Aproximación Funcional en Aprendizaje por Refuerzo Multi-Objetivo

Manuela Ruiz-Montiel, Lawrence Mandow, and José-Luis Pérez-de-la-Cruz

Universidad de Málaga. Campus de Excelencia Internacional Andalucía Tech
{mruiz,lawrence,perez}@lcc.uma.es

Resumen En este trabajo describimos y comparamos dos técnicas para combinar métodos de aproximación funcional y de escalarización, con el objetivo de resolver problemas de aprendizaje por refuerzo con espacios de estados de tamaño elevado y con múltiples objetivos. Se analizan los resultados de ambas técnicas al resolver un problema de aprendizaje[1] .

Keywords: Aprendizaje por Refuerzo, Aproximación Funcional, Optimización Multi-Objetivo, Escalarización

1. Introducción

El aprendizaje por refuerzo [1] (AR) es un área del aprendizaje automático encargada de aprender qué acciones elegir en un entorno determinado, con el objetivo de maximizar una recompensa acumulada a largo plazo. El aprendizaje sucede mediante la interacción con el entorno, recibiendo recompensas positivas o negativas tras tomar ciertas acciones.

Los algoritmos de AR funcionan aprendiendo el valor a largo plazo de escoger una determinada acción, es decir, la recompensa acumulada que podemos esperar. Por ejemplo, las técnicas de diferencias temporales aprenden un valor para cada par (estado, acción) del entorno. La opción más directa es almacenar los distintos valores en una tabla. Sin embargo, en muchos problemas reales el tamaño del espacio de estados y acciones es tan elevado que esta opción deja de ser factible. Las técnicas de aproximación de valores intentan aliviar esta situación, permitiendo usar una función en lugar de una tabla.

Por otra parte, la mayoría del trabajo existente en el ámbito del AR trata con recompensas escalares. Sin embargo, numerosos problemas se formulan mejor en términos de múltiples objetivos, donde las recompensas son vectores y sus componentes representan objetivos distintos y posiblemente en conflicto. Una opción es escalarizar la recompensa, de manera que se puedan aplicar las técnicas tradicionales de AR para un único objetivo.

En este trabajo describimos y comparamos dos mecanismos para integrar la aproximación funcional y la escalarización en un mismo algoritmo de AR. El

[1] La presentación de este trabajo está subvencionada por el Plan Propio de Investigación de la Universidad de Málaga - Campus de Excelencia Internacional Andalucía Tech.

primer mecanismo es una extensión directa de una técnica existente de AR multiobjetivo, en la cual se sustituye la tabla por una función de aproximación lineal. El segundo mecanismo emplea un enfoque distinto, ya que en lugar de escalarizar las recompensas, escalariza los valores aproximados. Hasta donde sabemos, esta segunda técnica no ha sido previamente descrita en la literatura.

Este artículo se estructura como sigue: en la Sección 2 se recogen algunos conceptos necesarios relativos al AR, a la aproximación funcional y al AR multiobjetivo. A continuación, en la Sección 3 se describen dos técnicas para integrar la aproximación funcional y la escalarización. En la Sección 4 se muestran dos conjuntos de experimentos destinados a resolver un problema sencillo mediante los dos mecanismos descritos anteriormente. En la Sección 5 se analizan los resultados arrojados por los experimentos y, finalmente, en la Sección 6 se exponen las conclusiones de este trabajo y las posibles líneas de trabajo futuro.

2. Antecedentes

2.1. Aprendizaje por Refuerzo

Siguiendo la notación de Sutton & Barto [1], los métodos de AR solucionan procesos de decisión en los cuales la transición de un estado s a otro estado s' mediante la acción a conlleva la obtención de una recompensa r. La solución consiste en una política que decide qué acción tomar en cada estado. La ejecución de una política a partir de un estado s_0 origina una secuencia de estados $s_0, s_1, ..., s_i, ...$ Para cada estado s_t de esta secuencia (siendo t un instante dado de tiempo), la recompensa acumulada que puede obtenerse viene dada por la expresión $R_t = \sum_{k=0}^{\text{ínf}} \gamma^k r_{t+k+1}$, donde γ es un factor de descuento que, si es menor que 1, le resta importancia a las recompensas obtenidas en el futuro. Los algoritmos de aprendizaje por refuerzo intentan encontrar la política que maximice esta recompensa acumulada.

Los métodos de diferencias temporales, y concretamente el algoritmo Q-learning, constituyen una de las técnicas más populares en el ámbito del AR. En Q-learning, cada par (s, a) se asocia con un valor $Q(s, a)$, y la política queda determinada por el siguiente juicio de valor: en el estado s, escoger la acción a tal que el valor $Q(s, a)$ es máximo. El enfoque más directo consiste en almacenar los valores $Q(s, a)$ en una tabla. Al realizar la transición desde un estado s a un estado s' a traves de la acción a, el valor $Q(s, a)$ se actualiza mediante la expresión $Q(s, a) \leftarrow Q(s, a) + \alpha\delta$, donde $\alpha \in [0, 1]$ es la tasa de aprendizaje y δ es la denominada diferencia temporal. Esta diferencia se define como $\delta = r + \gamma max_{a'} Q(s', a') - Q(s, a)$. En cada paso del episodio se eligen los pares (s, a) a actualizar mediante una estrategia de explotación-exploración. Con una probabilidad ϵ el algoritmo explora, es decir, elige la acción a tomar de manera aleatoria para intentar mejorar la política existente probando nuevas opciones. Por el contrario, con una probabilidad $1 - \epsilon$ se utiliza política aprendida hasta el momento, con el fin de evaluarla.

2.2. Aproximación Funcional

En muchos casos prácticos, el número de pares (s, a) es tan elevado que no es factible almacenarlos en una tabla. Una posible solución es usar un aproximador para aprender los valores $Q(s, a)$. Para ello es necesario seleccionar un conjunto de características o rasgos relevantes f_i para describir cada par (s, a) y poder definir los valores $Q(s, a)$ como una función de dichos rasgos. En este trabajo hemos utilizado una función lineal de los rasgos. Más concretamente, postulamos una función $Q(s, a) = \theta_1 f_1(s, a) + \cdots + \theta_n f_n(s, a)$ donde $f_i(s, a)$ es el rasgo i-ésimo del estado s' que surge al aplicar la acción a sobre el estado s, y θ_i es el i-ésimo coeficiente de la función lineal Q. Así, lo que el algoritmo ha de aprender ahora es cuáles son estos coeficientes, utilizando la regla de aprendizaje $\theta_i \leftarrow \theta_i + \alpha\delta f_i(s, a)$.

$Q(\lambda)$ es una generalización de Q-learning en la cual se puede controlar el nivel de *bootstrapping*, es decir, la utilización de predicciones anteriores para actualizar los valores. Si $\lambda = 0$, el método es equivalente a Q-learning puro. Valores más altos de λ acercan el método a la técnica de Monte Carlo, en la cual es necesario esperar hasta que la recompensa acumulada está disponible para actualizar los valores. Dado que el uso de *bootstrapping* puede afectar a la convergencia de la función $Q(s, a)$ cuando se usan aproximadores, es deseable contar con un mecanismo para regularlo. Por tanto, $Q(\lambda)$ es más apropiado para utilizar este tipo de aproximadores lineales que Q-learning puro. En este trabajo hemos empleado el algoritmo $Q(\lambda)$ con un aproximador lineal [1] (p. 213).

2.3. Aprendizaje por Refuerzo Multi-Objetivo

En numerosas ocasiones, la recompensa r que se obtiene al realizar la transición entre dos estados se formaliza mejor en términos de múltiples objetivos, es decir, mediante un vector $\boldsymbol{r}$.

Existen dos enfoques para tratar problemas con múltiples objetivos mediante técnicas de aprendizaje por refuerzo: (1) política única y (2) políticas múltiples [2]. En el primer paradigma se transforma la recompensa en un valor escalar a partir de ciertas preferencias conocidas a priori. De este modo se pueden seguir utilizando las técnicas de aprendizaje por refuerzo para un único objetivo. En el segundo paradigma se intentan aprender todas las políticas que dan lugar a recompensas acumuladas óptimas. Incluso en este caso se pueden seguir utilizando mecanismos para escalarizar la recompensa, resolviendo el problema repetidas veces con diferentes configuraciones de preferencias o pesos para cada objetivo [3], [4]. Concretamente este es el enfoque que hemos seguido en este trabajo, utilizando una función de escalarización lineal.

Hasta donde sabemos, no existen trabajos que estudien la interacción entre estas técnicas de escalarización lineal y la utilización de métodos de aproximación de funciones. En la próxima sección proponemos dos métodos para combinar dichas técnicas.

3. Aproximación y Escalarización Lineal en $Q(\lambda)$ Multi-Objetivo

En esta sección describimos dos métodos para combinar las técnicas de aproximación de funciones y escalarización lineal en el algoritmo $Q(\lambda)$. El primer método no introduce ninguna novedad, pues se trata de utilizar el algoritmo $Q(\lambda)$ repetidas veces con diferentes configuraciones de preferencias, escalarizando la recompensa y utilizando un mecanismo de aproximación lineal, manejando únicamente una función $Q(s, a)$. El segundo método sí introduce una nueva técnica para combinar la aproximación y la escalarización, ya que aprende una función lineal para cada objetivo, manteniendo así múltiples funciones $Q(s, a)$, y luego escalariza dichas funciones en lugar de agregar directamente la recompensa.

3.1. Aproximación de Valores Escalarizados Linealmente

Supongamos que utilizamos una función lineal para escalarizar la recompensa $\boldsymbol{r} = (r_1, \cdots, r_m)$. Es decir, $r = w_1 r_1 + \cdots + w_m r_m$. Podemos utilizar distintas configuraciones de preferencias $\boldsymbol{w}$ para dar lugar a distintas recompensas escalares. Con cada configuración $\boldsymbol{w}$ podemos lanzar un proceso distinto de aprendizaje.

Para realizar la aproximación necesitamos también un conjunto de n rasgos que definan cada par (s, a), de forma que el algoritmo aprenderá un conjunto de n coeficientes para la función de aproximación lineal. En el Cuadro 1 se describe el pseudocódigo para esta técnica, teniendo en cuenta la escalarización lineal de la recompensa y la utilización de rasgos continuos normalizados entre 0 y 1.

3.2. Escalarización Lineal de Valores Aproximados

Supongamos ahora que utilizamos una función lineal para escalarizar un vector formado por diferentes funciones de valor $\boldsymbol{Q}(s, a) = (Q_1(s, a), \cdots, Q_m(s, a))$.

1. Sean $\boldsymbol{\theta}$ y $\boldsymbol{e}$ vectores con n componentes (una por cada rasgo)
2. Inicializar $\boldsymbol{\theta}$ arbitrariamente, e.g., $\boldsymbol{\theta} = \boldsymbol{0}$ 3. Repetir (para cada episodio):
4. $\quad \boldsymbol{e} = \boldsymbol{0}$
5. $\quad s \leftarrow$ estado inicial del episodio
6. $\quad$ Repetir (para cada paso del episodio):
7. $\quad\quad$ Para todo $a \in A(s)$:
8. $\quad\quad\quad Q(s, a) \leftarrow \sum_{i=1}^{n} f_i(s, a)\theta_i$
9. $\quad\quad a \leftarrow argmax_a Q(s, a)$ con prob. $1 - \epsilon$, e.o.c. $a \leftarrow$ acción aleatoria $\in A(s)$
10. $\quad\quad$ Tomar la acción a, observar $r = w_1 r_1 + \cdots + w_m r_m$ y siguiente estado s'
11. $\quad\quad \delta \leftarrow r - Q(s, a)$
12. $\quad\quad$ Para todo $i \in 1, \cdots, n, e_i \leftarrow e_i + f_i(s, a)$ (acumulación de trazas)
13. $\quad\quad$ Si s' es final, entonces $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\delta\boldsymbol{e}$, ir al siguiente episodio
14. $\quad\quad$ Para todo $a' \in A(s')$:
15. $\quad\quad\quad Q(s', a') \leftarrow \sum_{i=1}^{n} f_i(s', a')\theta_i$
16. $\quad\quad \delta \leftarrow \delta + \gamma max_{a' \in A(s')} Q(s', a')$
17. $\quad\quad \boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\delta\boldsymbol{e}$
18. $\quad\quad \boldsymbol{e} \leftarrow \gamma\lambda\boldsymbol{e}$
19. $\quad\quad s \leftarrow s'$

Cuadro 1. Aproximación de valores escalarizados mediante $Q(\lambda)$ lineal

Es decir, tenemos que $Q(s,a) = w_1 Q_1(s,a) + \cdots + w_m Q_m(s,a)$, donde el vector $\boldsymbol{w}$ representa los pesos o preferencias para cada objetivo, al igual que en la sección anterior, aunque ahora no utilizaremos este vector para escalarizar la recompensa sino los valores $Q_i(s,a)$.

La diferencia fundamental entre este enfoque y el anterior es que ahora se aprende una función $Q_i(s,a)$ para cada objetivo, y la función $Q(s,a)$ que finalmente utilizaremos como política vendrá dada por la escalarización lineal del vector de funciones $\boldsymbol{Q}(s,a)$. Así, necesitaremos m conjuntos de rasgos para cada par (s,a), de manera que el algoritmo aprenderá a su vez m conjuntos de coeficientes para las funciones de aproximación lineal. En esta configuración podemos utilizar distintos conjuntos de rasgos para cada objetivo, aunque por simplicidad supondremos que todos los conjuntos de rasgos son de tamaño n, y por tanto todos los conjuntos de coeficientes también son de este tamaño. En el Cuadro 2 se describe el pseudocódigo para esta técnica, utilizando rasgos continuos normalizados entre 0 y 1.

4. Experimentos

En esta sección utilizaremos las dos técnicas descritas anteriormente para resolver un problema de optimización multiobjetivo que puede ser formalizado en términos de estados, acciones y recompensas. Concretamente se trata de un problema geométrico en el que se pretenden generar formas bidimensionales de 9 bloques cuadrados mediante una regla aditiva. Esta regla ubica un bloque en un hueco libre junto a algún bloque existente (ver Figura 1).

En este problema, los estados son las distintas formas que pueden generarse mediante la regla. Las acciones disponibles para cada estado son las distintas

1. Sean $\boldsymbol{\theta}_1, \cdots \boldsymbol{\theta}_m$ m vectores con n componentes cada uno
2. Sean $\boldsymbol{e}_1, \cdots \boldsymbol{e}_m$ m vectores con n componentes cada uno
3. Inicializar $\boldsymbol{\theta}_i$ arbitrariamente, e.g., $\boldsymbol{\theta}_i = \mathbf{0}$
4. Repetir (para cada episodio):
5. Para todo $i \in 1, \cdots, m, \boldsymbol{e}_i = \mathbf{0}$
6. $s \leftarrow$ estado inicial del episodio
7. Repetir (para cada paso del episodio):
8. Para todo $a \in A(s)$:
9. $Q(s,a) \leftarrow \sum_{i=1}^{m} w_i Q_i(s,a)$
10. donde $Q_i(s,a) \leftarrow \sum_{j=1}^{n} f_{ij}(s,a)\theta_{ij}$
11. $a \leftarrow argmax_a Q(s,a)$ con prob. $1-\epsilon$, e.o.c. $a \leftarrow$ acción aleatoria $\in A(s)$
12. Tomar la acción a, observar $\boldsymbol{r} = (r_1, \cdots, r_m)$ y siguiente estado s'
13. Para todo $i \in 1, \cdots, m, \delta_i \leftarrow r_i - Q_i(s,a)$
14. Para todo $i \in 1, \cdots, m, \boldsymbol{e}_i \leftarrow \boldsymbol{e}_i + \boldsymbol{f}_i(s,a)$ (acumulación de trazas)
15. Si s' es final, entonces $\forall i \in 1, \cdots, m, \boldsymbol{\theta}_i \leftarrow \boldsymbol{\theta}_i + \alpha\delta_i \boldsymbol{e}_i$, ir al siguiente episodio
16. Para todo $a' \in A(s')$:
17. $Q(s',a') \leftarrow \sum_{i=1}^{m} w_i Q_i(s',a')$
18. donde $Q_i(s',a') \leftarrow \sum_{j=1}^{n} f_{ij}(s',a')\theta_{ij}$
19. $a' \leftarrow argmax_{a'} Q(s',a')$
20. Para todo $i \in 1, \cdots, m, \delta_i \leftarrow \delta_i + \gamma Q_i(s',a')$
21. Para todo $i \in 1, \cdots, m, \boldsymbol{\theta}_i \leftarrow \boldsymbol{\theta}_i + \alpha\delta\boldsymbol{e}_i$
22. Para todo $i \in 1, \cdots, m, \boldsymbol{e}_i \leftarrow \gamma\lambda\boldsymbol{e}_i$
23. $s \leftarrow s'$

Cuadro 2. Escalarización de valores aproximados mediante $Q(\lambda)$ lineal

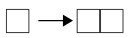

Figura 1. Regla geométrica utilizada en el ejemplo.

posibilidades de aplicación de la regla. Por ejemplo, en el estado inicial, formado por un único bloque, tenemos cuatro posibles acciones: ubicar el nuevo bloque arriba, abajo, a la derecha o a la izquierda. Un estado final es toda aquella forma que tenga nueve bloques.

Tendremos en cuenta dos objetivos: maximizar la compacidad de las formas y maximizar su perímetro. La recompensa es un vector nulo en todos los estados no finales. En los estados finales, la recompensa es un vector de dos componentes (r_1, r_2) dados por las siguientes expresiones:

$$r_1 = \begin{cases} c_{norm} & \text{si } c_{norm} \leq 0{,}6 \\ 0{,}6 + \frac{c_{norm}-0{,}6}{2} & \text{e.o.c.} \end{cases} \quad (1)$$

$$r_2 = \begin{cases} p_{norm} & \text{si } p_{norm} \leq 0{,}8 \\ 0{,}8 + \frac{c_{norm}-0{,}8}{2} & \text{e.o.c.} \end{cases} \quad (2)$$

Donde c_{norm} es la compacidad normalizada y se calcula dividiendo la compacidad $c = 9/p^2$ (donde p es el perímetro) entre la compacidad máxima, que es $c_{max} = 9/144$, es decir, $c_{norm} = 144/p^2$. p_{norm} es el perímetro normalizado y se calcula dividiendo p entre el perímetro máximo, es decir, $p_{norm} = p/20$.

Utilizando esta recompensa se obtienen tres soluciones no dominadas que pueden ser obtenidas mediante tres configuraciones distintas de pesos. Dos de las tres soluciones representan más de una forma (las correspondientes a maximizar únicamente el perímetro y a maximizar tanto el perímetro como la compacidad), y una de ellas representa únicamente a una forma (la correspondiente a maximizar la compacidad). La frontera y las distintas soluciones asociadas a cada punto no dominado, así como los pesos que dan lugar a su obtención junto con las recompensas escalares asociadas, se pueden visualizar en la Figura 2.

Para poder aplicar la aproximación funcional es necesario definir un conjunto de rasgos que caractericen los distintos estados de manera apropiada. Hemos considerado cinco rasgos normalizados entre 0 y 1.

- $f_1(s, a)$: número de bloques cuadrados con un vecino[2] en la forma que surge al aplicar la acción a al estado s, dividido entre el número final de bloques que ha de tener la forma (9)
- $f_2(s, a)$: análogo al rasgo $f_1(s, a)$, con dos vecinos
- $f_3(s, a)$: análogo al rasgo $f_1(s, a)$, con tres vecinos
- $f_4(s, a)$: análogo al rasgo $f_1(s, a)$, con cuatro vecinos
- $f_5(s, a)$: número de bloques que contiene la forma, dividido entre el número final de bloques (9)

[2] Para un bloque dado, los vecinos considerados son aquellos que tienen un lado en común con dicho bloque, es decir, no se consideran los vecinos ubicados en las diagonales.

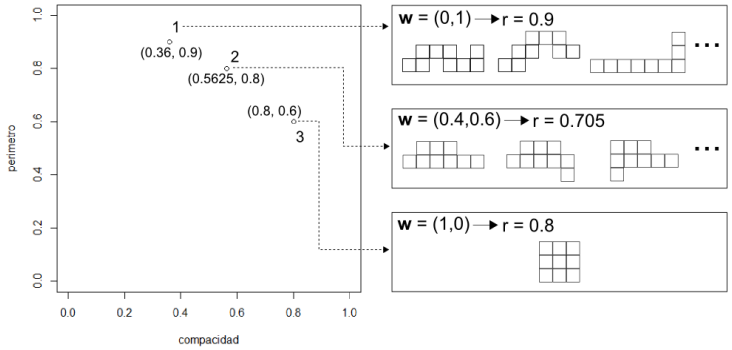

Figura 2. Frontera de Pareto y soluciones geométricas del problema considerado.

Se han llevado a cabo 100 ejecuciones de cada técnica para cada combinación de pesos, con 5000 episodios de aprendizaje y utilizando la siguiente parametrización: $\alpha = 0{,}1, \epsilon = 0{,}3, \gamma = 1, \lambda = 0{,}5$. Se han registrado dos medidas:

1. Rendimiento *online*: recompensas obtenidas en determinados episodios del proceso de aprendizaje (concretamente cada 100 episodios)
2. Rendimiento de las políticas: cada 100 episodios se ha utilizado la política aprendida hasta el momento para generar 100 formas y evaluar la recompensa media de dichas formas

Para representar ambos rendimientos gráficamente, las recompensas se han escalarizado mediante la combinación de pesos que corresponda en cada caso. Además, en cada gráfica se incluye una línea punteada horizontal que corresponde a la recompensa escalarizada que se espera obtener.

En la Figura 3 se pueden observar los distintos rendimientos para cada configuración de pesos de la técnica descrita en la Sección 3.1, es decir, la que aproxima valores escalarizados linealmente. En la Figura 4 se pueden observar los rendimientos para la técnica descrita en la Sección 3.2, es decir, la que escalariza linealmente valores aproximados.

5. Discusión

A la luz de las gráficas ilustradas en las figuras 3 y 4, los rendimientos de las técnicas descritas son equivalentes para el ejemplo considerado. Si observamos las gráficas de la columna de la derecha, correspondientes al rendimiento de las políticas, para las configuraciones de pesos $\boldsymbol{w} = (0, 1)$ y $\boldsymbol{w} = (1, 0)$, las soluciones correspondientes se alcanzan dentro de los 5000 episodios de aprendizaje. Para la configuración $\boldsymbol{w} = (0{,}4, 0{,}6)$ podemos observar que serían necesarios más episodios para alcanzar la solución óptima. Igualmente, en ambos casos el rendimiento online es equivalente, siendo inferior al rendimiento de las políticas por la influencia del factor de exploración ϵ.

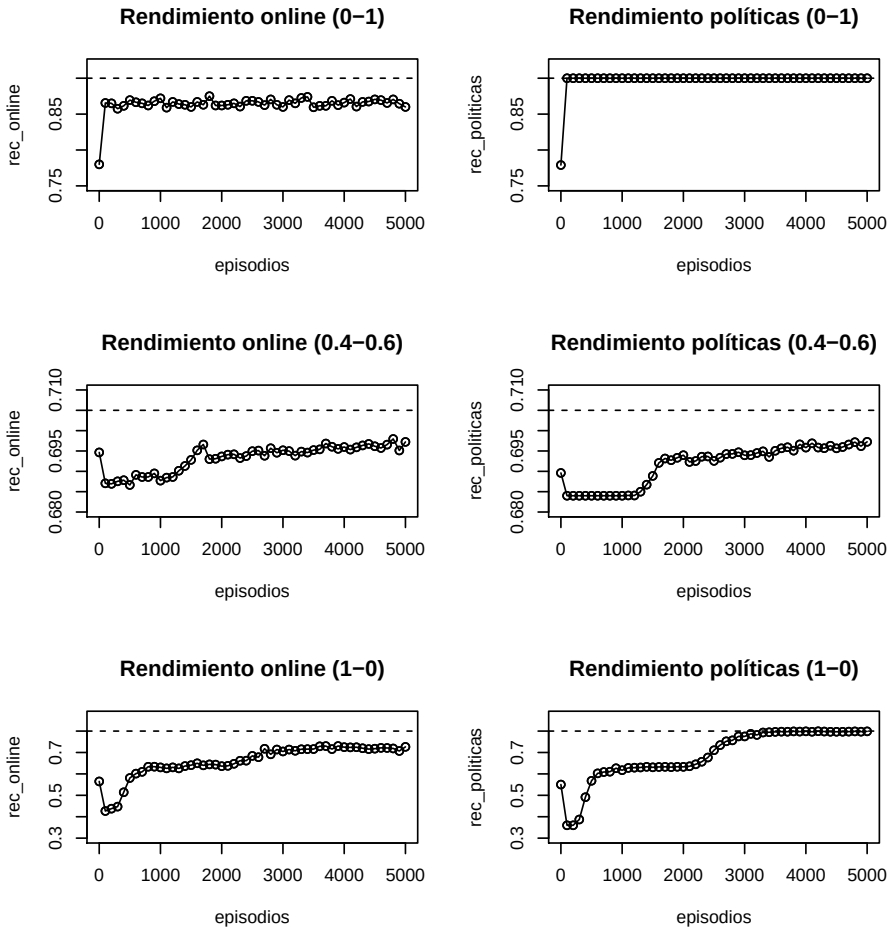

Figura 3. Rendimientos para la técnica de aproximación de valores escalarizados linealmente.

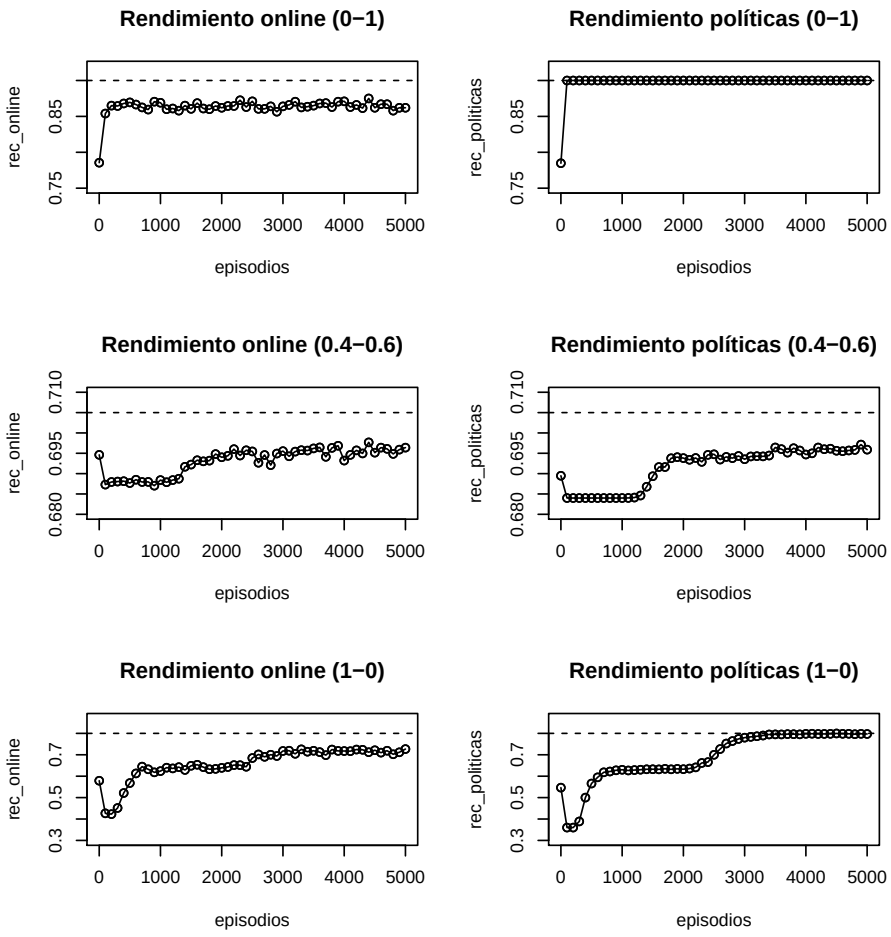

Figura 4. Rendimientos para la técnica de escalarización lineal de valores aproximados.

Sin embargo, la técnica de escalarización de valores aproximados, introducida en este trabajo, podría ofrecer cierta ventaja en otros ejemplos que puedan beneficiarse de utilizar conjuntos distintos de rasgos para aprender distintos objetivos, ya que tenemos una función $Q_i(s, a)$ para cada objetivo y por tanto cada una de ellas puede utilizar un conjunto propio de características. Esto podría ser útil si se conocen rasgos expertos que puedan acelerar el proceso de aprendizaje de un objetivo concreto y que sólo sean relevantes en la obtención de dicho objetivo. También podría suceder que diferentes objetivos se aprendan mejor con distintos parámetros (es decir, distintas tasas de aprendizaje, exploración, descuento o *bootstrapping* para cada objetivo).

6. Conclusiones y Trabajo Futuro

En este trabajo hemos descrito y comparado dos técnicas para combinar métodos de aproximación de funciones y de escalarización lineal de múltiples objetivos en el algoritmo de diferencias temporales $Q(\lambda)$.

La primera técnica consiste en utilizar diferentes configuraciones de pesos para escalarizar las recompensas vectoriales y utilizar métodos ya existentes de aproximación de funciones. La segunda técnica, por el contrario, utiliza las configuraciones de preferencias para escalarizar las funciones de valor aproximado. Hasta donde sabemos, esta segunda técnica no ha sido previamente descrita en la literatura.

Para comparar ambos métodos hemos llevado a cabo una serie de experimentos sobre un problema sencillo. Los rendimientos de los experimentos correspondientes a ambas técnicas han resultado ser equivalentes. No obstante, la técnica introducida en este trabajo podría ser de utilidad en situaciones que puedan beneficiarse de usar conjuntos diferentes de rasgos o de parámetros para distintos objetivos.

Como trabajo futuro se pretende aplicar la técnica de escalarización de valores aproximados a un problema más sofisticado, donde podamos beneficiarnos de la utilización de distintos conjuntos de rasgos o parámetros.

Referencias

1. Sutton, R., Barto, A.: Reinforcement learning: an introduction. MIT Press, Cambridge, Ma. (1998)
2. Roijers, D.M., Vamplew, P., Whiteson, S., Dazeley, R.: A survey of multi-objective sequential decision-making. Journal of Artificial Intelligence Research (JAIR) **48** (2013) 67–113
3. Castelletti, A., Corani, G., Rizzolli, A., Soncini-Sessa, R., Weber, E.: Reinforcement learning in the operational management of a water system. In: IFAC Workshop on Modeling and Control in Environmental Issues. (2002) 325–330
4. Natarajan, S., Tadepalli, P.: Dynamic preferences in multi-criteria reinforcement learning. ICML '05, ACM (2005) 601–608