Aprendizaje por **refuerzo continuo** y **retroalimentación interactiva**

Ángel Ayala
Ingeniero en Computación.

Claudio Henríquez
Magíster en Ingeniería Informática.

Francisco Cruz
Doctor en Ciencias de la Computación.

Las investigaciones en el área de sistemas inteligentes han permitido encontrar varios métodos para que una máquina pueda obtener conocimiento, uno de éstos corresponde a aprendizaje por refuerzo. El problema de este método es el tiempo que requiere para aprender a resolver un problema, por lo que este trabajo aborda el enfoque de aprendizaje por refuerzo interactivo como una manera de solución para el entrenamiento de agentes robóticos. Por otra parte este trabajo aborda los problemas desde una representación continua además del enfoque interactivo. Para esto se experimenta con entornos simulados que poseen diferentes representación en su vector de estado para mostrar la eficiencia de este enfoque bajo una determinada probabilidad de interacción. Los resultados obtenidos evidencian una convergencia mayor del aprendizaje en términos de la recompensa acumulada por los agentes, demostrando la eficiencia del enfoque presentado.

Introducción

En la actualidad la tecnología nos ha brindado la posibilidad de ejecutar tareas repetitivas y de gran esfuerzo gracias a mecanismos robóticos programados para realizar una actividad en específico, por ejemplo, en una fábrica de producción lineal existen una variedad de estructuras robóticas, ubicadas una después de la otra, que complementan sus funciones para elaborar un producto específico. Sin embargo, existen actividades de mayor complejidad que los sistemas de producción en línea no son capaces de realizar bajo una metodología clásica de desarrollo de algoritmos [1], y se hace necesaria la implementación de técnicas de inteligencia artificial, que proporcionen el conocimiento necesario al sistema para la ejecución de tareas [2].

Los algoritmos de aprendizaje por refuerzo o *Reinforcement Learning* (RL), son un acercamiento computacional de aprendizaje desde la interacción, el cual está enfocado en aprendizajes mediante interacción orientados por objetivos [3]. Estos algoritmos brindan la capacidad para que los robots (o sistemas automatizados) puedan ejecutar tareas de mayor complejidad debiendo, tal como los humanos, aprender a ejecutarla para conseguir el objetivo. Sin embargo, estos algoritmos poseen un problema de rendimiento, los cuáles pueden derivar en una implementación costosa en la cantidad de tiempo empleado para alcanzar el aprendizaje [1].

En este trabajo se presenta un algoritmo como extensión del enfoque de RL que permite la interacción con otro agente utilizando un vector de estado continuo llamado aprendizaje de refuerzo interactivo o *Interactive Reinforcement Learning* (IRL). Hemos implementado un escenario simulado para probar los algoritmos propuestos. Aunque el enfoque de IRL acelera el proceso de aprendizaje de RL clásico, existe una motivación adicional para explorar otras características para mejorar la convergencia utilizando IRL mediante una representación de estado bajo dominio continuo.

Aprendizaje por refuerzo y enfoque interactivo

Aprendizaje por refuerzo o *Reinforcement learning* (RL) [3] que en sus inicios posee dos aristas principales, una hace referencia a la psicología animal definiendo el aprendizaje como el resultado del ensayo y error, y el segundo, a la solución del problema de control óptimo mediante funciones de valor y programación dinámica. Sin embargo, aparece luego una tercera arista que involucra métodos de diferenciación temporal, que da inicio al campo moderno de RL.

El método de RL es cualquier manera efectiva de resolver problemas de *Markov decision processes* (MDP), que poseen una estrecha relación con los problemas de control óptimo, donde la programación dinámica permite acercar el aprendizaje mediante la iteración sucesiva de aproximaciones a la respuesta correcta. La definición de RL está basado en el principio de *Law Effect* (Ley de Efecto) [4], el cual establece que: de muchas respuestas efectuadas frente a la misma situación, aquellas que conllevan una satisfacción mayor para el animal, permitirá un fortalecimiento entre la acción y la situación, lo que aumenta la probabilidad en que éste repita dicha acción para esa situación, así su contraparte, si su respuesta conlleva malestar, se debilita este lazo entre acción y situación, disminuyendo la probabilidad en que vuelva a ocurrir.

Basándose en el contexto de aprendizaje animal, el término refuerzo aparece posterior al principio de Thorndike de Ley de Efecto, que establece el fortalecimiento de patrones de conducta como el resultado de un animal que recibe un estímulo en una relación temporal con otro estímulo o respuesta, donde algunos psicólogos amplían el concepto con el debilitamiento de estos patrones, que generan un cambio en el comportamiento [3].

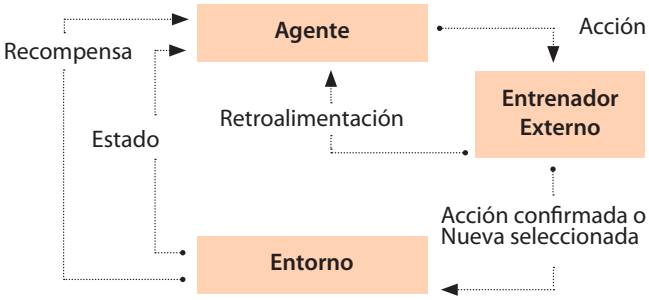


Figura 1

Diagrama de Aprendizaje por Refuerzo Interactivo mediante configuración de política

La idea de ensayo y error, bajo un acercamiento computacional, aparece por primera vez en un reporte de 1948, en el que Alan Turing describe un diseño de "pleasue-pain system" (sistema de placer dolor) que funciona bajo el principio de la Ley de Efecto: "Cuando se alcanza una configuración para la que la acción es indeterminada, se realiza una selección aleatoria de los datos faltantes y se realiza la entrada apropiada en la descripción, tentativamente, y se aplica. Cuando el estímulo del dolor ocurre, todas las entradas tentativas se cancelan y cuando ocurre un estímulo del placer, se hacen permanentes.", traducido [5].

Procesos de decisión Markoviano

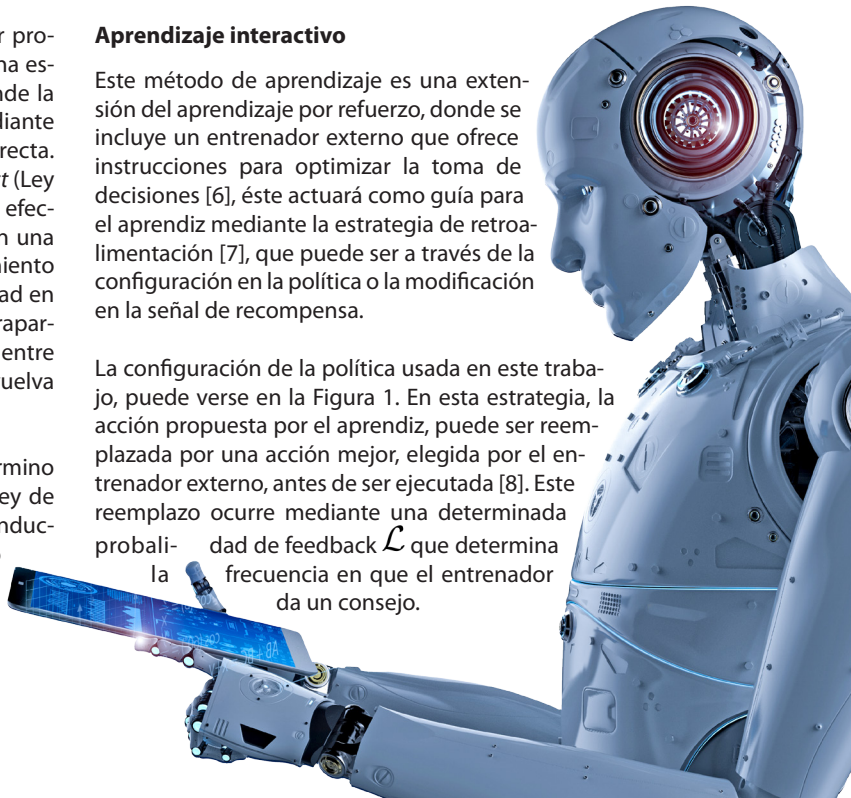
En *Markov decision processes* (MDP), el vector de estado obtenido desde el entorno debe proveer toda la información necesaria [3], como las mediciones de los sensores, pero puede contener mucho más que esto. Esta representación de estado puede ser una estructura compleja compuesta por una secuencia de sensaciones. A esto se le llama *Markov property*.

Con esta representación de estados se puede implementar una interfaz de agente-ambiente, en donde el actor o agente interactúa con su entorno [3] para llevar a cabo una tarea. sta interacción se efectúa mediante una función de transición que, para determinada acción ejecutada por el agente, modifica el entorno observable que a su vez le entrega una señal de recompensa. Con esto se establece que para resolver un MDP mediante RL, este debe poseer un espacio de estados, acciones, una función de transición y una función de recompensa.

Aprendizaje interactivo

Este método de aprendizaje es una extensión del aprendizaje por refuerzo, donde se incluye un entrenador externo que ofrece instrucciones para optimizar la toma de decisiones [6], éste actuará como guía para el aprendiz mediante la estrategia de retroalimentación [7], que puede ser a través de la configuración en la política o la modificación en la señal de recompensa.

La configuración de la política usada en este trabajo, puede verse en la Figura 1. En esta estrategia, la acción propuesta por el aprendiz, puede ser reemplazada por una acción mejor, elegida por el entrenador externo, antes de ser ejecutada [8]. Este reemplazo ocurre mediante una determinada probabilidad de feedback $\mathcal{L}$ que determina la frecuencia en que el entrenador da un consejo.



Aprendizaje por refuerzo continuo

En RL convencional, la mayoría de las veces, solo se consideran MDP con vectores de estados y acción bajo un dominio discreto [9], sin embargo, en muchas aplicaciones del mundo real la discretización del espacio no es muy útil, ya que la generalización se hace más difícil de experiencias pasadas y el aprendizaje se vuelve lento. Una manera para aprender desde un dominio continuo es Q-learning [10], que provee al agente la capacidad de aprender a actuar de manera óptima mediante la experiencia de las consecuencia de las acciones ejecutadas. Es un método incremental de programación dinámica que mediante sucesivas actualizaciones mejora la calidad de determinadas acciones para determinados estados.

Además, otro enfoque es un algoritmo capaz de manejar un espacio continuo de estados y acciones es el Continuos Actor-Critic Automaton (CACLA) que posee la habilidad de encontrar soluciones reales continuas, una mejor generalización de las propiedades y una rápida selección de acción como se muestra en [11].

Escenario experimental

El escenario está basado en el péndulo invertido (véase Figura 2), que requiere un controlador basado en retroalimentación de circuito cerrado, cuyo comportamiento debe balancear un poste conectado a un carro dirigido por motores. El movimiento del carro está restringido a un movimiento de un eje horizontal mediante una pista, y el poste es libre de moverse en este eje a través de un pivote [12].

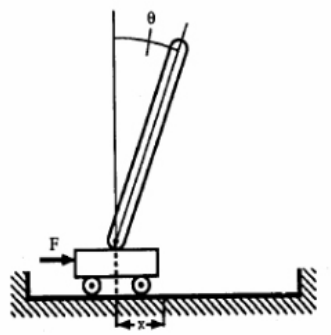


Figura 2

Ilustración del péndulo invertido, desde [12]

Péndulo invertido MDP

Para la implementación de los algoritmos de RL se definen los siguientes elementos según la definición de su MDP:

- **Estados:** El vector de estado posee una representación continua conformado por $< x, x', \theta, \theta' >$, correspondientes a la posición del carro respecto al centro de la pista, la velocidad del carro, el ángulo de inclinación del poste con el carro, y la velocidad angular del poste, respectivamente.

- **Acciones:** Las acciones poseen una representación discreta que corresponden a la dirección en que se debe mover el carro en la pista, izquierda que ejerce la fuerza en un sentido o derecha que ejerce una fuerza en sentido contrario.

- **Función de Transición**: Para obtener una variación del vector de estado, se debe aplicar una fuerza al carro de magnitud constante obtenida de una ecuación que posee las cuatro variables de entradas al sistema. La variación se obtiene mediante el uso de ecuaciones diferenciales propias del modelo físico del entorno [12].

- **Función de Recompensa**: Mientras el agente mantenga el poste en sentido vertical se le entregará una recompensa igual a 1, y si a éste se le cae, o traspasa los límites de la pista, la recompensa es igual a 0.

Diseño e implementación de los agentes

Para el entorno mencionado anteriormente, una simulación es configurada en donde, los agentes implementados deben ser capaces de obtener una alta recompensa durante el progreso del episodio. Para este problema del péndulo invertido, se implementan dos agentes, uno capaz de interactuar mediante un vector de estados discreto, y otro mediante un vector de estados continuo.

Para los diferentes agentes, es necesario diseñar la manera en que se decide la acción a efectuar en su entorno donde se requiere un balance óptimo entre explotación y exploración del espacio de acciones, para lo cual la información disponible para esta decisión depende de las acciones tomadas con anterioridad. Con esto el agente debe explorar el espacio de acción compensando las buenas acciones ya exploradas con otras que nunca intentó [2]. Para la solución de esta compensación entre explotación y exploración, se implementa el método ϵ-greedy, que posee un factor de exploración ϵ seleccionado aleatoriamente desde una distribución normal [1], esta estrategia en general converge en una mayor recompensa sin quedar atrapado en un mínimo local [3].

Para un primer acercamiento a este algoritmo se implementa el método off-policy Q-learning [13], este método de aprendizaje le permite al agente la capacidad de actuar de manera óptima, mediante la experiencia de la consecuencia de las acciones sin la necesidad de construir un mapa del dominio Markoviano.

Enfoque interactivo

Un primer acercamiento para IRL es mediante la implementación de un agente externo, el entrenador, que retroalimenta con una selección de acción según su configuración de política, a un segundo agente dada una determinada probabilidad $\mathcal{L}$. Dicho lo anterior, el desarrollo del algoritmo extiende el agente RL, modificando la forma en que selecciona la acción involucrando la política del entrenador como se muestra en el algoritmo 1. Esta implementación se efectúa para los agentes de los dos entornos en sus respectivas representaciones.

Algorithm 1 Agente IRL

function SELECCIONARACCION (S_t)
  if valor Aleatorio < $\mathcal{L}$ **Then**
    $a_t \leftarrow$ función de valor del agente entrenador para s_t
  else
    $at \leftarrow$ selecciona acción según política propia para s_t
  return a_t

Representación discreta

Un primer acercamiento para la implementación del algoritmo para el péndulo invertido es abordar su representación discreta, para simplificar la complejidad en encontrar una política óptima de aprendizaje. Dado que el vector de estado para el sistema corresponde a valores continuos, se debe discretizar en rangos con límites superior e inferior. El enfoque BOXES [14], divide el espacio de estados del sistema en regiones discretas llamadas particiones o boxes que reduce la complejidad del problema, con esto el método actualiza la acción para cada partición (box).

En la implementación de Q-learning [13] para este tipo de representación, se genera una matriz Q que almacena la recompensa obtenida para cada par de estado acción, permitiendo la simplificación en la estimación de la función de valor, validando que "tan buena" es la acción tomada dado un determinado estado, esta definición de "bueno" está basado en términos de la recompensa o retorno esperado para una determinada política [3], la implementación del agente discreto se puede observar en el algoritmo 2.

Algorithm 2 Agente Discreto

1. Inicializar matriz Q con valor uniforme [-1,1]
2. **for** Episodio = 1, M **do**
3. Observar s_0 desde el entorno
4. Discretización de s_0 desde el entorno
5. **while** s_t no sea terminal **do**
6. Seleccionar acción at según política e-greedy
7. Observar s_{t+1}, r_t
8. Discretización de s_{t+1} con BOXES
9. Actualizar matriz Q para el par s_t, a_t
10. $Q(s_t, a_t) = Q(s_t, a_t) + \alpha\,[r_t + \gamma max Q(s_{t+1}, a) - Q(s_t, a_t)]$
11. $s_t \leftarrow s_{t+1}$
12. Reducir valores de ϵ y α-

Representación continua

Dado que la representación continua pertenece al conjunto de los números reales, es impracticable almacenar en memoria todos los posibles pares estado-acción, es por esto que para la solución en espacios continuos se utiliza una función de aproximación [9], la cual permite estimar el valor Q a partir del vector de estado sin necesidad de almacenarlo como un valor dentro de la memoria, sino como una función Q.

Para la implementación en esta representación, se utilizan dos redes neuronales del tipo perceptrón multicapa, una se utilizará para el entrenamiento del agente y la segunda para calcular el valor esperado para la función Q. Además se implementa una optimización de Temporal-Difference learning en donde se almacenan en memoria una tupla de < $estado_t$, $acción_t$, $recompensa_t$, $estado_{t+1}$ > la cual se utilizará para la obtención del conjunto para el entrenamiento de las redes neuronales [3].

Cada red neuronal posee una arquitectura de 4x24x24x2, la cual posee 4 entradas correspondientes al vector de estado, 2 capas ocultas con 24 neuronas en cada una, y 2 salidas que representan las dos acciones que puede seleccionar el agente. Los pesos iniciales son definidos aleatoriamente desde una distribución normal y el control del entrenamiento corresponde al error cuadrático medio. Este método corresponde a una implementación incremental de Q-learning [10] implementado en el algoritmo 3.

Algorithm 3 Agente Continuo

1. Inicializar memoria D con largo N
2. Inicializar función Q con pesos uniformes [-1,1]
3. Inicializar aproximador Q con pesos uniformes [-1,1]
4. **for** episodio = 1, M **do**
5. Observar s_0 desde el entorno
6. **while** s_t no sea terminal **do**
7. Seleccionar acción a_t según política e-greedy
8. Observar s_t+1, r_t
9. Almacenar en D la tupla (s_t, a_t, r_t, s_{t+1}, esTerminal)
10. Actualizar función Q desde D
11. $s_t \leftarrow s_{t+1}$
12. Asignar pesos de función Q al aproximador Q
13. Reducir valor de ϵ

Comparación y análisis de resultados

Con el fin de conseguir una correcta comparación de resultados, 50 agentes fueron entrenados con las mismas configuraciones para la convergencia de aprendizaje.

Para la simulación del entorno, se usó la librería OpenAI Gym [15] para instanciar *CartPole-v1* que define la duración máxima de cada episodio en 500 unidades de tiempo, siendo este el valor máximo que se puede obtener como recompensa; un valor mínimo de 475 de recompensa indica que la tarea se ejecutó de manera exitosa para el episodio durante el entrenamiento del agente. Además se considera que la tarea está aprendida si completa satisfactoriamente la ejecución en 50 episodios consecutivos.

Resultados representación discreta

Los parámetros utilizados durante el entrenamiento están en la Tabla 1. Los parámetros, con excepción del factor de descuento, disminuyen entre episodios desde un valor inicial hasta un mínimo fijo. La variación de los parámetros está dada por la ecuación.

Parámetro	Valor
Factor de descuento ()	0.99
Tasa de exploración ()	[1, 0.01]
Tasa de aprendizaje ()	[0.5, 0.1]

Tabla 1

Parámetros para el agente con representación discreta.

$$MAX(param_{max}, MIN(param_{min}, param - \log_{10}(\frac{\text{ep}+1}{25}))) \quad (1)$$

Los resultados obtenidos con la configuración anterior (Figura 3) muestran que la convergencia del aprendizaje desde el episodio 150 en adelante, permite al agente mejorar su rendimiento en los siguientes episodios, siendo capaz de obtener la máxima recompensa desde el episodio 247, consiguiendo mantener el poste vertical durante más de 50 episodios consecutivos. Para esta representación, el enfoque interactivo del agente muestra que la convergencia del aprendizaje se alcanza a partir del episodio 100 en adelante, sin embargo, el agente logra ejecutar la tarea satisfactoriamente desde el episodio 232, desde donde es capaz de mantener el poste vertical por más de 50 episodios consecutivos.

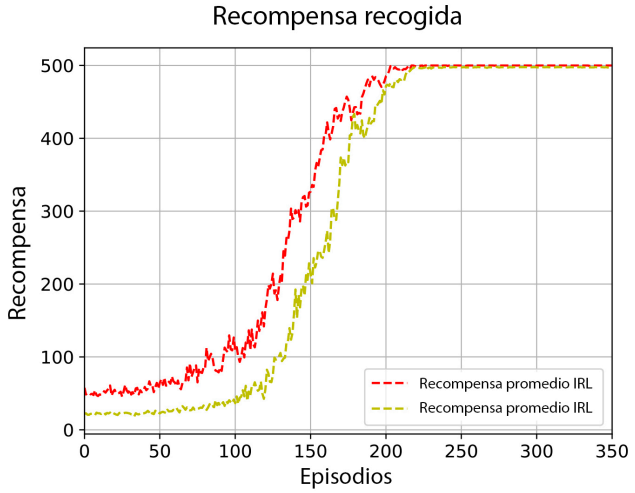


Figura 3

Resultado del entrenamiento de 50 agentes para el entorno CartPole-v1 con discretización del vector de estado con BOXES [14], y una probabilidad de feedback $\mathcal{L}$ de 0.3

Resultados representación continua

Para el algoritmo con representación continua, sólo el parámetro la tasa de exploración disminuye con el tiempo según la ecuación (2), y son mostrados en la Tabla 2.

Parámetro	Valor
Factor de descuento ()	0.99
Tasa de exploración ()	[1, 0.01]
Tasa de aprendizaje ()	[0.00.1]

Tabla 2

Parámetros para el agente con representación continua.

$$param_{t+1} = param_t * 0.999$$

Como se muestra en la figura 4, para esta representación el agente autónomo de RL converge después del episodio 55 y alcanza una recompensa máxima de 460 en el episodio 215, siendo incapaz de mantenerse estable para los siguientes episodios. Sin embargo, bajo el enfoque interactivo, hay una convergencia más rápida de aprendizaje que comienza en el episodio 40 y alcanza un valor máximo de recompensa de 450 en el episodio 155.

Los resultados anteriores indican que aunque el enfoque interactivo no excede los valores de recompensa obtenidos durante el entrenamiento, este enfoque permite una convergencia más rápida del aprendizaje. Además, el uso de un asesor también beneficia la convergencia, sin embargo, es crucial contar con un buen consejo para poder explotarlo, de lo contrario, un consejo inconsistente puede ser perjudicial para el proceso de aprendizaje.

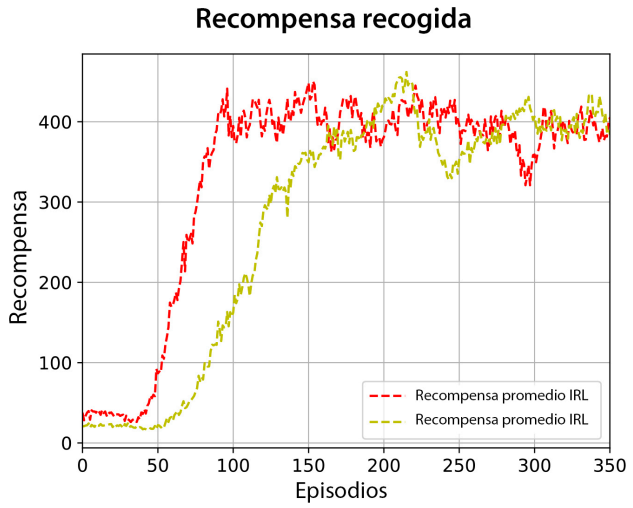


Figura 4

Resultados del entrenamiento de 50 agentes para entorno CartPole-v1 con representación continua y una probabilidad de feedback $\mathcal{L}$ de 0.3.

Conclusiones

En este trabajo hemos demostrado que un agente interactivo con un espacio de estado continuo es capaz de obtener una convergencia de aprendizaje más rápida y conseguirla en un menor número de episodios en comparación con el agente autónomo para el mismo problema. En este enfoque interactivo, al utilizar un espacio de estado continuo, el agente logra una convergencia de aprendizaje aún más rápida.

Aunque la representación discreta logra un rendimiento estable en términos de recompensa, esto puede no ser implementado en todos los escenarios del mundo real.

Además, el enfoque interactivo para el agente de aprendizaje por refuerzo permite una mejor convergencia con respecto al enfoque autónomo utilizando la misma probabilidad de retroalimentación de la política.

Para trabajos futuros se contempla la implementación del agente IRL para espacios de acción continuo. Por ejemplo, usando el algoritmo Continuos Actor-Critic Learning Automaton [9], se espera conseguir mejores resultados en términos de tiempo necesario para la convergencia de aprendizaje.

Agradecimientos

Los autores agradecen el apoyo parcial de la Universidad Central de Chile bajo el proyecto de investigación CIP2017030.

Bibliografía

[1] F. Cruz, S. Magg, C. Weber, and S. Wermter, "Improving reinforcement learning with interactive feedback and affordances," in 4th International Conference on Development and Learning and on Epigenetic Robotics, pp. 165-170, Oct 2014.

[2] S. Marsland, Machine Learning: An Algorithmic Perspective. New York: Chapman and Hall/CRC, 1st ed., apr 2009.

[3] R. S. Sutton and A. G. Barto, Introduction to Reinforcement Learning. Cambridge, MA, USA: MIT Press, 1st ed., 1998.

[4] E. L. Thorndike, Animal intelligence; experimental studies,. New York,The Macmillan company,, 1911.

[5] A. M. Turing, "Turing, intelligent machinery, a heretical theory," Alan M. Turing, pp. 128-134, 1948.

[6] F. Cruz, S. Magg, Y. Nagai, and S. Wermter, "Improving interactive reinforcement learning: What makes a good teacher?," Connection Science, vol. 30, no. 3, pp. 306-325, 2018.

[7] A. L. Thomaz and C. Breazeal, "Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance," in Proceedings of the 21st National Conference on Artificial Intelligence - Volume 1, AAAI'06, pp. 1000-1005, AAAI Press, 2006.

[8] T. Cederborg, I. Grover, C. L. Isbell, and A. L. Thomaz, "Policy shaping with human teachers," in Proceedings of the 24th International Conference on Artificial Intelligence, IJCAI'15, pp. 3366-3372, AAAI Press, 2015.

[9] H. van Hasselt and M. A. Wiering, "Reinforcement learning in continuous action spaces," in 2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning, pp. 272-279, April 2007.

[10] C. J. Watkins and P. Dayan, "Q-learning," Machine Learning, vol. 8, pp. 279-292, May 1992.

[11] J. Zhong, C. Weber, and S. Wermter, "A predictive network architecture for a robust and smooth robot docking behavior," Paladyn, vol. 3, pp. 172-180, 12 2012.

[12] J. Brownlee, "The pole balancing problem: A benchmark control theory problem," Tech. Rep. 7-01, Swinburne University of Technology, Melbourne, Victoria, Australia, jul 2005.

[13] C. J. Watkins, Learning from Delayed Rewards. PhD thesis, King's College, Cambridge, UK, May 1989.

[14] D. Miche and R. A. Chambers, "9 boxes: An experiment in adaptive control," 1968.

[15] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," ArXiv e-prints, June 2016.