A continuación se propone como solución el método del Lagrangiano aumentado.

Lagrangiano aumentado

Suponiendo un problema primal convexo como el mostrado en (7.4) pero únicamente con restricciones de desigualdad, se define en [8] el *Lagrangiano aumentado* como:

$$\mathcal{L}_\rho(x, \lambda) = f_0(x) + \lambda^T(Gx - h) + \frac{\rho}{2} \|Gx - h\|_2^2$$

donde$\rho > 0$ es el parámetro de penalización. De esta manera se soluciona el problema de la linealidad mencionado por Arrow y Hurwicz y se puede emplear dicho método de nuevo.

Ascenso dual

Por último, se presenta otra alternativa al método Arrow-Hurwicz o a su versión basada en el Lagrangiano aumentado.

Atendiendo a la definición del Lagrangiano (8.2) y a la formulación del problema dual (8.3), se podrá obtener un punto primal óptimo de la siguiente manera:

$$x^\star = \arg\min_x \mathcal{L}(x, \lambda^\star)$$

Aprovechando esta condición, el método de *ascenso dual* resuelve el problema del punto de silla de la siguiente manera [8]:

$$\begin{aligned} x_{t+1} &:= \arg\min_x \mathcal{L}(x, \lambda_t) \\ \lambda_{t+1} &:= [\lambda_t + \alpha_\lambda \nabla_\lambda \mathcal{L}(x_{t+1}, \lambda_t)]_+ \end{aligned}$$

es decir, para cada actualización de λ se busca de manera exacta el argumento que minimiza el Lagrangiano. Este proceso se repite iterativamente hasta alcanzar la convergencia, y por ende el punto de silla.

8.4. Condiciones de optimalidad de Karush-Kuhn-Tucker

A continuación se establecen una serie de condiciones que permitirán garantizar que el punto de silla encontrado corresponde a una solución óptima.

Supóngase $x^\star$ un punto primal óptimo, $(\lambda^\star, \nu^\star)$ puntos duales óptimos, y que la brecha de dualidad es cero. Puesto que $x^\star$ minimiza $\mathcal{L}(x, \lambda^\star, \nu^\star)$ sobre x, se deduce que su

gradiente debe anularse en $x^\star$, es decir:

$$\nabla f_0(x^\star) + \sum_{i=1}^{m} \lambda_i^\star \nabla f_i(x^\star) + \sum_{i=1}^{p} \nu_i^\star \nabla h_i(x^\star) = 0$$

Por tanto, deberán cumplirse las siguiente condiciones:

$$\begin{aligned} f_i(x^\star) &\leq 0, \qquad i = 1, \ldots, m \\ h_i(x^\star) &= 0, \qquad i = 1, \ldots, p \\ \lambda_i^\star &\geq 0, \qquad i = 1, \ldots, m \\ \lambda_i^\star f_i(x^\star) &= 0, \qquad i = 1, \ldots, m \\ \nabla f_0(x^\star) + \sum_{i=1}^{m} \lambda_i^\star \nabla f_i(x^\star) + \sum_{i=1}^{p} \nu_i^\star \nabla h_i(x^\star) &= 0 \end{aligned}$$

las cuales se conocen como condiciones de Karush-Kuhn-Tucker (KKT). En resumen, para todo problema de optimización que cumpla la condición de dualidad fuerte, cualquier par de puntos primal y dual óptimos deberán satisfacer las condiciones KKT anteriores.

Cuando el problema primal sea convexo, las condiciones KKT serán suficientes para que los puntos sean primal y dual óptimos. Además, si el problema de optimización convexo satisface la condición de Slater, las condiciones KKT serán condiciones necesarias y suficientes para garantizar la optimalidad [2].

Capítulo 9

Ecuaciones de Bellman y teoría dual

Tal y como se describió en el capítulo 3, el problema que se tratará de resolver será el de encontrar una política de comportamiento óptima que maximice la recompensa acumulada a lo largo de un horizonte de tiempo finito o infinito, empezando desde un estado inicial s_o cualquiera de nuestro problema. La búsqueda de dicha política se llevará a cabo a través de las funciones valor, funciones de los estados que estiman cómo de bueno es para el agente estar en un estado concreto.

En el capítulo 5 se abordó la resolución del problema en cuestión por medio de las ecuaciones de Bellman, para problemas de pequeña escala. Se derivó tanto la solución en forma cerrada como la solución basada en métodos recursivos tales como programación dinámica (para el caso en que se conoce el modelo del entorno) o TD (para el caso en que no se conoce el modelo del entorno).

En este capítulo se va a presentar un enfoque alternativo basado en la teoría dual, expuesta en el capítulo 8, con el que se logrará resolver el problema de control óptimo planteado de una forma poco estudiada hasta el momento: a través de la exploración del espacio de políticas. Para ello, se formulará inicialmente el problema primal de optimización que resuelve las ecuaciones óptimas de Bellman, a continuación su problema dual asociado y se derivarán importantes propiedades de la variable dual. Finalmente, se resolverá el problema dual haciendo uso del método de ascenso dual descrito en la sección 8.3.3, con ligeras modificaciones.

9.1. Formulación del problema primal

De cara a establecer una representación dual que permita resolver problemas de programación dinámica y de aprendizaje por refuerzo, será necesario reformular el problema

de programación dinámica enunciado en el capítulo 5 como un problema de optimización. Este problema de optimización será el que denominemos problema primal, y permitirá resolver las ecuaciones óptimas de Bellman. Para comenzar con el desarrollo que nos permita llegar a la formulación del problema primal, resultará de utilidad recordar algunas propiedades y conceptos básicos en lo referido a programación dinámica y procesos de decisión de Markov.

9.1.1. De programación dinámica al problema de optimización multiobjetivo

Según el teorema 5.2 y de acuerdo a la definición del operador de Bellman óptimo (5.2), T, se sabe que para cualquier función valor v deberá cumplirse:

$$v \leq Tv \leq T^2v \leq \cdots \leq T^kv \leq \lim_{N\to\infty} T^Nv = v^* = Tv^* \tag{9.1}$$

es decir, existe una única solución v^* a la ecuación de punto fijo $Tv = v$.

Supongamos ahora la existencia de un vector v tal que:

$$v \geq Tv \tag{9.2}$$

Haciendo un desarrollo similar al mostrado en (9.1) se llega al siguiente resultado:

$$v \geq \lim_{N\to\infty} T^Nv \geq v^* = Tv^* \tag{9.3}$$

de donde se deduce que el vector v más pequeño posible que satisfaría la condición (9.2) sería v^*.

De acuerdo a esta conclusión, parece claro que podríamos expresar nuestro problema de búsqueda de la función valor óptima como un problema de optimización cuyo objetivo sería encontrar el mínimo vector v que satisficiera la condición (9.2). Expresado de manera formal, el problema que se querrá resolver será:

$$\begin{aligned} \underset{v}{\text{minimize}} \quad & v \\ \text{subject to} \quad & v \geq Tv \end{aligned} \tag{9.4}$$

donde Tv define la siguiente familia de funciones:

$$(Tv)(s) \triangleq \max_{a\in\mathcal{A}} \left[\mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{ss'}^a v\left(s'\right) \right], \qquad \forall s \in \mathcal{S} \tag{9.5}$$

Se puede observar que la maximización se realiza sobre un conjunto de funciones convexas, más concretamente afines, y es bien sabido que el máximo sobre un conjunto

de funciones afines es también una función afín [2]. Por tanto, se puede afirmar que Tv define una familia de funciones afines a trozos.

Si se expresa el problema (9.4) en forma escalar, se llega a que para cada componente del vector v –o lo que es lo mismo, para cada estado del MDP que define nuestro entorno–, el problema a resolver es el siguiente:

$$\begin{aligned} \underset{v(s)}{\text{minimize}} \quad & v(s) \\ \text{subject to} \quad & v(s) \geq \max_{a \in \mathcal{A}} \left[\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v\left(s'\right) \right] \end{aligned} \tag{9.6}$$

para todo $s \in \mathcal{S}$, que como puede inferirse, está expresado como un problema en forma de epigrafo (ver Anexo A). Teniendo el problema de optimización en este formato, transformarlo al programa lineal equivalente consistirá simplemente en expresar la restricción de desigualdad como un conjunto de $|\mathcal{S}||\mathcal{A}|$ desigualdades diferentes, es decir:

$$\begin{aligned} \underset{v(s)}{\text{minimize}} \quad & v(s) \\ \text{subject to} \quad & v(s) \geq \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v\left(s'\right), \qquad \forall a \in \mathcal{A} \end{aligned} \tag{9.7}$$

para todo $s \in \mathcal{S}$.

Este tipo de problemas de optimización en los que tanto la función objetivo como las restricciones son lineales y únicamente existen restricciones de desigualdad, se conocen como programas lineales en forma de desigualdad.

Con este desarrollo se consigue hacer desaparecer el operador «máx» de las condiciones de desigualdad, el cual es no lineal, para transformar nuestro problema de optimización en un problema de optimización lineal. Si se deshacen estos cambios hasta volver de nuevo al problema original expresado en forma vectorial, se tiene que el problema (9.4) se puede reformular de la siguiente manera:

$$\begin{aligned} \underset{v}{\text{minimize}} \quad & v \\ \text{subject to} \quad & \Xi^T v \geq \mathcal{R} + \gamma \mathcal{P} v \end{aligned} \tag{9.8}$$

donde Ξ es una matriz de dimensión $|\mathcal{S}| \times |\mathcal{S}|\,|\mathcal{A}|$ formada por $\mathcal{S}$ bloques fila de 1s, cada una de dimensión $|\mathcal{A}|$, dispuestos en diagonal:

$$\Xi = \begin{pmatrix} 1 \cdots 1 & & & \\ & 1 \cdots 1 & & \\ & & \ddots & \\ & & & 1 \cdots 1 \end{pmatrix}$$

Como se puede observar, el problema (9.8) es un problema de optimización multiobjetivo convexo, pues se quiere optimizar el vector v, compuesto de $|\mathcal{S}|$ objetivos escalares afines diferentes, sujeto a $|\mathcal{S}|\,|\mathcal{A}|$ restricciones de desigualdad también afines. Además, de acuerdo a (9.3) y al teorema 5.2 del punto fijo de Banach, se puede garantizar que la superficie de puntos óptimos de Pareto de este problema multiobjetivo estará formada por un único punto: v^*.

Con ánimo de facilitar la comprensión del desarrollo anterior, se presenta a continuación una interpretación gráfica del nuevo problema obtenido.

Interpretación gráfica

Las restricciones impuestas en nuestro problema de optimización constituyen un sistema de desigualdades lineales y finito, y por tanto definen un poliedro en $\mathbb{R}^{|\mathcal{S}|}$.

$$v(s) \geq \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v\left(s'\right), \qquad \forall a \in \mathcal{A}, \forall s \in \mathcal{S} \tag{9.9}$$

Para ejemplificar esta idea, supóngase un MDP con 2 posibles estados, y en cada estado 2 posibles acciones; es decir, $|\mathcal{S}| = 2$ y $|\mathcal{A}| = 2$. Una interpretación del poliedro definido por las restricciones (9.9) para este problema podría ser la mostrada en la figura 9.1.

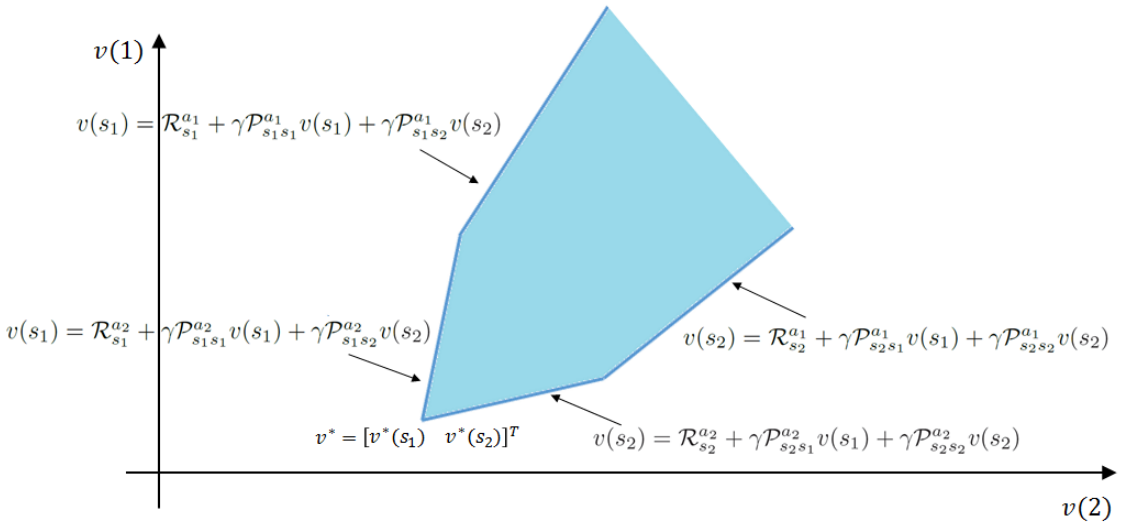


Figura 9.1: Interpretación gráfica del problema de optimización multiobjetivo que permite resolver las ecuaciones óptimas de Bellman para el MDP de 2 estados y 2 acciones planteado. Sombreado en azul se encuentra el conjunto de puntos que cumplen las restricciones de desigualdad impuestas por el problema.

Tal y como se puede observar, la función valor óptima v^* será la esquina «sudoeste» de este poliedro; es decir, el mínimo vector v para el que se cumplirán las restricciones

de desigualdad en cada problema individual (9.7), y por tanto el punto óptimo de Pareto de nuestro problema multiobjetivo.

9.1.2. Problema primal expresado como un programa lineal

Conocido ya el problema multiobjetivo a resolver, se sabe que aplicando la técnica de escalarización presentada en 7.3.3 se podrá encontrar el punto óptimo de Pareto v^*, solución de las ecuaciones óptimas de Bellman. De este modo, y partiendo del problema (9.7), la nueva función objetivo a minimizar será la suma ponderada de las funciones objetivo de cada problema individual:

$$\sum_{s\in\mathcal{S}} \lambda(s)v(s) = \lambda^T v$$

para cualquier $\lambda > 0$, con lo cual, el problema de optimización multiobjetivo convexo (9.7) se podrá reformular como el siguiente programa lineal en forma de desigualdad:

$$\begin{aligned} \underset{v(s)}{\text{minimize}} \quad & \sum_{s\in\mathcal{S}} \lambda(s)v(s) \\ \text{subject to} \quad & v(s) \geq \mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{ss'}^a v\left(s'\right) \end{aligned} \tag{9.10}$$

para cualquier $\lambda(s) > 0$, para todo $s \in \mathcal{S}$ y todo $a \in \mathcal{A}$. Expresado en forma vectorial:

$$\begin{aligned} \underset{v}{\text{minimize}} \quad & \lambda^T v \\ \text{subject to} \quad & \Xi^T v \geq \mathcal{R} + \gamma\mathcal{P}v \end{aligned} \tag{9.11}$$

para cualquier $\lambda > 0$.

El programa lineal (9.11) será por tanto el problema primal que resuelva las ecuaciones óptimas de Bellman.

9.2. Derivación del problema dual

Siguiendo el enfoque dual propuesto al comienzo del capítulo, a continuación se presenta la derivación del problema dual asociado al problema primal (9.11). Como veremos al final de esta sección, será esta representación dual de la que se deduzcan importantes propiedades que darán pie al desarrollo de nuevos algoritmos de aprendizaje por refuerzo.

Lagrangiano

Para llegar al problema dual, primero será necesario formular el Lagrangiano asociado al problema (9.11):

$$\begin{aligned}\mathcal{L}(v,d) &= \sum_{s\in\mathcal{S}}\lambda(s)v(s) + \sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} d(s,a)\left(\mathcal{R}_s^a + \gamma\sum_{s'\in\mathcal{S}}\mathcal{P}_{ss'}^a v\left(s'\right) - v(s)\right) \\ &= \lambda^T v + d^T\left(\mathcal{R} + \gamma\mathcal{P}v - \Xi^T v\right)\end{aligned} \tag{9.12}$$

donde d es el vector de dimensión $|\mathcal{S}|\,|\mathcal{A}| \times 1$ cuyas componentes son los multiplicadores de Lagrange asociados a cada una de las restricciones de desigualdad (9.9) del problema primal. De la formulación general para la función dual de Lagrange expuesta en 8.1, se sabe que además deberá cumplirse $d \geq 0$.

Función dual

A partir del Lagrangiano (9.12) se puede obtener la función dual asociada como:

$$\begin{aligned}g(d) &= \inf_{v\in\mathbb{R}^{|\mathcal{S}|}} \mathcal{L}(v,d) \\ &= \inf_{v\in\mathbb{R}^{|\mathcal{S}|}}\left(\sum_{s\in\mathcal{S}}\lambda(s)v(s) + \sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} d(s,a)\left(\mathcal{R}_s^a + \gamma\sum_{s'\in\mathcal{S}}\mathcal{P}_{ss'}^a v\left(s'\right) - v(s)\right)\right) \\ &= \sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} d(s,a)\mathcal{R}_s^a + \inf_{v\in\mathbb{R}^{|\mathcal{S}|}}\sum_{s'\in\mathcal{S}} v(s')\left(\lambda(s') + \gamma\sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} d(s,a)\mathcal{P}_{ss'}^a - \sum_{a\in\mathcal{A}} d(s',a)\right)\end{aligned}$$

sujeto a $\lambda(s) > 0$ y $d(s,a) \geq 0$, $\forall s \in \mathcal{S}$ y $\forall a \in \mathcal{A}$. Expresado en forma vectorial resulta en:

$$\begin{aligned}g(d) &= \inf_{v\in\mathbb{R}^{|\mathcal{S}|}} \mathcal{L}(v,d) \\ &= \inf_{v\in\mathbb{R}^{|\mathcal{S}|}}\left(\lambda^T v + d^T\left(\mathcal{R} + \gamma\mathcal{P}v - \Xi^T v\right)\right) \\ &= d^T\mathcal{R} + \inf_{v\in\mathbb{R}^{|\mathcal{S}|}}\left(\lambda^T + \gamma d^T\mathcal{P} - d^T\Xi^T\right)v, \qquad \forall\lambda > 0, d \geq 0\end{aligned} \tag{9.13}$$

Dicha función puede ser fácilmente obtenida analíticamente ya que el término del cual se busca el ínfimo, $\left(\lambda^T + \gamma d^T\mathcal{P} - d^T\Xi^T\right)v$, es lineal, y las funciones lineales únicamente están acotadas por abajo cuando son iguales a cero. Por tanto, $g(d) = -\infty$ excepto cuando $\lambda + \gamma\mathcal{P}^T d - \Xi d = 0$, en cuyo caso $g(d) = d^T\mathcal{R}$:

$$g(d) = \begin{cases} d^T\mathcal{R} & \lambda + \gamma\mathcal{P}^T d - \Xi d = 0 \\ -\infty & \text{en cualquier otro caso}\end{cases}, \qquad \forall\lambda > 0, d \geq 0 \tag{9.14}$$

Para facilitar la formulación final del problema dual, será apropiado asignarle un valor al vector λ. En [5] se sugiere escoger un vector λ compuesto por escalares positivos cuya suma total sea uno, es decir, el vector de escalarización deberá cumplir las siguientes

condiciones:

$$\lambda(s) > 0,\ \forall s \in \mathcal{S} \qquad\qquad \sum_{s\in\mathcal{S}} \lambda(s) = 1$$

Recalcar que cualquier vector λ cuyas componentes fueran positivas serviría, pero la condición añadida de que sumen 1 nos permite interpretar el vector de escalarización como una distribución de probabilidad de los estados. Atendiendo a esta interpretación y de acuerdo a [3], se escogerá como vector de escalarización $\mu = \begin{bmatrix}\mu(s_1) & \cdots & \mu(s_{|\mathcal{S}|})\end{bmatrix}$, de dimensión $|\mathcal{S}| \times 1$, el cual define la distribución inicial sobre los estados de la cadena de Markov que representa nuestro problema. Bajo estas condiciones, la función dual (9.14) resulta en:

$$g(d) = \begin{cases} \sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}} d(s,a)\mathcal{R}_s^a & \mu(s') + \gamma \sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}} d(s,a)\mathcal{P}_{ss'}^a - \sum_{a\in\mathcal{A}} d(s',a) = 0 \\ -\infty & \text{en cualquier otro caso} \end{cases} \tag{9.15}$$

para todo $s' \in \mathcal{S}$, sujeto a $d(s,a) \geq 0$, para todo $s \in \mathcal{S}$ y todo $a \in \mathcal{A}$. Expresado en forma vectorial se llega a:

$$g(d) = \begin{cases} d^T\mathcal{R} & \mu + \gamma\mathcal{P}^T d - \Xi d = 0 \\ -\infty & \text{en cualquier otro caso} \end{cases}, \qquad d \geq 0 \tag{9.16}$$

Problema dual

De manera estricta, el problema dual de Lagrange asociado al programa lineal (9.11) será maximizar la función dual (9.16) sujeta a $d \geq 0$,es decir:

$$\begin{aligned} \underset{d}{\text{maximize}} \quad & g(d) = \begin{cases} d^T\mathcal{R} & \mu + \gamma\mathcal{P}^T d - \Xi d = 0 \\ -\infty & \text{en cualquier otro caso} \end{cases} \\ \text{subject to} \quad & d \geq 0 \end{aligned} \tag{9.17}$$

Teniendo en cuenta el hecho de que $g(d)$ es finita solamente cuando $\mu + \gamma\mathcal{P}^T d - \Xi d = 0$, podemos formular un problema equivalente haciendo estas restricciones de igualdad explicitas:

$$\begin{aligned} \underset{d}{\text{maximize}} \quad & d^T\mathcal{R} \\ \text{subject to} \quad & \mu + \gamma\mathcal{P}^T d - \Xi d = 0 \\ & d \geq 0 \end{aligned} \tag{9.18}$$

Expresado en forma escalar se tiene:

$$\begin{aligned} \underset{d(s,a)}{\text{maximize}} \quad & \sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} d(s,a)\mathcal{R}_s^a \\ \text{subject to} \quad & \mu(s') + \gamma \sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} d(s,a)\mathcal{P}_{ss'}^a - \sum_{a\in\mathcal{A}} d(s',a) = 0 \\ & d(s,a) \geq 0 \end{aligned} \tag{9.19}$$

para todo $s' \in \mathcal{S}$, todo $s \in \mathcal{S}$ y toda $a \in \mathcal{A}$.

Se dirá por tanto que $d(s,a)$ es una solución factible del problema dual equivalente siempre que sea no negativo y además satisfaga la restricción de igualdad mostrada en (9.19).

De este modo se ha conseguido derivar el problema dual como un programa lineal en forma estándar.

9.2.1. Interpretación de la variable dual

La importancia de llegar a esta formulación dual radica en que, de acuerdo a [5], a partir de la variable dual d se va a ser capaz de extraer una política de comportamiento estacionaria π_d para el MDP que modela nuestro problema de control. Además, si el MDP en cuestión sigue esa política π_d, la solución al problema dual será un vector d^{π_d} perteneciente al conjunto de puntos factibles del problema.

Teorema 9.1. *Para cada $\pi \in \Pi^{MR}$, $s \in \mathcal{S}$ y $a \in \mathcal{A}$, se define $d^\pi(s,a)$ como:*

$$d^\pi(s,a) = \sum_{j\in S} \mu(j) \sum_{t=0}^{\infty} \gamma^t \mathbb{P}\left(S_t = s, A_t = a \mid S_o = j, \pi\right) \tag{9.20}$$

Entonces $d^\pi(s,a)$ es una solución factible del problema dual.

La demostración del teorema 9.1 aparece detallada en el Anexo B.

Teorema 9.2. *Supóngase que $d(s,a)$ es una solución factible del problema dual. En consecuencia $\sum_{a\in\mathcal{A}} d(s,a) > 0$ para todo $s \in \mathcal{S}$. Definamos la política estacionaria estocástica π_d como:*

$$\pi_d(a \mid s) = \mathbb{P}\left(\pi_d(s) = a\right) = \frac{d(s,a)}{\sum_{a'\in\mathcal{A}} d(s,a')} \tag{9.21}$$

Entonces $d^{\pi_d}(s,a)$, tal y como se definió en (9.20) es una solución factible del problema dual, y además $d^{\pi_d}(s,a) = d(s,a)$ para todo $a\in\mathcal{A}$, y $s \in \mathcal{S}$.

La demostración del teorema 9.2 aparece detallada en el Anexo C.

La variable dual $d(s,a)$, definida en (9.20), representa por tanto la probabilidad conjunta total descontada bajo la distribución inicial de estados μ de que el sistema se encuentre en el estado s y se tome la acción a. De este modo, cuando se multiplica por $\mathcal{R}_s^a$ y

se suma sobre todos los pares estado-acción (función objetivo del problema dual (9.19)), se obtiene la recompensa total descontada esperada, resultado de seguir la política π_d:

$$\sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} d(s,a)\mathcal{R}_s^a = \sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}}\sum_{j\in S}\mu(j)\sum_{t=0}^{\infty}\gamma^t\mathbb{P}\left(S_t=s, A_t=a \mid S_o=j,\pi_d\right)\mathcal{R}_s^a \tag{9.22}$$

A partir de esta interpretación se desprende la siguiente relación entre el problema primal y el dual:

Corolario 9.1. *De acuerdo a la definición de función valor y según lo establecido en los teoremas 9.1 y 9.2 se deduce que el problema primal y el dual se relacionan a través de su función objetivo de la siguiente manera:*

$$\sum_{j\in\mathcal{S}}\mu(j)v^{\pi_d}(j) = \sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} d(s,a)\mathcal{R}_s^a \tag{9.23}$$

y generalizando para una política $\pi\in\Pi$ cualquiera:

$$\sum_{j\in\mathcal{S}}\mu(j)v^{\pi}(j) = \sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} d^{\pi}(s,a)\mathcal{R}_s^a \tag{9.24}$$

La demostración del corolario 9.1 aparece detallada en el Anexo D.

Hasta el momento, gracias al teorema 9.1 se han podido vincular soluciones factibles con políticas estacionarias estocásticas. Del mismo modo, en el teorema 9.2 se ha mostrado la manera de generar políticas estocásticas a partir de una solución factible.

A continuación se van a hacer unas deducciones análogas, pero para el caso en que se tiene una solución básica factible. Las siguientes proposiciones serán de gran utilidad cuando se trate la búsqueda de la política óptima.

Proposición 9.1. *Supóngase que d es una solución básica factible del programa lineal dual. Entonces $\pi_d\in\Pi^{MD}$.*

La demostración de la proposición 9.1 aparece detallada en el Anexo E.

Proposición 9.2. *Supóngase que $\pi\in\Pi^{MD}$. Entonces d^{π} es una solución básica factible del programa lineal dual.*

La demostración de la proposición 9.2 aparece detallada en el Anexo F.

Solución óptima y política óptima

Sabiendo ya que vamos a poder extraer una política a partir de la variable dual d, parece lógico pensar que si obtenemos la solución óptima del problema dual, habremos conseguido también la política de comportamiento óptima de nuestro problema de control.

Además, en base a las proposiciones anteriores, si dicha solución óptima es una solución básica factible, se podrá asegurar que la política óptima es determinista. Para formalizar todas estas ideas, se enuncian a continuación los siguientes teoremas:

Teorema 9.3. *Asumiendo que la suposición 2 se cumple, existe una solución básica factible óptima acotada d^* para el programa lineal dual.*

La demostración del teorema 9.3 aparece detallada en el Anexo G.

Teorema 9.4. *Asumiendo que la suposición 2 se cumple, si d^* es una solución óptima del programa lineal dual, entonces π_{d^*} es una política óptima.*

La demostración del teorema 9.4 aparece detallada en el Anexo H.

Teorema 9.5. *Asumiendo que la suposición 2 se cumple, si d^* es una solución básica óptima del programa lineal dual, entonces π_{d^*} es una política óptima determinista.*

Demostración. La demostración de este teorema se deduce del teorema 9.4 y de la proposición 9.1. □

En resumen, encontrar la variable dual óptima d^* implicará por el argumento de dualidad fuerte encontrar también la variable primal óptima v^*. Según el teorema 9.2 se sabe además que siguiendo la política π_{d^*} extraída de la variable dual óptima se obtiene de nuevo la variable dual óptima, es decir, $d^*(s,a) = d^{\pi_{d^*}}(s,a)$. Uniendo todos estos conceptos se deduce que, una vez encontrado d^*, si se sigue la política π_{d^*} la variable dual seguirá siendo óptima, lo cual implica también encontrar la variable primal óptima, solución óptima de las ecuaciones de Bellman. Por pura definición, una política con la que se obtiene la solución óptima de las ecuaciones de Bellman es una política óptima. De este modo, se garantiza que π_{d^*} es la política óptima de nuestro problema. Si además se satisface la suposición 2, de acuerdo a los teoremas 9.3 y 9.5 se podrá afirmar también que la política óptima π_{d^*} es determinista.

A la vista de esta interpretación de la variable dual, lo que se querrá ahora será encontrar la variable dual óptima. Para ello habrá que resolver el problema dual.

9.3. Solución del problema dual

Como se ha podido ver, todo el desarrollo anterior tiene su origen en el método de programación dinámica. La programación dinámica permite resolver las ecuaciones de Bellman cuando se tiene conocimiento del modelo del entorno. Por ello, en las secciones anteriores de este capítulo se ha dado por supuesto que se conocía dicho modelo. Sin embargo, hay muchos casos en los que el modelo es desconocido y el agente tiene que aprender de la interacción con el entorno, tendiendo a reforzar aquellas acciones que den

lugar a las respuestas más favorables. En resumen, lo que se conoce como aprendizaje por refuerzo.

Para solucionar el problema dual existen diversos métodos de muy distinta naturaleza. No obstante, en este trabajo vamos a centrarnos únicamente en los métodos basados en gradiente para la resolución de problemas con restricciones, como los presentados en la sección 8.3.3. La idea subyacente al uso de este tipo de métodos será desarrollada en el siguiente capítulo, pero a grandes rasgos vendrá motivada porque nos van a permitir emplear una aproximación estocástica del gradiente obtenida a partir de las muestras de experiencia generadas al interactuar con el entorno siguiendo la política extraída de la variable dual. En definitiva, los métodos de gradiente van a ser la opción más favorable para el desarrollo de algoritmos de aprendizaje por refuerzo basados en la teoría dual.

De entre las distintas técnicas expuestas en 8.3.3, la propuesta por K.J. Arrow y L. Hurwicz deberá ser descartada ya que cuando el Lagrangiano es lineal tanto en la variable primal como en la dual, tal y como sucede en nuestro caso de acuerdo a (9.12), este método no converge y se mantiene en un estado oscilante [7]. En lo que respecta a la elección entre los dos métodos restantes, ascenso dual y Lagrangiano aumentado, se elegirá el primero de ellos. No obstante, en un futuro se estudiará la resolución de este mismo problema pero mediante el método del Lagrangiano aumentado, y las ventajas e inconvenientes que ello pueda suponer.

9.3.1. Método de ascenso dual

En el capítulo anterior se estudió la interpretación de la dualidad de Lagrange como un punto de silla y cómo el método de ascenso dual nos permite encontrarlo. Dado que nuestro problema primal satisface la condición de dualidad fuerte por tratarse de un programa lineal, se puede garantizar tal y como se detalló en la sección 8.3.2 que encontrar el punto de silla del Lagrangiano equivale a encontrar las variables primal y dual óptimas, es decir, v^* y d^*. De este modo, mediante la técnica de ascenso dual se va a poder derivar la política de comportamiento óptima de nuestro problema de control.

Recordando de la sección 8.3.3, la formulación del método de ascenso dual para un problema con restricciones de desigualdad es la siguiente:

$$\begin{aligned} v_{k+1} &\coloneqq \arg\min_{v} \mathcal{L}(v, d_k) \\ d_{k+1} &\coloneqq [d_k + \alpha \nabla_d \mathcal{L}(v_{k+1}, d)]_+ \end{aligned} \tag{9.25}$$

definiéndose $[\quad]_+$ como la proyección $u_+ = \max\{u, 0\}$ para cada componente del vector u.

Es decir, primero se minimiza el Lagrangiano sobre la variable primal, y a continuación se maximiza sobre la variable dual mediante ascenso por gradiente, y proyectamos la

actualización de d sobre el conjunto de números reales no negativos $\mathbb{R}_+$ con el objetivo de garantizar el cumplimiento de la restricción $d \geq 0$ del problema dual (9.17). Para nuestro problema concreto, definido por el problema primal (9.11) y con Lagrangiano dado por (9.12), el método de ascenso dual mostrado en (9.25) resulta en:

$$\begin{aligned} v_{k+1} &\coloneqq \arg\min_{v} \mu^T v + d_k^T \left(\mathcal{R} + \gamma \mathcal{P} v - \Xi^T v\right) \\ d_{k+1} &\coloneqq \left[d_k + \alpha \left(\mathcal{R} + \gamma \mathcal{P} v_{k+1} - \Xi^T v_{k+1}\right)\right]_+ \end{aligned} \tag{9.26}$$

Como ya se mencionó al comienzo de esta sección, el Lagrangiano del problema que se está buscando resolver es lineal tanto en la variable primal como en la dual. De este modo, la actualización de la variable primal según (9.26) dado un vector d, va a suponer la minimización de una función afín. Parece claro notar que esto va a ser un problema, pues las funciones afines no están acotadas por abajo y únicamente se minimizan en el infinito, de manera que la variable primal acabaría divergiendo.

Observación 9.1. Dado que el Lagrangiano es también lineal en la variable dual, va a ocurrir lo mismo con la maximización del Lagrangiano en d dado un vector v. No obstante, esto no va a suponer un problema ya que lo que a nosotros nos interesa no es d, sino π_d, y tal y como se enuncia en el teorema 9.2, $\pi_d\left(a \mid s\right)$ se obtiene tras un proceso de normalización de $d(s,a)$ sobre las acciones, para cada $s \in \mathcal{S}$. Asumiendo que se cumple la suposición 2, de acuerdo a los teoremas 9.3 y 9.5 a partir de d^* se podrá extraer una política π_{d^*} determinista, de manera que cuando se alcance dicha política determinista, podremos asumir que se ha encontrado el máximo del Lagrangiano sobre la variable dual.

Para solventar el problema de la linealidad en v y seguir manteniendo la simplicidad del método de ascenso dual, se va a dejar de lado la idea de minimizar el Lagrangiano en la variable primal y se va a formular una versión modificada, adaptada para el caso en que el Lagrangiano es lineal en las dos variables de optimización. La alternativa que se propone es actualizar v con la función valor obtenida de seguir la política π_{d_k} extraída de la variable dual, es decir:

$$\begin{aligned} v_{k+1} &\coloneqq v^{\pi_{d_k}} \in \mathcal{F}^{PR} \\ d_{k+1} &\coloneqq \left[d_k + \alpha \left(\mathcal{R} + \gamma \mathcal{P} v_{k+1} - \Xi^T v_{k+1}\right)\right]_+ \end{aligned} \tag{9.27}$$

donde $\mathcal{F}^{PR}$ es la región factible del problema primal. El hecho de que $v^{\pi_{d_k}}$ tenga que pertenecer a $\mathcal{F}^{PR}$ es una condición necesaria para poder alcanzar la solución óptima. Para obtener un vector v que cumpla estas condiciones, se resolverán las ecuaciones de Bellman siguiendo la política π_{d_k}.

Teorema 9.6. *Cualquier punto v que satisfaga las ecuaciones de Bellman para una política dada, será un punto factible del problema primal que permite resolver las ecuaciones*

óptimas de Bellman.

La demostración del teorema 9.6 aparece detallada en el Anexo I.

Con esta nueva manera de actualizar v, se conseguirá asegurar que las variables primal y dual siguen estando acopladas y que pertenecen al conjunto de puntos factibles. Además, permite hacer un interpretación muy conveniente del método de ascenso dual alternativo propuesto:

1. La actualización de la variable primal v se puede entender como una etapa de *predicción* en la cual se obtiene la función valor para una política dada por d_k.

2. La actualización de la variable dual d se puede entender como una etapa de *control* en la cual se aprende una política que mejora la función valor $v^{\pi_{d_k}}$.

Recordando el corolario 9.1, las variables primal y dual se relacionan de la siguiente manera:

$$\sum_{j\in\mathcal{S}} \mu(j) v^{\pi_d}(j) = \sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}} d(s,a)\mathcal{R}_s^a$$

De ello se desprende que maximizar la variable dual d equivale a maximizar la recompensa total descontada esperada, o lo que es lo mismo, maximizar la función valor de cada estado. En consecuencia, en cada actualización de la variable dual que se realice se obtendrá una política nueva igual o mejor que la anterior. Dado que la actualización del vector d contempla además el cumplimiento de las restricciones del problema primal, cuando se haya alcanzado la solución óptima v^* del problema primal, la política π_d extraída de la variable dual no podrá mejorarse más y la política nueva obtenida será igual que la anterior. Se podrá afirmar entonces, según el corolario 9.1 y el teorema 9.4, que se ha alcanzado la solución óptima del problema dual, y por consiguiente, que se ha encontrado la política de control óptima.

Como vemos, esta formulación alternativa del método de ascenso dual podrá seguir garantizando encontrar los puntos óptimos de los problemas primal y dual.

Para diferenciar la versión alternativa del método de ascenso dual (9.27) que se acaba de presentar de su versión original (9.25), de aquí en adelante nos referiremos a la primera de ellas por el nombre de *Bellman-ascenso dual*, pues la actualización de la variable primal v ahora se lleva a cabo mediante la resolución de la ecuación de Bellman de la función valor de estados.

Capítulo 10

Desarrollo de un algoritmo primal-dual novel

En el capítulo anterior se ha estudiado un enfoque primal-dual del problema de control óptimo a resolver, enfoque poco explorado hasta el momento en la literatura relativa al aprendizaje por refuerzo. Se demostró cómo a partir de la variable dual $d(s, a)$ del problema dual asociado se puede extraer una política de comportamiento óptima, y se llegó a la conclusión de que resolver el problema dual equivalía a resolver el problema de control planteado. Para resolver el problema dual, se presentaron diferentes alternativas de entre las cuales se eligió el método de ascenso dual, al cual se le hicieron algunas modificaciones en la etapa de actualización de la variable primal para adaptarlo a la casuística de nuestro problema. A esta versión modificada del método de ascenso dual se le dio el nombre de Bellman-ascenso dual, de manera que pudiéramos diferenciarla de la técnica de ascenso dual original.

En este capítulo, se va a formalizar un algoritmo que nos permita resolver el problema de control óptimo a través del método Bellman-ascenso dual desarrollado, cuando el conjunto de estados y acciones es pequeño y discreto. Se derivará una versión en la cual se conoce el modelo del entorno de nuestro problema, o *basada en modelo* y otra en la que carecemos de él, o *libre de modelo*. Para la aplicación en problemas de aprendizaje por refuerzo, será esta última versión la que nos resulte de mayor interés, pues permitirá incorporar las muestras de experiencia generadas al interactuar con el entorno a lo largo del tiempo.

A continuación, se pondrá a prueba el algoritmo planteado, demostrando de manera empírica la convergencia de cada etapa del método Bellman-ascenso dual. Para ello, se usará como problema de estudio un MDP relativamente sencillo, conocido en el mundo del aprendizaje por refuerzo como *random walk*. Por último, tras haber validado el algoritmo, se evaluará y comparará con el estado del arte para el problema anteriormente citado y

otro de mayor dificultad: el problema del paseo por el acantilado.

10.1. Algoritmo Bellman-ascenso dual

El método Bellman-ascenso dual va a permitir hacer una clara distinción de los dos sub-problemas típicos que se plantean a la hora de resolver problemas de aprendizaje por refuerzo: predicción y control. Más concretamente, se va a hacer la siguiente interpretación:

1. La actualización de la variable primal v se puede entender como una etapa de *predicción* en la cual se obtiene la función valor para una política dada por d_k.

2. La actualización de la variable dual d se puede entender como una etapa de *control* en la cual se aprende una política que mejora la función valor $v^{\pi_{d_k}}$.

Para mayor comodidad y comprensión lectora, se recuerda a continuación el método Bellman-ascenso dual desarrollado en el capítulo 9:

$$\begin{aligned} v_{k+1} &\coloneqq v^{\pi_{d_k}} \in \mathcal{F}^{PR} \\ d_{k+1} &\coloneqq \left[d_k + \alpha\left(\mathcal{R} + \gamma\mathcal{P}v_{k+1} - \Xi^T v_{k+1}\right)\right]_+ \end{aligned} \tag{10.1}$$

donde $\mathcal{F}^{PR}$ es la región factible del problema primal, y para obtener un vector v que cumpla estas condiciones se propone resolver la ecuación de Bellman siguiendo la política π_{d_k}.

Bajo el enfoque de predicción y control, típico de aprendizaje por refuerzo, se van a analizar a continuación ambas etapas de actualización de (10.1) y a presentar diferentes formas de abordarlas en función de si se conoce el modelo del entorno o no. Tras este análisis, se formularán dos versiones del algoritmo Bellman-ascenso dual: una basada en modelo y otra libre de modelo.

10.1.1. Predicción

Tal y como se detalló en el capítulo 9, la actualización de la variable primal (o etapa de predicción) va a consistir en la resolución de la ecuación de Bellman de la función valor de estados, cuando se sigue la política π_{d_k}. Como se vio en el capítulo 5, la ecuación de Bellman se podrá resolver de manera exacta, lo que llamaremos predicción basada en modelo, o de forma aproximada a través de la interacción con el entorno, a lo que nos referiremos como predicción libre de modelo. De nuevo, será el método basado en la interacción con el entorno el que nos interese de cara a formular un algoritmo competente de aprendizaje por refuerzo.

Predicción basada en modelo

Cuando se conoce el modelo del entorno de nuestro problema, se puede llevar a cabo la actualización de la variable primal de dos formas principalmente: resolviendo la ecuación de Bellman a través de su solución en forma cerrada (4.33)–(4.34), o mediante programación dinámica (ver sección 5.1.2), iterando hasta asegurar un cierto grado de convergencia δ_v de la función valor. Dado que las dos técnicas van a derivar en la solución exacta de la ecuación de Bellman, será indistinto[1] usar una u otra. Por ello, en lo que sigue del documento se trabajará con la solución basada en DP, mostrada en el algoritmo 10.1.

Algoritmo 10.1 Predicción basada en modelo: actualización mediante la resolución de la ecuación de Bellman a través de programación dinámica.

Entrada: π_{d_k}, la política a evaluar.
Salida: v_{k+1}, la función valor de estados para la política π_{d_k}.

1: Inicializar $v(s)$ arbitrariamente (e.g., $v(s) = 0$), para todo $s \in \mathcal{S}$.
2: **repetir**
3: $\quad \Delta \leftarrow 0$
4: $\quad$ **para** todo $s \in \mathcal{S}$ **hacer**
5: $\quad\quad v_{antigua} \leftarrow v(s)$
6: $\quad\quad v(s) \leftarrow \sum_{a \in \mathcal{A}} \pi_{d_k}(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v(s')\right)$
7: $\quad\quad \Delta \leftarrow \text{máx}\left[\Delta, |v(s) - v_{antigua}|\right]$
8: **hasta que** $\Delta < \delta_v$
9: $v_{k+1} \leftarrow v$
10: **devolver** v_{k+1}

Predicción libre de modelo

Cuando no se conozca el modelo del entorno de nuestro problema, a través de la interacción con dicho entorno se podrá estimar la solución de la ecuación de Bellman de varias maneras. De entre todas ellas, se va a considerar la presentada en la sección 5.2, TD. Por limitaciones de tiempo no fue posible probar otras alternativas como el método de Monte-Carlo o TD(λ); no obstante, en un futuro se estudiarán las ventajas e inconvenientes que puedan suponer estas dos técnicas. El pseudocódigo de esta etapa de predicción se muestra en el algoritmo 10.2.

[1] Será indistinto usar una técnica u otra en lo que al resultado final se refiere. Si se evaluase el coste computacional, una técnica sería más eficiente que la otra debido a la inversión de matrices. No obstante, dado que esa métrica se escapa del alcance de este trabajo, no será tenida en cuenta.

Algoritmo 10.2 Predicción libre de modelo: actualización mediante la estimación de la solución a la ecuación de Bellman a través de TD.

Entrada: π_{d_k}, la política a evaluar. α_{TD}, la tasa de aprendizaje.
Salida: v_{k+1}, la estimación de la función valor de estados $v^{\pi_{d_k}}$.

Inicializar $v(s)$ arbitrariamente (e.g., $v(s) = 0$), para todo $s \in \mathcal{S}$.
repetir(para cada episodio)
    $\Delta \leftarrow 0$
    Inicializar s
    repetir(para cada paso en el episodio)
        $v_{antigua} \leftarrow v(s)$
        Escoger la acción $a \sim \pi_{d_k}(\cdot|s)$
        Tomar la acción a y observar r, s'
        $v(s) \leftarrow v(s) + \alpha_{TD}(r + \gamma v(s') - v(s))$
        $\Delta \leftarrow \text{máx}\left[\Delta, |v(s) - v_{antigua}|\right]$
        $s \leftarrow s'$
    hasta que s sea terminal
hasta que no podamos correr más episodios o $\Delta < \delta_v$
$v_{k+1} \leftarrow v$
devolver v_{k+1}

Como puede apreciarse, ahora la solución a la ecuación de Bellman se obtiene a través de la experiencia generada, de manera que se podrá estimar tan bien como número de episodios experimentados (o simulados) haya disponibles. Resulta conveniente recordar que por muchas muestras de experiencia de que se dispongan, TD siempre va a introducir un ligero sesgo tal y como se vio en la sección 5.2, de manera que nunca se llegará a converger al valor óptimo de la función valor. Además, también existirá un cierto nivel de ruido proporcional a la tasa de aprendizaje α.

La actualización de v mediante la aproximación por TD se realizará tantas veces como nos permita el numero de episodios disponibles, o hasta asegurar un determinado grado de convergencia δ_v de la función valor, lo que ocurra antes.

10.1.2. Control

La actualización de la variable dual (o etapa de control), consistirá en la maximización en d del Lagrangiano (9.12) para una determinada variable primal v, mediante ascenso por gradiente. Como se vio en el capítulo 9, esta maximización se traducirá en aprender una política que mejore la función valor $v^{\pi_{d_k}}$ previamente obtenida en la etapa de predicción.

Del mismo modo que en el caso anterior, se van a presentar dos formas de llevar a cabo esta fase en función de si se conoce el modelo del entorno o no.

Control basado en modelo

Cuando se conoce el modelo del entorno de nuestro problema, se puede actualizar d iterando según lo establecido en (10.1). Esto da lugar al algoritmo 10.3.

Algoritmo 10.3 Control basado en modelo: actualización mediante ascenso por gradiente exacto en la variable dual d.

Entrada: v_{k+1}, la función valor de estados nueva obtenida en la etapa de predicción. d_k, la variable dual actual que se quiere mejorar. $\alpha_{_D}$, la tasa de aprendizaje.
Salida: d_{k+1}, la variable dual mejorada.

1: $d(s,a) \leftarrow d_k(s,a)$.
2: **para** todo $(s,a) \in \{\mathcal{S} \times \mathcal{A}\}$ **hacer**
3: $\quad d(s,a) \leftarrow d(s,a) + \alpha_{_D}(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_{k+1}(s') - v_{k+1}(s))$
4: $\quad d(s,a) \leftarrow \max(0, d(s,a))$
5: $d_{k+1} \leftarrow d$
6: **devolver** d_{k+1}

Control libre de modelo

En este punto es donde cobra gran importancia la idea de emplear métodos basados en gradiente para la resolución del problema dual. Cuando se desconoce el modelo del entorno de nuestro problema, no se puede conocer el gradiente del Lagrangiano en la variable dual, es decir, la actualización:

$$d_{k+1} \coloneqq [d_k + \alpha \nabla_d \mathcal{L}(v_{k+1}, d)]_+ = \left[d_k + \alpha \left(\mathcal{R} + \gamma \mathcal{P} v_{k+1} - \Xi^T v_{k+1}\right)\right]_+$$

ya no se podrá realizar. O no de manera exacta.

Lo que a continuación se propone es usar una aproximación estocástica del gradiente, $\widehat{\nabla_d \mathcal{L}}(v_{k+1}, d)$, obtenida a partir de las muestras de experiencia recogidas de la interacción con el entorno, de manera que la actualización de d se realice a través de un ascenso por gradiente estocástico:

$$\widehat{d}_{k+1} \coloneqq \left[\widehat{d}_k + \alpha \widehat{\nabla_d \mathcal{L}}(v_{k+1}, d)\right]_+$$

Para deducir la aproximación estocástica del gradiente que se empleará, será necesario partir de la derivada parcial del Lagrangiano respecto a $d(s,a)$:

$$\frac{\partial \mathcal{L}(v,d)}{\partial d(s,a)} = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v\left(s'\right) - v(s)$$

De este modo, lo que se va a hacer es:

1. Aproximar el valor esperado de la recompensa, $\mathcal{R}_s^a$ definida en (4.13), por la recompensa instantánea obtenida en el instante actual, r.

2. Aproximar el valor esperado de la función valor del siguiente estado, $\sum_{s' \in \mathcal{S}} \mathcal{P}^a_{ss'} v\,(s')$, por la estimación instantánea de la función valor en el siguiente estado, $v(s')$.

Teniendo en cuenta estas aproximaciones, se llega a la siguiente expresión para el gradiente:

$$\frac{\widehat{\partial \mathcal{L}}(v,d)}{\partial d(s,a)} = r + \gamma v(s') - v(s)$$

Por tanto, la actualización de d a través de ascenso por gradiente estocástico se puede expresar finalmente como:

$$\begin{aligned} \widehat{d}_{k+1}(s,a) &\coloneqq \left[\widehat{d}_k(s,a) + \alpha \frac{\widehat{\partial \mathcal{L}}(v_{k+1},d)}{\partial d(s,a)}\right]_+ \\ &= \left[\widehat{d}_k(s,a) + \alpha\left(r + \gamma v(s') - v(s)\right)\right]_+ \end{aligned} \tag{10.2}$$

siendo s el estado actual, s' el estado al que se transita y r la recompensa instantánea obtenida en la transición de s a s'. Si bien es cierto que existen más alternativas para la formulación del ascenso por gradiente estocástico, en este trabajo se ha escogido la implementación mostrada en (10.2). No obstante, en el futuro se probarán otras opciones tales como momentum, AdaGrad o RMSProp entre otras, y se estudiarán las ventajas e inconvenientes de emplear dichos métodos.

Como puede observarse, mediante esta actualización de la variable dual va a ser posible tener en cuenta las muestras de experiencia obtenidas en cada paso de cada episodio, acercándonos así al enfoque de aprendizaje por refuerzo que se estaba buscando.

Resulta de interés resaltar que, a diferencia del caso en que sí se disponía del modelo del entorno, en el cual se podía actualizar d en una sola iteración ya que se conocía la expresión exacta del gradiente, ahora nuestra estimación del gradiente se va a parecer tanto más a la original en función de cuantas muestras de experiencia dispongamos. Es por ello que será necesario actualizar d con las muestras obtenidas de correr varios episodios, pues así se conseguirá una aproximación más exacta del gradiente, y por consiguiente, una mejor estimación de la actualización de d. De este modo, la actualización de d se refinará de manera iterativa, bien hasta que no se disponga de más episodios que correr, o bien hasta asegurar un cierto grado de convergencia δ_{π_d} de la política que vayamos extrayendo en cada iteración[2].

El algoritmo de control resultante de usar el método de actualización descrito se muestra en el algoritmo 10.4.

Al igual que ocurría en la predicción libre de modelo, de nuevo, podremos estimar tan bien como número de episodios experimentados (o simulados) haya disponibles. No

[2]Tal y como se detalló en la observación 9.1, la garantía de convergencia no la tendremos en la variable dual, sino en la política extraída de la variable dual. Como consecuencia, habrá que medir el nivel de convergencia a partir de la política π_d que vayamos extrayendo en cada iteración.

Algoritmo 10.4 Control libre de modelo: actualización mediante ascenso por gradiente estocástico en la variable dual d.

Entrada: v_{k+1}, la función valor de estados nueva obtenida en la etapa de predicción. d_k, la variable dual actual que se quiere mejorar. $\alpha_{_D}$, la tasa de aprendizaje.

Salida: d_{k+1}, la variable dual mejorada.

$d(s,a) \leftarrow d_k(s,a)$
repetir(para cada episodio)
    $\Delta \leftarrow 0$
    Inicializar s
    repetir(para cada paso en el episodio)
        Escoger la acción $a \sim \pi_{d_k}(\cdot|s)$
        $d_{antigua} \leftarrow d(s,a)$
        Tomar la acción a y observar r, s'
        $d(s,a) \leftarrow d(s,a) + \alpha_{_D}(r + \gamma v_{k+1}(s') - v_{k+1}(s))$
        $d(s,a) \leftarrow \text{máx}\,(0, d(s,a))$
        $\Delta \leftarrow \text{máx}\left[\Delta, |\pi_d\,(a \mid s) - \pi_{d_{antigua}}\,(a \mid s)\,|\right]$
        $s \leftarrow s'$
    hasta que s sea terminal
hasta que no podamos correr más episodios o $\Delta < \delta_{\pi_d}$
$d_{k+1} \leftarrow d$
devolver d_{k+1}

obstante, los métodos de gradiente estocástico presentan dos problemas adicionales: (1) cuentan con un cierto ruido debido a la aproximación, y (2) aparece un sesgo proporcional a la tasa de aprendizaje α que impide converger al valor óptimo deseado del parámetro que se quiere estimar [9].

Estos dos últimos problemas en principio no afectarán al algoritmo ya que, tal y como se remarcó en la observación 9.1, no se buscará la convergencia de la variable dual a su valor óptimo, sino la obtención de una política óptima determinista, y esta última se puede conseguir sin necesidad de que d tome su valor exacto esperado.

Una vez analizada cada etapa del método Bellman-ascenso dual en sus dos versiones según si se conoce el modelo o no, se puede pasar a formular el algoritmo propiamente dicho.

10.1.3. Algoritmo Bellman-ascenso dual basado en modelo (BDA-MB)

El algoritmo 10.5 muestra la implementación para el caso en que se conoce el modelo del entorno del problema. Para la etapa de predicción se ha escogido emplear la actualización de la variable primal por medio de programación dinámica.

Algoritmo 10.5 Algoritmo Bellman-ascenso dual basado en modelo.

Entrada: $\alpha_{_D}$, la tasa de aprendizaje de la etapa de control.
Salida: π^*, la política óptima.

Inicializar $v(s)$ arbitrariamente (e.g., $v(s) = 0$), para todo $s \in \mathcal{S}$
Inicializar $d(s, a)$ con cualquier valor positivo, para todo $(s, a) \in \{\mathcal{S} \times \mathcal{A}\}$
$d_k \leftarrow d$
$v_k \leftarrow v$
repetir ▷ Bucle principal de Bellman-ascenso dual
    repetir ▷ Etapa de predicción: evaluación de la política
        $\Delta \leftarrow 0$
        para todo $s \in \mathcal{S}$ **hacer**
            $v_{antigua} \leftarrow v(s)$
            $v(s) \leftarrow \sum_{a \in \mathcal{A}} \pi_{d_k}(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v(s')\right)$
            $\Delta \leftarrow \text{máx}\left[\Delta, |v(s) - v_{antigua}|\right]$
    hasta que $\Delta < \delta_v$
    $v_{k+1} \leftarrow v$
    para todo $(s, a) \in \{\mathcal{S} \times \mathcal{A}\}$ **hacer** ▷ Etapa de control: mejora de la política
        $d(s, a) \leftarrow d(s, a) + \alpha_{_D}(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_{k+1}(s') - v_{k+1}(s))$
        $d(s, a) \leftarrow \text{máx}\,(0, d(s, a))$
    $\xi = \|v_{k+1} - v_k\|_2^2$
    $d_k \leftarrow d$
    $v_k \leftarrow v_{k+1}$
hasta que $\xi < \delta$
$\pi^* \leftarrow \pi_{d_k}$
devolver π^*

De aquí en adelante, será común referirnos al algoritmo 10.5 por las siglas BDA-MB, derivadas de su nombre en inglés *(Bellman-Dual Ascent - Model Based)*.

10.1.4. Algoritmo Bellman-ascenso dual libre de modelo (BDA-MF)

Para la implementación de este algoritmo se hará uso de los dos esquemas presentados para predicción y control cuando no se conoce el modelo del entorno, pero con una ligera modificación en el criterio de convergencia.

Como se puede ver en (10.1), el método Bellman-ascenso dual consiste en dos procesos que interactúan entre sí, uno haciendo la función valor consistente con la política actual (actualización de la variable primal) y el otro mejorando la política respecto a la función valor actual (actualización de la variable dual). En el algoritmo 10.5 Bellman-ascenso dual basado en modelo, estas dos etapas se alternan, llegando cada una a la convergencia antes de que comience la otra. Esta misma idea pretende reflejarse en las etapas de predicción y control libre de modelo presentadas en los algoritmos 10.2 y 10.4 cuando se dice que el bucle principal sólo se detendrá cuando no podamos correr más episodios o se haya alcanzado un cierto nivel de convergencia δ, lo que ocurra antes. No obstante, y

al igual que ocurre con el método GPI (ver subsección 5.1.2), asegurar la convergencia (o en su defecto agotar todos los episodios que vamos a ser capaces de correr intentando buscarla) no será necesario. Dado que en el capítulo 9 se demostró que las dos etapas de actualización avanzan siempre en la dirección de los valores óptimos, bastará con actualizar ambas variables a partir de la experiencia generada con un número N_{epi} reducido de episodios. De esta manera, y siguiendo el mismo principio subyacente a GPI, se conseguirá alcanzar tanto la política óptima π_{d^*} como la función valor óptima v^* mediante un proceso iterativo y alterno de mejora. La idea que se ha pretendido transmitir queda mejor reflejada en la figura 5.2.

Relacionado con estas cuestiones de convergencia, no deberá perderse de vista que se trata de un algoritmo que basa su comportamiento en las muestras que obtiene al interactuar con el medio. Por tanto, cuanto mejor sea el muestreo del entorno, más acertado será el modelo que aprenda del mismo y por tanto se podrá descubrir la política óptima. Con este objetivo en mente, será necesario asegurar un compromiso entre el grado de exploración y de explotación del agente. Una manera de conseguirlo será seguir una política $\epsilon - \pi_{d_k}$, la cual consistirá en usar la siguiente regla de decisión para escoger la acción a cuando el entorno se encuentre en el estado s:

$$a = \begin{cases} a \sim \pi_{d_k}(\cdot|s) & \text{con probabilidad } 1-\epsilon \\ \text{acción escogida de manera uniforme en } \mathcal{A} & \text{con probabilidad } \epsilon \end{cases} \tag{10.3}$$

donde $\epsilon \in (0, 1)$ es el parámetro de exploración. Como se puede deducir, cuanto mayor sea ϵ, mayor será la exploración del entorno y cuanto menor sea, mayor será la explotación de la información disponible.

Antes de pasar a presentar el algoritmo, será necesario resaltar otro matiz. Observando detenidamente los algoritmos 10.2 y 10.4, se puede notar que las dos etapas actualizan sus respectivas variables con muestras de experiencia generada tras seguir la política π_{d_k}, con lo cual, parece sensato emplear las mismas muestras de experiencia en las dos etapas. De este modo, se consigue reducir el número de episodios necesarios para aprender, sin alterar el funcionamiento del algoritmo. Para que esto sea posible se introducirá el concepto de *memoria de repetición*. Esta memoria almacenará en un conjunto de datos $\mathbb{S} = \{e_t\}_{t=1}^{T}$, muestras e_t de la experiencia recabada por el agente al interactuar con el medio a lo largo de varios episodios. La muestra e_t que se almacenará será la tupla $e_t = s_t, a_t, r_{t+1}, s_{t+1}$.

Teniendo en cuenta todas estas consideraciones, el algoritmo Bellman-ascenso dual libre de modelo desarrollado será el mostrado en el algoritmo 10.6. De aquí en adelante, será común referirnos a él por las siglas BDA-MF, derivadas de su nombre en inglés *(Bellman-Dual Ascent - Model Free)*.

Algoritmo 10.6 Algoritmo Bellman-ascenso dual libre de modelo.

Entrada: α_{TD}, la tasa de aprendizaje de la etapa de predicción. α_D, la tasa de aprendizaje de la etapa de control. ϵ, el parámetro de exploración.

Salida: π, la política óptima π^* aproximada.

Inicializar $v(s)$ arbitrariamente (e.g., $v(s) = 0$), para todo $s \in \mathcal{S}$
Inicializar $d(s, a)$ con cualquier valor positivo, para todo $(s, a) \in \{\mathcal{S} \times \mathcal{A}\}$
$d_k \leftarrow d$
$v_k \leftarrow v$
repetir ▷ Bucle principal de Bellman-ascenso dual
    $\xi \leftarrow 0$
    Inicializar la memoria de repetición $\mathbb{S} = \{\}$ vacía
    repetir(para cada episodio) ▷ Etapa de predicción: evaluación de la política
        Inicializar s
        repetir(para cada paso en el episodio)
            Escoger la acción $a \sim \epsilon - \pi_{d_k}(\cdot|s)$
            Tomar la acción a y observar r, s'
            Aumentar la memoria de repetición $\mathbb{S}$ con la muestra actual $(s,\ a,\ r,\ s')$
            $v(s) \leftarrow v(s) + \alpha_{TD}(r + \gamma v(s') - v(s))$
            $\xi \leftarrow \text{máx}\left[\xi, \|v(s) - v_k(s)\|_2^2\right]$
            $s \leftarrow s'$
        hasta que s sea terminal
    hasta que se hayan corrido N_{epi} episodios
    $v_{k+1} \leftarrow v$
    para para cada muestra de $\mathbb{S}$ **hacer** ▷ Etapa de control: mejora de la política
        Recuperar $(s,\ a,\ r,\ s')$
        $d(s, a) \leftarrow d(s, a) + \alpha_D(r + \gamma v_{k+1}(s') - v_{k+1}(s))$
        $d(s, a) \leftarrow \text{máx}\,(0, d(s, a))$
    $d_k \leftarrow d$
    $v_k \leftarrow v_{k+1}$
hasta que no se puedan correr más episodios o $\xi < \delta$
$\pi \leftarrow \pi_{d_k}$
devolver π

Destacar que los N_{epi} episodios que se corren en cada iteración del bucle principal, son diferentes de una iteración a otra. Otra alternativa sería iterar varias veces sobre los mismos N_{epi} episodios hasta converger en ambas etapas, técnica que se conoce como *entrenamiento por lotes*. No obstante, esta versión no se implementó, aunque se plantea la evaluación de su comportamiento en el futuro.

Así concluye la presentación del algoritmo Bellman-ascenso dual novel.

Como se puede apreciar, desde el punto de vista de la problemática de aprendizaje por refuerzo, la implementación que será de mayor interés será la última que se detalló, Bellman-ascenso dual libre de modelo, ya que va a permitir aprender a partir de la inter-

acción con el entorno. Esta interpretación primal-dual para el desarrollo de algoritmos de aprendizaje por refuerzo no ha sido prácticamente explotada aún a día de hoy. Por tanto, el algoritmo 10.6, que será el foco de atención en lo que resta de documento, supondrá una aportación teórica muy interesante a la comunidad de aprendizaje por refuerzo.

10.2. Validación del algoritmo Bellman-ascenso dual libre de modelo

Ya con el algoritmo BDA-MF formalizado, la siguiente tarea consistirá en verificar la validez del modelo. Para ello, se probará cada etapa por separado y, una vez tengamos garantías de que funcionan, se hará una prueba funcional del algoritmo completo. Más concretamente, las pruebas que se van a realizar son las siguientes:

1. Prueba de convergencia de BDA-MB: lo primero de todo será comprobar si el algoritmo funciona cuando conocemos el modelo del entorno de nuestro problema.

2. Prueba de convergencia de la etapa de predicción libre de modelo: una vez comprobado que el algoritmo basado en modelo es capaz de resolver el problema, se harán varias pruebas para determinar si la etapa de predicción cuando se desconoce el modelo converge. Para ello se fijarán tres políticas conocidas y se estimará la función valor obtenida de seguir dichas políticas. Si se obtiene el valor esperado, se dará por superada esta prueba y se podrá afirmar que la etapa de predicción converge.

3. Prueba de convergencia de BDA-MF: llegados a esta fase, si las dos pruebas anteriores se han completado de manera satisfactoria, podremos extrapolar que si combinamos la etapa de predicción sin conocimiento del modelo con la de control supuesto conocido el modelo, podremos resolver nuestro problema con éxito. Por tanto, el siguiente paso natural será probar el algoritmo BDA-MF completo. En función de los resultados que se obtengan en esta fase se podrá justificar o no la validez del algoritmo.

Para esta primera fase de validación, las pruebas se realizarán sobre un problema típico en aprendizaje por refuerzo: *random walk*. El MDP que modela este problema se muestra en la figura 10.1.

1. El problema tendrá $|\mathcal{S}| = 13$ estados, de los cuales 2 serán estados terminales, $T1$ y $T2$.

2. El estado inicial será el central, $s = 6$.

3. El conjunto $\mathcal{A}$ estará formado por dos acciones: $a_1 = izquierda$, $a_2 = derecha$.

4. Habrá 7 estados con recompensa. De entre esos estados, la recompensa del estado $T2$ será la mayor de todas. El vector de recompensas sobre los estados será el siguiente:
$$\mathcal{R} = [2,\ 0,\ 1,\ 2,\ 0,\ 0,\ 0,\ 0,\ 2,\ 0,\ 1,\ 1,\ 150]^T$$

5. La política óptima será ir siempre a la derecha de manera que acabemos en el estado $T2$, consiguiendo la mayor recompensa total acumulada.

6. La matriz de transición que modele el entorno del problema será estocástica. Más concretamente, la probabilidad de tomar la acción escogida será 0.8, y la probabilidad de que se lleve a cabo la contraria será 0.2. Es decir, de acuerdo a la figura 10.1, suponiendo que estamos en el estado $s = 6$ y que escogemos la acción $a_2 = derecha$ tendremos:
$$\begin{aligned} p(s' = 7 \mid s = 6, a = a_2) &= 0{,}8 \\ p(s' = 5 \mid s = 6, a = a_2) &= 0{,}2 \end{aligned} \tag{10.4}$$
Puesto que el problema tiene $|\mathcal{S}| = 13$ estados y $|\mathcal{A}| = 2$ acciones, la matriz de transiciones $\mathcal{P}$ será de dimensión 26×13. Dado que no aportaría información nueva sobre el problema más allá de la que se ha ejemplificado en (10.4), no se mostrará su representación.

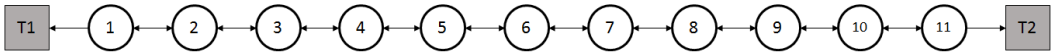


Figura 10.1: MDP que modela el problema *random walk* que se usará en las pruebas de validación del algoritmo BDA-MF.

Presentado ya el problema, se pueden comenzar las pruebas.

10.2.1. Convergencia del algoritmo Bellman-ascenso dual basado en modelo

Para la primera de las pruebas, se implementó el algoritmo 10.5: BDA-MB. Bajo este esquema, y conocido el vector de recompensas $\mathcal{R}$ y la matriz de transición $\mathcal{P}$ se llevaron a cabo varias ejecuciones para diferentes valores de la tasa de aprendizaje de la etapa de control, es decir, α_D. Puesto que en este problema sencillo conocemos la política óptima, se evaluó directamente la convergencia del algoritmo a través del error cuadrático entre la política extraída de la variable dual y la política óptima, promediando los resultados de 50 experimentos independientes

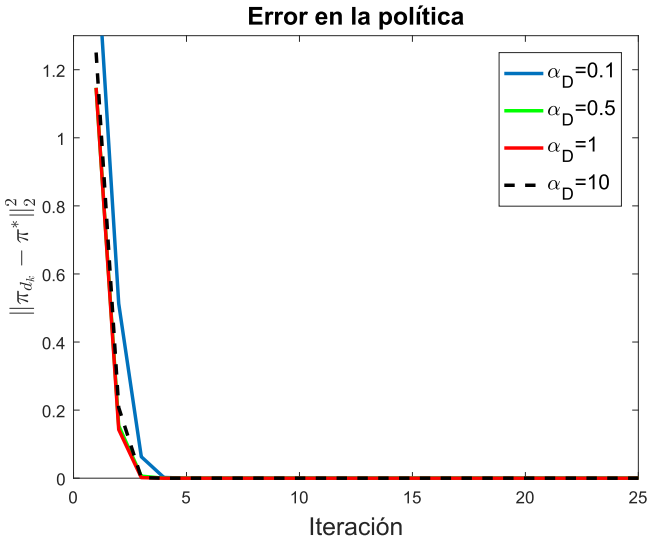


Figura 10.2: Error cuadrático medio de la política obtenida al resolver el problema *random walk* a través del algoritmo BDA-MB, para distintos valores de α_D.

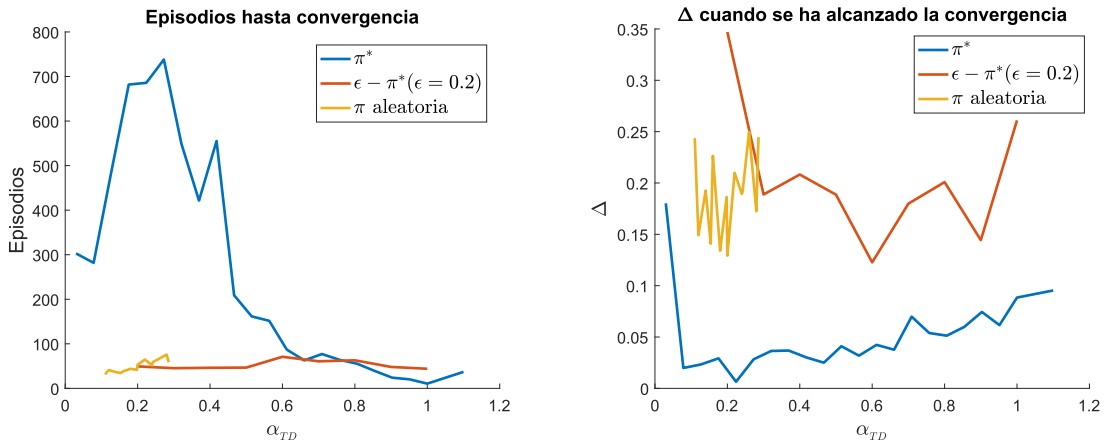
Tal y como se puede ver en la figura 10.2, en todos los casos se consiguió converger a la política óptima en un número pequeño de iteraciones. Además, a partir de valores de α_D mayores que 0.5 la velocidad de convergencia no mejora. Se encuentra que el mínimo número de iteraciones necesarias para alcanzar la política óptima es 4.

10.2.2. Convergencia de la etapa de predicción libre de modelo

Comprobado que el algoritmo funciona en su versión basada en modelo, se pasa ahora a probar la etapa de predicción con TD, la cual se corresponde con el algoritmo 10.2. Como cabe esperar, la velocidad de convergencia de TD dependerá de α_{TD}. Por tanto, se estudiará la convergencia para distintos valores de α_{TD}. Para realizar esta prueba, se estimó la función valor para tres políticas (parámetros de entrada del algoritmo 10.2) diferentes:

1. La política óptima, π^*.
2. Una política epsilon-óptima, $\epsilon - \pi^*$, con $\epsilon = 0{,}2$.
3. Una política generada aleatoriamente.

El motivo de haber escogido estas políticas es que nos permitirán modelar el comportamiento del algoritmo BDA-MF conforme vaya aprendiendo: al principio se inicializará con una política aleatoria que será altamente exploratoria. A continuación, irá convergiendo hacia la política óptima, comportándose de manera similar a una $\epsilon - \pi^*$ con un parámetro de exploración aun relativamente alto, y finalmente, convergerá a la política óptima.



(a) (b)

Figura 10.3: Resultados de convergencia de la etapa de predicción libre de modelo mediante TD. La figura (a) representa el número de episodios necesarios para garantizar la convergencia en v, en función de α_{TD}. La figura (b) representa el error relativo entre la iteración actual y la anterior en el momento de convergencia, en función de α_{TD}.

Para esta prueba, a diferencia del caso anterior, ya se podrá medir la convergencia a partir del error cuadrático medio en la política, pues esta es ahora un parámetro de entrada. Por tanto, habrá que utilizar un criterio basado en el valor que va tomando la función valor. La medida que eligió fue la siguiente:

$$\Delta = \text{máx}\left[\Delta, \frac{\|v_{k+1}(s) - v_k(s)\|_2}{\|v_{k+1}(s)\|_2}\right]$$

es decir, el error entre cada iteración de TD expresado de manera relativa a la función valor del estado actual (en el mismo estado), de manera que se detectase convergencia cuando $\Delta < \delta_v$.

Tras escoger el criterio de convergencia, se hicieron una serie de pruebas para diferentes valores de α_{TD}, promediando los resultados de 50 experimentos independientes. La figura 10.3 muestra dos gráficas con los resultados de mayor interés: una en la que se representa el número de episodios necesarios para garantizar la convergencia en v, en función del valor que tome α_{TD}, y otra que representa para cada valor de α_{TD}, la cantidad $\frac{\|v_{k+1}(s)-v_k(s)\|_2}{\|v_{k+1}(s)\|_2}$ que ha provocado la convergencia, que de alguna manera representa el porcentaje de variación de la iteración actual a la anterior, y por tanto cuán buena ha sido la convergencia.

Lo primero que llama la atención al ver estas gráficas es que ninguna de las tres curvas empieza y acaba en el mismo punto. El hecho de que unas empiecen antes o después se debe a que con determinados valores de α_{TD}, y en función de la política

evaluada, no se alcanzaba la convergencia con el número de episodios simulados, 5000, el cual consideramos más que suficiente para el tipo de problema evaluado. De manera similar, el hecho de que unas curvas acaben antes o después es debido a que a partir de un valor α_{TD} concreto, la estimación de v diverge y no se alcanza nunca el valor esperado.

Lo segundo que vemos es que el rango de valores de α_{TD} válido cuando la política es aleatoria es considerablemente pequeño en comparación con los otros dos casos. Como quedará demostrado en los resultados que aparecerán en este capítulo, ese rango no es realmente significativo ya que para esta prueba no se está teniendo en cuenta que la política mejore. Dado que cuando usemos el algoritmo BDA-MF se alternarán las etapas de predicción y control cada N_{epi} episodios, realmente nunca se va a tener un caso similar al mostrado en la curva amarilla de la figura 10.3, pues rápidamente se mejorará la política y dejará de ser aleatoria.

Para concluir, y de acuerdo a lo resultados de la figura 10.3, podemos garantizar la convergencia de la etapa de predicción mediante TD, y por tanto pasar a la prueba del algoritmo BDA-MF completo.

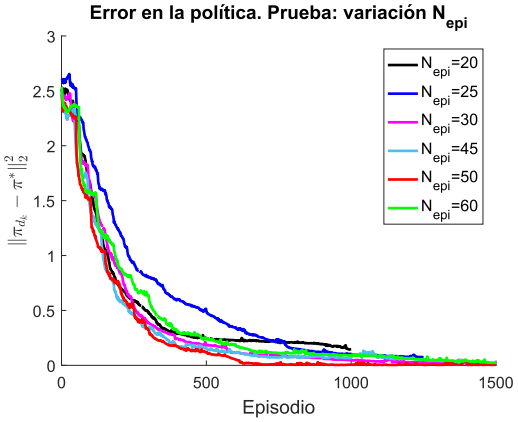
10.2.3. Convergencia del algoritmo Bellman-ascenso dual libre de modelo

Por último, pasamos a probar el algoritmo BDA-MF completo, según la implementación mostrada en el algoritmo 10.6. Ahora, la velocidad de convergencia dependerá de los siguientes parámetros: N_{epi}, α_D, α_{TD} y ϵ. Con el objetivo de estudiar la influencia de cada parámetro en la convergencia global del algoritmo, se llevaron a cabo diferentes pruebas variando cada uno de ellos, y promediando los resultados de 50 experimentos independientes. De nuevo, dado que conocemos la política óptima a la que deberemos converger retomaremos el criterio de convergencia que se empleó en la primera prueba, basado en el error cuadrático medio de la política obtenida respecto de la óptima esperada. Los resultados que se obtuvieron fueron los mostrados en la figura 10.4.

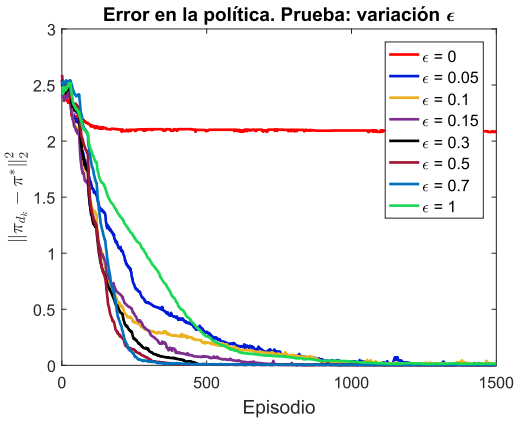
Como comentario general en vista de los resultados, de nuevo podemos garantizar la convergencia a la política óptima.

Entrando en los detalles particulares de cada gráfica, vemos en la figura 10.4 (a), que parece haber un punto de inflexión de mayor velocidad de convergencia cuando se simulan $N_{epi} = 50$ episodios en cada etapa del algoritmo. El hecho de que con 60 episodios el comportamiento sea peor, puede ser debido a que, como la etapa de predicción y de control están altamente acopladas, a partir de 50 episodios la aproximación del gradiente comienza a ser demasiado ruidosa. En consecuencia, este error se propaga a la estimación de d y la política derivada de la variable dual empieza a empeorar.

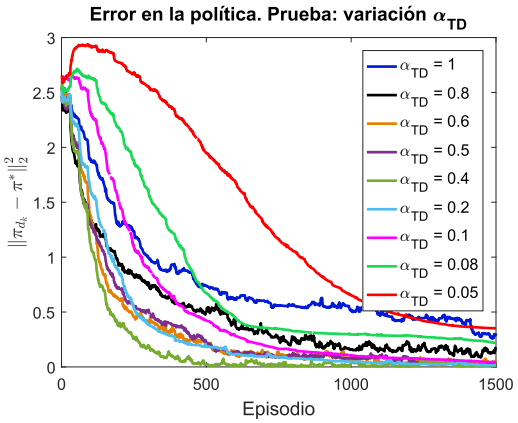
En base a los resultados mostrados en la figura 10.4 (b), se deduce que el algoritmo funciona adecuadamente para valores de ϵ altos, comprendidos entre 0.3 y 0.7. Es decir,



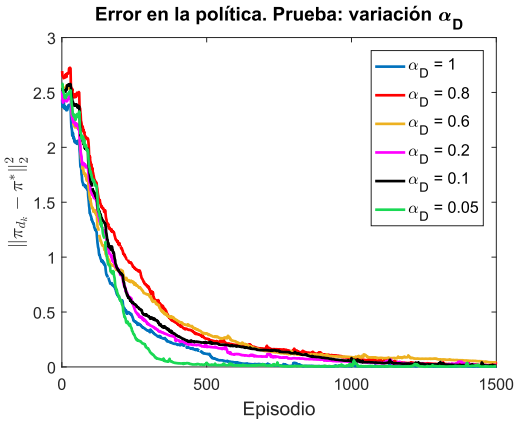
(a) Variación del número de episodios.



(b) Variación del parámetro de exploración.



(c) Variación de α_{TD}.



(d) Variación de α_D.

Figura 10.4: Error en la política al variar los parámetros característicos del algoritmo BDA-MF

este algoritmo funciona bien con mucha exploración, nunca llegando al extremo en que todo es exploración ($\epsilon = 1$), caso en el cual, como se puede apreciar, empeora la convergencia. Como cabía esperar, se observa también que cuando únicamente se explota la información disponible ($\epsilon = 0$), no es capaz de encontrar la política óptima, pues no hay manera de aprender el modelo del entorno. De alguna manera, este último caso equivale a re-aprender lo que ya se conoce, sin posibilidad de aumentar el conocimiento sobre el entorno del problema.

En la figura 10.4 (c) se aprecia un comportamiento similar al que ocurría en la figura (a): parece haber un punto de inflexión cuando $\alpha_{TD} = 0{,}4$. Una vez más, se piensa que esto sea debido al fuerte acoplo entre las etapas de predicción y control, ya que si aumentamos α_{TD}, aumentará también el ruido en las estimación de v. Puesto que la etapa de predicción empleará esa estimación de v para iterar sobre d, se puede dar la situación

de que para valores de α_{TD} grandes (en este caso mayores que 0.4), el ruido que presenta v sea tal que empeore la aproximación del gradiente, y en consecuencia la estimación de d y la política derivada de ella.

Finalmente, de la figura 10.4 (d) se concluye que cuanto menor es α_D, más rápida es la convergencia. En principio, esta idea puede parecer contraintuitiva, pues en los métodos de gradiente, cuanto menor es el paso de aprendizaje, más lenta es la convergencia. Se cree que este efecto pueda estar causado por el acoplo entre las etapas de predicción y control por el siguiente motivo: en la situación inicial, la estima que se tiene de v dista mucho de ser buena. Con una mala estimación de v, si se ejecuta la fase de control con un paso α_D alto, se determinará una política mala muy rápido. De este modo, el algoritmo deberá reajustarse desde una configuración peor hasta lograr alcanzar la óptima. Sin embargo, si α_D se escoge pequeño, aun partiendo de una mala estimación de v, la política convergerá de manera progresiva hacia la política óptima.

De este modo, se da por comprobada la validez del algoritmo Bellman-ascenso dual libre del modelo desarrollado para resolver problemas de aprendizaje por refuerzo. Además de validar el algoritmo, se ha podido conocer también el proceso típico de calibración, y de qué manera influye cada variable en el comportamiento global. Todas estas nociones serán tenidas en cuenta en lo que resta del capítulo, de cara a evaluar el algoritmo con otros problemas típicos de aprendizaje por refuerzo.

10.3. Evaluación del algoritmo

Para concluir el capítulo, se va a evaluar el algoritmo primal-dual novel desarrollado a través de la ejecución en dos problemas tipo, cada uno de ellos con dos versiones: una en la que la matriz de transición es determinista y otra en la que se le añade cierto carácter aleatorio. Para cada problema que se estudie, enfrentaremos los resultados obtenidos con aquellos alcanzados mediante dos soluciones ampliamente extendidas en la actualidad: SARSA y Q-learning. De este modo, se pretende comparar el algoritmo BDA-MF con el estado del arte.

Con el objetivo de situar al lector en el contexto de cada problema, antes de mostrar los resultados de cada ejecución se presentará el problema a resolver al igual que se hizo en el caso del MDP *random walk* empleado durante la fase de validación del algoritmo.

10.3.1. Metodología de evaluación

Cuando se trabaja con problemas reales, no conocemos cuál va a ser la política óptima de antemano. De hecho, precisamente por este motivo nace la necesidad de desarrollar métodos de que permitan aprender el comportamiento óptimo. Por ello, la métrica que

se empleó en las pruebas de validación anteriores, $\|\pi_{d_k} - \pi^*\|_2^2$, no podrá ser usada ahora para evaluar cuán buena es la convergencia de nuestro algoritmo.

Tal y como se comentó en los primeros capítulos, el objetivo del aprendizaje será maximizar la recompensa total acumulada que se recibe del entorno a largo plazo. Por tanto, de cara a evaluar cómo de bien se comporta un algoritmo de aprendizaje por refuerzo, el indicador que más se suele emplear es el retorno G (4.19)[3]. Se puede intuir fácilmente que cuanto mayor sea el retorno, mejor será la política obtenida, y que cuando dicho retorno se estabilice y deje de aumentar a medida que se simulan episodios, significará que se ha encontrado una política estable, candidata de ser la óptima. Cambiando el enfoque del problema, el retorno representa las ganancias que recibimos, de este modo, parece claro notar que siempre querremos maximizar nuestro beneficio en el menor tiempo (episodios) posible.

Otra observación importante antes de comenzar a mostrar resultados será la distinción entre la fase de aprendizaje y la fase de explotación o ejecución.

1. Durante la fase de aprendizaje, se utiliza una política con un cierto carácter exploratorio. Es decir, se aprenderá por medio de una política $\epsilon - \pi_d$ que dé pie a muestrear el entorno y aprender a comportarse en estados no visitados hasta el momento, y que podrían llevar a mejorar la política actual disponible.

2. Una vez se ha aprendido, el algoritmo interactuará con el entorno siguiendo la política real π_d, o lo que es lo mismo, eliminando el carácter exploratorio. Esta fase será de explotación del conocimiento disponible.

Para ejemplificar la importancia de estas dos fases, consideremos un ejemplo en que se quiere enseñar a conducir a un coche «inteligente». Antes de que el coche autónomo pueda ser utilizado por personas, será necesario enseñarle a conducir; es decir, las consecuencias de cada acción que decida tomar. Para ello, se emplean entornos virtuales de simulación de manera que puede aprender qué acciones son correctas y cuales no sin causar daños. Durante esta fase de aprendizaje, se da pie a que el coche pruebe nuevas acciones y aprenda si le van a suponer una mejora o no. Un ejemplo podría ser que en una curva hacia la izquierda, el coche decidiese girar a la derecha. El resultado en la vida real sería catastrófico, pero durante la simulación supone un refinamiento del modelo estimado, pues se aprende qué acciones no deben ser tomadas según el estado en que nos encontremos. Una vez hemos entrenado nuestro sistema de conducción, permitiendo ese ligero carácter exploratorio, ya podría hacerse una primera prueba real (aunque de nuevo por seguridad en un entorno virtual). Para ello, se deshabilitaría la exploración ($\epsilon = 0$) y únicamente

[3]Para poder comparar diferentes valores de retorno en una mismo problema, todas las ejecuciones deberán comenzar en el mismo estado inicial. Si esto no se hiciera así, las conclusiones extraídas de la G obtenida en dos pruebas diferentes, empezando en estados distintos, no tendría sentido.

se explotaría el conocimiento aprendido. Esta fase sería la de ejecución o explotación. El objetivo de esta fase es evaluar cómo de buena es la política que se ha alcanzado gracias a la fase de aprendizaje con exploración de manera que, si el retorno obtenido en esta etapa no es satisfactorio, se volvería de nuevo a la fase de aprendizaje hasta alcanzar el objetivo fijado. Una vez alcanzado el objetivo deseado, podría hacerse una primera prueba en un entorno real.

En base a estas dos fases, para la calibración de los parámetros N_{epi}, α_D, α_{TD} y ϵ del algoritmo BDA-MF deberemos fijarnos en dos curvas:

1. La curva de aprendizaje: representará la evolución del retorno a lo largo de los episodios simulados durante la fase de aprendizaje. Con esta curva, podremos hacernos una idea del número de episodios necesario para asegurar que G se estabiliza, o lo que es lo mismo, en cuántos episodios se consigue aprender una política estable, y por la naturaleza del algoritmo, óptima. Es decir, esta gráfica estará orientada a medir y comparar velocidades de convergencia.

2. La curva de explotación tras convergencia: representará el retorno conseguido al evaluar[4] la política estable obtenida durante el aprendizaje. Es decir, con esta curva podremos medir y comparar al final de cada experimento (esto es, cuando ya se ha estabilizado G), el beneficio obtenido con la política que se ha conseguido. En principio, si se sigue una política concreta sin exploración ni aprendizaje, el valor de G obtenido debería ser constante para todos los episodios simulados. No obstante, puede darse el caso de que (1) el ruido de estimación del gradiente sea muy alto y la política extraída de la variable dual no llegue a ser nunca determinista, aunque sí muy cercana, o (2) la matriz de transición del problema evaluado tenga cierto carácter estocástico como el que se ejemplificó en el problema *random walk* de validación del algoritmo. En cualquier caso, la curva de explotación siempre deberá ser una gráfica constante, o constante pero ligeramente ruidosa. Por este motivo, y para facilitar la lectura de los resultados, en lugar de enseñar una gráfica de esta naturaleza, se hará el promedio de los valores de G obtenidos en cada episodio del experimento de evaluación simulado y se mostrará en una tabla.

De cara a decidir qué parámetros son mejores o peores se premiará, primero a aquellos que alcancen el valor más alto de G en la curva de explotación tras convergencia, y a continuación los que aseguren una convergencia más rápida en la gráfica de aprendizaje. En resumen, tal y como se dijo al comienzo de la sección, se buscará maximizar nuestro beneficio en el menor tiempo (episodios) posible. Para dejar claro este enfoque, supongamos que disponemos de una plataforma virtual de e-Trading en la que queremos implementar

[4]Para generar esta curva únicamente se simularán episodios que siguen la política a evaluar, sin aprendizaje alguno.

una solución de aprendizaje por refuerzo para que maximice nuestras ganancias monetarias. De poco sirve tener un algoritmo que consiga hacernos ricos en 100 años, si hay otro que nos consigue ganancias considerablemente razonables en un plazo de 2 años.

Conocida ya la metodología y los criterios a seguir para evaluar nuestro algoritmo y poder compararlo con SARSA y Q-learning, pasamos a presentar los problemas y los resultados obtenidos.

10.3.2. Problema bajo estudio 1: random walk (transiciones deterministas)

Presentación del problema

El MDP que modela este problema se muestra en la figura 10.1.

1. El problema tendrá $|\mathcal{S}| = 13$ estados, de los cuales 2 serán estados terminales, $T1$ y $T2$.

2. El estado inicial será el primero no terminal a la izquierda, $s = 1$.

3. El conjunto $\mathcal{A}$ estará formado por dos acciones: $a_1 = izquierda$, $a_2 = derecha$.

4. Habrá 7 estados con recompensa. De entre esos estados, la recompensa del estado $T2$ será la mayor de todas. El vector de recompensas sobre los estados será el siguiente:
$$\mathcal{R} = [0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 1]^T$$

5. La política óptima será ir siempre a la derecha de manera que acabemos en el estado $T2$, consiguiendo el mayor retorno posible.

6. La matriz de transición que modele el entorno del problema será determinista. Es decir, de acuerdo a la figura 10.1, suponiendo que estamos en el estado $s = 6$ y que escogemos la acción $a_2 = derecha$ tendremos:
$$\begin{aligned} p(s' = 7 \mid s = 6, a = a_2) &= 1 \\ p(s' = 5 \mid s = 6, a = a_2) &= 0 \end{aligned} \tag{10.5}$$
Puesto que el problema tiene $|\mathcal{S}| = 13$ estados y $|\mathcal{A}| = 2$ acciones, la matriz de transiciones $\mathcal{P}$ será de dimensión 26×13. Dado que no aportaría información nueva sobre el problema más allá de la que se ha ejemplificado en (10.5), no se mostrará su representación.

Como puede haberse notado, ahora el estado desde el cual se comenzará será $s = 1$. El motivo de empezar en este estado será porque nos permitirá estudiar el tiempo que tarda en propagarse la función valor de extremo a extremo del MDP.

Por último, decir que el factor de descuento γ escogido para el cálculo del retorno fue $\gamma = 0{,}9$.

Calibración de parámetros

Para la calibración de los parámetros N_{epi}, α_D, α_{TD} y ϵ de BDA-MF, se tuvieron en cuenta las siguientes consideraciones respecto al algoritmo 10.6 planteado:

1. En lugar de detener la ejecución en función de la convergencia de la función valor, tal y como se detalló en el algoritmo 10.6, ahora fijaremos un número máximo de iteraciones posibles N_{iter}, de manera que si el algoritmo nunca llegase a converger por el motivo que fuera, no se quedase ejecutando de manera infinita. De este modo, el número total de episodios que se simulen será conocido e igual a $N_{epi}^{TOT} = N_{iter}N_{epi}$, y lo podremos controlar. En concreto, para todas las pruebas de calibración se escogió $N_{iter} = 500$, cantidad que como veremos está sobredimensionada con el objetivo de observar claramente la convergencia en las gráficas de resultados.

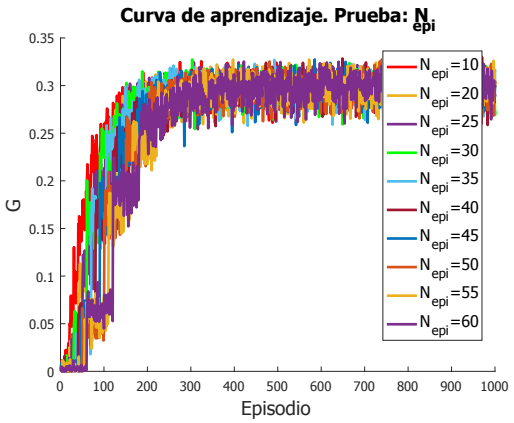
2. Se repitieron 50 experimentos independientes y se promediaron los resultados obtenidos en cada uno de ellos

Las gráficas de calibración obtenidas para cada parámetro fueron las mostradas en la figura 10.5. En vista de los resultados se deduce que:

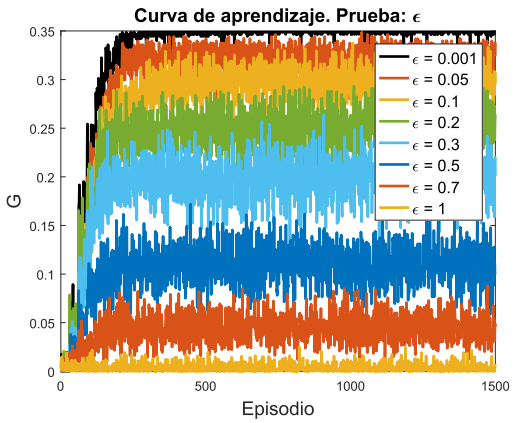
1. Cuanto menor es el número de episodios N_{epi}, más frecuentemente se actualizan los valores de v y d, de manera que antes se alcanza un valor de G estable; es decir, convergemos antes a la política óptima, tal y como se puede ver en la figura 10.5(a).

2. Cuanto menor es ϵ, menor es la exploración y por tanto menor es la varianza del retorno obtenido. No obstante, dado que se explora menos, se tarda más en converger. Esto queda reflejado si comparamos los casos $\epsilon = 0{,}001$ y $\epsilon = 0{,}3$, por ejemplo, de la gráfica 10.5(b): cuando $\epsilon = 0{,}3$, G se estabiliza en un número menor de episodios que cuando $\epsilon = 0{,}001$. En contraparte, su varianza es mucho mayor.

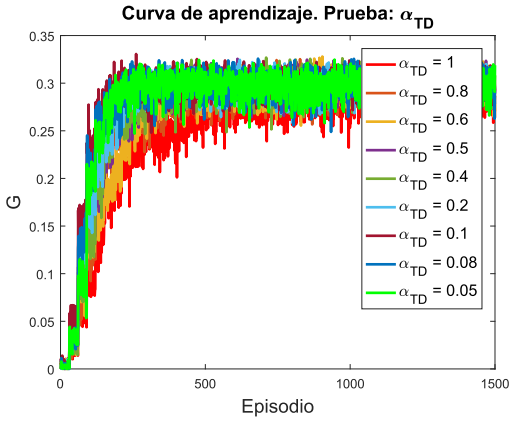
3. Cuanto menor es α_{TD}, antes se converge a un valor de G estable, tal y como se puede ver en la figura 10.5(c). De nuevo, esta idea puede parecer contraintuitiva pues en los métodos de gradiente, cuanto menor es el paso de aprendizaje, más lenta es la convergencia. Al igual que se comentó en la misma prueba pero con el problema de validación, se cree que esto pueda ser debido al acoplo que presentan las etapas de predicción y control: valores altos de α_{TD} provocarán que el ruido que presente la estimación de v sea tal que empeore la estimación de d y la política derivada de ella. Como resultado, se conseguiría una convergencia más lenta a la máxima recompensa total acumulada. En este caso, se cree que no aparece el punto



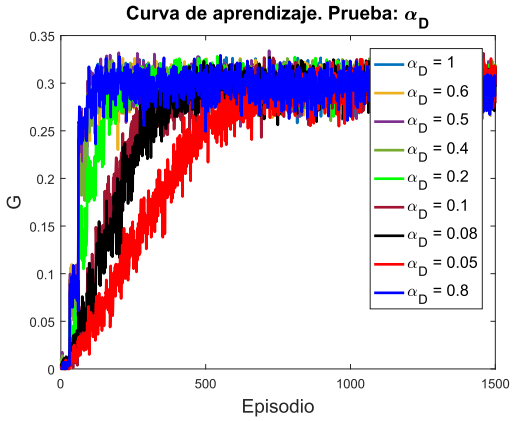
(a) Variación del número de episodios.



(b) Variación del parámetro de exploración.



(c) Variación de α_{TD}.



(d) Variación de α_D.

Figura 10.5: Calibración de los parámetros característicos del algoritmo BDA-MF para el problema *random walk* con matriz de transición determinista: evolución del retorno a lo largo de los episodios simulados.

de inflexión que se mencionó en el problema de validación porque para valores más pequeños de α_{TD}, la lentitud de convergencia pasa a predominar sobre la mejora que suponga en la disminución del ruido de estimación.

4. Cuanto mayor es α_D, antes se converge a un valor de G estable, tal y como se puede ver en la figura 10.5(d). Esta conclusión puede parecer a priori contradictoria al caso de las pruebas de validación, donde se concluyó justamente lo contrario para un problema similar, pero no es así. Como vemos, α_D multiplica a un término dependiente de la función valor $v(s)$, y esta depende de la recompensa. En el caso del problema de validación, la recompensa del estado terminal deseado era 150; en este caso, la recompensa es 1. Parece claro que el α_D se encargará de escalar la aproximación del gradiente con intención de que esta tenga un orden de magnitud

similar a la variable que se desea actualizar, en este caso d.

Se muestra a continuación una tabla resumen con los resultados de explotación tras la convergencia, para los mismos parámetros que se usaron en las pruebas de la figura 10.5:

Variación N_{epi}	G
$N_{epi} = 10$	0,3487
$N_{epi} = 20$	0,3487
$N_{epi} = 25$	0,3487
$N_{epi} = 30$	0,3487
$N_{epi} = 35$	0,3487
$N_{epi} = 40$	0,3487
$N_{epi} = 45$	0,3487
$N_{epi} = 50$	0,3487
$N_{epi} = 55$	0,3487
$N_{epi} = 60$	0,3487

(a)

Variación ϵ	G
$\epsilon = 0{,}001$	0,3487
$\epsilon = 0{,}05$	0,3487
$\epsilon = 0{,}1$	0,3487
$\epsilon = 0{,}15$	0,3487
$\epsilon = 0{,}2$	0,3487
$\epsilon = 0{,}3$	0,3487
$\epsilon = 0{,}4$	0,3487
$\epsilon = 0{,}5$	0,3487
$\epsilon = 0{,}7$	0,3487
$\epsilon = 1$	0,3487

(b)

Variación α_{TD}	G
$\alpha_{TD} = 1$	0,3487
$\alpha_{TD} = 0{,}8$	0,3487
$\alpha_{TD} = 0{,}6$	0,3487
$\alpha_{TD} = 0{,}5$	0,3487
$\alpha_{TD} = 0{,}4$	0,3487
$\alpha_{TD} = 0{,}2$	0,3487
$\alpha_{TD} = 0{,}1$	0,3487
$\alpha_{TD} = 0{,}08$	0,3487
$\alpha_{TD} = 0{,}05$	0,3487

(c)

Variación α_D	G
$\alpha_{TD} = 1$	0,3487
$\alpha_{TD} = 0{,}8$	0,3487
$\alpha_{TD} = 0{,}6$	0,3487
$\alpha_{TD} = 0{,}5$	0,3487
$\alpha_{TD} = 0{,}4$	0,3487
$\alpha_{TD} = 0{,}2$	0,3487
$\alpha_{TD} = 0{,}1$	0,3487
$\alpha_{TD} = 0{,}08$	0,3487
$\alpha_{TD} = 0{,}05$	0,3487

(d)

Cuadro 10.1: Calibración de los parámetros característicos del algoritmo BDA-MF para el problema *random walk* con matriz de transición determinista: retorno obtenido al evaluar la política estable obtenida durante el aprendizaje.

Como vemos, en todos los casos se converge al mismo valor. Para justificar el valor del retorno obtenido para las diferentes políticas que se hayan derivado en cada prueba, resultará conveniente acudir a la definición de G:

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

En base a su expresión se deduce que el valor teórico esperado si se siguiese la política óptima (ir hacia la derecha siempre) sería, empezando desde el estado $s = 1$:

$$G = \gamma^{10} 1 = 0{,}9^{10} = 0{,}3487 \tag{10.6}$$

siendo 10 el mínimo número de transiciones hacia la derecha necesarias para llegar a $T2$. De ello se deduce que todas las pruebas han sido capaces de encontrar la política óptima en más o menos episodios.

Presentación de resultados óptimos y comparación con SARSA y Q-learning

A partir de los resultados anteriores, se llegó a la conclusión de que los parámetros óptimos para el problema que se está tratando son:

N_{epi}	α_D	α_{TD}	ϵ
10	0,8	0,05	0,5

Cuadro 10.2: Parámetros óptimos del algoritmo BDA-MF, para el problema *random walk* con matriz de transición determinista.

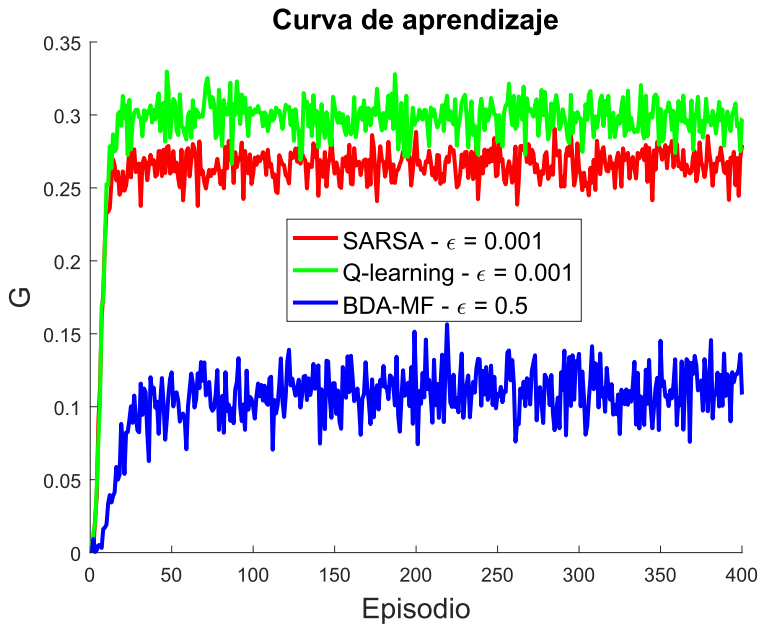
Dado que el objetivo de este trabajo es el de desarrollar un nuevo algoritmo de aprendizaje por refuerzo que sea competitivo frente a los ya existentes, se realizó una comparación con dos de los algoritmos de uso más extendido a día de hoy: SARSA y Q-learning. Al igual que para nuestro algoritmo, se llevó a cabo un proceso de calibración de los parámetros de estos dos métodos con el objetivo de poder comparar la versión óptima de cada algoritmo aplicado al problema *random walk* bajo estudio.

Como hemos podido ver en la figura 10.5(b), el punto al que converge G dependerá fuertemente del parámetro ϵ de aprendizaje. Esto va a suponer un problema de cara a comparar los distintos algoritmos, pues cada uno alcanzará su comportamiento óptimo para diferentes valores de ϵ; es decir, el problema que surge es que no habrá manera de comparar el número de episodios en el que converge cada algoritmo porque cada uno convergerá a un valor diferente.

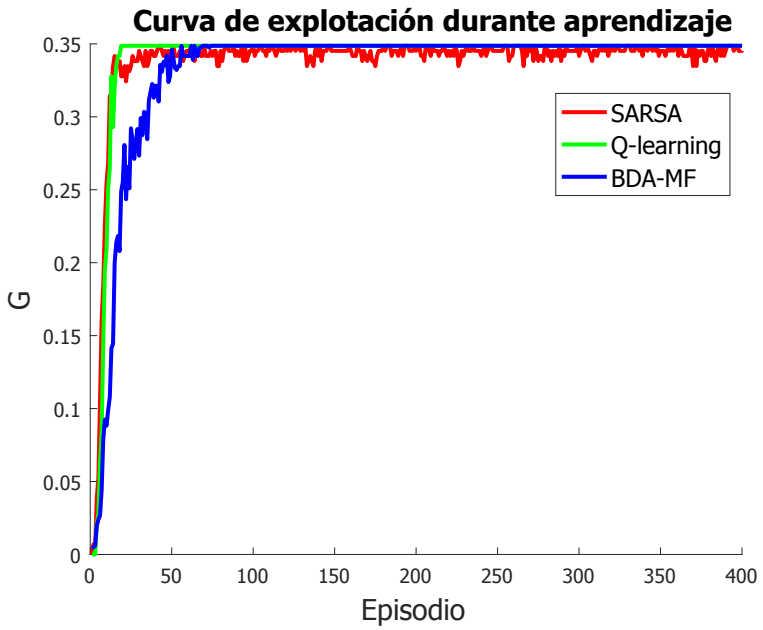
A efectos de resolver este problema y poder comparar cómo de buena es la convergencia de G en cada algoritmo de manera independiente al parámetro ϵ con el que se aprende, será necesario definir una nueva métrica: la curva de explotación durante el aprendizaje. Esta curva permitirá ver la evolución del algoritmo a medida que vamos aprendiendo, pero quitando el efecto exploratorio y realizando únicamente la explotación del conocimiento disponible al final de cada episodio[5]. En otras palabras, al final de cada episodio se correrá el algoritmo con la política disponible π_{d_k} y $\epsilon = 0$. Para comprobar la utilidad de esta nueva gráfica a la hora de analizar los resultados, se muestran en la figura 10.6 los resultados óptimos de la curva de aprendizaje y la curva de explotación

[5]Al igual que se definió para la curva de explotación tras convergencia, en este proceso únicamente se evalúa la política, sin aprender nada nuevo.

durante el aprendizaje, para cada algoritmo. De nuevo, se realizaron 50 experimentos independientes y se promediaron los resultados.



(a) Evolución durante el aprendizaje.



(b) Evaluación del aprendizaje tras cada episodio.

Figura 10.6: Comparativa de los algoritmos BDA-MF, SARSA y Q-learning para el problema *random walk* con matriz de transición determinista. En (*a*) se muestra la curva de aprendizaje y en (*b*) la curva de explotación del aprendizaje realizado en cada episodio, ambas con los parámetros óptimos de cada algoritmo.

De acuerdo a la gráfica 10.6 (b), ahora sí va a resultar posible decir qué algoritmo converge más rápido. La tabla 10.3 recoge, a modo resumen, el episodio en que converge cada algoritmo y el valor de G al que se converge cuando dejamos de explorar y se sigue la política que ha provocado la convergencia.

	Episodios hasta convergencia	G tras convergencia ($\epsilon = 0$)
SARSA	35	0,3417
Q-learning	20	0,3487
BDA-MF	75	0,3487

Cuadro 10.3: Número de episodios necesarios hasta converger a una solución estable (columna central), y retorno obtenido de la explotación de la política aprendida tras la convergencia (columna derecha), para el problema *random walk* con matriz de transición determinista.

Comparando la columna G con su valor óptimo esperado, (10.6), podemos apreciar que se ha conseguido llegar a la política óptima con los tres algoritmos.

En base a estos resultados, se concluye que nuestro algoritmo no consigue llegar a ser tan bueno como SARSA y Q-learning en este problema.

Antes de pasar al siguiente caso de estudio, se aclarará un hecho que puede haber llamado la atención en la figura 10.6: como se vio en la sección 5.2.2, la política objetivo en SARSA es $\epsilon - greedy$. Es por ello que su curva de explotación durante el aprendizaje, mostrada en rojo en la figura 10.6 (b), nunca llega a converger al valor exacto sino que mantiene pequeñas fluctuaciones.

10.3.3. Problema bajo estudio 2: random walk (transiciones aleatorias)

Presentación del problema

El problema que ahora se va a evaluar es el mismo que se empleó en las pruebas de validación de BDA-MF, de manera que se dará por presentado.

Calibración de parámetros

El proceso a seguir en la calibración es el mismo que se siguió en la sección anterior, pero adaptado al nuevo problema. En consecuencia, mostrar estas gráficas otra vez no aportaría información nueva más allá de lo ya explicado sobre las figuras 10.4 o 10.5,

Por este motivo, de aquí a lo que queda de capítulo no se mostrarán más gráficas del proceso de calibración, y se pasará directamente a mostrar los resultados óptimos.

Presentación de resultados óptimos y comparación con SARSA y Q-learning

Los parámetros óptimos del algoritmo BDA-MF para el problema que se está tratando son:

N_{epi}	α_D	α_{TD}	ϵ
30	0,4	0,2	0,4

Cuadro 10.4: Parámetros óptimos del algoritmo BDA-MF, para el problema *random walk* con matriz de transición aleatoria.

Los resultados óptimos de la curva de aprendizaje y la curva de explotación durante el aprendizaje para SARSA, Q-learning y BDA-MF se muestran en la figura 10.7.

Antes de continuar, resaltar que el hecho de que se eligiese $\epsilon = 0{,}4$ en los tres algoritmos se debe a que coincidió ser el valor óptimo para los tres.

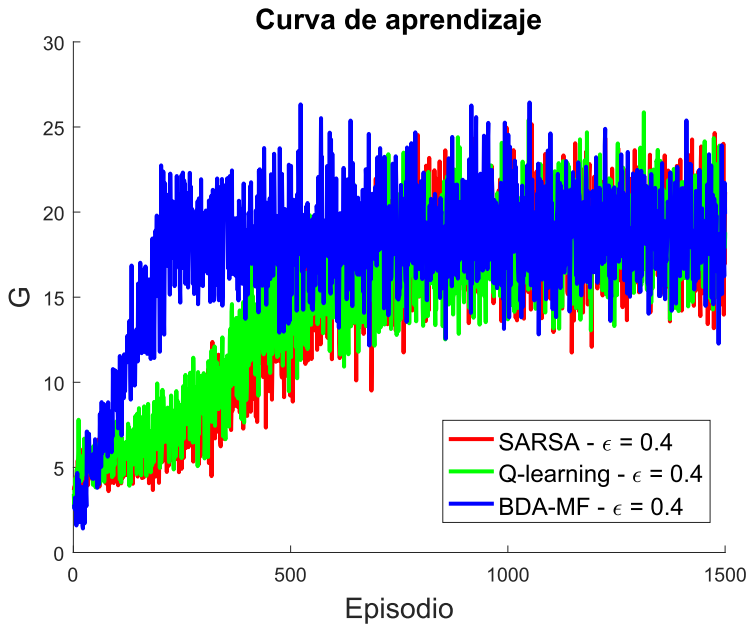
Comparando las gráficas 10.6 y 10.7, podemos notar que en esta última los resultados son más ruidosos. Esto se debe a que ahora la matriz de transición del problema provocará que no siempre se lleve a cabo la acción deseada, tengamos exploración o no. Por tanto, si iniciamos el episodio en el estado $s = 1$, aunque conozcamos la política óptima de ir siempre a la derecha, algunas veces iremos a la izquierda acabando el episodio con un retorno $G = r(s = T1) = 2$. Dado que estas gráficas son el resultado de un promedio de 50 experimentos independientes, en el mismo episodio de diferentes experimentos podremos acabar con la recompensa del estado terminal $T1$ o con la recompensa obtenida de acabar correctamente en $T2$, teniendo por tanto el comportamiento fluctuante observado en la figura 10.7 (b) una vez se ha convergido.

La tabla 10.5 recoge, a modo resumen, el episodio en que converge cada algoritmo y el valor de G al que se converge cuando explotamos la política aprendida.

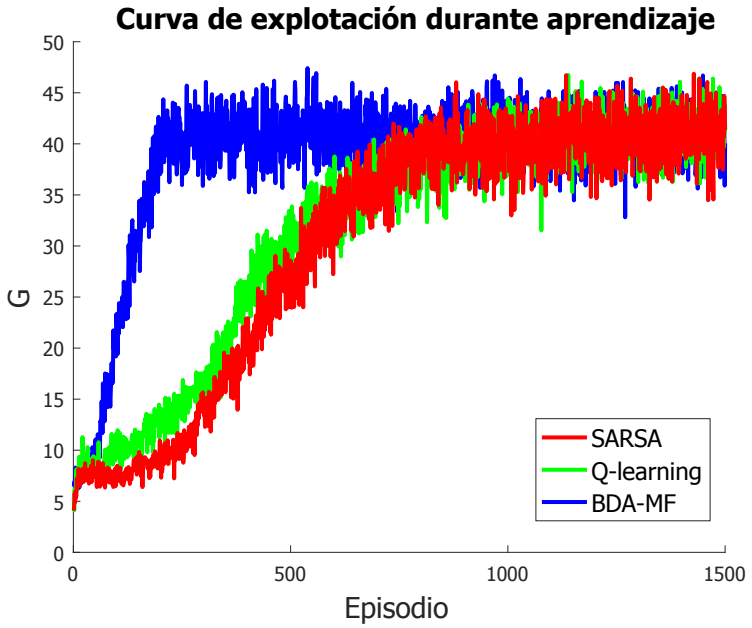
	Episodios hasta convergencia	G tras convergencia ($\epsilon = 0$)
SARSA	770	40,4532
Q-learning	820	41,1833
BDA-MF	200	41,1163

Cuadro 10.5: Número de episodios necesarios hasta converger a una solución estable (columna central), y retorno obtenido de la explotación de la política aprendida tras la convergencia (columna derecha), para el problema *random walk* con matriz de transición aleatoria.

Como vemos, la situación ha cambiado sustancialmente respecto al caso en el que la matriz de transición era determinista: ahora el algoritmo BDA-MF desarrollado converge más rápido que los ya conocidos SARSA y Q-learning. Por tanto, parece que nuestro algoritmo será capaz de enfrentarse mejor que los otros a entornos que presentan cierta



(a) Evolución durante el aprendizaje.



(b) Evaluación del aprendizaje tras cada episodio.

Figura 10.7: Comparativa de los algoritmos BDA-MF, SARSA y Q-learning para el problema *random walk* con matriz de transición aleatoria. En (a) se muestra la curva de aprendizaje y en (b) la curva de explotación del aprendizaje realizado en cada episodio, ambas con los parámetros óptimos de cada algoritmo.

incertidumbre en las transiciones.

Con el objetivo de poder sacar conclusiones más sólidas, a continuación se estudiará

otro problema distinto al *random walk*.

10.3.4. Problema bajo estudio 3: paseo por el acantilado (transiciones deterministas)

Presentación del problema

Ahora vamos a trabajar con otro problema muy común en el aprendizaje por refuerzo: el paseo por el acantilado (*cliff walking problem*). Se trata de una tarea episódica que cuenta con un estado inicial y un estado absorbente final que será nuestro objetivo. Para moverse de un estado a otro contiguo, el agente puede elegir de entre cuatro posibles acciones: arriba, abajo, izquierda y derecha. La recompensa es -1 en todas las transiciones excepto en aquellas que hacen que nos caigamos al acantilado o que alcancemos el estado final objetivo. Caer en la región del acantilado supone una recompensa de -100 puntos y volver de vuelta al estado inicial. Llegar al estado final objetivo, por convención, tiene una recompensa asociada de 0 puntos. Cuando este estado final se alcanza, el agente es también mandado de vuelta al estado inicial.

Si la acción que tomamos implica chocarse contra una pared, nos quedaremos en el mismo estado y se obtendrá la recompensa típica de haber realizado una transición, es decir, -1.

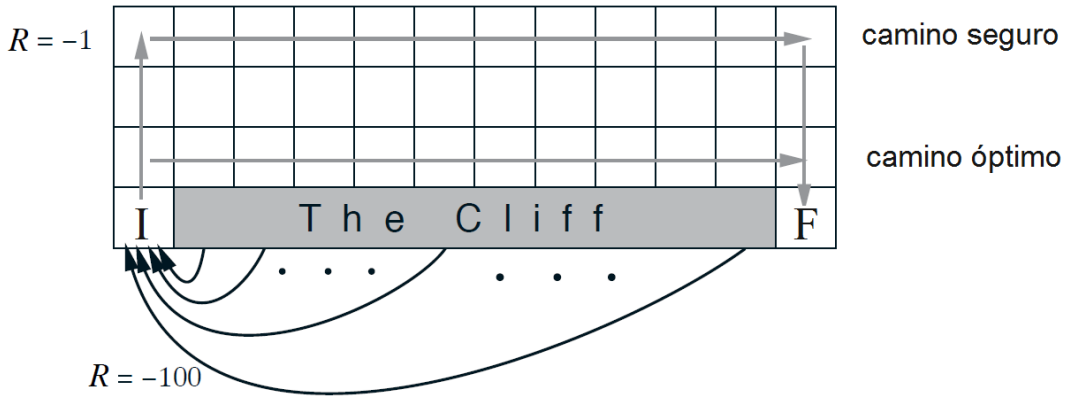


Figura 10.8: Tablero que modela el problema del paseo por el acantilado [1].

En la figura 10.8, vemos que el problema puede ser representado mediante una malla de 4×12 casillas, lo que hace un total de $|\mathcal{S}| = 48$ estados. De estos 48 estados, 11 serán terminales, siendo uno de ellos, el estado final objetivo y los otros 10 la región del acantilado. Resulta sencillo observar que, de nuevo, podremos modelar el problema como un MDP, sólo que ahora el conjunto de acciones será $|\mathcal{A}| = 4$, $\mathcal{A} = \{a_1, a_2, a_3, a_4\} =$ {izquierda, arriba, derecha, abajo}.

Como consideraciones adicionales, añadir que la matriz de transición que modele el

entorno del problema será determinista y que el estado inicial en todas las pruebas será la esquina inferior izquierda, que se corresponde con el estado marcado con una I (inicio) en la figura 10.8. Del mismo modo, el estado final objetivo será siempre la esquina inferior derecha, marcada con la letra F (final).

Por último, decir que el factor de descuento γ escogido para el cálculo del retorno fue $\gamma = 0{,}99$.

Presentación de resultados óptimos y comparación con SARSA y Q-learning

Los parámetros óptimos del algoritmo BDA-MF para el problema que se está tratando son:

N_{epi}	α_D	α_{TD}	ϵ
20	0,1	0,08	0,1

Cuadro 10.6: Parámetros óptimos del algoritmo BDA-MF, para el problema *paseo por el acantilado* con matriz de transición determinista.

Los resultados óptimos de la curva de aprendizaje y la curva de explotación durante el aprendizaje para SARSA, Q-learning y BDA-MF se muestran en la figura 10.9.

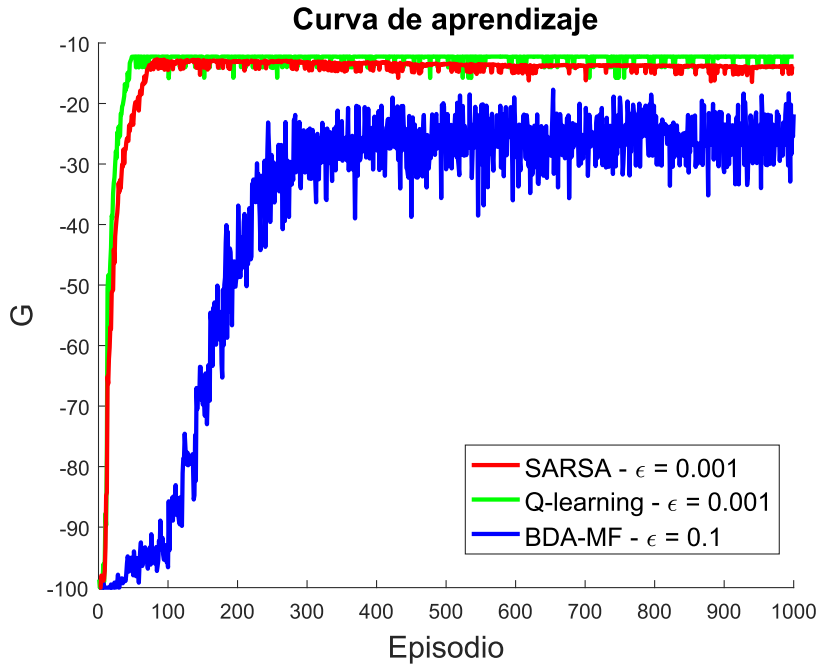
La tabla 10.7 recoge, a modo resumen, el episodio en que converge cada algoritmo y el valor de G al que se converge cuando explotamos la política aprendida.

	Episodios hasta convergencia	G tras convergencia ($\epsilon = 0$)
SARSA	130	$-12{,}2352$
Q-learning	50	$-11{,}3615$
BDA-MF	275	$-15{,}119$

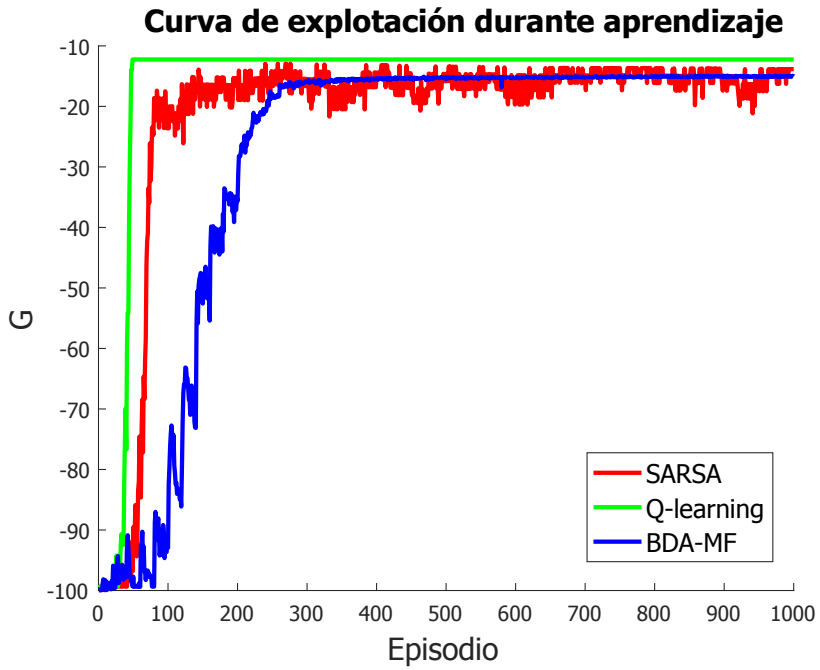
Cuadro 10.7: Número de episodios necesarios hasta converger a una solución estable (columna central), y recompensa total acumulada obtenida de la explotación de la política aprendida tras la convergencia (columna derecha), para el problema *paseo por el acantilado* con matriz de transición determinista.

Antes de pasar a analizar los resultados de convergencia, se va a calcular el valor del retorno que esperamos obtener. En el problema del paseo por el acantilado, típicamente se distinguen 3 soluciones, y para cada una de ellas habrá un retorno asociado:

1. Camino óptimo: en la figura 10.8 se puede apreciar que el camino óptimo es aquel que va pegado al acantilado, pues el que maximiza la recompensa. Para llegar desde el estado inicial al final por este camino, habrá que realizar 13 transiciones de las cuales 12 tienen una recompensa asociada de -1 puntos, y una, la última, tiene una



(a) Evolución durante el aprendizaje.



(b) Evaluación del aprendizaje tras cada episodio.

Figura 10.9: Comparativa de los algoritmos BDA-MF, SARSA y Q-learning para el problema *paseo por el acantilado* con matriz de transición determinista. En (*a*) se muestra la curva de aprendizaje y en (*b*) la curva de explotación del aprendizaje realizado en cada episodio, ambas con los parámetros óptimos de cada algoritmo.

recompensa asociada de 0 puntos. De este modo, definiendo los vectores:

$$\Gamma = [\gamma^0, \gamma^1, \gamma^2, \gamma^3, \gamma^4, \gamma^5, \gamma^6, \gamma^7, \gamma^8, \gamma^9, \gamma^{10}, \gamma^{11}, \gamma^{12}, \gamma^{13}]^T$$
$$R_{camino\,ópt} = [-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 0]^T$$

podemos expresar G como:

$$G^{ópt} = \Gamma^T R_{camino\,ópt} = -11{,}3615$$

2. Camino seguro: en la figura 10.8 se puede apreciar que el camino seguro es aquel que va lo más alejado posible del acantilado, bordeando el tablero. Este camino será el que presente menor riesgo de caer al acantilad. Para llegar desde el estado inicial al final por este camino, habrá que realizar 17 transiciones de las cuales 16 tienen una recompensa asociada de -1 puntos, y una, la última, tiene una recompensa asociada de 0 puntos. Siguiendo el mismo proceso que en el caso anterior, se llega a:

$$G^{seguro} = \Gamma^T R_{camino\,seguro} = -14{,}8542$$

3. Camino conservador: en la figura 10.8, el camino conservador sería el del medio, pues no arriesga tanto como el óptimo ni genera tan poco beneficio como el seguro. Es decir, se trata de un compromiso entre las dos opciones anteriores. Para llegar desde el estado inicial al final por este camino, habrá que realizar 15 transiciones de las cuales 14 tienen una recompensa asociada de -1 puntos, y una, la última, tiene una recompensa asociada de 0 puntos. De nuevo, la recompensa total acumulada esperada si se sigue este camino se podrá calcular como:

$$G^{compromiso} = \Gamma^T R_{camino\,compromiso} = -13{,}1254$$

Con estos valores y la tabla 10.7, parece claro sacar una relación entre cada algoritmo y cada camino existente hasta el estado final objetivo. Como vemos, Q-learning aprenderá sin problemas la política óptima, obteniéndose un retorno de $-11{,}3615$. SARSA sin embargo aprenderá, en unos experimentos la política óptima y en otros la conservadora. De este modo, al hacer el promedio de los 50 experimentos simulados se obtiene un valor intermedio: $-12{,}2352$. Esto se debe a que su política objetivo es $\epsilon-greedy$, de manera que la política final es ligeramente oscilante, aunque tendiendo más a la óptima tal y como se pudo ver al analizar la variable que acumula los valores de G tras cada experimento. Por último, BDA-MF aprenderá la política que lleva por el camino seguro, e incluso a veces una peor. De ahí que el valor de G obtenido sea ligeramente más negativo que G^{seguro}.

En base a los resultados de la figura 10.9 y de la tabla 10.7, se concluye al igual

que se hizo en el problema bajo estudio 1 que para este nuevo problema, el algoritmo BDA-MF desarrollado funcionará peor que SARSA y que Q-learning, tanto en términos de velocidad de convergencia como de la recompensa total acumulada obtenida.

10.3.5. Problema bajo estudio 4: paseo por el acantilado (transiciones aleatorias)

Presentación del problema

Por último, se pasará a probar el problema del paseo del acantilado pero cuando la matriz de transición tiene cierto carácter aleatorio. Más concretamente, la probabilidad de tomar la acción escogida ahora será de 0.7, y la probabilidad de que se lleve a cabo otra de las tres acciones contrarias será de 0.1 cada una. Es decir, si escogemos ir hacia arriba, finalmente iremos hacia arriba con probabilidad 0.7, a la derecha con probabilidad 0.1, a la izquierda con probabilidad 0.1 y abajo con probabilidad 0.1, sumando un total de 1.

Con este grado de incertidumbre en la toma de acciones, la nueva política óptima pasa a ser el camino del medio de la figura 10.8.

Presentación de resultados óptimos y comparación con SARSA y Q-learning

Los parámetros óptimos del algoritmo BDA-MF para el problema que se está tratando son:

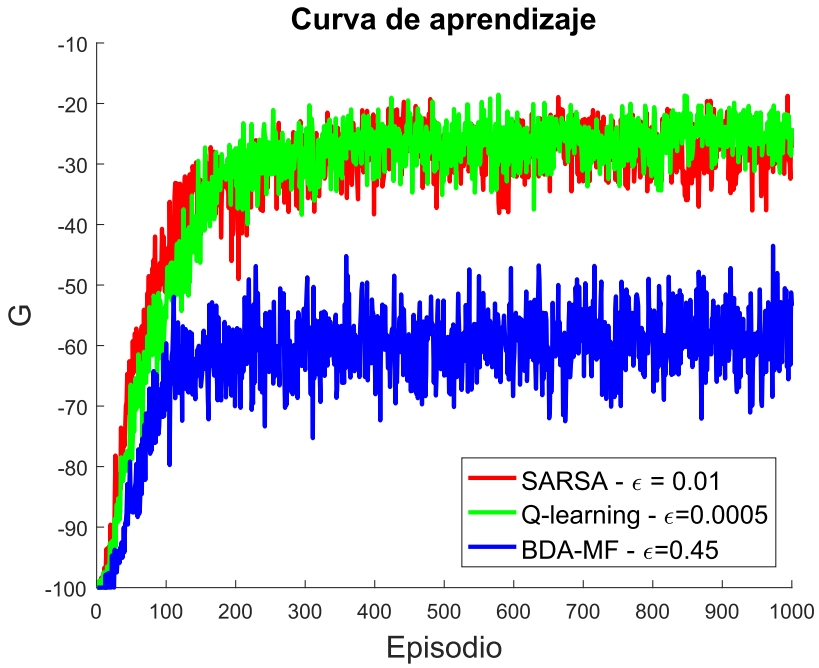
N_{epi}	α_D	α_{TD}	ϵ
5	3	0,2	0,45

Cuadro 10.8: Parámetros óptimos del algoritmo BDA-MF, para el problema *paseo por el acantilado* con matriz de transición determinista.

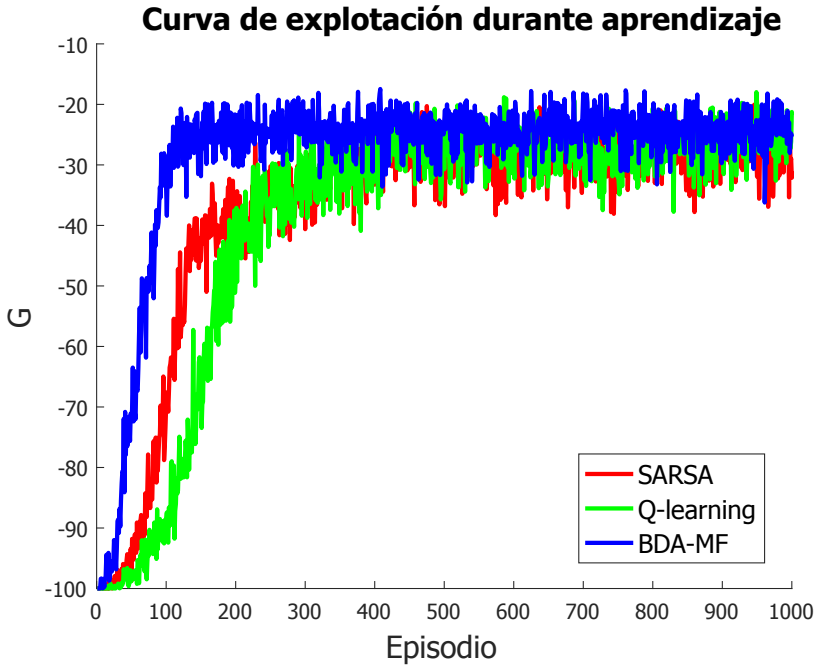
Aunque el número de episodios N_{epi} sea pequeño, recordar que el número total de episodios simulados viene dado por $N_{epi}^{TOT} = N_{iter}N_{epi}$, habiéndose escogido $N_{iter} = 500$.

Los resultados óptimos de la curva de aprendizaje y la curva de explotación durante el aprendizaje para SARSA, Q-learning y BDA-MF se muestran en la figura 10.10.

La tabla 10.9 recoge, a modo resumen, el episodio en que converge cada algoritmo y el valor de G al que se converge cuando explotamos la política aprendida.



(a) Evolución durante el aprendizaje.



(b) Evaluación del aprendizaje tras cada episodio.

Figura 10.10: Comparativa de los algoritmos BDA-MF, SARSA y Q-learning para el problema *paseo por el acantilado* con matriz de transición aleatoria. En (a) se muestra la curva de aprendizaje y en (b) la curva de explotación del aprendizaje realizado en cada episodio, ambas con los parámetros óptimos de cada algoritmo.

	Episodios hasta convergencia	G tras convergencia ($\epsilon = 0$)
SARSA	350	$-31{,}5419$
Q-learning	380	$-26{,}2466$
BDA-MF	120	$-24{,}7039$

Cuadro 10.9: Número de episodios necesarios hasta converger a una solución estable (columna central), y retorno obtenido de la explotación de la política aprendida tras la convergencia (columna derecha), para el problema *paseo por el acantilado* con matriz de transición aleatoria.

Al igual que ocurrió con el problema *random walk*, en el problema del paseo por el acantilado la situación mejora cuando la matriz de transición es aleatoria: ahora el algoritmo BDA-MF converge más rápido y con un retorno mayor que SARSA y Q-learning.

10.4. Análisis y discusión de los resultados

De cara a facilitar la comparativa, se recogen en la tabla 10.10 los resultados de explotación de las pruebas anteriores:

	Episodios hasta convergencia	G tras convergencia ($\epsilon = 0$)
SARSA	35	0,3417
Q-learning	20	0,3487
BDA-MF	75	0,3487

(a) Random walk (transiciones deterministas)

	Episodios hasta convergencia	G tras convergencia ($\epsilon = 0$)
SARSA	770	40,4532
Q-learning	820	41,1833
BDA-MF	200	41,1163

(b) Random walk (transiciones aleatorias)

	Episodios hasta convergencia	G tras convergencia ($\epsilon = 0$)
SARSA	130	$-12{,}2352$
Q-learning	50	$-11{,}3615$
BDA-MF	275	$-15{,}119$

(c) Acantilado (transiciones deterministas)

	Episodios hasta convergencia	G tras convergencia ($\epsilon = 0$)
SARSA	350	$-31{,}5419$
Q-learning	380	$-26{,}2466$
BDA-MF	120	$-24{,}7039$

(d) Acantilado (transiciones aleatorias)

Cuadro 10.10: Resultados de explotación para los cuatro problemas bajo estudio evaluados.

Como se ha podido comprobar con los problemas anteriores, BDA-MF ha estado a la altura de algoritmos de tan extendido uso como son SARSA y Q-learning. Aunque con

el mismo objetivo, la idea que subyace a estos algoritmos es muy diferente: mientras que SARSA y Q-learning constituyen métodos que se conocen como basados en la función valor y trabajan explorando el espacio de funciones valor para posteriormente extraer la política, BDA-MF cambia el enfoque y pasa a buscar directamente en el espacio de políticas mediante un ascenso por gradiente.

En vista de los resultados que se han ido mostrando, se puede afirmar que la búsqueda en el espacio de políticas que realiza BDA-MF requiere de un alto nivel exploración. De este modo, cuando la matriz de transición es determinista, se hace necesario un valor de ϵ alto (si lo comparamos con el que toman SARSA o Q-learning) para asegurar la convergencia. Y es precisamente el hecho de que exista una gran necesidad de exploración lo que hace que su convergencia sea más lenta que SARSA o Q-learning, ya que cuanto mayor sea ϵ, menor será la probabilidad de que la siguiente acción que tomemos siga la política que se está construyendo y que apunta a ser la óptima. En resumen, lo que a priori es una ventaja porque permite explorar el espacio de políticas y converger, más tarde supone una desventaja ya que no da pie a reforzar las acciones que conducen al máximo retorno. Además, no habrá que perder de vista que SARSA y Q-learning emplean políticas de tipo greedy o ϵ-greedy, lo cual favorece la convergencia ya que siempre se toma la acción que maximiza el retorno esperado.

Las conclusiones que se acaban de extraer quedan contrastadas cuando se analizan los mismos problemas, pero cuando la matriz de transición tiene cierto carácter aleatorio. En estos casos, aunque el agente no lo elija, existe un nivel de exploración que viene impuesto por las transiciones del medio: el hecho de que la acción llevada a cabo finalmente por el entorno no sea siempre la que escogió el agente, se traduce en exploración. De este modo, la velocidad de convergencia de BDA-MF resulta ser considerablemente mejor que la de SARSA y Q-learning. Lo que ahora es una ventaja para el algoritmo que en este trabajo se ha desarrollado, pasa a ser una desventaja para SARSA y Q-learning, que aunque también basen su funcionamiento en la exploración del espacio de estados y acciones, un nivel tan alto de exploración parece jugar en su contra.

En resumen, el algoritmo BDA-MF de búsqueda en el espacio de políticas ha resultado ser competitivo. No obstante, su funcionamiento mejora sustancialmente cuanto mayor es la exploración del espacio de estados y acciones, lo cual supone una desventaja en problemas cuyo entorno se modela de manera determinista por los motivos explicados anteriormente. Si por el contrario el medio presenta algún carácter estocástico en las transiciones, BDA-MF sería una opción a considerar a nivel de implementación.

Capítulo 11

Extensión del algoritmo BDA-MF a problemas de gran escala

En el capítulo 10 se ha presentado el algoritmo Bellman-ascenso dual libre de modelo (BDA-MF), el cual permitía resolver problemas de aprendizaje por refuerzo de pequeña escala. Lo que ahora se propone es la extensión de este algoritmo al caso en que los conjuntos de estados y acciones sean muy grandes o continuos. Para ello, se hará uso de aproximaciones de la función valor de la misma forma que se explicaron en el capítulo 6:

$$v^{\pi}(s,\omega) = \bar{\phi}(s)^{T}\omega \tag{11.1}$$

$$q(s,a_m,\theta) = \phi(s,a_m)^{T}\theta \tag{11.2}$$

es decir, aproximaciones lineales. En concreto, para el algoritmo BDA-MF se empleará la descrita por (11.1), pues se trabajará con la función valor de estados.

Para evaluar el comportamiento del nuevo algoritmo derivado, se probará en dos problemas típicos: el primero de ellos será el problema del paseo por la cadena (*chain walk*). Aunque el espacio de estados y de acciones de este problema sea pequeño y discreto, servirá como prueba de concepto para verificar que BDA-MF con aproximación lineal de la función valor funciona. Tras ello, se pasará a probar en un problema de mayor complejidad donde el espacio de estados es continuo y el de acciones se mantiene pequeño y discreto. Este problema se conoce en la literatura como el problema del coche de montaña (*mountain car* [1]).

Por último, recalcar que nos centraremos únicamente en la versión libre de modelo por ser la de mayor utilidad en la práctica.

11.1. Algoritmo Bellman-ascenso dual con aproximación lineal de funciones

Para derivar este nuevo algoritmo, habrá que partir de la formulación del problema primal (9.11), pero esta vez sustituyendo v por su versión aproximada. Teniendo en cuenta que la forma vectorial de (11.1) es la siguiente:

$$v^{\pi} = \overline{\Phi}\omega$$

donde $\overline{\Phi}$ será la matriz de dimensión $S \times N$ de funciones base dada por (6.12), el problema primal correspondiente será:

$$\begin{aligned} \underset{v}{\text{minimize}} \quad & \mu^T \overline{\Phi}\omega \\ \text{subject to} \quad & \Xi^T \overline{\Phi}\omega \geq \mathcal{R} + \gamma \mathcal{P} \overline{\Phi}\omega \end{aligned} \tag{11.3}$$

con Lagrangiano:

$$\mathcal{L}(v, d) = \mu^T \overline{\Phi}\omega + d^T \left(\mathcal{R} + \gamma \mathcal{P} \overline{\Phi}\omega - \Xi^T \overline{\Phi}\omega \right), \qquad d \geq 0$$

Expresado en formal escalar:

$$\mathcal{L}(v, d) = \sum_{s \in \mathcal{S}} \mu(s) \bar{\phi}(s)^T \omega + \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} d(s, a) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \bar{\phi}(s')^T \omega - \bar{\phi}(s)^T \omega \right)$$

siendo $d(s, a) \geq 0$, $\forall s \in \mathcal{S}$ y $\forall a \in \mathcal{A}$, y $\bar{\phi}$ el vector de características de longitud N que representa el estado s, dado por (6.3). Lo siguiente que se deberá hacer es adaptar la formulación Bellman-ascenso dual al caso aproximado:

$$\begin{aligned} v_{k+1} &\coloneqq v^{\pi_{d_k}} \in \mathcal{F}^{PR} \\ d_{k+1} &\coloneqq \left[d_k + \alpha \nabla_d \mathcal{L}(v_{k+1}, d) \right]_+ \end{aligned} \tag{11.4}$$

donde $\mathcal{F}^{PR}$ es la región factible del problema primal. Para ello, nuevamente distinguiremos las etapas de predicción y de control.

11.1.1. Predicción

La actualización de la variable primal de (11.4), como se vio en capítulos anteriores, se corresponderá con la etapa de predicción. Una vez más, para obtener la función valor para la política dada por la variable dual d_k, se resolverá la ecuación de Bellman para la función valor de estados. Dado que ahora se están usando aproximaciones lineales, la

evaluación de la política consistirá en encontrar el parámetro ω óptimo que resuelva la ecuación de Bellman proyectada, tal y como se explicó en la subsección 6.2.1. Recordando las expresiones (6.25)–(6.27) y adaptándolas para el caso de la función valor de estados, se llega a que ω^* se puede obtener de la siguiente manera:

$$\omega^* = \left(\overline{\Gamma} - \gamma\overline{\Lambda}\right)^{-1} \overline{z} \tag{11.5}$$

definiéndose:

$$\overline{\Gamma} \triangleq \overline{\Phi}^\top \overline{D\Phi} \tag{11.6}$$

$$\overline{\Lambda} \triangleq \overline{\Phi}^\top \overline{D\Pi\mathcal{P}\Phi} \tag{11.7}$$

$$\overline{z} \triangleq \overline{\Phi}^\top \overline{D\Pi\mathcal{R}} \tag{11.8}$$

Dado que se está considerando el caso en que se desconoce el modelo, habrá que aproximar los términos (11.6)–(11.8) a partir de muestras y finalmente resolver (11.5) con los términos aproximados. Para ello, se empleará el método LSTD que, como se pudo ver, implementa esta solución basada en muestras. La versión del algoritmo 6.1 adaptada para las funciones valor de estados se detalla en el algoritmo 11.1, y será por tanto el que se use en la etapa de predicción.

Algoritmo 11.1 Algoritmo LSTD para la evaluación de la política sobre las funciones valor de estados.

Entrada: π, la política a ser evaluada. Funciones base $\bar{\phi}$.
Salida: ω, el parámetro estimado para la aproximación lineal de v^π.

Inicializar $\overline{\Gamma} = 0_{N\times N}$, $\overline{\Lambda} = 0_{N\times N}$, $\overline{z} = 0_N$
repetir(para cada episodio)
    Inicializar s y observar $\bar{\phi}(s)$
    repetir(para cada paso en el episodio)
        Tomar la acción a y observar $r, s', \bar{\phi}(s')$
        Escoger la acción $a' \sim \pi(\cdot|s)$
        $\overline{\Gamma} \leftarrow \overline{\Gamma} + \bar{\phi}(s)\bar{\phi}(s)^T$
        $\overline{\Lambda} \leftarrow \overline{\Lambda} + \bar{\phi}(s)\bar{\phi}(s')^T$
        $\overline{z} \leftarrow \overline{z} + \bar{\phi}(s)r$
        $s \leftarrow s'$
        $a \leftarrow a'$
    hasta que s sea terminal
hasta que no podamos correr más episodios
Calcular: $\omega = \left(\overline{\Gamma} - \gamma\overline{\Lambda}\right)^{-1} \overline{z}$
devolver $\hat{v}_\omega = \overline{\Phi}\omega$

11.1.2. Control

La actualización de la variable dual de (11.4) se corresponderá con la etapa de control. Para poder presentar el algoritmo de control, se define a continuación el gradiente del Lagrangiano respecto de la variable dual:

$$\begin{aligned}\nabla_d \mathcal{L}(v_{k+1}, d) = &\quad \nabla_d \left(\mu^T v_{k+1} + d^T \left(\mathcal{R} + \gamma \mathcal{P} v_{k+1} - \Xi^T v_{k+1}\right)\right) \\ = &\quad \nabla_d \left(\mu^T \overline{\Phi} \omega_{k+1} + d^T \left(\mathcal{R} + \gamma \mathcal{P} \overline{\Phi} \omega_{k+1} - \Xi^T \overline{\Phi} \omega_{k+1}\right)\right) \\ = &\quad \mathcal{R} + \gamma \mathcal{P} \overline{\Phi} \omega_{k+1} - \Xi^T \overline{\Phi} \omega_{k+1} \\ = &\quad \nabla_d \mathcal{L}(\omega_{k+1}, d)\end{aligned}$$

De este modo, se podrá reescribir (11.4) en términos del vector de parámetros:

$$\begin{aligned}\omega_{k+1} &\coloneqq \omega^{\pi_{d_k}} \in \mathcal{F}^{PR} \\ d_{k+1} &\coloneqq \left[d_k + \alpha \nabla_d \mathcal{L}(\omega_{k+1}, d)\right]_+\end{aligned} \tag{11.9}$$

y sustituyendo el gradiente:

$$\begin{aligned}\omega_{k+1} &\coloneqq \omega^{\pi_{d_k}} \in \mathcal{F}^{PR} \\ d_{k+1} &\coloneqq \left[d_k + \alpha \left(\mathcal{R} + \gamma \mathcal{P} \overline{\Phi} \omega_{k+1} - \Xi^T \overline{\Phi} \omega_{k+1}\right)\right]_+\end{aligned} \tag{11.10}$$

Si expresamos la actualización de la variable dual de manera escalar, en términos de las derivadas parciales:

$$d_{k+1}(s,a) \coloneqq \left[d_k(s,a) + \alpha \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \bar{\phi}(s')^T \omega_{k+1} - \bar{\phi}(s)^T \omega_{k+1}\right)\right]_+$$

se aprecia que de nuevo podremos hacer una aproximación estocástica del gradiente como la que se hizo en (10.2), llegando a:

$$\widehat{d}_{k+1}(s,a) \coloneqq \left[\widehat{d}_k(s,a) + \left(r + \gamma \bar{\phi}(s')^T \omega_{k+1} - \bar{\phi}(s)^T \omega_{k+1}\right)\right]_+$$

donde s el estado actual, s' el estado al que se transita y r la recompensa instantánea obtenida en la transición de s a s'.

Bajo las mismas ideas expuestas en la subsección 10.1.2, la actualización de d se llevará a cabo con las muestras obtenidas de correr varios episodios, pues así se conseguirá una aproximación más exacta del gradiente, y por consiguiente, una mejor estimación de la actualización de d.

Todas las ideas que se han presentado se recogen en el algoritmo 11.2. Como vemos, la actualización de d se refinará de manera iterativa, bien hasta que no se disponga de

más episodios que correr, o bien hasta asegurar un cierto grado de convergencia δ_{π_d} de la política que se vaya extrayendo en cada iteración

Algoritmo 11.2 Control libre de modelo basado en aproximaciones lineales: actualización mediante ascenso por gradiente estocástico en la variable dual d.

Entrada: ω_{k+1}, el vector de parámetros nuevo obtenido en la etapa de predicción. d_k, la variable dual actual que se quiere mejorar. $\alpha_{_D}$, la tasa de aprendizaje. Funciones base $\bar{\phi}$.

Salida: d_{k+1}, la variable dual mejorada.

$d(s,a) \leftarrow d_k(s,a)$
repetir(para cada episodio)
  $\Delta \leftarrow 0$
  Inicializar s
  repetir(para cada paso en el episodio)
    Escoger la acción $a \sim \pi_{d_k}(\cdot|s)$
    $d_{antigua} \leftarrow d(s,a)$
    Tomar la acción a y observar $r, s', \bar{\phi}(s')$
    $d(s,a) \leftarrow d(s,a) + \alpha_{_D}(r + \gamma\bar{\phi}(s')^T\omega_{k+1} - \bar{\phi}(s)^T\omega_{k+1})$
    $d(s,a) \leftarrow \text{máx}\,(0, d(s,a))$
    $\Delta \leftarrow \text{máx}\left[\Delta, |\pi_d\,(a \mid s) - \pi_{d_{antigua}}\,(a \mid s)\,|\right]$
    $s \leftarrow s'$
  hasta que s sea terminal
hasta que no podamos correr más episodios o $\Delta < \delta_{\pi_d}$
$d_{k+1} \leftarrow d$
devolver d_{k+1}

11.1.3. Algoritmo Bellman-ascenso dual con aproximación lineal libre de modelo (BDALA-MF)

Por último, se van a juntar las dos etapas anteriores para presentar el algoritmo Bellman-ascenso dual con aproximación lineal y libre de modelo (*Bellman-Dual Ascent with Linear Approximation - Model Free*, BDALA-MF). Para ello se tendrán en cuenta las mismas consideraciones que se hicieron en la subsección 10.1.4: el bucle principal sólo se detendrá cuando no podamos correr más episodios o se haya alcanzado un cierto nivel de convergencia δ, lo que ocurra antes; se actualizarán ambas variables a partir de la experiencia generada con un número N_{epi} reducido de episodios; se seguirá una política $\epsilon - \pi_{d_k}$ como la que se define en (10.3); y se hará uso de una memoria de repetición $\mathbb{S}$ que permitirá emplear las mismas muestras en la etapa de predicción y de control.

El pseudocódigo de BDALA-MF se muestra en el algoritmo 11.3.

Algoritmo 11.3 Algoritmo Bellman-ascenso dual con aproximación lineal, libre de modelo.

Entrada: $\alpha_{_D}$, la tasa de aprendizaje de la etapa de control. ϵ, el parámetro de exploración. Funciones base $\bar{\phi}$.

Salida: π, la política óptima π^* aproximada.

Inicializar $\overline{\Gamma} = 0_{N\times N}$, $\overline{\Lambda} = 0_{N\times N}$, $\overline{z} = 0_N$
Inicializar $d(s,a)$ con cualquier valor positivo, para todo $(s,a) \in \{\mathcal{S} \times \mathcal{A}\}$
$d_k \leftarrow d$
repetir ▷ Bucle principal de Bellman-ascenso dual
    $\xi \leftarrow 0$
    Inicializar la memoria de repetición $\mathbb{S} = \{\}$ vacía
    repetir(para cada episodio) ▷ Etapa de predicción: evaluación de la política
        Inicializar s, a y observar $\bar{\phi}(s)$
        repetir(para cada paso en el episodio)
            Tomar la acción a y observar $r, s', \bar{\phi}(s')$
            Escoger la acción $a' \sim \epsilon - \pi_{d_k}(\cdot|s)$
            Aumentar la memoria de repetición $\mathbb{S}$ con la muestra actual (s, a, r, s')
            $\overline{\Gamma} \leftarrow \overline{\Gamma} + \bar{\phi}(s)\bar{\phi}(s)^T$
            $\overline{\Lambda} \leftarrow \overline{\Lambda} + \bar{\phi}(s)\bar{\phi}(s')^T$
            $\overline{z} \leftarrow \overline{z} + \bar{\phi}(s)r$
            $s \leftarrow s'$
            $a \leftarrow a'$
        hasta que s sea terminal
    hasta que se hayan corrido N_{epi} episodios
    $\omega_{k+1} = \left(\overline{\Gamma} - \gamma\overline{\Lambda}\right)^{-1} \overline{z}$
    para para cada muestra de $\mathbb{S}$ **hacer** ▷ Etapa de control: mejora de la política
        Recuperar (s, a, r, s')
        $d_{antigua} \leftarrow d(s,a)$
        $d(s,a) \leftarrow d(s,a) + \alpha_{_D}(r + \gamma\bar{\phi}(s')^T\omega_{k+1} - \bar{\phi}(s)^T\omega_{k+1})$
        $d(s,a) \leftarrow \max(0, d(s,a))$
        $\xi \leftarrow \max\left[\xi, |\pi_d(a \mid s) - \pi_{d_{antigua}}(a \mid s)|\right]$
    $d_k \leftarrow d$
    $\omega_k \leftarrow \omega_{k+1}$
hasta que no se puedan correr más episodios o $\xi < \delta$
$\pi \leftarrow \pi_{d_k}$
devolver π

11.2. Evaluación del algoritmo

En el capítulo 6 se presentó el algoritmo LSPI como una de las soluciones basadas en aproximaciones lineales más extendidas en la actualidad en lo que a métodos para resolver problemas de RL cuando el problema es de gran escala se refiere. El objetivo del algoritmo BDALA-MF desarrollado en este capítulo será por tanto llegar a alcanzar

resultados similares a los de LSPI, y en el mejor de los casos, superarlo.

A continuación, se pasa a evaluar el rendimiento del algoritmo 11.3 con dos problemas típicos que se mencionaron al comienzo del capítulo: el paseo por la cadena o (*chain walk*) y el problema del coche de montaña (*mountain car*). Para ello, se resolverán ambos problemas con el algoritmo desarrollado en este capítulo y con LSPI de cara a, finalmente, enfrentar los resultados de ambas ejecuciones y poder medir el grado de eficacia de nuestro algoritmo novel.

La estructura de esta sección será la misma que la análoga del capítulo 10.

11.2.1. Metodología de evaluación

La metodología en la evaluación de los resultados será la misma que se presentó en la subsección 10.3.1. A grandes rasgos, se resume en que nuestro principal foco de interés se situará en el retorno obtenido, medido a través de las curvas de aprendizaje y de explotación tras convergencia.

11.2.2. Problema bajo estudio 5: paseo por la cadena

Presentación del problema

El problema del paseo por la cadena consiste en un MDP de naturaleza similar al problema *random walk* del capítulo anterior. Aunque el espacio de estados y de acciones de este problema sea pequeño y discreto, servirá como prueba de concepto para verificar la validez del método BDA-MF con aproximación lineal de la función valor propuesto.

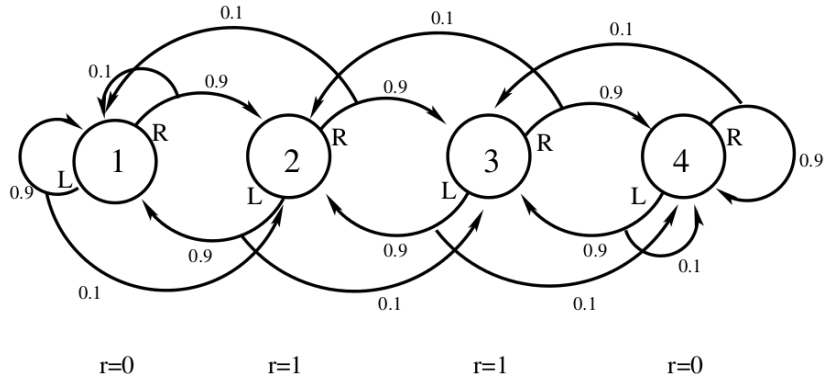


Figura 11.1: Problema del paseo por la cadena.

El problema del paseo por la cadena se representa por una cadena con 4 estados (numerados del 1 al 4) como la mostrada en la figura 11.1. Existen 2 acciones posibles: izquierda (*left*, L) y derecha (*right*, R); es decir: $|\mathcal{A}| = 2$, $\mathcal{A} = \{a_1, a_2\} = \{\text{L, R}\}$.

En lo que respecta a la matriz de transición, esta presenta un ligero carácter aleatorio de manera que, si elegimos movernos en una dirección, se tendrá éxito con probabilidad 0,9, cambiando de estado en la dirección deseada, y fracaso con probabilidad 0,1, cambiando de estado en la dirección opuesta.

Como se puede apreciar a partir de la figura 11.1, no habrá estados terminales; al alcanzar los estados 1 o 4 se seguirá la ejecución con total normalidad, sin que finalice ningún episodio. De este modo, la simulación de un episodio terminará cuando se alcance un número máximo de pasos determinado por el agente. Todos los episodios comenzarán en el estado $s = 1$.

El vector de recompensas sobre los estados será el siguiente:

$$\mathcal{R} = [0,\ 1,\ 1,\ 0]^T$$

Parece entonces claro que la política óptima será:

$$\pi(s_1) = R,\ \pi(s_2) = R,\ \pi(s_3) = L,\ \pi(s_4) = L$$

Respecto a las funciones base que se emplearán, se escogerá un conjunto de 3 características polinómicas de la forma $\overline{\phi} = \left[1, s, s^2\right]$, donde s es el número del estado.

Por último, añadir que el factor de descuento γ escogido para el cálculo del retorno fue $\gamma = 0{,}9$.

Calibración de parámetros

El proceso de calibración seguido consiste en el mismo análisis que se realizó en el problema bajo estudio 1 del capítulo 10, pero adaptado al nuevo problema. Dado que mostrar las gráficas de calibración de nuevo no aportaría información adicional, se pasa directamente a presentar los resultados obtenidos tras encontrar los parámetros óptimos del algoritmo para el problema actual.

Siguiendo este mismo criterio, tampoco se hará mención al proceso de calibración en el siguiente problema bajo estudio.

Presentación de resultados óptimos y comparación con LSPI

Todos los resultados que se muestran a continuación fueron obtenidos mediante el promedio de 100 experimentos independientes.

Los parámetros óptimos del algoritmo BDALA-MF para el problema que se está tratando son:

N_{epi}	α_D	ϵ
1	0,5	0,2

Cuadro 11.1: Parámetros óptimos del algoritmo BDALA-MF, para el problema *paseo por la cadena*.

Aunque el número de episodios N_{epi} sea pequeño, recordar que el número total de episodios simulados viene dado por $N_{epi}^{TOT} = N_{iter}N_{epi}$, habiéndose escogido $N_{iter} = 300$.

Los resultados óptimos de la curva de aprendizaje y la curva de explotación durante el aprendizaje para LSPI y BDALA-MF se muestran en la figura 11.2. Como se puede observar, para este problema los valores del parámetro de exploración óptimo son bajos en ambos casos. En consecuencia, las curvas de aprendizaje y de explotación durante el aprendizaje convergen a valores similares ya que un valor ϵ pequeño implica explotación del conocimiento aprendido, que es lo que precisamente se hace en la gráfica 11.2(b).

La tabla 11.2 recoge, a modo resumen, el episodio en que converge cada algoritmo y el valor de G al que se converge cuando explotamos la política aprendida.

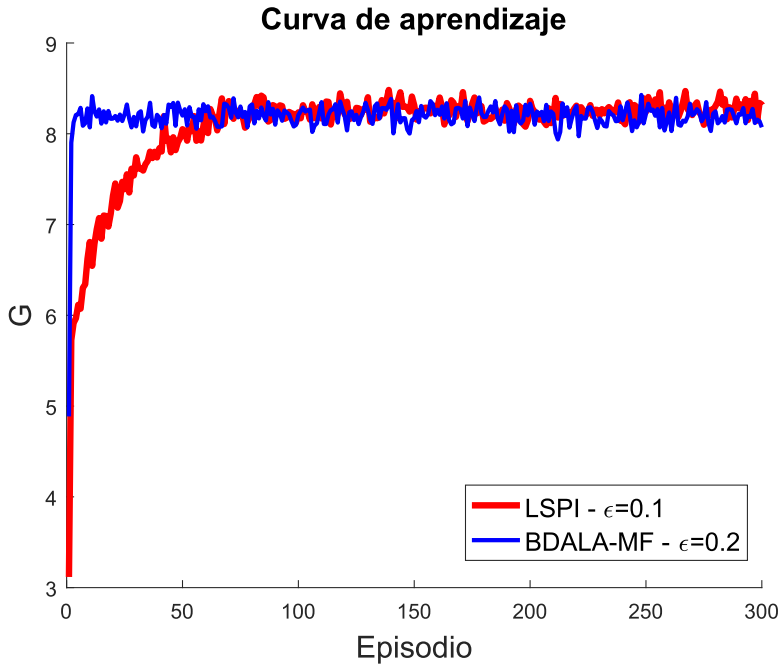
	Episodios hasta convergencia	G tras convergencia ($\epsilon = 0$)
LSPI	80	8,5288
BDALA-MF	2	9,1425

Cuadro 11.2: Número de episodios necesarios hasta converger a una solución estable (columna central), y retorno obtenido de la explotación de la política aprendida tras la convergencia (columna derecha), para el problema *paseo por la cadena*.

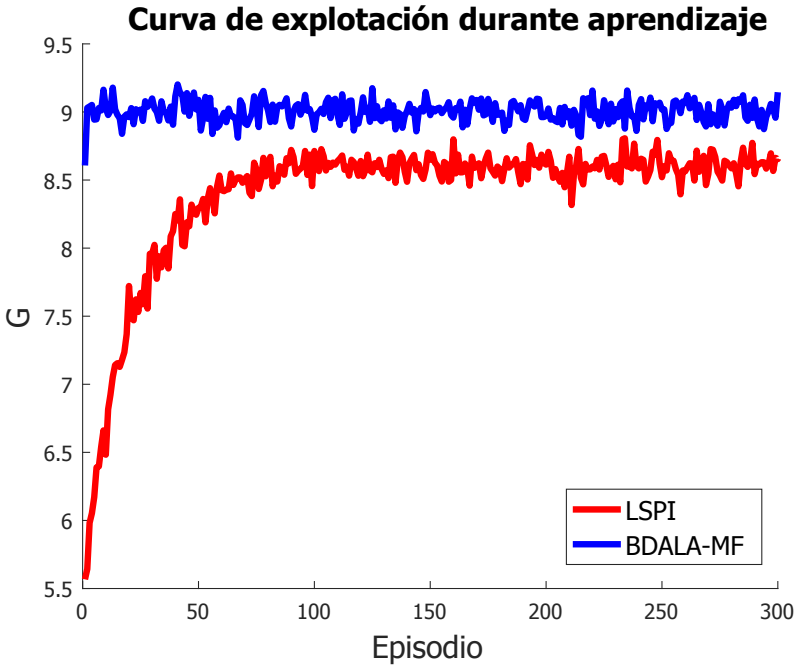
En vista de los resultados obtenidos, se puede concluir que esta prueba de concepto ha sido exitosa: BDALA-MF consigue converger a un retorno mayor que LSPI y en un número insignificante de episodios. La convergencia de LSPI a un valor menor, se traduce en que, en determinados experimentos de entre los 100 promediados, no se conseguía encontrar la política óptima.

El hecho de que haya funcionado tan bien esta prueba se piensa que pueda ser debido a que la matriz de transición del problema evaluado no es determinista. Tal y como se vio en la sección 10.3, el algoritmo Bellman-ascenso dual se ve favorecido cuando las transiciones presentan cierta aleatoriedad, pues ello se traduce en exploración. Como se concluyó en la sección 10.4, la exploración será el aspecto fundamental para la búsqueda en el espacio de políticas llevada a cabo por BDALA-MF.

Por último, comentar que el valor de G al que convergen las gráficas de la figura 11.2 presentan cierto ruido. Esto es debido principalmente a que la matriz de transición no es determinista, con lo cual la acción elegida por el agente no es siempre la tomada finalmente por el entorno.



(a) Evolución durante el aprendizaje.



(b) Evaluación del aprendizaje tras cada episodio.

Figura 11.2: Comparativa de los algoritmos BDALA-MF y LSPI para el problema *paseo por la cadena*. En (a) se muestra la curva de aprendizaje y en (b) la curva de explotación del aprendizaje realizado en cada episodio, ambas con los parámetros óptimos de cada algoritmo.

11.2.3. Problema bajo estudio 6: el coche de montaña

A continuación se pone a prueba el algoritmo BDALA-MF con un problema de mayores exigencias.

Presentación del problema

El problema del coche de montaña, una prueba estándar en el campo de aprendizaje por refuerzo, es un problema en el cual un coche con poca potencia tiene que conseguir subir la ladera de una colina. Puesto que la gravedad es más fuerte que el motor del coche, incluso cuando este acelera al máximo no es capaz de subir la ladera. El coche comienza situado en el medio del valle y debe aprender a aprovechar la energía potencial subiendo la colina opuesta para ganar aceleración en la caída. Es decir, deberá balancearse entre las laderas del valle para conseguir ganar la inercia suficiente como para salir de él (por el lado derecho). La figura 11.3 ilustra el problema.

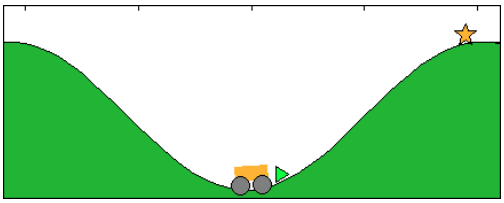


Figura 11.3: Problema del coche de montaña.

Cabe destacar que este problema ha sido empleado como banco de pruebas en varios artículos de aprendizaje por refuerzo. Aunque parezca relativamente sencillo, el problema del coche de montaña es comúnmente empleado porque requiere que el agente de RL aprenda a partir de dos variables continuas: posición y velocidad.

Para un estado dado (posición y velocidad) del coche, el agente tendrá la posibilidad de conducir a la izquierda, a la derecha o no usar el motor. Mientras que el objetivo no se alcance, el agente recibirá una recompensa negativa en cada paso temporal que transcurra. Además, el agente no dispone de información sobre el objetivo hasta que lo consigue alcanzar por primera vez.

Respecto a las funciones base que se emplearán, esta vez se escogerán características de tipo RBF. El número de características N_v a emplear será ahora uno de los parámetros de optimización también. Recordar que para construir los vectores de RBF, cada dimensión del problema se dividirá en N_v partes que determinarán la media (o centroide) de la distribución Gaussiana con que se aproximará cada estado.

Adaptación de BDALA-MF al caso continuo

Observando la etapa de control del algoritmo 11.3, se puede intuir que ahora, el hecho de que el espacio de estados sea bidimensional y continuo en cada una de las variables (posición y velocidad) va a suponer un problema, pues no se podrá llevar a cabo la actualización de la variable dual de manera tabular/exacta. Por ello, habrá que emplear también algún tipo de aproximación en la variable dual. De entre todos los tipos que se estudiaron en la sección 6.1, se decidió escoger la técnica de agregación de estados: particionar el conjunto de estados en N_d subconjuntos separados. De este modo, se puede seguir empleando el mismo esquema que el mostrado en el algoritmo 11.3 con la salvedad de que ahora cada estado en el que se encuentre el coche estará asociado a una de las N_d particiones. En consecuencia, la política que se extraiga de la variable dual será una política generalista en el sentido de que, para todos los estados que caigan dentro de la misma partición, se tomará la misma decisión sin tener en cuenta la variación entre dos estados consecutivos. La modificación que se le hizo a la etapa de control de BDALA-MF se muestra en el algoritmo 11.4.

Algoritmo 11.4 Modificación de la etapa de control del algoritmo BDALA-MF, incluyendo aproximación de la variable dual mediante agregación de estados.

1: **para** para cada muestra de $\mathbb{S}$ **hacer** ▷ Etapa de control: mejora de la política
2: Recuperar $(s,\, a,\, r,\, s')$
3: Encontrar la partición $\mathcal{S}_n$ a la que pertenece el estado s
4: $d_{antigua} \leftarrow d(\mathcal{S}_n, a)$
5: $d(\mathcal{S}_n, a) \leftarrow d(\mathcal{S}_n, a) + \alpha_D(r + \gamma\bar{\phi}(s')^T\omega_{k+1} - \bar{\phi}(s)^T\omega_{k+1})$
6: $d(\mathcal{S}_n, a) \leftarrow \text{máx}\,(0, d(\mathcal{S}_n, a))$
7: $\xi \leftarrow \text{máx}\left[\xi, |\pi_d\,(a \mid \mathcal{S}_n) - \pi_{d_{antigua}}\,(a \mid \mathcal{S}_n)\,|\right]$

En cuanto a la partición del espacio de estados realizada, este se dividió en $N_d = N_v^2$ partes. Para determinar estas N_v^2 partes, se tomaron los N_v centroides definidos en las RBF para la dimensión «velocidad» del estado, los otros N_v definidos en las las RBFs para la dimensión «posición», y se hicieron todas las posibles combinaciones (sin repetición). De esta manera, para cada par posición-velocidad fue posible determinar su partición más cercana en base a su distancia hasta la partición.

Presentación de resultados óptimos y comparación con LSPI

Todos los resultados que se muestran fueron obtenidos mediante el promedio de 50 experimentos independientes.

Los parámetros óptimos del algoritmo BDALA-MF para el problema que se está tratando son:

N_{epi}	α_D	ϵ	N_v
1	0,0001	0,01	6

Cuadro 11.3: Parámetros óptimos del algoritmo BDALA-MF, para el problema del *coche de montaña*.

El número total de episodios simulados viene dado por $N_{epi}^{TOT} = N_{iter}N_{epi}$, habiéndose escogido $N_{iter} = 300$.

Los resultados óptimos de la curva de aprendizaje y la curva de explotación durante el aprendizaje para LSPI y BDALA-MF se muestran en la figura 11.4.

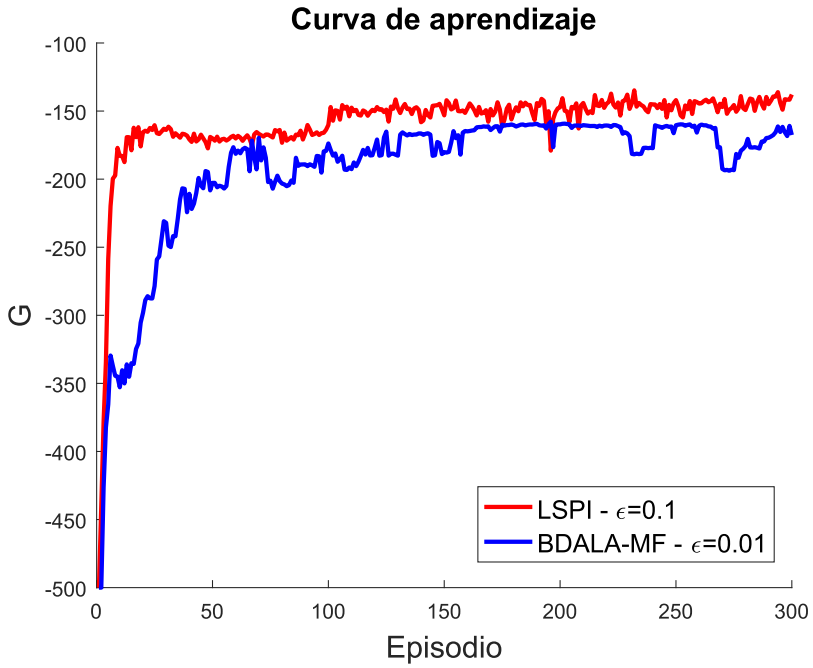
La tabla 11.4 recoge, a modo resumen, el episodio en que converge cada algoritmo y el valor de G al que se converge cuando explotamos la política aprendida. Como se puede deducir a partir de la figura 11.4 y de la tabla 11.4, BDALA-MF ahora no funcionará tan bien como en el problema bajo estudio anterior, pues se tardará más en converger y con un retorno menor. En vista de las conclusiones que se sacaron en la sección 10.4, la ventaja que LSPI consigue frente a BDALA-MF será debido a que el entorno se comporta de manera determinista.

	Episodios hasta convergencia	G tras convergencia ($\epsilon = 0$)
LSPI	100	$-149{,}1$
BDALA-MF	155	$-160{,}7$

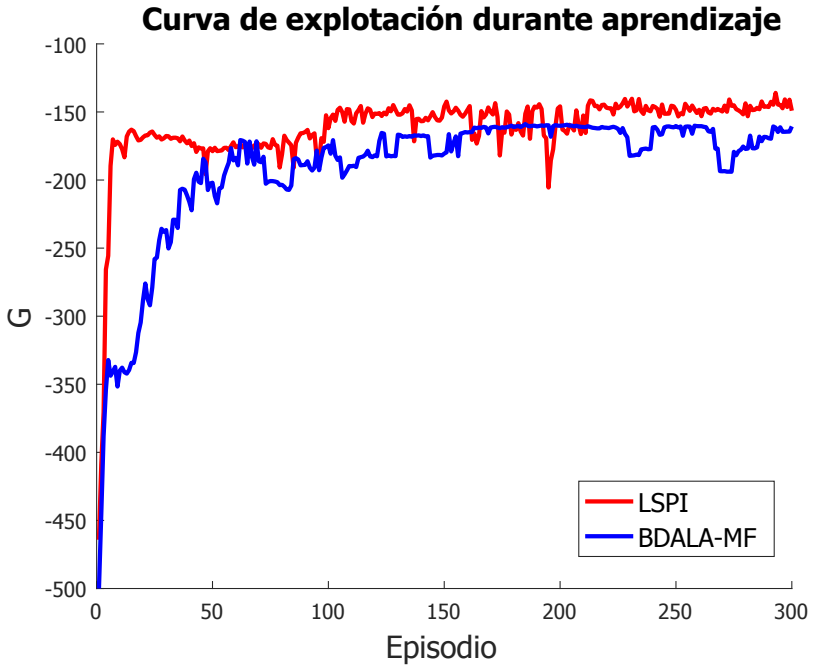
Cuadro 11.4: Número de episodios necesarios hasta converger a una solución estable (columna central), y retorno obtenido de la explotación de la política aprendida tras la convergencia (columna derecha), para el problema del *coche de montaña*.

Si se compara el número de episodios necesario para alcanzar la convergencia mostrado en la tabla 11.4 con lo que gráficamente se puede observar en la figura 11.4 (b), se aprecia una ligera discrepancia en el caso de LSPI que se explica de la siguiente manera: para obtener el número de episodios hasta convergencia en la tabla 11.4 se empleó el criterio de alcanzar el 95 % del retorno tras convergencia, criterio que se cumple a los 100 episodios. No obstante, por inspección visual de la figura 11.4 (b) se observa que cuando han transcurrido 100 episodios, prácticamente se ha convergido, el problema es que no al retorno máximo sino a un valor cercano. De manera práctica, se podría considerar que LSPI converge en aproximadamente 20 episodios de acuerdo a las gráficas. No obstante, dependerá de la aplicación final el poder considerar uno u otro valor como suficiente para asegurar la convergencia del algoritmo.

Para concluir, únicamente mencionar el detalle del ruido que presentan las gráficas de la figura 11.4. Si bien es cierto que ahora el entorno es determinista y cabría esperar una convergencia constante, hay que tener en cuenta que cuando se introducen aproximaciones se pierde todo el carácter ideal y esperado posible. Dado que ahora la política se extrae



(a) Evolución durante el aprendizaje.



(b) Evaluación del aprendizaje tras cada episodio.

Figura 11.4: Comparativa de los algoritmos BDALA-MF y LSPI para el problema del *coche de montaña*. En (*a*) se muestra la curva de aprendizaje y en (*b*) la curva de explotación del aprendizaje realizado en cada episodio, ambas con los parámetros óptimos de cada algoritmo.

por medio (o a partir) de una estimación de la función valor, si esta es poco precisa, dará lugar a políticas ligeramente oscilantes y en consecuencia convergencias ruidosas.

11.3. Análisis y discusión de los resultados

De cara a facilitar la comparativa, se recogen en la tabla 11.5 los resultados de explotación de las pruebas anteriores:

	Episodios hasta convergencia	G tras convergencia ($\epsilon = 0$)
LSPI	80	8,5288
BDALA-MF	2	9,1425

(a) Paseo por la cadena (entorno aleatorio, conjunto de estados pequeño y discreto)

	Episodios hasta convergencia	G tras convergencia ($\epsilon = 0$)
LSPI	100	$-149{,}1$
BDALA-MF	155	$-160{,}7$

(b) Coche de montaña (entorno determinista, conjunto de estados continuo)

Cuadro 11.5: Resultados de explotación para los dos problemas bajo estudio evaluados.

A pesar de que la complejidad de los problemas sea muy diferente, al comparar el comportamiento de BDALA-MF relativo a LSPI en cada caso de estudio, se deduce que el haber aproximado la variable dual ha degradado notablemente el desempeño del algoritmo desarrollado. Para estas pruebas se ha utilizado una discretización del conjunto de estados, pero esta es sólo una opción de entre muchas disponibles. Sin ir más lejos, otra alternativa hubiese sido probar una aproximación lineal de la variable dual mediante RBFs del mismo modo que se hizo con la función valor. Dado que de este modo se podría tener en cuenta la variación entre dos estados consecutivos de una manera más precisa, se intuye que esta elección habría mejorado la convergencia. No obstante, por limitaciones de tiempo no se pudo probar, de manera que se plantea como una idea futura a desarrollar.

Capítulo 12

Conclusiones y trabajo futuro

12.1. Conclusiones

En este trabajo se ha presentado el desarrollo de un algoritmo para la resolución del problema de aprendizaje por refuerzo, basado en un punto de vista alternativo y poco estudiado hasta el momento en la literatura: la exploración del espacio de políticas.

Para alcanzar este nuevo enfoque, se ha partido del problema de control óptimo y su formulación como un programa lineal, hasta llegar al problema dual asociado. A raíz del problema dual, se han hecho una serie de interpretaciones derivadas de la formulación del problema de aprendizaje por refuerzo como un MDP, que han permitido establecer un vínculo entre la variable dual del problema dual y la política de comportamiento del problema de control. De este modo, se ha llegado a la conclusión de que encontrar la solución óptima del problema dual equivale a encontrar la política de comportamiento óptima, y por tanto a resolver el problema de RL. Tras darle esta interpretación a la variable dual, se ha presentado una manera de resolver el problema dual basada en el conocido método de ascenso dual, al cual se le han hecho una serie de modificaciones dando lugar al método Bellman-ascenso dual.

Mediante la elección de este método basado en gradiente, se pudo formular un algoritmo que obtiene una estimación de dicho gradiente de manera estocástica a partir de las muestras de experiencia generadas al interactuar con el entorno. Como resultado, se derivó el algoritmo de aprendizaje por refuerzo BDA-MF.

Con intención de comparar el desempeño de BDA-MF con el estado del arte, se resolvieron una serie de problemas típicos y se compararon los resultados obtenidos con SARSA y Q-learning. Tras estas pruebas, pudo observarse que el nuevo algoritmo desarrollado se comportaba mejor que los ya existentes en situaciones en que el problema presentaba cierto carácter aleatorio. A raíz de ello, se llegó a la conclusión de que BDA-MF es más sensible a la exploración que SARSA y Q-learning.

Una vez formalizado y probado el algoritmo novel con problemas de pequeña escala, se extendieron todas estas ideas al caso en que el conjunto de estados es grande o continuo, dando lugar al algoritmo BDALA-MF. Este algoritmo basa su funcionamiento en las mismas ideas que el anterior, pero ahora se lleva a cabo la estimación de la función valor y de la variable dual a través de aproximaciones lineales.

Finalmente, se comparó el comportamiento de BDALA-MF con el estado del arte mediante la resolución de dos problemas típicos, y se llegó a la conclusión de que la aproximación de la variable dual degrada considerablemente el comportamiento del algoritmo desarrollado.

12.2. Trabajo futuro

Como líneas de trabajo futuro, se plantean una serie de mejoras de los algoritmos presentados en este documento.

La primera de ellas consiste en resolver el problema de la linealidad del Lagrangiano, detallado en la sección 9.3. Para ello se propone tomar el Lagrangiano aumentado, el cual introduce un término cuadrático que podría además mejorar las propiedades de convergencia del algoritmo. Al solucionar el problema de la linealidad, se estudiaría la posibilidad de utilizar el método Arrow-Hurwicz como alternativa a la técnica Bellman-ascenso dual empleada.

La segunda mejora que se propone, relativa a la etapa de predicción de BDA-MF, consiste en sustituir la resolución de la ecuación de Bellman mediante TD por una versión similar pero que incorpora algunas mejoras: TD(λ). Aunque este método no ha sido estudiado en este trabajo, a grandes rasgos consiste en una versión intermedia entre la estimación por TD y Monte-Carlo, de manera que se consigue mantener una varianza reducida, como ocurre con TD, a la par que se lleva a cabo la reducción del sesgo característica del método de Monte-Carlo.

Una tercera consideración de cara a aumentar el rendimiento de los algoritmos desarrollados, será la mejora en la aproximación del gradiente. En lugar de emplear una estimación basada en la muestra del instante actual, se propone hacer uso de métodos más sofisticados como AdaGrad, RMSProp o Adam. Con ello se pretende acelerar la convergencia de los gradientes, y por tanto la búsqueda de la política óptima.

La última, y quizás más importante, de las mejoras que se proponen, es la extensión del método Bellman-ascenso dual a problemas de gran escala mediante el uso de aproximaciones de funciones no lineales, más concretamente a través de redes neuronales. La combinación de la teoría de aprendizaje por refuerzo con los conceptos relativos a redes neuronales es lo que a día de hoy se conoce como aprendizaje por refuerzo profundo (*Deep Reinforcement Learning*, DPL), y está dando lugar a nuevos algoritmos muy potentes,

empleados por grandes empresas como Google, Facebook o Microsoft. Por ello, combinar las ideas del algoritmo aquí planteado con la aproximación mediante redes neuronales, podría suponer un gran avance en el aprendizaje por refuerzo profundo a través de la exploración del espacio de políticas.

Bibliografía

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning : An Introduction*. MIT Press, 1998.

[2] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[3] T. Wang, D. Lizotte, M. Bowling, and D. Schuurmans, "Dual representations for dynamic programming," *Machine Learning Research*, 2008.

[4] Y. Chen and M. Wang, "Stochastic primal-dual methods and sample complexity of reinforcement learning," *CoRR*, vol. abs/1612.02516, 2016.

[5] M. L. Puterman, *Markov Decision Process: discrete stochastic dynamic programming*. John Wiley & Sons, 2005.

[6] J. N. T. Dimitris Bertsimas, *Introduction to Linear Optimization*. Athena Scientific Series in Optimization and Neural Computation, 6, Athena Scientific, 1997.

[7] K. J. Arrow and L. Hurwicz, *Studies in Linear and Non-Linear Programming*. No. 2 in Stanford Mathematical Studies in the Social Sciences, Stanford University Press, 1958.

[8] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, pp. 1–122, Jan. 2011.

[9] A. Sayed, "Adaptation, learning, and optimization over networks," *Found. Trends Mach. Learn.*, vol. 7, pp. 311–801, July 2014.

[10] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in Neural Information Processing Systems*, 2000.

Anexos

Anexo A

Antes de explicar el vínculo entre la minimización de una función lineal definida a trozos y el problema del epigrafo, se va a hacer una pequeña introducción a estos conceptos.

Definición de epigrafo

El grafo de una función $f : \mathbb{R}^n \to \mathbb{R}$ se define como:

$$\{(x, f(x)) \mid x \in \text{dom } f\}$$

el cual es un subconjunto de $\mathbb{R}^{n+1}$. El *epigrafo* de una función $f : \mathbb{R}^n \to \mathbb{R}$ se define como:

$$\text{epi } f = \{(x, t) \mid x \in \text{dom } f, f(x) \leq t\}$$

el cual es un subconjunto de $\mathbb{R}^{n+1}$. Esta definición se ilustra en la figura A.1.

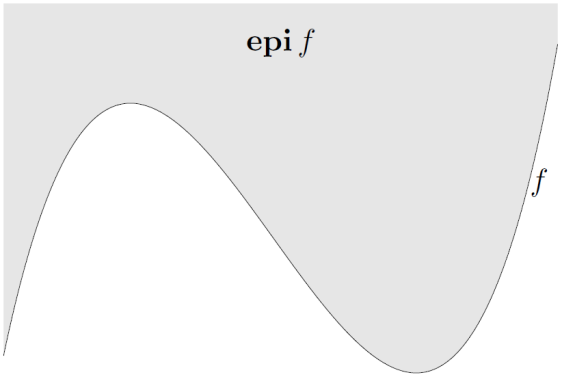


Figura A.1: Sombreado en gris se muestra el epigrafo de una función f. El límite inferior, mostrado en negro, representa el grafo de f [2].

Problema de optimización en forma de epigrafo

La *forma de epigrafo* del problema estándar (7.1) es la siguiente:

$$\begin{array}{lll} \underset{t}{\text{minimize}} & t & \\ \text{subject to} & f_o(x) - t \leq 0 & \\ & f_i(x) \leq 0, & i = 1, \ldots, m \\ & h_i(x) = 0, & i = 1, \ldots, m \end{array} \tag{A.1}$$

cuyas variables son $x \in \mathbb{R}^n$ y $t \in \mathbb{R}$. Se puede ver fácilmente que esta forma es equivalente al problema original: (x, t) es optimo para (A.1) sí y solo sí x es óptimo para (7.1) y $t = f_o(x)$.

El problema (A.1) en forma de epigrafo se puede interpretar geométricamente como un problema de optimización en el espacio del grafo (x, t): buscamos minimizar t sobre el epigrafo de f_o, sujeto a las restricciones de x. Esta interpretación se ilustra en la figura A.2.

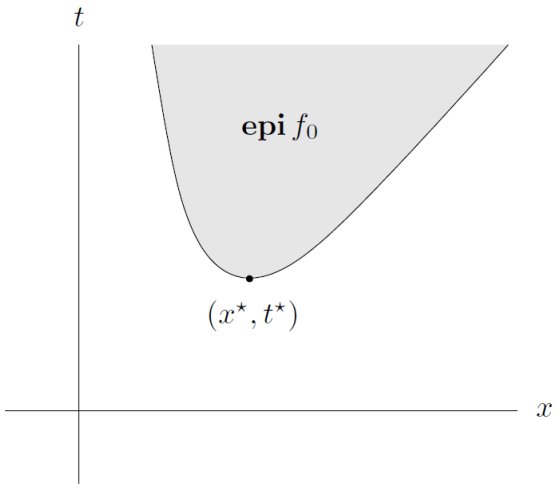


Figura A.2: Interpretación geométrica del problema en forma de epigrafo, para un problema sin restricciones. El problema a resolver será encontrar el punto del epigrafo (sombreado en gris) que minimiza t; es decir, el punto más «pequeño» del epigrafo. El punto óptimo es (x^*, t^*) [2].

Minimización de una función lineal a trozos

Consideremos el problema sin restricciones de minimizar la siguiente función convexa lineal a trozos:

$$f(x) = \max_{i=1\ldots m} (a_i^T x + b_i) \tag{A.2}$$

Este problema puede ser transformado a un programa lineal equivalente, primero

convirtiéndolo en un problema en forma de epigrafo:

$$\begin{aligned} \underset{t}{\text{minimize}} \quad & t \\ \text{subject to} \quad & \underset{i=1\ldots m}{\text{máx}} \, (a_i^T x + b_i) \leq t \end{aligned} \tag{A.3}$$

y a continuación expresando la desigualdad como un conjunto de m desigualdades separadas:

$$\begin{aligned} \underset{t}{\text{minimize}} \quad & t \\ \text{subject to} \quad & a_i^T x + b_i \leq t, \qquad i = 1 \ldots m \end{aligned} \tag{A.4}$$

De este modo se consigue expresar un problema como (A.2) en forma de programa lineal en forma de desigualdad, y variables x y t.

Comparando (9.5) con (A.2), se deduce por analogía con (A.4) que (9.6) está expresado como un problema en forma de epigrafo.

Anexo B

Antes de comenzar la demostración, será conveniente introducir algunas definiciones de gran utilidad en los desarrollos subsecuentes. La primera de ellas será $\rho^\pi(s, j)$, la cual representa la distribución de probabilidad descontada sobre los estados, condicionada al estado inicial j y siguiendo la política π:

$$\rho^\pi(s, j) = \sum_{t=0}^{\infty} \gamma^t \mathbb{P}[S_t = s \mid S_o = j, \pi] \tag{B.5}$$

De manera análoga, se define $\rho^\pi(s, a, j)$ como la distribución de probabilidad descontada sobre los estados y acciones, condicionada al estado inicial j y siguiendo la política π:

$$\rho^\pi(s, a, j) = \sum_{t=0}^{\infty} \gamma^t \mathbb{P}[S_t = s, A_t = a \mid S_o = j, \pi] \tag{B.6}$$

La relación existente entre $\rho^\pi(s, j)$ y $\rho^\pi(s, a, j)$ es por tanto la siguiente:

$$\rho^\pi(s, a, j) = \rho^\pi(s, j)\pi(a \mid s)$$

$$\rho^\pi(s, j) = \sum_{a \in \mathcal{A}} \rho^\pi(s, a, j) = \sum_{a \in \mathcal{A}} \rho^\pi(s, j)\pi(a \mid s)$$

Atendiendo a estas definiciones, según (9.20) podremos expresar $d^\pi(s, a)$ de la siguien-

te manera:

$$d^\pi(s,a) = \sum_{j\in S} \mu(j)\rho^\pi(s,a,j) = \sum_{j\in S} \mu(j) \sum_{t=0}^{\infty} \gamma^t \mathbb{P}[S_t = s, A_t = a \mid S_o = j, \pi] \tag{B.7}$$

o de forma equivalente:

$$d^\pi(s,a) = \sum_{j\in S} \mu(j)\rho^\pi(s,j)\pi(a \mid s) = \sum_{j\in S} \mu(j) \sum_{t=0}^{\infty} \gamma^t \mathbb{P}[S_t = s \mid S_o = j, \pi]\pi(a \mid s) \tag{B.8}$$

Con estas representaciones alternativas de $d^\pi(s,a)$ en mente, podemos comenzar la demostración del teorema 9.1.

Demostración. Para que $d^\pi(s,a)$ sea una solución factible del programa lineal dual, deberán satisfacerse las restricciones que aparecen en (9.19). Es decir:

$$\begin{aligned} & d^\pi(s,a) \geq 0 \\ & \mu(s') + \gamma \sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}} \mathcal{P}^a_{ss'} d^\pi(s,a) - \sum_{a\in\mathcal{A}} d^\pi(s',a) = 0 \end{aligned}$$

Por simple inspección de (B.7) o (B.8), se puede verificar que la restricción de no negatividad se cumplirá siempre, pues todos los términos de los que depende $d^\pi(s,a)$ son no negativos. Por tanto, únicamente queda por demostrar el cumplimiento de la segunda restricción:

$$\gamma \sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}} \mathcal{P}^a_{ss'} d^\pi(s,a) = \sum_{a\in\mathcal{A}} d^\pi(s',a) - \mu(s') \tag{B.9}$$

Para ello, vamos a desarrollar el lado izquierdo de la igualdad (B.9). Según la definición de $d^\pi(s,a)$ propuesta:

$$\gamma \sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}} \mathcal{P}^a_{ss'} d^\pi(s,a) = \gamma \sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}} \mathcal{P}^a_{ss'} \sum_{j\in S} \mu(j) \sum_{t=0}^{\infty} \gamma^t \mathbb{P}[S_t = s, A_t = a \mid S_o = j, \pi] \tag{B.10}$$

donde el producto $\gamma \sum_{a\in\mathcal{A}} \mathcal{P}^a_{ss'} \gamma^t \mathbb{P}[S_t = s, A_t = a \mid S_o = j, \pi]$ se puede expresar de la siguiente manera:

$$\gamma \sum_{a\in\mathcal{A}} \mathcal{P}^a_{ss'} \gamma^t \mathbb{P}[S_t = s, A_t = a \mid S_o = j, \pi] = \gamma^{t+1} \mathbb{P}[S_{t+1} = s' \mid S_o = j, \pi]$$

Devolviendo este cambio a (B.10) se llega a:

$$\begin{aligned}\gamma \sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} \mathcal{P}^a_{ss'} d^\pi(s,a) &= \gamma \sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} \mathcal{P}^a_{ss'} \sum_{j\in S}\mu(j)\sum_{t=0}^{\infty}\gamma^t \mathbb{P}[S_t = s, A_t = a \mid S_o = j, \pi] \\ &= \sum_{j\in S}\mu(j)\sum_{t=0}^{\infty}\gamma^{t+1}\mathbb{P}[S_{t+1} = s' \mid S_o = j, \pi]\end{aligned} \tag{B.11}$$

A continuación, vamos a buscar una forma más conveniente de expresar el sumatorio en t. Para ello partimos de:

$$\sum_{t=0}^{\infty}\gamma^t\mathbb{P}[S_t = s' \mid S_o = j, \pi] = \gamma^0\mathbb{P}[S_o = s' \mid S_o = j, \pi] + \sum_{t=1}^{\infty}\gamma^t\mathbb{P}[S_t = s' \mid S_o = j, \pi] \tag{B.12}$$

donde:

$$\mathbb{P}[S_o = s' \mid S_o = j, \pi] = \begin{cases} 1 & \text{si } s' = j \\ 0 & \text{si } s' \neq j \end{cases} \tag{B.13}$$

Por comodidad en los desarrollos, vamos a definir $\delta(s' \mid j)$ como el elemento $(s', j) - ésimo$ de la matriz $I_{|\mathcal{S}|}$. De este modo, y haciendo además un ajuste en el índice t del sumatorio, podremos expresar (B.12) como:

$$\begin{aligned}\sum_{t=0}^{\infty}\gamma^t\mathbb{P}[S_t = s' \mid S_o = j, \pi] &= \delta(s' \mid j) + \sum_{t=1}^{\infty}\gamma^t\mathbb{P}[S_t = s' \mid S_o = j, \pi] \\ &= \delta(s' \mid j) + \sum_{t=0}^{\infty}\gamma^{t+1}\mathbb{P}[S_{t+1} = s' \mid S_o = j, \pi]\end{aligned} \tag{B.14}$$

donde, como vemos, el segundo sumando del lado derecho de la última igualdad es el término que aparece en (B.11). Despejando dicho término de (B.14) tenemos:

$$\sum_{t=0}^{\infty}\gamma^{t+1}\mathbb{P}[S_{t+1} = s' \mid S_o = j, \pi] = \sum_{t=0}^{\infty}\gamma^t\mathbb{P}[S_t = s' \mid S_o = j, \pi] - \delta(s' \mid j)$$

y al sustituirlo en (B.11) se llega a:

$$\begin{aligned}\gamma \sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} \mathcal{P}^a_{ss'} d^\pi(s,a) &= \gamma \sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} \mathcal{P}^a_{ss'} \sum_{j\in S}\mu(j)\sum_{t=0}^{\infty}\gamma^t \mathbb{P}[S_t = s, A_t = a \mid S_o = j, \pi] \\ &= \sum_{j\in S}\mu(j)\sum_{t=0}^{\infty}\gamma^{t+1}\mathbb{P}[S_{t+1} = s' \mid S_o = j, \pi] \\ &= \sum_{j\in S}\mu(j)\left(\sum_{t=0}^{\infty}\gamma^t\mathbb{P}[S_t = s' \mid S_o = j, \pi] - \delta(s' \mid j)\right)\end{aligned} \tag{B.15}$$

A partir de la última igualdad de (B.15) se deduce:

1. De la relación existente entre (B.5) y (B.6):

$$\begin{aligned}\sum_{j\in S}\mu(j)\sum_{t=0}^{\infty}\gamma^t\mathbb{P}[S_t=s'\mid S_o=j,\pi] &= \sum_{j\in S}\mu(j)\sum_{a\in\mathcal{A}}\sum_{t=0}^{\infty}\gamma^t\mathbb{P}[S_t=s'\mid S_o=j,\pi]\pi(a\mid s')\\ &= \sum_{a\in\mathcal{A}}\left(\sum_{j\in S}\mu(j)\rho^{\pi}(s',j)\pi(a\mid s')\right)\\ &= \sum_{a\in\mathcal{A}}\sum_{j\in S}\mu(j)\rho^{\pi}(s',a,j)\\ &= \sum_{a\in\mathcal{A}}d^{\pi}(s',a)\end{aligned} \tag{B.16}$$

2. De (B.13):

$$\sum_{j\in S}\mu(j)\delta(s'\mid j)=\mu(s') \tag{B.17}$$

Con lo cual, (B.15) resulta en:

$$\begin{aligned}\gamma\sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}}\mathcal{P}^a_{ss'}d^{\pi}(s,a) &= \gamma\sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}}\mathcal{P}^a_{ss'}\sum_{j\in S}\mu(j)\sum_{t=0}^{\infty}\gamma^t\mathbb{P}[S_t=s,A_t=a\mid S_o=j,\pi]\\ &= \sum_{j\in S}\mu(j)\sum_{t=0}^{\infty}\gamma^{t+1}\mathbb{P}[S_{t+1}=s'\mid S_o=j,\pi]\\ &= \sum_{j\in S}\mu(j)\left(\sum_{t=0}^{\infty}\gamma^t\mathbb{P}[S_t=s'\mid S_o=j,\pi]-\delta(s'\mid j)\right)\\ &= \sum_{a\in\mathcal{A}}d^{\pi}(s',a)-\mu(s')\end{aligned} \tag{B.18}$$

De este modo, se demuestra el cumplimiento de la segunda restricción:

$$\gamma\sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}}\mathcal{P}^a_{ss'}d^{\pi}(s,a)=\sum_{a\in\mathcal{A}}d^{\pi}(s',a)-\mu(s')$$

y se concluye que $d^{\pi}(s,a)$, tal y como se define en (9.20), es una solución factible del programa lineal dual (9.19). □

Anexo C

Demostración. Supongamos que $d(s,a)$ es una solución factible del programa lineal dual, y definamos $u(s)$ como:

$$u(s) = \sum_{a\in\mathcal{A}} d(s,a)$$

El hecho de que $u(s) > 0$ se desprende de la restricción de igualdad del programa lineal dual (9.19), de la no negatividad de $d(s,a)$ y de que μ es positivo.

Si reescribimos la restricción de igualdad del programa lineal dual (9.19) en términos de u obtenemos:

$$\begin{aligned}\mu(j) &= u(j) - \gamma \sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}} \mathcal{P}^a_{sj} d(s,a) \\ &= u(j) - \gamma \sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}} \mathcal{P}^a_{sj} d(s,a) \frac{u(s)}{\sum_{a'\in\mathcal{A}} d(s,a')}\end{aligned}$$

donde, según (9.21):

$$\begin{aligned}\mu(j) &= u(j) - \gamma \sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}} \mathcal{P}^a_{sj} d(s,a) \mathbb{P}[\pi_d(s) = a] u(s) \\ &= u(j) - \gamma \sum_{s\in\mathcal{S}} \mathcal{P}^{\pi_d}_{sj} u(s)\end{aligned}$$

Expresando este resultado en forma matricial, tenemos:

$$\mu^T = u^T \left(I_{|\mathcal{S}|} - \gamma \mathcal{P}^{\pi_d} \right)$$

y despejando para u:

$$u^T = \mu^T \left(I_{|\mathcal{S}|} - \gamma \mathcal{P}^{\pi_d} \right)^{-1} \tag{C.19}$$

De [5] sabemos que (C.19) se puede expresar como:

$$u^T = \mu^T \left(I_{|\mathcal{S}|} - \gamma \mathcal{P}^{\pi_d} \right)^{-1} = \mu^T \left[\sum_{t=1}^{\infty} (\gamma \mathcal{P}^{\pi_d})^{t-1} \right] = \mu^T \left[\sum_{t=0}^{\infty} (\gamma \mathcal{P}^{\pi_d})^{t} \right]$$

o en forma escalar:

$$u(s) = \sum_{k\in\mathcal{S}} \mu(k) \sum_{t=0}^{\infty} \gamma^t \sum_{a\in\mathcal{A}} \mathbb{P}[S_t = s, A_t = a \mid S_o = k, \pi_d]$$

Por tanto, se deduce de (9.20) que:

$$u(s) = \sum_{k\in\mathcal{S}} \mu(k) \sum_{t=0}^{\infty} \gamma^t \sum_{a\in\mathcal{A}} \mathbb{P}[S_t = s, A_t = a \mid S_o = k, \pi_d] = \sum_{a\in\mathcal{A}} d^{\pi_d}(s,a)$$

y se puede concluir:

$$\sum_{a\in\mathcal{A}} d(s,a) = \sum_{a\in\mathcal{A}} d^{\pi_d}(s,a) \tag{C.20}$$

Para terminar con la demostración, de (B.8) y (C.20) llegamos a:

$$\begin{aligned} d^{\pi_d}(s,a) &= \sum_{j\in S}\mu(j)\sum_{t=0}^{\infty}\gamma^t\mathbb{P}[S_t = s \mid S_o = j,\pi_d]\pi_d(a\mid s) \\ &= \sum_{j\in S}\mu(j)\sum_{t=0}^{\infty}\gamma^t\mathbb{P}[S_t = s \mid S_o = j,\pi_d]\frac{d(s,a)}{\sum_{a'\in\mathcal{A}} d(s,a')} \end{aligned} \tag{C.21}$$

Dado que:

$$\sum_{a\in\mathcal{A}} d^{\pi_d}(s,a) = \sum_{j\in S}\mu(j)\sum_{t=0}^{\infty}\gamma^t\mathbb{P}[S_t = s \mid S_o = j,\pi_d]$$

se deduce de (C.20) y (C.21) que:

$$d^{\pi_d}(s,a) = \sum_{a\in\mathcal{A}} d^{\pi_d}(s,a)\frac{d(s,a)}{\sum_{a'\in\mathcal{A}} d^{\pi_d}(s,a')} = d(s,a)$$

De esta manera se da por demostrado el teorema 9.2. □

Anexo D

Recordando el desarrollo que se hizo en la sección 9.1, el problema primal que nos permite resolver las ecuaciones de Bellman de manera alternativa a la programación dinámica es el siguiente, expresado tanto en forma escalar como vectorial:

$$\begin{array}{lll} \underset{v(s)}{\text{minimize}} & \displaystyle\sum_{s\in\mathcal{S}}\mu(s)v(s) & \\ \text{subject to} & v(s) \geq \mathcal{R}_s^a + \gamma\displaystyle\sum_{s'\in\mathcal{S}}\mathcal{P}_{ss'}^a v\,(s') & \end{array} \qquad \begin{array}{lll} \underset{v}{\text{minimize}} & \mu^T v & \\ \text{subject to} & \Xi^T v \geq \mathcal{R} + \gamma\mathcal{P}v & \end{array} \tag{D.22}$$

para todo $s \in \mathcal{S}$ y todo $a \in \mathcal{A}$.

En la sección 9.2.1 se definió la función objetivo del problema dual (9.19) como la recompensa total descontada esperada, resultado de seguir la política π_d:

$$\sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} d(s,a)\mathcal{R}_s^a = \sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}}\sum_{j\in S}\mu(j)\sum_{t=0}^{\infty}\gamma^t\mathbb{P}[S_t = s, A_t = a \mid S_o = j,\pi_d]\mathcal{R}_s^a \tag{D.23}$$

Haciendo una interpretación similar de la función objetivo $\sum_{s\in\mathcal{S}}\mu(s)v(s)$ del problema

primal (D.22), se demostrará el cumplimiento de las igualdades:

$$\sum_{j\in\mathcal{S}} \mu(j) v^{\pi_d}(j) = \sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}} d(s,a)\mathcal{R}_s^a$$

$$\sum_{j\in\mathcal{S}} \mu(j) v^{\pi}(j) = \sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}} d^{\pi}(s,a)\mathcal{R}_s^a$$

establecidas en el corolario 9.1.

Demostración. De acuerdo a la relación que se puede encontrar en [10], podemos expresar $v^{\pi}(j)$ como:

$$v^{\pi}(j) = \sum_{s\in\mathcal{S}} \rho^{\pi}(s,j) \sum_{a\in\mathcal{A}} \pi(a \mid s)\mathcal{R}_s^a$$

donde $\rho^{\pi}(s,j)$, tal y como se definió en (B.5), representa la distribución de probabilidad descontada sobre los estados, condicionada al estado inicial j y siguiendo la política π.

De este modo:

$$\begin{aligned} v^{\pi}(j) &= \sum_{s\in\mathcal{S}} \rho^{\pi}(s,j) \sum_{a\in\mathcal{A}} \pi(a \mid s)\mathcal{R}_s^a \\ &= \sum_{s\in\mathcal{S}} \sum_{t=0}^{\infty} \sum_{a\in\mathcal{A}} \gamma^t \mathbb{P}[S_t = s \mid S_o = j, \pi]\pi(a \mid s)\mathcal{R}_s^a \\ &= \sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}} \sum_{t=0}^{\infty} \gamma^t \mathbb{P}[S_t = s, A_t = a \mid S_o = j, \pi]\mathcal{R}_s^a \end{aligned}$$

Si ahora se multiplica $v^{\pi}(j)$ por la distribución inicial de estados $\mu(s)$ y se suma sobre todos los estados (función objetivo del problema primal (D.22)), se llega a:

$$\begin{aligned} \sum_{j\in\mathcal{S}} \mu(j) v^{\pi}(j) &= \sum_{j\in S} \mu(j) \sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}} \sum_{t=0}^{\infty} \gamma^t \mathbb{P}[S_t = s, A_t = a \mid S_o = j, \pi]\mathcal{R}_s^a \\ &= \sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}} \sum_{j\in S} \mu(j) \sum_{t=0}^{\infty} \gamma^t \mathbb{P}[S_t = s, A_t = a \mid S_o = j, \pi]\mathcal{R}_s^a \end{aligned} \tag{D.24}$$

Comparando este resultado con la expresión (D.23), se deduce que para una política $\pi \in \Pi$ cualquiera se cumplirá:

$$\sum_{j\in\mathcal{S}} \mu(j) v^{\pi}(j) = \sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}} d^{\pi}(s,a)\mathcal{R}_s^a$$

Si en lugar de seguir una política cualquiera π se sigue aquella obtenida a partir de

la variable dual según el teorema 9.2, se demuestra que:

$$\sum_{j\in\mathcal{S}} \mu(j)v^{\pi_d}(j) = \sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} d^{\pi_d}(s,a)\mathcal{R}_s^a = \sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} d(s,a)\mathcal{R}_s^a$$

Por tanto, se puede afirmar que la función objetivo de los problemas primal y dual representan el mismo concepto: la recompensa total descontada esperada, resultado de seguir una política π.

De manera más informal, se puede intuir esta relación por la propia definición de función valor −recompensa total descontada esperada cuando se empieza en el estado j y a continuación se sigue la política π−, ya que si ponderamos la función valor de cada estado por la probabilidad de empezar en dicho estado y lo sumamos, $\sum_{s\in\mathcal{S}} \mu(s)v^{\pi}(s)$, se obtiene la recompensa total descontada esperada, resultado de seguir la política π. □

Anexo E

Demostración. En la demostración del teorema 9.2, se dedujo que, para cada $s \in \mathcal{S}$, $\sum_{a\in\mathcal{A}} d(s,a) > 0$. Además, dado que d es una solución básica factible y el programa lineal dual tiene $|\mathcal{S}|$ restricciones de igualdad, d podrá tener como mucho $|\mathcal{S}|$ componentes positivas. De estos dos argumentos se deduce por tanto que para cada s, $d(s,a) > 0$ para una única acción $a \in \mathcal{A}$. Por tanto, π_d tal y como se definió en el teorema 9.2 será igual a 1 o a 0, y en consecuencia es determinista. □

Anexo F

Demostración. Supongamos que d^π es una solución factible pero no básica. Entonces existen unas soluciones básicas factibles w y z, y β, $0 \leq \beta \leq 1$ tal que $d(s,a) = \beta w(s,a) + (1-\beta)z(s,a)$. Dado que $w(s,a)$ y $z(s,a)$ son básicas y distintas, sabemos de la primera parte del teorema 9.2 que $\sum_{a\in\mathcal{A}} w(s,a) > 0$ y $\sum_{a\in\mathcal{A}} z(s,a) > 0$ para cada $s \in \mathcal{S}$. Puesto que $w \neq z$, para algún $s \in \mathcal{S}$ existirán dos acciones a y a' para las cuales $w(s,a) > 0$ y $z(s,a') > 0$. En consecuencia d tendrá más de una componente no nula para ese estado $s \in \mathcal{S}$ y π_d ya no será determinista. Puesto que $\pi = \pi_d$, esto contradice la hipótesis de que π es determinista. □

Anexo G

Demostración. Sea d una solución factible del programa lineal dual, a partir del teorema 9.2 sabemos que existe una política $\pi \in \Pi^{MR}$ para la cual $d = d^\pi$. De (9.24) se sabe que:

$$\sum_{j\in\mathcal{S}} \mu(j) v^\pi(j) = \sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}} d^\pi(s,a)\mathcal{R}_s^a \tag{G.25}$$

Por la desigualdad de Cauchy-Schwarz y las propiedades de la norma[1] además se cumple:

$$\left(\sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} d^\pi(s,a)\mathcal{R}_s^a\right)^2 \leq \left(\sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} (d^\pi(s,a))^2\right)\left(\sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} (\mathcal{R}_s^a)^2\right) = \|d^\pi\|_2^2 \|\mathcal{R}\|_2^2$$

$$\sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} d^\pi(s,a)\mathcal{R}_s^a \leq \|d^\pi\|_2 \|\mathcal{R}\|_2 \leq \|d^\pi\|_1 \|\mathcal{R}\|_1 \tag{G.26}$$

Debido a la no negatividad de d por ser una solución factible, podemos expresar $\|d^\pi\|_1$ como:

$$\begin{aligned}
\|d^\pi\|_1 &= \sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} |d^\pi(s,a)| = \sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} d^\pi(s,a) \\
&= \sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}}\sum_{j\in S} \mu(j) \sum_{t=0}^{\infty} \gamma^t \mathbb{P}[S_t = s, A_t = a \mid S_o = j, \pi] \\
&= \sum_{t=0}^{\infty}\sum_{j\in S} \mu(j) \sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} \gamma^t \mathbb{P}[S_t = s, A_t = a \mid S_o = j, \pi] \\
&= \sum_{t=0}^{\infty} \gamma^t = (1-\gamma)^{-1}
\end{aligned} \tag{G.27}$$

Combinando los resultados de (G.25), (G.26) y (G.27) se deduce que la función objetivo del problema dual (y del primal) estará acotada por:

$$\sum_{j\in\mathcal{S}} \mu(j) v^\pi(j) = \sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} d^\pi(s,a)\mathcal{R}_s^a \leq (1-\gamma)^{-1} \max_{a\in\mathcal{A}} \max_{s\in\mathcal{S}} \|\mathcal{R}_s^a\|_1 \tag{G.28}$$

y por tanto, la solución óptima del problema dual tendrá una cota superior.

En [5] se demuestra que si el problema primal o el dual tiene una solución óptima finita, entonces tiene una solución óptima que es solución básica factible. En base a este argumento y a (G.28),queda demostrado que existe una solución básica factible óptima acotada d^* para el programa lineal dual. □

[1] A partir de la desigualdad de Cauchy-Schwarz puede demostrarse que $\|x\|_2 \leq \|x\|_1 \leq \sqrt{n}\,\|x\|_2$

Anexo H

Demostración. De la sección 9.1 se sabe que el problema primal (9.10) que resuelve las ecuaciones óptimas de Bellman es un programa lineal. En este tipo de problemas de optimización, de acuerdo a la teoría detallada en la sección 8.2.2, siempre se satisfacen las condiciones de Slater en su forma débil ya que tanto la función objetivo como las restricciones de desigualdad son afines. Como consecuencia del cumplimiento de estas condiciones, se puede garantizar la existencia de dualidad fuerte y por tanto la brecha de dualidad óptima es cero, es decir:

$$\sum_{s\in\mathcal{S}} \mu(s)v^*(s) = \sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} d^*(s,a)\mathcal{R}_s^a$$

Si además tenemos en cuenta (9.23), podemos extender la igualdad anterior a:

$$\sum_{s\in\mathcal{S}} \mu(s)v^*(s) = \sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} d^*(s,a)\mathcal{R}_s^a = \sum_{s\in\mathcal{S}} \mu(s)v^{\pi_{d^*}}(s)$$

donde $v^{\pi_{d^*}}$ será la función valor obtenida al seguir la política π_{d^*} extraída de la variable dual óptima d^*. En vista del resultado anterior, parece claro que:

$$\sum_{s\in\mathcal{S}} \mu(s)v^*(s) = \sum_{s\in\mathcal{S}} \mu(s)v^{\pi_{d^*}}(s) \iff v^*(s) = v^{\pi_{d^*}}$$

Por tanto se concluye que, si la función valor obtenida al seguir la política π_{d^*} es igual a la función valor óptima, la política π_{d^*} es la política óptima:

$$\pi_{d^*} = \pi*$$

De este modo queda demostrado que la política extraída de la variable dual óptima es la política óptima. □

Anexo I

Demostración. Del capítulo 4 se sabe que las ecuaciones de Bellman de la función valor de estados y de acciones-estados para una política π vienen dadas por:

$$v^\pi(s) = \sum_{a\in\mathcal{A}} \pi(a \mid s) q^\pi(s,a)$$
$$q^\pi(s,a) = \mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{ss'}^a v\left(s'\right)$$

respectivamente. Además:

$$0 \leq \pi(a \mid s) \leq 1,\ \forall s \in \mathcal{S},\ \forall a \in \mathcal{A} \qquad \sum_{a \in \mathcal{A}} \pi(a \mid s) = 1,\ \forall s \in \mathcal{S}$$

De todo ello se deduce que siempre deberá cumplirse:

$$v^{\pi}(s) \geq q^{\pi}(s, a), \qquad \forall s \in \mathcal{S},\ \forall a \in \mathcal{A} \tag{I.29}$$

Por otro lado, tenemos que la restricción impuesta por el problema primal (9.10) es la siguiente:

$$v^{\pi}(s) \geq \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v^{\pi}\left(s'\right) \tag{I.30}$$

Como vemos, el lado derecho de (I.30) es por definición $q^{\pi}(s, a)$. Por tanto, y de acuerdo a (I.29), queda demostrado que cualquier punto v que satisfaga las ecuaciones de Bellman para una política dada, será un punto factible del problema primal (9.10) que resuelve las ecuaciones óptimas de Bellman. □