Revista Iberoamericana de Automática e Informática Industrial

Revista Iberoamericana de Automática e Informática Industrial 16 (2019) 273-283

www.revista-riai.org

Metodología de programación dinámica aproximada para control óptimo basada en datos

Henry Díaz[a,*], Leopoldo Armesto[b], Antonio Sala[a]

[a] *Instituto U. de Automática e Informática Industrial, Universitat Politècnica de València, Cno. Vera, s/n, 46022, Valencia, España.*
[b] *Instituto de Diseño y Fabricación, Universitat Politècnica de València, Cno. Vera, s/n, 46022, Valencia, España.*

Resumen

En este artículo se presenta una metodología para el aprendizaje de controladores óptimos basados en datos, en el contexto de la programación dinámica aproximada. Existen soluciones previas en programación dinámica que utilizan programación lineal en espacios de estado discretos, pero que no se pueden aplicar directamente a espacios continuos. El objetivo de la metodología es calcular controladores óptimos para espacios de estados continuos, basados en datos, obtenidos mediante una estimación inferior del coste acumulado a través de aproximadores funcionales con parametrización lineal. Esto se resuelve de forma no iterativa con programación lineal, pero requiere proporcionar las condiciones adecuadas de regularización de regresores e introducir un coste de abandono de la región con datos válidos, con el fin de obtener resultados satisfactorios (evitando soluciones no acotadas o mal condicionadas).

Palabras Clave:
Control inteligente, Programación Dinámica Aproximada, Aprendizaje Neuronal, Control Óptimo.

Approximate Dynamic Programming Methodology for Data-based Optimal Controllers

Abstract

In this article, we present a methodology for learning data-based approximately optimal controllers, within the context of learning and approximate dynamic programming. There are previous solutions in dynamic programming that use linear programming in discrete state space, but cannot be applied directly to continuous space. The objective of the methodology is to calculate data-based optimal controllers for continuous state space, these controllers are obtained by a lower estimation of the accumulated cost through functional approximators with linear parameterization. This is solved non-iteratively with linear programming, but it requires to provide appropriate conditions for regressor regularization and to introduce a cost of leaving the region with valid data, in order to obtain satisfactory results (avoiding unrestricted or poorly conditioned solutions).

Keywords:
Intelligent Control, Approximate Dynamic Programming, Neural Learning, Optimal Control.

1. Introducción

El diseño de controladores basados en optimización es de gran relevancia actual en múltiples facetas teóricas (Ariño et al., 2010) y prácticas (Armesto et al., 2015; Duarte-Mermoud and Milla, 2018; Rubio et al., 2018).

Si los modelos son lineales, el control óptimo lineal cuadrático (conocido como LQR, del inglés Linear Quadratic Regulator), es una metodología bien estudiada en muchos textos (Albertos and Sala, 2006). En el caso no-lineal, en general no existen soluciones con una expresión analítica explícita del control óptimo. Una alternativa sería hacer control predictivo (Camacho and Bordons, 2010) no lineal, resolviendo problemas de optimización en línea (Allgöwer and Zheng, 2012; Ziogou et al., 2013) o iterativa (Ariño et al., 2014; Armesto et al.,

*Autor para correspondencia: hendia@posgrado.upv.es
To cite this article: H. Díaz, L. Armesto, A. Sala. 2019. Approximate Dynamic Programming Methodology for Data-based Optimal Controllers. Revista Iberoamericana de Automática e Informática Industrial 16, 273-283. https://doi.org/10.4995/riai.2019.10379

2015); no obstante, las técnicas de control predictivo se dejan, intencionalmente, fuera de los objetivos de este manuscrito porque se quieren plantear soluciones que impliquen el ajuste de controladores aproximadamente óptimos a horizonte infinito, basados en datos, en vez de en simulaciones a horizonte finito propias del control predictivo no lineal.

El área de control inteligente (Santos, 2011) es, desde sus inicios, muy activa en este tipo de problemas, dando lugar a las disciplinas que se conocen como aprendizaje por refuerzo (Sutton and Barto, 1998) o programación dinámica aproximada (Busoniu et al., 2010; Lewis and Liu, 2013).

Ambas concepciones están muy relacionadas. Estas técnicas buscan expresiones explícitas de funciones de valor $V(x)$ relacionadas con la ecuación de Bellman (Sutton and Barto, 1998) y controladores asociados que sean una *aproximación* razonablemente cercana al controlador óptimo. El concepto de aprendizaje por refuerzo en literatura suele estar más orientado hacia la mejora progresiva de un control a base de repeticiones de la tarea por experimentación en línea, mientras que el concepto de programación dinámica aborda problemas de aprendizaje de controladores óptimos a partir de un modelo o conjunto de datos prefijado.

Para obtener las funciones de valor anteriormente mencionadas o, en algunos casos, la denominada función de acción-valor $Q(x, u)$ (Lewis and Vrabie, 2009), los algoritmos más extendidos son los denominados "iteración de política" (*policy iteration* en lengua inglesa, PI) o "iteración de valor" (*value iteration* en la literatura en lengua inglesa, VI), que progresivamente van mejorando los estimados de las funciones y controladores intervinientes, véase, por ejemplo Lewis and Liu (2013); Busoniu et al. (2010) para más detalles. Desafortunadamente, dichos algoritmos sólo funcionan con garantías de convergencia en ciertas condiciones, básicamente, espacios de estado y control discretos de modo que los estimados de las funciones y acciones de control sean meras tablas de datos (Busoniu et al., 2010).

En un caso más general, se necesita usar aproximadores funcionales para funciones de valor $V(x, \theta)$, $Q(x, u, \theta)$ o controladores $u = \pi(x, \theta)$. El teorema de aproximación funcional universal afirma que muchos aproximadores pueden aproximar a un error arbitrariamente pequeño una función continua en una zona de modelado compacta añadiendo un suficiente número de regresores, polinomios, neuronas, etc. (Hornik et al., 1989). Por tanto, dicha capacidad de aproximación sienta las bases para su uso en programación dinámica aproximada.

Es interesante mencionar que, dentro del aprendizaje de controladores óptimos, existen técnicas que ajustan directamente los parámetros θ de una política de control $\pi(x, \theta)$ a base de repeticiones exhaustivas de experimentos donde θ varía aleatoriamente; estas estrategias son conocidas como *búsqueda de políticas*, y en el trabajo (Deisenroth et al., 2013) se hace una revisión en profundidad de dichas metodologías. Estas opciones de exploración están fuera de los objetivos que se pretenden plantear en este trabajo.

El problema fundamental de la programación dinámica aproximada es que las garantías de optimalidad se pierden al usar aproximadores $V(x, \theta)$ y, asimismo, los algoritmos iterativos clásicos PI o VI podrían no converger (Fairbank and Alonso, 2012), por pérdida de contractividad (Busoniu et al., 2010); ello podría requerir cambios en ellos, utilizando, por ejemplo, iteraciones descendiendo por gradiente en lo que se denomina minimización del residuo de Bellman (Antos et al., 2008; Díaz et al., 2018). Como alternativas a las metodologías iterativas, cambiando igualdades por desigualdades en la ecuación de Bellman, reformulaciones del problema como un problema de programación lineal pueden obtener, sin iteraciones, la solución óptima en casos de espacios de estado y control discretos, o una aproximación a la misma (De Farias and Van Roy, 2003). El coste computacional de una solución de programación lineal es mayor que el de una iteración PI/VI, pero sólo es necesario ejecutarlo una vez, lo cual, añadido a que se eliminan los problemas de convergencia, supone una ventajosa opción. Además, teniendo en consideración que el número de iteraciones que los algoritmos PI/VI requieren para converger, suele ser por lo general elevado, el tiempo de cómputo total de los algoritmos es también elevado, particularmente cuando el factor de descuento tiende a 1 (Condon, 1992).

El objetivo de este trabajo es presentar una metodología que adapte De Farias and Van Roy (2003) a espacios de estados continuos. En efecto, el uso directo de las técnicas en los trabajos arriba citados en problemas de estabilización y control óptimo con regresores y datos arbitrariamente elegidos puede plantear problemas de programación lineal no acotados o mal condicionados numéricamente. Este trabajo desarrolla una metodología, basada en desigualdades de programación lineal, que incluye detección de los posibles problemas numéricos en las desigualdades, regularización/selección de regresores e inclusión de un coste final en trayectorias que abandonen una región de trabajo. Con ello, se podrán obtener soluciones adecuadas en problemas de control óptimo en espacios continuos, a partir de un conjunto de datos que explore suficientemente el espacio de estados.

La estructura de este trabajo es la siguiente: en la sección 2 se discuten los trabajos preliminares relacionados con el artículo y el planteamiento del problema; en la sección 3 se describe la metodología de programación dinámica aproximada basada en datos propuesta; en la sección 4 se muestran los resultados de la metodología para dos sistemas de primer orden (tipo LQR) y otro de segundo orden (problema de montaña-carrito); finalmente, en la sección 5 se extraerán unas conclusiones del trabajo.

2. Preliminares

Consideremos un modelo de un sistema dinámico discreto no lineal:

$$x_+ = f(x, u) \tag{1}$$

donde $x \in \mathbb{R}^n$ es el estado y $u \in \mathbb{R}^m$ y x_+ representa el estado sucesor. Una política $u = \pi(x)$ es una función $\mathbb{R}^n \to \mathbb{R}^m$ que representa el controlador en bucle cerrado del sistema.

El coste de aplicar una política $\pi(x)$ partiendo de un estado inicial x_0 se define como:

$$V_\pi(x_0) := \sum_{k=0}^{\infty} \gamma^k L(x_k, \pi(x_k)) \tag{2}$$

siendo x_k el estado en el instante de tiempo k, $L(x, u)$ una función escalar $\mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$, también conocida como " coste inmediato" y $0 < \gamma < 1$ el factor de descuento.

El objetivo del control óptimo es encontrar una política óptima $\pi^*(x)$ que minimice $V_\pi(x)$. En este sentido, es bien conocido que el coste de una política debe verificar la Ecuación de Bellman:

$$V_\pi(x) = L(x, \pi(x)) + \gamma V_\pi(f(x, \pi(x))) \tag{3}$$

y que la función de valor de la política óptima, denotada como $V^*(x)$, verifica:

$$V^*(x) = \min_u L(x, u) + \gamma V^*(f(x, u)) \tag{4}$$

y, a partir de ella, la política óptima puede ser recuperada con:

$$\pi^*(x) = \arg\min_u L(x, u) + \gamma V^*(f(x, u)). \tag{5}$$

2.1. Programación dinámica aproximada

Basado en la ecuación de Bellman (3), dada una política $\pi(x)$, si definimos el error de diferencia temporal como:

$$\epsilon(x, \theta) := V_\pi(x, \theta) - L(x, \pi(x)) - \gamma V_\pi(f(x, \pi(x)), \theta) \tag{6}$$

entonces la solución aproximada al problema de "evaluación de la política $\pi(x)$" (*policy evaluation* en literatura en lengua inglesa) viene dada por:

$$\theta^*_\pi := \arg\min_\theta \|\epsilon(x, \theta)\|^2 \tag{7}$$

donde

$$\|\epsilon(x, \theta)\|^2 := \sum_{i=1}^{N} \epsilon(x_i, \theta)^2 \tag{8}$$

esto es, se trata de minimizar el error cuadrático de la ecuación de Bellman (6) sobre una serie de N datos disponibles.

Análogamente, dada una función de valor $V_\pi(x, \theta)$, se puede obtener una mejora de la política (*policy improvement* en la literatura en lengua inglesa) mediante:

$$\pi_+(x, \theta) := \arg\min_u L(x, u) + \gamma V_\pi(f(x, u), \theta) \tag{9}$$

de modo que, si $V_\pi(x, \theta)$ fuera coincidente con la función de valor "exacta" $V_\pi(x)$ se puede demostrar que $V_{\pi_+}(x, \theta) \leq V_\pi(x, \theta)$.

Iteración de política. La ejecución iterativa de los pasos de "evaluación" y de "mejora" de la política es un algoritmo de aprendizaje que se denomina iteración de política (PI), Lewis and Vrabie (2009).

Iteración de valor. A partir de un estimado inicial de una función de valor $V(x, \theta_0)$, las siguientes iteraciones

$$\theta_{k+1} := \arg\min_{\hat{\theta}} \left\| V(x, \hat{\theta}) - \min_u (L(x, u) + \gamma V(f(x, u), \theta_k)) \right\|^2 \tag{10}$$

son denominadas en la literatura como iteración de valor (VI), ver Lewis and Vrabie (2009).

De forma conveniente, en programación dinámica aproximada se suele usar aproximadores funcionales lineales en los parámetros

$$V(x, \theta) = \psi^T(x)\theta \tag{11}$$

de modo que las optimizaciones anteriores se convierten en problemas de mínimos cuadrados lineales resolubles mediante una pseudoinversa.

Es bien conocido que los algoritmos PI y VI en su versión discreta (no aproximada), donde los parámetros ajustables se implementan mediante tablas (Sutton and Barto, 1998) convergen a la función de valor y política óptima. También es bien conocido que, en general, en el caso aproximado dicha convergencia no está garantizada. Sólo en el caso de que se cumplan determinadas condiciones de contractividad (Busoniu et al., 2010) se puede garantizar convergencia a un cierto "punto fijo" que podrá estar o no suficientemente cerca del óptimo para producir una política razonablemente buena en una aplicación y parametrización concretas. El problema es que probar dicha contractividad para un aproximador $V(x, \theta)$ suele ser difícil o imposible, dado que los teoremas asociados no son constructivos. Sólo en casos muy especiales puede probarse dicha contractividad, como la estructura de tabla de interpolación planteada en Busoniu et al. (2010); no obstante esta propuesta tiene como desventaja el excesivo número de parámetros a ajustar y su poca aplicabilidad a espacios de estados de dimensión elevada.

2.1.1. Programación lineal

En espacios de estado y de acciones de control discretos, dados por $\tilde{\mathbb{X}}$ y $\tilde{\mathbb{U}}$ respectivamente (en este caso sí se da la contractividad, como antes se ha comentado), la solución a la que llegan PI/VI puede ser obtenida planteando la ecuación de Bellman (4) sustituyendo igualdades por desigualdades:

$$V^*(x) \leq L(x, u) + \gamma V^*(x_+) \;\forall\; x \in \tilde{\mathbb{X}}, u \in \tilde{\mathbb{U}} \tag{12}$$

y, dado que existe una solución que verifica la igualdad (la óptima), si se *maximizara* el índice

$$J_{LP} := \sum_{i=1}^{N} V^*(x_i) \tag{13}$$

se obtendría la solución óptima del problema de programación dinámica *exacta*, siendo N la cardinalidad de $\tilde{\mathbb{X}}$ y x_i una enumeración arbitraria de sus elementos, ver Manne (1960); Denardo (1970) para detalles.

En el caso de programación dinámica *aproximada*, esto es, al utilizar un aproximador funcional $V(x, \theta)$, en general no es posible obtener el ajuste perfecto a la solución óptima en todos los puntos (Si et al., 2004). En ese caso, De Farias and Van Roy (2003) propone sustituir (13) por una media ponderada para obtener una cota inferior del coste, como consecuencia de reemplazar igualdades en (4) por desigualdades en (12).

2.2. Planteamiento del problema

Como se ha discutido, PI/VI presentan problemas de convergencia difíciles de saber "a priori" para una cierta parametrización. Por el contrario, las versiones basadas en programación lineal no son iterativas y, por tanto, no existe el problema de la falta de convergencia/contractividad.

Sin embargo, estas últimas técnicas todavía presentan problemas en aplicaciones de control óptimo en espacios de estados contínuos. En concreto, para obtener sus soluciones, De Farias and Van Roy (2003) asume que los estados sucesores x_+ en

(12) aparecen también en el lado izquierdo de otra de las desigualdades, porque asume un espacio de estados discreto completo. Sin embargo, en un sistema con estado continuo, y un número finito de desigualdades (asociadas a un conjunto finito de datos), esto no se puede garantizar y pueden aparecer soluciones no acotadas (o muy mal condicionadas numéricamente).

El objetivo de este trabajo es plantear una metodología completa, a partir de un conjunto de datos, que incluye: planteamiento de desigualdades (12), análisis de la factibilidad y condicionamiento del problema de programación lineal, regularización de regresores, e introducción de coste final. Con esta metodología, se han podido resolver satisfactoriamente los problemas de control óptimo propuestos en la sección de ejemplos, obteniendo una cota inferior que aproxima la función de valor óptima.

3. Metodología de programación dinámica aproximada

En este trabajo, consideraremos que disponemos de un conjunto $\mathcal{D}$ de N tripletes de datos obtenidos, bien de simulación de un modelo o de experimentación real[1]:

$$\mathcal{D} \quad := \quad \{(x_1, u_1, x_{1+}), (x_2, u_2, x_{2+}), \ldots, (x_N, u_N, x_{N+})\} \quad (14)$$

donde $x_{i+} := f(x_i, u_i)$ es el estado sucesor ante la acción u_i. El conjunto de datos no tiene por qué estar ordenado temporalmente en la metodología que sigue. Se supondrá que todos los x_i, x_{i+} están en una región de operación $\mathbb{X}$ donde se van a situar determinados regresores/neuronas para aprendizaje.

Para que la optimización sea razonable en términos de coste computacional, asumiremos $V(x, \theta)$ parametrizado linealmente según la expresión (11).

El primer concepto que se va a discutir es la regularización de regresores para que los problemas de programación lineal considerados en la sección anterior obtengan una solución acotada superiormente.

La programación lineal debe plantear restricciones $A\theta \leq b$. En nuestro caso, plantearemos una desigualdad para cada dato de $\mathcal{D}$, de modo que introduciendo en (12) la parametrización (11), exigiremos la diferencia temporal (temporal difference, TD en lengua inglesa):

$$\left(\psi^T(x_k) - \gamma\psi^T(x_{k+})\right) \cdot \theta \leq L(x_k, u_k) \quad (15)$$

de modo que podremos definir las matrices asociadas a las restricciones lineales como:

$$A := \begin{pmatrix} \psi^T(x_1) - \gamma\psi^T(x_{1+}) \\ \vdots \\ \psi^T(x_N) - \gamma\psi^T(x_{N+}) \end{pmatrix} \quad b := \begin{pmatrix} L(x_1, u_1) \\ \vdots \\ L(x_N, u_N) \end{pmatrix} \quad (16)$$

El índice de coste (13) será reformulado como la siguiente suma ponderada sobre los puntos del conjunto de datos:

$$J_{LP,\omega}(\theta) := \sum_{k=1}^{N} \omega(x_k)\psi^T(x_k)\theta \quad (17)$$

donde $\omega(x_k)$ es una ponderación positiva arbitraria para enfatizar, si se desea, el ajuste en determinadas regiones del espacio de estados (o simplemente $\omega(x_k) = 1$, por defecto), dado que al usar el aproximador estamos suponiendo que no podremos obtener la solución "ideal" V^*. El resultado será un índice de coste:

$$J_{LP,\omega}(\theta) = f^T\theta \quad (18)$$

siendo

$$f^T = \sum_{k=1}^{N} \omega(x_k)\psi^T(x_k) \quad (19)$$

un vector de coste que sólo depende de los datos.

Normalización de regresores. Como la función de valor a aproximar es positiva, vamos a asumir que los regresores son no negativos en el resto del trabajo, esto es $\psi_j(x) \geq 0$ para todo $x \in \mathbb{X}$. Sin pérdida de generalidad, podemos suponer también que $\max_x \psi_j(x) = 1$ porque cualquier escalado de los regresores podrá ser absorbido por el parámetro θ_j asociado a él. De este modo, todos los elementos del vector f en (18) serán no negativos y podrán fijarse umbrales de activación para resolver diferentes problemas que puedan aparecer (discutidos a continuación), tomando 1 como "valor de referencia".

Con esta normalización, dado que $L(x, u) \geq 0$, se verifica que todos los elementos de b son, asimismo, no negativos y, por consiguiente $\theta = 0$ será una solución factible del problema de maximizar $f^T\theta$ sujeto a $A\theta \leq b$. Dado que la factibilidad, entonces, está garantizada, esto significa que la metodología propuesta de programación lineal, o bien funciona y obtiene un valor de θ^* acotado inferiormente, solución del problema de programación dinámica aproximada planteado, o bien falla por obtener una solución no acotada superiormente.

3.1. Causas de soluciones no acotadas o de problemas numéricos

La matriz A definida en (16) es de tamaño $N \times M$ donde la fila k-ésima viene dada por:

$$\psi^T(x_k) - \gamma\psi^T(x_{k+}) \quad (20)$$

y en el vector b, de tamaño $N \times 1$, aparecerá $L(x_k, u_k)$ en el elemento en posición k.

Dado que las restricciones son $A\theta \leq b$, si en una columna de A no existe al menos un elemento estrictamente positivo, la programación lineal dará una solución no acotada porque ese θ_j puede crecer sin límite haciendo cada vez menor el lado izquierdo de la desigualdad.

Dado que, según se ha supuesto arriba, los regresores están normalizados a máximo unidad, una solución confiable numéricamente debería dar valores de los parámetros θ resultantes de la optimización del orden de magnitud de las funciones de valor. En caso de que el máximo elemento de la columna de A sea muy pequeño, el resultado podría dar lugar a valores muy grandes de θ. Esto tampoco se considerará deseable, dado que obtener una función de valor "pequeña" como interpolación o

[1] Aprender un buen controlador requiere de una exploración densa del espacio de estados y acciones de control, lo cual puede no resultar fácil de forma experimental. Para que el problema LP estuviera bien planteado, se necesitaría una etapa inicial de *clustering* para poner regresores sólo donde hubiera datos, e introducir un coste "final" si el controlador mueve al estado hacia regiones no exploradas. Estas consideraciones quedan, no obstante, fuera del alcance de este trabajo.

diferencia entre valores "grandes" de parámetros θ (por ejemplo, $30 = 2{,}323441 * 10^6 - 2{,}323411 * 10^6$) no parece que sea una opción aconsejable en aplicaciones: el comportamiento de los controladores aprendido no debería depender de las últimas cifras significativas de los parámetros estimados.

Vamos a analizar separadamente las causas de que ocurran las situaciones arriba mencionadas, para poder solucionarlas posteriormente en una metodología de regularización de regresores:

a) Regresor demasiado "plano". Definamos $\mathrm{TD}_{jk} := \psi_j(x_k) - \gamma\psi_j(x_{k_+})$, como el factor que multiplica al parámetro del regresor j en el dato k. Si TD_{jk} es muy pequeño en todos los datos, el resultado de la optimización dará unos valores de parámetros estimados muy grandes y mal condicionados, por tanto deberíamos evitar que $\mathrm{TD}_{jk} \approx 0$ para todo k. Esto puede ocurrir por dos causas:

- Para todos los u del conjunto de datos $x_{k+} \approx x_k$ en los puntos donde el regresor está activo, esto es, el regresor se activa en las inmediaciones de un punto de "equilibrio" del que las acciones de control disponibles en el conjunto de datos no parecen capaz de moverlo.
- El estado sí se mueve significativamente, pero el regresor es "demasiado plano" y ψ_j no cambia mucho entre x_k y x_{k+}.

Obviamente, la relevancia de este problema aumenta conforme el factor de descuento γ se acerca a la unidad.

b) Regresor se activa en x_+ pero no en x. Si el regresor se activa en todos los casos igual o más en el estado sucesor que en el estado origen, entonces (20) resulta negativo y el problema PL daría no acotado. Ello significa que los estados sucesores "entran" en una región donde determinado regresor está activo, pero no existen muestras de datos que "salgan" de dicha región.

c) Regresor no activo sobre el conjunto de datos. También dará problemas numéricos un regresor que no se active nunca (ni en los datos x ni en los datos x_+). El parámetro estaría, en esa situación extrema ($\psi_j = 0$) libre; en general, si se activa "poco" ($\psi_j(x_k) \approx 0$ para todo k) el parámetro θ_k puede llegar a tener un gran valor y distorsionar la superficie de función de valor obtenida.

3.2. Metodología de regularización de regresores

Basándose en las ideas anterior, para que un problema esté bien planteado, y dé resultados confiables numéricamente, la propuesta que se realiza en este trabajo es:

1. Generar unos regresores candidatos

$$\psi(x) = [\psi_1(x) \dots \psi_M(x)]$$

   y escalarlos a [0,1] en una región de operación $\mathbb{X}$. Estos regresores podrían ser, por ejemplo, neuronas de base radial, polinomiales (inspirados en el hecho de que en plantas lineales con L cuadrática la función de valor es cuadrática), etc., ver sección 3.4.
2. Comprobar que en los puntos del conjunto de datos, la activación de ellos es suficiente, esto es, que $\max_k \psi_j(x_k) \geq \beta_0$ para todo $k = 1, \dots, N$, $j = 1 \dots, M$ siendo $1 > \beta_0 > 0$ un umbral de activación adecuado; por ejemplo, dado que los regresores están normalizados a máximo unidad, requerir $\beta_0 = 0{,}5$ parece una opción razonable porque el máximo del regresor en $\mathbb{X}$ es 1, y con $\beta = 0{,}5$ existe al menos un dato que lo activa como poco al 50 % de su activación máxima. En caso contrario, los datos no excitan suficientemente a ese regresor y sería candidato a ser eliminado (o, alternativamente, si se activa en una zona importante del espacio de estados donde no hay datos, deberíamos generar nuevos datos adecuadamente).
3. Comprobar que $\max_k \mathrm{TD}_{jk} \geq \beta_1$ siendo $1 > \beta_1 > 0$ un parámetro de regularización, entendido como umbral de "diferencia temporal de activación" (para asegurar que, de un estado a su sucesor, el regresor difiere lo suficiente). El máximo valor posible de TD_{jk} es 1, que se produciría si el estado de origen está en el máximo del regresor, $\psi_j(x_k) = 1$, y el sucesor estuviera muy alejado del mismo, $\psi_j(x_{k_+}) = 0$. Por tanto, para evitar problemas de regresor plano/excitación insuficiente, la propuesta es que exista al menos un dato donde $\mathrm{TD}_{jk} \geq \beta_1$, siendo β_1 un valor entre el 1 % y el 5 % de ese salto máximo (unidad). En caso de que esta condición no se cumpla, debería aumentarse la amplitud de las señales de control (mayor exploración), disminuirse γ (para que la diferencia temporal crezca) o cambiar la distribución/función de activación de los regresores, para evitar resultados con mal condicionamiento.

3.2.1. Incorporación de coste final

Si hay trayectorias del sistema que abandonan la región de operación $\mathbb{X}$, el coste podría no estar adecuadamente definido por falta de datos fuera de dicha región y, también, en muchas ocasiones, por falta de regresores en ella (no tiene mucho sentido añadir neuronas que se activan en zonas donde no hay datos). Esto podría ocasionar, también, deformaciones no esperadas en la función de valor aprendida o problemas numéricos de los considerados en la sección 3.1. Por ello, para evitar un problema que, formalmente, no está bien planteado (la solución es "indefinida" para trayectorias que abandonan $\mathbb{X}$), se propone establecer un "coste final" $J_{fuera}(x)$ indicando qué sucede cuando se abandona la región de operación. Esta construcción es común en el planteamiento de problemas de control óptimo, dando "premios" o "castigos" según por donde se abandona $\mathbb{X}$, véase, por ejemplo, Sutton and Barto (1998).

Con ello, se podría incorporar a $\mathcal{D}$ los datos disponibles donde $x \in \mathbb{X}$ pero $x_+ \notin \mathbb{X}$ que, en la definición inicial de $\mathcal{D}$ tras la ecuación (14) no se podían tener en cuenta. Así, para esos datos, en vez de la diferencia temporal (15) implementaremos

[2] De hecho, el coste de violar restricciones podría ser entendido como "infinito". No obstante, para evitar distorsiones en la función de valor ajustada se aconseja usar un valor "suficientemente alto" de J_{fuera}, pero no "demasiado alto", dado que los regresores propuestos en la sección 3.4 sólo son aproximadores funcionales universales de funciones *contínuas*; obviamente, el significado concreto de "alto pero no demasiado" es dependiente del problema.

la modificación:

$$V(x_k, \theta) \leq L(x_k, u_k) + \gamma J_{fuera}(x_+) \tag{21}$$

para todos aquellos k tales que $x_{k+} \notin \mathbb{X}$.

Nótese que, si se desea que el controlador óptimo intente que las trayectorias nunca abandonen $\mathbb{X}$ podría fijarse J_{fuera} a un valor muy alto[2].

Construcción del controlador. Nótese que la propuesta basada en datos únicamente aproxima la función de valor $V(x, \theta)$. Por tanto, la política óptima asociada requeriría resolver el problema de optimización a un paso dado por la expresión (9). Con regresores arbitrarios podría no resultar conveniente dicha resolución en línea, por lo que podría resultar recomendable calcular el controlador fuera de línea y programarlo como una tabla de interpolación o un segundo aproximador funciona basándose en las ideas de los conceptos Actor-Crítico de la literatura (Grondman et al., 2012), si bien estos aspectos constituyen detalles de implementación que no afectan a la metodología de ajuste de $V(x, \theta)$ propuesta.

3.3. Discusión sobre la aproximación

Como existen un número finito de datos y regresores, los resultados de la optimización (18) son una aproximación de la función de valor óptima real. En el *mejor* de los casos tendríamos un gran número de datos tanto en x como en u equiespaciados y distribuidos por las regiones $\mathbb{X}$ y $\mathbb{U}$ respectivamente que nos permitirán "capturar" la función de valor más próxima a la función de valor óptima real si ese número de datos tendiera a infinito. No obstante, en situaciones prácticas ello no es posible:

- La granularidad en u hace que la estrategia sea subóptima, esto es, que el coste estimado pueda ser *superior* al óptimo con una acción de control "contínua" (granularidad infinitamente fina).
- La granularidad finita en x hace que la ecuación de Bellman (desigualdad) sólo se cumpla en los puntos en los que se dispone de datos. En puntos "intermedios" del espacio de estados, no presentes en $\mathcal{D}$, no tenemos garantía de que se verifique la ecuación de Bellman, porque no se lo hemos exigido al aproximador en las restricciones. Con mas datos, habría más restricciones y la cota de la función de valor obtenida no sería igual de alta para un mismo estado.
- El número finito de regresores también produce una aproximación de la función de valor óptima que podría ser mejor con mayor número de los mismos.

Por las cuestiones arriba indicadas, para que los resultados sean aproximadamente correctos, el conjunto de datos y de acciones de control probadas tiene que ser lo suficientemente denso, para que la programación lineal obtenga una cota inferior de la función de valor que se aproximará progresivamente a la función de valor óptima conforme los regresores vayan siendo más ricos: como en todo problema de identificación, cuantos más regresores (más parámetros ajustables), más datos serán necesarios para asegurarse que la aproximación no está ajustada a unos pocos datos en particular, sino que generalice adecuadamente.

3.4. Propuestas de regresores

Mientras el problema de aprendizaje se plantee con una estructura de aproximador de la función de valor que sea lineal en parámetros, todos los resultados y discusiones considerados anteriormente tienen validez. La forma de los regresores ψ más adecuada depende del problema a resolver, y de cómo aproximen la función de valor idealmente óptima. Como dicha función es desconocida *a priori*, se suelen usar regresores "caja negra" estándar en muchas aplicaciones. En esta sección se detallan los regresores polinomiales, gaussianos y las tablas de interpolación, como propuestas de uso, aunque en cada problema podrían existir determinadas funciones no lineales del estado (energía mecánica, balances termodinámicos) que, incorporadas en los regresores podrían ajustar bien la función de valor con pocos parámetros; estas consideraciones particulares dependientes de cada problema no entran en los objetivos de este trabajo.

Regresores polinomiales. El caso más sencillo sería el aproximador cuadrático en el estado $V(x, \theta) = x^T P(\theta) x$ donde θ simbolizaría los elementos de la diagonal de la matriz P, así como los elementos sobre dicha diagonal (al ser P simétrica, no es necesario establecer parámetros separados para los elementos de la matriz por debajo de la diagonal). Este aproximador cuadrático podría obtener la función de valor exacta de la solución de un controlador lineal cuadrático (LQR, en la literatura en lengua inglesa), si se explorara el espacio completo de acciones de control. No obstante, debido a la exploración de un conjunto finito de acciones de control, el resultado obtenido será solamente una aproximación a dicho LQR.

Asimismo, en casos no lineales o, incluso, en casos lineales con coste J_{fuera} la función de valor óptima no será cuadrática, por lo que se requerirá el uso de otro tipo de aproximadores. Podrían ser, obviamente, polinomios de mayor grado.

En los ejemplos discutidos en la próxima sección, se han utilizado los aproximadores funcionales radiales y tablas de interpolación, que pasan a detallarse.

Redes neuronales radiales. Las redes neuronales radiales (Park and Sandberg, 1991; Gil and Páez, 2007; Yañez-Badillo et al., 2017), redes RBF en la literatura en lengua inglesa, son aproximadores funcionales de la forma:

$$V(x, \theta) = \sum_{i=1}^{h} \theta_i e^{-(x-c_i)^T \Sigma^{-1} (x-c_i)} \tag{22}$$

donde el parámetro Σ es un parámetro de varianza (que, por simplicidad, se ha decidido que sea idéntico en todas las neuronas, aunque podría no ser el caso si así se deseara), y c_i es el "centro" de activación de la neurona i-ésima, de modo que la activación es unitaria cuando $x = c_i$. De este modo, esta red neuronal verifica las condiciones de normalización requeridas en la sección anterior.

En un espacio de operación $\mathbb{X}$ formado por un hiperrectángulo, se propone dividir cada dimensión en un cierto número de puntos, a elegir por el usuario de modo que los centros de activación conformen un mallado regular. La matriz Σ se sugiere que sea una matriz diagonal tal que la activación en el centroide de la neurona adyacente más próxima sea de 0,5.

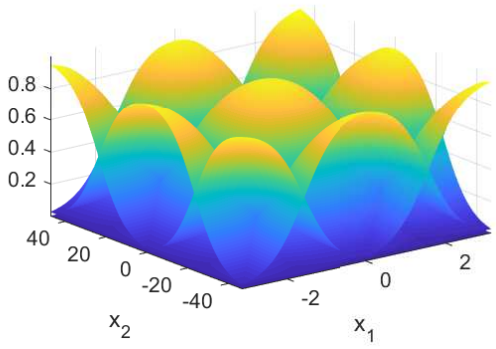


Figura 1: 2D función de activación de red neuronal gaussiana

Tablas de interpolación. En el caso de que los propios estados x en el conjunto de datos $\mathcal{D}$ formen un mallado equiespaciado (similar al de los centroides de red neuronal justo descritos), Busoniu et al. (2010) propone utilizar una tabla de interpolación, equivalente a una función de activación producto cartesiano de funciones triangulares en cada dimensión. En ese caso, la citada referencia probó contractividad del aproximador en los términos necesarios para asegurar convergencia de las iteraciones de valor y de política. En nuestro caso, usando programación lineal, podemos obtener (sin iteraciones, con una sóla ejecución del optimizador) la misma función de valor que dichas iteraciones, según se ha comprobado en los ejemplos.

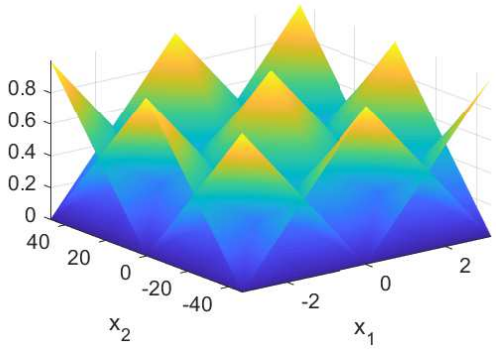


Figura 2: 2D función de activación de tabla de interpolación

4. Ejemplos

En esta sección se plantearán ejemplos sobre un sistema lineal, para valorar las prestaciones obtenidas con la metodología propuesta en un entorno cuya solución analítica es conocida (aunque sin considerar saturación). Posteriormente, se ilustrará la propuesta sobre un ejemplo de segundo orden popular en la literatura de aprendizaje por refuerzo y programación dinámica.

Ejemplo 1:Sistema lineal de primer orden, regresores neuronales, coste final. Consideremos el sistema lineal de primer orden

$$x_+ = Ax + Bu$$

con $A = 1$, $B = -0{,}5$, del que se dispone de datos (x,u) en una zona de operación dada por: $x \in [-5, 5]$, $u \in [-0{,}5, 0{,}5]$. Los datos de x están tomados con un mallado regular de 55 puntos, y los de u con un mallado de 21. El índice de coste viene definido por $L(x, u) = Qx^2 + Ru^2$, con $Q = 1$, $R = 10$, y un factor de descuento $\gamma = 0{,}9999$. También se ha añadido un coste final $J_{fuera} = 100$ para aquellas trayectorias cuyo estado abandona el intervalo $[-5, +5]$.

El objetivo de este ejemplo es comparar los resultados de nuestra propuesta con los resultados de un control óptimo LQR descontado, que resulta de resolver la siguiente ecuación de Riccati modificada con el factor de descuento γ:

$$S_+ = Q + \gamma A^T S A - \gamma^2 A^T S B(\gamma B^T S B + R)^{-1} B^T S A \qquad (23)$$

Se han comparado los resultados de regresores neuronales con 5 y 15 neuronas RBF uniformemente distribuidas, con parámetro $\Sigma = 1{,}11$ y $0{,}37$ respectivamente, así como los de una tabla de interpolación con el mismo número de puntos en la tabla que datos (esto es, 55), siguiendo las ideas de Busoniu et al. (2010).

La Figura 3 representa la función de valor (o cotas inferiores aproximadas de la misma, como resultado de la programación lineal) obtenida con las diferentes propuestas, a saber: regulador LQR descontado en negro, tabla de interpolación de 55 puntos en azul (etiquetado como LUT, del acrónimo inglés *Look-Up Table* ampliamente utilizado), una red neuronal (*neural network* en inglés, NN) de 5 neuronas en verde y otra de 15 neuronas en rojo. Se observa que, a medida que el regresor es más potente (mayor número de parámetros ajustables), la cota inferior de la función de valor obtenida por programación lineal se va haciendo cada vez mayor, siendo la más alta de todas la obtenida por la tabla de interpolación.

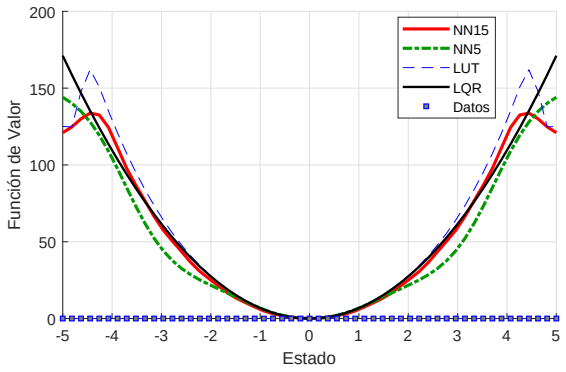


Figura 3: Función de valor o cota inferior.

Nótese que la cota de la tabla de interpolación podría considerarse como la más cercana al óptimo "exacto", dado que tiene 55 parámetros ajustables. Dicha cota está por encima de la del LQR (en algunos puntos) debido a la exploración con un número finito de acciones de control saturadas (seleccionadas del conjunto $\{-0{,}5, 0{,}5\}$, mientras que el regulador lineal óptimo podría producir acciones intermedias; la existencia de saturación también aumenta el coste óptimo respecto al LQR). También existe una deformación en los extremos izquierdo y derecho de la región porque, intencionalmente, el coste por abandonar la zona de operación se ha fijado a un valor constante inferior al del LQR: con ello, la política óptima cerca de los mencionados extremos es desistir, no alcanzar el origen, y simplemente abandonar la región de modelado. Obviamente, si no

se deseara ese efecto, se solventaría poniendo una penalización terminal mayor.

La Figura 4 representa las leyes de control aprendidas por cada una de las propuestas, con el mismo código de colores y etiquetas.

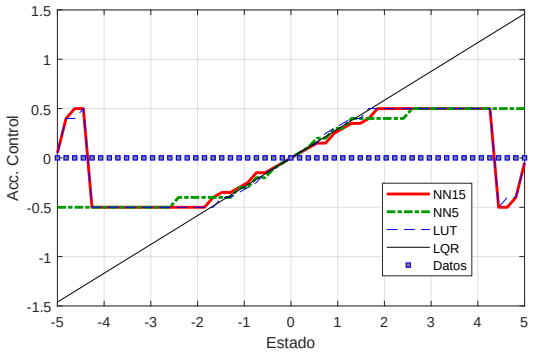


Figura 4: Ley de control

Se observa que cerca del origen los controladores se aproximan a la "discretización" del controlador lineal, como intuitivamente era de esperar. La saturación y el efecto del coste de abandonar la zona de modelado $\mathbb{X}$ son evidentes en las zonas alejadas del origen, como se había discutido arriba.

Ejemplo 2: Sistema de segundo orden (Montaña-Carrito). Consideremos el problema de alcanzar una posición objetivo acelerando una masa con energía limitada, descrito en la Figura 5. Variaciones de este problema han sido abordadas en otras referencias de literatura como Sutton and Barto (1998); Kretchmar and Anderson (1997); Busoniu et al. (2010).

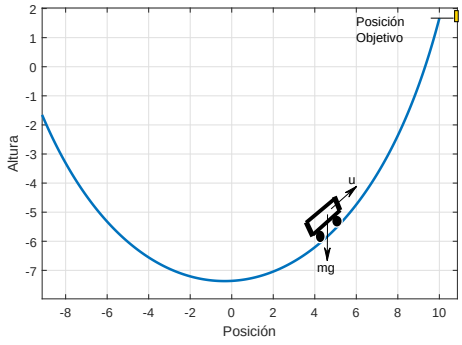


Figura 5: Sistema *Montaña-Carrito*

Dicho sistema puede ser representado por el siguiente modelo de un sistema de segundo orden donde la posición es p, la velocidad es v y u la acción de control (fuerza tangencial):

$$\dot{p} = v,\ \dot{v} = u - mg \cdot \sin(0{,}12p + 0{,}04)$$

con una masa de $m = 1$ kg y la gravedad $g = 9{,}81$ m/s^2. El término senoidal denota el efecto sobre la aceleración tangencial de la pendiente de la curva azul de la Figura 5 cuando el carrito se encuentra en la posición p.

El objetivo de control es llevarlo a la posición objetivo ($x \geq 10$) con una acción de control tomada en el intervalo $[-2, 2]$. El coste de etapa se ha establecido como:

$$L(x, u) = 1 + 0{,}2u^2$$

esto es, como una ponderación tanto del tiempo (número de muestras) transcurrido hasta alcanzar el objetivo como de la acción de control necesaria para conseguirlo. El factor de descuento se ha establecido a $\gamma = 0{,}999$.

Nótese que, intencionalmente, los límites preestablecidos de la acción de control no son suficientes para poder vencer la gravedad, de modo que el controlador óptimo debe aumentar su energía cinética balanceándose alrededor del "valle" hasta conseguir una energía suficiente para subir. Dado que el objetivo es "alejarse" del fondo, el resultado será, como es de prever, una dinámica inestable alrededor de dicho punto inferior. El objetivo del ejemplo es comprobar cómo un controlador neuronal puede aprender dicha estrategia.

La zona de modelado $\mathbb{X}$ donde se van a generar datos para aprendizaje es el rectángulo determinado por el intervalo de posición $x \in [-10, +10]$ m, y el intervalo de velocidad $[-15, 15]$ m/s. El coste por abandonar la zona de modelado será de 0 si se abandona por el lado derecho, y 900 si se abandona por el lado izquierdo (este valor se ha establecido, tal y como se discute en la nota al pie 2, para que sea suficientemente alto para que las trayectorias eviten el extremo izquierdo –hay algo mucho mejor a la derecha–, pero sin ser demasiado alto para evitar distorsionar la aproximación de la función de valor de las figuras 6(a) y 7(a) en puntos interiores a $[-10, 10]$). Como los datos abarcan una zona con velocidades menores de 15 m/s, si el valor absoluto de la velocidad en un estado sucesor supera 15 m/s también se dará un coste por abandonar $\mathbb{X}$ de 900.

El mallado de datos se ha realizado con 31 puntos en el eje de posición y otros 31 en el eje de velocidad. El control ha sido discretizado tomando un mallado de 11 puntos. Todos los mallados son equiespaciados.

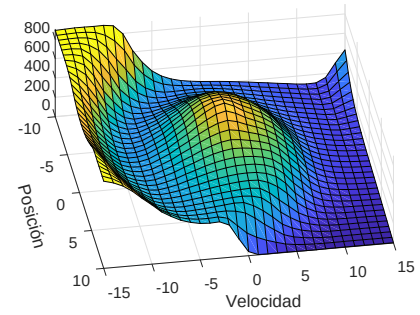
La metodología va a ser evaluada con unos regresores RBF y con unos regresores basados en tabla de interpolación bidimensional, detallados a continuación.

Los resultados con un modelo de 15x15 neuronas RBF equiespaciadas (con un parámetro $\Sigma = diag(\{0{,}74, 1{,}11\})$, teniendo un total de 225 parámetros ajustables) están representados en la Figura 6 de la página siguiente, donde se presenta la cota de la función de valor, el controlador obtenido (mapa de contornos), la simulación temporal y el plano de fase donde se observa la trayectoria de la posición y velocidad hasta que se alcanza el lado derecho, momento en el que se interrumpe la simulación. El controlador ha sido obtenido optimizando a un paso la ecuación (9) sobre el conjunto discreto $\mathbb{X}$ y, posteriormente, usando una tabla de interpolación de u en línea para obtener el valor de la acción de control en puntos intermedios fuera del conjunto de datos inicial aunque, obviamente, también hubiese sido razonable el optimizar (9) en linea, pero quizás con mayor coste de cómputo..

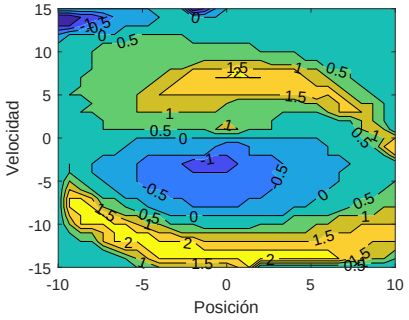
Si hubiésemos utilizado los regresores asociados a una tabla de interpolación bidimensional (31x31 puntos, con 961 parámetros ajustables), los resultados serían los que aparecen en la Figura 7, también en la página siguiente. Se observa que la red neuronal ha aprendido una ley de control bastante similar con casi cinco veces menos parámetros ajustables, y que la forma

de la función de valor aprendida es también similar (realmente, lo importante para que el mapa de control sea adecuado es poder estimar correctamente la diferencia temporal, relacionada con la "pendiente" del mapa de V, esto es, la relación entre $V(x)$ y $V(x_+)$; las diferencias de nivel "constante" en la función de valor no producen cambios significativos en la ley de control resultante).

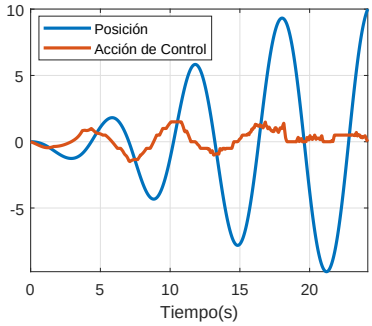
Como se observa en las simulaciones, los resultados no son muy diferentes en la primera opción (red neuronal) con casi cinco veces menos parámetros ajustables que la tabla de interpolación. Con la metodología propuesta, el controlador consigue levantar el carro hasta la posición de salida objetivo.



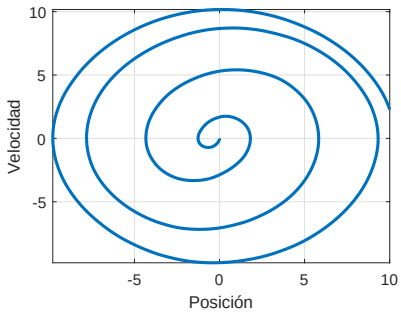
(a) Cota de función de valor



(b) Ley de control

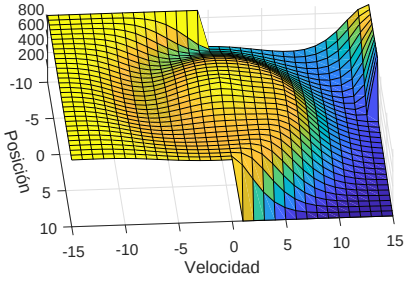


(c) Simulación temporal

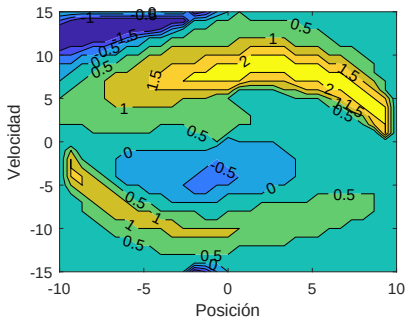


(d) Plano de fase

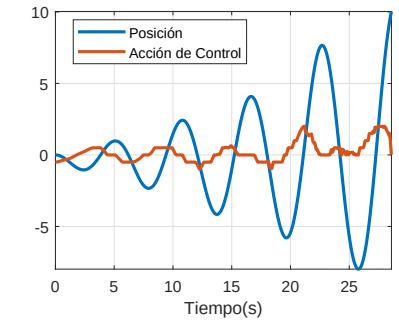
Figura 6: Ley de control y simulación con 15x15 neuronas RBF.



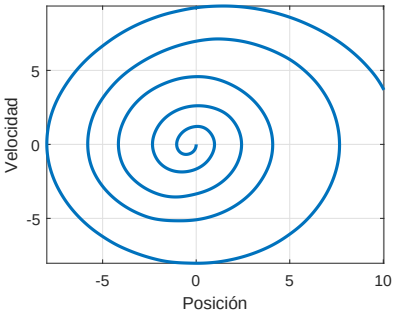
(a) Cota de función de valor



(b) Ley de control



(c) Simulación temporal



(d) Plano de fase

Figura 7: Ley de control con tabla de interpolación 31x31.

Como era de esperar, la cota inferior de la función de valor conseguida con 961 parámetros ajustables es superior (más cercana a la función de valor óptima, seguramente) a la conseguida por la red neuronal 15x15, aunque las simulaciones y las acciones de control asociadas no son demasiado diferentes[3] y nuestra propuesta tiene, básicamente, cuatro veces menos parámetros ajustables.

Para comprobar las ventajas comparativas de nuestra propuesta, se ha implementado el algoritmo de iteración de valor clásico sobre el mismo problema. Obviamente, dado que la tabla de interpolación de 961 parámetros ajustables verifica las condiciones de contractividad necesarias (Busoniu et al., 2010), el algoritmo VI converge, tras aproximadamente 250 iteraciones, básicamente a la misma solución que la que nuestra propuesta encuentra (Figura 7). Sin embargo, con las redes neuronales de 15x15 dichas iteraciones no convergen, por lo que no es posible aplicar el algoritmo VI con regresores con un número de parámetros reducido, mientras que mediante programación lineal sí es posible sin mayor problema.

Ejemplo 3: Planificación de trayectorias.. Este ejemplo considera el robot ABB IRB 360 con configuración Delta de la Figura 8. El objetivo del control óptimo es alcanzar una configuración final desde una configuración inicial, con un control cinemático de la posición (x, y, z) del efector final. El control deberá minimizar un índice de coste inmediato:

$$L(x, e_p) = \|\dot{x}\|^2 + \|\dot{y}\|^2 + \|\dot{z}\|^2 + \|e_p\|^2$$

siendo e_p el error de posición respecto a la posición deseada $(x_d, y_d, z_d) = [-0{,}3 \quad 0{,}1 \quad 0{,}1]^T$ y el factor de descuento se ha establecido en $\gamma = 0{,}999$. El espacio de trabajo ha sido el conjunto $\{-0{,}4 \leq x \leq 0{,}4, -0{,}4 \leq y \leq 0{,}4, 0 \leq z \leq 0{,}24\}$, en el que se ha establecido un mallado de granularidad 11x11x11. Las acciones de control a probar han sido $\{-0{,}25, -0{,}125, 0, +0{,}125, +0{,}25\}$ m/s en cada uno de los tres ejes cartesianos.

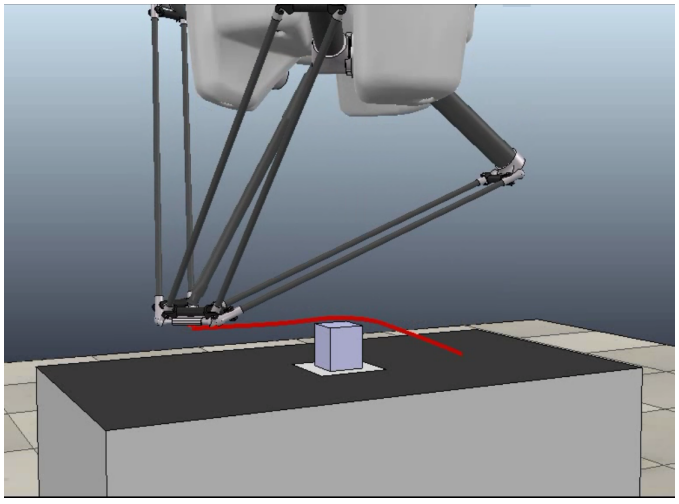


Figura 8: Robot ABB IRB 360

Dicho espacio de trabajo contiene un obstáculo prismático con el que se debe evitar colisiones. Aplicando la metodología de este trabajo, se ha considerado que dicho obstáculo está fuera de la región de operación válida; además, como es habitual en muchas propuestas de planificación de movimientos Latombe (2012), se ha agrandado el tamaño de dicho obstáculo con una celda del mallado anteriormente propuesto, y se ha comprobado con el software V-REP (Rohmer et al., 2013) que la estructura del robot de la figura no colisiona con el obstáculo interno cuando el efector final se encuentra en el borde del prisma ampliado.

Por todo ello, se ha asignado un coste $J_{fuera} = 1e5$ a las configuraciones que abandonan el espacio de trabajo o que colisionan con el prisma agrandado. El regresor elegido ha sido una tabla de interpolación, donde los nodos asociados al obstáculo han sido eliminados para evitar soluciones no acotadas, tal y como se ha discutido en la Sección 3.

La figura 8 presenta en rojo la trayectoria descrita por el efector final del robot desde una posición inicial cuyas coordenadas cartesianas son $[0{,}2 \quad -0{,}2 \quad 0{,}04]^T$.

5. Conclusiones

En este trabajo se ha presentado una metodología para el aprendizaje de controladores óptimos basados en datos. La solución se plantea bajo la perspectiva de la obtención de una cota inferior "aproximada" de la función de valor en un problema de programación dinámica. Esta solución no tiene los problemas de convergencia de otros algoritmos iterativos clásicos.

Se ha planteado un problema basado en programación lineal con regresores genéricos (polinomiales, RBFs, triangulares, etc...). Además se han analizado las condiciones en las que este problema proporcionará soluciones razonablemente adecuadas y bien condicionadas y, por tanto, puedan ser consideradas como una buena aproximación del problema a resolver con unos pocos parámetros, en comparación con implementaciones basadas en interpolación de tablas (con convergencia garantizada a costa de un gran número de parámetros ajustables).

Aunque, en los ejemplos de este trabajo, se han usado redes neuronales RBF equiespaciadas o regresores polinomiales, en cada problema concreto podría contemplarse otra distribución de neuronas u otros tipos de regresores, etc. para intentar obtener prestaciones razonables con un número reducido de parámetros. Estas consideraciones no entran dentro de los objetivos del presente trabajo, ya que dependen de cada problema específico a resolver.

Agradecimientos

Agradecemos al Ministerio de Economía de España, la Unión Europea DPI2016-81002-R (AEI/FEDER, UE), y al Gobierno de Ecuador (Beca SENESCYT) la financiación recibida para la línea de investigación objeto de este trabajo.

[3] La selección del número de neuronas es un compromiso entre calidad de ajuste, simplicidad de la descripción y coste computacional. Se ha escogido 15x15 porque granularidades inferiores aprendían funciones de valor más distorsionadas que no sacaban al carrito del valle, y granularidades superiores se comportaban de modo similar a la presentada.

Referencias

Albertos, P., Sala, A., 2006. Multivariable control systems: an engineering approach. Springer, London, U.K.

Allgöwer, F., Zheng, A., 2012. Nonlinear model predictive control. Vol. 26. Birkhäuser.

Antos, A., Szepesvári, C., Munos, R., 2008. Learning near-optimal policies with bellman-residual minimization based fitted policy iteration and a single sample path. Machine Learning 71 (1), 89–129.

Ariño, C., Pérez, E., Querol, A., Sala, A., 2014. Model predictive control for discrete fuzzy systems via iterative quadratic programming. In: Fuzzy Systems (FUZZ-IEEE), 2014 IEEE International Conference on. IEEE, pp. 2288–2293.

Ariño, C., Pérez, E., Sala, A., 2010. Guaranteed cost control analysis and iterative design for constrained takagi–sugeno systems. Engineering Applications of Artificial Intelligence 23 (8), 1420–1427.

Armesto, L., Girbés, V., Sala, A., Zima, M., Šmídl, V., 2015. Duality-based nonlinear quadratic control: Application to mobile robot trajectory-following. IEEE Transactions on Control Systems Technology 23 (4), 1494–1504.

Busoniu, L., Babuska, R., De Schutter, B., Ernst, D., 2010. Reinforcement learning and dynamic programming using function approximators. CRC press, Boca Raton, FL, USA.

Camacho, E. F., Bordons, C., 2010. Control predictivo: Pasado, presente y futuro. Revista Iberoamericana de Automática e Informática Industrial 1 (3), 5–28.

Condon, A., 1992. The complexity of stochastic games. Information and Computation 96 (2), 203 – 224.

Díaz, H., Armesto, L., Sala, A., 2018. Fitted q-function control methodology based on takagi-sugeno systems. IEEE Transactions on Control Systems Technology, 1–12.

De Farias, D. P., Van Roy, B., 2003. The linear programming approach to approximate dynamic programming. Operations research 51 (6), 850–865.

Deisenroth, M. P., Neumann, G., Peters, J., et al., 2013. A survey on policy search for robotics. Foundations and Trends® in Robotics 2 (1–2), 1–142.

Denardo, E. V., 1970. On linear programming in a markov decision problem. Management Science 16 (5), 281–288.

Duarte-Mermoud, M., Milla, F., 2018. Estabilizador de sistemas de potencia usando control predictivo basado en modelo. Revista Iberoamericana de Automática e Informática industrial 0 (0).

Fairbank, M., Alonso, E., June 2012. The divergence of reinforcement learning algorithms with value-iteration and function approximation. In: The 2012 International Joint Conference on Neural Networks (IJCNN). pp. 1–8.

Gil, R. V., Páez, D. G., 2007. Identificación de sistemas dinámicos utilizando redes neuronales rbf. Revista iberoamericana de automática e informática industrial RIAI 4 (2), 32–42.

Grondman, I., Busoniu, L., Lopes, G. A., Babuska, R., 2012. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 42 (6), 1291–1307.

Hornik, K., Stinchcombe, M., White, H., 1989. Multilayer feedforward networks are universal approximators. Neural Networks 2 (5), 359 – 366.

Kretchmar, R. M., Anderson, C. W., 1997. Comparison of CMACs and radial basis functions for local function approximators in reinforcement learning. In: Neural Networks, 1997., International Conference on. Vol. 2. IEEE, pp. 834–837.

Latombe, J.-C., 2012. Robot motion planning. Vol. 124. Springer.

Lewis, F. L., Liu, D., 2013. Reinforcement learning and approximate dynamic programming for feedback control. Wiley, Hoboken, NJ, USA.

Lewis, F. L., Vrabie, D., 2009. Reinforcement learning and adaptive dynamic programming for feedback control. Circuits and Systems Magazine, IEEE 9 (3), 32–50.

Manne, A. S., 1960. Linear programming and sequential decisions. Management Science 6 (3), 259–267.

Park, J., Sandberg, I. W., 1991. Universal approximation using radial-basis-function networks. Neural computation 3 (2), 246–257.

Rohmer, E., Singh, S. P., Freese, M., 2013. V-rep: A versatile and scalable robot simulation framework. In: Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on. IEEE, pp. 1321–1326.

Rubio, F. R., Navas, S. J., Ollero, P., Lemos, J. M., Ortega, M. G., 2018. Control Óptimo aplicado a campos de colectores solares distribuidos. Revista Iberoamericana de Automática e Informática industrial 0 (0).

Santos, M., 2011. Un enfoque aplicado del control inteligente. Revista Iberoamericana de Automática e Informática Industrial RIAI 8 (4), 283–296.

Si, J., Barto, A. G., Powell, W. B., Wunsch, D., 2004. Handbook of Learning and Approximate Dynamic Programming (IEEE Press Series on Computational Intelligence). Wiley-IEEE Press.

Sutton, R. S., Barto, A. G., 1998. Reinforcement learning: An introduction. Vol. 1. MIT press Cambridge.

Yañez-Badillo, H., Tapia-Olvera, R., Aguilar-Mejía, O., Beltran-Carbajal, F., 2017. Control neuronal en línea para regulación y seguimiento de trayectorias de posición para un quadrotor. Revista Iberoamericana de Automática e Informática Industrial RIAI 14 (2), 141–151.

Ziogou, C., Papadopoulou, S., Georgiadis, M. C., Voutetakis, S., 2013. On-line nonlinear model predictive control of a pem fuel cell system. Journal of Process Control 23 (4), 483–492.