

Master Degree Thesis:
**Aprendizaje por refuerzo aplicado al problema del
pastoreo**

Jesus Oroya

Advisor: Roberto Santana
Intelligent Systems Group,
Department of Computer Science and Artificial Intelligence,
University of the Basque Country UPV/EHU,
Paseo Manuel de Lardizabal, 1
Donostia, 20018 Gipuzkoa, Spain
djoroya@gmail.com

Índice general

Master Degree Thesis:	1
Jesus Oroya	

PARTE I

Aprendizaje por refuerzo

1. Proceso de decisión de Markov	9
1.1. Definición formal	9
1.2. Ejemplo del control de una partícula	11
1.3. Maximización de la recompensa esperada	13
2. Programación dinámica	14
2.1. Funciones valor estado	14
2.2. Funciones valor de estado-acción	16
2.3. Operadores de Bellman (OB)	17
3. Algoritmos de aprendizaje por refuerzo	18
3.1. Value Iteration	18
3.2. <i>Q-Iteration</i>	19
3.3. <i>Q-learning</i>	19
4. Movimiento de una partícula en un potencial	20
4.1. Descripción del problema	20
4.2. Comparación entre algoritmos	22

PARTE II

Problema de pastoreo de ovejas

5. Modelo de Interacción	26
5.1. Notación	27
5.2. Hipótesis del modelo	27
5.3. Fuerzas de interacción	28
5.4. Modelo dinámico en tiempo continuo	29
5.5. Modelo dinámico en tiempo discreto	30
6. Aprendizaje por refuerzo para el problema del pastoreo	31

6.1. Proceso de decisión de Markov	31
6.2. Metodología <i>Deep Deterministic Policy Gradient</i>	32
7. Política Heurística y pre-entrenamiento del actor	34
7.1. Política Heurística	34
7.2. Pre-entrenamiento	36
8. Resultados numéricos	36
9. Conclusiones	36

Índice de figuras

1. Proceso de decisión de Markov.	12
2. Evolución del sistema bajo una política.	13
3. Comportamiento de la partícula en un potencial	23
4. Espacio de fases modificado	24
5. Fuerzas de Interacción entre Individuos	29
6. Ejemplo de la dinámica	30
7. Comportamiento de la política heurística.	35
8. Evolución temporal de la recompensa	37
9. Evolución temporal de la posición de las ovejas	37

Índice de cuadros

1. Notación del estado y acciones del sistema	27
2. Fuerzas de interacción	29

Introducción

Muchas de las tareas que se nos presentan cada día pueden considerarse como problemas de optimización dinámica. En estos problemas existe un agente que toma decisiones dependiendo de las observaciones que recoge en cada momento. Un ejemplo representativo es la conducción humana. En este problema el agente que conduce debe tomar una serie de decisiones para llegar a su destino. En este caso es evidente que la experiencia ayuda a optimizar la conducción¹. Este sencillo ejemplo nos muestra como el proceso de aprendizaje debe ser entendido. La prueba y error de un agente que intenta encontrar la mejor estrategia. Esta idea es la semilla que motivo el desarrollo del aprendizaje por refuerzo.

El aprendizaje por refuerzo se remonta al artículo [13], donde se presentó el algoritmo de *Q-learning*. Este es una modificación de algoritmo de programación dinámica *Q-Iteration*, donde se considera que la naturaleza del problema es desconocida. A cambio de este conocimiento, el problema nos permite tomar medidas de la evolución del sistema. *Q-learning* es el primer algoritmo capaz de aprender estrategias óptimas mediante la experiencia. Gracias a este algoritmo somos capaces de desarrollar mejores estrategias a partir de alguna previamente conocida.

En este trabajo utilizaremos esta característica para mejorar la estrategia de un agente que tiene como objetivo pastar ovejas. El problema de pastoreo se define como: Suponiendo que tenemos un conjunto de ovejas y el control de unos pocos perros, ¿Cómo debería ser el movimiento estos perros para llevar a las ovejas a un punto deseado? Este problema es claramente un problema de optimización dinámica donde los perros deberán tomar decisiones dependiendo de las observaciones que tomen las ovejas. Utilizaremos la metodología de aprendizaje por refuerzo para obtener estrategias para el movimiento de los perros

En esta tesis nos centraremos en entender el aprendizaje por refuerzo y su aplicación en el problema de pastoreo. Debido a esto, esta tesis esta estructurada en dos parte principales:

1. **Aprendizaje por refuerzo:** la primera un parte es el marco teórico donde nos hemos centrado en exponer de una manera muy básica los elementos clave en el aprendizaje por refuerzo, de forma que podamos entender como y por qué funcionan los algoritmos emblemáticos como *Value Iteration* o *Q-learning*. Allí se presentan ejemplos básicos que permiten el entendimiento de los elementos presentados.
2. **Problema de pastoreo:** En esta segunda parte se aplica el aprendizaje por refuerzo para resolver el problema de pastoreo. Se propone una reducción de

¹ Cada una de las persona puede tener una distinta noción de optimización. Por ejemplo, para algunas personas, los pasos óptimos serán los que les permita llegar más rápido a su destino y para otras será más óptimo las estrategias que les consuma menos gasolina

variables con el fin de agregar robustez a la estrategia, además de un pre-entrenamiento de un red neuronal que define la estrategia con ayuda de una estrategia heurística.

Parte I

Aprendizaje por refuerzo

El aprendizaje por refuerzo es una técnica de aprendizaje automático capaz de resolver problemas de forma dinámica. Esta técnica se define como un proceso iterativo donde existe un agente capaz de tomar decisiones y un sistema dinámico influenciado por este. Debido a la interacción entre el agente y el sistema dinámico, el agente puede conducir el sistema a configuraciones de interés. Estas configuraciones vienen caracterizadas por una función de coste que es máxima cuando la configuración deseada es alcanzada. Por otra parte, esta metodología utiliza el principio de la programación dinámica para obtener una estrategia de actuación, sin necesidad de conocer la ecuación del sistema dinámico; esta es la característica principal del aprendizaje por refuerzo.

A lo largo de este capítulo describiremos formalmente el aprendizaje por refuerzo como un proceso de decisión de Markov. A continuación introduciremos la programación dinámica como método para abordar el problema presentado. Por último, mostraremos algoritmos estándar para abordar el problema presentado y ejemplos concretos para estos algoritmos.

1. Proceso de decisión de Markov

Un proceso de decisión de Markov es un proceso dinámico en tiempo discreto y probabilístico capaz de modelizar una gran variedad de procesos. Este marco matemático fue introducido en la década de 1950 en el artículo [3].

1.1. Definición formal

Definición 1 (Proceso de decisión de Markov) Se define como una tupla de cuatro objetos:

- Una variable aleatoria $S_t \in \mathcal{S}$, que representa el estado en la iteración t .
- Una variable aleatoria $A_t \in \mathcal{A}$, que representa la acción en la iteración t .
- Una variable aleatoria $R_t \in \mathcal{R}$, que representa la recompensa obtenida en la iteración t .
- Una distribución de probabilidad:

$$\mathbb{P}(S_t = s', R_t = r | S_{t-1} = s, A_t = a) = p(s', r | s, a) \quad (1.1)$$

Donde la función $p : \mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ será llamada la **función dinámica** del proceso de decisión de Markov. Esta representa la probabilidad de transitar al estado s' y obtener la recompensa r , habiendo estado en el estado s y habiendo tomado la acción a . $\square$

Observación 1 Aunque el espacio de recompensas puede ser infinito, a lo largo de este capítulo consideraremos que el espacio de recompensas $\mathcal{R}$ es un espacio discreto y finito. Debido a esto siempre podemos sumar en todo el espacio $\mathcal{R}$.

Por otro lado, existen otras distribuciones de probabilidad que pueden ser obtenidas de la función dinámica. Estas distribuciones son la distribución de probabilidad de transición $p_t(s'|s, a)$ y la distribución de probabilidad de recompensa $p_r(r|s, a)$.

La distribución de probabilidad de transición $p_t(s'|s, a)$ se puede obtener como:

$$p_t(s'|s, a) = \sum_{r \in \mathcal{R}} p(s', r|s, a) \quad (1.2)$$

Mientras que la distribución de probabilidad de recompensa $p_r(r|s, a)$ se obtiene de la siguiente manera:

$$p_r(r|s, a) = \sum_{s' \in \mathcal{S}} p(s', r|s, a) \quad (1.3)$$

Por otra parte, podemos obtener la recompensa esperada debido a la acción a y estando en el estado s como una función $\rho : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{R}$ tal que:

$$\rho(s, a) = \mathbb{E}[R_t | S_{t-1} = s, A_t = a] = \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} r p(s', r|s, a) \quad (1.4)$$

Aunque en la definición formal de un proceso de decisión de Markov es necesario definir la distribución de probabilidad de la dinámica $p(s', r|s, a)$, es habitual definir el proceso de decisión de Markov mediante la distribución de probabilidad de transición $p_t(s'|s, a)$ y la distribución de probabilidad recompensa $p_r(r|s, a)$. Además también es posible definir un proceso de decisión de Markov, mediante la recompensa esperada $\rho(s, a)$ y la distribución de probabilidad de transición $p_t(s'|s, a)$. Estas tres formas de definir los procesos de decisión de Markov son equivalentes, sin embargo dependiendo de la aplicación una es más conveniente que otra (véase el ejemplo de la sección 1.2).

Con respecto al comportamiento del agente, se define la política para su caracterización.

Definición 2 (Política) Es una función $\pi : \mathcal{S} \rightarrow \mathcal{A}$, que determina las acciones que se tomará en cada iteración de proceso de decisión de Markov dependiendo en el estado en el que se encuentre el sistema. Esta función π vive en un espacio de funciones que llamaremos Π . $\square$

Dadas estas definiciones previas podemos formular cualquier proceso de decisión de Markov. Con el objetivo ejemplificar las definiciones presentadas anteriormente veremos un ejemplo.

1.2. Ejemplo del control de una partícula

Sea el espacio de estados $\mathcal{S} = \{s_1, s_2, s_3\}$ y el espacio de acciones $\mathcal{A} = \{\leftarrow, =, \rightarrow\}$. Además sea la distribución de probabilidad de transición $p(s', r|s, a)$, donde suponemos que la variable aleatoria recompensa r y la variable aleatoria de estado siguiente s' son independientes. Entonces podemos factorizar la distribución de probabilidad de la dinámica:

$$p(s', r|s, a) = p_r(r|s, a)p_t(s'|s, a) \quad (1.5)$$

En este caso podemos definir la distribución $p_r(r|s, a)$ y la distribución $p_t(s'|s, a)$ por separado. Definiremos estas dos distribuciones de forma determinista. Para ello utilizaremos la función delta de Kronecker δ definida como:

$$\delta(x, y) = \begin{cases} 0 & x \neq y \\ 1 & x = y \end{cases}$$

En primer lugar, definiremos la distribución de probabilidad de transición $p_t(s'|s, a)$. Para ello asociaremos cada uno de los estados $\{s_1, s_2, s_3\}$ a los vectores unitarios de $\mathbb{R}^3$:

$$\{s_1, s_2, s_3\} \rightarrow \{(1, 0, 0), (0, 1, 0), (0, 0, 1)\} \quad (1.6)$$

Definimos la función $f_s : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ como:

$$f_s(s, a) = \delta(a, \leftarrow) \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} s + \delta(a, =) \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} s + \delta(a, \rightarrow) \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} s \quad (1.7)$$

Con ayuda de esta función $f_s(s, a)$ podemos definir la distribución de probabilidad $p_t(s'|s, a)$ como:

$$p_t(s'|s, a) = \delta(s', f_s(s, a)) \quad (1.8)$$

En la Figura (1) se muestra una representación gráfica de la distribución de probabilidad de transición. Allí podemos ver que esta definición simplemente representa una partícula que puede moverse en una recta con tres posiciones y

que las acciones que podemos tomar representan un movimiento a la derecha, un movimiento a la izquierda o quedarnos en la posición en la que nos encontramos (véase la Figura (2)).

En segundo lugar, definiremos la distribución de probabilidad de recompensa $p_r(r|s, a)$. Para ello definimos la función $f_r : \mathcal{S} \times \mathcal{A} \rightarrow \{-2, -1, 9, 10\}$, como:

$$f_r(s, a) = \begin{cases} 10 - \delta(a, \rightarrow) - \delta(a, \leftarrow) & s = s_2 \\ -1 - \delta(a, \rightarrow) - \delta(a, \leftarrow) & s \neq s_2 \end{cases} \quad (1.9)$$

Entonces podemos definir la distribución de probabilidad $p_r(r|s, a)$ como:

$$p_r(r|s, a) = \delta(r, f_r(s, a)) \quad (1.10)$$

En este caso, suponemos que el hecho de moverse penaliza en una unidad la recompensa. Además que el sistema alcanza el máximo en la posición $s = s_2$

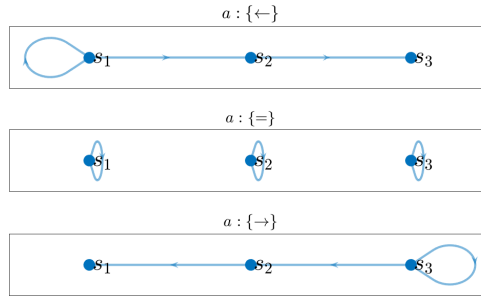


Figura 1. Proceso de decisión de Markov.

Para cada posible acción se puede definir una matriz que a su vez se puede representar como un grafo.

Por otra parte podemos calcular la recompensa esperada para este sistema:

$$\rho(s, a) = \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} p(s', r|s, a) = \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} p_t(s'|s, a) p_r(r|s, a) \quad (1.11)$$

Utilizando las ecuaciones (1.10) y (1.8):

$$\rho(s, a) = \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} \delta(s', f_r(s, a)) \delta(r, f_r(s, a)) = f_r(s, a) \quad (1.12)$$

Se puede comprobar que $\rho(s, a) = f_r(s, a)$, por lo que en este ejemplo definir la distribución de probabilidad de recompensa $p_r(r|s, a)$ o la recompensa esperada

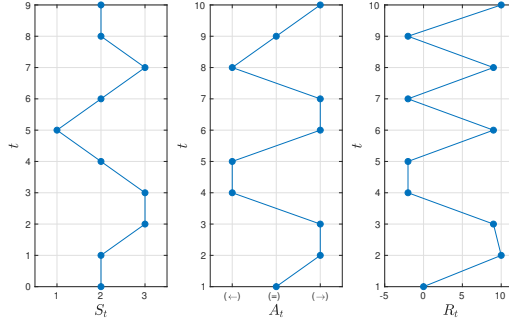


Figura 2. Evolución del sistema bajo una política.

De una manera intuitiva, el agente actúa en el sistema como si tuviera un mando derecha-izquierda.

$\rho(s, a)$ son equivalentes.

Podemos notar que la distribución de probabilidad de transición $p_t(s'|s, a)$ y la distribución de probabilidad de recompensa $p_r(r|s, a)$, caracteriza completamente el proceso de decisión de Markov.

1.3. Maximización de la recompensa esperada

El aprendizaje por refuerzo tiene como objetivo la maximización el retorno esperado con respecto a la política $\pi \in \Pi$. Esta maximización se puede formular de la siguiente manera:

Problema 1 Definimos el **problema de maximización** como:

$$\max_{\pi \in \Pi} \mathbb{E}_{\pi} \left[\sum_{\tau=0}^T \gamma^{\tau} R_{\tau+1} \middle| S_0 = s \right] \quad (1.13)$$

Donde $\mathbb{E}_{\pi}$ representa la esperanza con respecto a todas las posibles trayectorias en el espacio de estados tomando la política π , y donde $\gamma \in [0, 1]$.

Observación 2 El factor de descuento γ impide que el retorno esperado diverja cuando el horizonte temporal es $T = \infty$, siempre que las recompensas estén acotadas en todo el espacio $\mathcal{R}$. Un segundo propósito de este factor es dar más importancia las recompensas obtenidas en un tiempo cercano.

La solución a este problema también tiene un carácter especial, por que lo definiremos.

Definición 3 Definimos la **política óptima**, π_* como:

$$\pi_* \in \arg \max_{\pi \in \Pi} \mathbb{E}_{\pi} \left[\sum_{\tau=0}^T \gamma^{\tau} R_{\tau+1} \middle| S_0 = s \right] \quad (1.14)$$

□

Entonces un proceso de decisión de Markov en síntesis consiste maximizar el retorno esperado con respecto a la política $\pi \in \Pi$ y sujeto a la distribución de probabilidad de la dinámica $p(s', r|s, a)$.

2. Programación dinámica

La programación dinámica es una metodología de optimización, desarrollada en [2]. Esta utiliza funciones especiales llamadas funciones valor las cuales tienen leyes de recurrencia que permiten obtener la solución del problema de maximización (1.13).

2.1. Funciones valor estado

Las funciones valor son la base de la programación dinámica, sus leyes de recurrencia nos permiten construir algoritmos en el marco del aprendizaje por refuerzo.

Definición 4 (Función valor de estado) Es una función, $v^\pi : \mathcal{S} \rightarrow \mathbb{R}$:

$$v^\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} | S_0 = s \right] \quad (2.1)$$

Donde hemos tomado el horizonte temporal, $T = \infty$. □

Observación 3 En este documento tomaremos la definición 4 como la definición de función valor de estado, tomando $T = \infty$. Estos problemas son llamados procesos de decisión de Markov con horizonte infinito. Sin embargo, el horizonte temporal T puede ser finito aunque requiere una definición ligeramente distinta. Pueden verse más definiciones en [7].

Definición 5 (Función valor de estado óptima)

$$v_*(s) = \max_{\pi \in \Pi} v^\pi(s) \quad (2.2)$$

Esta función tiene una clara interpretación en el proceso de decisión de Markov. Esta es la solución del problema de optimización (1.13) para una condición inicial $S_0 = s$. □

Podemos notar que existe una relación de recurrencia para la función valor de estado $v^\pi(s)$:

$$v^\pi(s) = \mathbb{E}_\pi \left[\sum_{\tau=0}^{\infty} \gamma^\tau R_{\tau+1} | S_t = s \right] = \mathbb{E}_\pi \left[R_1 + \gamma \sum_{\tau=1}^{\infty} \gamma^{\tau-1} R_{\tau+1} | S_t = s \right] \quad (2.3)$$

Deshaciendo la sumatoria de la esperanza en el primer paso, obtenemos:

$$v^\pi(s) = \sum_{s',r} p(s',r|s,\pi(s)) \left[r + \underbrace{\gamma \mathbb{E}_\pi \left[\sum_{\tau=1}^{\infty} \gamma^{\tau-1} R_{\tau+1} | S_t = s' \right]}_{v^\pi(s')} \right] \quad (2.4)$$

Y con ello obtenemos la ecuación de Bellman para la función valor de estado.

Colorario 1 (Ecuación de Bellman) Entonces la función valor de estado $v^\pi(s)$ cumple la siguiente ecuación:

$$v^\pi(s) = \sum_{s',r} p(s',r|s,\pi(s)) \left[r + \gamma v^\pi(s') \right] \quad (2.5)$$

Utilizando las ecuaciones (1.4) y (1.2) podemos re-escribir la ecuación de Bellman en función de la recompensa esperada $\rho(s, a)$:

$$v^\pi(s) = \rho(s, \pi(s)) + \gamma \sum_{s'} p(s'|s, \pi(s)) v^\pi(s') \quad (2.6)$$

Por otro lado, también podemos obtener una relación de recurrencia para la función valor de estado óptima. Tomando la definición de la función de estado óptima (2.2) y la expresión anterior, obtenemos:

$$v_*(s) = \max_{\pi \in \Pi} \left[\rho(s, \pi(s)) + \gamma \sum_{s'} p(s'|s, \pi(s)) v^\pi(s') \right] \quad (2.7)$$

Colorario 2 (Ecuación de Bellman óptima) Entonces la función valor de estado óptima $v_*(s)$ cumple la siguiente ecuación:

$$v_*(s) = \max_{a'} \left[\rho(s, a') + \gamma \sum_{s'} p(s'|s, a') v_*(s') \right] \quad (2.8)$$

Esta expresión nos permite recuperar la política óptima tomando el argumento de la maximización anterior:

$$\pi_*(s) = \arg \max_{a'} \left[\rho(s, a') + \gamma \sum_{s'} p(s'|s, a') v_*(s') \right] \quad (2.9)$$

Entonces, podemos obtener la política $\pi_*(s)$ si conocemos la función valor $v_*(s) \quad \forall s \in \mathcal{S}$.

2.2. Funciones valor de estado-acción

En la programación dinámica se define otras dos funciones muy útiles y en ocasiones más conveniente que la función valor de estado. Estas son llamadas la función valor estado-acción y la función valor de estado-acción óptima.

Definición 6 (Función valor de estado-acción) Es una función, $q^\pi : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ tal que:

$$q^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{\tau=0}^{\infty} \gamma^\tau R_{t+\tau+1} | S_t = s, A_t = a \right] \quad (2.10)$$

Esta función es similar a la función valor de estado pero con un grado de libertad adicional a la hora de elegir la primera acción. Las sucesivas acciones vienen dada por la política π . $\square$

Definición 7 (Función valor de estado-acción óptima) Es una función, $q_* : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ tal que:

$$q_*(s, a) = \max_{\pi \in \Pi} q^\pi(s, a) \quad (2.11)$$

La función de valor estado-acción óptima tiene una interpretación en el proceso de decisión de Markov. Esta es la solución del problema de optimización (1.13) para una condición inicial $S_0 = s$ y tomando la primera acción $A_0 = a$. $\square$

Existe una relación entre a función valor estado $v^\pi(s)$ y la función valor estado-acción $q^\pi(s)$:

$$q^\pi(s, a) = \rho(s, a) + \gamma \sum_{s'} p_t(s' | s, a) v^\pi(s') \quad (2.12)$$

Esta relación se puede extender a las funciones de valor óptimas, utilizando (2.11) en (2.12) y la definición de la funciones valor óptima (2.2) obtenemos:

$$q_*(s, a) = \rho(s, a) + \gamma \sum_{s'} p_t(s' | s, a) v_*(s') \quad (2.13)$$

Además existe otra relación importante que se puede obtener desde la definición de $q^\pi(s, a)$. Esta es:

$$v_*(s) = \max_{a \in \mathcal{A}} q_*(s, a) \quad (2.14)$$

Por último, utilizando (2.14) y (2.13) obtenemos la ecuación de Bellman para la función valor de estado-acción:

Colorario 3 (Ecuación de Bellman estado-acción óptima)

$$q_*(s, a) = \rho(s, a) + \gamma \sum_{s'} p_t(s' | s, a) \max_{a' \in \mathcal{A}} q_*(s', a') \quad (2.15)$$

Estas leyes de recurrencia nos permitirán encontrar la función valor mediante la expresión (2.14) además de la política óptima π^* mediante la siguiente expresión:

$$\pi_*(s) = \arg \max_{a'} \left[\rho(s, a') + \gamma \sum_{s'} p(s'|s, a') v_*(s') \right] \quad (2.16)$$

$$\pi_*(s) = \arg \max_{a'} q_*(s, a') \quad (2.17)$$

2.3. Operadores de Bellman (OB)

Los operadores de Bellman son una herramienta necesaria para comprender los algoritmos básicos de aprendizaje por refuerzo. Estos nos proveen teoremas fundamentales que nos aseguran la convergencia de los algoritmos. Para ello seguiremos el discurso seguido en [7].

Denotaremos con la letra N para representa la cardinalidad del espacio de estados $\mathcal{S}$.

Definición 8 (OB de estado óptimo) Dado un un espacio vectorial $\mathbb{R}^N$ podemos definir el operador de Bellman $\mathcal{T}_v : \mathbb{R}^N \rightarrow \mathbb{R}^N$ como:

$$\mathcal{T}_v \mathcal{V}(s) = \max_{a \in \mathcal{A}} \left[\rho(s, a) + \gamma \sum_{s' \in \mathcal{S}} p_t(s'|s, a) \mathcal{V}(s') \right] \quad (2.18)$$

□

Es fácil ver que v^* es punto fijo del operador $\mathcal{T}_v$, es decir que al aplicar el operador $\mathcal{T}^v$ al vector v^* obtenemos otra vez gracias a que v^* cumple la ecuación de Bellman (2.6).

Además $\mathcal{T}_v$ cumple con la contracción en la norma L_∞ . Es decir, para todo $\mathcal{V}_1, \mathcal{V}_2 \in \mathbb{R}^N$, si $\mathcal{V}_1 \leq \mathcal{V}_2$

$$\|\mathcal{T}_v \mathcal{V}_1 - \mathcal{T}_v \mathcal{V}_2\|_\infty \leq \gamma \|\mathcal{V}_1 - \mathcal{V}_2\|_\infty \quad (2.19)$$

Por lo tanto debido al teorema de punto fijo de Banach [1], empezando en cualquier estimación inicial de la función valor de estado $\mathcal{V}_0$, la aplicación sucesivas del operador de Bellman nos conduce inevitablemente al punto fijo del operador.

De la misma manera se puede definir los operadores de Bellman para la función valor de estado-acción óptima $q_*(s, a)$. Para ello denotaremos M como la cardinalidad del espacio de acciones.

Definición 9 (OB de estado-acción óptimo) Dado un espacio vectorial $\mathbb{R}^N \times \mathbb{R}^M$ podemos definir el operador de Bellman $\mathcal{T}_q : \mathbb{R}^N \times \mathbb{R}^M \rightarrow \mathbb{R}^N \times \mathbb{R}^M$ como:

$$\mathcal{T}_q \mathcal{Q}(s, a) = \rho(s, a) + \gamma \sum_{s' \in \mathcal{S}} p_t(s'|s, a) \max_{a' \in \mathcal{A}} \mathcal{Q}(s', a') \quad (2.20)$$

Este operador tienen propiedades análogas al operador de Bellman $\mathcal{T}_v$. Esto nos permite decir que mediante las sucesivas aplicaciones del operador $\mathcal{T}_q$ para cualquier vector $\mathcal{Q}(s, a)$ convergeremos $q^*(s, a)$

3. Algoritmos de aprendizaje por refuerzo

Definidos ya los operadores de Bellman somos capaces de entender algunos algoritmos de aprendizaje por refuerzo basado en el teorema de punto fijo. Presentaremos tres algoritmos: *Value Iteration*, *Q-Iteration* y *Q-learning*.

3.1. Value Iteration

Gracias a que el operador de Bellman de estado $\mathcal{T}_v$ cumple el teorema de punto fijo de Banach, se puede realizar el algoritmo llamado *Value Iteration* [12, cap. 4.4]. Si consideramos un espacio de estados finito $\mathcal{S}$ con cardinalidad N . Entonces la función valor $v_*(s)$ puede ser representado por un vector $\mathcal{V}^* \in \mathbb{R}^N$.

Entonces si elegimos una semilla inicial $\mathcal{V}^*$, podemos aplicar el operador de Bellman $\mathcal{T}_v$ de manera sucesiva con el fin de converger a la función valor de estado óptima. Este procedimiento puede ser aplicado hasta que la norma de L^∞ de la diferencia de la función valor anterior y la siguiente sea menor que una cierta tolerancia *tol* (véase el Algoritmo 1).

Algoritmo 1 *Value Iteration*

```

1: procedure VALUE-ITERATION( $\mathcal{V}^*, tol$ )
2:    $k \leftarrow 0$ 
3:    $\mathcal{V}_k \leftarrow \mathcal{V}^*$ 
4:   while  $error \leq tol$  do
5:      $k \leftarrow k + 1$ 
6:     for  $\forall s \in \mathcal{S}$  do
7:        $\mathcal{V}_k(s) \leftarrow \mathcal{T}_v \mathcal{V}_{k-1}(s)$ 
8:     end for
9:      $error \leftarrow \|\mathcal{V}_k - \mathcal{V}_{k-1}\|_\infty$ 
10:  end while
11:   $\pi_k^*(s) = \arg \max_{a' \in \mathcal{A}} \left[ r(s, a') + \gamma \sum_{s'} p(s'|s, a') v_*(s') \right]$ 
12:  return:  $[\mathcal{V}_k(s), \pi_k^*(s)]$ 
13: end procedure

```

3.2. *Q-Iteration*

De la misma manera que la función valor de estado óptima $v_*(s)$, la función valor estado-acción óptima $q_*(s, a)$ cumple el teorema de punto fijo de Banach. Esto nos permite aplicar el mismo procedimiento utilizado en el algoritmo anterior. En este caso elegiremos una semilla inicial $\mathcal{Q}^* \in \mathbb{R}^{N+M}$ y aplicaremos el operador de Bellman $\mathcal{T}_q$ hasta que la norma L^∞ de la variación de la función valor de estado-acción sea menor que una cierta tolerancia tol (véase el algoritmo 2).

Algoritmo 2 *Q-Iteration*

```

1: procedure Q-ITERATION( $\mathcal{Q}^*, tol$ )
2:    $k \leftarrow 0$ 
3:    $\mathcal{Q}_k \leftarrow \mathcal{Q}^*$ 
4:   while  $error \leq tol$  do
5:      $k \leftarrow k + 1$ 
6:     for  $\forall (s, a) \in \mathcal{S} \times \mathcal{A}$  do
7:        $\mathcal{Q}_k(s, a) \leftarrow \mathcal{T}_q \mathcal{Q}_{k-1}(s, a)$ 
8:     end for
9:      $error \leftarrow \|\mathcal{Q}_k - \mathcal{Q}_{k-1}\|_\infty$ 
10:  end while
11:   $\mathcal{V}_k(s) = \max_{a \in \mathcal{A}} \mathcal{Q}_k(s, a)$ 
12:   $\pi_k^*(s) = \arg \max_{a \in \mathcal{A}} \mathcal{Q}_k(s, a)$ 
13:  return: ( $\mathcal{Q}_k(s, a)$  ,  $\mathcal{V}_k(s)$  ,  $\pi_k^*(s)$ )
14: end procedure

```

3.3. *Q-learning*

El algoritmo de *Q-learning* es una variación del algoritmo *Q-Iteration*. Debemos notar que para cada paso del algoritmo de *Q-Iteration* necesitamos aplicar el operador $\mathcal{T}_q$ para cada uno de los estados del espacio de estado. En el caso en el que la dinámica es desconocida es imposible evaluar el operador $\mathcal{T}_q$ ya que este depende de la distribución de probabilidad de transición $p_t(s'|s, a)$.

En el contexto del algoritmo de *Q-Learning* aunque no tenemos conocimiento de la dinámica, si que tenemos acceso a las medidas $(s_{t-1}, a_t, s_t, r_t) \mid \forall a_t$, donde podemos elegir a_t como queramos. Con ayuda de estas medidas podemos aplicar el operador de Bellman $\mathcal{T}^q$ de manera aproximada.

Entonces si la aplicación del operador de Bellman se escribe de la siguiente manera:

$$\mathcal{T}_q \mathcal{Q}(s, a) = \rho(s, a) + \gamma \sum_{s' \in \mathcal{S}} p_t(s'|s, a) \max_{a' \in \mathcal{A}} \mathcal{Q}(s', a') \quad (3.1)$$

Utilizando las medidas (s_{t-1}, a_t, s_t, r_t) aplicamos el operador de Bellman aproximado de la siguiente manera:

$$\mathcal{T}_q \mathcal{Q}(s_{t-1}, a_t) \approx r_t + \gamma \max_{a' \in \mathcal{A}} \mathcal{Q}(s_t, a') \quad (3.2)$$

Estamos obviando la esperanza por lo que es necesario ponderar el resultado de aplicar el operador $\mathcal{T}_q$ y el valor previo $\mathcal{Q}(s_{t-1}, a_t)$. Esto se traduce a:

$$\mathcal{T}_q \mathcal{Q}(s_{t-1}, a_t) \approx (1 - \alpha) \mathcal{Q}(s_{t-1}, a_t) + \alpha \left[r_t + \gamma \max_{a' \in \mathcal{A}} \mathcal{Q}(s_t, a') \right] \quad (3.3)$$

Donde $\alpha \in (0, 1)$ es un hiper-parámetro que nos ayuda a aproximar mejor la esperanza que estamos obviando.

Este procedimiento nos ayuda a obtener una aproximación para el operador de Bellman $\mathcal{T}_q$, sin embargo no nos dice nada sobre como tomar las acciones. Por lo general, la elección de las acciones se realiza mediante técnicas ambiciosas (*Greedy*). Junto a *Q-learning* lo más usual es utilizar la técnica de ϵ -Greedy para tomar las acciones en cada iteración. Esta toma la acción óptima que nos daría la función valor de estado acción o una completamente aleatoria. Esta elección depende de una variable aleatoria ϵ que determina si se toma una u otra.

De esta manera la acción se toma de la siguiente manera:

$$a_{t+1} \leftarrow \begin{cases} \arg \max_{a' \in \mathcal{A}} [\mathcal{Q}(s_t, a')] & \text{si } rand \geq \epsilon \\ aleatoria \in \mathcal{A} & \text{si } rand < \epsilon \end{cases} \quad (3.4)$$

Si solo tomáramos las acciones óptimas proporcionadas por la función $\mathcal{Q}^*$ (una aproximación de $q^*(s, a)$), probablemente no exploremos otras regiones de $\mathcal{S} \times \mathcal{A}$ convergiendo a un mínimo local del algoritmo. El paso de *Q-Learning* y la elección de la acción ϵ -Greedy forman el algoritmo (3).

4. Movimiento de una partícula en un potencial

Presentamos un problema de física clásica en el que existe una partícula en un potencial, como ejemplo para mostrar el funcionamiento de los algoritmos presentados.

4.1. Descripción del problema

El potencial elegido tiene la forma presentada en la Figura (3). Este sistema queda caracterizado mediante la posición x y velocidad v en un instante de tiempo dado. Definiremos el problema dentro de una región acotada:

$$x \in [-6, 6] \quad v \in [-6, 6] \quad (4.1)$$

Algoritmo 3 *Q learning*

```
1: procedure Q-LEARNING( $\mathcal{Q}^*, s_0, tol, \alpha, \epsilon$ )
2:    $k \leftarrow 0$ 
3:    $\mathcal{Q} \leftarrow \mathcal{Q}^*$ 
4:   while  $error \leq tol$  do
5:      $t \leftarrow t + 1$ 
6:     if  $\epsilon > rand$  then
7:        $a_t \leftarrow$  acción aleatoria
8:     else
9:        $a_t \leftarrow \arg \max_{a' \in \mathcal{A}} [\mathcal{Q}_t(s_t, a')]$ 
10:    end if
11:    Actuar con  $a_t$  y medir  $r_t$  y  $s_{t+1}$ 
12:     $\mathcal{Q}(s_t, a_t) \leftarrow (1 - \alpha)\mathcal{Q}(s_t, a_t) + [r(s_t, a_t) + \gamma \max_{a' \in \mathcal{A}} \mathcal{Q}(s_{t+1}, a')]$ 
13:  end while
14:  return  $\{a_t\}_{t>0}$ 
15: end procedure
```

Por otra parte, debido a que la formulación presentada esta en tiempo discreto, para resolver este problema necesitamos un esquema numérico que nos permita convertir la variable continua t a una variable discreta. Elegiremos el esquema de euler para obtener el sistema dinámico discreto.

$$\frac{d}{dt} \begin{bmatrix} x(t) \\ v(t) \end{bmatrix} = f(x(t), v(t), a(t)) \quad (4.2)$$

Entonces de un problema en tiempo continuo:

$$\underbrace{\frac{d}{dt} \begin{bmatrix} x(t) \\ v(t) \end{bmatrix} = \begin{bmatrix} v(t) \\ -\nabla P(x(t)) - \frac{1}{2}x(t) + a(t) \end{bmatrix}}_{\text{Tiempo continuo}} \quad (4.3)$$

Pasamos a un problema en tiempo discreto

$$\underbrace{\begin{bmatrix} x_{t+1} \\ v_{t+1} \end{bmatrix} = \begin{bmatrix} x_t + \Delta t v_t \\ v_t + \Delta t [-\nabla P(x_t) - \frac{1}{2}x_t + a_t] \end{bmatrix}}_{\text{Tiempo discreto}} \quad (4.4)$$

Donde se ha tomando como tamaño de paso $\Delta t = 0.0725$.

Además se ha discretizado el espacio de estados $\mathcal{S}$ y el de acciones $\mathcal{A}$ de la siguiente manera:

- $\mathcal{S} = \underbrace{\{-6.0000, -5.9195, \dots, 5.9195, 6.0000\}}_{150 \text{ puntos en } x} \times \underbrace{\{-6.0000, \dots, 5.9195, 6.0000\}}_{150 \text{ puntos en } v}$
- $\mathcal{A} = \{-20, -10, 0, 10, 20\}$

Entonces definimos la recompensa esperada como:

$$\rho(s, a) = \rho([x, v], a) = 100 \exp(-(x^2 + v^2)/0.5^2) \quad (4.5)$$

Dado en este caso el sistema es determinista, por lo que el operador de Bellman se puede escribir como:

$$\mathcal{TV}(s) = 100 \exp(-(x^2 + v^2)/0.5^2) + \max_{a \in \mathcal{A}} [\mathcal{V}(f(s, a))] \quad (4.6)$$

A continuación compararemos los algoritmos presentados en este problema concreto.

4.2. Comparación entre algoritmos

Utilizaremos el algoritmo *Q-Iteration* y el algoritmo *Q-learning* para resolver el problema planteado. Además, con el fin de comparar los algoritmos de aprendizaje por refuerzo con metodologías que tienen en cuenta el conocimiento de la dinámica del sistema, resolveremos el siguiente problema de control óptimo:

$$\begin{aligned} \max_{a(t) \in L^2} \int_0^\infty -(\|s(t)\|^2 + \|a(t)\|^2) dt \\ \text{sujeto a :} \\ \dot{s} = As(t) + Bu(t) \end{aligned} \quad (4.7)$$

Donde A y B son las matrices jacobianas de f con respecto al estado $s = [x, v]$ y la acción a , evaluados en $s = [0, 0]$ y $a = 0$. Este control tipo es llamado regulador cuadrático lineal (LQR) [11].

En la Figura (3) se compara el espacio de fases libre, el espacio de fases con la política obtenida por los algoritmos *Q-iteration* y *Q-learning* con respecto al espacio de fases obtenida mediante el regulador lineal cuadrático y el espacio de fases del sistema libre.

- **Espacio de fases libre:** Las trayectorias del sistema convergen a los puntos donde el potencial es mínimo. Podemos identificar estos pozos en la Figura (3) como los puntos rojos.
- **Espacio con control LQR:** Las trayectorias del sistema convergen al punto donde la recompensa es máxima. Por lo que se puede decir que resuelve el problema de manera óptima.
- **Espacio con Q-Iteration:** Las trayectorias del sistema convergen al punto donde la recompensa es máxima. Sin embargo si lo comparamos con el control LQR este está lejos de ser óptimo en regiones lejanas al origen.
- **Espacio con Q-learning:** Las trayectorias que están cerca al origen convergen al punto donde la recompensa es máxima, sin embargo cuando nos alejamos del origen el espacio de fases se asemeja al espacio de fases libre. Esto se debe a que el algoritmo de *Q-Learning* solo ha explorado cerca del origen y al encontrar un máximo de la recompensa la exploración hacia otras regiones del espacio se reduce.

Los algoritmos de aprendizaje por refuerzo pueden resolver el problema planteado pero con una eficacia menor que las metodologías que tienen en cuenta la dinámica.

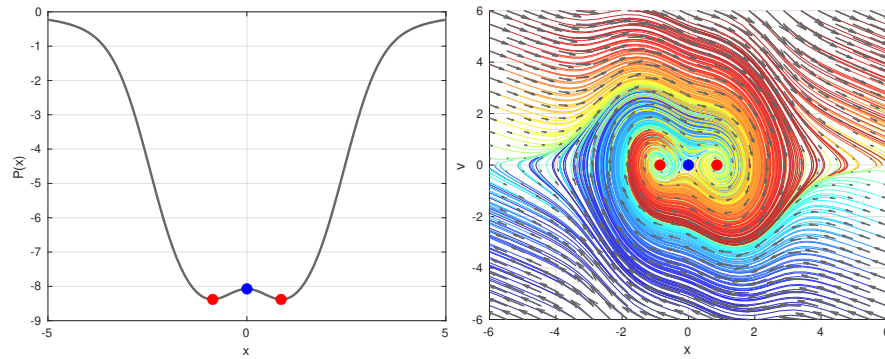


Figura 3. A la izquierda se muestra el potencial en el que se mueve la partícula del ejemplo. Además a la derecha se muestra el comportamiento del espacio de fases del sistema. Allí están dibujadas distintas trayectorias dentro del espacio de fases representadas con distintos colores. Podemos ver como todas las trayectorias acaban en los atractores representados por los puntos rojos, mientras que las trayectorias escapan del repulsor representado por el punto azul.

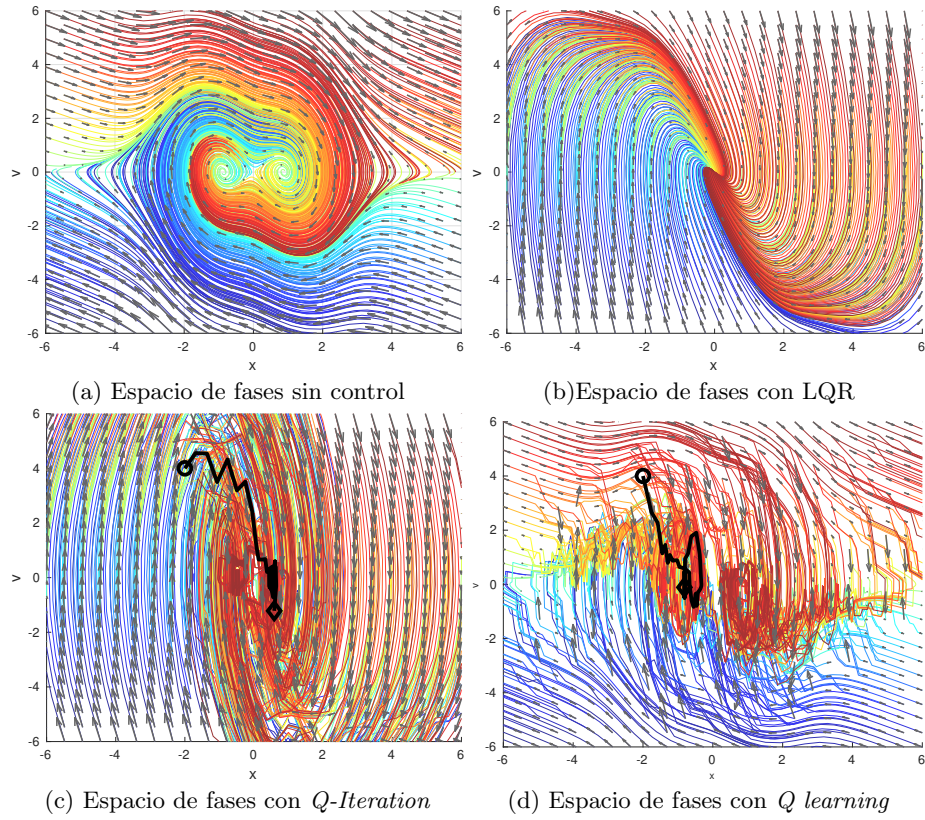


Figura 4. Comparamos el espacio de fase del sistemas con la política obtenida mediante LQR (Figura b), la política obtenida mediante el algoritmo Q -Iteration (Figura c) y la dinámica libre (Figura a). Por último, en la Figura (c) y (d) añadimos una trayectoria particular pintada de negro.

Parte II

Problema de pastoreo de ovejas

El problema del pastoreo de ovejas mediante perros es un problema típico de interacción de individuos. En este problema el movimiento de los perros es controlado con el fin de conducir a un rebaño de ovejas a una localización deseada. Para resolver este problema es necesario: (1) modelar las interacciones entre todos los individuos implicados y (2) definir un problema de optimización que permita obtener una secuencia de aceleraciones óptima. Extenderemos más estos puntos a continuación:

1. El modelado de estos sistemas ha sido estudiado en trabajos como [10] o [4]. En donde se describen interacciones radiales repulsivas entre el par oveja-perro e interacciones atractivas radiales entre el par oveja-oveja. Este tipo de interacciones hace que el modelado del sistema perros-ovejas sea intuitivo y muy parecido a los sistemas de partículas en mecánica celeste o en sistemas de partículas eléctricamente cargadas.
2. El problema de optimización deberá minimizar la distancia de la posición media de las ovejas y de un punto objetivo $\mathbf{u}_T$, tomando como las variables de optimización las trayectorias de los perros ². La evolución de las posiciones de las ovejas quedarán determinadas por el movimiento de los perros siguiendo un modelo dinámico. En [5] se plantea este como un problema de control óptimo considerando solo una oveja y solo un perro, obteniendo una manera de tomar las aceleraciones de los perros dependiendo de la posición de la oveja. En [6], se generaliza el resultado anterior a un sistema como varios perros y varias ovejas, donde las aceleraciones de los perros son calculados dadas las posiciones iniciales de las ovejas. Es decir, los perros necesitan observar las posiciones de las ovejas en un momento inicial y luego son capaces de tomar una secuencia de aceleraciones para llevarlas al punto objetivo.

Estos dos resultados son capaces de conducir a las ovejas si se conocen todas las posiciones de los individuos y el modelo dinámico del cual se rigen. Sin embargo, en la práctica el modelo dinámico elegido puede diferir en cierta medida del comportamiento real, causando un gran desfase en estas soluciones. Otro problema que puede surgir en estas soluciones son los errores de medida en la posición de las ovejas, variables de las cuales depende las acciones a seguir por

² Las trayectorias de los perros pueden ser determinada de distintas maneras. En este trabajo determinaremos las trayectorias mediante las aceleraciones de los perros.

parte de los perros. Es decir, para que los perros sepan como actuar deben ser capaces de medir con precisión la posición de cada una de las ovejas, problema que se acentúa cuando el número de ovejas es grande. En otras palabras, las soluciones presentadas son poco robustas ante variaciones del sistema, por lo que aún siendo soluciones óptimas son poco pragmáticas.

Dicho todo esto pareciera que las soluciones presentadas anteriormente son demasiado sofisticadas para el problema del pastoreo. Es conocido que existen estrategias desarrolladas por los pastores y sus perros que permiten resolver este problema sin necesidad de un modelado matemático; como por ejemplo una estrategia que llamaremos *empujar y girar*. Esta estrategia consiste en conducir a los perros detrás de las ovejas y girar de manera que la velocidad de las ovejas apunte al punto objetivo. Esta solución es muy robusta ya que solo necesita la posición media de las ovejas y no con gran exactitud. Además solo es necesario que la dinámica respete las interacciones de atracción y repulsión descritas en [4]. Aunque esta solución está lejos de ser óptima³, es robusta. Sin embargo esta estrategia tiene limitaciones cuando la dispersión de las ovejas es grande con respecto a la distancia entre la media de las posiciones de las ovejas y la posición del perro considerado.

El aprendizaje por refuerzo nos puede ayudar a encontrar una política que sea capaz de reproducir el comportamiento de la política heurística cuando la dispersión de las ovejas es pequeña y actuar de otra manera cuando la dispersión es grande. Por otro parte, gracias a que el aprendizaje por refuerzo es capaz de resolver el problema de optimización sin necesidad del conocimiento de la ecuación dinámica, podemos realizar una reducción de variables. De esta manera, podemos diseñar una política que solo dependa de unas pocas observaciones de la misma forma que lo hace la estrategia de *empujar y girar*. Estas son las aportaciones de este trabajo por lo que en este capítulo desarrollaremos los detalles.

A continuación seguiremos el siguiente esquema:

1. Introduciremos el modelo dinámico para la interacción entre las ovejas y los perros.
2. Construiremos el proceso de decisión de Markov asociado al problema.
3. Describiremos la metodología con la que resolveremos el problema.
4. Presentaremos los resultados numéricos.

5. Modelo de Interacción

Introduciremos notación básica para la descripción del sistema dinámico.

³ Óptima en el sentido que sea el mínimo esfuerzo que necesitan realizar los perros para llevar su tarea a cabo

5.1. Notación

Llamaremos N_o al número de ovejas implicadas en el problema, mientras que llamaremos N_p al número de perros. Además para describir el modelo del movimiento de los individuos seguiremos la notación presentada en la Tabla (1)

Notación	Descripción
$\mathbf{u}_o^i \in \mathbb{R}^2$	Posición de la oveja i -ésima $\forall i \in \{1, 2, \dots, N_o\}$
$\mathbf{v}_o^i \in \mathbb{R}^2$	Velocidad de la oveja i -ésima $\forall i \in \{1, 2, \dots, N_o\}$
$\mathbf{u}_p^j \in \mathbb{R}^2$	Posición del perro j -ésimo $\forall j \in \{1, 2, \dots, N_p\}$
$\mathbf{v}_p^j \in \mathbb{R}^2$	Velocidad del perro j -ésimo $\forall j \in \{1, 2, \dots, N_p\}$
$\mathbf{a}_i \in \mathbb{R}^2$	Aceleración del perro j -ésimo $\forall j \in \{1, 2, \dots, N_p\}$
$\mathbf{u}_T \in \mathbb{R}^2$	Posición objetivo

Cuadro 1. Notación del estado y acciones del sistema

Por otra parte, llamaremos a $\mathbf{U}_o$ al conjunto de las posiciones de las posiciones de las ovejas, $\mathbf{U}_p$ al conjunto de las posiciones de los perros, $\mathbf{V}_o$ al conjunto de las velocidades de las ovejas y $\mathbf{V}_p$ al conjunto de las velocidades de los perros. Es decir:

$$\mathbf{U}_o = \begin{pmatrix} \mathbf{u}_o^1 \\ \mathbf{u}_o^2 \\ \vdots \\ \mathbf{u}_o^{N_o} \end{pmatrix} \quad \mathbf{U}_p = \begin{pmatrix} \mathbf{u}_p^1 \\ \mathbf{u}_p^2 \\ \vdots \\ \mathbf{u}_p^{N_p} \end{pmatrix} \quad \mathbf{V}_o = \begin{pmatrix} \mathbf{v}_o^1 \\ \mathbf{v}_o^2 \\ \vdots \\ \mathbf{v}_o^{N_o} \end{pmatrix} \quad \mathbf{V}_p = \begin{pmatrix} \mathbf{v}_p^1 \\ \mathbf{v}_p^2 \\ \vdots \\ \mathbf{v}_p^{N_p} \end{pmatrix} \quad (5.1)$$

Además llamaremos $\mathbf{u}_{om} \in \mathbb{R}^2$ a la posición media de las ovejas, es decir:

$$\mathbf{u}_{om} = \frac{1}{N_o} \sum_{k=1}^{N_o} \mathbf{u}_o^k \quad (5.2)$$

Por último llamaremos $cov(\mathbf{U}_o) \in \mathcal{M}_{2 \times 2}(\mathbb{R})$ a la matriz de co-varianza de las posiciones de las ovejas.

$$cov(\mathbf{U}_o) = \frac{1}{N_p} \sum_{k=1}^{N_o} (\mathbf{u}_o^k - \mathbf{u}_{om})(\mathbf{u}_o^k - \mathbf{u}_{om})^T \quad (5.3)$$

5.2. Hipótesis del modelo

Este modelo asume las siguientes hipótesis:

1. Los perros pueden acercarse a las ovejas pero con un límite razonable que sea como mínimo la suma de los radios mínimos de las ovejas y los perros

2. Las ovejas tienden a agruparse mientras que no exista ninguna alteración de su trayectoria por parte de los perros.
3. Las ovejas tienden a alejarse cuando un perro se acerca.

Estas hipótesis son planteadas en [6], además se indica como podemos elegir estas fuerzas de interacción. Con el fin de tener interpretabilidad de los parámetros definiremos nuestras propias fuerzas de interacción.

5.3. Fuerzas de interacción

Estas interacciones serán radiales de manera que solo se puede repeler o atraer en la dirección que define la línea que une a los individuos. Es decir, dado un agente con posición $\mathbf{u}_1 \in \mathbb{R}^2$ y otro agente con posición $\mathbf{u}_2 \in \mathbb{R}^2$ la fuerza de interacción entre ellos estará en la dirección $(\mathbf{u}_1 - \mathbf{u}_2)$ y esta dependerá de la distancia entre los individuos $r = |\mathbf{u}_1 - \mathbf{u}_2|$, donde $|\cdot|$ es la norma euclídea.

A continuación presentaremos las fuerzas de interacción implicadas en este problema.

- **Fuerza de interacción oveja-oveja** (g_{oo}): Esta debe ser atractiva de modo que puedan formar un rebaño. Sin embargo si permitimos que las ovejas se atraigan siempre tendremos a todas en el mismo punto, lo cual desde un punto de vista pragmático es imposible. De modo que deberán tener un radio mínimo de atracción, r_{min}^o que representará de alguna manera el tamaño de la oveja. Proponemos la siguiente fuerza de interacción:

$$g_{oo}(r) = -\left(\frac{1}{r} - \frac{1}{r_{min}^o}\right) \quad (5.4)$$

- **Fuerza de interacción oveja-perro** (g_{op}): La fuerza de interacción del perro y la oveja debe ser siempre repulsiva de modo que una oveja pueda escapar del perro. Proponemos la siguiente fuerza de interacción:

$$g_{op}(r) = -F_m e^{-r^2/\sigma^2} \quad (5.5)$$

Donde σ representa la distancia de interacción entre las ovejas. Cuando $r = \sigma$ la fuerza es $g_{op}(\sigma) = -F_m/e$ mientras que en hipotético caso en el que la oveja y el perro esten en el mismo lugar $g_{op}(0) = -F_m$.

- **Fuerza de interacción perro-perro** (g_{pp}): Al igual que la fuerza de interacción *oveja-oveja* es necesario impedir las colisiones de los perros dado que nos pueden llevar a soluciones imposibles en el problema real. Proponemos la siguiente fuerza de interacción.

$$g_{pp}(r) = -\left(\frac{1}{r} - \frac{1}{r_{min}^p}\right) \quad (5.6)$$

En la Figura (5) mostramos el comportamiento de estas fuerzas con respecto a la distancia de los individuos implicados.

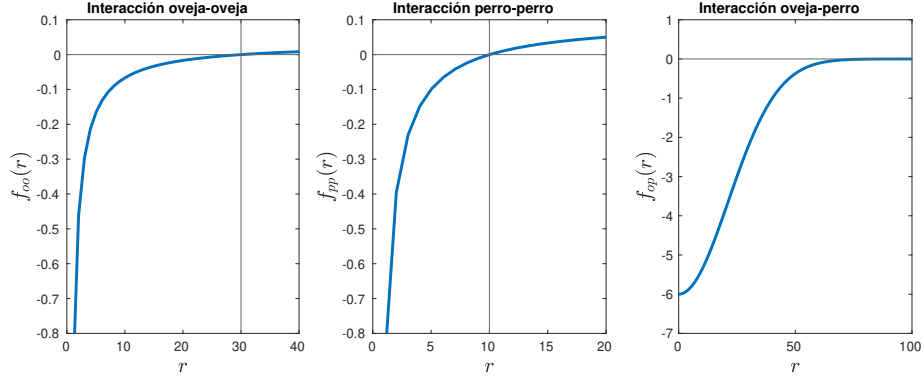


Figura 5. Fuerzas de interacción entre los individuos con $r_{min}^p = 10$, $r_{min}^o = 30$ y $\sigma = 50$. Los valores negativos indican repulsión mientras que los positivos atracción.

5.4. Modelo dinámico en tiempo continuo

El modelo dinámico que seguirá nuestro sistema de individuos se rige por el siguiente sistema de ecuaciones diferenciales:

$$\dot{\mathbf{u}}_o^i = +\mathbf{v}_o^i \quad (5.7)$$

$$\dot{\mathbf{u}}_p^j = +\mathbf{v}_p^j \quad (5.8)$$

$$\dot{\mathbf{v}}_o^i = -\nu_o \mathbf{v}_o^i + G_{oo}(\mathbf{u}_o^i, \{\mathbf{u}_o^k\}_{k=1}^{N_o}) + G_{op}(\mathbf{u}_o^i, \{\mathbf{u}_p^k\}_{k=1}^{N_p}) \quad (5.9)$$

$$\dot{\mathbf{v}}_p^j = -\nu_p \mathbf{v}_p^j + G_{pp}(\mathbf{u}_p^j, \{\mathbf{u}_p^k\}_{k=1}^{N_p}) + \mathbf{a}_j \quad (5.10)$$

$$\forall i \in \{1, \dots, N_o\} \quad (5.11)$$

$$\forall j \in \{1, \dots, N_p\} \quad (5.12)$$

Donde las fuerzas de interacción $G_{op}(\cdot)$, $G_{oo}(\cdot)$ y $G_{pp}(\cdot)$ son descritas en la tabla (2) y las constantes ν_o y ν_p representan los coeficientes de rozamiento que amortiguan el movimiento de los individuos. De esta manera para mantener el movimiento de los individuos es necesario una aceleración no nula.

Descripción	Expresión
Aceleración del perro j debido al resto de los perros	$G_{pp}(\mathbf{u}_p^j, \{\mathbf{u}_p^k\}_{k=1}^{N_p}) = \sum_{k \neq j}^{N_p} g_{pp}(\mathbf{u}_p^k - \mathbf{u}_p^j) \frac{(\mathbf{u}_p^k - \mathbf{u}_p^j)}{ \mathbf{u}_p^k - \mathbf{u}_p^j }$
Aceleración de la oveja i debido al conjunto de perros	$G_{op}(\mathbf{u}_o^i, \{\mathbf{u}_p^k\}_{k=1}^{N_p}) = \sum_{k=1}^{N_p} g_{op}(\mathbf{u}_p^k - \mathbf{u}_o^i) \frac{(\mathbf{u}_p^k - \mathbf{u}_o^i)}{ \mathbf{u}_p^k - \mathbf{u}_o^i }$
Aceleración de la oveja i debido al resto de las ovejas	$G_{oo}(\mathbf{u}_o^i, \{\mathbf{u}_o^k\}_{k=1}^{N_o}) = \sum_{k \neq i}^{N_o} g_{oo}(\mathbf{u}_o^k - \mathbf{u}_o^i) \frac{(\mathbf{u}_o^k - \mathbf{u}_o^i)}{ \mathbf{u}_o^k - \mathbf{u}_o^i }$

Cuadro 2. Fuerzas de interacción

Por último, llamaremos $\mathbf{x}$ al conjunto de aceleraciones y posiciones de todos los individuos, $\mathbf{a}$ al conjunto de aceleraciones de todos los perros y $F : \mathbb{R}^{(4N_p+4N_o)} \rightarrow \mathbb{R}^{(4N_p+4N_o)}$ a la ecuación diferencial de evolución presentada en la expresión (5.7). De esta manera, podemos referirnos a la ecuación diferencial (5.7) como:

$$\mathbf{x} = \begin{pmatrix} \mathbf{U}_o \\ \mathbf{U}_p \\ \mathbf{V}_o \\ \mathbf{V}_p \end{pmatrix} \quad \mathbf{a} = \begin{pmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \cdot \\ \cdot \\ \mathbf{a}_{N_p} \end{pmatrix} \Rightarrow \dot{\mathbf{x}}(t) = F(\mathbf{x}(t), \mathbf{a}(t)) \quad (5.13)$$

Con el objetivo de ejemplificar el problema en la Figura (6) mostramos tres ejemplos del comportamiento del sistema.

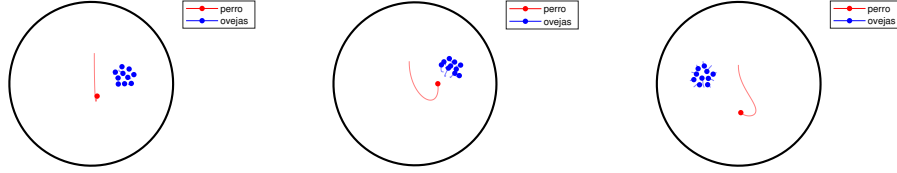


Figura 6. Ejemplo de la dinámica

5.5. Modelo dinámico en tiempo discreto

Para utilizar las herramientas propuestas en aprendizaje por refuerzo es necesario discretizar en el tiempo. Es por ello que es necesario elegir un mallado temporal donde se proyectará el problema. Además es necesario escoger un esquema de discretización para la ecuación diferencial.

En este caso para resolver la ecuación diferencial escogeremos el método numérico de resolución de ecuaciones diferenciales *runge-kutta* de orden cuarto. De esta manera, la ecuación diferencial se convierte en un problema en pasos discretos:

$$\dot{\mathbf{x}}(t) = F(\mathbf{x}(t), \mathbf{a}(t)) \rightarrow \begin{cases} \mathbf{x}_{t+1} = \mathbf{x}_t + \frac{1}{6}\Delta t(k_1 + k_2 + k_3 + k_4) \\ \text{donde : } \begin{cases} k_1 = F(\mathbf{x}_t, \mathbf{a}_t) \\ k_2 = F(\mathbf{x}_t + \frac{1}{2}k_1\Delta t, \mathbf{a}_t) \\ k_3 = F(\mathbf{x}_t + \frac{1}{2}k_2\Delta t, \mathbf{a}_t) \\ k_4 = F(k_3\Delta t, \mathbf{a}_t) \end{cases} \end{cases} \quad (5.14)$$

Si llamamos $F_d : \mathbb{R}^{4N_e+4N_d} \times \mathbb{R}^{2N_d} \rightarrow \mathbb{R}^{4N_e+4N_d}$ a:

$$F_d(\mathbf{x}, \mathbf{a}) = \mathbf{x} + \frac{1}{6} \Delta t (k_1 + k_2 + k_3 + k_4) \quad (5.15)$$

$$k_1 = F(\mathbf{x}, \mathbf{a}) \quad (5.16)$$

$$k_2 = F(\mathbf{x} + \frac{1}{2} k_1 \Delta t, \mathbf{a}) \quad (5.17)$$

$$k_3 = F(\mathbf{x} + \frac{1}{2} k_2 \Delta t, \mathbf{a}) \quad (5.18)$$

$$k_4 = F(k_3 \Delta t, \mathbf{a}) \quad (5.19)$$

Podemos escribir la dinámica en un paso temporal discreto de la siguiente manera:

$$\mathbf{x}_{t+1} = F_d(\mathbf{x}_t, \mathbf{a}_t) \quad (5.20)$$

De esta manera hemos llegado a un sistema dinámico determinista que nos generará las observaciones en el proceso de decisión de Markov.

6. Aprendizaje por refuerzo para el problema del pastoreo

Dado que el aprendizaje por refuerzo puede resolver el problema planteado sin conocimiento de la distribución de probabilidad de transición $p(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t)$, utilizaremos esta propiedad para reducir las variables de estado. Esta reducción se realiza con el finalidad de mitigar la dependencia de la política con respecto a todas las posiciones de las ovejas y con eso obtener una política robusta. Supondremos que el sistema dinámico puede ser descrito por las posiciones de los perros, $\mathbf{U}_p$, la posición media de los ovejas, $\mathbf{u}_{em}$ y su matriz de co-varianza, $cov(\mathbf{U}_o)$. Es decir, el agente deberá tomar decisiones sobre las aceleraciones de los perros teniendo en cuenta solamente estas variables.

6.1. Proceso de decisión de Markov

Entonces podemos plantear un proceso de decisión de Markov con los siguientes componentes:

- **Estado:** Es la composición de la posición media de las ovejas, su matriz de co-varianza y la posición de cada uno de los perros.

$$\mathbf{s} = \begin{bmatrix} \mathbf{u}_{om} \\ - \\ cov(\mathbf{U}_o) \\ - \\ \mathbf{U}_d \end{bmatrix} \quad (6.1)$$

- **Acción:** Al igual que en el caso inicial, las acciones posibles son las aceleraciones de cada uno de los perros.

$$\mathbf{a} = \begin{bmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \vdots \\ \mathbf{a}_{N_p} \end{bmatrix} \quad (6.2)$$

- **Recompensa:** La función de recompensa será:

$$r(\mathbf{s}, \mathbf{a}) = [\|\mathbf{u}_{om}(t) - \mathbf{u}_T\|^2 + \|\text{cov}(\mathbf{U}_o)\|_{fro}^2 + \beta\|\mathbf{a}(t)\|^2] \quad (6.3)$$

Donde:

- $\|\mathbf{u}_{om} - \mathbf{u}_T\|^2$ es el término que mide la distancia de la media de las ovejas al punto objetivo
- $\|\text{cov}(\mathbf{U}_o)\|_{fro}^2$ es la norma de Frobenious de la matriz de co-varianza. En otras palabras se requiere que las ovejas estén lo más juntas posibles.
- $\beta\|\mathbf{a}(t)\|^2$ es el coste de las acciones. En otras palabras este término es grande si el módulo de la aceleraciones de los perros son grandes, por lo que si el perro es capaz de lograr su objetivo con coste por la acción pequeña se considerará un agente eficiente.
- **Dinámica.** Supondremos que existe una distribución de probabilidad de transición entre un estado $\mathbf{s}_t$ y $\mathbf{s}_{t+1}$

$$p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) \quad (6.4)$$

Para nosotros es desconocida pero de la que podemos tomar medidas.

Observación 4 Deberemos notar que aunque la dinámica $\mathbf{x}_{t+1} = F_d(\mathbf{x}_t, \mathbf{a}_t)$ es determinista, la dinámica $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ es estocástica. Esto sucede debido el estado $\mathbf{s}$ solo tiene en cuenta propiedades globales de las ovejas y no cada una de las posiciones y velocidades de las ovejas. Esto hace que para un estado $\mathbf{s}$ existan varios posibles estados compatibles $\mathbf{x}$.

6.2. Metodología *Deep Deterministic Policy Gradient*

Utilizaremos el algoritmo presentado en [8], mediante la implementación [9]. En este algoritmo se utiliza una función definida por parámetros, $\pi_\theta(\mathbf{s})$, que aproxima la política y otra función definida por parámetros $q_\omega(\mathbf{s}, \mathbf{a})$ que aproxima la función valor estado-acción q^{π_θ} . En este trabajo utilizaremos redes neuronales para aproximar $q_\omega(\mathbf{s}, \mathbf{a})$ y $\pi_\theta(\mathbf{s})$.

- $q_\omega(\mathbf{s}, \mathbf{a})$ es una perceptrón multi-capas con 50 neuronas y 3 capas. Los parámetros ω se refiere al conjunto de pesos y bias de toda la red neuronal.
- $\pi_\theta(\mathbf{s})$ es una perceptrón multi-capas con 100 neuronas y 3 capas. Los parámetros θ se refiere al conjunto de pesos y bias de toda la red neuronal.

El algoritmo *Deep Deterministic Policy Gradient* consta de dos pasos principales:

- **Paso 1. Actualización de la política**($\pi_\theta(\mathbf{s})$): Si recordamos del capítulo anterior, el problema de optimización a resolver se puede escribir como:

$$v_*(\mathbf{s}) = \max_{\pi \in \Pi} v^\pi(\mathbf{s}) \quad (6.5)$$

Si tenemos en cuenta que la política está parametrizada por θ , el problema de optimización se reduce a:

$$v_*(\mathbf{s}) = \max_{\theta \in \mathbb{R}^p} v^{\pi_\theta}(\mathbf{s}) \quad (6.6)$$

Entonces para resolver el problema podemos obtener el gradiente de la función valor $\nabla_\theta v^{\pi_\theta}(\mathbf{s})$ y realizar un ascenso por gradiente. Por otra parte, sabiendo que $v^\pi(\mathbf{s}) = q^\pi(\mathbf{s}, \pi(\mathbf{s}))$ y utilizando la regla de la cadena podemos calcular el gradiente como:

$$\nabla_\theta v^{\pi_\theta}(\mathbf{s}) = \nabla_\theta [q^{\pi_\theta}(\mathbf{s}, \pi_\theta(\mathbf{s}))] = \left| \nabla_a q^{\pi_\theta}(\mathbf{s}, \mathbf{a}) \right|_{a=\pi_\theta(\mathbf{s})} \nabla_\theta \pi(\mathbf{s}) \quad (6.7)$$

Entonces simplemente podemos realizar la optimización de la siguiente manera:

$$\theta_{k+1} \leftarrow \theta_k + \alpha \nabla_\theta v^{\pi_\theta}(\mathbf{s}) \quad (6.8)$$

Sin embargo, para ello es necesario conocer $q^{\pi_\theta}(\mathbf{s}, \mathbf{a})$, este problema se abordará en el segundo paso.

- **Paso 2. Actualización de la función valor estado-acción**($q_\omega(\mathbf{s}, \mathbf{a})$): Esta función se puede aproximar por otra función que dependa de parámetros $q_\omega(\mathbf{s}, \mathbf{a})$. Dado que sabemos que la función $q^{\pi_\theta}(\mathbf{s}, \mathbf{a})$ puede ser actualizada mediante el algoritmo de *q-learning* (ecuación 3.2). Para ello es necesario tener tuplas $(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}, r_t)$ de medidas.

$$q^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t) \leftarrow y(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}, r_{t+1}) \quad (6.9)$$

$$y(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}, r_{t+1}) = r_{t+1} + \gamma q^{\pi_\theta}(\mathbf{s}_{t+1}, \pi_\theta(\mathbf{s}_{t+1})) \quad (6.10)$$

Teniendo un historial de M transiciones $B = \{\mathbf{s}_t^m, \mathbf{a}_t^m, \mathbf{s}_{t+1}^m, r_{t+1}^m\}_{m=1}^M$, podemos encontrar los parámetros ω de $q_\omega(\mathbf{s}, \mathbf{a})$ que mejor aproximen $q^{\pi_\theta}(\mathbf{s}, \mathbf{a})$ de la siguiente manera:

$$\min_{\omega} \left[\sum_{\{\mathbf{s}, \mathbf{a}, \mathbf{s}', r\} \in B} [q_\omega(\mathbf{s}, \mathbf{a}) - y(\mathbf{s}, \mathbf{a}, \mathbf{s}', r)]^2 \right] \quad (6.11)$$

Observación 5 La expresión (6.11) es el paso de *q-learning* en el caso en el que el espacio de estados $\mathcal{S}$ y de acciones $\mathcal{A}$, son continuos. Dado que el algoritmo de *q-learning* requiere la actualización puntual para un estado y acción concreta, éste es impracticable en espacios continuos. La alternativa es tomar varias medidas $(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}, r_t)$ y actualizar los parámetros de la función $q_{\omega}^{\pi_{\theta}}(s, a)$. Entonces el paso de *q-learning* se convierte en un problema de aprendizaje supervisado.

En [8], no solo realizan estos dos pasos, además de la actualización de $\pi_{\theta}(\mathbf{s})$ y $q_{\omega}(\mathbf{s}, \mathbf{a})$, se tienen en cuenta otras dos funciones $\pi'_{\theta}(\mathbf{s})$ y $q'_{\omega}(\mathbf{s}, \mathbf{a})$. Cuando se actualizan los parámetros θ y ω se realiza por medio de una media ponderada entre los nuevos parámetros dada por el algoritmo presentado y los parámetros θ' y ω' . De esta manera la evolución de los parámetros ω y θ es más lenta. En caso de no realizarlo así, en [8] se informa inestabilidad del algoritmo, sin llegar a converger nunca. Presentamos este Algoritmo en (4)

Algoritmo 4 DDPG

```

1: procedure DDPG( $q_{\omega}(\mathbf{s}, \mathbf{a}), \pi_{\theta}(\mathbf{s})$ )
2:   Definimos la política  $\pi'_{\theta} \leftarrow \pi_{\theta}$ 
3:   Definimos la función valor  $q'_{\omega}(\mathbf{s}, \mathbf{a}) \leftarrow q_{\omega}(\mathbf{s}, \mathbf{a})$ 
4:   for true do
5:     Observamos el estado  $\mathbf{s}_t$  y actuamos según  $\mathbf{a}_t \leftarrow \pi_{\theta}(\mathbf{s}) + \mathcal{N}(0, \sigma)$ 
6:     Ejecutamos la acción  $\mathbf{a}_t$  y observamos  $\mathbf{s}_{t+1}$  y  $r_{t+1}$ 
7:     Guardamos la experiencia  $\{\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}, r_{t+1}\}$  una base de datos  $\mathcal{D}$ 
8:     Tomamos de manera aleatoria  $M$  experiencias de  $\mathcal{D}$ 
9:     Actualizamos la  $q'_{\omega}(\mathbf{s}, \mathbf{a})$  según la expresión (6.11)
10:    Actualizamos la política  $\pi_{\theta}(\mathbf{s})$  según la ecuación (6.8)
11:     $\theta' \leftarrow \tau\theta + (1 - \tau)\theta'$ 
12:     $\omega' \leftarrow \tau\omega + (1 - \tau)\omega'$ 
13:  end for
14: end procedure

```

7. Política Heurística y pre-entrenamiento del actor

Con el fin de empezar con una política que sea capaz de realizar acciones que no sea totalmente aleatorias, pre-entrenaremos $\pi_{\theta}(\mathbf{s}, \mathbf{a})$ con datos generados con la política heurística que presentaremos en esta sección.

7.1. Política Heurística

La idea de política es conducir a los perros a la posición trasera de las ovejas, en la línea que une la posición de la media de los perros $\mathbf{u}_{em}$ y la posición del punto objetivo $\mathbf{u}_T$. Esta idea ha sido propuesta en [5] para el problema con un

solo perro y una sola oveja, sin embargo siempre que la co-varianza de las ovejas sea pequeña esta política es capaz de estabilizar la media de las ovejas al punto objetivo.

Esta política se logra considerando que los perros pueden acelerar en dos direcciones. La primera dirección es la formada por la posición del perro y el punto medio de las ovejas, mientras que la segunda es la dirección perpendicular. Entonces, dado un perro j con posición $\mathbf{u}_j$ su política utilizará la siguiente expresión para tomar una acción:

$$\pi_j(\mathbf{s}) = h_r(|\mathbf{u}_d^j - \mathbf{u}_{em}|)(\mathbf{u}_d^j - \mathbf{u}_{em}) + h_p(\phi)(\mathbf{u}_d^j - \mathbf{u}_{em})^\perp \quad (7.1)$$

Donde h_r y h_p son:

$$h_r(r) = -\frac{1}{r} + \frac{1}{r_{min}^p} \quad (7.2)$$

$$h_p(\phi) = 1 - \cos(\phi) \quad (7.3)$$

Y ϕ es el ángulo formado por el vector $(\mathbf{u}_{om} - \mathbf{u}_T)$ y $(\mathbf{u}_{om} - \mathbf{u}_d)$ (véase Figura 7).

De esta manera un perro puede moverse en dos direcciones una acercándose a las ovejas y otra en dirección perpendicular. Entonces el perro se acercará a las ovejas hasta un r_{min} y se colocará en la parte trasera para empujarlas al objetivo.

Aunque esta política puede controlar el sistema, puede no funcionar correctamente si las ovejas no están previamente agrupadas. Dado que esta diseñada para acercarse al punto medio a una distancia de r_{min} es ciertas ocasiones es posible que esta acción disperse a las ovejas.

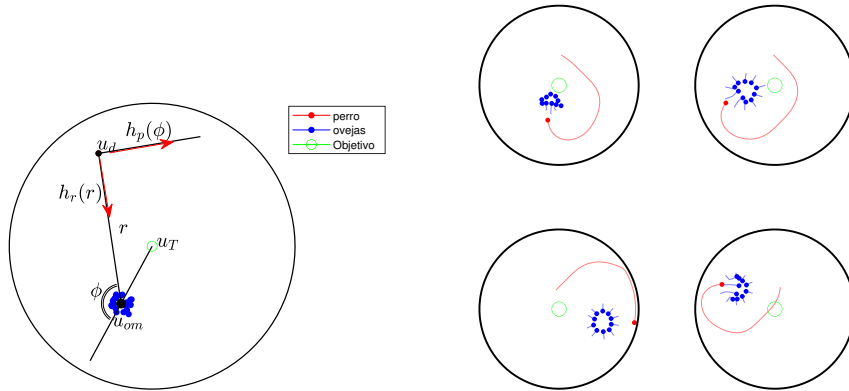


Figura 7. Comportamiento de la política heurística

7.2. Pre-entrenamiento

Entrenaremos la política $\pi_\theta(\mathbf{s}, \mathbf{a})$ mediante el algoritmo de gradiente estocástico y los datos generados por la política heurística. De esta manera, la política inicial del algoritmo *Deep Deterministic Policy Gradient* mejorará la semilla inicial mediante la interacción con el sistema (véase el algoritmo 5).

Algoritmo 5 *DDPG con Pre-entrenamiento*

- 1: **procedure** DDPG-PRE-ENTRENADO
 - 2: Definimos la política π_θ como un perceptrón multicapa.
 - 3: Generamos un trayectorias con $\mathbf{s}_0$ aleatorias siguiendo la política π_{ref} .
 - 4: Entrenamos la red neuronal π_θ con las trayectorias generadas
 - 5: Definimos la función valor estado-acción como una red neuronal $q_\omega(\mathbf{s}, \mathbf{a})$
 - 6: Ejecutamos el algoritmo DDPG con la política π_θ .
 - 7: **end procedure**
-

8. Resultados numéricos

Se ha realizado experimentos numéricos para el procedimiento propuesto. Esto hace que el agente inicial sea capaz de resolver en cierta medida el problema planteado. En la Figura (8) se muestra la comparación entre la política heurística, la política pre-entrenada y la política entrenada mediante aprendizaje por refuerzo. Se puede ver como la política heurística y la política pre-entrenada tiene un comportamiento periódico debido a que aunque estas dos políticas sean capaces de llevar la media de las ovejas al punto objetivo, no logra ser estable. Las ovejas son alejadas del punto objetivo, sin embargo la política heurística siempre vuelve a acercarlas al punto de interés. La solución obtenida por aprendizaje por refuerzo realiza el mismo movimiento que la política heurística, sin embargo esta ya no es oscilatoria, si no que utiliza más tiempo para llevar a las ovejas al objetivo (véase la Figura (9)). En la Figura (8), se puede visualizar como la recompensa obtenida por la política por aprendizaje por refuerzo es peor en el primer intervalo de tiempo. Sin embargo según se va considerando más tiempo, esta política es mejor. Esto se debe a que aunque la política heurística lleva más rápido a las ovejas al objetivo, luego sigue gastando energía (la norma del la acción) en mantenerlas en el centro.

9. Conclusiones

En este trabajo se ha estudiado el aprendizaje por refuerzo desde primeros principios para el entendimiento de este. En pocas palabras, el aprendizaje por refuerzo es una extensión de la programación dinámica pero basado en experiencias $(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}, r_t)$. La ecuación de Bellman puede verse como un operador de

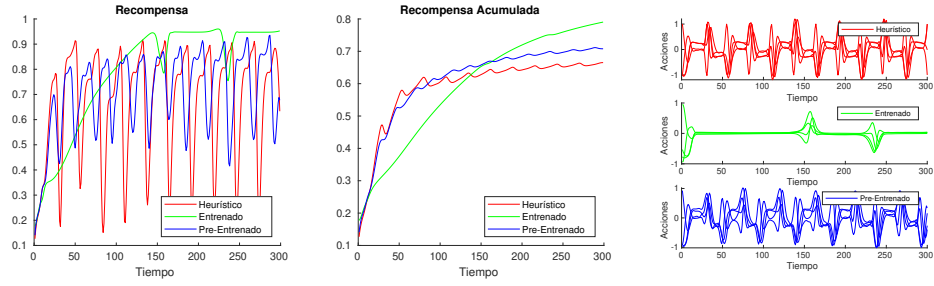


Figura 8. Comparación en tres la recompensa, la recompensa acumulada y las aceleraciones de los agentes

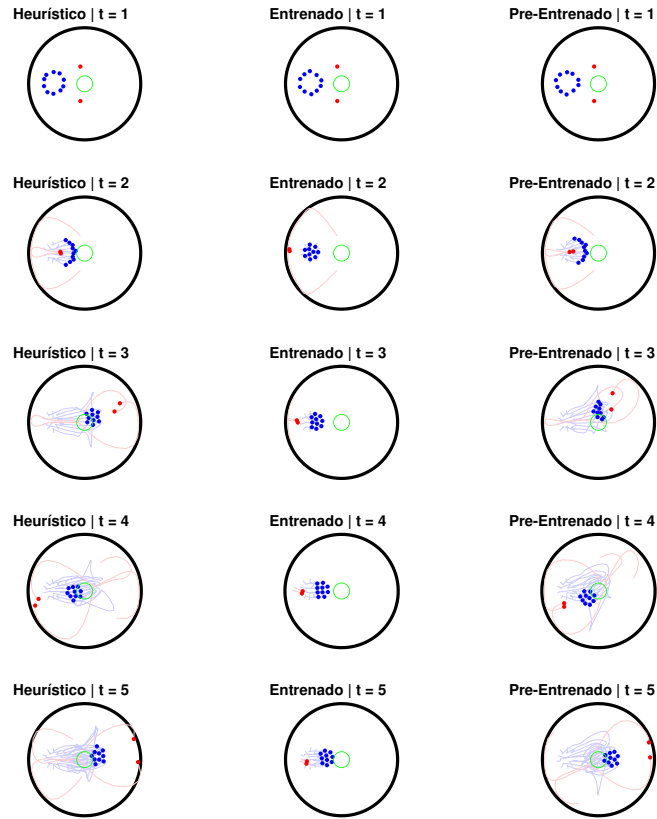


Figura 9. Capturas del movimiento del sistema para cuatro instantes de tiempo. La política heurística y la política pre-entrenada tiene el mismo comportamiento. Estas son capaces de llegar a las ovejas al punto central, sin embargo no logran mantenerlas sin ejercer más acciones. La política obtenida después del entrenamiento mediante aprendizaje por refuerzo es capaz de recogerlas, aunque de manera más lenta.

funciones que al ser aplicado iterativamente inevitablemente se cae a un punto fijo. Este punto fijo coincide con la función valor óptima, objeto capaz de devolvernos la política óptima. Los algoritmos aprendizaje por refuerzo son aproximaciones este operador de Bellman, ya que en la mayoría de los casos la aplicación de este operador es imposible, por falta de conocimiento de la dinámica o por que el estado de acciones y/o estados es muy grande.

La metodología de aprendizaje por refuerzo en problema con espacio de acciones y estados continuos son un gran reto para la comunidad científica. Aunque en esta tesis se ha utilizado el algoritmo *Deep Deterministic Policy Gradient* existen varias aproximaciones para resolver el problema con estado y acciones continuos. En el caso concreto del problema de pastoreo se obtiene un resultado insatisfactorio en comparación con políticas heurísticas. Aunque logrará mejorar el coste final no es capaz de aprender estrategias obvias como recoger las ovejas o conducir las rápidamente al punto objetivo para luego no ejercer ninguna acción. Estos problemas pueden deberse a distintas razones como puede ser la elección de la arquitectura de la política o de la función valor estado-acción. Sin embargo, el [8] no abordan estas cuestiones, por el contrario sus funciones de aproximación que se utilizan contienen aproximadamente 10^5 parámetros lo cual hace que el número de iteraciones necesarias para obtener un resultado aceptable sea 10^6 . En esta tesis se ha logrado realizar 10^4 pasos del algoritmo *DDPG*, sin embargo no se ha obtenido un cambio significativo en la política, aún siendo una buena política inicial.

Por otro lado, otra razón por la que no se obtenga un resultado significativo, puede deberse a la falta de simetría en las funciones de aproximación. En este problema existen simetrías que deberían respetar las funciones de aproximación de la política y de la función valor de estado-acción. Las arquitecturas de las redes neuronales para la función valor $q_\omega(\mathbf{s}, \mathbf{a})$ debe ser tal que ante un cambio de entre las coordenadas de un perro i y un perro j , la función se mantenga inmutable. Es decir, dado que en este problema los perros son agentes indistinguibles, no es importante cual de los dos perros haga una acción específica, si no que el conjunto de perros actúen de una manera concreta. La exploración mediante aprendizaje por refuerzo puede romper esta simetría intrínseca en el problema obteniendo funciones valor y políticas no simétricas y que claramente no pueden ser una solución óptima del problema.

Este problema es abordado por la comunidad de aprendizaje por refuerzo mediante la metodología de problemas multi-agente. En este contexto la función valor para un todos los agentes, es la misma. De manera que la política es la misma para todos los agentes y cada uno de ellos aporta experiencia a una función valor común. Es podría ser una extensión de este trabajo.

Referencias

1. BANACH, S. Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales. *Fund. math* 3, 1 (1922), 133–181.
2. BELLMAN, R. The theory of dynamic programming. Tech. rep., Rand corp santa monica ca, 1954.
3. BELLMAN, R. A markovian decision process. *Indiana Univ. Math. J.* 6 (1957), 679–684.
4. COPPINGER, R., AND FEINSTEIN, M. *How dogs work*. University of Chicago Press, 2015.
5. ESCOBEDO, R., IBAÑEZIBA IBAÑEZ, A., AND ZUAZUA, E. Optimal strategies for driving a mobile agent in a Guidance by repulsion Model. Tech. rep., 2015.
6. KO, D., AND ZUAZUA, E. Asymtotic behavior and control of a Guidance by repulsion model. Tech. rep., 2019.
7. LAZARIC, A. Decision Processes and Dynamic Programming. *Course on Reinforcement Learning* (2013), 160–183.
8. LILICRAP, T. P., HUNT, J. J., PRITZEL, A., HEES, N., EREZ, T., TASSA, Y., SILVER, D., AND WIERSTRA, D. Continuous Control with deep reinforcement learning. Tech. rep., 2015.
9. MATHWORKS. Matlab - deep deterministic implementation. <https://www.mathworks.com/help/reinforcement-learning/ug/ddpg-agents.html>, 2016.
10. REYNOLDS, C. W. Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques* (1987), pp. 25–34.
11. SONTAG, E. D. *Mathematical control theory: deterministic finite dimensional systems*, vol. 6. Springer Science & Business Media, 2013.
12. SUTTON, R. S., AND BARTO, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.
13. WATKINS, C. J., AND DAYAN, P. Q-learning. *Machine learning* 8, 3-4 (1992), 279–292.