https://doi.org/10.17979/spudc.9788497498081.0710

Aprendizaje por Refuerzo con Búsqueda de Políticas: Simulación y Aplicación a un Sistema Electromecánico

Jose Manuel Pastor, Henry Díaz, Leopoldo Armesto, Antonio Sala
{jopasal3@posgrado hendia@posgrado larmesto@idf asala@isa}.upv.es
Universitat Politècnica de València,
C/Camino de Vera s/n, 46022, Valencia, España

Resumen

La búsqueda de políticas (Policy Search) es un subcampo en el aprendizaje por refuerzo altamente extendido en el ámbito de la automática, ya que sus técnicas permiten trabajar con espacios de estados y acción de orden elevado. Este trabajo se centra en los algoritmos existentes de búsqueda de políticas con modelo-libre basados en métodos por gradiente, aplicándolos tanto en sistemas simulados lineales sencillos como a un sistema electromecánico real formado por un péndulo de 1DoF. El propósito de este estudio es analizar comparativamente sus principales características así como ventajas frente a otros, a la hora de ser aplicados en sistemas más complejos.

Palabras clave Aprendizaje por refuerzo, búsqueda de políticas, control óptimo, gradiente, gradiente natural.

1 Introducción

El Aprendizaje por refuerzo (*Reinforcement Learning, RL*) representa una serie de técnicas diseñadas para resolver problemas de decisión secuencial [20]. La principal idea de este conjunto de técnicas es obtener una política de las acciones del sistema que permita maximizar una señal de recompensa numérica a partir de experimentación [18]. La gran ventaja del aprendizaje por refuerzo reside en su separación con el conocimiento del sistema, de esta forma es posible realizar la mejora de una acción de control basándose únicamente en las señales de recompensa recibidas y sin necesidad de tener un modelo del sistema a controlar [16] [10] [2].

La búsqueda de políticas (PS, del inglés Policy Search), a diferencia de técnicas basadas en la función de valor, se basa en encontrar los parámetros adecuados para la parametrización de una política dada. Los métodos de búsqueda de política más extendidos son los conocidos como modelo-libre (*Model-Free Policy Search*) [3] [4]. Estos métodos basan el aprendizaje en el muestreo de trayectorias y no trabajan con un modelo teórico o estimado. Son muy extendidos en la robótica debido a que, muchas veces, es más sencillo aprender una política que aprender un modelo preciso del sistema [3].

El trabajo está estructurado de la siguiente forma: en la sección 2 se define el problema a resolver. La sección 3 describe los conceptos básicos de aprendizaje por refuerzo basado en la búsqueda de políticas. La sección 4 describe los métodos de política por gradiente y gradiente natural aplicados al aprendizaje de políticas del alto nivel. En la sección 5 se realiza la simulación en sistemas lineales de primer y segundo orden. En la sección 6 se describe la aplicación de los algoritmos de aprendizaje a un péndulo de un grado de libertad y se finaliza con la sección 7 de conclusiones.

2 Definición del problema

El problema en aprendizaje por refuerzo puede formularse como el problema de controlar un agente en un *Procesos de Decisión de Markov (Markov Decision Processes,MDP)* [11] de la forma:

$$x_{t+1} \sim p(x_{t+1}|x_t, u_t) \tag{1}$$

siendo $x_t \in \mathbb{R}^n$ el estado y $u_t \in \mathbb{R}^m$ la acción de control que viene determinada por una política de control en bucle cerrado (determinista o estocástica) definida como:

$$u_t = \pi_\theta(x_t) \tag{2}$$

siendo θ el vector de parámetros de la política $\pi_\theta(x_t)$.

Denotamos a $R(\tau) \in \mathbb{R}$ a la función de recompensa, siendo $\tau = \{x_{0:T}, u_{0:T}\}$ una trayectoria del sistema. Por tanto, la realización de una trayectoria o experimento conlleva un valor asociado de recompensa que permite determinar la calidad de dicha trayectoria. Cuando la trayectoria es evaluada en su conjunto se le denomina también episodio [8] siendo $0 : T$ el intervalo de duración del episodio. En un sistema estocástico como en (1), la función de recompensa esperada, se puede determinar realizando múltiples experimentos con

una misma política, aplicando el operador de Esperanza Matemática:

$$J(\theta) = \mathbb{E}[R(\tau)|\pi] = \int R(\tau)p_\theta(\tau)d\tau \qquad (3)$$

donde $p_\theta(\tau) = \prod_{t=0}^{T-1} p(x_{t+1}|x_t, \pi_\theta(x_t))$ es la probabilidad de distribución de obtener una trayectoria con la política $\pi_\theta \equiv \pi_\theta(x_t)$ y $\mathbb{E}$ representa la esperanza matemática.

El problema de optimización busca la política π_θ^* que maximiza la recompensa esperada. De esta forma, este puede ser formalizado matemáticamente:

$$\pi_\theta^* = \arg\max_{\pi_\theta} J(\theta) \qquad (4)$$

3 Búsquedas de políticas libres de modelo

Los métodos PS libres de modelo se les puede caracterizar considerando su estrategias de evaluación de políticas, estrategias de actualización de políticas [7][8], y sus estrategias de exploración [5].

Las estrategias de exploración determinan cómo son creadas las nuevas trayectorias para el paso siguiente de evaluación de la política. La estrategia de exploración es esencial para la eficiencia de los métodos PS libres de modelo, ya que se requiere crear variabilidad en la generación de trayectorias para obtener un buen grado de exploración y determinar la actualización de la política óptima. Sin embargo, una excesiva exploración incrementa la probabilidad de encontrar puntos de inestabilidad del sistema[9][3][8]. Debido a esto, la mayoría de métodos PS libres de modelo usan políticas estocásticas para la exploración, limitándose a carácteres lineales. Las estrategias de exploración pueden categorizarse en: basadas en pasos (*step-based*) y basadas en episodios (*episode-based*). Mientras la exploración *step-based* usa una acción exploratoria en cada paso(instante) de tiempo, la exploración *episode-based* directamente cambia el vector de parámetros θ de la política solamente al inicio del episodio.

Otra forma de estructurar las estrategias de exploración se basa en el espacio de exploración. La exploración en el espacio de acciones se realiza añadiendo directamente un ruido de exploración ϵ_u a las acciones ejecutadas $u_t = \mu(\theta, t) + \epsilon_u$, las políticas que basadas en pasos descomponen la trayectoria τ en pasos simples $(\boldsymbol{x}_0, \boldsymbol{u}_0, \boldsymbol{x}_1, \boldsymbol{u}_1, ...)$ y evalúan su calidad de las acciones simples. Las estrategias que usan esta exploración se denominan políticas de bajo nivel (*low-level policies*) [3] [19] [10] [22]. Por otra parte la exploración en el espacio de parámetros perturba el vector de parámetros θ al inicio del episodio, en este caso las políticas que se basan en ella se denominan de alto nivel (*upper-level policies*) [12] [6].

La estrategia de evaluación de las políticas determina la forma de evaluación de la calidad de las trayectorias ejecutadas. Siguiendo la clasificación anterior, también se pueden distinguir estrategias de evaluación de política basadas en pasos y basadas en episodios. Las estrategias de evaluación evalúan directamente la trayectoria completa usando el retorno ya sea al final del episodio o a lo largo de toda la trayectoria.

Finalmente, la estrategia de actualización de la política usa la calidad de la evaluación para determinar la actualización de la política. Las estrategias de actualización pueden ser clasificadas de acuerdo al método de optimización usado por el algoritmo PS.

4 Métodos de política de alto nivel por gradiente

En el presente documento se analizan e implementan los métodos basados en episodios tanto para la exploración como evaluación y que usan los métodos de ascensión por gradiente para la optimización en la actualización de la política en alto nivel.

Para ello, (2) es determinista, y se define una nueva política sobre la que se determinan los parámetros de bajo nivel θ a partir de los parámetros de alto nivel ω de una política dada (normalmente una distribución normal $\mathcal{N}(\mu, \sigma)$):

$$\theta \sim \pi_\omega(\theta) \qquad (5)$$

El objetivo pues es obtener ω^* que maximiza:

$$J_\omega = \mathbb{E}_{\pi_\omega}[R(\theta)] \qquad (6)$$

Nótese que, en cualquier caso, para un mismo valor de parámetros de bajo nivel, los valores obtenidos para $R(\theta)$ pueden ser distintos dado que el sistema es estocástico.

Los métodos de política de gradiente (PG,*Policy Gradient*) usan el ascenso del gradiente para maximizar el retorno esperado de J_ω. En el ascenso del gradiente, la dirección de actualización del parámetro está dada por el gradiente $\nabla_\omega J_\omega$ ya que apunta a la dirección más pronunciada del retorno esperado. La actualización de la política del gradiente está dada por tanto como:

$$\pi_{\omega,k+1} = \pi_{\omega,k} + \alpha\nabla_\omega J_\omega \qquad (7)$$

donde α es la tasa de aprendizaje especificado por el usuario y la actualización de PG viene dada por:

$$\nabla_\omega J_\omega = \int_\theta \nabla_\omega \pi_\omega(\theta) R(\theta) d\theta \tag{8}$$

4.1 Política de gradientes de ratio de probabilidad

Estos algoritmos hacen uso del llamado ratio de probabilidad (*likelihood-ratio*) [22] que se sirve de una identidad $\nabla \pi_\omega(y) = \pi_\omega(y) \log \pi_\omega(y)$ para la actualización de la política. Insertando este artificio en la ecuación (8) obtenemos:

$$\begin{aligned} \nabla_\omega J_\omega &= \int \pi_\omega(\theta) \nabla_\omega \log \pi_\omega(\theta) R(\theta) d\theta \\ &= \mathbb{E}_{\pi_\omega(\theta)}[\nabla_\omega \log \pi_\omega(\theta) R(\theta)] \end{aligned} \tag{9}$$

Donde la esperanza matemática sobre $\pi_{\omega(\theta)}$ es aproximada usando una suma sobre las trayectorias $\tau^{[i]} = \{x_0^{[i]}, u_0^{[i]}, x_1^{[i]}, u_1^{[i]} ...\}$ para cada uno de los experimentos realizados con parámetros de bajo nivel distintos $\theta^{[i]}$, que proporcionan una evaluación de la recompensa $R^{[i]}$ asociada al experimento.

Los gradientes resultantes están afectados de una fuerte varianza que puede ser reducida a través de un *base-line* [14] [15] de tal forma que:

$$\nabla_\omega J_\omega = \mathbb{E}_{\pi_\omega(\theta)}[\nabla_\omega \log \pi_\omega(\theta)(R(\theta) - b)] \tag{10}$$

Donde se puede demostrar como aparece en [3] que la PG permanece no desviada siendo b un parámetro libre que es seleccionado de manera que minimice la varianza de la estimación del gradiente.

4.1.1 Algoritmo PEPG (Paremeter Exploring Policy Gradient)

Los métodos de ratio de probabilidad basados en episodios directamente actualizan la política de alto nivel $\pi_\omega(\theta)$ para escoger el vector de parámetros θ de las políticas de bajo nivel $\pi_\theta(u_t|x_t, t)$. Estos métodos optimizan el retorno esperado de J_ω. El gradiente puede ser directamente obtenido a partir de la ecuación (9) resultando en:

$$\nabla_\omega^{PE} J_\omega = \mathbb{E}_{\pi_\omega(\theta)}[\nabla_\omega \log \pi_\omega(\theta)(R(\theta) - b)] \tag{11}$$

El algoritmo Parameter Exploring Parameter (PEPG) introduce *bias* que minimiza la varianza de estimación, de forma que, para cada uno de los componentes del gradientes dicho *bias* se ajusta como sigue:

$$b_h^{PGPE} = \frac{\mathbb{E}_{\pi_\omega(\theta)}[(\nabla_{\omega_h} \log \pi_\omega(\theta))^2 R(\theta)]}{\mathbb{E}_{\pi_\omega(\theta)}[(\nabla_{\omega_h} \log \pi_\omega(\theta))^2]} \tag{12}$$

El pseudocódigo se muestra en el algoritmo 1

Algorithm 1 Parameter Exploring Policy Gradient[3]

Require: Parametrización de la política ω
Require: Conjunto de datos $D = \{\theta^{[i]}, R^{[i]}\}$
1: **para** cada dimensión h de ω **hacer**
Estimar *base-line* óptimo:
2: $b_h^{PGPE} = \frac{\sum_{i=1}^N \left(\nabla_{\omega_h} \log \pi_\omega(\theta^{[i]})\right)^2 R^{[i]}}{\sum_{i=1}^N \left(\log \nabla_{\omega_h} \pi_\omega(\theta^{[i]})\right)^2}$
Derivada estimada para dimensión h de ω:
3: $\nabla_{\omega_h}^{PE} J_\omega = \frac{1}{N} \sum_{i=1}^N \nabla_\omega \log \pi_{\omega_h}(\theta^{[i]})(R^{[i]} - b_h^{PGPE})$
fin bucle

4.2 Políticas de gradiente natural

Los gradientes naturales (NG,*Natural Gradient*) [1] permiten tener una convergencia más rápida que los métodos tradicionales de gradiente para la optimización de los parámetros de una distribución $\pi_\omega(\theta)$. Para ello, estos métodos limitan el cambio máximo que puede existir entre la distribución $\pi_\omega(\theta)$ entre pasos de actualización del aprendizaje, ya que si ésta varía demasiado puede dañar gravemente el aprendizaje.

Para medir la distancia de cambio de la distribución $\delta\omega(\theta)$ se utiliza una aproximación de la Divergencia de Kullback-Leiber (*Kullback-Leiber divergence,KL*). Siendo ésta es una medida de similitud entre dos distribuciones que puede ser aproximada a través de la Matriz de Información de Fisher (*Fisher information matrix*) cuando $\delta\omega(\theta)$ es lo suficientemente pequeño:

$$\mathrm{KL}(\pi_{\omega+\delta\omega}(\theta)||\pi_\omega(\theta)) \approx \delta\omega^T F_\omega \delta\omega. \tag{13}$$

siendo,

$$F_\omega = \mathbb{E}_{\pi_\omega(\theta)}[\nabla_\omega \log \pi_\omega(\theta) \nabla_\omega \log \pi_\omega(\theta)^T] \tag{14}$$

La actualización del gradiente natural $\delta\omega^{NG}$ es definida como la actualización $\delta\omega$ que es más similar a la actualización del gradiente tradicional o *"vanilla"* $\delta\omega^{VG}$ que tiene una limitación de distancia en el espacio de la distribución tal que:

$$\mathrm{KL}(\pi_{\omega+\delta\omega}(\theta)||\pi_\omega(\theta)) \leq \epsilon \tag{15}$$

De esta forma, se puede definir la optimización como:

$$\delta\omega^{NG} = \arg\max_{\pi_\theta} \delta\omega^T \delta\omega^{VG} \quad \text{s.a. } \delta\omega^T F_\omega \delta\omega \leq \epsilon \tag{16}$$

Los gradientes naturales evitan convergencias prematuras en marcos donde existan pasos de actualización muy agresivos o máximos muy abruptos gracias a sus propiedades de convergencia isotrópicas [1] [17].

La aplicación de estos gradientes naturales tiene un gran grado de utilidad en el campo búsqueda de políticas de aprendizaje por refuerzo. Se define el gradiente natural para políticas de alto nivel como sigue:

$$\nabla_{\omega}^{NG} J_{\omega} = F_{\omega}^{-1} \nabla_{\omega} J_{\omega} \tag{17}$$

Donde $\nabla_{\omega} J_{\omega}$ puede ser estimado a través de cualquier método tradicional de gradiente.

4.2.1 Algoritmo NES(Natural Evolution Estrategy)

Las propiedades de los gradientes naturales pueden ser aplicadas satisfactoriamente a algoritmos con actualización y evaluación episódica [17] [21]. Aunque estos algoritmos provienen del campo de Algoritmos Evolutivos siguen realizando un ascenso en el gradiente de una función de forma, que en el caso del aprendizaje por refuerzo es la recompensa esperada global J_{ω} la función a maximizar. Es por esto que estos algoritmos pueden ser categorizados como métodos de política de gradiente.

Estos algoritmos se basan en el cálculo del gradiente tradicional (11) y la matriz de información de Fisher resultando en la siguiente expresión:

$$\nabla_{\omega}^{NES} = F_{\omega}^{-1} \nabla_{\omega}^{PE} J_{\omega} \tag{18}$$

donde la matriz de información de Fisher para las políticas de alto nivel se obtiene a partir de:

$$F_{\omega} = E_{\pi_{\omega}(\theta)}[\nabla_{\omega} \log \pi_{\omega}(\theta) \nabla_{\omega} \log \pi_{\omega}(\theta)^T] \tag{19}$$

La primera vez que se aplicaron los gradientes naturales en espacio de parámetros (política de alto nivel) fue en el algoritmo *NES (Natural Evolution Strategy)* [21] donde la matriz de información de Fisher fue calculada empíricamente, sin embargo, esta obtención empírica es problemática al provocar que la matriz no sea invertible debido a redundancias en el parámetro del error. Para solucionar esto en [17] se presenta una solución cerrada de la matriz de información de Fisher para políticas Gaussianas en el espacio de parámetros que será usada en el algoritmo NES.

La comparación entre los algoritmos PEPG y NES puede verse en [13] donde se indica que el algoritmo NES es más eficiente para problemas de baja dimensión mientras que el algoritmo PEPG tiene ventajas para problemas de mayor dimensión ya que la actualización de segundo orden del gradiente natural incrementa en dificultad.

El pseudocódigo se muestra en el algoritmo 2

Algorithm 2 Natural Evolution Estrategy

Require: Parametrización de la política ω
Require: Conjunto de datos $D = \{\theta^{[i]}, R^{[i]}\}$
1: **for** para cada dimensión h de ω **hacer**
2: Estimar *base-line* óptimo:
3: $b_h^{PGPE} = \frac{\sum_{i=1}^{N} \left(\nabla_{\omega_h} \log \pi_{\omega}(\theta^{[i]})\right)^2 R^{[i]}}{\sum_{i=1}^{N} \left(\nabla_{\omega_h} \log \pi_{\omega}(\theta^{[i]})\right)^2}$
4: Derivada estimada para dimensión h de ω:
5: $\nabla_{\omega_h}^{PE} J_{\omega} = \frac{1}{N} \sum_{i=1}^{N} \log \nabla_{\omega} \pi_{\omega_h}(\theta^{[i]})(R^{[i]} - b_h^{PGPE})$
6: $F_{\omega} = \frac{1}{N} \sum_{i=1}^{N} [\nabla_{\omega} \log \pi_{\omega}(\theta^{[i]}) \nabla_{\omega} \log \pi_{\omega}(\theta^{[i]})^T]$
7: $\nabla_{\omega}^{NES} = F_{\omega}^{-1} \nabla_{\omega}^{PE} J_{\omega}$
fin for

5 Simulaciones implementadas

Las simulaciones realizadas corresponden a un sistema lineal de primer orden y a un sistema lineal de segundo orden en los cuales la dinámica del sistema no es conocida. Se ha definido la ley control como una realimentación lineal del estado $u(x) = Kx$. Los modelos presentados solo se usan para generar las trayectorias pero no intervienen en el algoritmo de aprendizaje. Las políticas aprendidas por los algoritmos implementados se han comparado con las políticas óptimas del LQR(*Linear-quadratic regulator*).

Los modelos de los sistemas simulados considerados tienen la forma $x_{k+1} = Ax_k + Bu_k$.

Y la recompensa esperada se ha definido como una función exponencial:

$$R(\theta) = e^{-\beta \frac{1}{N} \sum_{i=1}^{N} \frac{1}{2}(x_i^T Q x_i + u_i^T R u_i)} \tag{20}$$

donde β es un factor de reescalado de la función exponencial, y Q es una matriz $n \times n$ semidefinida positiva y R una matriz $m \times m$ definida positiva.

5.1 Sistema lineal de primer orden

El sistema utilizado en la simulación:

$$A = [1] \quad B = [1] \tag{21}$$

y para el coste se han utilizado con $Q = [1]$ y $R = [0.1]$.

Se han aplicado ambos algoritmos al sistema anterior, con una inicialización de la política Gaussiana de alto nivel $\pi_{\omega} = \mathcal{N}(\mu, \sigma)$ en parámetros $\mu = 0.5$ y $\sigma = 0.3$ y un valor en el parámetro

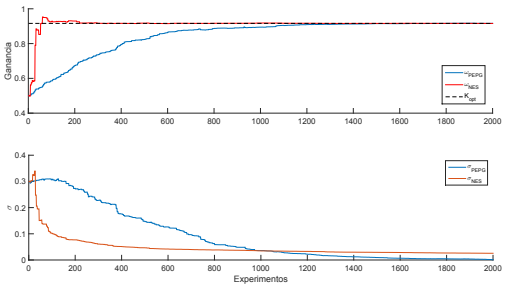

Figura 1: Política de alto nivel aprendida usando los algoritmos NES y PEPG en un sistema lineal de primer orden.

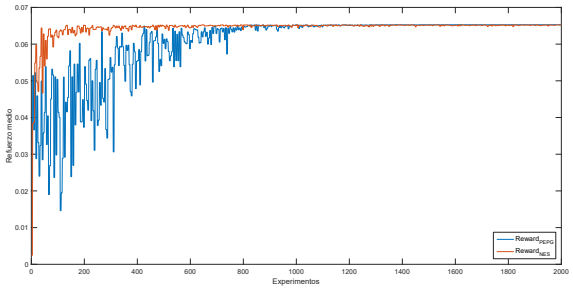

Figura 2: Comparación del refuerzo medio obtenido mediante los algoritmos NES y PEPG en un sistema lineal de primer orden.

$\beta = 0.05$ en la función exponencial del refuerzo. Además se ha definido un valor inicial del estado de $x_0 = 10$.

Para la simulación se han realizado 500 iteraciones del algoritmo y el set de experimentos simulados en cada iteración ha sido de 4.

Observando los resultados obtenidos se puede verificar que para sistemas de bajo número de dimensiones el Algoritmo NES presenta mejores prestaciones[13], converge alrededor del experimento 300 en comparación con el algoritmo PEPG que los hace alrededor del experimento 1300. En la Figura 1 se muestra el proceso de aprendizaje de los algoritmos PEPG y NES hasta alcanzar la política óptima LQR.

La Figura 2 muestra la evolución del refuerzo medio obtenido. Obervando esta figura se puede establecer que el algoritmo PEPG es mucho más susceptible al ruido del sistema ya que al no estar su gradiente limitado por la divergencia KL, este toma valores muy dispares en los primeros experimentos provocando que tarde más en aprender la solución óptima del sistema.

Lo mismo sucede en los experimentos finales, donde el ruido provoca en el algoritmo PEPG una separación de la política óptima debido al su efecto en el gradiente, mientras que en el algoritmo NES este efecto queda minimizado al estar limitado el cambio en el gradiente entre actualizaciones de política.

5.2 Sistema de segundo orden

El sistema de segundo orden que se ha utilizado para la simulación es:

$$A = \begin{bmatrix} 0.9 & 0.1 \\ 0 & 0.8 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{22}$$

Con Q y R como:

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad R = [2] \tag{23}$$

Los parámetros de la política de alto nivel a aprender son $\pi_\omega = \mathcal{N}(\mu, \Sigma)$ donde $\mu = [\mu_1, \mu_2]$ y $\Sigma = [\sigma_1, \sigma_2]$. Se han inicializado como $\omega = N([0.5, 0.15], [0.1, 0.1])$. Se ha seleccionado un valor en el parámetro $\beta = 0.05$ en la función exponencial del refuerzo y con un valor inicial del estado de:

$$\mathbf{x}_0 = \begin{bmatrix} 10 \\ 10 \end{bmatrix} \tag{24}$$

Se ha realizado 500 iteraciones de cada algoritmo, en cada iteración se han realizado 4 experimentos.

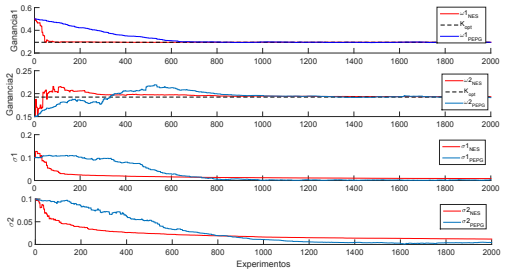

Figura 3: Política de alto nivel aprendida usando los algoritmos NES y PEPG en un sistema lineal de segundo orden.

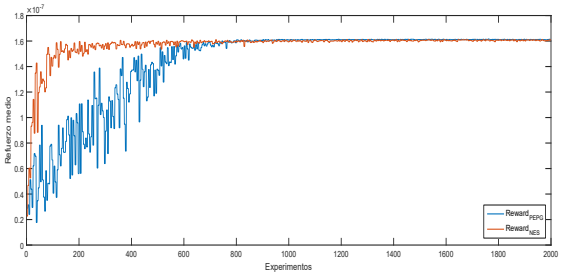

Figura 4: Comparación del refuerzo medio obtenido mediante los algoritmos NES y PEPG en un sistema lineal de segundo orden.

En este caso al igual que para el primer orden se puede apreciar que el algoritmo NES aprende la política óptima mucho antes Figuras 3 y 4. En concreto para este caso, consigue converger a las ganancias óptimas LQR alrededor del experimento 250 mientras que el algoritmo PEPG no alcanza esos valores de convergencia hasta el experimento 800.

Finalmente, en esta simulación de segundo orden también se aprecia lo presentado en [3] sobre la varianza de los gradientes naturales, esta tarda en cerrarse permitiendo encontrar antes las ganancias óptimas del sistema.

6 Experimentación real: Péndulo de un grado de libertad(1DoF)

Las implementaciones reales de los algoritmos se han hecho sobre un banco de experimentos mostrado en la Figura 5. El banco de pruebas esta formado por un péndulo de 1DoF cuyo actuador es un motor DC, el péndulo posee un sensor de posición de efecto hall en el primer eslabón.

El experimento diseñado se ha definido como la búsqueda del controlador PD para una trayectoria del péndulo desde una posición inicial de 30º hasta su posición de reposo en 0º.

Al igual que para el caso anterior se ha definido la recompensa como una función exponencial parametrizada por β:

$$R(\beta) = e^{-\beta J_\omega} \tag{25}$$

Por lo tanto, se deberá elegir aquel vector de parámetros θ que minimice la función de coste J_ω:

$$J_\omega = \sum_{i=1}^{\tau} x_{[i]}^T Q x_{[i]} + u_{[i]}^T R u_{[i]} \tag{26}$$

Donde x es el estado formado por la posición y velocidad del péndulo, u la acción de control y se definen Q y R como:

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \quad R = [0.1] \tag{27}$$

Para comprobar las actuaciones de los algoritmos se ha generado una superficie con un barrido de catorce valores en los dos parámetros del controlador obteniendo la recompensa en cada par de ganancias para un $\beta = 5 \times 10^{-5}$.

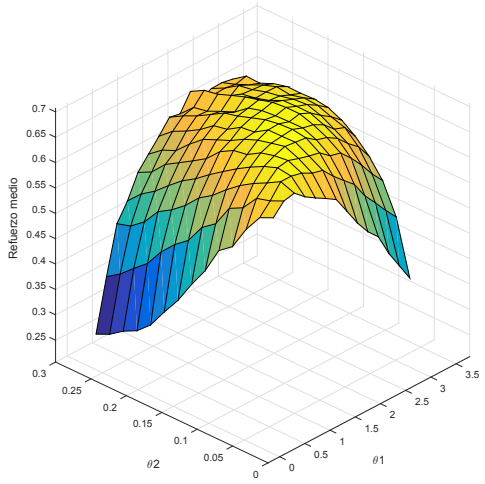

Figura 6: Superficie obtenida para el barrido de ganancias realizada con un promedio de 4 experimentos por par de ganancias

Se observa en la Figura 6 que la superficie presenta los máximos de recompensa en su zona central. Obviamente, esa superficie se supone *desconocida* para el algoritmo de búsqueda de política, y deberá intentar acercarse a su máximo mediante la experimentación y aprendizaje.

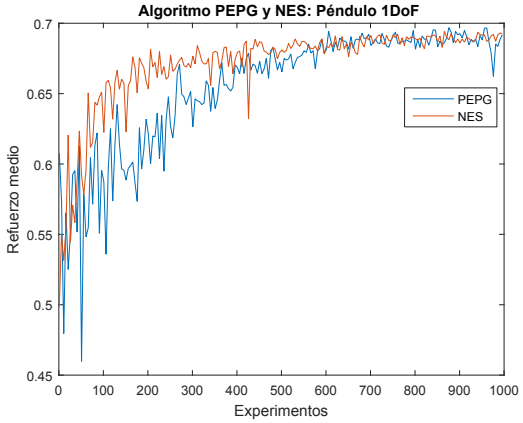

Figura 7: Comparación del Refuerzo medio obtenido usando los algoritmos de aprendizaje PEPG y NES sobre un péndulo de 1DoF.

En la Figura 7 se muestra el refuerzo medio obtenido de cada uno de los algoritmos PEPG y NES aplicados en el péndulo de un grado de libertad (1DoF), se realizaron 200 iteraciones del algoritmo cada una de las iteraciones con 5 experimentos. De manera práctica se puede verificar que el refuerzo medio converge más rápido usando el algoritmo NES comparado con PEPG.

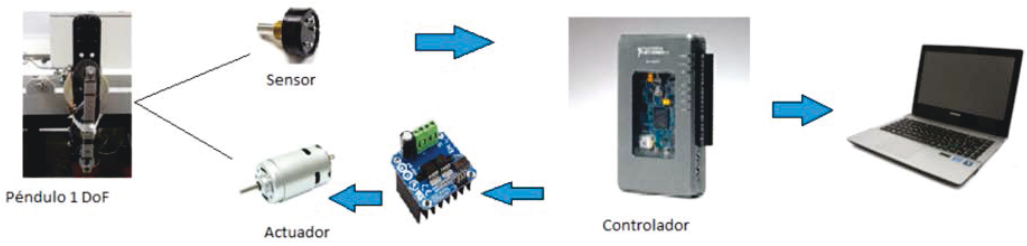

Figura 5: Sistema implementado para pruebas de aprendizaje usando PS.

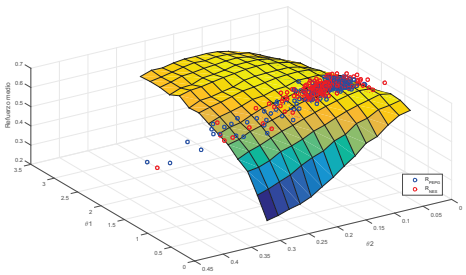

Figura 8: Refuerzos obtenidos respecto a las ganancias aprendidas aplicando los algoritmos PEPG y NES al péndulo de 1DOF.

En la Figura 8 se muestra la superficie del coste generada al realizar el barrido de ganancias y sobre esta superficie el coste de las ganancias obtenidas por los dos algoritmos de aprendizaje aplicados en el péndulo real. El mejor coste obtenido en el barrido de ganancias en la superficie corresponde a las ganacias $\theta = [1.458, 0.0963]$, los valores de las ganancias aprendidas por el algoritmo NES son $\theta = [1.2954, 0.0956]$ y las ganancias del algoritmo PEPG son $\theta = [1.4392, 0.0991]$, en ambos casos los valores obtenidos se encuentran en las proximidades al valor de las ganancias correspondientes al mejor coste en el barrido realizado.

7 Conclusiones

En este trabajo se ha presentado la implementación de algoritmos de aprendizaje por refuerzo basados en la búsqueda de política. Se ha realizado simulaciones para sistemas lineales de primer y segundo orden, así como también la aplicación de estos algoritmos en un péndulo de un grado de libertad, obteniendo como resultado la convergencia de las ganancias aprendidas a las ganancias del LQR en el caso de las simulaciones y en el caso real las ganancias aprendidas se encuentran cercanas a las ganancias que producen el mejor coste usando un barrido de ganancias.

Se ha verificado que el uso del gradiente natural(NES) en la actualización de la política consigue una convergencia mucho más rápida en comparación con las políticas de gradiente(PEPG), al menos en los experimentos y las decisiones de parámetros del algoritmo aquí probados.

La gran cantidad de experimentos necesarios para aprender crean la necesidad de mejorar o usar otros algoritmos en los cuales se pueda reducir la cantidad de experimentos, esto debido a que en el caso de bancos de pruebas reales se pueden causar daños o averías en el proceso de aprendizaje por la gran cantidad de experimentos necesarios y el tiempo que conlleva realizar la experimentación.

Agradecimientos

Los autores agradecen al Gobierno de España (DPI2011-27845-C02-01 y DPI2013-42302-R9), a la Generalitat Valenciana (PROMETEOII/2013/004) y al Gobierno de Ecuador (Beca SENESCYT) la financiación recibida para llevar a cabo la investigación aquí presentada.

Referencias

[1] Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

[2] Andrew G Barto, Richard S Sutton, and Chris JCH Watkins. Learning and sequential decision making. In *Learning and computational neuroscience*. Citeseer, 1989.

[3] Marc Peter Deisenroth, Gerhard Neumann, Jan Peters, et al. A survey on policy search for robotics. *Foundations and Trends in Robotics*, 2(1-2):1–142, 2013.

[4] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learn-

ing: A survey. *Journal of artificial intelligence research*, pages 237–285, 1996.

[5] Jens Kober, Erhan Oztop, and Jan Peters. Reinforcement learning to adjust robot movements to new situations. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 2650, 2011.

[6] Jens Kober and Jan R Peters. Policy search for motor primitives in robotics. In *Advances in neural information processing systems*, pages 849–856, 2009.

[7] Jan Peters, Katharina Mülling, and Yasemin Altun. Relative entropy policy search. In *AAAI*, 2010.

[8] Jan Peters and Stefan Schaal. Policy gradient methods for robotics. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 2219–2225. IEEE, 2006.

[9] Jan Peters and Stefan Schaal. Reinforcement learning of motor skills with policy gradients. *Neural networks*, 21(4):682–697, 2008.

[10] Jan Peters, Sethu Vijayakumar, and Stefan Schaal. Reinforcement learning for humanoid robotics. In *Proceedings of the third IEEE-RAS international conference on humanoid robots*, pages 1–20, 2003.

[11] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[12] Thomas Rückstieß, Martin Felder, and Jürgen Schmidhuber. State-dependent exploration for policy gradient methods. In *Machine Learning and Knowledge Discovery in Databases*, pages 234–249. Springer, 2008.

[13] Thomas Rückstiess, Frank Sehnke, Tom Schaul, Daan Wierstra, Yi Sun, and Jürgen Schmidhuber. Exploring parameter space in reinforcement learning. *Paladyn, Journal of Behavioral Robotics*, 1(1):14–24, 2010.

[14] Frank Sehnke, Christian Osendorfer, Thomas Rückstieß, Alex Graves, Jan Peters, and Jürgen Schmidhuber. Policy gradients with parameter-based exploration for control. In *Artificial Neural Networks-ICANN 2008*, pages 387–396. Springer, 2008.

[15] Frank Sehnke, Christian Osendorfer, Thomas Rückstieß, Alex Graves, Jan Peters, and Jürgen Schmidhuber. Parameter-exploring policy gradients. *Neural Networks*, 23(4):551–559, 2010.

[16] Masashi Sugiyama. *Statistical Reinforcement Learning: Modern Machine Learning Approaches*. CRC Press, 2015.

[17] Yi Sun, Daan Wierstra, Tom Schaul, and Juergen Schmidhuber. Efficient natural evolution strategies. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 539–546. ACM, 2009.

[18] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[19] Richard S Sutton, David A McAllester, Satinder P Singh, Yishay Mansour, et al. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*, volume 99, pages 1057–1063. Citeseer, 1999.

[20] Marco Wiering and Martijn van Otterlo. *Reinforcement Learning: State-of-the-art*, volume 12. Springer Science & Business Media, 2012.

[21] Daan Wierstra, Tom Schaul, Jan Peters, and Juergen Schmidhuber. Natural evolution strategies. In *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on*, pages 3381–3387. IEEE, 2008.

[22] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.