CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL INSTITUTO POLITÉCNICO NACIONAL

Unidad Zacatenco

Departamento de Control Automático

Control Óptimo y Robusto de Robots Manipuladores basado en Técnicas de Aprendizaje por Refuerzo

Tesis que presenta:

M. en C. José Adolfo Perrusquía Guzmán

para obtener el grado de

Doctor en Ciencias

en la especialidad de

Control Automático

Directores de la tesis:

Dr. Wen Yu Liu

Dr. Alberto Soria López

Ciudad de México Agosto, 2020

Agradecimientos

Agradezco en primer lugar a mis papás, Adolfo y Graciela, por siempre creer en mi, apoyarme en cada decisión y mostrarme con su ejemplo la perseverancia que se necesita para cumplir un sueño. Por eso y más los amo. También agradezco a mi hermano, Fer, que me apoyó, toleró y demostró que todo puede realizarse con dedicación. A mi abuela, mis tíos y primos por apoyarme, quererme y siempre estar al pendiente de mi. Su apoyo fue incondicional y una fuente inagotable de motivación. Los amo a todos.

Agradezco de manera muy especial al Dr. Wen Yu por compartir su experiencia, consejo y sobre todo paciencia en la realización de este trabajo. Gracias por todo el apoyo, amistad y mostrarme cada día con su ejemplo lo que representa ser un buen investigador, colega y sobre todo un buen ser humano. También agradezco enormemente al Dr. Alberto Soria por exigirme, aconsejarme y motivarme a entregar un trabajo de calidad. Aprecio mucho su confianza, preocupación, amistad y solidaridad.

Agradezco a tomas mis amistades que me apoyaron y creyeron en mi desde mi ingreso al Doctorado. Nombrarlos me tomaría un capítulo entero, sin embargo, ustedes saben quienes son. Agradezco que me escucharán, que compartieran su cubículo y departamento, las salidas, risas y todas las grandes experiencias.

Le doy las gracias al Centro de Investigación y Estudios Avanzados y a su Departamento de Control Automático por darme la oportunidad de ser un miembro de su plantel y proporcionarme los medios materiales e intelectuales para mi formación como investigador y permitir encontrar en mis compañeros y asesores una fuente inagotable de amistad y apoyo. Agradezco el apoyo económico que se me brindó para la asistencia a congresos con fines complementarios a mi formación.

Agradezco al Consejo Nacional de Ciencia y Tecnología que por medio de su Programa de Posgrados de Calidad y la beca de manutención de doctorado que me fue otorgada me perimitió dedicarme en mi trabajo de tesis haciendo posible la realización de un proyecto de investigación de calidad y de alto beneficio en mi formación.

¡Gracias a todos!

José Adolfo Perrusquía Guzmán.

"*El hombre razonable se adapta al mundo; el hombre irrazonable persiste en intentar adaptar al mundo hacia él. Por lo tanto, todo progreso depende del hombre irrazonable.*"

George Bernard Shaw.

Dedicado a mis padres:
Adolfo y Graciela.
En especial a mi tío:
Héctor Raúl Olvera (QEPD).

Control Óptimo y Robusto de Robots Manipuladores basado en Técnicas de Aprendizaje por Refuerzo

por

M. en C. José Adolfo Perrusquía Guzmán

Sometido al Departamento de Control Automático
en Agosto 2020, en cumplimiento parcial de los
requerimientos para el grado de
Doctor en Ciencias en Control Automático

Resumen

En esta tesis se presentan técnicas de aprendizaje por refuerzo para el diseño de controladores óptimos y robustos para sistemas no lineales, en especial, robots manipuladores. El aprendizaje por refuerzo es una herramienta de aprendizaje automático que permite encontrar controladores óptimos y adaptables utilizando únicamente mediciones del estado, control y una señal de recompensa. La aplicación de técnicas de aprendizaje por refuerzo en robots es desafiante debido al problema de dimensionalidad provocado por el aumento exponencial de datos. Además el diseño de la recompensa es una tarea no trivial e indispensable en el desempeño del controlador. Estos problemas se hacen más visibles en el diseño de controladores robustos, donde el controlador debe ser capaz de compensar perturbaciones exógenas o endógenas manteniendo buen desempeño del sistema en lazo cerrado. Por lo tanto el problema de control robusto requiere considerar nuevas señales de control y estados que, en un caso ideal, no eran considerados.

En esta tesis, se propone utilizar diferentes herramientas para lidiar con el problema de dimensionalidad y diseño de la recompensa. Rigurosas pruebas de convergencia son realizadas para cada algoritmo, además su desempeño es comparado con controladores clásicos de la teoría de control lineal y no lineal. La clave principal es el uso de aproximadores paramétricos y no paramétricos. Los aproximadores parámetricos utilizan el algoritmo de K-means para generar una familia de aproximadores. Por el otro lado, los aproximadores no paramétricos brindan una manera elegante de lidiar con el problema de dimensionalidad. Finalmente para el problema de diseño de la recompensa se propone modelar el problema de control como un problema de optimización bajo restricciones, el cual permite encontrar un control robusto y óptimo en términos de la peor perturbación disponible.

Optimal and Robust Control of Robot Manipulators based on Reinforcement Learning Techniques

by

M.S. José Adolfo Perrusquía Guzmán

Submitted to the Automatic Control Department
on August 2020, in partial fulfilment of the
requirements for the degree of
Doctor in Philosophy in Automatic Control

Abstract

This thesis presents reinforcement learning techniques for the design of optimal and robust controllers for non-linear systems, especially robot manipulators. Reinforcement learning is a machine learning tool that finds an optimal and adaptable controllers using only states and control measures and a reward signal. The application of reinforcement learning techniques in robots is challenging due to the dimensionality problem caused by the exponential increase in data. Furthermore, the design of the reward is a non-trivial and indispensable task in the controller performance. These problems are more visible in the design of robust controllers, where the controller must be able to compensate for external or endogenous disturbances while it maintains good performance of the closed-loop system. Therefore, in the robust control problem we require to consider new control signals and states that, in an ideal case, were not considered.

In this thesis, it is proposed to use different tools to deal with the problem of dimensionality and design of the reward. Rigorous convergence proofs are performed for each algorithm, in addition the algorithms performance are compared with classical controllers of linear and non-linear control theory. The main key is the use of parametric and nonparametric approximators. Parametric approximators use the K -means algorithm to generate a family of approximators. On the other hand, nonparametric approximators provide an elegant way to deal with the dimensionality problem. Finally, for the reward design problem, it is proposed to model the control problem as an optimization problem under constraints, which allow us to find a robust and optimal control in terms of the available worst-case disturbance.

Índice general

Introducción **1**

Motivación y Antecedentes . . . 2

Controladores clásicos basados en el modelo . . . 3

Controladores clásicos libres de modelo . . . 4

Controladores basados en Aprendizaje por Refuerzo . . . 5

Objetivos . . . 11

Contribuciones . . . 12

Estructura de la tesis . . . 12

Publicaciones . . . 15

1. Programación Dinámica y Aprendizaje por Refuerzo **19**

1.1. Procesos de Decisión de Markov (MDP) . . . 19

1.1.1. Retornos . . . 22

1.1.2. Funciones de valor y Ecuaciones de Bellman . . . 23

1.1.3. Iteración de la función de valor y la póliza . . . 27

1.2. Aprendizaje por Diferencia Temporal . . . 29

1.2.1. Q-Learning . . . 29

1.3. Conclusión . . . 36

2. Aproximación del Aprendizaje por Refuerzo **39**

2.1. Aproximadores . . . 39

2.1.1. Aproximadores paramétricos . . . 40

2.1.2. K-means Clustering . . . 41

2.1.3. Aproximadores no-paramétricos . . . 42

2.2. Aproximación de la iteración . . . 43

2.3. ADPRL de Sistemas en Tiempo Continuo . . . 48

2.4. Aproximación del RL en tiempo continuo 50
2.4.1. Diferenciación de Euler hacia atrás: Gradiente residual y TD(0) . . 52
2.5. Conclusión 57

3. Aprendizaje por Refuerzo Robusto 59
3.1. Control Robusto en Tiempo discreto 59
3.2. Aprendizaje Robusto: Tiempo discreto 61
3.2.1. Problema de Sobreestimación 61
3.2.2. Estimación del valor Esperado Mínimo 63
3.2.3. La regla de kNN 66
3.2.4. El algoritmo kNN-TD 68
3.2.5. El algoritmo kNN-TD modificado 70
3.3. Control Robusto en Tiempo continuo 78
3.4. Aprendizaje Robusto: Tiempo continuo 80
3.4.1. Algoritmos Actor-Críticos 80
3.5. Conclusión 82

4. Simulaciones y Experimentos 83
4.1. Tiempo discreto 83
4.1.1. Balanceo del sistema carro-péndulo 83
4.1.2. Control de posición de un robot planar de 2 GDL 87
4.2. Tiempo continuo 89
4.2.1. Balanceo del sistema carro-péndulo 89
4.2.2. Control de posición de un robot planar de 2 GDL 92
4.3. Discusión 94
4.4. Conclusión 94

5. Conclusiones 95
5.1. Observaciones Finales 95
5.1.1. Para Control óptimo 96
5.1.2. Para Control robusto 97
5.2. Trabajo futuro 98

A. Herramientas del Aprendizaje por Refuerzo 101
A.1. Estimación del Valor de las acciones 101
A.1.1. Implementación incremental 101

A.1.2. Seguimiento de un problema no-estacionario 102

B. Modelos de los Robots **105**

B.1. Robot planar de 2 GDL 105

B.2. Sistema carro-péndulo 106

Bibliografía **108**

Índice de Figuras

1. Esquemas de control de posición con compensación/precompensación de modelo . . . 3
2. Esquema de control de posición clásico de Robots . . . 4
3. Esquema de control basado en aprendizaje por refuerzo . . . 5
4. Interacción controlador-sistema . . . 5
5. Estructura de la tesis . . . 13

1.1. Interacción controlador-sistema . . . 21
1.2. División de los algoritmos de DP y RL . . . 26

2.1. Aproximación de la Iteración de Q . . . 44

4.1. Comparaciones de la posición del péndulo: Caso discreto . . . 85
4.2. Error medio de los métodos de RL . . . 86
4.3. Control u_t . . . 87
4.4. Robot planar de 2 GDL . . . 88
4.5. Resultados robot planar: Caso discreto . . . 89
4.6. Comparaciones de la posición del péndulo: Caso continuo . . . 91
4.7. Curvas de aprendizaje de la función Q para $\dot{x}_c = \dot{q} = 0$ . . . 91
4.8. Comparaciones de IEC . . . 92
4.9. Resultados robot planar: Caso continuo . . . 93

B.1. Robot planar de 2 GDL . . . 105
B.2. Problema de balanceo del sistema carro-péndulo . . . 107

Índice de Tablas

1.1. Relación entre un sistema de control y algoritmo de DP/RL 20
1.2. Control Óptimo de sistemas en Tiempo discreto 30
1.3. Iteración de valor y póliza para el control LQR discreto 31

2.1. Control Óptimo de sistemas en Tiempo Continuo 49

4.1. Hiperparámetros de aprendizaje del Carro-Péndulo: Caso Discreto 84
4.2. Hiperparámetros de aprendizaje del Robot Planar: Caso Discreto 88
4.3. Hiperparámetros de aprendizaje del Carro-Péndulo: Caso Continuo 90
4.4. Hiperparámetros de aprendizaje del Robot Planar: Caso Continuo 93

B.1. Parámetros del sistema Carro-Péndulo 107

Lista de Algoritmos

1.1. Iteración de la función Q 28
1.2. Iteración de la póliza a partir de funciones Q 28
1.3. Q-learning 34
1.4. Sarsa 37
2.1. Clustering K-means 42
2.2. Aproximación de Q-learning 45
2.3. Aproximación de Sarsa 45
2.4. Q-learning para sistemas en tiempo continuo 51
2.5. Aproximación de Q-learning para sistemas en tiempo continuo 53
3.1. Aprendizaje kNN-TD 69
3.2. Aprendizaje kNN-TD para espacio de acciones grandes 71
3.3. Aprendizaje kNN-TD modificado (LS-DA) 72
3.4. Aprendizaje kNN-TD modificado para espacio de acciones grandes (LS-LA) 76
3.5. Aprendizaje Actor-Crítico 81

Introducción

El aprendizaje por refuerzo (RL por sus siglas en inglés) [1] es un área del aprendizaje automático de alto interés para la comunidad científica. Se basa en teorías de la psicología donde se da una interpretación del cómo un ser vivo aprende y toma decisiones, es decir y sin perder formalidad, mediante prueba-error y recompensa-castigo[1]. Bajo este enfoque, se ha desarrollado una amplia teoría tanto de control y de cómputo para brindar una mayor autonomía a sistemas estáticos o dinámicos y reducir la intervención de algún experto. El diseño de los algoritmos de aprendizaje se basa en la teoría de control óptimo y programación dinámica donde se busca de forma iterativa un controlador que minimice o maximice una cierta función de costo. La convergencia de los algoritmos de aprendizaje por refuerzo es validada por el teorema de contracción (también conocido como teorema de punto fijo de Banach) [2] el cual demuestra la existencia y unicidad de puntos fijos del problema de aprendizaje; además proporciona herramientas para demostrar que las trayectorias del sistema convergen al punto fijo.

En la última década, el aprendizaje por refuerzo ha tenido un fuerte empuje y uso en distintas áreas de control, en especial, el control de robots. Este interés emergente se debe a que sus métodos son libres de modelo y tienen la capacidad de obtener controladores óptimos y adaptables mediante su principio de funcionamiento basado en recompensas y/o castigos. Esto brinda al robot mayor autonomía y menor intervención de un usuario/experto en comparación con los controladores clásicos[2].

Sin embargo, los principales problemas del aprendizaje por refuerzo son más notables en el contexto del control de robots manipuladores [3], en especial el problema de la dimensionalidad[3]. Esto se debe a que existe un incremento en el costo computacional por

[1]La recompensa-castigo es un valor numérico escalar proporcionado por una función de costo diseñada por el usuario que se desea minimizar o maximizar en un horizonte finito o infinito.

[2]En esta tesis, el término de controlador clásico envuelve a todo tipo de controlador cuyo diseño es obtenido mediante teoría de control lineal y no lineal.

[3]El problema de dimensionalidad consiste en un aumento exponencial en el espacio de datos disponibles provocando que los datos tengan mayor dispersión.

los múltiples grados de libertad (GDL) que posee el robot además hace que el diseño de la recompensa sea relativamente más complejo. En el caso del control robusto, el problema de dimensionalidad incrementa debido a que las perturbaciones (error de modelado, mediciones, agentes exógenos, etc.) modifican al estado y por ende al control, entonces se necesita tomar en cuenta dichos cambios provocando que se incremente el número de datos a utilizar. Para dar solución al problema de dimensionalidad se requiere el uso de aproximadores[4] y la intervención de un experto que modifique el algoritmo de aprendizaje, sin embargo, esto provoca que se pierda la autonomía natural del algoritmo de aprendizaje y sea preferible utilizar un controlador clásico.

En la presente tesis, se diseñan distintos controladores (en tiempo discreto y tiempo continuo) basados en métodos de aprendizaje por refuerzo para aplicaciones de control óptimo y robusto de robots manipuladores. Para brindar una mayor autonomía, los controladores no requieren conocimiento de la dinámica del robot o ambiente de interacción, además presentan un desempeño robusto en presencia de perturbaciones y dan solución a problemas clave del aprendizaje por refuerzo: dimensionalidad, diseño de la recompensa y sobreestimación de las acciones. Se utilizan herramientas de aprendizaje automático para el diseño de aproximadores tales como funciones Gaussianas, k-vecinos cercanos y K-means clustering, los cuales son tomados en cuenta para la demostración de convergencia de los controladores propuestos mediante el uso de la propiedad de contracción y algunas herramientas de ecuaciones diferenciales.

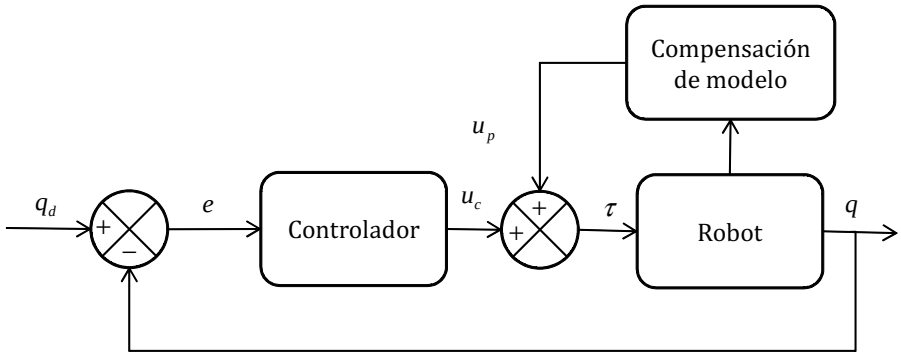
Motivación y Antecedentes

Para motivar la relevancia de la tesis es necesario conocer brevemente las limitantes de los controladores clásicos y del aprendizaje por refuerzo.

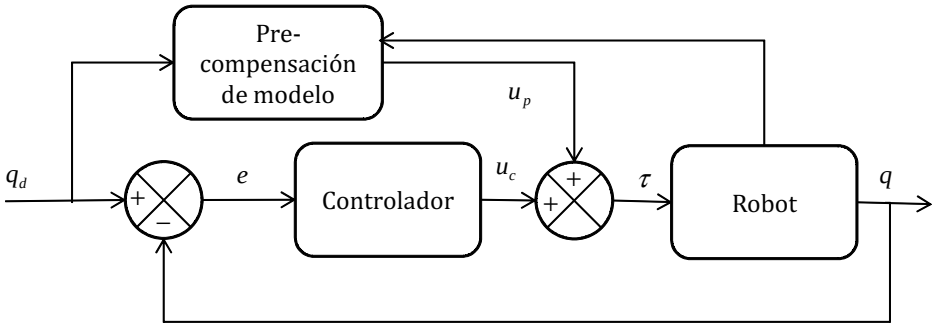
El control de robots manipuladores tiene sus inicios a principios del siglo pasado y ha sido un tema de interés para el desarrollo de teoría y aplicaciones industriales. Las principales aportaciones teóricas han sido desarrolladas mediante uso de la teoría de control lineal y no lineal, logrando que el robot sea capaz de realizar una tarea específica (control de posición, fuerza o ambas) de forma precisa y automatizada.

A continuación se presenta de forma breve los esquemas generales de algunos controladores por posición ampliamente utilizados en la literatura [4], los cuales serán retomados

[4]Un aproximador es una combinación lineal de funciónes lineales o no lineales que permite aproximar o estimar una función mediante un mapeo del espacio de parámetros al espacio de la función.



(a) Control con compensación de modelo



(b) Contro con pre-compensación de modelo

Figura 1: Esquemas de control de posición con compensación/precompensación de modelo

en los próximos capítulos para compararlos con los esquemas de aprendizaje por refuerzo propuestos.

Controladores clásicos basados en el modelo

Los controladores clásicos son basados en el modelo dinámico, es decir, se tiene conocimiento completo o parcial de la dinámica del robot. En estos casos (sin considerar perturbaciones) es posible diseñar controladores que garanticen el seguimiento perfecto de la referencia deseada al utilizar un lazo de compensación o pre-compensación de la dinámica. Este lazo compensa la dinámica del manipulador y establece una dinámica deseada más simple [5–7].

Los esquemas de control con compensación y pre-compensación de modelo en el espacio articular se observa en la Figura 1 donde q_d es la referencia de posición deseada, q es la posición articular del robot, $e = q_d - q$ es el error articular, u_p es el compensador o pre-compensador de la dinámica, u_c es el control proveniente del controlador y $\tau = u_p + u_c$ es

el par aplicado.

En el caso de que no se desee utilizar un control con compensación o pre-compensación, los esquemas clásicos (ver Figura 2) utilizan el conocimiento de la dinámica para diseñar las ganancias de los controladores. Los controladores lineales más famosos como: Proporcional-Derivativo (PD) [4], regulador cuadrático lineal (LQR por sus siglas en inglés) y el Proporcional-Integral-Derivativo (PID) [8], entre otros, han sido desarrollados para sistemas lineales donde se requiere que la dinámica del robot sea linealizada en algún punto de operación (típicamente es el origen).

El control LQR [9–11] es de suma importancia en esta tesis debido a que esta diseñado para obtener el control óptimo que minimiza una cierta función de costo, el cual es el principal motivador en el diseño de algoritmos de aprendizaje por refuerzo [12].

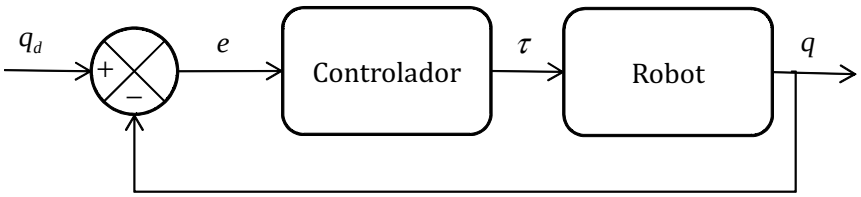


Figura 2: Esquema de control de posición clásico de Robots

Controladores clásicos libres de modelo

Cuando no se tiene conocimiento exacto de la dinámica del manipulador no es posible diseñar los controladores anteriores y al tener error de modelado entonces se pierde precisión y robustez en el sistema en lazo cerrado. Sin embargo, se han desarrollado controladores libres de modelo, tales como el control PID [13,14], modos deslizantes (SMC por sus siglas en inglés) [8, 15], redes neuronales [16], entre otros. Estos controladores son sintonizados de acuerdo a una planta en específico bajo ciertas condiciones (perturbaciones, fricción, parámetros). Cuando se presentan nuevas condiciones los controladores no presentan el mismo comportamiento llegando inclusive a la inestabilidad y por ello requieren ser sintonizados nuevamente por un experto.

Los controladores libres de modelo (lineales o no lineales) tienen buen desempeño para diferentes tareas y son relativamente sencillos de sintonizar, sin embargo, no pueden garantizar un desempeño óptimo debido a que no son sintonizados acorde a un criterio de optimización y además requieren ser nuevamente sintonizados cuando se presenta un cambio en los parámetros del sistema, referencia o una perturbación.

Controladores basados en Aprendizaje por Refuerzo

Recientemente se han utilizado técnicas de aprendizaje por refuerzo con el fin de diseñar un control óptimo y/o robusto (mediante el uso de la teoría de programación dinámica) que no requiera conocimiento de la dinámica del sistema y que además sea capaz de adaptarse a cambios internos o externos en el lazo de control (ver Figura 3).

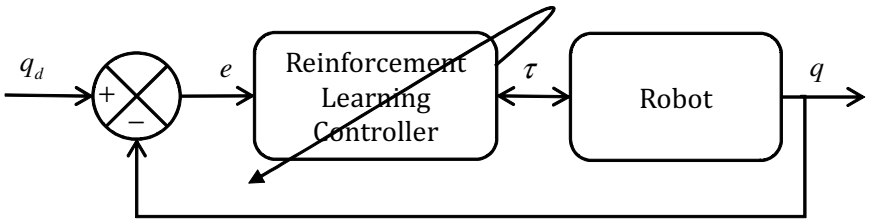


Figura 3: Esquema de control basado en aprendizaje por refuerzo

Un algoritmo de aprendizaje por refuerzo consta de un controlador que interactúa con un sistema o proceso mediante el uso de estados y acciones, y recibe recompensas acorde a una función de recompensa. El esquema general se observa en la Figura 4 donde x_t es el estado en el tiempo t, u_t es la acción tomada en el tiempo t y r_{t+1} es la recompensa obtenida en un instante de tiempo $t+1$. En el siguiente capítulo se explicará cada elemento de un algoritmo de aprendizaje por refuerzo a detalle.

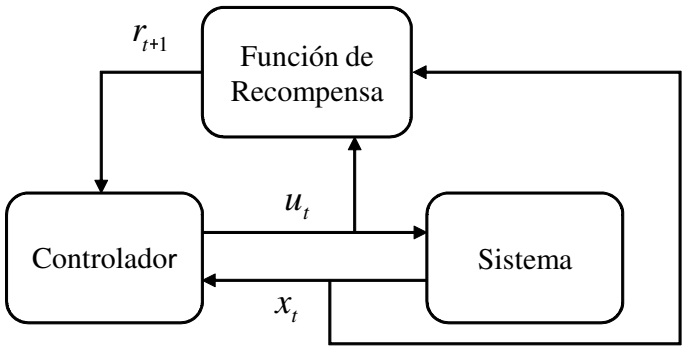


Figura 4: Interacción controlador-sistema

Al ejecutar una acción u_t en el estado x_t se obtiene un estado siguiente x_{t+1} de acuerdo a una función de transición de estados o dinámica denotada por $f(x_t, u_t)$. La calidad de cada transición es medida mediante la recompensa r_{t+1} generada por una función de recompensa $\rho(x_t, u_t)$. El desempeño del controlador es dictaminado por su póliza $h(x_t)$: un mapeo de estados a acciones, que indica que acción debe tomarse en cada estado.

El principal objetivo de los algoritmos de aprendizaje por refuerzo es minimizar o maximizar el retorno [12], que consiste en las recompensas acumuladas a lo largo de la interacción controlador-sistema. En esta tesis se consideran retornos de horizonte infinito con descuento, los cuales acumulan las recompensas obtenidas a lo largo de trayectorias "infinitas" empezando en un instante de tiempo $t = 0$ ponderando cada recompensa por un factor de descuento $\gamma \in [0, 1)$ que decrece de manera exponencial cuando el tiempo incrementa:

$$\gamma^0 r_1 + \gamma^1 r_2 + \gamma^2 r_3 + \cdots \tag{1}$$

Las recompensas dependen del curso que hayan tomado las trayectorias de estado-acción, la cual depende de la póliza utilizada:

$$x_0,\ u_0 = h(x_0),\ x_1,\ u_1 = h(x_1), \cdots .$$

El desafío principal de los algoritmos de aprendizaje por refuerzo es obtener una solución que optimiza el retorno a largo plazo utilizando únicamente información de la recompensa. Este problema se reduce a encontrar la póliza óptima, denotada por h^*, que maximice o minimice el retorno (1) para cualquier estado inicial. Una forma de obtener una póliza óptima es calculando los retornos mínimos/máximos [2] utilizando funciones de valor[5]. Por ejemplo, la función de valor Q óptima, denotada por Q^*, contiene para cada par de estado-acción (x, u) el retorno óptimo obtenido al tomar inicialmente una acción u en el estado x y después elegir las acciones óptimas a partir del segundo estado:

$$\begin{aligned} Q^*(x, u) \quad &= \gamma^0 r_1 + \gamma^1 r_2 + \gamma^2 r_3 + \cdots \\ &\text{cuando } x_0 = x,\ u_0 = u,\ \text{y las acciones óptimas son tomadas para } x_1, x_2, \ldots \end{aligned}$$

Por el otro lado, la función de valor V óptima, denotada por V^*, contiene para cada estado, el retorno óptimo obtenido al seguir una póliza óptima h^*:

$$\begin{aligned} V^*(x) &= \gamma^0 r_1 + \gamma^1 r_2 + \gamma^2 r_3 + \cdots \\ &\text{cuando } x_0 = x,\ u_0 = h^*(x_0),\ u_1 = h^*(x_1),\ u_2 = h^*(x_2). \end{aligned}$$

La función de valor Q incluye información acerca de la calidad de las transiciones. Por el contrario las funciones V solo describen la calidad de los estados. En el Capítulo 1 se abordará más a fondo las diferencias y relación de cada función de valor.

[5]Una función de valor en un sistema de control se le denomina función de costo.

La función de valor Q óptima puede ser obtenida mediante el uso de diferentes algoritmos de aprendizaje por refuerzo. Los algoritmos más famosos son conocidos como de diferencia temporal (TD por sus siglas en inglés), que realizan una diferencia entre dos funciones de valor Q evaluadas en pares de estado-acción en diferentes tiempos t y $t+1$. Esta diferencia se le conoce como *de un sólo paso* debido a que únicamente utiliza el tiempo actual y posterior.

Algunos esquemas de aprendizaje de diferencia temporal clásicos son Monte-Carlo [1], Q-learning [17], Sarsa [18], algoritmos críticos [19], entre otros. La póliza óptima puede ser obtenida al elegir en cada estado, una acción $h^*(x)$ que minimice/maximice la función de valor óptima Q^* en ese estado

$$h^*(x) \in \underset{u}{\operatorname{argmin}}\, Q^*(x,u) \quad \text{ó} \quad h^*(x) \in \underset{u}{\operatorname{argmax}}\, Q^*(x,u). \tag{2}$$

Cuando la póliza óptima es encontrada, entonces el algoritmo de aprendizaje por refuerzo seleccionará, en cada paso de tiempo, las acciones que minimicen/maximicen el retorno. A esto se le conoce como un algoritmo codicioso debido a que se explota el conocimiento adquirido. Por el otro lado, cuando la póliza óptima aún no es encontrada, entonces se dice que el algoritmo esta explorando [1].

Los algoritmos previos son diseñados para sistemas en tiempo discreto y pares de estado-acción discretos. Sin embargo, cuando los pares de estado-acción son grandes [6] o continuos, entonces los algoritmos de diferencia temporal discretos no pueden obtener la póliza óptima debido al incremento del espacio estado-acción.

Problema de dimensión

Cuando el espacio de entrada (estados y acciones) es grande o continuo, como en el caso de robots manipuladores, entonces los algoritmos de aprendizaje por refuerzo clásicos no pueden ser implementados directamente debido al incremento exponencial en el costo computacional [2,20] para explotar todo el espacio de entrada, provocando que los algoritmos no converjan a ninguna solución. Este problema es conocido como la "maldición" de dimensionalidad de los algoritmos de aprendizaje automático. El problema de dimensión incrementa en robots manipuladores debido a que cada GDL tiene su propio espacio de entrada [21,22] provocando que aumente el número posible de combinaciones que se pue-

[6] En esta tesis, se utilizarán los acrónimos LS-DA para hablar de espacios de estados grandes y espacios de acciones discretas, y LS-LA para espacios estados y acciones grandes. Se utiliza el acrónimo CT para abreviar tiempo continuo.

de llegar a tener entre todos los posibles estados y acciones o controles. Otro agravante al problema de dimensionalidad es la presencia de perturbaciones, debido a que se debe considerar nuevos estados y controles.

Para resolver el problema de dimensionalidad se ha utilizado métodos de aprendizaje por refuerzo combinados con técnicas basadas en modelo [23–25], es decir, que se tiene conocimiento o una aproximación de la función de transición de estados o dinámica f. Estos métodos de aprendizaje son muy populares en una clase especial de algoritmos llamados búsqueda de la póliza [3,26–32], los cuales buscan directamente la póliza en lugar de obtenerla a partir de de la función de valor óptima Q^*. Sin embargo, estos métodos requieren conocimiento del modelo para disminuir la dimensión del espacio de entrada. Para evitar el uso del modelo del robot se cuenta con una amplia diversidad de algoritmos libres de modelo utilizando metodologías similares a los algoritmos en tiempo discreto. La clave principal de estos algoritmos es el diseño de una recompensa adecuada y el uso de aproximadores, los cuales son funciones simples encargadas de disminuir el costo computacional de los algoritmos de aprendizaje por refuerzo en presencia de un espacio de entrada grande o continuo. Estos aproximadores se utilizan para parametrizar la función de valor, la póliza o ambas, mediante un vector de parámetros θ y un vector de funciones Φ diseñadas por el usuario. Por ejemplo, la función de valor Q puede ser aproximada por:

$$\widehat{Q}(x, u; \theta) = \Phi^\top(x, u)\theta. \tag{3}$$

Esta parametrización se le conoce como aproximación del aprendizaje por refuerzo, el cual será discutido detalladamente en el Capítulo 2. La aproximación del aprendizaje por refuerzo encuentra el valor del vector de parámetros óptimos θ^* tal que la parametrización $\Phi^\top\theta^*$ se aproxime a la función estimada óptima (función de valor $\widehat{Q}^* \approx Q^*$ o póliza $\widehat{h}^* \approx h^*$).

Los aproximadores más simples radican en la disminución del espacio de entrada ya sea utilizando métodos "a mano" [33–39] donde se buscan regiones que tengan buen desempeño (la recompensa sea minimizada o maximizada) y acelere el tiempo de aprendizaje. Otros métodos aprenden mediante los datos de entrada (es similar a los algoritmos de aprendizaje en tiempo discreto), sin embargo, el tiempo de aprendizaje aumenta [40,41]. Otras técnicas se basan en acciones previamente establecidas de forma secuencial y relacionada, es decir, se tienen definidas las acciones que se deben tomar en cada instante de tiempo con el fin de que por sí solas realicen una tarea sencilla [42–46]. Los problemas principales de los métodos basados en la disminución del espacio es que requiere a un experto para

obtener las regiones (donde la recompensa es mínima o máxima) y el conjunto de acciones predefinidas que permitan acelerar el tiempo de aprendizaje.

Para que los algoritmos de aprendizaje aprendan únicamente de los datos de entrada sin intervención de un experto se utiliza una combinación lineal de aproximadores como se muestra en (3). Algunos de los aproximadores más utilizados en el control de robots son inspirados en morfología humana [47,48], redes neuronales [49–51], modelos locales [48,52] y procesos de regresión Gaussiana [53–56]. El éxito de estos aproximadores se debe a la elección adecuada de sus parámetros e hiperparámetros, el cual es un trabajo difícil para cualquier problema de control y que a su vez requiere una mínima pero importante intervención de un experto.

En esta tesis se propone utilizar dos diferentes aproximadores. El primer aproximador utiliza funciones radiales básicas (RBF), las cuales son funciones localizables que parametrizan la función de valor y/o póliza usando p funciones radiales y p parámetros. La aproximación del aprendizaje por refuerzo busca los p parámetros óptimos tal que la parametrización se aproxime a la función estimada (función de valor o póliza); dando así una posible solución al problema de dimensionalidad. Debido a que son funciones localizables, se utiliza el algoritmo de K-means para ubicar el centro de cada RBF permitiendo generar una familia de aproximadores. Esta propuesta es válida para cualquier aproximador localizable. El segundo aproximador se basa en la regla de k-vecinos cercanos, el cual es un aproximador no paramétrico que aproxima la función de valor usando únicamente k vecinos. Este aproximador permite trabajar con algoritmos con variantes de espacios grandes (LS-DA y LS-LA).

Problema de diseño de la recompensa

Un punto de suma importancia es el diseño de la recompensa $\rho(x, u)$. Un mal diseño de la recompensa puede implicar largo tiempo de aprendizaje, convergencia a soluciones erróneas o que simplemente que el algoritmo nunca converja a alguna solución. Por el otro lado, el diseño adecuado de la recompensa ayuda al algoritmo de aprendizaje a encontrar la póliza óptima de forma más rápida. A este problema se le conoce como la "maldición" del diseño de recompensa [57]. Cuando se utilizan los métodos libres de modelo, la recompensa debe ser diseñada de tal forma que se adapte a cambios en el sistema y posibles errores, lo cual es sumamente útil en problemas de control robusto donde se requiera que el controlador sea capaz de compensar las perturbaciones o acotar las perturbaciones para presentar un desempeño óptimo.

En esta tesis, se propone diseñar la recompensa $\rho(x, u)$ como un problema de optimización para problemas de control óptimo y se diseña como un problema de optimización bajo restricciones para problemas de control robusto. El problema de optimización se propone como funciónes cuadráticas en términos del estado y control. Las restricciones vienen dadas por cotas superiores del estado, control y perturbación. Esta modificación permite al algoritmo de aprendizaje por refuerzo aprender pólizas h^* robustas y casi-óptimas para sistemas con perturbaciones iguales o menores a la cota superior impuesta por la restricción.

Problema de sobreestimación de las acciones

El problema de sobreestimación de las acciones ocurre en diversos algoritmos de aprendizaje por refuerzo por utilizar el operador mín o máx ya sea en la regla de actualización, e.g. Q-learning, o por medio de algún método de selección de acciones codiciosas, e.g. el método ε-codicioso. Al utilizar el comando mín / máx para aproximar la función de valor óptima Q^* se puede llegar a obtener valores sesgados cuando se tienen muestras que no son independientes e idénticamente distribuidas (i.i.d. por sus siglas en inglés). Ese sesgo afecta directamente a la obtención de la póliza óptima y por lo tanto se sobreestiman las acciones a utilizar en cada estado [17].

Para dar solución al problema de sobreestimación, algunos autores propusieron utilizar una modificación al algoritmo de Q-learning conocido como Q-learning doble, el cual esta basada en la regla de actualización de Q-learning y el uso de una técnica llamada doble estimador. Esta técnica utiliza dos estimadores: uno para encontrar la función de valor óptima Q^* y otro estimador para encontrar la póliza óptima h^*. Esto permite obtener estimados sin sesgo y evitar la sobrestimación, sin embargo su diseño es para sistemas discretos. Otra alternativa para dar solución al problema de sobreestimación es mediante el uso de algoritmos actor críticos (AC por sus siglas en inglés), los cuales tienen un funcionamiento similar al doble estimador porque utilizan dos aproximadores. Un aproximador parametriza una función de valor $V(x)$ y el otro aproximador parametriza la póliza $h(x)$.

En esta tesis, se propone utilizar la técnica del doble estimador en conjunto con el aproximador de k vecinos cercanos para evitar el problema de sobrestimación para problemas con espacios grandes. Además se utiliza el aproximador de RBFs en conjunto con el algoritmo actor-crítico para problemas con espacios continuos.

En resumen, en esta tesis se propone utilizar técnicas de aprendizaje por refuerzo para aplicaciones de control óptimo y robusto de robots manipuladores que den solución a los

problemas de dimensionalidad, diseño de la recompensa y problema de sobreestimación de las acciones con mínima intervención del usuario. Además se compara su desempeño con controladores clásicos basados en el modelo y libres de modelo con el fin de observar su versatilidad, adaptabilidad y robustez. Los algoritmos de aprendizaje por refuerzo son diseñados en tiempo discreto y continuo utilizando herramientas de programación dinámica y aproximadores brindados de la teoría de aprendizaje automático tales como K-means clustering, funciones Gaussianas RBFs y k vecinos cercanos. La convergencia a la solución óptima, casi óptima o robusta de cada algoritmo es demostrada mediante el uso de la propiedad de contracción y herramientas de ecuaciones diferenciales.

Objetivos

Diseñar controladores óptimos y robustos para el control de robots manipuladores basados en herramientas de aprendizaje por refuerzo que requieran una mínima intervención de un experto en presencia de perturbaciones y sin conocimiento de la dinámica del manipulador. Además de proponer soluciones a los problemas habituales del aprendizaje por refuerzo (dimensionalidad, diseño de recompensa y sobreestimación de las acciones).

El objetivo principal es dividido en objetivos particulares para facilitar su cumplimiento. Los objetivos particulares a realizar en la presente tesis son:

1. Investigar el estado del arte del aprendizaje por refuerzo aplicado en el control de robots manipuladores para identificar los problemas específicos en el aprendizaje por refuerzo y las soluciones propuestas por otros autores.

2. Utilizar herramientas de aprendizaje automático y programación dinámica para resolver el problema de dimensionalidad y diseño de la recompensa que facilite el diseño de los controladores.

3. Demostrar la convergencia de los algoritmos de aprendizaje por refuerzo propuestos mediante el uso de la propiedad de contracción.

4. Proponer controladores óptimos y robustos basados en los métodos de aprendizaje por refuerzo que garanticen el objetivo de control en presencia de perturbaciones y sin conocimiento de la dinámica del manipulador.

5. Comparar el desempeño de los controladores propuestos con controladores clásicos de la teoría de control considerando la presencia y ausencia de perturbaciones.

Contribuciones

Las contribuciones de esta tesis se presentan a continuación:

1. Se utilizan aproximadores basados en funciones RBFs y k vecinos cercanos para lidiar con el problema de dimensionalidad.

2. Se propone utilizar un aproximador basado en RBFs para el diseño de los algoritmos de Q-learning y Actor-Críticos (AC) en tiempo discreto y tiempo continuo.

3. Se propone el uso del algoritmo de K-means para generar una familia de aproximadores paramétricos basado en funciones RBF o cualquier función localizable.

4. Se propone utilizar el método del doble estimador en conjunto con el aproximador de k vecinos cercanos para el diseño de algoritmos de aprendizaje por refuerzo sin sesgo y que las pólizas eviten sobrestimar las acciones de control en espacios grandes (LS-DA y LS-LA).

5. Cada algoritmo de aprendizaje por refuerzo proporcionado en esta tesis es analizado y demostrado analíticamente mediante el uso de la propiedad de contracción y herramientas de ecuaciones diferenciales.

6. La recompensa de los algoritmos de aprendizaje es diseñada como un problema de optimización bajo restricciones para el diseño de controladores robustos bajo perturbaciones.

Estructura de la tesis

La tesis es escrita para introducir la teoría del aprendizaje por refuerzo de forma general aplicado a sistemas lineales y no-lineales. La tesis se estructura de acuerdo al diagrama de la Figura 5:

- **Capítulo 1. Programación Dinámica y Aprendizaje por Refuerzo**. Dentro de este capítulo se menciona los elementos que comprenden un algoritmo de aprendizaje en el enfoque de programación dinámica y el de aprendizaje por refuerzo. Se presenta el diseño del algoritmo más popular de programación dinámica conocido como regulador cuadrático lineal (LQR) en términos de la función de valor, recompensa y póliza. Se establece una clasificación de los algoritmos de aprendizaje por

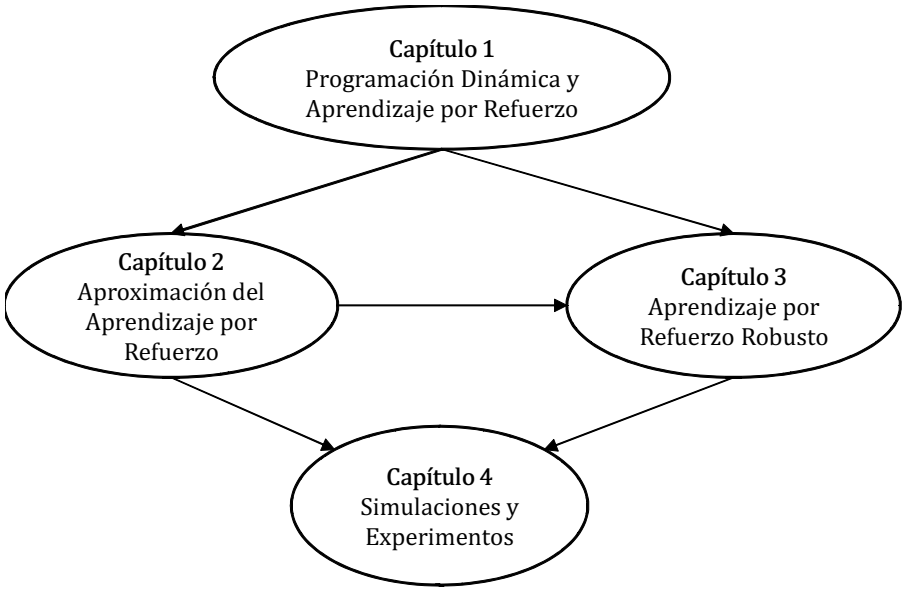


Figura 5: Estructura de la tesis

refuerzo de acuerdo a la forma en que encuentran la póliza óptima. Se proporcionan dos algoritmos de diferencia temporal (TD) conocidos como Q-learning y Sarsa, los cuales encuentran la función de valor óptima Q^* de manera iterativa; mientras que la póliza óptima h^* es obtenida mediante la selección de la acción u que minimice la función Q^* en un cierto estado x. Se presentan dos métdosos de exploración y explotación conocidos como ε-codicioso y softmax, los cuales servirán para explorar el espacio de estados-acciones en búsqueda de la póliza óptima o explotar el aprendizaje adquirido para minimizar el retorno. La convergencia de cada algoritmo es demostrada mediante la propiedad de contracción.

- **Capítulo 2. Aproximación del Aprendizaje por Refuerzo**. En este capítulo se dan herramientas para la aproximación del aprendizaje por refuerzo para lidiar con el maldición de la dimensionalidad. Se propone utilizar un aproximador basado en una combinación lineal de RBFs. Se utiliza el algoritmo de K-means para ubicar el centroide de forma aleatoria de cada RBF, permitiendo generar una familia de aproximadores. Este enfoque es válido para cualquier función localizable. Los algoritmos de aprendizaje por refuerzo del capítulo pasado, i.e., Q-learning y Sarsa, son aproximados mediante una parametrización dada por un vector de funciones RBF Φ y un vector de parámetros θ. Estos algoritmos encuentran de manera iterativa el vector de parámetros óptimos θ^* tal que la parametrización aproxime

a la función de valor $\Phi^\top \theta^* \approx Q$; mientras que la póliza óptima es encontrada mediante la selección de la acción que minimice la función de valor aproximada, i.e, $h^* \in \text{argmin}_u \widehat{Q}^*(x, u; \theta)$. Además, se brinda la teoría para diseñar algoritmos de programación dinámica (LQR) y aprendizaje por refuerzo en tiempo continuo. Se proporciona el algoritmo de Q-learning en tiempo continuo y su aproximación basado en un algoritmo tipo gradiente y la parametrización propuesta. Se proporciona la convergencia de los algoritmos de aprendizaje por refuerzo aproximado en tiempo discreto y tiempo continuo utilizando la propiedad de contracción y herramientas de ecuaciones diferenciales.

- **Capítulo 3: Aprendizaje por Refuerzo Robusto**. En este capítulo se muestra una breve teoría para el diseño de controladores óptimos y robustos en tiempo discreto y tiempo continuo. Se propone diseñar la recompensa como un problema de optimización con restricciones para el diseño de controladores robustos en presencia de perturbaciones. Las restricciones vienen dadas por cotas superiores del estado, control y perturbación, de tal forma que el algoritmo de aprendizaje sea capaz de compensar las perturbaciones que sean igual o menor a la cota superior dada y manteniendo un desempeño cuasi-óptimo. Todos los algoritmos de este capítulo son diseñados usando esta recompensa. En sistemas en tiempo discreto, se propone utilizar un aproximador basado en la regla de k-vecinos cercanos y el método de doble estimador para solucionar el problema de sobreestimación de las acciones en un espacio de entrada LS-DA y LS-LA. Los algoritmos LS-DA y LS-LA encuentran la función de valor óptima/robusta de forma iterativa usando dos estimadores. Un estimador se encarga en estimar la función de valor Q^* óptima/robusta y el otro estimador encuentra la póliza óptima/robusta h^*. En tiempo continuo, se propone utilizar el algoritmo de Q-learning o el algoritmo Actor-Crítico para evitar la sobreestimación de las acciones. De manera similar, el algoritmo Actor-Crítico utiliza dos aproximadores. Un aproximador parametriza la función de valor V mientras que el otro aproximador parametriza a la póliza h. El algoritmo de aprendizaje encuentra el vector de parámetros óptimos de cada aproximador tal que $\widehat{V} \approx V$ y $\widehat{h} \approx h$. Convergencia de los algoritmos es dada mediante la propiedad de contracción.

- **Capítulo 4: Simulaciones y Experimentos**. Dentro de este capítulo se expone el funcionamiento óptimo y robusto de los algoritmos de aprendizaje propuestos (tiempo discreto y continuo) en los Capítulos 2 y 3 en dos problemas: la estabilización

de un sistema carro-péndulo en presencia de perturbaciones paramétricas y el control de posición de un robot planar de 2 GDL en presencia de una perturbación acotada y variante en el tiempo. Además se compara el desempeño de los algoritmos de aprendizaje propuestos con controladores clásicos tales como LQR, PID y SMC, con el fin de observar su robustez y versatilidad ante perturbaciones.

- **Capítulo 5: Conclusiones**. Este capítulo brinda las conclusiones finales de la presente tesis enfocado en el diseño de controladores óptimos y robustos basado en los resultados obtenidos dentro de esta tesis. Además se brindan algunas áreas de oportunidad teórico y experimentales para trabajo futuro.

Publicaciones

- **Revista Internacional**.

  1. A. Perrusquía, W. Yu, A. Soria, "Position/force control of robot manipulators using Reinforcement Learning", *Industrial Robot: The International Journal of Robotics Research and Application*, vol. 46, no. 2, pp. 267-280, 2019. (**Factor de Impacto: 1.190**).
  2. A. Perrusquía, W. Yu, "Human-in-the-loop control using Euler Angles", *Journal of Intelligent & Robotic Systems*, vol. 97, no. 1, pp. 271-285, 2020. (**Factor de Impacto: 2.259**).
  3. W. Yu, A. Perrusquía, "Simplified Stable Admittance control using end-effector orientations", *International Journal of Social Robotics*, DOI: 10.1007/s12369-019-00579-y. (**Factor de Impacto: 2.516**).
  4. A. Perrusquía, W. Yu, "Robust control under worst-case uncertainty for unknown nonlinear systems using modified reinforcement learning", *Journal of Robust and Nonlinear Control*, vol. 30, no. 7, pp. 2920-2036, 2020. (**Factor de Impacto: 3.953**).
  5. A. Perrusquía, W. Yu, "Robot position/force control in unknown environment using hybrid reinforcement learning", *Cybernetics and Systems*, vol. 51, no. 4, pp.542-560, 2020. (**Factor de Impacto: 1.197**).
  6. W. Yu, A. Perrusquía, X. Li, "Multi-agent Reinforcement Learning for Redundant Robot Control in Task-space", *International Journal of Machine Lear-*

ning and Cybernetics, DOI: 10.1007/s13042-020-01167-7. **(Factor de Impacto: 3.844).**

7. A. Perrusquía, W. Yu. Discrete-time $\mathcal{H}_2$ Neural Control Using Reinforcement Learning. *IEEE Transactions on Neural Networks and Learning Systems*. 2020. **(Factor de Impacto: 11.683)**.
8. A. Perrusquía, W. Yu. Neural $\mathcal{H}_2$ control Using Continuous time Reinforcement Learning. *IEEE Transactions on System, Man, and Cybernetics-Part B*. **(Factor de Impacto: 10.21)**.
9. A. Perrusquía, W. Yu. Continuous time reinforcement learning for robust control under worst-case uncertainty. *International Journal of Systems Science*. **(Factor de Impacto: 2.185).**

- **Congreso Internacional**.

1. A. Perrusquía, W. Yu, A. Soria, R. Lozano, "Stable admittance control without inverse kinematics", *20th IFAC World Congress (IFAC17)*, Toulouse, France, pp. 16402-16407, 2017.
2. A. Perrusquía, W. Yu, "Task space human-robot interaction using angular velocity Jacobian", *2019 International Symposium on Medical Robotics (ISMR19)*, Atlanta, USA, 2019.
3. A. Perrusquia, W. Yu, A. Soria, "Large space dimension Reinforcement Learning for Robot Position/Force Discrete Control", *6th International Conference on Control, Decision and Information Technologies (CoDIT19)*, Paris France, 2019.
4. A. Perrusquía, W. Yu, A. Soria, "Optimal contact force of Robots in Unknown Environments using Reinforcement Learning and Model-free controllers", *2019 16th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE19)*, Mexico city, Mexico, 2019.
5. A. Perrusquía, W. Yu, X. Li, "Impedance control without environment model by Reinforcement Learning", *10th International Conference on Intelligent Control and Information Processing (ICICIP 2019)*, Marrakesh Morocco, 2019.
6. A. Perrusquía, W. Yu, "Neural $\mathcal{H}_2$ control using reinforcement learning for unknown nonlinear systems", *IEEE World Congress on Computational Intelligence (WCCI 2020)*, Glasgow UK, 2020.

7. A. Perrusquía, W. Yu, X. Li, "Redundant Robot Control Using Multi Agent Reinforcement Learning", *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE 2020)*. Zoom Meeting, 2020.

8. A. Perrusquía, W. Yu, X. Li, "Robust Control in the Worst Case Using Continuous Time Reinforcement Learning", *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2020.

Capítulo 1

Programación Dinámica y Aprendizaje por Refuerzo

En este capítulo se introduce el problema y los conceptos básicos de la programación dinámica y el aprendizaje por refuerzo. Estos conceptos son de suma utilidad para el diseño de los algoritmos de aprendizaje que serán presentados en los siguientes capítulos. Se aborda el caso determinista usando sistemas en tiempo discreto. Se proporcionan algunos de los algoritmos más utilizados del aprendizaje por refuerzo que no requieren modelo del sistema/ambiente y que son diseñados para un espacio de estado-acción discreto. Además se brindan algunas técnicas para la exploración del espacio estado-acción.

1.1. Procesos de Decisión de Markov (MDP)

Antes de enunciar los elementos principales que componen un algoritmo de programación dinámica (DP por sus siglas en inglÍes) y de aprendizaje por refuerzo (RL por sus siglas en ingés), es necesario establecer una relación conceptual de un sistema de control y un algoritmo de DP/RL como se observa en la Tabla 1.1, donde el subíndice t indica el valor de la señal en el instante de tiempo t.

Con base a los conceptos de la Tabla 1.1 se tienen las siguientes observaciones.

Observación 1. *En un sistema de control, un controlador interactúa con un sistema o proceso mediante mediciones del estado y la aplicación de una señal de control.*

Observación 2. *En un algoritmo de aprendizaje DP/RL, un agente interactúa con su ambiente mediante uso de tres señales: una señal de estado del ambiente, una señal de*

Tabla 1.1: Relación entre un sistema de control y algoritmo de DP/RL

Señal	**Sistema de Control**	**Algoritmo de DP/RL**
	Controlador	Agente
	Sistema/Proceso	Ambiente
x_t	Estado	Estado
u_t	Señal de control	Acción
$f(x_t)$	Dinámica	Función de transición
$\rho(x,u) = r_{t+1}$	Función de utilidad	Recompensa inmediata
$J(x_t) = V(x_t)$	Función de costo	Función de valor
$h(x_t)$	Control	Póliza

acción que permite al agente modificar el estado ambiente y una señal de recompensa escalar que provee una realimentación al agente de su desempeño.

De las definiciones anteriores se puede observar una clara relación entre cada concepto en el diseño de controladores óptimos. Se desea encontrar la señal de control que maximice o minimice la función de costo acorde a una función de utilidad, es decir, se desea maximizar o minimizar la siguiente ecuación

$$J(x_t) = \underbrace{\sum_{i=t}^{\infty} \overbrace{\rho(x_i, u_i)}^{\text{función de utilidad}}}_{\text{función de costo}}.$$

Por el otro lado, un algoritmo de DP/RL por su naturaleza de diseño requiere el uso de una función de valor para minimizar o maximizar el total de recompensas obtenidas. En cada instante de tiempo, el controlador/agente recibe mediciones del estado del sistema/ambiente y aplica una acción/control el cual provoca la transición a un nuevo estado. El controlador/agente recibe esta nueva medición y el ciclo se repite. A lo largo de esta tesis se utilizarán dichos conceptos en comunión para facilitar la lectura. En especial, se utilizará la palabra acción y recompensa de forma indiferente entre un sistema de control y algoritmo de DP/RL. En esta tesis se desea minimizar la señal de control.

Un concepto exclusivo de los algoritmos de DP/RL es el de póliza. El desempeño del controlador depende de su póliza: una función que mapea del espacio de estados al espacio de las acciones. En un sistema de control la póliza es equivalente al control utilizado, e.g., un control PD, PID, LQR, etc. La evolución del proceso es descrito por su dinámica o función de transición, la cual determina cómo los estados cambian debido a la aplicación

de las distintas acciones del controlador. La dinámica o transición de estados puede ser determinista o estocástica. En el caso determinista, al tomar una acción en un cierto estado siempre resulta en el mismo estado siguiente, mientras que en el caso estocástico, el estado siguiente es una variable aleatoria [2]. En esta tesis, la transición de estados son deterministas. Una forma sencilla de modelar un problema de DP/RL es mediante el uso de un proceso de decisión de Markov (MDP por sus siglas en inglés). Un MDP modela la interacción del controlador-sistema utilizando un espacio de estados, un espacio de acciones, una función de recompensa y la dinámica/función de transición del sistema. En la Figura 1.1 (repetida de la Figura 4) se proporciona el esquema de la interacción controlador-sistema.

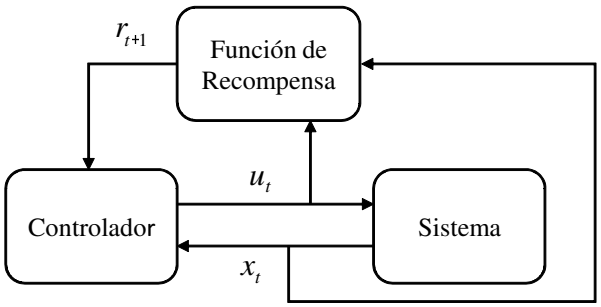


Figura 1.1: Interacción controlador-sistema

Un MDP determinista [2] es definida por la tupla (X, U, f, ρ), donde X es el espacio de estados, U es el espacio de las acciones que puede efectuar el controlador, f es la dinámica del sistema y ρ es la función de recompensa (que evalúa el desempeño del control aplicado). Cuando la acción u_t es aplicada en el estado x_t en el instante de tiempo t, el estado cambia a x_{t+1}, de acuerdo a la dinámica $f : X \times U \to X$

$$x_{t+1} = f(x_t, u_t). \tag{1.1}$$

Al mismo tiempo, el controlador recibe una señal de recompensa escalar r_{t+1}, de acuerdo a la función de recompensa $\rho : X \times U \to \mathbb{R}$

$$r_{t+1} = \rho(x_t, u_t). \tag{1.2}$$

La recompensa evalúa el efecto inmediato de la acción u_t de pasar del estado x_t a x_{t+1}, pero en general no menciona nada acerca de sus efectos a largo plazo.

El controlador elige las acciones acorde a su póliza $h : X \to U$, usando

$$u_t = h(x_t). \tag{1.3}$$

Una condición suficiente para determinar tanto el siguiente estado x_{t+1} y la recompensa r_{t+1} es conocer f y ρ, el estado actual x_t y la acción actual u_t. Esto se le conoce como la propiedad de Markov, que menciona que el siguiente estado solo depende del estado actual y no de la secuencia de estados que lo preceden.

1.1.1. Retornos

En la teoría de DP/RL, el principal objetivo es encontrar la póliza óptima que minimice el retorno, que consiste en la recompensa acumulada en el transcurso de la interacción. En el caso más simple, el retorno de horizonte infinito esta dado por la siguiente expresión:

$$R^h(x_t) = \sum_{i=t}^{\infty} r_{i+1} = \sum_{i=t}^{\infty} \rho(x_i, h(x_i)) \tag{1.4}$$

donde $x_{t+1} = f(x_t, h(x_t))$ para $t \geq 0$. Este método tiene sentido en aplicaciones donde existe una noción natural de un tiempo final, esto es, cuando la interacción controlador-sistema se interrumpe naturalmente en subsecuencias, mejor conocidos como episodios. Cada episodio termina en un estado llamado estado final, seguido de una reiniciación hacia un estado inicial. Sin embargo, la mayoría de los casos los problemas de DP/RL tienen tareas continuas, es decir, la interacción controlador-sistema no se interrumpe naturalmente en episodios, en cambio la interacción controlador-sistema continua creciendo sin limite.

Otro tipo de retorno usa el concepto de descuento, donde el controlador busca seleccionar acciones o controles tal que la suma de las recompensas descontadas que recibe sea mínima. El retorno con horizonte infinito descontado esta dado por:

$$R^h(x_t) = \sum_{i=t}^{\infty} \gamma^{i-t} r_{i+1} = \sum_{i=t}^{\infty} \gamma^{i-t} \rho(x_i, h(x_i)) \tag{1.5}$$

donde $\gamma \in [0, 1)$ es un factor de descuento. El factor de descuento puede ser interpretado como una medida de que tanto el controlador considerará las nuevas recompensas. El descuento asegura que el retorno será siempre acotado si las recompensas son acotadas. Nótese que si $\gamma = 1$ se obtiene el retorno sin descuento (1.4).

La selección de γ generalmente involucra una disyuntiva entre la calidad de la solución

y la velocidad de convergencia del algoritmo de DP/RL. Si γ es muy pequeño, la solución podría ser insatisfactoria porque no toma en cuenta las suficientes recompensas después de un largo número de instantes de tiempo.

Retorno de una muestra específica

El objetivo es utilizar el descuento como una probabilidad de finalización parcial o equivalentemente un grado parcial de finalización. Para cualquier $\gamma \in [0, 1)$, un retorno $R^h(x_t)$ que tiene una finalización parcial de sólo un paso tiene una ponderación $1 - \gamma$ produciendo un retorno de sólo la primer recompensa r_1; para una finalización parcial de dos pasos se tiene una ponderación $(1 - \gamma)\gamma$ produciendo un retorno de $r_1 + r_2$ y así sucesivamente. La ponderación de finalización en tres pasos es $(1-\gamma)\gamma^2$, con γ^2 que refleja que la terminación no ocurrió en los primeros dos pasos. El retorno parcial es conocido como retorno parcial plano:

$$\bar{R}_{t:i} = r_{t+1} + r_{t+2} + \cdots + r_i, \;\; 0 \leq t < i \leq T \tag{1.6}$$

donde *plano* significa la ausencia de descuento, y *parcial* denota que estos retornos no se extienden infinitamente y se detienen en el instante de tiempo i, conocido como horizonte (T es el tiempo de terminación del episodio). El retorno completo convencional $R^h(x_t)$ puede ser visto como una suma de retornos parciales planos como se muestra a continuación:

$$\begin{aligned} R^h(x_t) =& r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots + \gamma^{T-t-1} r_T \\ =& (1-\gamma) r_{t+1} + (1-\gamma)\gamma(r_{t+1} + r_{t+2}) + (1-\gamma)\gamma^2(r_{t+1} + r_{t+2} + r_{t+3}) \\ & \vdots \\ & + (1-\gamma)\gamma^{T-t-2}(r_{t+1} + r_{t+2} + \cdots + r_{T-1}) + \gamma^{T-t-1}(r_{t+1} + r_{t+2} + \cdots + r_T) \\ =& (1-\gamma) \sum_{i=t+1}^{T-1} \gamma^{i-t-1} \bar{R}_{t:i} + \gamma^{T-t-1} \bar{R}_{t:T} \end{aligned} \tag{1.7}$$

1.1.2. Funciones de valor y Ecuaciones de Bellman

Existen dos tipos de funciones de valor: de estado-acción (funciones Q) y de estado (funciones V).

Definición 1. *La función Q de una póliza h, $Q^h : X \times U \to \mathbb{R}$, es escrita como la suma*

descontada de recompensas al tomar la acción u en el estado x siguiendo la póliza h.

$$Q^h(x,u) = \sum_{i=t}^{\infty} \gamma^{i-t} \rho(x_i, u_i) \tag{1.8}$$

donde $(x_t, u_t) = (x, u), x_{t+1} = f(x_t, u_t)$ *para* $t \geq 0$, *y* $u_t = h(x_t)$ *para* $t \geq 1$.

La expresión anterior puede ser escrita como:

$$Q^h(x,u) = \rho(x,u) + \gamma \sum_{i=t+1}^{\infty} \gamma^{i-t-1} \rho(x_i, h(x_i)) \tag{1.9}$$

$$= \rho(x,u) + \gamma R^h(f(x,u)) \tag{1.10}$$

donde el retorno (1.5) es utilizado. La función Q óptima es definida como la función Q mínima que puede ser obtenida por cualquier póliza:

$$Q^*(x,u) = \min_h Q^h(x,u) \tag{1.11}$$

Cualquier póliza h^* que elige en cualquier estado una acción con el menor y óptimo valor de Q, es decir, satisface

$$h^*(x) \in \operatorname*{argmin}_u Q^*(x,u) \tag{1.12}$$

es óptima (minimiza el retorno). En general, una póliza h que satisface:

$$h(x) = \operatorname*{argmin}_u Q(x,u) \tag{1.13}$$

se dice que es codiciosa en Q. Las funciones Q^h y Q^* son caracterizadas de forma recursiva por las ecuaciones de Bellman. Las ecuaciones de Bellman para Q^h puede ser obtenida de (1.9), como sigue:

$$\begin{aligned} Q^h(x,u) &= \rho(x,u) + \gamma \sum_{i=t+1}^{\infty} \gamma^{i-t-1} \rho(x_i, h(x_i)) \\ &= \rho(x,u) + \gamma \left[\rho(f(x,u), h(f(x,u))) + \gamma \sum_{i=t+2}^{\infty} \gamma^{i-t-2} \rho(x_i, h(x_i)) \right] \\ &= \rho(x,u) + \gamma Q^h(f(x,u), h(f(x,u))) \end{aligned} \tag{1.14}$$

La ecuación óptima de Bellman caracterizada por Q^*, establece que la función de valor

óptima de una acción u tomada en el estado x es igual a la suma de las recompensas inmediatas y el valor óptimo descontado obtenido por la mejor acción en el siguiente estado:

$$Q^*(x,u) = \rho(x,u) + \gamma \min_{u'} Q^*(f(x,u),u') \tag{1.15}$$

Definición 2. *La función V de una póliza h, $V^h : X \to \mathbb{R}$, es el retorno obtenido de iniciar en un cierto estado y seguir h. La función V puede ser calculada a partir de la función Q siguiendo la póliza h:*

$$V^h(x) = R^h(x) = Q^h(x, h(x)) \tag{1.16}$$

La función V óptima es la mejor función V que puede ser obtenida por cualquier póliza, y puede ser calculada a partir de la función Q óptima:

$$V^*(x) = \min_h V^h(x) = \min_u Q^*(x,u) \tag{1.17}$$

Cualquier póliza óptima h^* puede ser calculada a partir de V^*, usando el hecho que satisface:

$$h^*(x) \in \operatorname*{argmin}_u [\rho(x,u) + \gamma V^*(f(x,u))] \tag{1.18}$$

La función Q incluye información acerca de la calidad de las transiciones. Por el contrario, las funciones V solo describen la calidad de los estados; para inferir la calidad de las transiciones, se deben de tomar en cuenta de manera explícita. Las funciones V: V^h y V^* satisfacen las siguientes ecuaciones de Bellman:

$$V^h(x) = \rho(x, h(x)) + \gamma V^h(f(x, h(x))) \tag{1.19}$$

$$V^*(x) = \min_u [\rho(x,u) + \gamma V^*(f(x,u))] \tag{1.20}$$

Los algoritmos de DP y RL se diferencian en la dependencia del modelo. Los algoritmos de DP requieren conocimiento del modelo del proceso o sistema con el que se interactúa, mientras que los algoritmos de RL son libres de modelo y sólo requieren mediciones del estado y control. A su vez, los algoritmos de DP y RL puede ser divididos en tres subclases, según el camino elegido para encontrar la póliza óptima. Estas tres subclases son: iteración de la función de valor, iteración de la póliza y búsqueda de la póliza, y son caracterizadas por:

- *Iteración de la función de valor*. Estos algoritmos buscan la función de valor óptima,

que consiste en los retornos mínimos de cada estado o estado-acción. La función de valor óptima es usada para calcular la póliza óptima.

- *Iteración de la póliza.* Estos algoritmos evalúan sus pólizas mediante la construcción de sus propias funciones de valor (en lugar de la función de valor óptima), y usa esas funciones de valor para encontrar nuevas y mejoradas pólizas.

- *Búsqueda de la póliza.* Estos algoritmos utilizan técnicas de optimización para buscar directamente una póliza óptima.

Dentro de estas tres subclases de algoritmos de DP y RL, dos categorías más pueden ser distinguidas, llamados algoritmos fuera de línea y en línea. Los algoritmos fuera de línea usan una recopilación de datos por adelantado, mientras que los algoritmos en línea aprenden una solución mediante su interacción con el sistema. Los algoritmos en línea generalmente no tienen una recopilación de datos por adelantado, en cambio dependen en los datos que van adquiriendo mientras aprenden, y es muy útil cuando es muy difícil o costoso conseguir dichos datos. La mayoría de los algoritmos de RL en línea son incrementales.

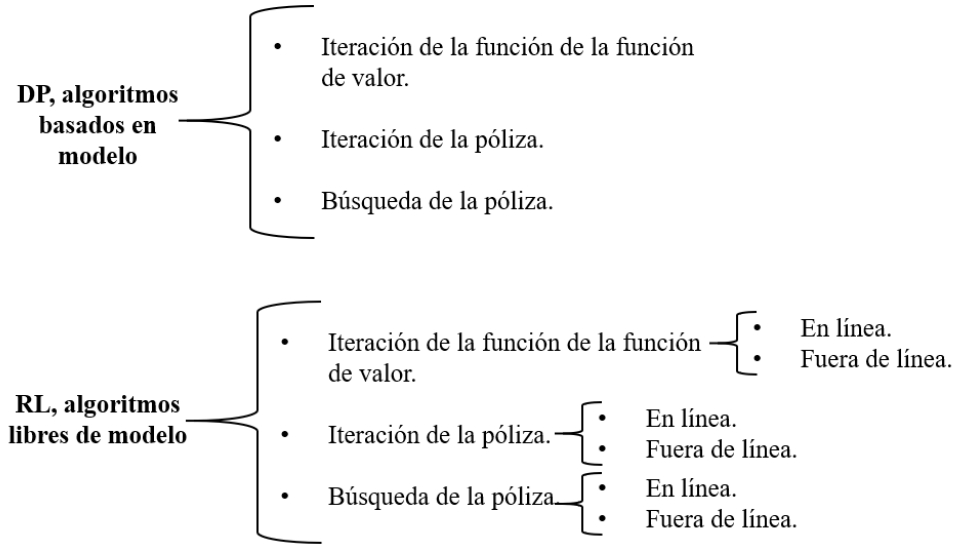


Figura 1.2: División de los algoritmos de DP y RL

Los algoritmos de RL en línea deben equilibrar su necesidad de conseguir información (mediante exploración de nuevas acciones o nuevos estados) con su necesidad de controlar el sistema de forma adecuada (mediante explotación de su conocimiento actual). Nótese que, a pesar que los algoritmos de RL en línea garantizan convergencia a la póliza óptima (bajo condiciones apropiadas) únicamente cuando el sistema no cambia a través del tiempo,

en la práctica, a veces también se aplican a sistemas que cambian lentamente, en cuyo caso se espera que la solución se adapte para tomar esos cambios en cuenta.

En la Figura 1.2 se muestra una de las posibles divisiones de los algoritmos de DP y RL propuesta por [2].

1.1.3. Iteración de la función de valor y la póliza

La técnica de iteración de la función de valor usa la ecuación de optimalidad de Bellman para calcular iterativamente la función de valor óptima, y derivar una póliza óptima.

Definición 3. *Sea $\mathcal{L}$ el conjunto de todas las funciones Q. Se define el operador de Bellman $\mathcal{H} : \mathcal{L} \to \mathcal{L}$ como*

$$[\mathcal{H}(Q)](x,u) = \rho(x,u) + \gamma \min_{u'} Q(f(x,u),u') \tag{1.21}$$

el cual calcula iterativamente el lado derecho de la ecuación de optimalidad de Bellman (1.15) para cualquier función Q.

El algoritmo de iteración de la función de valor Q empieza de una función arbitraria Q_0 y en cada iteración j actualiza la función Q usando:

$$Q^{j+1} = \mathcal{H}(Q^j) \tag{1.22}$$

Es posible demostrar que $\mathcal{H}$ es una contracción con factor $\gamma < 1$ en la norma infinito, esto es, para cualquier par de funciones Q_1 y Q_2, se satisface que:

$$\|\mathcal{H}(Q_1) - \mathcal{H}(Q_2)\| \leq \gamma \|Q_1 - Q_2\| \tag{1.23}$$

Debido a que $\mathcal{H}$ es una contracción, tiene un único punto de equilibrio. Además, la ecuación de optimalidad de Bellman (1.15) establece que Q^* es un punto de equilibrio de $\mathcal{H}$, esto es:

$$Q^* = \mathcal{H}(Q^*) \tag{1.24}$$

Por lo tanto el único punto de equilibrio de $\mathcal{H}$ es en realidad Q^*, y la iteración de Q converge a Q^* mientras $j \to \infty$. Además, la iteración de Q converge a una tasa de γ, en el sentido que $\|Q^{j+1} - Q^*\| \leq \gamma \|Q^j - Q^*\|$. Una póliza óptima puede ser calculada a partir de Q^* con (1.12). En el Algoritmo 1.1 se muestra el procedimiento para la iteración de la función Q.

Algoritmo 1.1 Iteración de la función Q

Entrada: dinámica f, función de recompensa ρ, factor de descuento γ
Inicializar la función Q, por ejemplo, $Q^0 \leftarrow 0$
repetir {en cada iteración $j = 0, 1, 2, \ldots$}
  para cada (x, u) **hacer**
    $Q^{j+1}(x, u) \leftarrow \rho(x, u) + \gamma \min_{u'} Q^j(f(x, u), u')$
  fin
hasta $[Q^{j+1} = Q^j]$
Salida: $Q^* = Q^j$

Los algoritmos de iteración de la póliza evalúan sus pólizas mediante la construcción de sus funciones de valor, y utilizan esas funciones de valor para encontrar nuevas y mejoradas pólizas. Como ejemplo representativo de iteración de la póliza considere un algoritmo fuera de línea que evalúa sus pólizas utilizando sus funciones Q. Este algoritmo inicia con una póliza arbitraria h_0. En cada iteración j, la función de valor Q^{h^j} de la póliza actual h^j es determinada; a este paso se le conoce como evaluación de la póliza. La evaluación de la póliza es realizada mediante la resolución de la ecuación de Bellman (1.14). Cuando la evaluación de la póliza es completada, se encuentra una nueva póliza h^{j+1} que es codiciosa en Q^h:

$$h^{j+1}(x) \in \operatorname*{argmin}_u Q^{h^j}(x, u) \tag{1.25}$$

Este paso es llamado mejora de la póliza. El algoritmo completo es mostrado en el Algoritmo 1.2. La secuencia de funciones Q producidas por la iteración de la póliza convergen asintóticamente a Q^* mientras $j \to \infty$. Simultáneamente, la póliza óptima h^* es obtenida.

Algoritmo 1.2 Iteración de la póliza a partir de funciones Q

Inicializar la póliza h^0
repetir {en cada iteración $j = 0, 1, 2, \ldots$}
  encontrar Q^{h^j}, la función Q de h^j {Evaluación de la póliza}
  $h^{j+1}(x) \in \operatorname{argmin}_u Q^{h^j}(x, u)$ {Mejora de la póliza}
hasta $h^{j+1} = h^j$
Salida: $h^* = h^j, Q^* = Q^{h^j}$

Antes de iniciar con los algoritmos de RL es necesario entender el caso idóneo, es decir, los algoritmos de DP. Como se mencionó anteriormente, los algoritmos de DP están diseñados con base al conocimiento de la dinámica del sistema o proceso. El algoritmo

más representativo es el regulador cuadrático lineal (LQR por sus siglas en inglés), el cual obtiene el control óptimo que minimiza una cierta función de costo. En la Tabla 1.2 se proporciona la teoría para el diseño del control LQR de sistemas discretos. Además en la Tabla 1.3 se muestra la forma de implementar el control LQR en línea de una forma iterativa.

1.2. Aprendizaje por Diferencia Temporal

El método de diferencia temporal (TD por sus siglas en inglés) resuelven las ecuaciones de Bellman en línea y resulta en una familia de controladores óptimos adaptables que aprenden la solución del problema de control óptimo sin conocer la dinámica completa del sistema [12]. El aprendizaje TD es un algoritmo de RL puramente en línea, cuyas acciones son mejoradas en tiempo real basándose en estimar sus funciones de valor mediante la observación de las mediciones de los datos a lo largo de las trayectorias del sistema.

La iteración de la póliza requiere la solución en cada paso de N ecuaciones lineales. La iteración de la función de valor requiere realizar la recursión en cada paso. Los métodos de aprendizaje por refuerzo TD están basados en las ecuaciones de Bellman y las resuelven sin utilizar conocimiento de la dinámica del sistema utilizando únicamente los datos observados a lo largo de una trayectoria del sistema. Por lo tanto, el aprendizaje TD es aplicable en aplicaciones de control por realimentación. TD actualiza los valores en cada paso de tiempo a medida que las observaciones de los datos se realizan a lo largo de una trayectoria. Periódicamente, el nuevo valor es usado para actualizar la póliza. Los métodos TD son relacionados con control adaptable en el sentido que ajustan sus valores y acciones en línea en tiempo real a lo largo de las trayectorias del sistema.

1.2.1. Q-Learning

Q-learning es un algoritmo de control adaptable que converge en línea a la solución óptima para sistemas completamente desconocidos. Esto es, resuelve las ecuaciones de Bellman (1.15) y (1.14) en línea mediante el uso de mediciones del estado y control a lo largo de sus trayectorias, sin conocer la función de transición de estados ó dinámica del sistema.

Q-learning, desarrollado por Watkins y Werbos, es un método simple para RL que trabaja para MDPs desconocidos, es decir, para sistemas cuya dinámica es completamente desconocida. Q-learning es llamado por Werbos como programación dinámica heurística

Tabla 1.2: Control Óptimo de sistemas en Tiempo discreto

Consideré el problema del regulador cuadrático lineal (LQR) en tiempo discreto, donde el MDP es determinista y satisface la ecuación de transición de estados:

$$x_{t+1} = Ax_t + Bu_t \tag{M1}$$

donde $x \in \mathbb{R}^n$ y $u \in \mathbb{R}^m$. La función de costo descontada de horizonte infinito asociada ó función de valor es:

$$\begin{aligned} J_t &= V(x_t) = \sum_{i=t}^{\infty} \gamma^{i-t} \rho(x_i, u_i) \\ &= \sum_{i=t}^{\infty} \gamma^{i-t} \left(x_i^\top S x_i + u_i^\top R u_i \right) \end{aligned} \tag{M2}$$

donde $S \in \mathbb{R}^{n \times n}$ y $R \in \mathbb{R}^{m \times m}$, que satisfacen $S = S^\top \geq 0$ y $R = R^\top > 0$; y $\gamma \in [0, 1)$ es un factor de descuento. Una ecuación en diferencias equivalente es:

$$\begin{aligned} V(x_t) = \; & x_t^\top S x_t + u_t^\top R u_t \\ & + \sum_{i=t+1}^{\infty} \gamma^{i-t} \left(x_i^\top S x_i + u_i^\top R u_i \right) \\ = \; & \rho(x_t, u_t) + \gamma V(x_{t+1}) \end{aligned} \tag{M3}$$

Asumiendo que la función de valor es cuadrática en el estado, se tiene:

$$V(x_t) = x_t^\top P x_t \tag{M4}$$

para alguna matriz P, se obtiene la ecuación de Bellman de la forma:

$$V(x_t) = x_t^\top P x_t = \rho(x_t, u_t) + x_{t+1}^\top \gamma P x_{t+1} \tag{M5}$$

si se utiliza la ecuación de transición de estado (M1), se obtiene:

$$\begin{aligned} V(x_t) = \; & x_t^\top P x_t = \rho(x_t, u_t) \\ & + (Ax_t + Bu_t)^\top \gamma P (Ax_t + Bu_t) \end{aligned} \tag{M6}$$

El problema de control óptimo puede ser formulado mediante el diseño de un control de la forma:

$$u_t^* = L x_t \tag{M7}$$

para alguna ganancia estabilizante $L \in \mathbb{R}^{m \times n}$, entonces se tiene:

$$\begin{aligned} V(x_t) = \; & x_t^\top P x_t = x_t^\top S x_t + x_t^\top L^\top R L x_t \\ & + x_t^\top (A + BL)^\top \gamma P (A + BL) x_t \end{aligned} \tag{M8}$$

Debido a que esta ecuación se válida para todas las trayectorias del estado, se obtiene que:

$$(A + BL)^\top \gamma P (A + BL) - P + S + L^\top R L = 0 \tag{M9}$$

la cual es una ecuación de Lyapunov. La ecuación de Bellman para el control LQR discreto es equivalente a una ecuación de Lyapunov. De (M6) se define el Hamiltoniano del control LQR discreto como:

$$\begin{aligned} H(x_t, u_t) = \; & \rho(x_t, u_t) - x_t^\top P x_t \\ & + (Ax_t + Bu_t)^\top \gamma P (Ax_t + Bu_t) \end{aligned} \tag{M10}$$

Una condición necesaria para optimalidad es la condición estacionaria $\partial H(x_t, u_t)/\partial u_t = 0$. La solución de la condición es el control óptimo dado por:

$$u_t^* = L x_t = -\left(B^\top P B + \frac{1}{\gamma} R \right)^{-1} B^\top P A x_t \tag{M11}$$

Sustituyendo la ecuación anterior en (M9) se obtiene la ecuación algebraica de Riccati en tiempo discreto (DARE por sus siglas en inglés)

$$A^\top P A - P + S - A^\top P B \left(B^\top P B + \frac{1}{\gamma} R \right)^{-1} B^\top P A = 0 \tag{M12}$$

La función de valor puede ser definida como:

$$\begin{aligned} V(x_t) &= \rho(x_t, u_t) + \gamma V(x_{t+1}) \\ &= x_t^\top S x_t + u_t^\top R u_t + \gamma x_{t+1}^\top P x_{t+1} \\ &= \begin{bmatrix} x_t \\ u_t \end{bmatrix}^\top \begin{bmatrix} \gamma A^\top P A + S & \gamma A^\top P B \\ \gamma B^\top P A & \gamma B^\top P B + R \end{bmatrix} \begin{bmatrix} x_t \\ u_t \end{bmatrix} \\ &= \begin{bmatrix} x_t \\ u_t \end{bmatrix}^\top H \begin{bmatrix} x_t \\ u_t \end{bmatrix}, \quad H = \begin{bmatrix} H_{xx} & H_{xu} \\ H_{ux} & H_{uu} \end{bmatrix} \end{aligned} \tag{M13}$$

El control óptimo puede ser obtenido por:

$$\begin{aligned} u_t^* &= -\frac{\partial V(x_t)}{\partial u_t} = -H_{uu}^{-1} H_{ux} x_t \\ &= -\left(B^\top P B + \tfrac{1}{\gamma} R \right)^{-1} B^\top P A x_t \end{aligned} \tag{M14}$$

Cuando el controlador es óptimo (M13) es igual a (M4). Por lo tanto, la relación entre P y H puede ser obtenida mediante:

$$\begin{bmatrix} x_t \\ u_t \end{bmatrix}^\top H \begin{bmatrix} x_t \\ u_t \end{bmatrix} = x_t^\top P x_t \tag{M15}$$

Nótese que $u_t = Lx_t$, la relación ente H y P es $P = \begin{bmatrix} I & L^\top \end{bmatrix} H \begin{bmatrix} I & L^\top \end{bmatrix}^\top$ Se define la siguiente función de valor de estado-acción Q:

$$Q(x_t, u_t) = V(x_t) = \begin{bmatrix} x_t \\ u_t \end{bmatrix}^\top H \begin{bmatrix} x_t \\ u_t \end{bmatrix} \tag{M16}$$

que también puede ser escrita como:

$$Q(x_t, u_t) = \rho(x_t, u_t) + \gamma Q(x_{t+1}, u_{t+1}) \tag{M17}$$

Tabla 1.3: Iteración de valor y póliza para el control LQR discreto

La ecuación de Bellman para el control LQR discreto puede ser usada para implementar iteración de la póliza (1.25) e iteración de la función de valor (1.22).

Algoritmo de Hewer

La iteración de la ecuación de Bellman para el LQR discreto o ecuación de Lyapunov es

$$V^{j+1}(x_t) = \rho(x_t, u_t) + \gamma V^{j+1}(x_{t+1}) \tag{M18}$$

La iteración de la póliza aplicada en (M5) resulta en

$$x_t^\top P^{j+1} x_t = \rho(x_t, u_t) + \gamma x_{t+1}^\top P^{j+1} x_{t+1} \tag{M19}$$

y la iteración de la póliza en (M9) resulta en la ecuación de Lyapunov

$$P^{j+1} = (A + BL^j)^\top \gamma P^{j+1}(A + BL^j) + S + (L^j)^\top R L^j \tag{M20}$$

En todos los casos la mejora de la póliza es:

$$\begin{aligned} h^{j+1}(x_t) &= L^{j+1} x_t \\ &= \text{argmin}_{u_t} \left(\rho(x_t, u_t) + \gamma x_{t+1}^\top P^{j+1} x_{t+1}\right) \end{aligned} \tag{M21}$$

que puede ser escrita explícitamente como:

$$L^{j+1} = -\left(B^\top P^{j+1} B + \frac{1}{\gamma} R\right)^{-1} B^\top P^{j+1} A \tag{M22}$$

El formato del algoritmo de iteración de la póliza (M20),(M22) se basa en soluciones repetitivas de las ecuaciones de Lyapunov en cada paso y es llamado *algoritmo de Hewer*. El algoritmo requiere que la ganancia inicial L^0 sea estabilizante.

Recursiones de Lyapunov

La ecuación de Bellman (M5) para la iteración de la función de valor esta dada por:

$$x_t^\top P^{j+1} x_t = x_t^\top S x_t + u_t^\top R u_t + x_{t+1}^\top P^j x_{t+1} \tag{M23}$$

y la iteración del valor dada la forma (M9) esta dada por la siguiente ecuación de Lyapunov recursiva

$$P^{j+1} = (A + BL^j)^\top P^j (A + BL^j) + S + (L^j)^\top R L^j \tag{M24}$$

En ambos casos, la mejora de la póliza esta dada por (M21) y (M22). El formato del algoritmo de iteración de la función de valor (M22) y (M24) es una *recursión de Lyapunov*, la cual es fácil de implementar y no requiere soluciones de la ecuación de Lyapunov en comparación de la iteración de la póliza.

El algoritmo de Hewer y algoritmo recursivo de Lyapunov son fuera de línea y requiere completo conocimiento de la dinámica del sistema (A, B) para encontrar el valor óptimo y control. Este algoritmo no requiere que la ganancia inicial sea estabilizante y puede ser inicializado con cualquier ganancia de realimentación.

dependiente de la acción (ADHDP por sus siglas en inglés), debido que la función Q depende de la entrada de control. Q-learning aprende la función Q (M17) usando los métodos TD al desempeñar una cierta acción u_t y midiendo los datos resultantes $x_t, x_{t+1}, \rho(x_t, u_t)$ (estado actual, estado siguiente y la recompensa, respectivamente) en cada instante de tiempo.

El algoritmo de iteración de valor para la función Q esta dada por (1.21). Basado en esto y el método de media muestral (A.3) presentado en el Apéndice A.1, la función Q es actualizada usando el siguiente algoritmo:

$$Q_{t+1}(x_t, u_t) = Q_t(x_t, u_t) + \alpha_t \left[\rho(x_t, u_t) + \gamma \min_{u'} Q_t(x_{t+1}, u') - Q_t(x_t, u_t)\right] \tag{1.26}$$

donde $\alpha_t \in (0, 1]$ es el parámetro de aprendizaje. El término en corchetes es el error de diferencia temporal que utiliza el algoritmo de iteración de valor para la búsqueda de la función Q óptima. Q-learning es un algoritmo que no se basa en la póliza debido que el aprendizaje de la función Q es independiente de la póliza.

Para establecer la convergencia del algoritmo Q-learning se utiliza el siguiente resultado

de aproximación estocástica.

Lema 1. *Considere el proceso estocástico* $(\zeta_t, \Delta_t, F_t), t \geq 0$, *donde* $\zeta_t, \Delta_t, F_t : X \to \mathbb{R}$ *satisface la ecuación:*

$$\Delta_{t+1}(z_t) = (1 - \zeta_t(z_t))\Delta_t(z_t) + \zeta_t(z_t)F_t(z_t) \tag{1.27}$$

donde $z_t \in X$. *Sea* P_t *una secuencia de campos* σ *crecientes, tal que* ζ_0 *y* Δ_0 *son* P_0*-medibles y* ζ_t, Δ_t *y* F_{t-1} *son* P_t*-medibles. Se asume que las siguientes condiciones son satisfechas:*

- *El conjunto* X *es finito.*
- $0 < \zeta_t(z_t) \leq 1$, $\sum_t \zeta_t(z_t) = \infty$, $\sum_t \zeta_t^2(z_t) < \infty$ *w.p.1 (con probabilidad 1) y* $\forall z \neq z_t$: $\zeta_t(z_t) = 0$.
- $\|E\{F_t(x)|P_t\}\| \leq \kappa\|\Delta_t\| + c_t$, *donde* $\kappa \in [0, 1)$ *y* c_t *converge a cero w.p.1.*
- $\text{VAR}\{F_t(z_t)|P_t\} \leq K(1+\kappa\|\Delta_t\|)^2$, *donde* K *es una constante positiva. La norma* $\|\cdot\|$ *denota la norma infinito.*

Entonces Δ_t *converge a cero con probabilidad uno (w.p.1. por sus siglas en inglés) [58].*

En el siguiente Teorema se expone la convergencia del algoritmo Q-learning:

Teorema 1. *Dado un MDP finito* (X, U, f, ρ), *el algoritmo Q-learning dado por (1.26) converge w.p.1 a la función óptima* Q^* *siempre y cuando*

$$\sum_t \alpha_t = \infty \quad \sum_t \alpha_t^2 < \infty \tag{1.28}$$

Debido a que $0 \leq \alpha_t \leq 1$, (1.28) requiere que todos los pares estado-acción sean visitados infinitamente a menudo. La prueba del Teorema 1 se presenta a continuación:

Demostración. Si se reescribe (1.26) como:

$$Q_{t+1}(x_t, u_t) = (1 - \alpha_t)Q_t(x_t, u_t) + \alpha_t \left(r_{t+1} + \gamma \min_{u'} Q_t(x_{t+1}, u')\right)$$

Se si resta por ambos lados de la ecuación anterior el término $Q^*(x_t, u_t)$ y sea

$$\Delta_t(x_t, u_t) = Q_t(x_t, u_t) - Q^*(x_t, u_t)$$

se obtiene:

$$\Delta_{t+1}(x_t, u_t) = (1 - \alpha_t)\Delta_t(x_t, u_t) + \alpha_t \left(r_{t+1} + \gamma \min_{u'} Q_t(x_{t+1}, u') - Q^*(x_t, u_t) \right)$$

Si se escribe:

$$F_t(x_t, u_t) = r_{t+1} + \gamma \min_{u'} Q_t(x_{t+1}, u') - Q^*(x_t, u_t)$$

De la anterior expresión y mediante el uso del operador $\mathcal{H}$ se tiene:

$$\begin{aligned} E\{F_t(x_t, u_t)|P_t\} &= [\mathcal{H}(Q_t)](x_t, u_t) - Q^*(x_t, u_t) \\ &= [\mathcal{H}(Q_t)](x_t, u_t) - [\mathcal{H}(Q^*)](x_t, u_t) \end{aligned}$$

Debido a que el operador $\mathcal{H}$ es una contracción se obtiene:

$$\|E\{F_t(x_t, u_t)|P_t\}\| \leq \gamma \|Q_t(x_t, u_t) - Q^*(x_t, u_t)\| \leq \gamma \|\Delta_t(x_t, u_t)\|$$

De aquí se observa que $c_t = 0$. Finalmente,

$$\begin{aligned} \text{VAR}\{F_t(x_t, u_t)|P_t\} &= E\left[\left(r_{t+1} + \gamma \min_{u'} Q_t(x_{t+1}, u') - [\mathcal{H}(Q_t)](x_t, u_t) \pm Q^*(x_t, u_t) \right)^2 \right] \\ &= E\left[\left(r_{t+1} + \gamma \min_{u'} Q_t(x_{t+1}, u') - [\mathcal{H}(Q_t)](x_t, u_t) \right)^2 \right] \\ &= \text{VAR}\left\{ r_{t+1} + \gamma \min_{u'} Q_t(x_{t+1}, u')|P_t \right\} \end{aligned}$$

y debido a que r_{t+1} es acotado, entonces claramente se verifica:

$$\text{VAR}\{F_t(x_t, u_t)|P_t\} \leq K \left(1 + \gamma \|\Delta_t(x_t, u_t)\|\right)^2$$

para alguna constante K. Entonces, por el Lema 1, Δ_t converge a cero w.p.1, y por lo tanto, Q_t converge a Q^* w.p.1. □

La primer condición de (1.28) no es difícil de satisfacer. Por ejemplo, una elección estándar es:

$$\alpha_t = \frac{1}{t+1} \tag{1.29}$$

En la práctica, el parámetro de aprendizaje requiere sintonización porque este influye el número de transiciones que se requiere para obtener una solución correcta. La elección correcta del parámetro de aprendizaje depende del problema a resolver.

La segunda condición de (1.28) se satisface si el controlador tiene una probabilidad distinta de cero de elegir cualquier acción en cada estado que se encuentre; esto se le conoce como exploración. El controlador sólo tiene que explotar su conocimiento actual para obtener un desempeño adecuado, por ejemplo, al elegir acciones codiciosas en la actual función Q. Esto se le conoce como el balance entre exploración-explotación de los algoritmos en línea de RL.

Dos métodos ampliamente utilizados para el balance entre exploración/explotación son ε-codicioso y softmax [1, 59]. Con ε-codicioso, el controlador elige en cada paso de tiempo una acción aleatoria con una probabilidad fija, $\varepsilon \in (0, 1)$, en lugar de elegir de forma codiciosa una acción de las acciones óptimas aprendidas con respecto a la función Q:

$$u_t = \begin{cases} \text{una acción aleatoria en } U & \text{con probabilidad } \varepsilon_t \\ u \in \operatorname{argmin}_{u'} Q_t(x_t, u') & \text{con probabilidad } 1 - \varepsilon_t \end{cases} \tag{1.30}$$

Algoritmo 1.3 Q-learning

1: **Entrada:** factor de descuento γ, término de exploración ε_t, parámetro de aprendizaje α_t.
2: Inicializar la función Q, por ejemplo, $Q_0 \leftarrow 0$
3: Medir estado inicial x_0
4: **para** cada paso de tiempo $t = 0, 1, \ldots$ **hacer**
5: $\quad u_t \leftarrow \begin{cases} u \in \operatorname{argmin}_{u'} Q_t(x_t, u') & \text{con probabilidad } 1 - \varepsilon_t \text{ (explotar)} \\ \text{una acción aleatoria en } U & \text{con probabilidad } \varepsilon_t \text{ (explorar)} \end{cases}$
6: $\quad$ Aplicar u_t, medir el siguiente estado x_{t+1} y recompensa $r_{t+1} = \rho(x_t, u_t)$
7: $\quad Q_{t+1}(x_t, u_t) \leftarrow Q_t(x_t, u_t) + \alpha_t \left[r_{t+1} + \gamma \operatorname{mín}_{u'} Q_t(x_{t+1}, u') - Q_t(x_t, u_t) \right]$
8: $\quad x_t \leftarrow x_{t+1}$.
9: **fin**

Por el contrario, softmax utiliza una selección de acciones probabilística que son determinadas por clasificar los estimados de la función de valor usando una distribución de Boltzmann:

$$Pr\{u|x_t\} = \frac{e^{\frac{Q_t(x_t,u)}{\tau_t}}}{\sum_b e^{\frac{Q_t(x,b)}{\tau_t}}} \tag{1.31}$$

donde τ_t es un parámetro positivo llamado temperatura. Altas temperaturas provocan que todas las acciones sean equiprobables, por el otro lado pequeñas temperaturas provoca la selección de acciones codiciosas [59].

El algoritmo de Q-learning con exploración ε-codicioso se muestra en el Algoritmo 1.3. Nótese que para este algoritmo se considera un entorno idealizado, en donde no se especifica una condición de finalización y no se produce una salida explícita. En cambio, el resultado del algoritmo es el mejoramiento del desempeño del controlador mientras interacciona con el sistema-proceso. Cuando Q-learning se detiene, la función Q resultante y la póliza codiciosa correspondiente pueden ser interpretadas como salidas y reutilizadas.

Sarsa es un algoritmo basado en la póliza propuesto por Rummery y Niranjan como una alternativa de Q-learning. El nombre de Sarsa se obtiene al juntar las iniciales (en inglés) de cada elemento de la tupla empleada por el algoritmo, las cuales son: estado, acción, recompensa, estado (siguiente), acción (siguiente). Formalmente, la tupla es escrita como $(x_t, u_t, r_{t+1}, x_{t+1}, u_{t+1})$. Sarsa comienza con una función Q inicial arbitraria Q_0 y la actualiza en cada paso usando tuplas de la forma anterior, como se muestra a continuación:

$$Q_{t+1}(x_t, u_t) = Q_t(x_t, u_\rangle + \alpha_t \left[r_{t+1} + \gamma Q_t(x_{t+1}, u_{t+1}) - Q_t(x_t, u_t)\right] \tag{1.32}$$

Sarsa requiere condiciones similares a las de Q-learning para converger a la función Q óptima, la cual demanda exploración y adicionalmente que la póliza exploratoria que se sigue asintóticamente sea codiciosa. En el siguiente teorema se establece la convergencia de Sarsa.

Teorema 2. *Dado un MDP finito (X, U, f, ρ), los valores Q_t calculados por el algoritmo de Sarsa (1.4) convergen a Q^* y la póliza aprendida $h(x_t)$ converge a una óptima $h^*(x_t)$ si la póliza aprendida es codiciosa en el límite con exploración infinita (GLIE por sus siglas en inglés) y además se satisface la condición (1.28).*

Demostración. De forma similar al algoritmo de Q-learning, la ley de actualización de Sarsa (1.4) es reescrita como:

$$\Delta_{t+1}(x_t, u_t) = (1 - \alpha_t)\Delta_t(x_t, u_t) + \alpha_t F_t(x_t, u_t)$$

donde $\Delta_t(x_t, u_t) = Q_t(x_t, u_t) - Q^*(x_t, u_t)$. $F(t)$ es escrita como:

$$\begin{aligned} F_t(x_t, u_t) =& r_{t+1} + \gamma \min_{u'} Q_t(x_{t+1}, u') - Q^*(x_t, u_t) + \gamma \left(Q_t(x_{t+1}, u_{t+1}) - \min_{u'} Q_t(x_{t+1}, u')\right) \\ \triangleq& r_{t+1} + \gamma \min_{u'} Q_t(x_{t+1}, u') - Q^*(x_t, u_t) + C_t(x_t, u_t) \\ \triangleq& F_t^Q(x_t, u_t) + C_t(x_t, u_t) \end{aligned}$$

donde $C_t(c_t, u_t) = \gamma\left(Q_t(x_{t+1}, u_{t+1}) - \text{mín}_{u'} Q_t(x_{t+1}, u')\right)$. $F_t^Q(x_t, u_t)$ es equivalente a la F_t de Q-learning, por lo tanto se sabe que $E\{F_t^Q(x_t, u_t)|P_t\} \leq \gamma\|\Delta_t(x_t, u_t)\|$ y sólo se tiene que demostrar que $C_t(x_t, u_t)$ converge a cero. Como se asume que la póliza es GLIE, es decir, las acciones no codiciosas son elegidas con probabilidades que van desapareciendo en cada iteración, entonces se garantiza la condición (1.28). Se define $Q_2(x_{t+1}, a) = \text{mín}_{u'} Q_t(x_{t+1}, u')$ donde a es la acción que minimiza la función Q en el estado siguiente con $\Delta_t^a = Q_t(x, u) - Q_2(x, u)$. Se suponen los siguientes casos:

- Se asume que $E\{Q_t(x_{t+1}, u_{t+1})|P_t\} \leq E\{Q_2(x_{t+1}, a)|P_t\}$. Entonces se satisface que $Q_t(x_{t+1}, u_{t+1}) \leq Q_t(x_{t+1}, a)$, por lo tanto:

$$\begin{aligned} E\{C_t(x_t, u_t)|P_t\} &= \gamma E\{Q_t(x_{t+1}, u_{t+1}) - Q_2(x_{t+1}, a)\} \\ &\leq \gamma E\{Q_t(x_{t+1}, a) - Q_2(x_{t+1}, a)\} \\ &\leq \gamma\|\Delta_t^a\| \end{aligned}$$

- Se asume que $E\{Q_2(x_{t+1}, a)|P_t\} \leq E\{Q_t(x_{t+1}, u_{t+1})|P_t\}$. Entonces se satisface que $Q_2(x_{t+1}, a) \leq Q_2(x_{t+1}, u_{t+1})$, por lo tanto:

$$\begin{aligned} E\{C_t(x_t, u_t)|P_t\} &= \gamma E\{Q_t(x_{t+1}, u_{t+1}) - Q_2(x_{t+1}, a)\} \\ &\leq \gamma E\{Q_t(x_{t+1}, u_{t+1}) - Q_2(x_{t+1}, u_{t+1})\} \\ &\leq \gamma\|\Delta_t^a\|. \end{aligned}$$

En ambos casos se obtiene la misma expresión y además la varianza está acotada, entonces por el Lema 1, Δ_t^a converge a cero y por lo tanto Δ_t también lo hace y Q_t converge a Q^*. □

En el Algoritmo 1.4 se muestra el algoritmo completo de Sarsa usando el método de exploración ε-codicioso.

1.3. Conclusión

En este capítulo se presentó la teoría principal de la programación dinámica y el aprendizaje por refuerzo. Se proporcionaron los elementos que componen un método de aprendizaje por refuerzo y algunos de sus algoritmos más utilizados como son Q-learning y Sarsa. La convergencia de los algoritmos es presentada mediante el uso de la propiedad

Algoritmo 1.4 Sarsa

1: **Entrada:** factor de descuento γ, término de exploración ε_t, parámetro de aprendizaje α_t.
2: Inicializar la función Q, por ejemplo, $Q_0 \leftarrow 0$
3: Medir el estado inicial x_0
4: $u_0 \leftarrow \begin{cases} u \in \operatorname{argmin}_{u'} Q_0(x_0, u') & \text{con probabilidad } 1-\varepsilon_0 \text{ (explotar)} \\ \text{una acción aleatoria en } U & \text{con probabilidad } \varepsilon_0 \text{ (explorar)} \end{cases}$
5: **para** cada paso de tiempo $t = 0, 1, \ldots$ **hacer**
6: Aplicar u_t, medir el siguiente estado x_{t+1} y recompensa $r_{t+1} = \rho(x_t, u_t)$
7: $u_{t+1} \leftarrow \begin{cases} u \in \operatorname{argmin}_{u'} Q_t(x_{t+1}, u') & \text{con probabilidad } 1-\varepsilon_{t+1} \\ \text{una acción aleatoria en } U & \text{con probabilidad } \varepsilon_{t+1} \end{cases}$
8: $Q_{t+1}(x_t, u_t) \leftarrow Q_t(x_t, u_t) + \alpha_t \left[r_{t+1} + \gamma Q_t(x_{t+1}, u_{t+1}) - Q_t(x_t, u_t) \right]$
9: $x_t \leftarrow x_{t+1}$, $u_t = u_{t+1}$.
10: **fin**

de contracción que posee la función de valor. Además se mostraron herramientas para la exploración/explotación del espacio de las acciones.

Capítulo 2

Aproximación del Aprendizaje por Refuerzo

En el capítulo pasado, se introdujo la teoría de DP (Programación Dinámica) y RL (Aprendizaje por Refuerzo) usando algunos de sus algoritmos más utilizados. Cuando el espacio de estado-acciones contiene un número infinito de elementos, es imposible recorrer todos lo pares de estado-acción en tiempo finito, por lo tanto los algoritmos del Capítulo anterior no presentarían buenos resultados. En su lugar, una aproximación de la ley de actualización debe ser utilizada, la cual considere únicamente un número finito de muestras de estado-acción. En este capítulo, se aborda el problema de espacios grandes y continuos de estado-acción utilizando métodos de aproximación del RL. Estos métodos de aproximación se basan en parametrizar las funciones de valor y/o póliza utilizando un vector de funciones propuestas por el usuario y un vector de parámetros que pondera a cada función propuesta. Los algoritmos de aproximación del RL buscan el vector de parámetros óptimo que aproxime la función de valor y/o póliza óptimas.

2.1. Aproximadores

En general, el problema de aproximación de funciones requiere que se seleccione una función entre una clase bien definida de funciones que coincida o "aproxime" una función objetivo. Esta clase de funciones son conocidas como aproximadores o aproximadores de funciones. Entre las principales clases de aproximadores se tienen: aproximadores paramétricos y no paramétricos, los cuales se explican en las siguientes secciones.

2.1.1. Aproximadores paramétricos

Los aproximadores paramétricos son un mapeo del espacio de parámetros al espacio de funciones que pretenden representar (en el ámbito de DP/RL, las funciones a representar son las funciones de valor y pólizas). La forma funcional del mapeo y el número de parámetros son establecidos por adelantado y no dependen de los datos. Los parámetros del aproximador son sintonizados usando los datos de la función objetivo, que en este caso serían los datos proporcionados por el par estado-acción.

Considérese un aproximador de la función Q parametrizado por un vector $\theta \in \mathbb{R}^p$. El aproximador es denotado por el mapeo $F : \mathbb{R}^p \to \mathcal{L}$, donde $\mathbb{R}^p$ es el espacio de los parámetros y $\mathcal{L}$ es el espacio de las funciones Q. Cada vector de parámetros θ proporciona una representación compacta de una función Q aproximada:

$$\widehat{Q}(x, u; \theta) = [F(\theta)](x, u) \tag{2.1}$$

donde $[F(\theta)](x, u)$ denota la función de valor $F(\theta)$ evaluada en el par estado-acción (x, u). Entonces, en lugar de guardar distintos valores de Q por cada par (x, u), solo es necesario guardar p parámetros. En general, el mapeo F puede ser no-lineal en los parámetros, sin embargo, los aproximadores linealmente parametrizables son preferidos en DP y RL, porque facilitan analizar las propiedades teóricas de los algoritmos de DP/RL.

Un aproximador de la función Q parametrizada linealmente emplea p funciones base (BFs por sus siglas en inglés) $\phi_1, \ldots, \phi_p : X \times U \to \mathbb{R}$ y un vector de parámetros θ de dimensión p. La aproximación de los valores de Q es calculada con:

$$[F(\theta)](x, u) = \sum_{l=1}^{p} \phi_l(x, u)\theta_l = \Phi^\top(x, u)\theta \tag{2.2}$$

donde $\Phi(x, u) = [\phi_1(x, u), \ldots, \phi_p(x, u)]^\top$ es el vector de BF.

En este trabajo se utilizan funciones radiales base Gaussianas normalizadas (NRBF por sus siglas en inglés) como aproximadores. Este tipo de RBF son definidas como:

$$\phi_i(x, u) = \frac{\exp\left(-\frac{1}{2}\begin{bmatrix} x - c_i \\ u - c_i \end{bmatrix}^\top \beta_i^{-1} \begin{bmatrix} x - c_i \\ u - c_i \end{bmatrix}\right)}{\sum_{i=1}^{D} \exp\left(-\frac{1}{2}\begin{bmatrix} x - c_i \\ u - c_i \end{bmatrix}^\top \beta_i^{-1} \begin{bmatrix} x - c_i \\ u - c_i \end{bmatrix}\right)} \tag{2.3}$$

donde D es el número de RBF por dimensión de la entrada, el vector $c_i \in \mathbb{R}^D$ denota el centro de la i-ésima RBF, y $\beta_i \in \mathbb{R}^{D\times D}$ es una matriz simétrica y positiva definida que representa el ancho de la RBF. Sin embargo, el problema principal de utilizar NRBFs es el número de RBFs a utilizar y la ubicación de sus centros. En la literatura [52, 60], los centros de las RBFs se fijan en un valor determinado o punto de equilibrio, de tal forma que cuando se encuentre cerca de este punto el valor de la RBF será mayor. Esto es útil para problemas de estabilización como los sistemas pendubot, acrobot y carro péndulo [61].

El número de RBFs depende de la dimensión de los datos de entrada, con pocas RBFs se obtiene una estimación incorrecta, por el otro lado, si se incrementar el número de RBFs se obtiene una buena estimación pero puede presentar el problema de sobreajuste. Una solución simple para obtener los centros de las RBFs es usando el algoritmo de clustering K-means [62–67] para particionar los datos de entrada en K clusters o agrupamientos, más aún, el número de clusters es igual al número de RBFs del aproximador.

2.1.2. K-means Clustering

Sea $\mathcal{Y} = \{y_i\}, i = 1, \dots n'$, el conjunto de puntos de dimensión n' a agrupar en un conjunto de K clusters, n' es el número de puntos de la entrada, $C = \{c_j, j = 1, \dots K\}$. Cada centroide de los cluster es una colección de valores característicos que definen los grupos resultantes. El algoritmo de clustering K-means encuentra una partición del conjunto tal que el error cuadrático entre la media del cluster y los puntos dentro del cluster sea minimizado. El error cuadrático entre la media, μ_j, del cluster c_j es definido como:

$$\mathcal{J}(c_j) = \sum_{y_i \in \mathbf{c}_j} \|y_i - \mu_j\|^2 \tag{2.4}$$

El objetivo principal de K-means es minimizar la suma del error cuadrático sobre todos los K clusters,

$$\mathcal{J}(C) = \min_{C} \sum_{j=1}^{K} \sum_{y_i \in c_j} \|y_i - \mu_j\|^2 \tag{2.5}$$

El algoritmo de K-means esta dado en el Algoritmo 2.1.

El algoritmo obtiene K centros aleatorios y permite obtener una familia de aproximadores usando NRBFs. El aproximador obtiene diferentes parámetros θ por tener centros aleatorios; para obtener los mismos parámetros se requiere fijar los valores de los centros.

Existen diversos tipos de aproximadores lineales como: regresión lineal local (LLR por

Algoritmo 2.1 Clustering K-means

1: **Entrada:** K,
2: Inicializar los centroides de los clusters $C = \{c_j, j = 1, \dots K\}$,
3: **repetir**
4: Encontrar el centroide más cercano a cada punto, $I_j \leftarrow \arg\min_{c_j \in C} \|y - c_j\|^2$,
5: Actualizar los centroides $c_j \leftarrow \frac{1}{|I_j|} \sum_{y_i \in I_j} y_i$
6: **hasta** las membresías de los clusters converjan

sus siglas en inglés) y *tile coding*, los cuales son abordados en [52, 60].

2.1.3. Aproximadores no-paramétricos

Los aproximadores no-paramétricos, a pesar de su nombre, tienen parámetros. Sin embargo, a diferencia de los aproximadores paramétricos, el número de parámetros y la forma del aproximador no-paramétrico son obtenidos mediante los datos disponibles.

Los aproximadores de tipo Kernel son los aproximadores no-paramétricos típicos. Considérese un aproximador de tipo de Kernel de la función Q. En este caso, la función kernel es una función definida sobre dos pares de estado-acción, $\kappa : X \times U \times X \times U \to \mathbb{R}$:

$$(x, u, x', u') \mapsto \kappa\left((x, u), (x', u')\right) \tag{2.6}$$

La función κ puede ser interpretada como el producto interno entre los vectores de características de sus dos argumentos en un espacio de características de alta dimensión. Entonces se obtiene un aproximador robusto con tan solo calcular kernels, sin trabajar explícitamente en el espacio de características.

Un kernel ampliamente utilizado es el kernel Gaussiano, el cual puede ser expresado (para la aproximación de la función Q) de la siguiente forma:

$$\kappa\left((x, u), (x', u')\right) = \exp\left(-\frac{1}{2}\begin{bmatrix} x - x' \\ u - u' \end{bmatrix}^\top \beta^{-1} \begin{bmatrix} x - x' \\ u - u' \end{bmatrix}\right) \tag{2.7}$$

Se observa que, cuando el par (x', u') es fijo, el kernel (2.7) tiene la misma forma que la RBF Gaussiana centrado en (x', u').

Si se asume que el conjunto de muestras estado-acción n_s es disponible. Luego para este conjunto de muestras, el aproximador de tipo kernel toma la forma:

$$\widehat{Q}(x,u;\theta) = \sum_{l=1}^{n_s} \kappa\left((x,u),(x',u')\right)\theta_l \tag{2.8}$$

Esta forma es similar al aproximador linealmente parametrizable (2.2). Sin embargo, existe una diferencial crucial entre los dos aproximadores. En el caso paramétrico, el número y forma de las BFs son definidas por adelantado, y por lo tanto produce una forma funcional fija del aproximador F. Por el contrario, en el caso no-paramétrico, el número de kernels y su forma y por lo tanto el número de parámetros y la forma funcional del aproximador, son determinadas por sus muestras. Se nota que los aproximadores no-paramétricos son conducidos por ciertos hiperparámetros, como el ancho β del kernel Gaussiano (2.7). Estos hiperparámetros influyen en la precisión del aproximador y requiere sintonización, el cual puede ser difícil realizar de forma manual.

2.2. Aproximación de la iteración

Se recuerda que la iteración de la función Q comienza con una función Q arbitraria Q_0 y en cada iteración j actualiza la función Q usando la regla (1.21). En la aproximación de la iteración de Q, la función Q^j no puede ser exactamente representada. En cambio, una versión aproximada es representada de forma compacta por un vector de parámetros $\theta^j \in \mathbb{R}^p$, usando un apropiado mapeo $F : \mathbb{R}^p \to \mathcal{L}$:

$$\widehat{Q}^j = F(\theta^j).$$

Esta aproximación de la función Q es proporcionada, en lugar de Q^j, como una entrada al mapeo de iteración $\mathcal{H}$. Entonces, la actualización de la iteración Q se convierte:

$$Q_a^{j+1} = (\mathcal{H} \circ F)(\theta^j). \tag{2.9}$$

Sin embargo, en general, la función Q recién encontrada Q_a tampoco puede ser guardada explicítamente. En cambio, debe ser también representada de forma aproximada, usando un nuevo vector de parámetros θ^{j+1}. Este vector de parámetros es obtenido usando un mapeo de proyección $\mathcal{P} : \mathcal{L} \to \mathbb{R}^p$:

$$\theta^{j+1} = \mathcal{P}(Q_a) \tag{2.10}$$

el cual asegura que $\widehat{Q}^{j+1} = F(\theta^{j+1})$ sea lo más cercano posible a Q_a. Para resumir, la aproximación de la iteración de Q empieza con un vector de parámetros arbitrario θ_0, y actualiza este vector cada iteración j usando la composición de mapeos $\mathcal{P}, \mathcal{H}$ y F:

$$\theta^{j+1} = (\mathcal{P} \circ \mathcal{H} \circ F)(\theta^j). \tag{2.11}$$

El algoritmo se detiene una vez que se encuentre un vector óptimo de parámetros $\widehat{\theta}^*$, es decir, $\widehat{\theta}^* = \mathcal{P} \circ \mathcal{H} \circ F(\widehat{\theta}^*)$. Idealmente, $\widehat{\theta}^*$ es cercano a un punto fijo θ^* de $\mathcal{P} \circ \mathcal{H} \circ F$. Dado $\widehat{\theta}^*$, una póliza codiciosa en $F(\widehat{\theta}^*)$ puede ser encontrada, es decir, una póliza h que satisface:

$$h(x) \in \operatorname{argmin}[F(\widehat{\theta}^*)](x, u) \tag{2.12}$$

La Figura 2.1 ilustra el funcionamiento de la aproximación de la iteración de Q y las relaciones entre los varios mapeos, vector de parámetros y las funciones Q consideradas por el algoritmo.

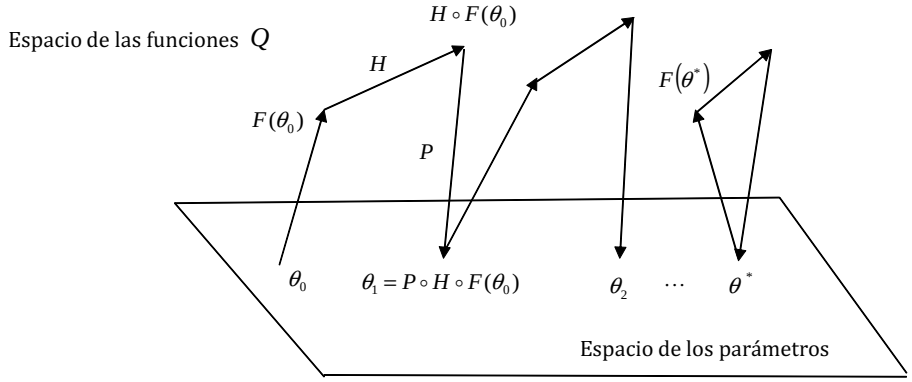


Figura 2.1: Aproximación de la Iteración de Q

La aproximación del algoritmo TD-learning se basa en el método de gradiente descendiente que minimiza el error cuadrático entre el valor óptimo (objetivo de aprendizaje) y el valor actual Q:

$$\begin{aligned}\theta_{t+1} &= \theta_t - \frac{1}{2}\alpha_t \frac{\partial}{\partial \theta_t}\left[Q^*(x_t, u_t) - \widehat{Q}_t(x_t, u_t; \theta_t)\right]^2 \\ &= \theta_t + \alpha_t \left[Q^*(x_t, u_t) - \widehat{Q}_t(x_t, u_t; \theta_t)\right] \frac{\partial \widehat{Q}_t(x_t, u_t; \theta_t)}{\partial \theta_t}.\end{aligned} \tag{2.13}$$

Al sustituir la parametrización (2.1) en (2.13) y modificar el objetivo de aprendizaje por

su aproximación se obtienen la aproximación de las reglas de actualización de Q-learning y Sarsa, respectivamente:

$$\theta_{t+1} = \theta_t + \alpha_t \left[r_{t+1} + \gamma \min_u \left(\Phi^\top(x_{t+1}, u)\theta_t \right) - \Phi^\top(x_t, u_t)\theta_t \right] \Phi(x_t, u_t) \tag{2.14}$$

$$\theta_{t+1} = \theta_t + \alpha_t \left[r_{t+1} + \gamma \Phi^\top(x_{t+1}, u_{t+1})\theta_t - \Phi^\top(x_t, u_t)\theta_t \right] \Phi(x_t, u_t) \tag{2.15}$$

Los algoritmos requieren suficiente exploración para una aproximación confiable. La aproximación de los algoritmos de Q-learning y Sarsa usando el algoritmo de clustering K-means se muestran en los Algoritmos 2.2 y 2.3. Ambos algoritmos usan la estrategia de exploración ε-codicioso.

Algoritmo 2.2 Aproximación de Q-learning

Entrada: γ, α_t, K,
Ejecutar el Algoritmo 2.1 de K-means,
Calcular las BFs $\phi_i(x_t, u_t)$ de acuerdo a los centroides obtenidos en el paso anterior,
Inicializar x_0, θ_0,
para cada paso de tiempo $t = 0, 1, 2, \ldots$ **hacer**
  $u_t \leftarrow \begin{cases} \operatorname{argmin}_u \left(\Phi^\top(x_t, u)\theta_t \right) & \text{con probabilidad } 1 - \varepsilon_t \text{ (explotar)} \\ \text{una acción aleatoria en } U & \text{con probabilidad } \varepsilon_t \text{ (explorar)} \end{cases}$
  Aplicar u_t, medir siguiente estado x_{t+1} y la recompensa r_{t+1},
  Actualizar θ_t usando (2.14)
fin

Algoritmo 2.3 Aproximación de Sarsa

Entrada: γ, α_t, K,
Ejecutar el Algoritmo 2.1 de K-means,
Calcular las BFs $\phi_i(x_t, u_t)$ de acuerdo a los centroides obtenidos en el paso anterior,
Inicializar x_0, θ_0,
$u_0 \leftarrow \begin{cases} \operatorname{argmin}_u \left(\Phi^\top(x_0, u)\theta_0 \right) & \text{con probabilidad } 1 - \varepsilon_t \text{(explotar)} \\ \text{una acción aleatoria en } U & \text{con probabilidad } \varepsilon_t \text{(explorar)} \end{cases}$
para cada paso de tiempo $t = 0, 1, 2, \ldots$ **hacer**
  Aplicar u_t, medir siguiente estado x_{t+1} y la recompensa r_{t+1},
  $u_{t+1} \leftarrow \begin{cases} \operatorname{argmin}_u \left(\Phi^\top(x_{t+1}, u)\theta_t \right) & \text{con probabilidad } 1 - \varepsilon_{t+1} \\ \text{una acción aleatoria en } U & \text{con probabilidad } \varepsilon_{t+1} \end{cases}$
  Actualizar θ_t usando (2.15)
fin

Los algoritmos expuestos requieren suficiente exploración para una aproximación con-

fiable. Para una iteración de la póliza, la póliza óptima esta dada por:

$$h^*(x) \in \underset{u}{\operatorname{argmin}} F(\theta^*) = \underset{u}{\operatorname{argmin}} \widehat{Q}(x, u; \theta)$$

La prueba de convergencia de los algoritmos de aproximación yacen en la propiedad de contracción:

$$\left\|(F^\dagger \circ \mathcal{H} \circ F)(\theta) - (F^\dagger \circ \mathcal{H} \circ F)(\theta')\right\| \leq \gamma' \left\|\theta - \theta'\right\|$$

donde $F^\dagger$ es una proyección de la pseudo-inversa de F y $\mathcal{H}$ es el operador de Bellman y γ' es una constante de contracción que es equivalente a un factor de descuento. Para la prueba de convergencia se considera que las siguientes suposiciones son satisfechas:

S1. El parámetro de aprendizaje satisface (1.28).

S2. Las funciones base satisfacen:

$$\sum_{i=1}^{K} |\phi_i(x, u)| \leq 1$$

S3. La contracción $\mathcal{H}$ satisface (1.23).

S4. Existe un mapeo $\mathcal{H}' = F^\dagger \circ \mathcal{H} \circ F$ con constante de contracción $\gamma' = \gamma$, tal que

$$\theta = F^\dagger(F(\theta))$$

y para cualquier θ, θ' y Q, Q' satisface:

$$\|F(\theta) - F(\theta')\| \leq \|\theta - \theta'\|$$
$$\left\|F^\dagger(Q) - F^\dagger(Q')\right\| \leq \|Q - Q'\|$$

S5. La varianza es acotada $\text{VAR} < \infty$.

La suposición **S1** requiere que todos los pares estado-acción sean visitados infinitamente, **S2-S4** son sencillas de probar por medio de la propiedad de contracción y que las funciones base son acotadas, **S5** es una consecuencia de las demás suposiciones.

Teorema 3. *Se supone que las suposiciones **S1-S5** se satisfacen y el mapeo $\mathcal{H}$ y $\mathcal{H}'$ son una contracción respecto a la norma infinito. Entonces la regla de actualización (2.13)*

converge al único punto de equilibrio θ^* con probabilidad 1 y lo siguiente se satisface:

$$\|Q^* - F(\theta^*)\| \leq \frac{2}{1-\gamma} e_\theta$$
$$\left\|Q^* - \widehat{Q}^{h^*}\right\| \leq \frac{4\gamma}{(1-\gamma)^2} e_\theta$$

donde $e_\theta = \min_\theta \|Q^* - F(\theta)\|$ es un error residual. Resultados similares se encuentran en [68].

Demostración. Se define el error entre la función de valor óptima Q^* y una aproximación $F(\theta)$ como:

$$e_\theta = \min_\theta \|Q^* - F(\theta)\|$$

Si se utiliza las suposiciones **S3** y **S4** se obtiene:

$$\begin{aligned}\|\theta - \mathcal{H}'(\theta)\| &\leq \left\|F^\dagger(F(\theta)) - F^\dagger(\mathcal{H}(F(\theta)))\right\| \\ &\leq \|F(\theta) - \mathcal{H}(F(\theta))\| \\ &\leq \|F(\theta) - Q^*\| + \|Q^* - \mathcal{H}(F(\theta))\| \leq e_\theta + \gamma e_\theta\end{aligned}$$

Si se utiliza el resultado anterior en el vector de parámetros óptimo θ^* se obtiene:

$$\begin{aligned}\|\theta^* - \theta\| &\leq \|\theta^* - \mathcal{H}'(\theta)\| + \|\mathcal{H}'(\theta) - \theta\| \\ &\leq \gamma \|\theta^* - \theta\| + (1+\gamma) e_\theta \leq \frac{1+\gamma}{1-\gamma} e_\theta\end{aligned}$$

Para la iteración de la función Q se tiene:

$$\begin{aligned}\|Q^* - F(\theta^*)\| &\leq \|Q^* - F(\theta)\| + \|F(\theta) - F(\theta^*)\| \\ &\leq e_\theta + \|\theta - \theta^*\| \leq e_\theta + \frac{1+\gamma}{1-\gamma} e_\theta \leq \frac{2}{1-\gamma} e_\theta\end{aligned}$$

Para un algoritmo de iteración de la póliza se tiene:

$$\begin{aligned}\left\|Q^* - \widehat{Q}^{h^*}\right\| &\leq \|Q^* - \mathcal{H}(F(\theta^*))\| + \left\|\mathcal{H}(F(\theta^*)) - \widehat{Q}^{h^*}\right\| \\ &\leq \gamma \|Q^* - F(\theta^*)\| + \gamma \left\|F(\theta^*) - \widehat{Q}^{h^*}\right\| \\ &\leq 2\gamma \|Q^* - F(\theta^*)\| + \gamma \left\|Q^* - \widehat{Q}^{h^*}\right\| \leq \frac{4\gamma}{(1-\gamma)^2} e_\theta\end{aligned}$$

Como se demostró que $\mathcal{H}'$ es una contracción, entonces converge a un único punto fijo θ^* con probabilidad 1. Además como las funciones base son acotadas, entonces la función de recompensa es acotada y por lo tanto la varianza VAR $< \infty$. □

Los algoritmos clásicos de RL son diseñados para sistemas en tiempo discreto ya sea para espacios discretos contables o espacios discretos grandes, sin embargo es posible extrapolar la idea en sistemas en tiempo continuo. Los sistemas en tiempo continuo lidian con espacios de estado-acción continuos por naturaleza, y también requieren aproximación o técnicas avanzadas de control. En las siguientes secciones se aborda el problema de RL en tiempo continuo el cual, es un problema de dimensionalidad con mayor dificultad que el presentado en el capítulo anterior. En la Tabla 2.1 se presenta el desarrollo del control óptimo para sistemas en tiempo continuo.

2.3. Programación Dinámica y Aprendizaje por Refuerzo Adaptable de Sistemas en Tiempo Continuo

Una póliza $h(x(t))$ se le llama admisible si es continua, estabiliza al sistema, y tiene una función de costo asociado. Se define la siguiente función de valor continuamente diferenciable para una póliza de control admisible h:

$$V^h(x(t)) = \int_t^\infty e^{-\gamma(\tau-t)} \left(x^\top Sx + h^\top Rh\right) d\tau. \tag{2.16}$$

La expresión anterior puede ser escrita de la forma del aprendizaje por refuerzo integral (IRL por sus siglas en inglés) [69, 70]:

$$V^h(x(t)) = \int_t^{t+T} e^{-\gamma(\tau-t)} \left(x^\top Sx + h^\top Rh\right) d\tau + e^{-\gamma T} V^h(x(t+T)) \tag{2.17}$$

$$= \int_t^{t+T} e^{-\gamma(\tau-t)} \rho(x(\tau), h(\tau)) d\tau + e^{-\gamma T} V^h(x(t+T)) \tag{2.18}$$

para cualquier $T > 0$. A esto se le conoce como ecuación de Bellman Integral. Se nota que la función de valor para una póliza admisible h es cuadrática también, es decir, $V^h(x(t)) = x^\top(t)Px(t)$.

Tabla 2.1: Control Óptimo de sistemas en Tiempo Continuo

Considere el sistema lineal e invariante en el tiempo (LTI por sus siglas en inglés)

$$\dot{x} = Ax + Bu, \quad x(t_0) = x_0, \ (t \geq t_0) \tag{N1}$$

donde $A \in \mathbb{R}^{n\times n}$ es la matriz del sistema, $B \in \mathbb{R}^{n\times m}$ es la matriz de acoplamiento a la entrada, $x(t) \in X \subseteq \mathbb{R}^n$ es el vector de variables de estado, $u(t) \in U \subseteq \mathbb{R}^m$ es la entrada de control, X es el espacio de funciones acotadas, y U es el conjunto compacto de acciones. Se asume que el par (A, B) es estabilizable.
El objetivo del control LQR es encontrar la póliza óptima $u^*(t)$ para el sistema (N1) que minimice el desempeño del índice cuadrático o función de costo

$$\begin{aligned} V(x(t)) &= J(x(t), \bar{u}[t:\infty)) \\ &= \int_t^\infty e^{-\gamma(\tau-t)} \left(x^\top Sx + u^\top Ru\right) d\tau \end{aligned} \tag{N2}$$

donde $\bar{u}[t:\infty) \triangleq \{u(\tau) : t \leq \tau < \infty\}$, $\gamma \geq 0$ es un factor de descuento, $S \in \mathbb{R}^{n\times n}$ y $R \in \mathbb{R}^{m\times m}$ son matrices simétricas y positivas definidas. El desempeño de la función de costo (N2) para pólizas fijas se convierten en la función de valor. Una póliza de control fija se define como $u(t) = -Kx(t)$, donde K es la ganancia del controlador.
Si se deriva a (N2) respecto al tiempo, se obtiene la siguiente ecuación de Bellman

$$\begin{aligned} &\dot{V}(x(t)) + x^\top(t)Sx(t) + u^\top(t)Ru(t) \\ &- \int_t^\infty \frac{\partial}{\partial t} e^{-\gamma(\tau-t)} \left(x^\top Sx + u^\top Ru\right) d\tau = 0 \end{aligned} \tag{N3}$$

La función de valor óptima $V^*(x(t))$ esta dada por:

$$V^*(x(t)) = \min_{\bar{u}[t:\infty)} \int_t^\infty e^{-\gamma(\tau-t)} \left(x^\top Sx + u^\top Ru\right) d\tau. \tag{N4}$$

Entonces, basados en (N3) y utilizando el principio de optimalidad de Bellman, se obtiene la siguiente ecuación de Hamilton-Jacobi Bellman (HJB)

$$\begin{aligned} &\min_{\bar{u}[t:\infty)} \Big\{ \dot{V}^*(x(t)) + x^\top(t)Sx(t) + u^\top(t)Ru(t) \\ &- \int_t^\infty \frac{\partial}{\partial t} e^{-\gamma(\tau-t)} \left(x^\top Sx + u^\top Ru\right) d\tau \Big\} = 0 \end{aligned} \tag{N5}$$

Para el problema de control óptimo, la función de valor es cuadrática en términos del vector de estado como en (M4)

$$V^*(x(t)) = x^\top(t)Px(t) \tag{N6}$$

para alguna matriz P la cual es positiva definida. Si se usa (N6) en (N5) y por la regla de derivación por el signo de la integral de Leibniz, se obtiene la ecuación diferencial descontada de Riccati:

$$\dot{x}^\top Px + x^\top P\dot{x} + x^\top Sx + u^\top Ru - \gamma x^\top Px = 0 \tag{N7}$$

Utilizando la dinámica del sistema (N1) en (N7) se obtiene

$$\begin{aligned} x^\top A^\top Px \quad &+u^\top B^\top Px + x^\top PAx + x^\top PBu \\ &+x^\top Sx + u^\top Ru - \gamma x^\top Px = 0 \end{aligned} \tag{N8}$$

La solución del problema de control óptimo se determina al derivar (N8) con respecto a u. El resultado es

$$u^*(t) = -Kx(t) = -R^{-1}B^\top Px(t) \tag{N9}$$

Substituyendo la ganancia de control K de (N9) en (N8) se obtiene la ecuación descontada algebraica de Riccati (ARE por sus siglas en inglés)

$$\left(A - \frac{\gamma}{2}I\right)^\top P + P\left(A - \frac{\gamma}{2}I\right) - PBR^{-1}B^\top P + S = 0 \tag{N10}$$

La solución del problema de control óptimo es definido por la realimentación de estado (N9), con P la única solución positiva definida para (N10). Bajo la estabilizabilidad de (A, B) y observabilidad $(S^{1/2}, A)$, existe una única solución positiva definida P para (N10). Más aún, se puede definir una función de valor óptima de estado-acción $Q(x(t), \bar{u}[t:\infty))$ como:

$$Q^*(x(t), \bar{u}[t:\infty)) = V^*(x(t)) = \begin{bmatrix} x \\ u \end{bmatrix}^\top H \begin{bmatrix} x \\ u \end{bmatrix} \tag{N11}$$

donde

$$\begin{aligned} H &= \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \\ &= \begin{bmatrix} \left(A - \frac{\gamma}{2}I\right)^\top P + P\left(A - \frac{\gamma}{2}I\right) + S & PB \\ B^\top P & R \end{bmatrix} \end{aligned}$$

Aplicando $\partial Q^*/\partial u = 0$ se obtiene el control óptimo:

$$\begin{aligned} u^*(t) &= -H_{22}^{-1}H_{21}x(t) \\ &= -R^{-1}B^\top Px(t) = -Kx(t) \end{aligned} \tag{N12}$$

Basado en (2.18), la función de valor de estado-acción Q para una póliza admisible $u = h(x(t))$ es definida por:

$$Q^h(x(t), \bar{u}(t)) = \int_t^{t+T} e^{-\gamma(\tau - t)} \left(x^\top S x + u^\top R u \right) d\tau + e^{-\gamma T} V^h(x(t+T)) \tag{2.19}$$

donde $\bar{u}(t) = \bar{u}[t : t+T) := \{u(\tau) : t \leq \tau < t+T\}$. Las condiciones de optimalidad para la póliza y función de valor usando funciones Q son las siguientes

$$Q^*(x(t), \bar{u}(t)) = \min_{h(x)} Q^h(x(t), \bar{u}(t)) \tag{2.20}$$

$$h^*(x(t)) = \operatorname*{argmin}_{\bar{u}(t)} Q^*(x(t), \bar{u}(t)) \tag{2.21}$$

$$V^*(x(t)) = \min_{\bar{u}(t)} Q^*(x(t), \bar{u}(t)). \tag{2.22}$$

Para cualquier póliza fija $h(x(t))$, se satisface que:

$$V^h(x(t+T)) = Q^h(x(t+T), \bar{u}(t+T)) \tag{2.23}$$

donde $\bar{u}(t+T) = \bar{u}[t+T : \infty) := \{u(\tau) : t+T \leq \tau \leq \infty\}$. Si se sustituye (2.23) en (2.18), se obtiene la ecuación de Bellman utilizando funciones Q:

$$Q^h(x(t), \bar{u}(t)) = \int_t^{t+T} e^{-\gamma(\tau - t)} \left(x^\top S x + u^\top R u \right) d\tau + e^{-\gamma T} Q^h(x(t+T), \bar{u}(t+T)). \tag{2.24}$$

En el Algoritmo 2.4 se presenta el algoritmo de Q-learning para sistemas en tiempo continuo.

Sin embargo no se puede implementar debido a su gran costo computacional, por lo tanto se requiere aproximar como se realizó anteriormente.

2.4. Aproximación del aprendizaje por refuerzo en tiempo continuo

La actualización de la función Q (2.25) es calculada para minimizar el error de diferencial temporal (TD) [71]

$$\xi(t) = Q^h(x(t), u(t))) - \int_t^{\infty} e^{-\gamma(\tau - t)} \left(x^\top S x + u^\top R u \right) d\tau \tag{2.27}$$

Algoritmo 2.4 Q-learning para sistemas en tiempo continuo

1: **Entrada:** γ, S y R

2: Elegir cualquier póliza admisible h^0. Sea el índice de iteración $j = 0$ hasta que converja.

3: **Actualización de la función** Q. Resolver la función Q utilizando

$$Q^{h^{j+1}}(x(t), \bar{u}(t)) = \int_t^{t+T} e^{-\gamma(\tau - t)} \left(x^\top S x + u^\top R u \right) d\tau + e^{-\gamma T} Q^{h^j}(x(t+T), \bar{u}(t+T)) \tag{2.25}$$

4: **Mejoramiento de la póliza** Actualizar la póliza de control

$$h^{j+1}(x(t)) = \underset{\bar{u}(t)}{\operatorname{argmin}}\, Q^{h^{j+1}}(x(t), \bar{u}(t)) \tag{2.26}$$

5: Actualizar $j = j + 1$ e ir al Paso 2.

Es simple de observar que la minimización del error TD (2.27) esta dada por al derivada temporal de (2.27)

$$\delta(t) = \min_t \dot{\xi}(t) = r(t) + \dot{Q}^h(x(t), u(t)) - \gamma Q^h(x(t), u(t)) \tag{2.28}$$

donde $r(t) = \rho(x(t), u(t)) = x^\top(t) S x(t) + u^\top(t) R u(t)$. La ecuación (2.28) es el equivalente en tiempo continuo del error TD presentado en [1].

Se considera la siguiente función objetivo:

$$E(t) = \frac{1}{2} |\delta(t)|^2. \tag{2.29}$$

A partir de (2.28), la regla de la cadena $\dot{Q}(x, u) = \frac{\partial Q(x,u)}{\partial x} \dot{x}(t) + \frac{\partial Q(x,u)}{\partial u} \dot{u}(t)$, donde $\dot{u} = \frac{\partial h(x)}{\partial x} \dot{x}(t)$ y la aproximación (2.2); el gradiente de la función objetivo respecto al vector de parámetros θ es:

$$\begin{aligned} \frac{\partial E(t)}{\partial \theta} &= \delta(t) \frac{\partial}{\partial \theta} \left[r(t) - \gamma \widehat{Q}(x, u; \theta) + \dot{\widehat{Q}}(x, u; \theta) \right] \\ &= \delta(t) \left[-\gamma \frac{\partial \widehat{Q}(x, u; \theta)}{\partial \theta} + \frac{\partial}{\partial \theta} \left(\frac{\partial \widehat{Q}(x, u; \theta)}{\partial x} \dot{x}(t) + \frac{\partial \widehat{Q}(x, u; \theta)}{\partial u} \dot{u}(t) \right) \right] \end{aligned} \tag{2.30}$$

Por lo tanto el algoritmo de gradiente descendente esta dado por $\dot{\theta} = -\alpha(t) \frac{\partial E(t)}{\partial \theta}$:

$$\dot{\theta} = \alpha(t)\delta(t)\left[\gamma\frac{\partial\widehat{Q}(x,u;\theta)}{\partial\theta} - \frac{\partial}{\partial\theta}\left(\frac{\partial\widehat{Q}(x,u;\theta)}{\partial x}\dot{x}(t) + \frac{\partial\widehat{Q}(x,u;\theta)}{\partial u}\dot{u}(t)\right)\right] \tag{2.31}$$

Un problema con el algoritmo anterior es su simetría en el tiempo. Debido a que la condición de frontera de la función de valor esta dada en $t \to \infty$, sería más apropiado actualizar los estimados pasados sin afectar los estimados futuros.

2.4.1. Diferenciación de Euler hacia atrás: Gradiente residual y TD(0)

Una forma de implementar el error TD es mediante la aproximación de Euler hacia atrás de la derivada $\dot{Q}(\cdot)$. Ignorando la póliza que se esta siguiendo y substituyendo

$$\dot{Q}(x(t),u(t)) = \frac{Q(x(t),u(t)) - Q(x(t-T),u(t-T))}{T}$$

en (2.28), se obtiene:

$$\delta(t) = r(t) + \frac{1}{T}\left[(1-\gamma T)\,Q(x(t),u(t)) - Q(x(t-T),u(t-T))\right] \tag{2.32}$$

Se nota que (2.32) coincide con el error TD convencional:

$$\delta_t = r_t + \gamma_2 Q_t(x_t,u_t) - Q_t(x_{t-T},u_{t-T}) \tag{2.33}$$

si se toma $\gamma_2 = 1-\gamma T$ y se reescala la función de valor como $Q_t(\cdot) = \frac{1}{T}Q(\cdot)$ con $T=1$. Entonces, el gradiente del error TD cuadrático (2.29) respecto al vector de parámetros $\vartheta = \theta(t-T)$ esta dada por:

$$\frac{\partial E(t)}{\partial\vartheta} = \frac{\delta(t)}{T}\left[(1-\gamma T)\frac{\partial}{\partial\vartheta}\left(\widehat{Q}(x(t),u(t);\vartheta) - \widehat{Q}(x(t-T),u(t-T);\vartheta)\right)\right] \tag{2.34}$$

Un algoritmo de gradiente descendiente simple utilizando la aproximación (2.2) esta dado por:

$$\dot{\theta} = \alpha(t)\frac{\delta(t)}{T}\left[-(1-\gamma T)\frac{\partial\widehat{Q}(x(t),u(t);\vartheta)}{\partial\vartheta} + \frac{\partial\widehat{Q}(x(t-T),u(t-T);\vartheta)}{\partial\vartheta}\right] \tag{2.35}$$

Una forma alternativa es únicamente actualizar $\widehat{Q}(x(t-T), u(t-T); \theta(t-T))$ sin modificar explícitamente $\widehat{Q}(x(t), u(t); \theta(t-T))$:

$$\dot{\theta}(t) = \alpha(t)\frac{\delta(t)}{T}\frac{\partial \widehat{Q}(x(t-T), u(t-T); \theta(t-T))}{\partial \theta(t-T)} \tag{2.36}$$

Las reglas de actualización (2.35) y (2.36) corresponden a los algoritmos de gradiente residual y TD(0), respectivamente. Otra forma de expresar (2.32) y 2.38) es mediante el siguiente desplazamiento:

$$\delta(t) = r(t+T) + \frac{1}{T}\left[(1-\gamma T)Q(x(t+T), u(t+T)) - Q(x(t), u(t))\right] \tag{2.37}$$

$$\dot{\theta}(t) = \alpha(t)\frac{\delta(t)}{T}\frac{\partial \widehat{Q}(x(t), u(t); \theta(t))}{\partial \theta(t)} \tag{2.38}$$

Algoritmo 2.5 Aproximación de Q-learning para sistemas en tiempo continuo

1: **Entrada:** $\gamma, \alpha(t), K$,
2: Ejecutar el Algoritmo de K-means 2.1,
3: Calcular las BFs $\phi_i(x, u)$ de acuerdo a los centroides obtenidos en el paso anterior,
4: Inicializar $x(0), \theta(0)$
5: Elegir una póliza inicial h^0 y sea $j = 0$ hasta que converja.
6: **Actualización del vector de parámetros**. Resolver el algoritmo de gradiente descendiente usando (2.38).
7: **Mejoramiento de la póliza**. Actualizar la póliza de control

$$h^{j+1} = \underset{u}{\operatorname{argmin}}\, \widehat{Q}^{h^{j+1}}(x, u; \theta) \tag{2.39}$$

8: Actualizar $j = j + 1$ y regresar al paso 5.

Si se utiliza el método de diferenciación hacia atrás de Euler en la derivada del vector de parámetros $\dot{\theta}$, se obtiene:

$$\theta^{h^{j+1}}(t+T) = \theta^{h^j}(t) + \alpha(t)\delta(t)\frac{\partial \widehat{Q}^{h^j}(x(t), u(t); \theta(t))}{\partial \theta(t)} \tag{2.40}$$

la cual es la misma expresión obtenida en las aproximaciones en tiempo discreto (2.14) y (2.15). La aproximación de Q-learning usando el algoritmo de clustering de K means se presenta en el Algoritmo 2.5.

El siguiente teorema da la convergencia de la aproximación del aprendizaje por refuerzo en tiempo continuo como:

$$\widehat{u}^*(t) = \widehat{h}^*(x(t)) = \underset{u}{\operatorname{argmin}}\, \widehat{Q}(x, u; \theta) \approx \underset{u}{\operatorname{argmin}}\, \Phi^\top(x, u)\theta \tag{2.41}$$

donde θ es actualizado por (2.38), $\Phi^\top(x, u)$ es obtenida de las BFs y el algoritmo de K-means, $\delta(t)$ es calculado mediante (2.37).

Teorema 4. *Si existe un punto de equilibrio θ^*, tal que*

$$\underset{u}{\text{mín}}[F(\theta)](x, u) = [F(\theta^*)](x, u) = \Phi^\top(x, u)\theta^*$$

entonces el punto de equilibrio es global y asintóticamente estable y además si el parámetro de aprendizaje satisface (1.28) entonces la función Q aproximada $\widehat{Q}(x, u; \theta)$ converge a un valor casi óptimo $\widehat{Q}^*$.

Para realizar la prueba del Teorema 4 se utiliza las siguientes suposiciones y teorema:

Suposición 1. *Existe un cilindro fijo de diámetro $\zeta > 0$ que contiene la trayectoria $\theta(t)$ de la ecuación diferencial ordinaria(ODE), $\dot{\theta} = f(\theta)$, en donde la suposición (1.28) es satisfecha.*

Suposición 2. *La ODE, $\dot{\theta} = f(\theta)$, es globalmente estable con un único punto de equilibrio estable θ^*.*

Suposición 3. *La ODE, $\dot{\theta} = f(\theta)$, puede ser escrita como:*

$$\dot{\theta} = \alpha(t)Y(\theta, O(t)) = f(\theta) \tag{2.42}$$

donde $O(t) = f(x(t), u(t), x(t + T))$ es una función de mediciones del estado y control. Además $Y(\cdot)$ satisface:

$$\|Y(\theta, O(t))\| \leq \chi(1 + \|\theta\|)^2 \tag{2.43}$$

para alguna constante $\chi > 0$. Esta suposición se refiere a que la norma de la función Y depende únicamente del vector de parámetros θ y no del estado y control.

Teorema 5. *Si las suposiciones 1-3 y la propiedad de contracción son satisfechas. Entonces se tienen las siguientes propiedades:*

1. *Si existe una función positiva* $W \in \mathbb{R}$ *de clase* C^2 *con derivadas acotadas tal que para toda* θ, $|\theta| \geq \varrho_0$

   (i) $W'(\theta) \cdot f(\theta) \leq 0$

   (ii) $W(\theta) \geq \psi|\theta|^2, \quad \psi > 0$

   entonces para toda acción u *y estado* x, *la dinámica* θ *es acotada y asintóticamente estable;*

2. *Si además existe* θ^* *tal que*

   (i)' $W'(\theta) \cdot f(\theta) < 0$ *para todo* $\theta \neq \theta^*$.

   (iii) $W(\theta) = 0$ *ssi* $\theta = \theta^*$

   entonces la dinámica θ *converge asintóticamente a* θ^*.

Demostración. La prueba del Teorema 5 viene dado en [58]. □

Mediante el teorema anterior se procede a probar el Teorema 4.

Demostración. Debido a que el estado y control son finitos y acotados por lo tanto O también, y además como las BFs $\Phi(\cdot)$ son acotadas entonces también las recompensas $r(t+T) = \rho(x,u)$ son acotadas. Por lo tanto Y satisface el supuesto 3 y (2.43).

Considerando (2.32), el vector de parámetros θ del aprendizaje por refuerzo en tiempo continuo (2.38) es actualizado por:

$$\dot{\theta} = \alpha(t)\left(r(t+T) + \frac{1}{T}(1-\gamma T)\Phi^\top(t+T)\theta(t) - \frac{1}{T}\Phi^\top(t)\,\theta(t)\right)\Phi(t) = f(\theta) \tag{2.44}$$

donde $\Phi(t+T) = \Phi(x(t+T), u(t+T))$ y $\Phi(t) = \Phi(x(t), u(t))$. El punto de equilibrio θ^* de (2.44) satisface la relación (2.11)

$$\theta^* = (\mathcal{P} \circ \mathcal{H} \circ F)(\theta^*) \tag{2.45}$$

donde $\mathcal{H}$ es el operador de Bellman, $\mathcal{P}$ es un operador de proyección. El operador de Bellman $\mathcal{H}$ viene dado por:

$$[\mathcal{H}(Q)](x,u) = r(t+T) + \frac{1}{T}\left((1-\gamma T)Q(x(t+T), u(t+T)) - Q(x(t), u(t))\right) \tag{2.46}$$

El operador de proyección $\mathcal{P}$ es definido por la proyección de la pseudo-inversa como:

$$[\mathcal{P}\widehat{Q}(x,u;\theta)] = \left[\Phi(x,u)\Phi^\top(x,u)\right]^{-1}\Phi(x,u)\widehat{Q}(x,u;\theta).$$

Para garantizar que el operador $\mathcal{H}$ es una contracción en θ, se requiere:

$$\sum_{i=1}^{K}|\phi_i(x,u)| \leq 1 \tag{2.47}$$

La condición (2.47) es evidente porque las funciones base son funciones Gaussianas. La función $f(\theta)$ puede ser reescrita como:

$$f(\theta) = f_1(\theta) + f_2(\theta)$$

con $f_1(\theta) = \left(r(t+T) + \frac{1}{T}(1-\gamma T)\Phi^\top(t+T)\theta\right)\Phi(t)$, $f_2(\theta) = -\frac{1}{T}\Phi(t)\Phi^\top(t)\theta$. Por la propiedad de contracción se tiene

$$\begin{aligned}\|f_1(\theta_1) - f_1(\theta_2)\| &\leq \frac{1-\gamma T}{T}\|\theta_1 - \theta_2\| \\ \|f_2(\theta_1) - f_2(\theta_2)\| &\leq \frac{1}{T}\|\theta_1 - \theta_2\|\end{aligned} \tag{2.48}$$

Por el otro lado, cualquier punto de equilibrio θ^* satisface $f(\theta^*) = 0$, ó de forma equivalente

$$f(\theta) = f(\theta) - f(\theta^*)$$

Para cualquier norma p se satisface

$$\frac{d}{dt}\|\theta(t)-\theta^*\|_p = \|\theta(t)-\theta^*\|_p^{1-p}\sum_{i=1}^{K}(\theta(t)_i - \theta_i^*)^{p-1}\cdot\{[f_1(\theta)_i - f_1(\theta^*)_i] - [f_2(\theta)_i - f_2(\theta^*)_i]\}$$

donde el subíndice i indica cada componente del vector de parámetros θ. Se utiliza la desigualdad de Hölder

$$\frac{d}{dt}\|\theta(t)-\theta^*\|_p \leq \|f_1(\theta) - f_1(\theta^*)\|_p - \|f_2(\theta) - f_2(\theta^*)\|_p.$$

Tomando el límite como $p \to \infty$ y utilizando (2.48) conlleva a la siguiente desigualdad

$$\frac{d}{dt}\|\theta(t)-\theta^*\| \leq -\gamma\|\theta(t)-\theta^*\| \tag{2.49}$$

La solución de la ODE (2.49) es

$$\|\theta(t) - \theta^*\| \leq e^{-\gamma t} \|\theta(0) - \theta^*\| \tag{2.50}$$

La solución (2.50) implica que el punto de equilibrio θ^* de (2.44) es globalmente asintóticamente estable. Esto satisface el supuesto 2. En el punto de equilibrio θ^* se satisface

$$\begin{aligned} f(\theta^*) &= f_1(\theta^*) + f_2(\theta^*) = 0 \\ f_1(\theta^*) &= -f_2(\theta^*) \end{aligned}$$

El punto de equilibrio θ^* se obtiene al minimizar la función $\widehat{Q}$,

$$\theta^* = \Phi^+(t) \left(r(t+T) + (1 - \gamma T) \min_u \Phi^\top(t+T)\theta^* \right) \tag{2.51}$$

donde $\Phi^+(\cdot) = \left(\Phi(t)\Phi^\top(t)\right)^{-1} \Phi(t)$ es la pseudo-inversa de Moore-Penrose. A partir de las tres condiciones (1.28), (2.43) y (2.50), entonces existe una función positiva $W(\theta) \in C^2$ con segunda derivada acorada, tal que [58]

$$\frac{dW(\theta)}{dt} = \frac{\partial W(\theta)}{\partial \theta} f(\theta) \leq 0, \quad W(\theta) = 0, \text{ ssi } \theta = \theta^* \tag{2.52}$$

donde $f(\theta)$ es definida en (2.44). Nótese que, debido a que $\alpha(t) \in (0, 1]$, (1.28) requiere que todos los pares de estado-acción sean visitados infinitamente seguido. Esto puede ser resuelto por medio de los métodos de exploración ε-codicioso o softmax, los cuales son análogos a la condición de excitación persistente (PE) de los sistemas de identificación. Se observa que en (2.52) para cualquier $\frac{\partial W(\theta)}{\partial \theta} > 0$, su primer derivada tiende a zero suficientemente rápido. A partir del Teorema 17 de [58], $\theta(t)$ converge a un valor casi óptimo θ^*, y el aprendizaje por refuerzo en tiempo continuo (2.37), (2.38) y (2.41) converge a $\widehat{Q}^*(x, u; \theta^*)$. □

2.5. Conclusión

En este capítulo se extiende la teoría presentada en el capítulo anterior para lidiar con espacios de estado-acción grandes o continuos. Se enfocó principalmente en aproximadores paramétricos apoyados del algoritmo de agrupamientos conocido como K-means, el cual sirvió para dividir el espacio en clusters que minimicen la distancia que existe entre cada

uno de sus elementos con su centroide. Cada centroide fungía como centro de las funciones base a diseñar.

Los algoritmos de aproximación se encuentran basados en el método de gradiente descendente. Se realizaron las aproximaciones para los algoritmos vistos en el capítulo anterior, en especial Q-learning. La convergencia de los algoritmos en tiempo discreto con espacios grandes y tiempo continuo son presentados mediante el uso de la propiedad de contracción de la función de valor.

Capítulo 3

Aprendizaje por Refuerzo Robusto

En los capítulos anteriores se ha abordado la teoría necesaria del aprendizaje por refuerzo en tiempo discreto y continuo. El problema de control ha sido resuelto bajo el enfoque del control óptimo y adaptable en la búsqueda de pólizas de control óptimas sin el conocimiento de la dinámica del ambiente (sistema).

En este capítulo se aborda el aprendizaje por refuerzo bajo un enfoque de control óptimo-robusto, en donde se busca encontrar pólizas de control óptimas y robustas en presencia de perturbaciones.

3.1. Control Robusto en Tiempo discreto

Se considera el siguiente sistema no lineal:

$$x_{t+1} = f(x_t) + g_1(x_t)\omega_t + g_2(x_t)u_t \tag{3.1}$$

donde $f(x_t) \in \mathbb{R}^n, g_1(x_t) \in \mathbb{R}^{n\times\omega}, g_2(x_t) \in \mathbb{R}^{n\times m}$ define la dinámica del sistema, $x_t \in \mathbb{R}^n$ es el estado, $u_t \in \mathbb{R}^m$ es el control, y las perturbaciones son descritas por $\omega_t \in \mathbb{R}^\omega$.

Si $\omega_t = 0$ y el control $u(x_t)$ es una ley de control admisible que minimiza la siguiente función de costo

$$J_2(x_t, u_t) = \sum_{i=0}^{\infty}(x_i^\top S x_i + u_i^\top R u_i), \tag{3.2}$$

entonces el controlador $u(x_t)$ se le conoce como la solución del control $\mathcal{H}_2$, donde $S \in \mathbb{R}^{n\times n}$ y $R \in \mathbb{R}^{m\times m}$ son las matrices de peso del problema $\mathcal{H}_2$. Por lo tanto, el problema a resolver puede ser definido como

$$V(x_t) = \min_u \left(\sum_{i=t}^{\infty} (x_i^\top S x_i + u_i^\top R u_i) \right), \forall x.$$

La función de valor puede ser definida como la ecuación de Bellman

$$\begin{aligned} V(x_t) &= \sum_{i=t}^{\infty} (x_i^\top S x_i + u_i^\top R u_i) \\ &= r_{t+1} + V(x_{t+1}), \end{aligned} \tag{3.3}$$

con la recompensa definida como $r_{t+1} = x_t^\top S x_t + u_t^\top R u_t$. El principio de optimalidad de Bellman arroja la función de valor óptima

$$V^*(x_t) = \min_u \left[r_{t+1} + V^*(x_{t+1}) \right]. \tag{3.4}$$

El control óptimo se obtiene como:

$$\begin{aligned} u^*(x_t) &= \operatorname*{argmin}_u \left[r_{t+1} + V^*(x_{t+1}) \right] \\ &= -\frac{1}{2} R^{-1} g_2^\top(x_t) \frac{\partial V^*(x_{t+1})}{\partial x_{t+1}}. \end{aligned} \tag{3.5}$$

Si $\omega_t \neq 0$ y el control $u(x_t)$ es una ley de control admisible que minimiza la función de costo

$$J_\infty(x_t, u_t, \omega_t) = \sum_{i=0}^{\infty} \left(x_i^\top S x_i + u_i^\top R u_i - \eta^2 \omega_i^\top \omega_i \right), \tag{3.6}$$

entonces $u(x_t)$ es la solución del problema de control $\mathcal{H}_\infty$, donde η es un factor de atenuación. De forma similar al caso del problema de control $\mathcal{H}_2$, la función de valor es definida como

$$V(x_t) = x_t^\top S x_t + u_t^\top R u_t - \eta^2 \omega_t^\top \omega_t + V(x_{t+1}).$$

La función de valor óptima puede ser obtenida al resolver un juego diferencial de suma-cero como:

$$V^*(x_t) = \min_u \max_\omega J_\infty(x_t, u, \omega).$$

Para resolver el juego de suma-cero, se requiere obtener la solución de la ecuación de Hamilton-Jacobi-Isaacs (HJI):

$$V^*(x_t) = x_t^\top S x_t + u_t^{*\top} R u_t^* - \eta^2 \omega_t^{*\top} \omega_t^* + V^*(x_{t+1}).$$

El control óptimo y la peor perturbación son:

$$u^*(x_t) = -\frac{1}{2}R^{-1}g_2^\top(x_t)\frac{\partial V^*(x_{t+1})}{\partial x_{t+1}} \tag{3.7}$$

$$\omega_t^* = \frac{1}{2\eta^2}g_1^\top(x_t)\frac{\partial V^*(x_{t+1})}{\partial x_{t+1}} \tag{3.8}$$

El control (3.7) es la solución del control robusto y (3.8) es la peor perturbación. Generalmente el control $\mathcal{H}_2$ no tiene buen desempeño en contra de perturbaciones, mientras que el control $\mathcal{H}_\infty$ tiene un desempeño pobre y buena robustez. Para unir las ventajas de cada problema de control surge el problema híbrido $\mathcal{H}_2/\mathcal{H}_\infty$, el cual es un problema de optimización con restricciones:

$$\begin{aligned} &\min_u J_2(x_t, u) \\ &\text{Sujeto a : } \ J_\infty(x_t, u_t, \omega_t) \leq 0 \end{aligned} \tag{3.9}$$

Una forma de obtener el control $\mathcal{H}_2/\mathcal{H}_\infty$ es mediante la siguiente parametrización:

$$u_\xi^*(x_t) = \xi u(x_t) + (\xi - 1)\frac{R^{-1}}{2}g_2^\top(x_t)\frac{\partial J_2^\xi(x_{t+1})}{\partial x_{t+1}} \tag{3.10}$$

donde $\xi \in (-1, 1)$. El parámetro ξ ayuda a obtener una familia de controladores estabilizantes que son robustos y óptimos, sin embargo como se vio en los capítulos anteriores, los controles (3.5), (3.7) y (3.10) requieren conocimiento de la dinámica del sistema.

3.2. Aprendizaje Robusto: Tiempo discreto

En la siguiente sección se abordará el problema de aprendizaje por refuerzo robusto en tiempo discreto. Los espacios de estado-acción son grandes.

3.2.1. Problema de Sobreestimación

Como se vio en los capítulos anteriores una forma de obtener pólizas óptimas sin el conocimiento de la dinámica del sistema es mediante el aprendizaje por refuerzo. Considérese el método Q-learning (1.26)

$$Q_{t+1}(x_t, u_t) = Q_t(x_t, u_t) + \alpha_t\left(q_t - Q_t(x_t, u_t)\right) \tag{3.11}$$

donde $q_t = r_{t+1} + \gamma \min_u Q(x_{t+1}, u)$ es el objetivo de aprendizaje. Debido a que las funciones de valor están en función de la recompensa y además esta última ha sido diseñado con base a un problema de control $\mathcal{H}_2$, entonces es posible diseñar un control híbrido $\mathcal{H}_2/\mathcal{H}_\infty$ mediante el diseño de una recompensa adecuada. La función de valor óptima y robusta se obtiene mediante el principio de optimalidad de Bellman como:

$$\begin{aligned} &Q^*(x_t, u_t) = \min_h Q^h(x_t, u_t) \\ &\text{Sujeto a: } \|u_t\| \leq \bar{u},\ \|x_t\| \leq \bar{x},\ \|\omega\| \leq \bar{\omega} \end{aligned} \tag{3.12}$$

donde $\bar{u}, \bar{x}$ y $\bar{\omega}$ son cotas superiores de las variables. Si la recompensa es diseñada para satisfacer (3.12), entonces la póliza de control $u_t = h(x_t)$ es una póliza óptima y robusta y la función de valor óptima Q^* es robusta en presencia de perturbaciones, $\omega \leq \bar{\omega}$. La recompensa puede ser estática o dinámica. La recompensa típica es diseñada de forma cuadrática y es resuelta mediante métodos de optimización convexa.

Sin embargo la principal desventaja de los métodos de diferencia temporal es la sobreestimación de los valores de las acciones de control. En una regla de actualización no-basada en la póliza como Q-learning, la sobreestimación se debe al tomar la función de valor mínima como una aproximación de la función de valor mínima esperada. En una regla de actualización basada en la póliza como Sarsa [18], la sobre-estimación se debe a las técnicas de exploración, e.g., exploración ε-codicioso y softmax. Una forma de resolver el problema de sobre-estimación es mediante la técnica del doble estimador, sin embargo existe la posibilidad que el algoritmo subestime los valores de las acciones de control.

El doble estimador divide el conjunto de muestras $D = \bigcup_{i=1}^{N} D_i$ en dos subconjuntos disjuntos, D^A y D^B, y utiliza dos conjuntos de estimadores sin sesgo $\mu^A = \{\mu_1^A, ..., \mu_N^A\}$ y $\mu^B = \{\mu_1^B, ..., \mu_N^B\}$ tal que $E\{\mu_i^A\} = E\{\mu_i^B\} = E\{Q_i\}$ para todo i. Los dos estimadores aprenden un estimado independiente del valor real de $E\{Q_i\}$ para todo i. Uno de estos estimadores, e.g., μ^A, es usado para determinar la acción minimizadora $u^* = \operatorname{argmin}_u \mu^A(u)$, y la otra, μ^B, provee el estimado de su valor, $\mu^B(u^*) = \mu^B(\operatorname{argmin}_u \mu^A(u))$. Este estimado no tendrá sesgo en el sentido que $E\{\mu^B(u^*)\} = Q(u^*)$ [1]. Cuando las variables son i.i.d., el doble estimador no presenta sesgo debido a que todos los valores esperados son iguales y $\Pr(u^* \in \operatorname{argmin}_i E\{Q_i\}) = 1$. Para adquirir experiencia, el algoritmo requiere explorar suficientemente el par estado-acción (x, u).

En el capítulo anterior, se abordó el problema de dimensionalidad y el cómo utilizar los aproximadores ya sean paramétricos o no paramétricos. En el caso de los aproximadores paramétricos, las BFs deben ser definidas por adelantado con conocimiento previo del

comportamiento del sistema; por el otro lado, los aproximadores no paramétricos toman una ventaja importante debido a que pueden ser aplicado a cualquier sistema y tiene menor número de parámetros a sintonizar.

Existen otra familia de aproximadores basados en aproximadores no paramétrico utilizando un enfoque probabilistico. La regla de actualización (3.11) puede ser obtenida directamente de la fórmula de valor esperado de una variable discreta [72], i.e., el valor esperado μ de una variable discreta aleatoria q con valores $q_1, q_2, \ldots, q_n$ y probabilidades representadas por $p(q_1), p(q_2), \ldots, p(q_n)$ es calculado por:

$$\mu = \sum_{i=1}^{n} q_i p(q_i), \tag{3.13}$$

para una variable discreta con tan sólo dos posibles valores, la fórmula anterior se reescribe como:

$$\mu = (1-\alpha)p + q\alpha \Longrightarrow \mu = (1-\alpha)\mu + q\alpha. \tag{3.14}$$

reemplazando μ por $Q_t(x,u)$ y q por el objetivo de aprendizaje, se obtiene el algoritmo de Q-learning (1.26).

A continuación se brindará una extensión del capítulo anterior usando un aproximador no paramétrico basado en el algoritmo de k-vecinos cercanos y la técnica del doble estimador para resolver el problema de sobre-estimación y dimensionalidad de los algoritmos de aprendizaje por refuerzo.

3.2.2. Estimación del valor Esperado Mínimo

Se considera un conjunto de N variables aleatorias $X = \{X_1, \ldots, X_N\}$. Se desea calcular el valor esperado mínimo de ese conjunto:

$$\min_i E\{X_i\} \tag{3.15}$$

Cuando no se tiene conocimiento de la forma funcional y parámetros de la distribución de las variables X_i, entonces es imposible determinar (3.15) de forma exacta. La mayoría de las veces, este valor es aproximado mediante la construcción de aproximadores para $E\{X_i\}$ para toda i. Sea $D = \cup_{i=1}^{N} D_i$ el conjunto de muestras, donde D_i es un subconjunto de muestras de la variable X_i. Se asume que las muestras en D_i son independientes e idénticamente distribuidas (i.i.d). Estimados sin sesgo para el valor esperado pueden ser obtenidos al computar el promedio muestral de cada variable: $E\{X_i\} = E\{\mu_i\} \approx \mu_i(D) \triangleq$

$\frac{1}{|D_i|}\sum_{d\in D_i} d$, donde μ_i es un estimado de la variable X_i. Este aproximador no tiene sesgo debido a que cada muestra $d \in D_i$ es un estimado sin sesgo del valor de $E\{X_i\}$. Por lo tanto el error en la aproximación consiste únicamente en la varianza del estimador y disminuye cuando se obtienen más muestras.

Se utiliza la siguiente notación: f_i denota la función de densidad de probabilidad (PDF) de la i-ésima variable X_i y $F_i(x) = \int_{-\infty}^{x} f_i(z)dz$ es la función de distribución acumulada (CDF) de la PDF. De forma similar, la PDF y CDF del i-ésimo estimador es denotado por f_i^μ y F_i^μ. El valor esperado mínimo puede ser expresado en términos de las PDF como $\text{mín}_i\, E\{X_i\} = \text{mín}_i \int_{-\infty}^{\infty} x f_i(x)dx$.

El estimador simple

Una forma simple de aproximar el valor en (3.15) es utilizando el valor del estimador mínimo

$$\min_i E\{X_i\} = \min_i E\{\mu_i\} \approx \min_i \mu_i(D). \tag{3.16}$$

Q-learning utiliza este método para aproximar el valor del siguiente estado al minimizar sobre los valores de las acciones estimadas en ese estado.

El estimador mínimo $\text{mín}_i\, \mu_i$ es distribuido de acuerdo a alguna PDF $f^\mu_{\text{mín}}$ que depende de las PDF del estimador f_i^μ. Para determinar esta PDF, se considera la CDP $F^\mu_{\text{mín}}(x)$, la cual da la probabilidad que el estimador mínimo es mayor o igual a x. Esta probabilidad es igual a la probabilidad de que todos los estimador son mayores a x: $F^\mu_{\text{mín}}(x) \triangleq \Pr(\text{mín}_i\, \mu_i \geq x) = \prod_{i=1}^N \Pr(\mu_i \geq x) \triangleq \prod_{i=1}^N F_i^\mu(x)$. El valor $\text{mín}_i\, \mu_i(D)$ es un estimado sin sesgo para $E\{\text{mín}_j\, \mu_j\} = \int_{-\infty}^{\infty} x f^\mu_{\text{mín}}(x)dx$, que por lo tanto esta dado por:

$$E\{\min_j \mu_j\} = \int_{-\infty}^{\infty} x \frac{d}{dx} \prod_{i=1}^{N} F_i^\mu(x)dx = \sum_j^N \int_{-\infty}^{\infty} x f_j^\mu(d) \prod_{i\neq j}^{N} F_i^\mu(x)dx. \tag{3.17}$$

Sin embargo, en (3.15) el orden del operador min y el operador esperanza matemática es al revés. Esto hace que el estimador mínimo $\text{mín}_i\, \mu_i(D)$ sea un estimado con sesgo para el $\text{mín}_i\, E\{X_i\}$.

El doble estimador

La sobre-estimación que resulta del estimador simple puede tener un gran impacto negativo en algoritmos que utilizan este método, como Q-learning. Por lo tanto, un método al-

ternativo para aproximar $\min_i E\{X_i\}$ es mediante el método llamado doble estimador. Este estimador usa dos conjuntos de estimadores: $\mu^A = \{\mu_1^A, \cdots, \mu_N^A\}$ y $\mu^B = \{\mu_1^B, \cdots, \mu_N^B\}$.

Ambos conjuntos de estimadores son actualizados con un subconjunto de las muestras tomadas, tal que $D = D^A \cup D^B$ y $D^A \cap D^B = \emptyset$ y $\mu_i^A(D) = \frac{1}{|D_i^A|} \sum_{d \in D_i^A} d$ y $\mu_i^B(D) = \frac{1}{|D_i^B|} \sum_{d \in D_i^B} d$. Similar al estimador simple μ_i, tanto μ_i^A y μ_i^B no presentan sesgo si se asume que todas las muestras están divididas de una forma adecuada, por ejemplo al azar, sobre los dos conjuntos de estimadores. Sea $Min^A(D) \triangleq \{j | \mu_j^A(D) = \min_i \mu_i^A(D)\}$ el conjunto de estimadores mínimos en $\mu^A(D)$. Debido que μ^B es un conjunto de estimadores independientes sin sesgo, se tiene $E\{\mu_j^B\} = E\{X_j\}$ para toda j, incluyendo $j \in Min^A(D)$. Sea a^* un estimador que minimiza $\mu^A : \mu_{a^*}^A(D) \triangleq \min_i \mu_i^A(D)$. Si existen múltiples estimadores que minimicen μ^A, entonces se puede utilizar $\mu_{a^*}^B$ como un estimado para $\min_i E\{\mu_i^B\}$ y por lo tanto también para $\min_i E\{X_i\}$ y se obtiene la aproximación:

$$\min_i E\{X_i\} = \min_i E\{\mu_i^B\} \approx \mu_{a^*}^B. \tag{3.18}$$

A medida que se obtienen más muestras la varianza de los estimadores decrece. En el límite, $\mu_i^A(D) = \mu_i^B(D) = E\{X_i\}$ para toda i y la aproximación en (3.18) converge al resultado correcto.

Se asume que las PDF son continuas. La probabilidad $\Pr(j = a^*)$ para cualquier j es entonces igual a la probabilidad que para todas $i \neq j$ arroja estimadores mayores. Por lo tanto $\mu_j^A(D) = x$ es mínimo para algún valor x con probabilidad $\prod_{i \neq j}^N \Pr(x < \mu_i^A)$. Integrando respecto a x da $\Pr(j = a^*) = \int_{-\infty}^{\infty} \Pr(\mu_j^A = x) \prod_{i \neq j}^N \Pr(x < \mu_i^A) dx \triangleq \int_{-\infty}^{\infty} f_j^A(x) \prod_{i \neq j}^N F_i^A(x) dx$, donde f_i^A y F_i^A son la PDF y CDF de μ_i^A. El valor esperado de la aproximación por el doble estimador esta dado por:

$$\sum_j^N \Pr(j = a^*) E\{\mu_j^B\} = \sum_j^N E\{\mu_j^B\} \int_{-\infty}^{\infty} f_j^A(x) \prod_{i \neq j}^N F_i^A(x) dx. \tag{3.19}$$

Si se compara (3.19) con (3.17), se observa que la diferencia es que el doble estimador utiliza $E\{\mu_j^B\}$ en lugar de x. El estimador simple sobrestima, porque x esta dentro de la integral y por lo tanto correlaciona con el producto monótamente creciente $\prod_{i \neq j} F_i^{\mu}(x)$. El doble estimador subestima porque las probabilidades $\Pr(j = a^*)$ suman a uno y por lo tanto el aproximador es un estimador ponderado de valores esperados sin sesgo, los cuales deben ser mayores o igual al valor esperado mínimo.

En el siguiente Lema, que es válido para el caso discreto y continuo, se demuestra de

forma general que el estimado $E\{\mu^B_{a^*}\}$ es un estimado sin sesgo de $\min_i E\{X_i\}$.

Lema 2. *Sea $X = \{X_1, ..., X_N\}$ un conjunto de variables aleatorias y $\mu^A = \{\mu^A_1, ..., \mu^A_N\}$ y $\mu^B = \{\mu^B_1, ..., \mu^B_N\}$ dos conjuntos de estimadores sin sesgo tal que $E\{\mu^A_i\} = E\{\mu^B_i\} = E\{X_i\}$, para toda i. Sea $\mathcal{M} \triangleq \{j | E\{X_j\} = \min E\{X_i\}\}$ el conjunto de elementos que minimizan los valores esperados. Sea a^* un elemento que minimiza $\mu^A : \mu^A_{a^*} = \min_i \mu^A_i$. Entonces $E\{\mu^B_{a^*}\} = E\{X_{a^*}\} \geq \min_i E\{X_i\}$. Más aún, la desigualdad es estricta sí y sólo sí $Pr(a^* \notin \mathcal{M}) > 0$.*

Demostración. Se asume que $a^* \in \mathcal{M}$. Entonces $E\{\mu^B_{a^*}\} = E\{X_{a^*}\} \triangleq \min_i E\{X_i\}$. Ahora se asume que $a^* \notin \mathcal{M}$ y se elige $j \in \mathcal{M}$. Entonces $E\{\mu^B_{a^*}\} = E\{X_{a^*}\} > E\{X_j\} \triangleq \min_i E\{X_i\}$. Estas dos posibilidades son mutuamente exclusivas, entonces la expectación combinada puede ser expresada por:

$$\begin{aligned} E\{\mu^B_{a^*}\} =& \Pr(a^* \in \mathcal{M}) E\{\mu^B_{a^*} | a^* \in \mathcal{M}\} + \Pr(a^* \notin \mathcal{M}) E\{\mu^B_{a^*} | a^* \notin \mathcal{M}\} \\ =& \Pr(a^* \in \mathcal{M}) \min_i E\{X_i\} + \Pr(a^* \notin \mathcal{M}) E\{\mu^B_{a^*} | a^* \notin \mathcal{M}\} \\ \geq& \Pr(a^* \in \mathcal{M}) \min_i E\{X_i\} + \Pr(a^* \notin \mathcal{M}) \min_i E\{X_i\} = \min_i E\{X_i\} \end{aligned}$$

donde la desigualdad es estricta sí y sólo sí $\Pr(a^* \notin \mathcal{M}) > 0$. Esto sucede cuando las variables tienen diferentes valores esperados, pero sus distribuciones coinciden. A diferencia del estimador simple, el doble estimador no presenta sesgo cuando las variables son i.i.d., debido a que todos sus valores esperados son igual a $P(a^* \in \mathcal{M}) = 1$. □

3.2.3. La regla de kNN

El algoritmo de k vecinos cercanos (kNN) es un método para clasificación de patrones. La regla de kNN clasifica ejemplos de entrenamiento sin etiquetar con la etiqueta mayor entre sus k-vecinos cercanos en el espacio de característica. Su desempeño depende de la distancia métrica utilizada para identificar el vecino cercano, generalmente se utiliza la métrica Euclidiana que se define como:

$$d = \sqrt{\sum_{i=1}^{n} (x_i - x)^2} \tag{3.20}$$

donde $x_1, x_2, ..., x_n$ son ejemplos de entrenamiento. La etiqueta de la clase asignada al ejemplo de prueba es determinado por el voto mayoritario de sus k vecinos cercanos.

Lema 3 (Convergencia de los k-vecinos cercanos [73]). *Sean x y $x_1, x_2, ...$ un conjunto de variables aleatorias independientes e idénticamente distribuidas (i.i.d) que toman valores en un espacio métrico separable X. Sea $y_k(x)$ el k-ésimo vecino cercano de x del conjunto $\{x_1, x_2, ..., x_n\}$ y se define $N_k(x) = \{y_1(x), .., y_k(x)\}$ como el conjunto de k-vecinos cercanos. Entonces para k tal que $k/n \to 0$, cuando $n \to \infty$, se tiene que $\|y_{1:k}(x) - x\| \to 0$ con probabilidad uno (w.p.1).*

El Lema anterior establece que el k-ésimo vecino cercano a x converge a x con probabilidad uno (w.p.1) cuando el tamaño de muestreo n incrementa con k fijo.

Observación 3. *En particular $y_k(x) \to x$ con probabilidad uno (w.p.1) para cualquier medida de probabilidad en un espacio Euclidiano n-dimensional.*

Demostración. Sea $S_x(r)$ una esfera $\{\tilde{x} \in X : d(x, \tilde{x}) \leq r\}$ de radio r centrado en x, donde d es la métrica definida en X. Se considera primero un punto $x \in X$ que tiene la propiedad que toda esfera $S_x(r), r > 0$, tiene una medida de probabilidad diferente de cero. Entonces, para cualquier $\delta > 0$,

$$\Pr\left\{\min_{k=1,2,...,n} d(x_k, x) \geq \delta\right\} = (1 - P(S_x(\delta))^n \to 0 \tag{3.21}$$

y por lo tanto, debido a que $d(x_k, x)$ es monóticamente decreciente en k, el vecino cercano a x converge a x con proabililidad uno [73]. □

En [73] se prueba que el error del algoritmo de kNN se encuentra entre el error Bayesiano y dos veces el error Bayesiano, como se muestra:

$$R^* \leq R \leq R^*\left(2 - \frac{M}{M-1}R^*\right) \tag{3.22}$$

donde R^* es el error de Bayes, R es el error de kNN y M es el número de clases de todo el conjunto de datos. En este trabajo se tiene únicamente dos clases; Clase A: una variable x_i en el espacio métrico X que satisface $x \in N_k(x)$ y la Clase B: el negado de la Clase A, entonces (3.22) se reescribe como:

$$R^* \leq R \leq 2R^*(1 - R^*) \leq 2R^*. \tag{3.23}$$

Cuando el conjunto de datos n se aproxima a infinito, entonces una k grande se desempeña mejor. Cuando $k \to \infty$, el desempeño del algoritmo es el óptimo, y el error se aproxima al error Bayesiano en el límite [74].

3.2.4. El algoritmo kNN-TD

El algoritmo de kNN es un aproximador no-paramétrico cuyas principales ventajas es el uso de espacios de estado grandes y un esquema de percepción con el estado actual. Primero, se requiere determinar el conjunto de los k-vecinos cercanos ($kNN = kNN_t$) del estado actual x, donde a cada vecino se le asociará un predictor de la función de valor $Q(i, u)$ y un peso w_i que es calculado por [72]:

$$w_i = \frac{1}{1 + d_i^2} \quad \forall i \in [1, ..., k]. \tag{3.24}$$

Para cada paso de tiempo se tendrá k clasificadores activos cuyos pesos vienen dados por la inversa de la distancia Euclidiana d_i. Después que el conjunto de kNN es obtenido, se requiere calcular la distribución de probabilidad $p(kNN)$ sobre kNN. Las probabilidades están implicadas por los pesos actuales $\{w\}$. Este vector debe ser normalizado para expresar una distribución de probabilidad $p(kNN)$. Las probabilidades son calculadas mediante:

$$p(i) = \frac{w_i}{\sum w_i} \quad \forall i \in kNN. \tag{3.25}$$

Para la selección de la acción de control, el algoritmo de kNN calcula un valor esperado por cada acción mediante un proceso colectivo de predictores. Debido a que este proceso involucra muchos predictores diferentes, uno por cada elemento del conjunto de kNN, entonces el proceso de reduce en estimar un valor esperado por cada objetivo de aprendizaje de los k vecinos:

$$\langle \mathcal{Q}(kNN, u) \rangle = \sum_{i=1}^{kNN} Q(i, u) p(i) \tag{3.26}$$

donde $p(i)$ adquiere el significado de la probabilidad $\Pr(\mathcal{Q}(kNN, u) = Q(i, u)|x_t)$ de que $\mathcal{Q}(kNN, u)$ tome el valor de $Q(i, u)$ dado el estado x_t en el instante de tiempo t. El mecanismo de selección de la acción puede ser obtenida de forma codiciosa del valor esperado como:

$$u^* = \underset{u}{\operatorname{argmin}}\, \mathcal{Q}(kNN, u). \tag{3.27}$$

El algoritmo de TD requiere una estimación del total de recompensa recibida que puede ser obtenida mediante la función de valor del estado actual x_{t+1}, i.e., $\min_u \mathcal{Q}(kNN', u)$ en un aprendizaje no basado en la póliza o por $\mathcal{Q}(kNN', u')$ en un aprendizaje basado en la póliza. El valor esperado para cualquier método para hallar la póliza es calculado por:

$$\langle \mathcal{Q}(kNN', u) \rangle = \sum_{i=1}^{kNN'} Q(i, u) p'(i) \tag{3.28}$$

donde $kNN' = kNN_{t+1}$ es el conjunto de los k-vecinos cercanos de x_{t+1}, $p'(i)$ son las probabilidades de cada vecino en el conjunto kNN'. El error TD (δ) para un algoritmo basado y no basado en la póliza viene dado por:

$$\delta_t = r_{t+1} + \gamma \mathcal{Q}_t(kNN_{t+1}, u_{t+1}) - \mathcal{Q}_t(kNN_t, u_t) \tag{3.29}$$

$$\delta_t = r_{t+1} + \gamma \min_u \mathcal{Q}_t(kNN_{t+1}, u) - \mathcal{Q}_t(knn_t, u_t). \tag{3.30}$$

Entonces la regla de actualización para el valor esperado del predictor de la función de valor es:

$$Q_{t+1}(i, u_t) = Q_t(i, u_t) + \alpha_t \delta_t p(i) \quad \forall i \in kNN_t. \tag{3.31}$$

El algoritmo de kNN-TD viene dado en el Algoritmo 3.1.

Algoritmo 3.1 Aprendizaje kNN-TD

Entrada: γ, α_t,
Inicializar el espacio de los clasificadores cl y $Q_0(cl, u)$ arbitrariamente
repetir {por cada episodio}
  Inicializar x_0
  $kNN_0 \leftarrow k$-vecinos cercanos de x_0
  $p(kNN_0) \leftarrow$ probabilidades de cada $cl \in kNN$
  $\mathcal{Q}_0(kNN_0, u) = Q_0(kNN_0, u) \cdot p(kNN_0)$ para todo u
  Elegir u_0 de x_0 de acuerdo a $\mathcal{Q}_0(kNN_0, u)$
  repetir {para cada paso de episodio $t = 0, 1, ...$}
    Tomar acción u_t y observar r_{t+1}, x_{t+1}
    $kNN_{t+1} \leftarrow k$-vecinos cercanos de x_{t+1}
    $p(kNN_{t+1}) \leftarrow$ probabilidades de cada $cl \in kNN_{t+1}$
    $\mathcal{Q}_t(kNN_{t+1}, u) = Q_t(kNN_{t+1}, u) \cdot p(kNN_{t+1})$ para todo u
    Elegir u_{t+1} de x_{t+1} de acuerdo a $\mathcal{Q}_t(kNN_{t+1}, u)$
    Actualizar Q_t usando (3.29) o (3.30) y (3.31)
    $x_t \leftarrow x_{t+1}$, $kNN_t \leftarrow kNN_{t+1}$, $u_t \leftarrow u_{t+1}$
  hasta x_t es terminal
hasta aprendizaje termina

Para espacio de acciones largas la selección de acción (3.27) y el cálculo del valor de la mejor acción (3.26) y (3.28) son modificadas. La selección de acción es independiente del valor $\mathcal{Q}(kNN', u)$. La selección de acción guarda las acciones recomendadas de cada

clasificador del conjunto kNN en una lista de acciones óptimas U^*. Entonces dado una lista de acciones $\mathcal{U}$, las mejores acciones son elegidas usando el siguiente índice:

$$I = \underset{u}{\operatorname{argmin}}\, Q(i,u)\ \forall i \in kNN \text{ y } \forall u \in \mathcal{U} \tag{3.32}$$

este índice contiene los índices de los valores de $\mathcal{U}$ donde los valores de $Q(i,u)$ son mínimos. La lista de acciones óptimas U^* es obtenida mediante:

$$U^* = \mathcal{U}[I]$$

La acción óptima esperada es calculada usando (3.13):

$$\langle u \rangle = \sum_{i=1}^{kNN} U^*(i)p(i)\ \ \forall i \in kNN \tag{3.33}$$

donde $p(i)$ es la probabilidad condicional $\Pr\{u = U^*(i)|x\}$ que u tome el valor de $U^*(i)$ dado el estado x. El error TD es calculado de la siguiente forma:

$$\delta_t = r_{t+1} + \gamma \mathcal{Q}_t(kNN_{t+1}, I_{t+1}) - \mathcal{Q}_t(kNN_t, I_t) \tag{3.34}$$

y la acción de valor $\mathcal{Q}(kNN, I)$ es calculado por

$$\langle \mathcal{Q}(kNN, I) \rangle = \sum_{i=1}^{kNN} \min_{\mathfrak{L}} Q(i, \mathfrak{L})p(i) \tag{3.35}$$

$$\langle \mathcal{Q}(kNN', I') \rangle = \sum_{i=1}^{kNN'} \min_{\mathfrak{L}} Q(i, \mathfrak{L})p(i) \tag{3.36}$$

donde $\mathfrak{L}$ es la lista $1...n = |\,\mathcal{U}|$. Se observa que este algoritmo usa dos estimadores simples para la actualización de la función Q: $\mathcal{Q}(kNN, I)$ y $\mathcal{Q}(kNN', I')$ que utiliza el operador min para estimar el valor esperado mínimo de $\mathcal{Q}$. El algoritmo kNN-TD para espacios de acción grandes o $\mathrm{Ex}\langle a \rangle$ [75] esta dado en el Algoritmo 3.2.

3.2.5. El algoritmo kNN-TD modificado

El aprendizaje kNN-TD modificado (LS-DA), como se muestra en el Algoritmo 3.3 utiliza el método del doble estimador para estimar el valor esperado mínimo de la función $\mathcal{Q}$ del siguiente estado, $\min_u E\{\mathcal{Q}(kNN', u)\}$, en un aprendizaje no basado en la póliza.

Algoritmo 3.2 Aprendizaje kNN-TD para espacio de acciones grandes

Entrada: γ, α_t,
Inicializar $\mathcal{U}$ y $\mathfrak{L}$
Inicializar el espacio de los clasificadores cl y $Q_0(cl, u)$
repetir {por cada episodio}
  Inicializar x_0
  $kNN_0 \leftarrow k$-vecinos cercanos de x_0
  $p(kNN_0) \leftarrow$ probabilidades de cada $cl \in kNN$
  $I_0 \leftarrow \text{argmin}_{\mathfrak{L}}\, Q_0(kNN_0, \mathfrak{L})$
  $u_0 \leftarrow \mathcal{U}[I] \cdot p(kNN_0)$
  $\mathcal{Q}_0(kNN_0, I) = \text{mín}_{\mathfrak{L}}\, Q_0(kNN_0, \mathfrak{L}) \cdot p(kNN_0)$
  repetir {para cada paso de episodio $t = 0, 1, ...$}
    Tomar acción u_t y observar r_{t+1}, x_{t+1}
    $kNN_{t+1} \leftarrow k$-vecinos cercanos de x_{t+1}
    $p(kNN_{t+1}) \leftarrow$ probabilidades de cada $cl \in kNN_{t+1}$
    $I_{t+1} \leftarrow \text{argmin}_{\mathfrak{L}}\, Q_t(kNN_{t+1}, \mathfrak{L})$
    $u_{t+1} \leftarrow \mathcal{U}[I_{t+1}] \cdot p(kNN_{t+1})$
    $\mathcal{Q}_t(kNN_{t+1}, I_{t+1}) = \text{mín}_{\mathfrak{L}}\, Q_t(kNN_{t+1}, \mathfrak{L}) \cdot p(kNN_{t+1})$
    Actualizar Q_t usando (3.34) y (3.31)
    $x_t \leftarrow x_{t+1}$, $kNN_t \leftarrow kNN_{t+1}$, $u_t \leftarrow u_{t+1}$,$I_t \leftarrow I_{t+1}$
  hasta x_t es terminal
hasta aprendizaje termina

Se almacenan dos funciones, Q^A y Q^B, y se utiliza dos subconjuntos separados de experiencias para aprenderlas. Esto sirve para desacoplar los dos estimadores tendiendo a reducir la susceptibilidad a variaciones aleatorias en r_{t+1} y estabiliza las acciones de valor. Adicionalmente, Q^A y Q^B son conmutados con probabilidad 0.5, esto significa que cada estimador es actualizado únicamente usando la mitad de experiencias y sólo existe un incremento marginal en el costo computacional al tener dos estimadores. Si se actualiza el valor Q^A, entonces la acción de control esta dada por:

$$u_t^* = \underset{u}{\text{argmin}}\, Q_t^A(kNN_{t+1}, u).$$

Por el otro lado, si se actualiza Q^B entonces:

$$v_t^* = \underset{u}{\text{argmin}}\, Q_t^B(kNN_{t+1}, u)$$

Las reglas de actualización para cada el estimador A y B están dadas en las siguientes ecuaciones:

$$\delta_t^{BA} = r_{t+1} + \gamma \mathcal{Q}_t^B(kNN', u_t^*) - \mathcal{Q}_t^A(kNN, u_t) \tag{3.37}$$

$$\delta_t^{AB} = r_{t+1} + \gamma \mathcal{Q}_t^A(kNN', v_t^*) - \mathcal{Q}_t^B(kNN, u_t) \tag{3.38}$$

$$Q_{t+1}^A(i, u_t) = Q_t^A(i, u_t). + \alpha_t \delta_t^{BA} p(i) \quad \forall i \in kNN_t \tag{3.39}$$

$$Q_{t+1}^B(i, u_t) = Q_t^B(i, u_t). + \alpha_t \delta_t^{AB} p(i) \quad \forall i \in kNN_t \tag{3.40}$$

Algoritmo 3.3 Aprendizaje kNN-TD modificado (LS-DA)

Entrada: γ, α_t,
Inicializar el espacio de clasificadores cl y $Q_0^A(cl, u)$ y $Q_0^B(cl, u)$ arbitrariamente
repetir {por cada episodio}
  Inicializar x_0
  $kNN_0 \leftarrow k$-vecinos cercanos de x_0
  $p(kNN_0) \leftarrow$ probabilidades de cada $cl \in kNN_0$
  $\mathcal{Q}_0^A(kNN_0, u) = Q_0^A(kNN_0, u) \cdot p(kNN_0)$ para todo u
  $\mathcal{Q}_0^B(kNN_0, u) = Q_0^B(kNN_0, u) \cdot p(kNN_0)$ para todo u
  Elegir u_0 de x_0 de acuerdo a $\mathcal{Q}_0^A(kNN_0, u)$ y $\mathcal{Q}_0^B(kNN_0, u)$
  repetir {para cada paso de episodio $t = 0, 1, ...$}
    Tomar acción u_t y observar r_{t+1}, x_{t+1}
    $kNN_{t+1} \leftarrow k$-vecinos cercanos de x_{t+1}
    $p(kNN_{t+1}) \leftarrow$ probabilidades de cada $cl \in kNN_{t+1}$
    $\mathcal{Q}_t^A(kNN_{t+1}, u) = Q_t^A(kNN_{t+1}, u) \cdot p(kNN_{t+1})$ para todo u
    $\mathcal{Q}_t^B(kNN_{t+1}, u) = Q_t^B(kNN_{t+1}, u) \cdot p(kNN_{t+1})$ para todo u
    Elegir (e.g.) aleatoriamente actualizar ya sea Q_t^A o Q_t^B
    si Actualizar Q_t^A **entonces**
      Definir $u_t^* = \text{argmin}_u \mathcal{Q}_t^A(kNN_{t+1}, u)$
      Actualizar Q_t^A usando (3.37) y (3.39)
    else si Actualizar Q_t^B **entonces**
      Definir $v_t^* = \text{argmin}_u \mathcal{Q}_t^B(kNN_{t+1}, u)$
      Actualizar Q_t^B usando (3.38) y (3.40)
    fin si
    $x_t \leftarrow x_{t+1}$, $kNN_t \leftarrow kNN_{t+1}$, $u_t \leftarrow u_{t+1}$
  hasta x_t es terminal
hasta aprendizaje termina

El siguiente lema proporciona las herramientas para probar la convergencia del algoritmo de aprendizaje kNN-TD modificado.

Lema 4. *Se considera un proceso estocástico* $(\zeta_t, \Delta_t, F_t), t \geq 0$, *donde* $\zeta_t, \Delta_t, F_t : X \to \mathbb{R}$

satisface las ecuaciones:

$$\Delta_{t+1}(z_t) = (1 - \zeta_t(z_t))\Delta_t(z_t) + \zeta_t(z_t)F_t(z_t) \tag{3.41}$$

donde $z_t \in Z$. *Sea* P_t *una secuencia de campos* σ *crecientes tal que* ζ_0 *y* Δ_0 *son* P_0*-medibles y* ζ_t, Δ_t *y* F_{t-1} *son* P_t*-medibles,* $t = 1, 2, ...$ *Se asume que lo siguiente se satisface: 1)*$\zeta_t(z_t) \in (0, 1], \sum_t \zeta_t(z_t) = \infty$, $\sum_t \zeta_t^2(z_t) < \infty$ *con probabilidad uno (w.p.1) y* $\forall z \neq z_t : \zeta_t(z) = 0$. *2)* $\|E\{F_t|P_t\}\| \leq \kappa\|\Delta_t\| + c_t$, *donde* $\kappa \in [0, 1)$ *y* c_t *converge a cero con probabilidad uno (w.p.1) 3)*$\text{VAR}\{F_t(z_t)|P_t\} \leq K(1 + \kappa\|\Delta_t\|)^2$, *donde* K *es alguna constante positiva. La norma* $\|\cdot\|$ *denota la norma infinito. Entonces* Δ_t *converge a cero con probabilidad uno (w.p.1) [58].*

El lema anterior es similar al lema presentado en [17], sin embargo la principal diferencia es que por medio de la regla kNN el conjunto X puede ser grande. La prueba de convergencia del aprendizaje kNN-TD modificado esta dado por el siguiente teorema.

Teorema 6. *Se considera que las siguientes condiciones son satisfechas: 1)* $\gamma \in [0, 1)$, *2)* $\alpha_t(x, u) \in (0, 1]$, $\sum_t \alpha_t(x, u) = \infty$, $\sum_t (\alpha_t(x, u))^2 < \infty$ *con probabilidad uno (w.p.1) y* $\forall (x_t, u_t) \neq (x, u) : \alpha(x, u) = 0$, *3)* $\text{VAR}\{R_{xu}^{x'}\} < \infty$. *Entonces dado un MDP ergódico, tanto* Q^A *y* Q^B *actualizados como se muestra en el Algoritmo 3.3 convergerán en el límite a la función de valor casi-óptima* Q^* *como se indica en la ecuación de optimalidad de Bellman (3.12) con probabilidad uno (w.p.1) si la póliza seguida asegura que cada para de estado-acción es visitado un número infinito de veces.*

Demostración. Las reglas de actualización del aprendizaje LS-DA son simétricas, por lo tanto es suficiente mostrar la convergencia de una de ellas. Si se utiliza la probabilidad condicional $\Pr(\mathcal{Q}_t^A(kNN, u) = Q_t^A(i, u)|x_t)$ y $\Pr(\mathcal{Q}_t^B(kNN, u) = Q_t^B(i, u)|x_t)$ y aplicando el Lema 3, se obtiene que $\mathcal{Q}^A(kNN, u) = Q^A(x, u)$ y $\mathcal{Q}^B(kNN, u) = Q^B(x, u)$. Si se aplica el Lema 4 con $P_t = \{Q_0^A, Q_0^B, x_0, u_0, \alpha_0, r_1, x_1, ..., x_t, u_t\}$, $Z = X \times U$, $\Delta_t = Q_t^A - Q_t^*$, $\zeta = \alpha$,$F_t(x_t, u_t) = r_{t+1} + \gamma Q_t^B(x_{t+1}, u_t^*) - Q_t^*(x_t, u_t)$, donde $u_t^* = \text{argmin}_u Q^A(x_{t+1}, u)$ y $\kappa = \gamma$. La condición 1) y 3) se satisfacen a consecuencia de las condiciones de este Teorema. Por lo tanto, solo se requiere probar la condición 2) del Lema 4. Se puede escribir $F_t(x_t, u_t)$ como:

$$F_t(x_t, u_t) = G_t(x_t, u_t) + \gamma \left(Q_t^B(x_{t+1}, u_t^*) - Q_t^A(x_{t+1}, u_t^*)\right)$$

donde $G_t = r_{t+1} + \gamma Q_t^A(x_{t+1}, u_t^*) - Q_t^*(x_t, u_t)$ es el valor de F_t si se considera el algoritmo de Q-learning (1.26). Se sabe que $E\{G_t|P_t\} \leq \gamma\|\Delta_t\|$, entonces se tiene que

$c_t = \gamma\left(Q_t^B(x_{t+1}, u_t^*) - Q_t^A(x_{t+1}, u_t^*)\right)$, y es suficiente probar que $\Delta_t^{BA} = Q_t^B - Q_t^A$ converge a cero. La actualización de Δ_t^{BA} en el tiempo t es:

$$\Delta_{t+1}^{BA}(x_t, u_t) = \Delta_t^{BA}(x_t, u_t) + \alpha_t(x_t, u_t)F_t^B(x_t, u_t) - \alpha_t(x_t, u_t)F_t^A(x_t, u_t)$$

donde $F_t^A(x_t, u_t) = r_{t+1} + \gamma Q_t^B(x_{t+1}, u_t^*) - Q_t^A(x_t, u_t)$ y $F_t^B(x_t, u_t) = r_{t+1} + \gamma Q_t^A(x_{t+1}, v_t^*) - Q_t^B(x_t, u_t)$. Entonces

$$\begin{aligned} E\{\Delta_{t+1}^{BA}(x_t, u_t)|P_t\} =& \Delta_t^{BA}(x_t, u_t) + E\{\alpha_t F_t^B(x_t, u_t) - \alpha_t F_t^A(x_t, u_t)|P_t\} \\ =& (1 - \alpha_t)\Delta_t^{BA}(x_t, u_t) + \alpha_t E\{F_t^{BA}(x_t, u_t)|P_t\} \end{aligned}$$

donde $E\{F_t^{BA}(x_t, u_t)|P_t\} = \gamma E\{Q_t^A(x_{t+1}, v_t^*) - Q_t^B(x_{t+1}, u_t^*)|P_t\}$. Debido a que la actualización de la función de valor(Q_t^A o Q_t^B) es aleatoria, entonces se tiene los siguientes casos:

1. Se asume que $E\{Q_t^A(x_{t+1}, v_t^*)|P_t\} \leq E\{Q_t^B(x_{t+1}, u_t^*)|P_t\}$. Luego por la definición de la selección de la acción v_t^* del Algoritmo 3.3, se tiene que la siguiente desigualdad es satisfecha: $Q_t^A(x_{t+1}, v_t^*) = \min_u Q_t^A(x_{t+1}, u) \leq Q_t^A(x_{t+1}, u_t^*)$, por lo tanto

$$\begin{aligned} E\{F_t^{BA}|P_t\} &= \gamma E\{Q_t^A(x_{t+1}, v_t^*) - Q_t^B(x_{t+1}, u_t^*)|P_t\} \\ &\leq \gamma E\{Q_t^A(x_{t+1}, u_t^*) - Q_t^B(x_{t+1}, u_t^*)|P_t\} \\ &\leq \gamma\|\Delta_t^{BA}\|. \end{aligned}$$

2. Se asume que $E\{Q_t^B(x_{t+1}, u_t^*)|P_t\} \leq E\{Q_t^A(x_{t+1}, v_t^*)|P_t\}$. Luego por la definición de la selección de la acción u_t^* del Algoritmo 3.3, se tiene que la siguiente desigualdad es satisfecha: $Q_t^B(x_{t+1}, u_t^*) = \min_u Q_t^B(x_{t+1}, u) \leq Q_t^B(x_{t+1}, v_t^*)$, por lo tanto

$$\begin{aligned} E\{F_t^{BA}|P_t\} &= \gamma E\{Q_t^A(x_{t+1}, v_t^*) - Q_t^B(x_{t+1}, u_t^*)|P_t\} \\ &\leq \gamma E\{Q_t^A(x_{t+1}, v_t^*) - Q_t^B(x_{t+1}, v_t^*)|P_t\} \\ &\leq \gamma\|\Delta_t^{BA}\|. \end{aligned}$$

Uno de los casos presentados debe cumplirse en cada paso de tiempo y en ambos casos se obtiene el mismo resultado. Aplicando el Lema 4 se obtiene la convergencia de Δ_t^{BA} a cero, y por lo tanto Δ_t converge también a cero y Q^A y Q^B convergen a Q^*. □

Para el aprendizaje kNN-TD con espacio de acciones grandes (LS-LA) existen dos estimadores simples para la predicción de la función Q: $\mathcal{Q}(kNN, I)$ y $\mathcal{Q}(kNN', I')$; aquí el Algoritmo 3.2 tiene dos estimadores simples en la misma regla de actualización, en el sentido de que cada estimador da un estimado de la función de valor para los conjuntos kNN y kNN', respectivamente, más aún estos estimadores son independientes. En el Algoritmo 3.3, sólo existe un estimador simple entonces el doble estimador puede ser aplicado directamente, sin embargo no es el caso para el Algoritmo 3.2. Aquí, estos dos estimadores son vistos como las dos funciones de valor $\mathcal{Q}^A$ y $\mathcal{Q}^B$, cada una almacena los valores de la función de valor para un cierto conjunto kNN, como se muestra a continuación:

$$\mathcal{Q}^A(kNN, I) = Q(kNN, u)p(kNN) \tag{3.42}$$

$$\mathcal{Q}^B(kNN', I') = Q(kNN', u)p(kNN') \tag{3.43}$$

A diferencia del Algoritmo 3.3, este algoritmo no requiere que las actualizaciones sean conmutadas con probabilidad 0.5, en cambio, se aprovecha la ventaja de que los predictores se encuentren en la misma regla, y se obtiene un aprendizaje doble serial. Las acciones minimizantes de los estimadores son:

$$u_t^A = \langle u^A \rangle = U^*(kNN)p(kNN) \tag{3.44}$$

$$u^B = \langle u^B \rangle = U^*(kNN')p(kNN') \tag{3.45}$$

Las acciones minimizantes son obtenidas mediante las mejores acciones recomendadas U^* dados los k-vecinos cercanos de los conjuntos kNN y kNN'. Las acciones minimizantes son utilizadas para estimar el valor mínimo de los estimadores. El error TD y la regla de actualización vienen dados por:

$$\delta_t = r_{t+1} + \gamma \mathcal{Q}_t^B(kNN', u_t^B) - \mathcal{Q}_t^A(kNN, u_t^A) \tag{3.46}$$

$$Q_{t+1}(i, u_t) = Q_t(i, u_t) + \alpha_t \delta_t p(i) \quad \forall i \in kNN_t \tag{3.47}$$

Aquí aparentemente los estimadores $\mathcal{Q}^A$ y $\mathcal{Q}^B$ son completamente independientes, sin embargo en cada paso final el nuevo valor de $\mathcal{Q}^A$ es $\mathcal{Q}^B$, i.e., $\mathcal{Q}^A \leftarrow \mathcal{Q}^B$, y el algoritmo continua iterando hasta que x es terminal. En el Algoritmo 3.4 se presenta el pseudo-algoritmo del aprendizaje kNN-TD modificado para espacio de acciones grandes.

Teorema 7. *Se considera que las condiciones del Teorema 6 son satisfechas. Entonces, dado un MDP ergódico, la función Q actualizada como se muestra en el Algoritmo 3.4*

converge en el límite a una función de valor casi-óptima Q^* como se indica en la ecuación de optimalidad de Bellman (3.12) con probabilidad uno (w.p.1) si la póliza seguida asegura que cada para de estado-acción es visitado un número infinito de veces.

Algoritmo 3.4 Aprendizaje kNN-TD modificado para espacio de acciones grandes (LS-LA)

Entrada: γ, α_t,
Inicializar $\mathcal{U}$ y $\mathfrak{L}$
Inicializar el espacio de clasificadores cl y $Q_0(cl, \mathfrak{L})$ arbitrariamente
repetir {para cada episodio}
  Inicializar x_0
  $kNN_0 \leftarrow k$-vecinos cercanos de x_0
  $p(kNN_0) \leftarrow$ probabilidades de cada $cl \in kNN_0$
  $I_0 \leftarrow \text{argmin}_{\mathfrak{L}}\, Q_0(kNN_0, \mathfrak{L})$
  $u_0 \leftarrow \mathcal{U}[I] \cdot p(kNN_0)$
  $\mathcal{Q}_0^A(kNN_0, u) = Q_0(kNN_0, u) \cdot p(kNN_0)\ \forall u$
  repetir {para cada paso de episodio $t = 0, 1, ...$}
    Tomar acción u_t y observar r_{t+1}, x_{t+1}
    $kNN_{t+1} \leftarrow k$-vecinos cercanos de x_{t+1}
    $p(kNN_{t+1}) \leftarrow$ probabilidades de cada $cl \in kNN_{t+1}$
    $I_{t+1} \leftarrow \text{argmin}_{\mathfrak{L}}\, Q_t(kNN_{t+1}, \mathfrak{L})$
    $u_{t+1} \leftarrow \mathcal{U}[I_{t+1}] \cdot p(kNN_{t+1})$
    $\mathcal{Q}_t^B(kNN_{t+1}, u) = Q_t(kNN_{t+1}, u) \cdot p(kNN_{t+1})\ \forall u$
    Obtener las acciones de control mediante (3.44) y (3.45)
    Actualizar Q_t usando (3.46) y (3.47)
    $x_t \leftarrow x_{t+1}$, $kNN_t \leftarrow kNN_{t+1}$, $u_t \leftarrow u_{t+1}$,$I_t \leftarrow I_{t+1}$, $\mathcal{Q}_t^A \leftarrow \mathcal{Q}_t^B$
  hasta x_t es terminal
hasta aprendizaje termina

Demostración. Ahora tanto el espacio de estado y acción son grandes, se asume que ambos espacios son independientes e idénticamente distribuidos (i.i.d.). Utilizando la probabilidad condicional $\Pr(\mathcal{Q}_t^A(kNN, u) = Q_t(i, u)|x_t)$ y $\Pr(\mathcal{Q}_t^B(kNN_{t+1}, u) = Q_t(i, u)|x_{t+1})$ y aplicando el Lema 3, se obtiene que $\mathcal{Q}_t^A(kNN, u) = Q_t(x_t, u)$ y $\mathcal{Q}_t^B(kNN_{t+1}, u) = Q_t(x_{t+1}, u)$ para todo u. La acción esperada óptima u_t es obtenida usando la lista de acciones óptima U^* y las probabilidades $p(i)$ que están en función de los k-vecinos cercanos del conjunto kNN, entonces la acción esperada óptima $\langle u_t \rangle = U^*(x_t)$ debido al Lema 3.

$Z = X \times U$, $\Delta_t = Q_t - Q_t^*$, $\zeta = \alpha$,$F_t(x_t, u_t) = r_{t+1} + \gamma Q_t(x_{t+1}, u_t^B) - Q_t^*(x_t, u_t)$, donde u_t^B esta dada por (3.45) y $\kappa = \gamma$. La condición 1) y 3) son satisfechas en consecuencia de

las condiciones del Teorema. Por lo tanto, sólo se requiere probar la condición 2) del Lema 4. Se puede escribir a $F_t(x_t, u_t)$ como:

$$F_t = G_t(x_t, u_t) + \gamma \left(Q_t(x_{t+1}, u_t^B) - \min_{u \in \mathcal{L}} Q_t(x_{t+1}, u) \right)$$

donde $G_t(x_t, u_t) = r_{t+1} + \gamma \min_{u \in \mathcal{L}} Q(x_{t+1}, u) - Q_t^*(x_t, u_t)$. Se sabe que $E\{G_t | P_t\} \leq \gamma \|\Delta_t\|$ [76], entonces se tiene que $c_t = \gamma \left(Q_t(x_{t+1}, u_t^B) - \min_{u \in \mathcal{L}} Q_t(x_{t+1}, u) \right)$. La convergencia a cero con probabilidad uno (w.p.1) de c_t se obtiene a partir del mecanismo de selección de acciones del algoritmo, i.e., la acción óptima es obtenida usando la mejor acción recomendada dado un k-vecino cercano, esto es similar a las pólizas "codiciosas en el límite con exploración infinita" (GLIE por sus siglas en inglés), donde las acciones no-codiciosas tienen probabilidades que van desapareciendo. Se supone que $Q_2(x_{t+1}, a) = \min_u Q_t(x_{t+1}, u)$ donde a es la acción que minimiza Q_t. La actualización para $\Delta_t^a = Q_t - Q_2$ es:

$$\Delta_{t+1}^a = (1 - \alpha_t)\Delta_t^a + \alpha_t c_t$$

Se consideran los siguientes casos:

1. La siguiente desigualdad es satisfecha $E\{Q_t(x_{t+1}, u_t^B) | P_t\} \leq E\{Q_2(x_{t+1}, a) | P_t\}$, por lo tanto

$$\begin{aligned} E\{c_t | P_t\} &= \gamma E\{Q_t(x_{t+1}, u_t^B) - Q_2(x_{t+1}, a) | P_t\} \\ &\leq \gamma E\{Q_t(x_{t+1}, a) - Q_2(x_{t+1}, a) | P_t\} \leq \gamma \|\Delta_t^a\| \end{aligned}$$

2. La siguiente desigualdad es satisfecha: $E\{Q_2(x_{t+1}, a) | P_t\} \leq E\{Q_t(x_{t+1}, u_t^B) | P_t\}$, por lo tanto:

$$\begin{aligned} E\{c_t | P_t\} &= \gamma E\{Q_t(x_{t+1}, u_t^B) - Q_2(x_{t+1}, a) | P_t\} \\ &\leq \gamma E\{Q_t(x_{t+1}, u_t^B) - Q_2(x_{t+1}, u_t^B) | P_t\} \\ &\leq \gamma \|\Delta_t^a\| \end{aligned}$$

En ambos casos la desigualdad final se mantiene con el mismo resultado deseado. Entonces si se aplica el Lema 4 se obtiene la convergencia de Δ_t^a a cero y por lo tanto Δ_t converge también a cero y Q converge a Q^*. □

Este algoritmo requiere suficiente exploración para seleccionar las acciones recomen-

dadas y encontrar la acción esperada óptima.

El problema de control robusto puede ser resuelto por cualquier método de aprendizaje por refuerzo siempre y cuando se diseñe una recompensa adecuada. La gran diferencia radica en la robustez del método a utilizar y la posible sobreestimación de las acciones debido a la naturaleza discreta de ciertos algoritmos.

3.3. Control Robusto en Tiempo continuo

Se considera el siguiente sistema no lineal:

$$\dot{x}_t = f(x_t) + g_1(x_t)u_t + g_2(x_t)\omega_t, \quad x_{t_0} = x_0, \ t \geq t_0 \tag{3.48}$$

donde $f(x_t) \in \mathbb{R}^n$, $g_1(x_t) \in \mathbb{R}^{n\times m}$, $g_2(x_t) \in \mathbb{R}^{n\times\omega}$ definen la dinámica del sistema no lineal, $x_t \in X \subset \mathbb{R}^n$ es el estado, $u_t \in U \subset \mathbb{R}^m$ es el control y $\omega \in \mathbb{R}^\omega$ es una perturbación. Cuando $\omega = 0$, el problema de control $\mathcal{H}_2$ busca una póliza de control admisible $u_t = h(x_t)$ que minimice la siguiente función de costo:

$$J_2(x_t, u_t) = V(x_t) = \int_t^\infty (x_\tau^\top S x_\tau + u_\tau^\top R u_\tau) e^{-\gamma(\tau - t)} d\tau. \tag{3.49}$$

La función de costo anterior es equivalente a la función de costo (N2) para sistemas en tiempo continuo con $S \in \mathbb{R}^{n\times n}$ y $R \in \mathbb{R}^{m\times m}$ las matrices de peso del estado y control, respectivamente. Cuando se deriva respecto al tiempo se obtiene la expresión (N3), que se vuelve a escribir para la facilidad del lector:

$$\dot{V}(x_t) = \int_t^\infty \frac{\partial}{\partial t} \left[x_\tau^\top S x_\tau + u_\tau^\top R u_\tau \right] e^{-\gamma(\tau - t)} d\tau - x_t^\top S x_t - u_t^\top R u_t.$$

Mediante el uso de la ecuación de Hamilton-Jacobi-Bellman (ver Tabla 2.1) y del hecho que la derivada de la ecuación de Bellman satisface:

$$\dot{V}(x_t) = \frac{\partial V(x_t)}{\partial x_t} \dot{x}_t \tag{3.50}$$

Entonces si se sustituye (3.50) en la derivada de la función de costo se obtiene:

$$\frac{\partial V^*(x_t)}{\partial x_t} \left(f(x_t) + g_1(x_t) u_t^* \right) = \gamma V(x_t) - x_t^\top S x_t - u_t^{*\top} R u_t^*. \tag{3.51}$$

El control óptimo se obtiene al diferenciar (3.51) respecto a u_t:

$$u_t^* = -\frac{R^{-1}}{2} g_1^\top(x_t) \frac{\partial V^{*\top}(x_t)}{\partial x_t} \tag{3.52}$$

La expresión anterior es la versión general de la solución del problema de control óptimo $\mathcal{H}_2$ o LQR para sistemas no lineales.

Si $\omega \neq 0$, el problema de control $\mathcal{H}_\infty$ busca una ley de control admisible $u_t = h(x_t)$ que minimice la siguiente función de costo:

$$J_\infty(x_t, u_t, \omega_t) = V(x_t) = \int_t^\infty (x_\tau^\top S x_\tau + u_\tau^\top R u_\tau - \eta^2 \omega_\tau^\top \omega_\tau) e^{-\gamma(\tau - t)} d\tau. \tag{3.53}$$

donde η es un factor de atenuación. Derivando (3.53) respecto al tiempo se obtiene:

$$\dot{V}(x_t) = \int_t^\infty \frac{\partial}{\partial t} \left[x_\tau^\top S x_\tau + u_\tau^\top R u_\tau - \eta^2 \omega_\tau^\top \omega_\tau \right] e^{-\gamma(\tau - t)} d\tau - x_t^\top S x_t - u_t^\top R u_t + \eta^2 \omega_t^\top \omega_t.$$

La función de valor óptima puede ser obtenida mediante la resolución de un juego de suma-cero:

$$V^*(x_t) = \min_u \max_\omega J_\infty(x_t, u, \omega),$$

cuya solución viene dada por la ecuación de Hamilton-Jacobi-Bellman-Isaacs (HJBI):

$$\frac{\partial V^*(x_t)}{\partial x_t} \left(f(x_t) + g_1(x_t) u_t^* + g_2(x_t) \omega_t^* \right) = \gamma V^*(x_t) - x_t^\top S x_t - u_t^\top R u_t + \eta^2 \omega_t^\top \omega_t \tag{3.54}$$

El control óptimo ya la peor perturbación vienen dadas por las siguientes expresiones:

$$u_t^* = -\frac{R^{-1}}{2} g_1^\top(x_t) \frac{\partial V^{*\top}(x_t)}{\partial x_t} \tag{3.55}$$

$$\omega_t^* = \frac{1}{2\eta^2} g_2^\top(x_t) \frac{\partial V^{*\top}(x_t)}{\partial x_t} \tag{3.56}$$

Las expresiones anteriores son el control óptimo y peor perturbación del problema de control robusto $\mathcal{H}_\infty$ para sistemas no lineales. De manera análoga, el control híbrido $\mathcal{H}_2/\mathcal{H}_\infty$ en tiempo continuo tiene la misma forma que en la expresión (3.10). Las soluciones del problema $\mathcal{H}_2$ (3.52) y del problema $\mathcal{H}_\infty$ (3.55) requieren conocimiento de la dinámica del sistema.

3.4. Aprendizaje Robusto: Tiempo continuo

El aprendizaje por refuerzo en tiempo continuo es una forma más natural de resolver el problema de control robusto. En este caso los métodos vistos en el capítulo anterior son válidos para resolver el problema, donde la única modificación es el diseño de la recompensa que debe satisfacer (3.12).

Con fines de facilitar el texto, el algoritmo de Q-learning en tiempo continuo es el siguiente:

$$\dot{\theta}(t) = \alpha(t)\frac{\delta(t)}{T}\frac{\partial \widehat{Q}(x(t), u(t); \theta(t))}{\partial \theta(t)} \tag{3.57}$$

$$\delta(t) = r(t+T) + \frac{1}{T}\left[(1-\gamma T)\widehat{Q}(x(t+T), u(t+T); \theta(t)) - \widehat{Q}(x(t), u(t); \theta(t))\right] \tag{3.58}$$

donde $\widehat{Q}(x, u; \theta) = \Phi^T(x, u)\theta$ es la aproximación de la función de valor. Este algoritmo es también conocido como algoritmo crítico debido a que "critica" el desempeño de la póliza seguida mediante la ecuación de optimalidad de Bellman.

Como se mencionó en el Capítulo 1, existe una tercer clase de algoritmos de aprendizaje por refuerzo llamados algoritmos de búsqueda de la póliza. Esta clase son ampliamente utilizados debido a su versatilidad y habilidad de lidiar con espacios de estado-acción grandes al actualizar la función de valor y póliza por separado. Dentro de esta clase de algoritmos se encuentran los algoritmos Actor-Críticos, los cuales serán explicados a continuación.

3.4.1. Algoritmos Actor-Críticos

Los algoritmos Actor-Críticos son una subclase de los algoritmos de búsqueda de la póliza donde se obtiene la póliza óptima utilizando dos agentes: un crítico y un actor. El funcionamiento del actor consiste en evaluar la póliza en el sistema/ambiente y es actualizada mediante el error TD proveniente del crítico, es decir, la función de valor y la póliza son actualizados por separado pero cada una depende de la otra.

El crítico utiliza una función de valor de estado $V(x(t))$ (2.18) en lugar de una función de estado-acción $Q(x(t), u(t))$ (2.24). De forma similar al capítulo anterior, el crítico puede ser expresado utilizando el método de diferenciación de Euler hacia atrás y una parametrización de la forma $\widehat{V}(x; \theta) = \Phi^T(x)\theta$. El crítico viene dado por las siguientes expresiones:

$$\dot{\theta}(t) = \alpha_c(t)\frac{\delta(t)}{T}\frac{\partial \widehat{V}(x(t);\theta(t))}{\partial \theta(t)} \tag{3.59}$$

$$\delta(t) = r(t+T) + \frac{1}{T}\left[(1-\gamma T)\widehat{V}(x(t+T);\theta(t)) - \widehat{V}(x(t);\theta(t))\right], \tag{3.60}$$

donde $\alpha_c(t)$ es el parámetro de aprendizaje del crítico. Se observa claramente que la actualización del critico es más sencilla debido a que no depende del espacio de acciones.

El actor de forma similar al crítico requiere una parametrización de la forma $\widehat{h}(x;\vartheta) = \Psi^T(x)\vartheta$, donde $\Psi(x)$ es el vector de BFs, ϑ es el vector de parámetros de la póliza. La regla de actualización del actor viene dado por la siguiente expresión:

$$\dot{\vartheta}(t) = \alpha_a(t)\Delta u(t)\frac{\delta(t)}{T}\frac{\partial \widehat{h}(x(t);\vartheta(t))}{\partial \vartheta(t)} \tag{3.61}$$

donde $\alpha_a(t)$ es el parámetro de aprendizaje del actor, $\Delta u(t) \backsim \mathcal{N}(0,\sigma^2)$ es un término de ruido aleatorio de exploración. El producto entre el término de exploración $\Delta u(t)$ y el error TD $\delta(t)$ sirve como un signo de conmutación para el gradiente de la póliza $\partial\widehat{h}(x;\vartheta)/\partial\vartheta$. Cuando la exploración $\Delta u(t)$ arroja un error TD positivo, la dirección de exploración es benéfico al desempeño y la póliza se ajusta hacia la acción perturbada. Por el otro lado, cuando el error TD es negativo, la póliza se ajusta lejos de esta perturbación. En el Algoritmo 3.5 se muestra el algoritmo Actor-Crítico para sistemas en tiempo continuo.

Algoritmo 3.5 Aprendizaje Actor-Crítico

1: **Entrada:** γ, $\alpha_a(t)$, $\alpha_c(t)$.
2: Inicializar $x(0)$, $\theta(0)$ y $\vartheta(0)$
3: **para** cada paso de tiempo $t = 0, T, 2T, \ldots$ **hacer**
4: Tomar $\Delta u(t)$ de forma aleatoria
5: Aplicar póliza $u(t) = \widehat{h}(x(t);\vartheta(t)) + \Delta u(t)$
6: Medir $x(t+T)$ y $r(t+T)$
7: Obtener error TD (3.60) y actualizar el crítico (3.59) y el actor (3.61)
8: $x(t) \leftarrow x(t+T)$.
9: **fin**

Los algoritmos Actor-Críticos son preferidos para aplicaciones de robots manipuladores debido a que son simples de diseñar, además que evitan que se infiera la póliza a partir de la función de valor. Estos métodos son análogos al método del doble estimador ya que se utilizan dos parametrizaciones (estimadores) para obtener la función de valor y la póliza.

El algoritmo AC en tiempo discreto es escrito como:

$$\begin{aligned} \theta_{t+1} &= \theta_t + \alpha_{c,t}(r_{t+1} + \gamma\Phi^\top(x_{t+1})\theta_t - \Phi^\top(x_t)\theta_t)\Phi(x_t) \\ \vartheta_{t+1} &= \vartheta_t + \alpha_{a,t}(r_{t+1} + \gamma\Phi^\top(x_{t+1})\theta_t - \Phi^\top(x_t)\theta_t)\Delta u_t\Psi(x_t) \end{aligned} \tag{3.62}$$

3.5. Conclusión

En este capítulo se aborda el diseño de controladores robustos utilizando el aprendizaje por refuerzo en tiempo discreto y tiempo continuo. La idea principal es el diseño de la recompensa en el sentido de un problema de minimización con restricciones. Se propusieron dos métodos basados en el aproximador de k-vecinos cercanos y la técnica del doble estimador para lidiar con los problemas de tiempo discreto cuyo espacio de entrada es grande. Para tiempo continuo se utilizó aproximadores Gaussianos y se incorporaron los algoritmos Actor-Críticos. La convergencia de cada algoritmo es demostrada mediante el uso de la propiedad de contracción.

Capítulo 4

Simulaciones y Experimentos

En este capítulo se mostrará el funcionamiento óptimo y robusto de los algoritmos de aprendizaje por refuerzo en tiempo discreto y continuo. Se realiza las comparaciones de cada una de las soluciones de los algoritmos de aprendizaje propuestos en esta tesis y además se mencionan sus ventajas frente a controladores lineales y no lineales. Los problemas a resolver son el balanceo de un sistema carro-péndulo (únicamente simulación) y el control de posición de un robot de 2 grados de libertad (GDL) (resultados experimentales).

4.1. Tiempo discreto

4.1.1. Balanceo del sistema carro-péndulo

El modelo dinámico y parámetros del sistema carro-péndulo es mostrado en el Apéndice B.2. El principal objetivo de control es balancear el péndulo al mover carro. Para lograr el balanceo del péndulo, la posición del carro es restringida bajo los límites $x_c \in [-5{,}5]$ m.

Los métodos de aprendizaje por refuerzo son entrenados para aprender el control estabilizante respecto al peor caso de perturbación paramétrica, es decir, se incrementaron los parámetros al 100 % de su valor original, e.g., de la Tabla B.1 se tiene que los parámetros son $m = 0.1$ kg, $M = 1$ kg y $l = 0.5$m, entonces la peor perturbación paramétrica es el doble de esos valores, i.e., $m = 0.2$ kg, $M = 2$ kg, $l = 1$ m. Dicho incremento se realiza en un tiempo de simulación $t > 10$ segundos, debido a que un controlador clásico puede estabilizar el sistema en un tiempo menor a 5 segundos. Las condiciones iniciales son $x_0 = [x_c, \dot{x}_c, q, \dot{q}]^\top = [0, 0, 0, 0.1]^\top$. Se utilizan 1000 episodios para entrenar los algoritmos. Cada episodio cuenta con 2000 pasos.

El algoritmo de Q-learning es diseñado mediante la aproximación (2.14) utilizando NRBFs como aproximadores paramétricos y el algoritmo de K-means para calcular la ubicación de los centroides. El algoritmo Actor-Crítico discreto (3.62) utiliza NRBFs como aproximadores tanto para el crítico como para el actor, donde por simplicidad se tiene que $\Phi(\cdot) = \Psi(\cdot)$. Se utilizan los Algoritmos 3.3 y 3.4 para diseñar los métodos LS-DA (espacio de estados grandes y acciones discretas) y LS-LA (espacios de estados y acción grandes), respectivamente. Todos los algoritmos utilizan la recompensa robusta. Los hiperparámetros de aprendizaje utilizados vienen dado en la Tabla 4.1.

Tabla 4.1: Hiperparámetros de aprendizaje del Carro-Péndulo: Caso Discreto

Parámetro	**Q-learning**	**AC**	**LS-DA**	**LS-LA**
α_t	0.3	-	0.09	0.3
$\alpha_{c,t}, \alpha_{a,t}$	-	0.3,0.05	-	-
γ	1.0	1.0	1.0	1.0
NRBF	$[10, 5, 10, 5, 5]^\top$	$[10, 5, 10, 5]^\top$	-	-
k	-	-	8.0	8.0

Cuando los algoritmos de aprendizaje por refuerzo logran estabilizar el sistema, entonces el error TD es modificado a:

$$\delta_t = \begin{cases} r_{t+1} - \mathcal{Q}_t^{A \circ B}(kNN_t, u_t) & \text{si estabiliza el sistema} \\ \text{Error TD (3.37),(3.38) o (3.46)} & \text{otro caso} \end{cases} \tag{4.1}$$

la expresión anterior sirve para iniciar nuevamente con el aprendizaje si existe un cambio en la dinámica del sistema, en otro caso se mantendrá con la misma solución.

Después que los controladores de RL han sido entrenados, se comparan con la solución del control LQR (solución $\mathcal{H}_2$) utilizando la solución de la ecuación de Riccati (M12) donde es necesario conocer la dinámica del sistema. El control LQR tiene la siguiente forma

$$u^*(x_t) = -Kx_t,$$

donde $K = [3.1623, 28.9671, 3.5363, 3.7803]^\top$. El control PID y por modos deslizantes (SMC) no requieren conocimiento de la dinámica. El control PID requiere un proceso de sintonización para encontrar ganancias adecuadas [13]. SMC [15] puede estabilizar el sistema si su ganancia es suficientemente grande, sin embargo causa el problema de

chattering. Por simplicidad se utiliza un SMC de primer orden de la forma

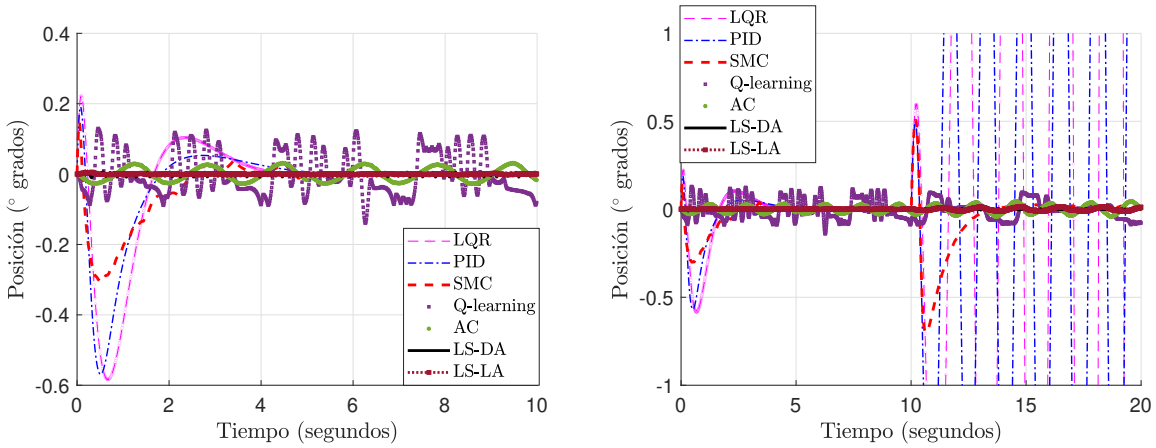
$$u(x_t) = K_m sign(Kx_t),$$

el cual utiliza la misma ganancia K del control LQR para la superficie de deslizamiento, con $K_m = 1$. El control PID tiene la forma:

$$u_t = K_p[x_c, q]^\top + K_i \int_0^t [x_c, q]^\top d\tau + K_d[\dot{x}_c, \dot{q}]^\top,$$

donde K_p, K_i y K_d son las ganancias proporcional, integral y derivativa, respectivamente. Se utilizó el toolbox de control de Matlab para sintonizar el control PID con los parámetros estándar del sistema, obteniendo los siguientes valores: $K_p = [5.12, 20.34]^\top$, $K_i = [1.54, 0.57]^\top$ y $K_d = [1.51, 1.56]^\top$.

En la Figura 4.1(a) se muestra el caso ideal (no hay perturbaciones). Todos los controladores, LQR, PID, SMC y de aprendizaje por refuerzo (Q-learning, AC, LS-DA, LS-LA) trabajan de forma adecuada y estabilizan el péndulo en su posición vertical. Después de 10 segundos, se modificaron los parámetros al incrementarlos por el 80 % de su valor real. En la Figura 4.1(b) se muestra los resultados del caso perturbado.



(a) Caso ideal. (b) Caso perturbado

Figura 4.1: Comparaciones de la posición del péndulo: Caso discreto

Los resultados muestran que el control LQR y PID son inestables debido al cambio paramétrico. SMC utiliza su técnica de chattering para compensar la perturbación. Los métodos de aprendizaje por refuerzo aprendieron a estabilizar el péndulo utilizando el peor caso (100 % de cambio paramétrico), donde muestran resultados satisfactorios y robustos.

Cuando los parámetros son incrementados por el 90 % de su valor real, el control SMC se inestabiliza debido a que su ganancia no es lo suficientemente grande para compensar la perturbación, sin embargo, los métodos propuestos de RL continúan siendo estables. Para estabilizar los controladores PID y SMC en presencia de perturbaciones se requiere sintonizarlos nuevamente, mientras que RL no lo requiere debido a que ya aprendió a estabilizar dichos casos.

En la Figura 4.2 se muestra la gráfica de barras del error medio $\overline{e} = \frac{1}{n}\sum_{i=1}^{n} e(i)$ de los métodos de RL. Las gráficas de barras se dividen en dos, la barra azul representa el error medio del método de aprendizaje cuando no hay perturbación, y la barra roja representa el error medio cuando la perturbación es aplicada. Todos los métodos de RL son robustos, Q-learning presenta error medio grande en comparación con los otros métodos de RL debido a que presenta el problema de sobreestimación. Cuando la perturbación es aplicada, el error medio se incrementa (ver barra roja), donde los métodos propuestos presentan un incremento menor.

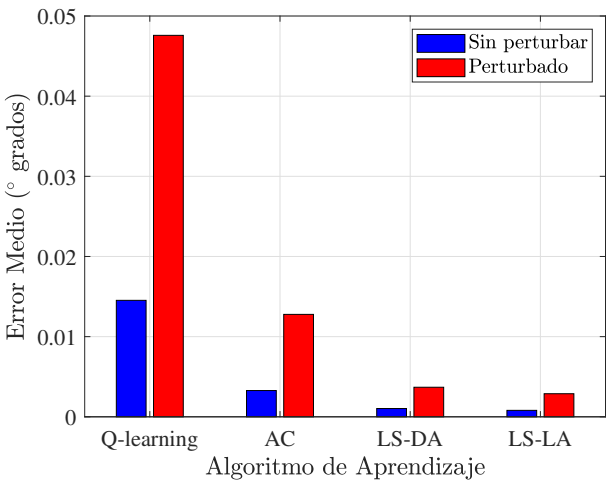


Figura 4.2: Error medio de los métodos de RL

Cuando la función Q converge, es posible obtener una póliza de control óptima y robusta usando el principio de optimalidad de Bellman como:

$$u_t^* = \underset{u}{\operatorname{argmin}}\, \mathcal{Q}^*(kNN_t, u). \tag{4.2}$$

Se observa que el control óptimo/robusto usa el conjunto de kNN, en lugar del estado x_t porque la función Q^* es diseñada mediante el uso de los k-vecinos cercanos.

Con los resultados mostrados surge la inquietud del por qué utilizar los métodos propuestos de LS-DA y LS-LA si se obtiene una respuesta casi óptima y robusta utilizando un algoritmo simple como Q-learning. La respuesta radica en la póliza aprendida, es decir, el método de Q-learning sobrestima las acciones debido el uso del operador mín, en cambio los métodos de LS-DA y LS-LA evitan dicha sobre-estimación como se observa en la Figura 4.3.

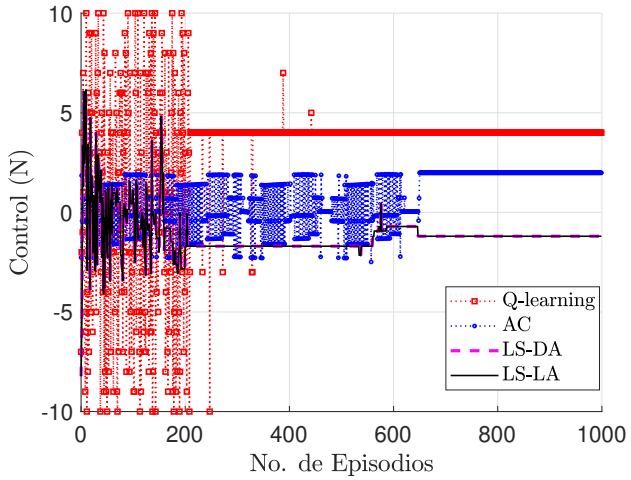


Figura 4.3: Control u_t

En la Figura anterior se observa que Q-learning, por su diseño, busca la póliza de control óptima en cada iteración y sobrestima las acciones debido a que toma la acción que minimiza la función de valor en un cierto estado, además que tanto Q-learning y AC son sensibles a la precisión del aproximador. Por el otro lado kNN-TD doble evita la sobre-estimación de las acciones mediante el uso del doble estimador. En el caso de LS-LA se obtiene una póliza más suave en comparación a los otros métodos debido a que se cuenta con más acciones posibles.

4.1.2. Control de posición de un robot planar de 2 GDL

Se utiliza un robot planar de 2 GDL (ver Apéndice B.1). El robot utilizado se observa en la Figura 4.4.

La posición deseada es $q_d = \left[\frac{5\pi}{6}, \frac{\pi}{4}\right]^\top$ rad. El objetivo de control es forzar las dos variables articulares q_1 y q_2 a la posición deseada q_d. En este caso, únicamente se compara el desempeño del control PID con los métodos de aprendizaje por refuerzo (Q-learning,

Figura 4.4: Robot planar de 2 GDL

Tabla 4.2: Hiperparámetros de aprendizaje del Robot Planar: Caso Discreto

Parámetro	Q**-learning**	**LS-DA**	**LS-LA**
α_t	0.5	0.4	0.1
γ	1.0	1.0	1.0
NRBF	$[20, 20, 10, 10]^\top$	-	-
k	-	9.0	9.0

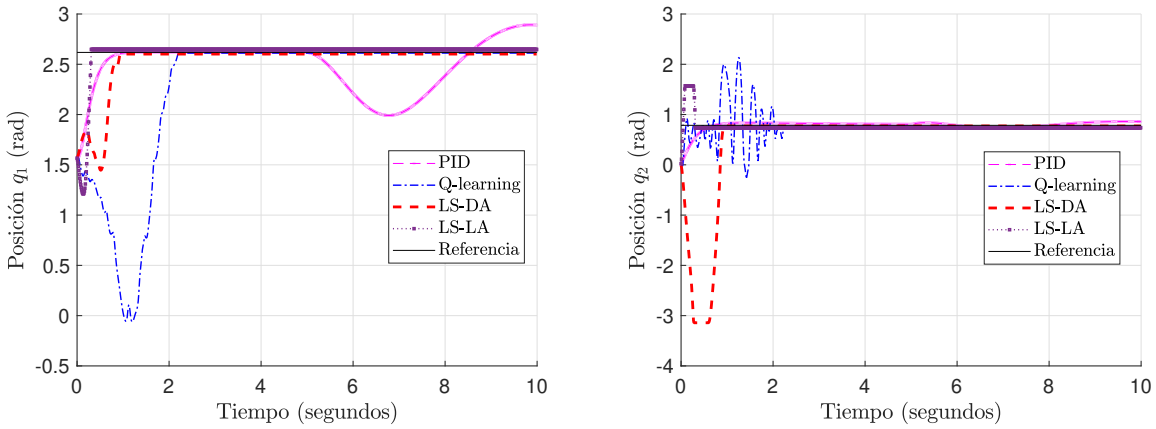
LS-DA,LS-LA). El proceso de aprendizaje es modelado como agentes desacoplados al tener recompensas diferentes para cada grado de libertad.

Se añade una perturbación de la forma:

$$\omega = 7sgn(\sin(3\pi t)) + 8\sin(5\pi t),$$

en la fase de aprendizaje de los métodos de RL en un tiempo de $t = 10$ segundos, la cual es la peor perturbación disponible.

Se sintonizaron las ganancias del control PID manualmente sin considerar perturbaciones, obteniendo $K_p = diag[90, 90]$, $K_i = diag[15, 15]$ Y $K_d = diag[50, 50]$. La sintonización de los hiperparámetros de aprendizaje es realizada de forma iterativa hasta obtener el mejor desempeño. Los parámetros óptimos de los métodos de aprendizaje vienen dados en la Tabla 4.2. Se utiliza 1,000 episodios con 1,000 pasos cada uno. Después del aprendizaje se fijan sus hiperparámetros. En el instante de tiempo $t = 5$ segundos, una perturbación de la forma $\omega = 5sgn[\sin(8\pi t)]$, es aplicada en el primer grado de libertad. Los resultados se observan en la Figura 4.5.



(a) Seguimiento de q_1 (b) Seguimiento de q_2

Figura 4.5: Resultados robot planar: Caso discreto

Se observa que el control PID no puede estabilizar q_1 y se presentan oscilaciones sostenidas cuando la perturbación es aplicada. Por el otro lado, los métodos de aprendizaje por refuerzo son robustos y logran el objetivo de control. Para q_2, todos los controladores son robustos y estables debido a la forma que fue modelado el problema. De forma similar al sistema carro péndulo, Q-learning sobrestima las acciones de control y su funcionamiento es similar a un control por modos deslizantes.

Debido a la naturaleza del problema es preferible utilizar agentes mutuamente excluyentes para garantizar el control de posición y tener un problema desacoplado.

4.2. Tiempo continuo

En esta sección de mostrará el funcionamiento de los algoritmos de aprendizaje por refuerzo robusto en tiempo continuo utilizando el método crítico (CT-CL) y el método Actor-Crítico (CT-ACL) para el problema de balanceo del sistema carro-péndulo y el control de posición del robot planar de 2 GDL.

4.2.1. Balanceo del sistema carro-péndulo

El modelo del sistema carro-péndulo se presenta en el Apéndice B.2. Se desea estabilizar el péndulo en la posición vertical al mover carro. En este caso se comparan los resultados de los algoritmos de CT-CL y CT-ACL con el control óptimo LQR utilizando la solución

de la ARE (N10).

Los métodos de aprendizaje por refuerzo son diseñados para satisfacer (3.12), donde aprenden a estabilizar el péndulo en la posición vertical utilizando la peor perturbación disponible. La perturbación es:

$$\omega = 5\sin(\pi t).$$

La sintonización de los parámetros de aprendizaje es realizada de forma iterativa hasta obtener el mejor desempeño. Los parámetros de aprendizaje óptimos y el número de RBFs utilizadas vienen dadas en la Tabla 4.3.

Tabla 4.3: Hiperparámetros de aprendizaje del Carro-Péndulo: Caso Continuo

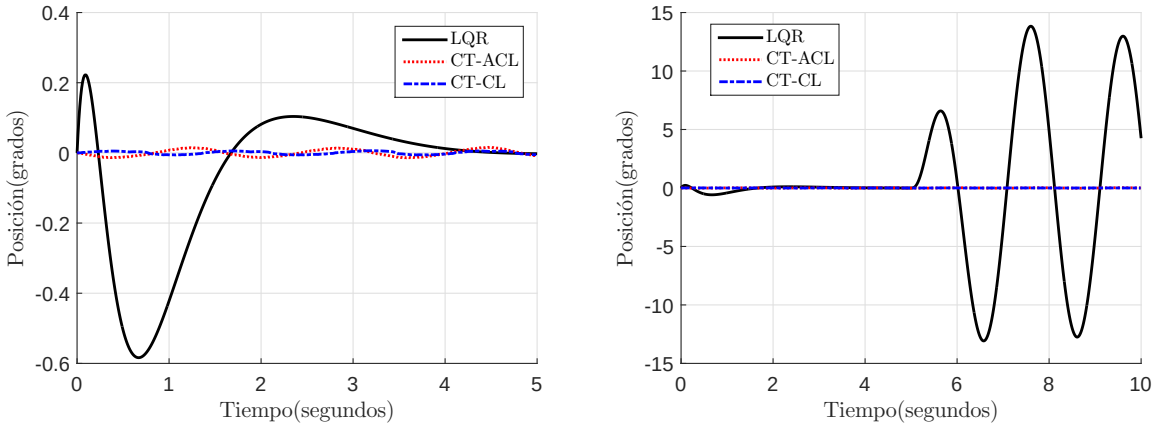
Parámetro	**Descripción**	**CT-CL**	**CT-ACL**
RBFs	No. de RBFs	$[10, 5, 10, 5, 5]^\top$	$[10, 5, 10, 5]^\top$
$\alpha(t)$	Parámetro de aprendizaje	0.3	-
γ	Factor de descuento	1.0	1.0
$\alpha_c(t)$	Parámetro del crítico	-	0.3
$\alpha_a(t)$	Parámetro del actor	-	0.05

El número de RBFs esta dado por un vector cuyos componentes son el número de RBFs utilizados en un cierto estado. Se observa que para CT-CL se tiene cinco dimensiones en lugar de cuatro porque se le añade el espacio de control ó espacio de acciones. Por simplicidad las BFs del actor y el crítico son las mismas, es decir, $\Phi(x) = \Psi(x)$. La condición inicial es $x_0 = [x_c, \dot{x}_c, q, \dot{q}]^\top = [0, 0, 0, 0.1]^\top$. Se utilizan 1000 episodios para entrenar los algoritmos de RL. Cada episodio tiene 1000 pasos.

Las simulaciones consisten en dos partes: cuando $t \leq 5$ no hay perturbación, es decir, $\omega = 0$, el cual es el caso ideal y su solución se observa en la Figura 4.6(a). Se observa que los algoritmos de RL y LQR trabajan adecuadamente y estabilizan el péndulo en los primeros 5 segundos. Cuando $t > 5$ segundos, se aplica una perturbación de $\omega = 4\sin(\pi t)$, el cual es es el caso perturbado y su solución viene dada en la Figura 4.6(b). La solución del control LQR es inestable en presencia de perturbaciones y no puede estabilizar la posición del péndulo, por el otro lado, los métodos de aprendizaje por refuerzo son estables y robustos porque aprendieron a estabilizar el peor caso, i.e., $\omega = 5\sin(\pi t)$.

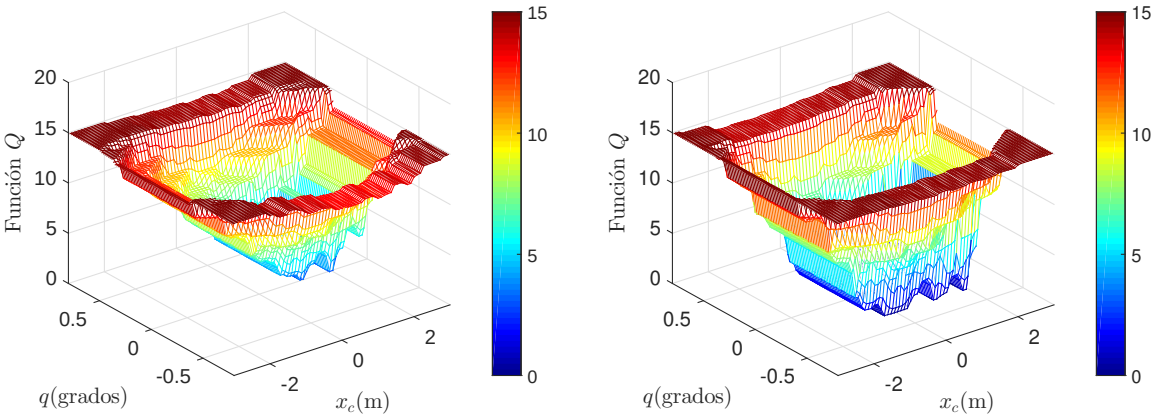
La principal diferencia entre el método CT-ACL y CT-CL es la robustez de la función de valor. El método CT-ACL aprende dos funciones separadas, la función de valor y la póliza. El actor es actualizado usando el error TD de la función de valor mientras que el

crítico es actualizado indirectamente por la póliza seguida. Por lo tanto, el crítico depende en que tan bien el actor aprende la póliza de control para alcanzar el objetivo de control. Por el otro lado, para el método CT-CL su función de valor tiene conocimiento de la póliza seguida que permite conocer como la póliza de control afecta el sistema en un cierto estado.



(a) Caso ideal. (b) Caso perturbado

Figura 4.6: Comparaciones de la posición del péndulo: Caso continuo



(a) Curva CT-ACL (b) Curva CT-CL

Figura 4.7: Curvas de aprendizaje de la función Q para $\dot{x}_c = \dot{q} = 0$

En la Figura 4.7 se muestra una de las curvas de la función Q entre la posición del carro x_c y la posición del péndulo q cuando la velocidad del carro y péndulo son cero, $\dot{x}_c = \dot{q} = 0$. La curva CT-ACL muestra que la función Q robusta (se recuerda que para

una póliza de control fija $h(x(t))$ la función de valor satisface $V(x(t)) = Q(x(t), h(x(t)))$, donde el péndulo es estabilizado en un cierto intervalo de la posición del carro. La curva CT-CL muestra una función Q robusta diferente, donde el intervalo de la posición del carro es más grande que el método CT-ACL. Este intervalo mayor significa que el carro puede estabilizar el péndulo en diferentes posiciones del carro y mantener la propiedad de robustez. Se utiliza la integral del error cuadrático (IEC) para mostrar la robustez del aprendizaje por refuerzo en tiempo continuo:

$$IEC = \int_{t_0}^{t} (\bar{k}\tilde{q})^2 d\tau,$$

donde $\tilde{q} = -q$ y $\bar{k}$ es un factor de escalamiento. El integrador de la IEC es reiniciado en cada cambio de signo de la perturbación, i.e., $reset = sign(\omega)$. Se utiliza un factor de escalamiento de $\bar{k} = 100$, y los resultados vienen dados en la Figura 4.8.

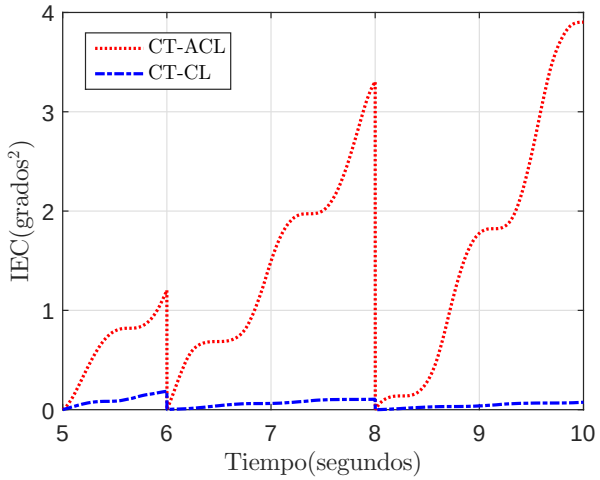


Figura 4.8: Comparaciones de IEC

De la gráfica del IEC se observa que para el método CT-ACL el error incrementa cuando el carro se mueve del valor casi óptimo y robusto, mientras que el método CT-CL muestra que el error de posición no aumenta porque su póliza de control es más robusta.

4.2.2. Control de posición de un robot planar de 2 GDL

Se utiliza el robot de 2 GDL de la Figura 4.4 para mostrar la robustez de los algoritmos de RL. La posición deseada viene dada en la Sección 4.1.2. La sintonización de los

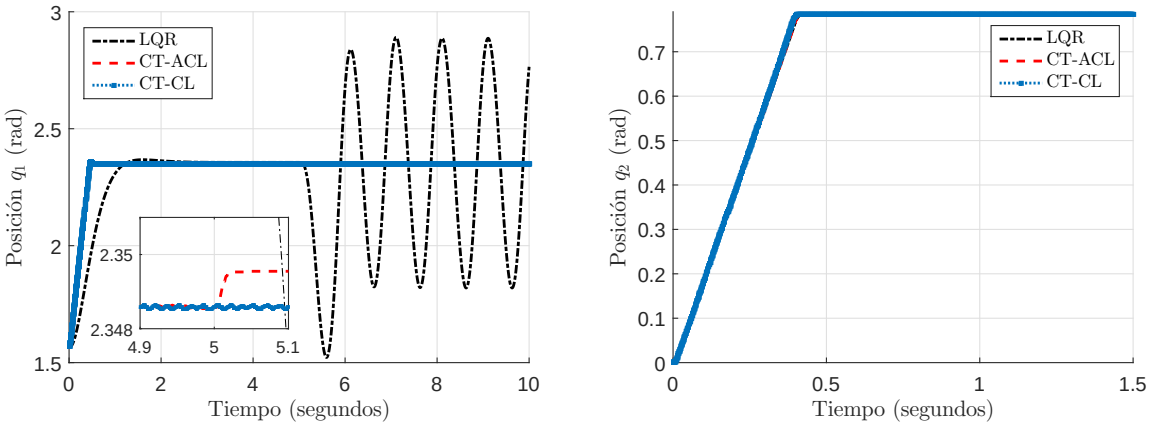
hiperparámetros es realizada mediante iteración hasta obtener el mejor desempeño. Los parámetros de aprendizaje óptimos y número de RBFs son dados en la Tabla 4.4.

Tabla 4.4: Hiperparámetros de aprendizaje del Robot Planar: Caso Continuo

Parámetro	**Descripción**	**CT-CL**	**CT-ACL**
RBFs	No. de RBFs	$[20, 10, 10]^\top$	$[20, 10]^\top$
$\alpha(t)$	Parámetro de aprendizaje	0.3	-
γ	Factor de descuento	1.0	1.0
$\alpha_c(t)$	Parámetro del crítico	-	0.3
$\alpha_a(t)$	Parámetro del actor	-	0.1

Debido a que se debe controlar 2 GDL se requiere dos funciones Q. Cada función Q utiliza el mismo número de RBFs y parámetros de aprendizaje. El número de episodios son 1,000 con 1,000 pasos por episodio. Los métodos de RL aprenden el objetivo de control utilizando la siguiente peor perturbación en el primer brazo del robot:

$$\omega = 30\sin(2\pi t)$$



(a) Seguimiento de q_1 (b) Seguimiento de q_2

Figura 4.9: Resultados robot planar: Caso continuo

Los resultados de las comparaciones son dadas en la Figura 4.9. En los primeros 5 segundos la perturbación es $\omega = 0$ por lo tanto los controladores presentan buen desempeño. Cuando $t > 5$, se aplica una perturbación de $\omega = 20\sin(2\pi t)$. La solución del control LQR

es inestable, mientras que los métodos de CT-ACL y CT-CL continúan siendo estables, mostrando buena robustez y pequeñas diferencias en los errores de posición. Para el ángulo q_2 todos los controladores son estables y cumplen el objetivo de control.

4.3. Discusión

Los algoritmos propuestos muestran buen desempeño y convergencia a soluciones robustas y casi óptimas. El aproximador kNN ayuda a aprender en un espacio de estado-acción grandes con menor costo computacional. Comparado con los controladores clásicos, tales como LQR, PID y SMC, el aprendizaje por refuerzo basado en la regla de kNN no es sólo robusta respecto a la peor perturbación disponible, sino que tiene buen desempeño en el tiempo transitorio. La recompensa robusta puede garantizar la convergencia del controlador a una solución casi óptima cuando no hay perturbación, y estabilidad con las perturbaciones más grandes. El principal problema de este enfoque es el diseño del espacio de estado-acción debido a que puede incrementar el costo computacional al tener dos estimadores. Una forma de evitar este problema, es mediante el uso de un algoritmo de aprendizaje continuo y utilizando parametrizaciones Gaussianas, los cuales presentan un buen desempeño y para su diseño se puede ocupar el método de K means del capítulo pasado. Se asume que la peor perturbación o una cota superior es conocida, en otro caso se requiere estimar la perturbación fuera de línea. La selección adecuada de los hiperpárametros es un trabajo difícil para cualquier tarea.

4.4. Conclusión

En este capítulo se aborda el diseño de controladores robustos utilizando el aprendizaje por refuerzo en tiempo discreto y tiempo continuo. La idea principal es el diseño de la recompensa en el sentido de un problema de minimización con restricciones. Se propusieron dos métodos basados en el aproximador de k-vecinos cercanos y la técnica del doble estimador para lidiar con los problemas de tiempo discreto cuyo espacio de entrada es grande. Para tiempo continuo se utilizó aproximadores Gaussianos y se incorporaron los algoritmos Actor-Críticos. La convergencia de los algoritmos son presentados. Simulaciones y experimentos son realizados utilizando el problema de estabilización del sistema carro-péndulo y un robot planar de 2 GDL, donde se compararon los algoritmos de aprendizaje propuestos con controladores clásicos con resultados satisfactorios.

Capítulo 5

Conclusiones

El aprendizaje por refuerzo es una herramienta bastante útil y de alto interés en la actual comunidad científica. A pesar de no ser un método reciente, su filosofía de funcionamiento en comunión con otras técnicas de aprendizaje automático y teoría de control han dado solución a problemas de interés brindando al controlador o agente una mayor autonomía y adaptabilidad. Debido a su amplia versatilidad se ha utilizado en tareas de control de robots manipuladores. Sin embargo no es de sorprenderse que problemas propios del aprendizaje por refuerzo sean más notorios en este tipo de tareas.

En esta tesis, se utilizaron técnicas de aprendizaje por refuerzo para el diseño de controladores óptimos y robustos para robots manipuladores. Los controladores presentan un desempeño óptimo y robusto por medio de aproximadores adecuados (paramétricos o no paramétricos). Adicionalmente se dieron algunas soluciones para lidiar con tres problemas específicos: problema de dimensionalidad, diseño de la recompensa y sobreestimación de las acciones de los algoritmos de aprendizaje. En este capítulo, se presentan las observaciones finales del trabajo realizado.

5.1. Observaciones Finales

A continuación se resumirán los resultados y conclusiones obtenidos en esta tesis. El aprendizaje por refuerzo es un amplio problema en el área de control y aprendizaje automático. En esta tesis, se enfocó en desarrollar controladores óptimos y robustos para robots manipuladores utilizando y proponiendo herramientas de aprendizaje por refuerzo, obteniendo los siguientes resultados:

5.1.1. Para Control óptimo

Los controladores óptimos clásicos asumen conocimiento exacto de la dinámica del ambiente o sistema. Sin embargo, esta asunción es bastante fuerte teniendo en su lugar estimados que afectan la optimalidad del controlador. El aprendizaje por refuerzo brinda una alternativa bastante eficiente de encontrar controladores óptimos y adaptables utilizando únicamente mediciones del estado y control. Ambos esquemas de control (clásicos y aprendizaje por refuerzo) son diseñados bajo la teoría de programación dinámica, donde el correcto diseño de las funciones de costo (funciones de valor) y utilidad (recompensa) es indispensable para el correcto funcionamiento y optimalidad del controlador. Esto se debe a que la función de utilidad o recompensa brinda una noción del desempeño del controlador y, que en este caso, se desea minimizar.

En esta tesis, por simplicidad se utilizaron funciones de utilidad cuadráticas en el estado y en el control. Las funciones de valor son de horizonte infinito y además se utilizó un factor de descuento para garantizar la convergencia de los algoritmos de aprendizaje y ponderar con menor medida las nuevas mediciones que el controlador vaya recibiendo. Esto es de bastante utilidad cuando no se tienen perturbaciones o cambios en la dinámica del sistema a controlar. Cabe mencionar que en esta tesis se desarrollaron algoritmos de aprendizaje por refuerzo en tiempo discreto y continuo.

Cuando se cuenta con un espacio discreto de estados y acciones contables, el uso de algoritmos de aprendizaje clásicos como Q-learning y Sarsa son idóneos para dar solución al problema en cuestión. Cuando se tienen espacios de estado y acciones grandes o continuos, como es el caso de robots manipuladores, es necesario utilizar aproximadores. Los aproximadores sirven para lidiar con el problema de dimensionalidad. En esta tesis se utilizaron aproximadores paramétricos y no paramétricos. Los aproximadores paramétricos utilizados son basados en NRBFs donde se propusó utilizar el algoritmo de K-means para ubicar los centros de cada RBF. Esto permite generar una familia de aproximadores que dan solución al problema de control con buenos resultados y menor costo computacional.

En el caso de utilizar espacios continuos, se propusó utilizar la aproximación hacia atrás de Euler y aproximadores basados en NRBF. Mediante el uso de las herramientas anteriores es posible obtener la versión continua del algoritmo de Q-learning y de una clase de algoritmos actor-críticos. La convergencia de cada algoritmo (discreto y continuo) es demostrada mediante el uso de la propiedad de contracción que poseen las funciones de costo y herramientas de ecuaciones diferenciales y álgebra lineal. Esto brinda una metodología general para cualquier aproximador paramétrico acotado.

5.1.2. Para Control robusto

Los algoritmos de aprendizaje por refuerzo son robustos debido a su adaptabilidad a nuevos estados y acciones, sin embargo, el diseño de la recompensa afecta directamente en qué tan robusto puede llegar a ser el controlador. El uso de un factor de descuento es negativo para el diseño de controladores robustos debido a que pondera con menor valor las nuevas recompensas recibidas. Por este motivo, en esta tesis se fijo el parámetro de descuento igual a la unidad.

Generalmente los controladores robustos basados en aprendizaje por refuerzo resuelven iterativamente la ecuación de Hamilton-Jacobi Bellman Isaacs, obteniendo a la salida el control óptimo/robusto y la peor perturbación que el sistema puede permitir para seguir garantizando un desempeño óptimo. En esta tesis se propusó una implementación distinta, es decir, en lugar de utilizar la metodología clásica se propone modelar el problema de control robusto como un problema de optimización bajo restricciones, donde se conoce la cota superior de la peor perturbación que el sistema puede llegar a tener. En este diseño, el algoritmo de aprendizaje aprende la póliza robusta y casi óptima considerando al sistema perturbado en el peor caso.

En la presencia de perturbaciones el controlador requiere considerar nuevos estados y acciones, entonces se requiere el uso de aproximadores. Aunado a esto se tiene un nuevo problema conocido como la sobreestimación de las acciones que ocurre al utilizar el operador del mínimo en las reglas de actualización y exploración. En esta tesis, para el caso discreto se propusó utilizar un aprendizaje por refuerzo modificado usando como aproximador el algoritmo de k vecinos cercanos y la técnica del doble estimador. Este aproximador mejora el costo computacional al utilizar únicamente k estados y no requiere procedimientos adicionales como en el caso de NRBFs. El doble estimador ayuda a evitar el problema de sobrestimación de las acciones utilizando dos funciones de valor ya sea conmutadas o en serie. La convergencia de cada algoritmo es demostrada mediante el uso de la propiedad de contracción. En comparación con controladores clásicos como PID, SMC y LQR, los algoritmos de aprendizaje por refuerzo presentan mayor robustez en presencia de perturbaciones iguales o menores al peor caso, en cambio, los controladores clásicos son inestables y requieren ser nuevamente sintonizados.

Para el caso continuo, se utilizó la versión continua de Q-learning y el algoritmo actor-crítico que, por naturaleza, utiliza dos funciones de valor que evita la sobreestimación de las acciones. Se demostró que el algoritmo de aprendizaje Q-learning presenta una mejor robustez en comparación del algoritmo actor-crítico debido a que el algoritmo actor crítico

utiliza una función de valor de estado que no toma en cuenta las acciones provocando que su robustez se limite a un cierto número de acciones. Además igual se observa que los algoritmos de aprendizaje presentan mejor desempeño que los controladores clásicos sin necesidad de volver a entrenar el algoritmo.

5.2. Trabajo futuro

Diversas áreas de oportunidad fueron encontradas al escribir esta tesis. Por la gran relevancia y capacidad de sinergía con otra clase de algoritmos los siguientes problemas deben ser resueltos:

- El uso de técnicas de aprendizaje por refuerzo profundo no ha sido correctamente estudiado en aplicaciones de control, siendo uno de los campos de estudio más amplios actualmente. Debido al éxito obtenido por los algoritmos de aprendizaje profundo en aplicaciones que manejan muchos datos, no es extraño que se ajusten a aplicaciones de aprendizaje por refuerzo utilizando redes neuronales profundas como aproximador de la función de valor. El desarrollo de nuevas técnicas con validación teórica y experimental provee una ganancia sustancial de esta nueva tendencia.

- Los algoritmos de aprendizaje propuestos en esta tesis utiliza un parámetro de aprendizaje constante basado en la técnica de gradiente descendiente, sin embargo existen otros métodos que pueden ser utilizados y probados. Estos métodos incluyen técnicas basadas en el Hessiano, regularización y validación cruzada.

- Los algoritmos de aprendizaje por refuerzo utilizados en esta tesis aprenden desde cero la póliza y función de valor óptima/robusta lo cual se traduce en un tiempo de aprendizaje largo. Existen diferentes técnicas que brindan conocimiento previo al algoritmo y ayuda acelerar el tiempo de convergencia como lo son: aprendizaje por demostraciones, aprendizaje por refuerzo Bayesiano, aprendizaje basado en modelos de referencia e inclusive aprendizaje por refuerzo inverso.

- Una técnica bien conocida del aprendizaje por refuerzo son los rasgos de elegibilidad que brindan al algoritmo de memoria. Por otro lado en colaboración con técnicas de aprendizaje profundo, el uso de LSTM (long-short term memory) brinda al algoritmo de aprendizaje de memoria con una efectividad diferente a los rasgos de elegibilidad. El uso de LSTM en un algoritmo de aprendizaje por refuerzo podría obtener mejores

resultados de aprendizaje en menor tiempo, inclusive una combinación entre rasgos de elegibilidad y LSTM es una propuesta interesante y abierta actualmente.

- En esta tesis, el diseño de la función de valor y recompensa guardan un diseño cuadrático habitual en aplicaciones de optimización. Sin embargo, es posible adoptar nuevos diseños que permita mejorar el aprendizaje y obtener controladores óptimos y robustos.

Apéndice A

Herramientas del Aprendizaje por Refuerzo

A.1. Estimación del Valor de las acciones

Una forma simple de estimar el valor de las acciones es promediar las recompensas r_i recibidas

$$Q_t(x,u) = \frac{\sum_{i=0}^{t} r_{i+1} \cdot m(u_i)}{\sum_{i=0}^{t} m(x_i,u_i)}, \qquad m(x_t,u_t) = \begin{cases} 1 & \text{si } (x_t,u_t) = (x,u) \\ 0 & \text{otro caso} \end{cases} \tag{A.1}$$

donde $m(\cdot)$ es una variable aleatoria que equivale a 1 si $(x_t, u_t) = (x, u)$ y 0 en caso contrario. Si el denominador es cero, entonces $Q_t(x, u)$ es definida con un cierto valor inicial, tal que $Q_0(x, u) = 0$. A este método se le conoce como el método de *media muestreal*.

A.1.1. Implementación incremental

Sea r_i la recompensa recibida después de la i-ésima selección de una acción arbitraria, y sea Q_t el estimado del valor de las acciones después que ha sido seleccionada t veces, la cual puede ser escrita como:

$$Q_t = \frac{r_1 + r_2 + \ldots + r_t}{t} = \frac{1}{t}\sum_{i=1}^{t} r_i.$$

La implementación más sencilla es utilizar todas las recompensas recibidas para estimar el valor de las acciones, sin embargo tiene una desventaja debido que los requerimientos computacionales y de memoria aumentan por el incremento de recompensas. Sin embargo, puede evitarse mediante la construcción de una formula incremental que requiera poco costo computacional para procesar nuevas recompensas. Dada Q_t y la recompensa en el siguiente instante de tiempo, r_{t+1}, la nueva estimación es calculada por:

$$\begin{aligned} Q_{t+1} &= \frac{1}{t+1}\sum_{i=1}^{t+1} r_i = \frac{1}{t+1}\left(r_{t+1} + \sum_{i=1}^{t} r_i\right) = \frac{1}{t+1}\left(r_{t+1} + tQ_t \pm Q_t\right) \\ &= Q_t + \frac{1}{t+1}\left(r_{t+1} - Q_t\right) \end{aligned} \tag{A.2}$$

que se mantiene inclusive cuando $t = 0$, obteniendo $Q_1 = r_1$ para una arbitraria Q_0. El término $\frac{1}{t+1}$ es un parámetro de aprendizaje, el cual cambia a cada instante de tiempo. Generalmente en procesamiento se utiliza un parámetro de aprendizaje de la forma $\frac{1}{t+1}$. Se denota el parámetro de aprendizaje por el símbolo α o α_t.

A.1.2. Seguimiento de un problema no-estacionario

El método de media muestreal mencionado anteriormente es adecuado en ambientes estacionarios, pero no en ambientes que cambian en el tiempo. En tales casos toma importancia considerar más las recompensas actuales que las pasadas. Una de las formas más populares de realizar esto es mediante el uso de un parámetro de aprendizaje fijo. Por ejemplo, la regla incremental (A.2) es modificada por:

$$Q_{t+1} = Q_t + \alpha\left(r_{t+1} - Q_t\right) \tag{A.3}$$

donde $\alpha \in (0, 1]$ es un parámetro de aprendizaje constante. Entonces (A.2) únicamente depende de un valor inicial Q_0 y la media de las recompensas pasadas:

$$\begin{aligned} Q_{t+1} &= \alpha r_{t+1} + (1-\alpha)Q_t \\ &= \alpha r_{t+1} + (1-\alpha)\left(\alpha r_t + (1-\alpha)Q_{t-1}\right) \\ &= \alpha r_{t+1} + \alpha(1-\alpha)r_t + \ldots + \alpha(1-\alpha)^t r_1 + (1-\alpha)^{t+1}Q_0 \\ &= (1-\alpha)^{t+1}Q_0 + \sum_{i=1}^{t+1}\alpha(1-\alpha)^{t+1-i}r_i \end{aligned} \tag{A.4}$$

La ponderación decae de forma exponencial acorde al exponente en el término $(1 - \alpha)$. Este método de actualización es denominado como *media ponderada exponencial*. Un resultado bien conocido de la teoría de aproximación estocástica proporciona la condición requerida para garantizar convergencia con probabilidad 1:

$$\sum_{t=0}^{\infty} \alpha_t = \infty \quad \text{y} \quad \sum_{t=0}^{\infty} \alpha_t^2 < \infty. \tag{A.5}$$

La primer condición garantiza que los pasos de tiempo son suficientemente grandes para eventualmente superar las condiciones iniciales y fluctuaciones aleatorias. La segunda condición garantiza que eventualmente los pasos de tiempo disminuyen lo suficiente para garantizar convergencia. Ambas condiciones de convergencia son satisfechas para el caso de media muestreal, $\alpha_t = \frac{1}{t+1}$, pero no para el caso de un parámetro de aprendizaje constante $\alpha_t = \alpha$.

Apéndice B

Modelos de los Robots

En este apéndice se proporciona el modelo dinámico de los robots manipuladores utilizados en este trabajo se emplea la convención de Denavit-Hartenberg [77] y la metodología de Euler-Lagrange [4].

B.1. Robot planar de 2 GDL

El modelo de un robot planar de 2 GDL (ver Figura B.1) es el siguiente:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau \tag{B.1}$$

donde $M(q)$ es la matriz de inercia, $C(q,\dot{q})$ es la matriz de Coriolis, $G(q)$ es el vector de pares gravitacionales y τ es el par aplicado en cada junta.

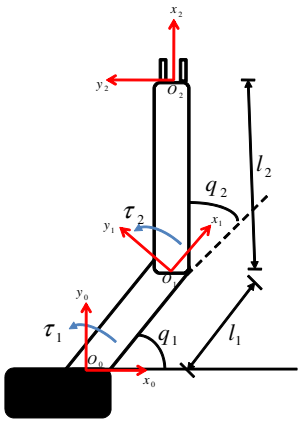


Figura B.1: Robot planar de 2 GDL

Para el robot planar, la dinámica es expresada como:

$$
\begin{aligned}
M(q) &= \begin{bmatrix} (m_1+m_2)l_1^2 + m_2 l_2^2 + 2m_2 l_1 l_2 \cos(q_2) + J_1 + J_2 & m_2 l_2^2 + m_2 l_1 l_2 \cos(q_2) + J_2 \\ m_2 l_2^2 + m_2 l_1 l_2 \cos(q_2) + J_2 & m_2 l_2^2 + J_2 \end{bmatrix} \\
C(q,\dot{q}) &= \begin{bmatrix} -m_2 l_1 l_2 \sin(q_2)\dot{q}_2 & -m_2 l_1 l_2 \sin(q_2)(\dot{q}_1 + \dot{q}_2) \\ m_2 l_1 l_2 \sin(q_2)\dot{q}_1 & 0 \end{bmatrix} \\
G(q) &= \begin{bmatrix} m_1 l_1 g \cos(q_1) + m_2 g\left(l_1 cos(q_1) + l_2 \cos(q_1+q_2)\right) \\ m_2 g\left(l_1 cos(q_1) + l_2 \cos(q_1+q_2)\right) \end{bmatrix}
\end{aligned}
\tag{B.2}
$$

donde $q = [q_1, q_2]^\top$, m_i, l_i, J_i son la masa, longitud e inercia del eslabón i, con $i = 1, 2$.

B.2. Sistema carro-péndulo

La dinámica del sistema carro-péndulo (ver Figura B.2) se expresa como:

$$M(\mathfrak{q})\ddot{\mathfrak{q}} + C(\mathfrak{q},\dot{\mathfrak{q}})\dot{\mathfrak{q}} + G(\mathfrak{q}) = Bu \tag{B.3}$$

donde:

$$
\begin{aligned}
M(\mathfrak{q}) &= \begin{bmatrix} M+m & ml\cos(q) \\ ml\cos(q) & ml^2 \end{bmatrix}, \quad C(\mathfrak{q},\dot{\mathfrak{q}}) = \begin{bmatrix} 0 & -ml\sin(q)\dot{q} \\ 0 & 0 \end{bmatrix}, \\
G(\mathfrak{q}) &= \begin{bmatrix} 0 \\ -mgl\sin(q) \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}
\end{aligned}
\tag{B.4}
$$

donde M es la masa del carro, m la masa del péndulo, l la longitud del péndulo, g la gravedad y F la fuerza aplicada en el carro. Las variables generalizadas están representadas por $\mathfrak{q} = [x_c, q]^\top$, donde x_c es la posición del carro y q el ángulo formado entre el péndulo y la vertical.

La dinámica del sistema (B.4) puede ser escrito como (3.48) como:

$$
\begin{aligned}
\dot{x} &= f(x,u) \\
&= \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ \frac{cd_1 - bd_2}{ac-b^2} \\ \frac{ad_2 - bd_1}{ac-b^2} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \frac{c}{ac-b^2} \\ \frac{b}{b^2-ac} \end{bmatrix} u
\end{aligned}
\tag{B.5}
$$

donde $a = m + M$, $b = ml\cos(q)$, $c = ml^2$, $d_1 = ml\sin(q)\dot{q}^2$ y $d_2 = mgl\sin(q)$. El vector de

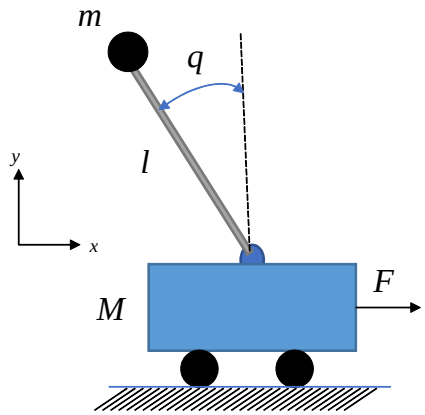


Figura B.2: Problema de balanceo del sistema carro-péndulo

estado es $x = (x_c, \dot{x}_c, q, \dot{q})^\top$ y $u = F$ es la entrada de control. Los parámetros del sistema carro-péndulo utilizados vienen dados en la Tabla B.1.

Tabla B.1: Parámetros del sistema Carro-Péndulo

Parámetro	**Descripción**	**Valor**
m	Masa del péndulo	0.1kg
M	Masa del carro	1.0kg
l	Longitud del péndulo	0.5 m
g	Aceleración de la gravedad	9.81 m/s^2

Bibliografía

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning. An Introduction*. MIT Press, 1998.

[2] L. Buşoniu, R. Babûska, and D. E. Bart De Schutter, *Reinforcement Learning and Dynamic Programming Using Function Approximators*. CRC Press. Taylor & Francis Group, 2009.

[3] J. Kober and J. Peter, "Policy search for motor primitives in robotics," *In Advances in Neural Information Processing Systems (NIPS)*, 2009.

[4] R. Kelly and V. Santibáñez, *Control de Movimiento de Robots Manipuladores*. Prentice Hall, 2003.

[5] N. Hogan, "Impedance control: an approach to manipulation," *Journal of Dynamic Systems, Measurement, and Control*, pp. 1–24, 1985.

[6] F. Ficuciello, L. Villani, and B. Siciliano, "Variable impedance control of redundant manipulators for intuitive human-robot physical interaction," *IEEE Transactions on Robotics*, vol. 31, no. 4, pp. 850–863, 2015.

[7] A. Manan Khan, D. Yun, M. Ali, J. Han, K. Shin, and C. Han, "Adaptive impedance control for upper limb assist exoskeleton," *IEEE International Conference on Robotics and Automation*, pp. 4353–4366, 2015.

[8] A. Perrusquía, W. Yu, A. Soria, and R. Lozano, "Stable admittance control without inverse kinematics," *20th IFAC World Congress(IFAC2017)*, 2017. Toulose, France.

[9] L. Márton, A. Scottedward Hodel, B. Lantos, and Y. Hung, "Underactuated robot control: Comparing LQR, subspace stabilization, and combined error metric approaches," *IEEE Transactions on Industrial Electronics*, vol. 55, no. 10, pp. 3724–3730, 2008.

[10] R. Johansson and M. W. Spong, "Quadratic optimization of impedance control," *Proc. IEEE Int. Conf. Robot. Autom*, vol. 1, pp. 616–621, 1994.

[11] M. Matinfar and K. Hashtrudi-Zaad, "Optimization-based robot compliance control: Geometric and linear quadratic approaches," *The International Journal of Robotics Research*, vol. 24, no. 8, pp. 645–656, 2005.

[12] F. L. Lewis, D. Vrabie, and K. G. Vamvoudakis, "Reinforcement learning and feedback control using natural decision methods to desgin optimal adaptive controllers," *IEEE Control Systems Magazine*, 2012.

[13] J. G. Romero, A. Donaire, R. Ortega, and P. Borja, "Global stabilisation of underactuated mechanical systems via PID passivity-based control.," *IFAC-PapersOnLine*, 2017.

[14] W. Yu, J. Rosen, and X. Li, "PID admittance control for an upper limb exoskeleton," *American Control Conference*, pp. 1124–1129, 2011.

[15] R. Xu and U. Özgüner, "Sliding mode control of a class of underactuated systems.," *Automatica*, vol. 44, 2014.

[16] W. Yu, R. Carmona Rodríguez, and X. Li, "Neural PID admittance control of a robot," *2013 American Control Conference (ACC13)*, 2013. Washington, DC USA.

[17] H. van Haselt, "Double Q-learning," *In Advances in Neural Information Processing Systems (NIPS)*, pp. 2613–2621, 2010.

[18] M. Ganger, E. Duryea, and W. Hu, "Double sarsa and double expected sarsa with shallow and deep learning," *Journal of Data Analysis and Information Processing*, 2016.

[19] C. Wang, Y. Li, S. Sam Ge, and T. Heng Lee, "Optimal critic learning for robot control in time-varying environments," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 10, 2015.

[20] W. B. Powell, "AI,OR and control theory: A rosetta stone for stochastic optimization," tech. rep., Princeton University, 2012.

[21] S. Hart and R. Grupen, "Learning generalizable control programs," *IEEE Transactions on Autonomous Mental Development*, vol. 3, no. 3, pp. 216–231, 2011.

[22] A. G. Barto and S. Mahadevan, "Recent advances in hierarchical reinforcement learning," *Discrete Event Dynamic Systems*, vol. 13, no. 4, pp. 341–379, 2003.

[23] C. G. Atkeson, A. Moore, and S. Stefan, "Locally weighted learning for control," *AI Review*, vol. 11, pp. 75–113, 1997.

[24] P. Abbeel, A. Coates, M. Quigley, and A. Y. Ng, "An application of reinforcement learning to aerobatic helicopter flight," *In Advances in Neural Information Processing Systems (NIPS)*, 2007.

[25] M. P. Deisenroth and C. E. Rasmusen, "PILCO: A model-based and data-efficient approach to policy search," *In 28th International Conference on Machine Learning (ICML)*, 2011.

[26] V. Gullapalli, J. Franklin, and H. Benbrahim, "Acquiring robot skills via reinforcement learning," *IEEE Control Systems Magazine*, vol. 14, no. 1, pp. 13–24, 1994.

[27] N. Kohl and P. Stone, "Policy gradient reinforcement learning for fast quadrupedal locomotion," *In IEEE International Conference on Robotics and Automation (ICRA)*, 2004.

[28] J. A. Bagnell and J. C. Schneider, "Autonomous helicopter control using reinforcement learning policy search methods," *In IEEE International Conference on Robotics and Automation (ICRA)*, 2001.

[29] H. Miyamoto, S. Schaal, F. Gandolfo, H. Gomi, Y. Koike, R. Osu, E. Nakano, Y. Wada, and M. Kawato, "A Kendama learning robot bases on bi-directional theory," *Neural Networks*, vol. 9, no. 8, pp. 1281–1302, 1996.

[30] J. Peters and S. Schaal, "Learning to control in operational space," *International Journal of Robotics Research*, vol. 27, no. 2, pp. 197–212, 2007.

[31] R. Tedrake, T. W. Zhang, and H. S. Seung, "Learning to walk in 20 minutes," *In Yale Workshop on Adaptive and Learning Systems*, 2005.

[32] M. P. Deisenroth, C. E. Rasmussen, and D. Fox, "Learning to control low-cost manipulator using data-efficient reinforcement learning.," *In Robotics: Science and Systems (R:SS)*, 2011.

[33] H. Benbrahim, J. Doleac, J. Franklin, and O. Selfridge, "Real-time learning: A ball on a beam," *International Joint Conference on Neural Networks (IJCNN)*, 1992.

[34] B. Nemec, M. Tamošiūnaitė, F. Wörgötter, and A. Ude, "Task adaption thorough exploration and action sequencing," *IEEE-RAS International Conference on Humanoid Robots (HUMANOIDS)*, 2009.

[35] M. Tokic, W. Ertel, and J. Fessler, "The crawler, a class room demonstrator for reinforcement learning," *International Florida Artificial Intelligence Research Society Conference (FLAIRS)*, 2009.

[36] H. Kimura, T. Yamashita, and S. Kobayashi, "Reinforcement learning of walking behaviour for a four-legged robot," *IEEE Conference on Decision and Control (CDC)*, 2001.

[37] R. A. Willgross and J. Igbal, "Reinforcement learning of behaviors in mobile robots using noisy infrared sensing," *Australian Conference on Robotics and Automation*, 1999.

[38] L. Paletta, G. Fritz, F. Kintzler, J. Irran, and G. Dorffner, "Perception and developmental learning of affordances in autonomous robots," *Hertzberg J., Beetz M. and Englert R., editors, KI 2007: Advances in Artificial Intelligence*, vol. 4667, pp. 235–250, 2007. Lecture Notes in Computer Science, Springer.

[39] C. Kwok and D. Fox, "Reinforcement learning for sensing staregies," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2004.

[40] T. Yasuda and K. Ohkura, "A reinforcement learning technique with an adaptive action generator for a multi-robot system," *International Conference on Simulation of Adaptive Behavior (SAB)*, 2008.

[41] J. H. Piater, S. Jodogne, R. Detry, D. Kraft, N. Krüger, O. Kroemer, and J. Peters, "Learning visual representations for perception-action systems," *International Journal of Robotics Research*, vol. 30, no. 3, pp. 294–307, 2011.

[42] P. Fidelman and P. Stone, "Learning ball acquisition on a physical robot," *International Symposium on Robotics and Automation (ISRA)*, 2004.

[43] G. D. Konidaris, S. Kuidersma, R. Grupen, and A. G. Barto, "Autonomous skill acquisition on a mobile manipulator," *AAAI Conference on Artificial Intelligence (AAAI)*, 2011.

[44] G. D. Konidars, S. Kuindersma, R. Grupen, and A. G. Barto, "Robot learning from demonstration by constructing skill trees," *International Journal of Robotics Research*, vol. 31, no. 3, pp. 360–375, 2012.

[45] R. Platt, R. A. Gruphen, and A. H. Fagg, "Improving grasp skills using schema structured learning," *International Conference on Development and Learning*, 2006.

[46] V. Soni and S. P. Singh, "Reinforcement learning of hierarchical skills on the Sony AIBO robot," *International Conference on Development and Learning (ICDL)*, 2006.

[47] C. An, C. G. Atkeson, and J. M. Hollerbach, "Model-based control of a robot manipulator," *MIT Press, Cambridge, MA, USA*, 1988.

[48] S. Schaal, "Learning from demonstration," *In Advances in Neural Information Processing Systems (NIPS)*, 1996.

[49] Y. Duan, B. Cui, and H. Yang, "Robot navigation based on fuzzy RL algorithm," *International Symposium on Neural Networks (ISNN)*, 2008.

[50] C. Gaskett, L. Fletcher, and A. Zelinsky, "Reinforcement learning for a vision based mobile robot," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2000.

[51] R. Hafner and M. Riedmiller, "Reinforcement learning on a omnidirectional mobile robot," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2003.

[52] I. Grondman, M. Vaandrager, L. Buşoniu, R. Babûska, and E. Schuitema, "Efficient model learning methods for actor-critic control," *IEEE Transactions on Systems, man, and cybernetics. Part B: Cybernetics*, vol. 42, no. 3, pp. 591–602, 2012.

[53] O. Kroemeri, R. Detry, J. Piater, and J. Peters, "Active learning using mean shift optimization for robot grasping," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2009.

[54] O. Kroemer, R. Detry, J. Piater, and J. Peter, "Combining active learning and reactive control for robot grasping," *Robotics and Autonomous Systems*, vol. 58, no. 9, pp. 1105–1116, 2010.

[55] A. Rottmann, C. Plagemann, P. Hilgers, and W. Burgard, "Autonomous blimp control using model-free reinforcement learning in a continuous state and action spac," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2007.

[56] K. Gräve, J. Stückler, and S. Behnke, "Learning motion skills from expert demonstrations and own experience using gaussian process regression," *Joint International Symposium on Robotics (ISR) and German Conference on Robotics (ROBOTIK)*, 2010.

[57] J. Kober, J. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[58] A. Benveniste, M. Métivier, and P. Priouret, *Adaptive Algorithms and Stochastic Approximations*, vol. 22. Berlin: Springer-Verlag, 1990. Applications of Mathematics.

[59] M. Tokic, "Adaptive ϵ-greedy exploration in reinforcement learning based on value differences," in *KI 2010: Advances in Artificial Intelligence* (R. In: Dillmann, J. Beyerer, U. Hanebeck, and T. Schultz, eds.), vol. 6359, pp. 203–210, Springer, Berlin, Heidelberg, 2010. Lecture Notes in Computer Science.

[60] I. Grondman, L. Buşoniu, and R. Babûska, "Model learning actor-critic algorithms: Performance evaluation in a motion control task," *51st IEEE Conference on Decision and Control*, pp. 5272–5277, 2012.

[61] I. Grondman, M. Vaandrager, L. Buşoniu, R. Babûska, and E. Schuitema, "Actor-critic control with reference model learning," *Proc. of the 18th World Congress The International Federation of Automatic Control*, pp. 14723–14728, 2011.

[62] W. Barbakh and F. C., "Clustering with reinforcement learning." In: Yin H., Tino P., Corchado E., Byrne W., Yao X. (eds) Intelligent Data Engineering and Automated Learning - IDEAL. Lecture Notes in Computer Science, vol 4881. Springer, Berlin, Heidelberg, 2007.

[63] S. Bose and M. Huber, "Semi-unsupervised clustering using reinforcement learning," *Proceedings of the Twenty-Ninth International Florida Artificial Intelligence Research Society Conference*, pp. 150–153, 2016.

[64] N. Tziortziotis and K. Blekas, "Model-based reinforcement learning using on-line clustering," *IEEE 24th International Conference on Tools with Artificial Intelligence (ICTAI)*, 2012.

[65] A. Likas, "A reinforcement learning approach to online clustering," *Neural Computation*, vol. 11, pp. 1915–1932, 1999.

[66] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognition Letters*, vol. 31, pp. 651–666, 2010.

[67] M. Sugiyama, H. Hachiya, C. Towell, and S. Vijayakumar, "Value function approximation on non-linear manifolds for robot motor control," *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'07)*, 2007.

[68] J. Tsitsiklis and B. Van Roy, "Feature based methods for large scale dynamic programming," *Machine Learning*, vol. 22, pp. 59–94, 1996.

[69] M. Palanisamy, H. Modares, F. L. Lewis, and M. Aurangzeb, "Continuous -time Q-learning for infinite-horizon discounted cost linear quadratic regulator problems," *IEEE Transactions on Cybernetics*, 2014.

[70] J. Y. Lee, J. B. Park, and Y. H. Choi, "Integral reinforcement learning for continuous-time input-affine nonlinear systems with simultaneous invariant explorations," *IEEE Transactions on Neural Networks and Learning Systems*, 2014.

[71] K. Doya, "Reinforcement learning in continuous time and space," *Neural Computation*, vol. 12, no. 1, pp. 219–245, 2000.

[72] J. A. Martín H., J. de Lope, and D. Maravall, "The kNN-TD reinforcement learning algorithm," *Springer-Verlag, J. Mira et al. (Eds): IWINAC 2009, Part I, LNCS 5601*, pp. 305–314, 2009.

[73] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. IT-13, pp. 21–27, 1967.

[74] S. Sun and R. Huang, "An adaptive k-nearest neighbor algorithm," *2010 Seventh International Conference on Fuzzy Systems and Knowledge Discovery*, 2010.

[75] J. A. Martín H. and J. de Lope, "Ex$\langle a \rangle$: An effective algorithm for continuous actions reinforcement learning problems," *Industrial Electronics*, 2009.

[76] F. S. Melo and M. I. Ribeiro, "Convergence of Q-learning with linear function approximation," *Proceedings of the European Control Conference*, 2007.

[77] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Dynamics and Control*. John Wiley, 2004.