Gradientes de política

PID_00275579

Laura Ruiz Dern

Tiempo mínimo de dedicación recomendado: 2 horas

Universitat Oberta de Catalunya

Laura Ruiz Dern

Doctora en Astronomía y Astrofísica por el Observatoire de Paris. Licenciada en Física y MSc en Astrofísica, Cosmología y Física de Partículas por la Universidad de Barcelona, y doble MSc en Gestión de Proyectos y Cooperación Internacional por la ESNECA Business School. Investigadora y científica de datos con experiencia en *data mining* y *machine learning*, y actualmente especializada en *deep learning* y *deep reinforcement learning*. Ha trabajado en varios proyectos de investigación en astrofísica y ciencia de datos del Centre National d'Études Spatiales (CNES); del Centre National de la Recherche Scientifique (CNRS), en colaboración con la European Space Agency (ESA), y del Centre Tecnològic de Catalunya (Eurecat). Desde 2014 compagina sus investigaciones con la divulgación científica y con la docencia universitaria (Observatoire de Paris, Universitat Oberta de Catalunya).

El encargo y la creación de este recurso de aprendizaje UOC han sido coordinados por el profesor: Jordi Casas Roma

Primera edición: septiembre 2021

Av. Tibidabo, 39-43, 08035 Barcelona
Autoría: Laura Ruiz Dern
Producción: FUOC

Índice

Introducción 5

Objetivos 6

1 Introducción a los gradientes de política 7
1.1 Parametrización de la política 7
1.2 Tipos de políticas 10
1.3 Función objetivo 11
1.4 Actualización de los parámetros 11
1.5 Potencial de los métodos PG 12

2 El teorema de los gradientes de política 13
2.1 Demostración del teorema 14

3 Principales algoritmos de gradientes de política 16
3.1 REINFORCE 16
3.2 REINFORCE con línea de base 19

4 Inconvenientes de los gradientes de política 24

Resumen 26

Glosario 27

Bibliografía 28

Introducción

En *deep learning* el sistema aprende a partir de un conjunto de datos para luego aplicar el modelo con datos nuevos. Mientras que en *reinforcement learning* el sistema aprende de forma dinámica, ajustando las acciones conforme va recibiendo *feedback* de las acciones ya tomadas y eligiendo aquellas que dan mayor recompensa. En el momento en que combinamos ambos tipos de aprendizaje o, con otras palabras, cuando se introducen redes neuronales en el aprendizaje por refuerzo, hablamos de *deep reinforcement learning*.

Como ya hemos visto, esta necesidad de incluir las redes neuronales surge de la dificultad de aplicar los métodos tabulares en espacios de estados de altas dimensiones. En esos casos, usar una tabla para almacenar las recompensas para cada acción es inviable. Pero usar una red neuronal como función de aproximación, ya sea para aproximar una función de valor o una política, permite aplicar aprendizaje por refuerzo en estas situaciones.

Mientras que en el módulo anterior detallábamos las DQN y todas sus variantes como métodos que parten de la función de valor, en este módulo nos centraremos en los métodos que se refieren a la política, los gradientes de política (en inglés *policy gradients*), los cuales aproximan la política óptima directamente sin necesidad de una función de valor aproximada. Es decir, que en lugar de interesarnos por el valor de cada acción y que el agente se guíe por estos valores obtenidos, el comportamiento del agente vendrá regido por una política que deberá ir mejorando a lo largo del proceso de aprendizaje.

En primer lugar, estableceremos los conceptos y las bases de los gradientes de política, y en segundo lugar nos adentraremos en este nuevo método, el teorema de base en el que se fundamenta, los dos tipos de algoritmos principales, y sus ventajas e inconvenientes, así como en qué se diferencian de las DQN.

Objetivos

Los principales objetivos que estudiaremos en este módulo son:

1. Entender el concepto de gradiente de política y las bases de este tipo de solución aproximada.
2. Comprender el teorema en el que se fundamentan los gradientes de política.
3. Conocer los distintos tipos de métodos de gradientes de política, sus ventajas e inconvenientes.

1. Introducción a los gradientes de política

Las soluciones aproximadas son clave en el aprendizaje profundo porque nos permiten trabajar con espacios de estados muy grandes o infinitos. La aproximación estándar suele ser la estimación de una función de valor de la cual se obtiene indirectamente una política. En este módulo exploramos otra manera de abordar el problema representando la propia política con una función de aproximación totalmente independiente de la función de valor.

Los **gradientes de política** (en inglés *policy gradients*, PG), igual que las DQN, combinan el aprendizaje por refuerzo con las técnicas del *deep learning* (DL). En ambos casos el objetivo es ajustar una red neuronal para aproximar una función a partir de la aproximación de su gradiente, mediante SGD, y usando este gradiente para actualizar los parámetros de la red. Como veíamos en el módulo anterior, la diferencia con el DL es que no disponemos de etiquetas de los datos de entrada, lo que hace más complejo o difícil el ajuste de hiperparámetros. Además, en PG nuestro objetivo será maximizar el rendimiento, por lo que la actualización de los parámetros se hará en la dirección del gradiente.

Dado que PG es un método de aprendizaje por refuerzo libre de modelo (*model-free*), igual que las DQN, no habrá ningún modelo que describa la dinámica del entorno. Es por eso que en cada instante t, se obtendrá y aplicará una acción a_t sugerida por la política, y se registrarán la recompensa r_t y el nuevo estado s_{t+1}.

1.1 Parametrización de la política

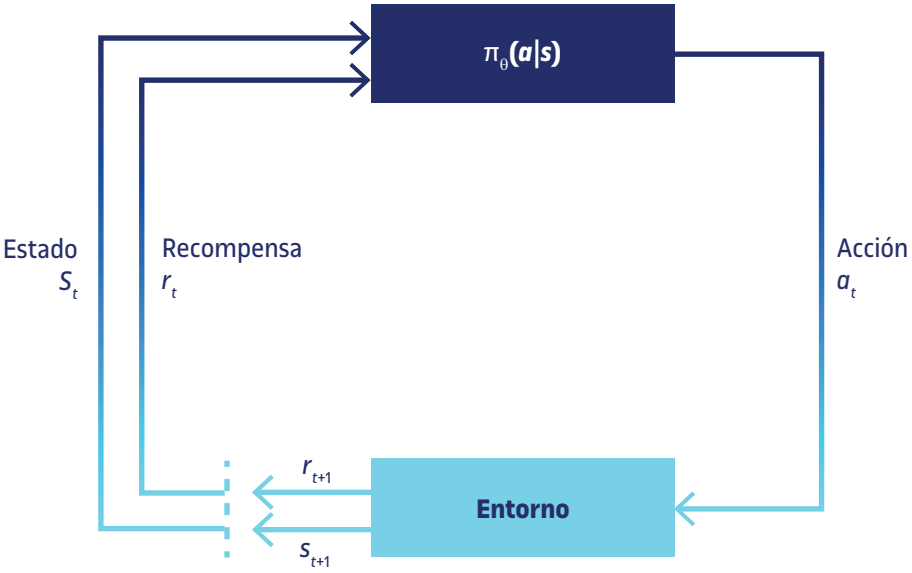
En los métodos de aproximación de la función de valor, donde aproximábamos la función de valor o de acción-valor (función Q), la elección sobre qué acción tomar en un estado dado venía determinada por la máxima recompensa esperada que se puede obtener en ese estado. Es decir, la única política existente, la que nos dice qué hacer en cada estado, era generada directamente de la estimación de la función de valor (o de la función de valor de acción). Así, en el caso de la función de valor Q hubiéramos podido definir la política formalmente como:

$$\pi(s) = \text{argmax}_a Q(s,a) \tag{1}$$

lo que nos da como política la acción con mayor Q.

Ahora consideramos otra aproximación para resolver el problema de los procesos de decisión de Markov (MDP) en espacios de estados de alta dimensionalidad: los **gradientes de política** (*policy gradients*, PG). Los PG consideran la política como una entidad separada e independiente de la función de valor. En este caso el modelo aprende directamente la política π_θ, que dado un estado nos dice qué acción realizar (figura 1), sin necesidad de saber su función de valor, es decir que por cada estado tenemos una distribución de acciones. El objetivo final seguirá siendo maximizar la recompensa, pero en este caso siguiendo una política parametrizada por θ.

Figura 1. Esquema del proceso de aprendizaje por refuerzo profundo mediante la política



Imaginemos, por ejemplo, que estamos de turismo visitando una ciudad donde no hemos estado antes y no disponemos de GPS ni de mapa ni nada que nos pueda guiar para llegar a ese museo al que queremos ir. Tenemos la opción de tomar una dirección y evaluarla respecto de nuestra posición inicial y de dónde queremos llegar; según el resultado (recompensa), tomamos otra dirección y la evaluamos; y así sucesivamente hasta conseguir el objetivo, evaluando la función de valor en cada acción que tomamos. Otra opción es preguntar a un residente de esa ciudad. En ese caso el residente nos dará una serie de instrucciones: «gira a la derecha, luego sigue recto, traspasas una gran avenida, y justo después cuando veas un monumento de un soldado gira a la izquierda y al fondo verás el museo»; es decir, nos estará dando una política que seguir. Vemos así que, en este caso, seguir la política será mucho más eficiente que ir calculando la función de valor en cada paso que damos; llegaremos mucho antes al museo.

Así, en este tipo de métodos, dado un estado, el agente obtiene las probabilidades para cada acción, elige la acción determinada por la política, observa el siguiente estado y obtiene una recompensa. Matemáticamente, estas probabilidades de acción se modelan con un estimador de función que, generalmente, suele ser una red neuronal. Cada vez que el agente interactúa con el entorno:

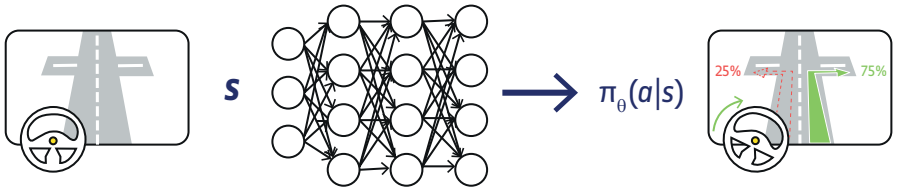
1) la red recibe el estado como entrada (píxeles o información del estado),

2) se ajustan los parámetros θ de la red neuronal de modo que las mejores acciones se presenten con más probabilidad en el futuro,

3) la red devuelve la distribución de probabilidad de las acciones.

La figura 2 muestra esquemáticamente este proceso tomando como ejemplo la conducción de un coche. La red nos proporciona el muestreo de probabilidades de cada acción posible en ese estado. En este caso, el coche puede girar a izquierda o a derecha, pero vemos que la acción de girar a la izquierda tiene menos importancia (dará menos recompensa) que girar a la derecha. En cada estado la política indicará qué opción es mejor.

Figura 2. Esquema del proceso de aprendizaje de una red neuronal de política parametrizada por θ. La red recibirá un estado como entrada y proporcionará una distribución de probabilidad de acciones a la salida



El proceso se repite hasta terminar el «juego». En ese momento se actualizan los parámetros de la red (*backpropagation*) para la próxima «partida» (episodio), como en todo MDP. Vemos que esto difiere de los métodos de la función de valor, en los que la actualización se realizaba tras cada acción tomada. Intuitivamente en el ejemplo del coche, una práctica constante (cada episodio) es la clave para aprender a conducir o para aprenderse un recorrido. Al final, el proceso global no deja de ser una cuestión de entrenar los propios instintos. Lo mismo ocurre en PG, en el que entrenamos una política basándonos en las observaciones, y el continuo entrenamiento nos permite ir afinando la política a seguir.

De este modo, la red permite al agente moverse libremente, pero con cada nuevo «juego» proporciona probabilidades más adecuadas para cada acción, llevando al agente a obtener mejores resultados (será más eficiente).

Así, con los gradientes de política lo que haremos será parametrizar directamente la política según:

$$\pi(a|s,\theta) = Pr\{A_t = a|S_t = s,\theta_t = \theta\} \quad (2)$$

que expresa la probabilidad de que la acción a sea tomada en el tiempo t dado un entorno en un estado s y un tiempo t con un vector de parámetros $\theta \in \mathbb{R}$.

Esta distribución $\pi(a|s,\theta)$ puede ser descrita en dos tipos de espacios de acción

Nota

En este texto usamos la notación a y s para referirnos a las acciones y los estados. Sin embargo, es importante notar que en la literatura, cuando se habla de estas mismas acciones y estados en el campo de los gradientes de política muchas veces se utiliza la notación u y x, respectivamente.

diferentes:

1) Espacios discretos: la distribución será normalmente categórica con una salida *softmax*.

2) Espacios continuos: la salida serán los parámetros de una distribución continua, como podrían ser la media y la varianza de una distribución Gaussiana.

1.2 Tipos de políticas

Antes de adentrarnos en los PG, es importante tener presentes los dos tipos de políticas: deterministas y estocásticas.

Una política será **determinista** cuando dado un estado la función devuelve directamente la acción a seguir: $\pi(s) = a$, es decir, la que proporciona la máxima recompensa. Este tipo de políticas se aplican en entornos deterministas en los que no hay incertidumbre sobre lo que se obtiene si se toma una acción u otra. Por ejemplo, en el juego del ajedrez, si movemos una pieza de la casilla B3 a la B4, nadie dudará de que la pieza pasará a estar en la casilla B4. El gradiente en este caso únicamente integra sobre el espacio de estados. Esta es la política que obtenemos indirectamente en los métodos DQN, en los que utilizamos la función de valor.

Una política será **estocástica** cuando dado un estado la función devuelve una distribución continua de probabilidad de las acciones a tomar: $\pi(a|s) = P(A_t|S_t)$. Este tipo de políticas se aplican en entornos inciertos, donde lo que obtenemos es una probabilidad distinta de tomar una acción u otra. Por ejemplo, la decisión de un robot de girar a derecha puede ser del 40 % y a izquierda del 60 %. El gradiente en este caso integra sobre los espacios de estados y de acciones a la vez. Gracias a esta distribución continua de probabilidad, una política estocástica será muy útil en espacios continuos. Una vez conocida esta distribución, se evalúa la política y se ajustan los parámetros con tal de acercar la media de la distribución a las acciones que devolvieron una mayor recompensa. Así en cada episodio el agente adquiere más confianza a medida que se reduce la desviación entre la probabilidad de las acciones tomadas y la recompensa obtenida. Cuando esta desviación se reduce a 0 obtenemos una política determinista. Este proceso nos permite aplicar SGD con tal de encontrar una solución óptima (al menos) local.

En PG nuestro objetivo será pues ajustar una red neuronal parametrizada por θ para que aprenda una política estocástica $\pi(a|s,\theta)$. Recordemos que esta política representa una distribución de probabilidad que mapea estados con acciones, y que el fin es maximizar la función objetivo (o medida del rendimiento) $J(\theta)$. El hecho de considerar la política estocástica nos permite hacer

exploración y simplificar el problema de la optimización dado que lo que pretendemos es maximizar la suma de recompensas esperada. Con una política determinista, obtenemos una acción por cada estado; pero imaginemos la situación en la que una acción tomada implica un gran cambio en el estado: la optimización se hace más difícil. Si, en cambio, disponemos de una distribución de acciones en cada estado, podemos desplazar un poco la distribución de modo que la suma de recompensas esperada no sea tan distinta entre un estado y el siguiente.

1.3 Función objetivo

En el caso no continuo la función objetivo $J(\theta)$ de los PG se define como la suma de recompensas entre el estado inicial y el estado final de un solo episodio:

$$J(\theta) \doteq v_{\pi_\theta}(s_0) = \mathbb{E}_{\tau\sim\pi_\theta(\tau)}[\gamma^t r(s_t,a_t)] \tag{3}$$

donde $v_{\pi_\theta}(s_0)$ es el valor de la suma esperada de recompensas (teniendo en cuenta el factor de descuento γ) para una trayectoria τ que comienza en el estado s_0 y sigue una política π_θ.

En el caso continuo esta función representaría el retorno medio:

$$J(\theta) = \lim_{h\to\infty} \frac{1}{h} \sum_{t=0}^{h} E[r(s_t,a_t)|A_{0:t} \sim \pi_\theta] = \lim_{t\to\infty} E[r(s_t,a_t)|A_{0:t} \sim \pi_\theta] \tag{4}$$

En este módulo nos centraremos únicamente en el caso episódico.

1.4 Actualización de los parámetros

A diferencia del *deep learning* donde buscábamos minimizar la función de pérdida, en PG nuestro objetivo será maximizar el rendimiento descrito por $J(\theta)$. Aunque es similar a la función de pérdida, no es ninguna función del error, sino de la puntuación que se gana. Por eso la actualización y búsqueda óptima de los parámetros se hará en la dirección del gradiente, aplicando el **ascenso del gradiente**.

En este método la actualización se realiza según:

$$\theta_{t+1} \doteq \theta_t + \alpha\nabla J(\theta_t) \tag{5}$$

donde $\nabla J(\theta_t) \in \mathbb{R}^{d'}$ es una estimación estocástica cuya esperanza se aproxima

al gradiente del rendimiento con respecto a θ, y α el tamaño de cada paso o velocidad de aprendizaje.

1.5 Potencial de los métodos PG

Los métodos basados en la política presentan una serie de ventajas respecto de los métodos basados en la función de valor de acción, que enumeramos a continuación:

- **Simplifican la representación de algunos problemas**: en muchas situaciones es más sencillo definir una política de forma paramétrica que intentar obtener la función de valor.

- **Mejor convergencia**: en los métodos *value-based* la elección de una acción es, en general, muy dependiente de cualquier cambio en la estimación de los valores de acción, lo que puede implicar una fuerte oscilación durante el entrenamiento. En los gradientes de política, todo depende de una política que mejora en cada episodio, ya que el gradiente busca los mejores parámetros en cada actualización. En general, esto mejora la eficiencia y garantiza la convergencia porque nos aseguramos que llegaremos, al menos, a un máximo local (y en el mejor de los casos, al máximo global).

- **Capaces de aprender políticas estocásticas**: las funciones de valor no pueden aprender políticas estocásticas porque en este tipo de métodos la política es determinista por defecto. Pero hemos visto que los PG sí pueden. Gracias a la distribución probabilística de las acciones, el agente puede explorar el espacio de estados libremente sin tener que tomar siempre la misma acción, como veíamos en el ejemplo del coche.

- **Mayor eficiencia en espacios de acción de alta dimensionalidad o espacios de acción continuos**: con una forma estocástica de la política, los PG permiten resolver problemas donde el espacio de estados y acciones es extremadamente elevado o no es discreto, algo que no es posible con los métodos de función de valor por una cuestión de optimización: encontrar el valor que maximiza la función de valor, especialmente cuando el estimador es una función no lineal (como las redes neuronales), puede resultar muy complicado (el proceso de discretización puede ser muy costoso). Encontrar una política es mucho más sencillo.

2. El teorema de los gradientes de política

Considerando el caso no continuo, recordemos que la función objetivo $J(\theta)$ la definimos como:

$$J(\theta) \doteq v_{\pi_\theta}(s_0) \tag{6}$$

donde v_{π_θ} es el valor real de la función para π_θ, y s_0 el estado inicial. Maximizar $J(\theta)$ es equivalente a maximizar v_{π_θ}.

Como explicábamos en el subapartado 1.4, la actualización se realiza según:

$$\theta_{t+1} \doteq \theta_t + \alpha \nabla J(\theta_t) \tag{7}$$

De modo que:

$$\nabla J(\theta) = \nabla v_{\pi_\theta}(s_0) \tag{8}$$

El problema está en que el rendimiento depende, a la vez, de las acciones y de la distribución de estados donde tienen lugar estas acciones, y que ambos están regidos por la política. De esta dependencia se desprende una función del entorno totalmente desconocida, que surge del efecto de la política sobre la distribución de estados. Sin conocer esta función resulta muy complejo calcular o estimar el gradiente respecto de la política.

El **teorema de los gradientes de política** permite resolver este problema proporcionando una expresión analítica del rendimiento respecto de la política, que evita tener que hacer derivadas de la distribución de estados.

$$\nabla J(\theta) \propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla \pi(a|s,\theta) \tag{9}$$

donde $\mu(s)$ es la distribución bajo la política π (es decir, la probabilidad de estar en el estado s cuando se sigue la política π), $q_\pi(s,a)$ es la función de valor de acción para la política, y $\nabla \pi(a|s,\theta)$ es el gradiente de π dado el estado s y los parámetros θ.

El teorema nos está diciendo que $\nabla J(\theta)$ es proporcional a la suma de la función q por el gradiente de las políticas de todas las acciones en los estados en los que el agente puede estar. Este resultado es importante porque en la mayoría de los casos no es posible (o muy costoso) modelizar la distribución de estados y la dinámica del entorno a la vez. Todos los algoritmos que usan este teorema son, en consecuencia, *model-free* dado que no se modeliza el entorno.

2.1 Demostración del teorema

El teorema se puede demostrar fácilmente. Consideraremos que π y todos los gradientes son siempre función de θ, sin indicarlo explícitamente. También tendremos en cuenta que la función de valor de estado se puede escribir en términos de la función de valor de acción. De esta manera, el componente de la izquierda del teorema se puede descomponer:

$$\nabla v_\pi(s) = \nabla \left[\sum_a \pi(a|s) q_\pi(s,a) \right], \quad \text{para todo s} \in \mathbb{S} \tag{10}$$

Por la regla del producto $(f \cdot g)' = f' \cdot g + f \cdot g'$:

$$\nabla v_\pi(s) = \sum_a \left[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \nabla q_\pi(s,a) \right] \tag{11}$$

Dado que $p(s_0,r|s,a) \doteq \Pr\{S_t = s_0, R_t = r | S_{t-1} = s, A_{t-1} = a\}$:

$$\nabla v_\pi(s) = \sum_a \left[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \nabla_{s',r} p(s',r|s,a)(r + v_\pi(s')) \right] \tag{12}$$

Y como $p(s'|s,a) = \Pr(S_t = s' | S_{t-1} = s, A_{t-1} = a) = \sum_{r \in \mathbb{R}} p(s',r|s,a)$, entonces:

$$\nabla v_\pi(s) = \sum_a \left[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s'|s,a) \nabla v_\pi(s') \right] \tag{13}$$

Si desarrollamos el término $\nabla v_\pi(s')$ para s' igual que hemos hecho hasta ahora para el estado s:

$$\nabla v_\pi(s) = \sum_a \left[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s'|s,a) \right.$$

$$\left. \sum_{a'} \left[\nabla \pi(a'|s') q_\pi(s',a') + \pi(a'|s') \sum_{s''} p(s''|s',a') \nabla v_\pi(s'') \right] \right] \tag{14}$$

Siguiendo el mismo procedimiento sucesivamente, vemos que podemos generalizar la expresión tal que:

$$\nabla v_\pi(s) = \sum_{x\in\mathbb{S}} \sum_{k=0}^{\inf} \Pr(s \to x,k,\pi) \sum_a \nabla\pi(a|x) q_\pi(x,a) \tag{15}$$

donde $\Pr(s \to x,k,\pi)$ es la probabilidad de pasar del estado s a un estado x en k pasos siguiendo la política π.

Introducimos ahora este resultado en la igualdad de la Eq. 6 donde $J(\theta) = v_{\pi^\theta}(s_0)$:

$$\begin{aligned}\nabla J(\theta) &= \nabla v_\pi(s_0) \\ &= \sum_s \left(\sum_{k=0}^{\inf} \Pr(s \to x,k,\pi) \right) \sum_a \nabla\pi(a|s) q_\pi(s,a)\end{aligned} \tag{16}$$

Consideremos ahora $h(s)$ como la probabilidad de que un episodio empiece en cada estado s, y $\eta(s)$ el número de pasos de tiempo en media en cada estado s para un único episodio. Entonces podemos describir $\eta(s)$ como:

$$\eta(s) = h(s) + \sum_{\bar{s}} \eta(\bar{s}) \sum_a \pi(a|\bar{s}) p(a|\bar{s},a), \quad \text{para todo s} \in \mathbb{S}$$

Así, la distribución *on-policy* será la fracción de tiempo en cada estado, normalizada para que sume 1:

$$\mu(s) = \frac{\eta(s)}{\sum_{s'} \eta(s')}, \quad \text{para todo s} \in \mathbb{S}$$

Si introducimos estas consideraciones en la ecuación de $\nabla J(\theta)$:

$$\begin{aligned}\nabla J(\theta) &= \sum_s \eta(s) \sum_a \nabla\pi(a|s) q_\pi(s,a) \\ &= \sum_{s'} \eta(s') \sum_s \frac{\eta(s)}{\sum_{s'} \eta(s')} \sum_a \nabla\pi(a|s) q_\pi(s,a) \\ &= \sum_{s'} \eta(s') \sum_s \mu(s) \sum_a \nabla\pi(a|s) q_\pi(s,a) \\ &\propto \sum_s \mu(s) \sum_a \nabla\pi(a|s) q_\pi(s,a) \qquad \text{(c.q.d.)}\end{aligned} \tag{17}$$

3. Principales algoritmos de gradientes de política

3.1 REINFORCE

El teorema de los gradientes de política facilita una expresión proporcional al gradiente. El término al que es proporcional $\nabla J(\theta)$ según el teorema es una suma sobre los estados, ponderados por unos pesos según la frecuencia en la que aparecen bajo la política π. Es decir, que si el agente sigue esta política π se encontrará con estos estados en la proporción fijada.

Sin embargo, de la expresión de este teorema nos falta por conocer $\pi(a|s,\theta)$. Además, necesitamos que el algoritmo nos proporcione una única acción A_t por cada instante t.

Podemos reescribir el teorema para que siga alguna forma de muestreo cuya esperanza sea igual o aproximada a la expresión del teorema de la Eq. 9:

$$\nabla J(\theta) \propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla \pi(a|s,\theta)$$

$$= \mathbb{E}_\pi \left[\sum_a q_\pi(S_t,a) \nabla \pi(a|S_t,\theta) \right] \tag{18}$$

donde la esperanza simplemente nos dice que hemos cogido un conjunto de pasos del proceso (interacciones con el entorno) y promediamos el gradiente.

Si tenemos en cuenta que el término $\nabla \pi(a|s,\theta)$ lo podemos desglosar se la siguiente manera:

Nota

$\frac{dx}{x} = [\ln(x)]' = \nabla \ln(x)$

$$\nabla \pi(a|s,\theta) = \frac{\pi(a|S_t,\theta)}{\pi(a|S_t,\theta)} \nabla \pi(a|S_t,\theta)$$

$$= \pi(a|S_t,\theta) \frac{\nabla \pi(a|S_t,\theta)}{\pi(a|S_t,\theta)}$$

$$= \pi(a|S_t,\theta) \nabla_\theta \ln \pi(a|S_t,\theta) \tag{19}$$

entonces,

$$\nabla J(\theta) \propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla \pi(a|s,\theta)$$

$$= \mathbb{E}_\pi \left[\sum_a q_\pi(S_t,a) \pi(a|S_t,\theta) \nabla_\theta \ln \pi(a|S_t,\theta) \right] \tag{20}$$

Reemplazamos ahora a por el muestreo $A_t \sim \pi$:

$$\nabla J(\theta) = \mathbb{E}_\pi \left[q_\pi(S_t,A_t) \nabla_\theta \ln \pi(A_t|S_t,\theta) \right] \tag{21}$$

y dado que $\mathbb{E}_\pi[G_t|S_t,A_t] = q_\pi(S_t,A_t)$, con G_T el retorno, finalmente obtenemos:

$$\nabla J(\theta) = \mathbb{E}_\pi \left[G_t \nabla_\theta \ln \pi(A_t|S_t,\theta) \right] \tag{22}$$

donde la esperanza del término $G_t \nabla_\theta \ln \pi(A_t|S_t,\theta)$ es el propio gradiente, y puede ser muestreada en cada instante t. Este es el conocido algoritmo clásico de **REINFORCE**. Este algoritmo forma parte de una familia de algoritmos propuestos por Williams (1992), aunque la prueba de su convergencia fue introducida por Sutton *et al.* (2000). Con este muestreo podemos ahora calcular la actualización del algoritmo según:

$$\theta_{t+1} \doteq \theta + \alpha G_t \ln \pi(A_t|S_t,\theta) \tag{23}$$

donde $\gamma^t = 1$ (factor de descuento) en este algoritmo porque consideramos el caso sin descuento. En el caso general:

$$\theta_{t+1} \doteq \theta + \alpha \gamma^t G_t \ln \pi(A_t|S_t,\theta) \tag{24}$$

Podemos ver que el hecho de que la actualización de REINFORCE incorpore el retorno G_t implica que esta se realiza al final de cada episodio, en línea con los algoritmos de Montecarlo. Cada actualización es proporcional al producto del retorno y el gradiente de la probabilidad de tomar la acción tomada, dividido por la probabilidad de tomar esa acción (definición que, como hemos visto, queda simplificada en la expresión del logaritmo natural). Intuitivamente, este logaritmo no es más que el vector de dirección en el espacio de parámetros: la actualización incrementa los parámetros del vector en esa dirección proporcionalmente al retorno (es decir, nos interesa que las acciones con mayor retorno tengan una probabilidad más alta de ser elegidas, y viceversa), e inversamente proporcional a la probabilidad de la acción (justamente para equilibrar esta prioridad que se da a las acciones de mayor retorno). Es decir, en otras palabras, que la actualización esperada sobre un episodio está en la misma dirección que el gradiente de rendimiento.

Como vimos anteriormente, el retorno G_t es la recompensa acumulada, es decir, el valor acumulado de todas las recompensas futuras. Pero dado que el número de episodios podría ser infinito, esta suma podría no converger nunca. Por esta razón redefiníamos el retorno introduciendo un factor de descuento $\gamma \in [0,1]$.

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... \tag{25}$$

Esta recompensa acumulada con factor de descuento también se conoce en inglés como *discounted rewards*. Recordemos que el objetivo del agente es seleccionar acciones que maximicen el retorno. Con la expresión anterior, esta maximización, en lugar de realizarse sobre la suma de recompensas, ahora se realizará sobre la recompensa actual más γ veces la recompensa siguiente, más γ^2 veces la siguiente, etc. Así, suponiendo que el agente se encuentra en el paso $t = 0$, el efecto que tendrá la recompensa del paso futuro $t + 2000$ será prácticamente irrelevante para la decisión actual del agente.

Para ejemplificar este proceso, imaginemos que el episodio entero dura $T = 100$ pasos. Como acabamos de explicar, el retorno en cada paso debe incluir las recompensas futuras. Así, empezando por el último paso, tendremos que:

$t = 100 \quad G_{100} = 0$

$t = 99 \quad G_{99} = r_{100} + \gamma G_{100} = r_{100}$

$t = 98 \quad G_{98} = r_{99} + \gamma G_{99} = r_{99} + \gamma r_{100}$

$t = 97 \quad G_{97} = r_{98} + \gamma G_{98} = r_{98} + \gamma(r_{99} + \gamma r_{100}) = r_{98} + \gamma r_{99} + \gamma^2 r_{100}$

...

$t = 0 \quad G_0 = r_1 + \gamma G_1 = r_1 + \gamma r_2 + \gamma^2 r_3 + ... + \gamma^{99} r_{100}$

De modo que la suma acumulada con factor de descuento en el caso episódico se puede simplificar como:

$$G_t = R_{t+1} + \gamma G_{t+1} = \sum_{k=0}^{T-t-1} \gamma^k R_{t+k+1} \tag{26}$$

De forma equivalente, esta ecuación se suele escribir habitualmente como:

$$G_t = \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k \tag{27}$$

El Algoritmo 1 muestra la implementación del método REINFORCE.

Algorithm 1 REINFORCE

Entrada: política parametrizada diferenciable $\pi(a|s,\theta)$ a evaluar
Parámetro del algoritmo: paso $\alpha > 0$
Inicializar los parámetros de la política $\theta \in \mathbb{R}^{d'}$ (p. ej., $\theta = 0$)
for cada episodio **do**
  Generar un episodio $S_0, A_0, R_1, S_1, A_1, ...R_T, S_T$ usando $\pi(\cdot|\cdot,\theta)$
  for cada paso del episodio, $t = 0,1,...T-1$: **do**
    $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$
    $\theta \leftarrow \theta + \alpha^{\theta} \gamma^t \nabla \ln \pi(A_t|S_t,\theta)$
  end for
end for

Comparemos ahora este algoritmo con los algoritmos de DQN. Es importante darse cuenta de que muchas de las técnicas que necesitábamos añadir entonces para mejorar los métodos DQN aquí no son necesarios. Si recordamos, el algoritmo de DQN inicial tenía ciertas flaquezas que resolvíamos incluyendo el ϵ-*greedy*, el *buffer* de repetición de experiencias (*experience replay*) y la red objetivo. Con los PG, la exploración ya se realiza automáticamente gracias a que la red devuelve una distribución uniforme de probabilidad de las posibles acciones; el comportamiento aleatorio del agente, necesario especialmente al principio del proceso de aprendizaje, ya está considerado, no hace falta ningún ϵ-*greedy*. Tampoco se requiere ningún *buffer* de repetición de experiencias, porque no podemos entrenar con los datos obtenidos de la política anterior, ya que el agente se orienta por la política no por los valores obtenidos de sus acciones; esto facilita la convergencia del algoritmo, y esta usualmente es más rápida que en DQN, aunque puede requerir más interacción con el entorno y hacer el proceso más lento. Por último, la red objetivo la utilizábamos para romper la correlación en el cálculo de los valores Q, pero en los PG no calculamos ningún valor de Q, por lo que esta técnica como tal no nos sirve directamente.

Vemos pues que los PG, y en concreto el método REINFORCE, presentan buenas propiedades de convergencia, aunque no están exentos de inconvenientes, puesto que, por ejemplo, el aprendizaje puede ser más lento debido a su alta variabilidad y también sufren de correlación entre estados.

3.2 REINFORCE con línea de base

Efectivamente, uno de los principales problemas de los métodos clásicos basados en la política es su fuerte variabilidad. Imaginemos que en un episodio el agente tiene «mucha suerte» y acierta siempre las mejores acciones obteniendo una recompensa en el último paso del episodio muchísimo más grande que en el primer paso. En los siguientes episodios no tiene por qué ocurrir lo

mismo, es decir, que lo haga casualmente tan bien (figura 3). Pero a efectos prácticos, este episodio (en el que lo ha hecho muy bien) tendrá más peso en el gradiente final, aunque no refleja la realidad del problema y hace el sistema más inestable.

Figura 3. Ejemplificación de la alta variabilidad del método REINFORCE. El agente puede hacerlo muy bien en cada paso de un episodio por casualidad (por ejemplo, el episodio 2 en la imagen), pero en otros tomar acciones menos buenas, causando una fuerte variabilidad entre los retornos de todos los episodios, afectando al gradiente final

Para solucionar el problema de la variabilidad, el teorema de los gradientes de política se puede generalizar (Sutton, 2000; Sutton y Barto, 2018, 2ª edición) introduciendo una línea de base (o *baseline*), $b(S_t)$, tal que:

$$\nabla J(\theta) \approx \mathbb{E}_T \left[\sum_{t=0}^{T-1} \nabla_\theta \ln \pi_\theta(A_t|S_t,\theta) \left(G_t - b(S_t)\right) \right] \tag{28}$$

Esta línea de base puede ser cualquier función o valor aleatorio que sea independiente de la acción, y puede ser estrictamente 0, puesto que de esta manera garantizamos que no altera el teorema original, y la ecuación anterior seguiría siendo válida:

$$\sum_a b(s)\nabla\pi(a|s,\theta) = b(s)\nabla\sum_a \pi(a|s,\theta) \tag{29}$$

$$= b(s)\nabla 1 \tag{30}$$

$$= 0 \tag{31}$$

Queda probado así que el valor esperado de la recompensa no queda nunca alterado.

De este modo conseguimos reducir la variabilidad de la estimación del gradiente sin añadir ningún tipo de sesgo, mejorando así el aprendizaje de una política óptima. A este algoritmo se le conoce como *REINFORCE con línea de*

base. La actualización será ahora:

$$\theta_{t+1} \doteq \theta + \alpha(G_t - b(S_t))\nabla \ln \pi(A_t|S_t,\theta) \tag{32}$$

¿Qué función o valor escoger para la línea de base $b(s)$? Generalmente nos encontramos con dos tipos o técnicas:

1) *Whitening*. Es la técnica más sencilla y común en la búsqueda de una mejor optimización del proceso de aprendizaje. Consiste en normalizar el retorno de cada paso del episodio considerando como *baseline* ya sea la media del retorno $\bar{G}_t$, la media dividida por la desviación estándar del retorno $\frac{\bar{G}}{\sigma_G}$, o la media estandarizada $\frac{G_i - \bar{G}}{\sigma_G}$.

En el caso, por ejemplo, de la media dividida por la desviación estándar del retorno, tendríamos:

$$G'_t = G_t - b(S_t) = G_t - \frac{\bar{G}}{\sigma_G} \tag{33}$$

Y, en consecuencia, teniendo en cuenta la ecuación 32, la actualización de θ sigue:

$$\theta_{t+1} \doteq \theta_t + \alpha G'_t \nabla \ln \pi(A_t|S_t,\theta) \tag{34}$$

2) Línea de base aprendida. Como se comentaba, técnicamente, cualquier línea de base que no dependa de la acción puede ser usada. Sin embargo, la opción más ideal sería usar el valor verdadero del estado en la política actual, de modo que si el retorno obtenido es mejor que el esperado, los gradientes también mejorarán.

Pero obtener el valor verdadero de un estado es muy complicado, ya que requiere disponer de un número infinito de muestras. La técnica de «línea de base aprendida» (o *learned baseline*) consiste en calcular una aproximación para la estimación del retorno esperado bajo la política actual. Es decir, lo que propone es intentar aprender una función de valor en paralelo al aprendizaje de la política.

Para ello podemos considerar el *baseline* directamente como una estimación del valor de estado, $b(S_t) = \hat{v}(S_t,\mathbf{w})$, en la cual la variable objetivo puede ser el propio retorno G_t, ya que este valor no deja de ser una muestra de la función de valor de la política actual. De modo que ahora G'_t será:

$$G'_t = G_t - b(s) = G_t - \hat{v}(S_t,\mathbf{w}) \tag{35}$$

Es importante resaltar que predecir el retorno esperado en dicho estado de esta forma implica tener que considerar dos tipos de pérdidas y calcular sus correspondientes actualizaciones de parámetros:

a) Pérdida resultante del error cuadrático medio entre el valor aprendido y el retorno observado G_t. La actualización de $\mathbf{w}$ de la función de valor estimada será:

$$w_{t+1} \doteq w_t + \alpha_{\mathbf{w}} G'_t \nabla \hat{v}(S_t, \mathbf{w}) \tag{36}$$

b) Pérdida del propio algoritmo REINFORCE al substraer la línea de base aprendida. En este caso los parámetros θ se actualizarán de la misma forma que en el caso anterior pero usando la nueva G'_t y una velocidad de aprendizaje α_θ específica:

$$\theta_{t+1} \doteq \theta_t + \alpha_\theta G'_t \nabla \ln \pi(A_t|S_t, \theta) \tag{37}$$

En la práctica, únicamente estamos añadiendo una salida extra a la red neuronal y utilizando esta salida (el valor aprendido) como línea de base. Y los parámetros $\alpha_{\mathbf{w}}$ y α_θ serán hiperparámetros a ajustar.

Vemos pues que, para la técnica de *whitening*, podemos directamente reutilizar el Algoritmo 1 del caso clásico de REINFORCE, simplemente restando el *baseline* $b(s)$ al retorno G_t. Para la técnica de estimación de la función de valor de estado, en cambio, se requieren algunos pasos más. El Algoritmo 2 muestra la implementación del algoritmo de REINFORCE con línea de base con esta segunda técnica de determinación de $b(s)$:

Algorithm 2 REINFORCE con línea de base de estimación de la función de valor de estado

Entrada: política parametrizada diferenciable $\pi(a|s,\theta)$ a evaluar
Entrada: parametrización de un valor del estado diferenciable $\hat{v}(s,\mathbf{w})$
Parámetros del algoritmo: paso $\alpha^\theta > 0$, $\alpha^{\mathbf{w}} > 0$
Inicializar los parámetros de la política $\theta \in \mathbb{R}^{d'}$ y los pesos del valor del estado $\mathbf{w} \in \mathbb{R}^{d'}$ arbitrariamente (p. ej. $\theta = 0$, $\mathbf{w} = 0$)
for cada episodio **do**
  Generar un episodio $S_0, A_0, R_1, S_1, A_1, ...R_T, S_T$ usando $\pi(\cdot|\cdot,\theta)$
  for cada paso del episodio, $t = 0,1,...T-1$: **do**
    $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$
    $\delta \leftarrow G - \hat{v}(S_t, \mathbf{w})$
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla \hat{v}(S_t, \mathbf{w})$
    $\theta \leftarrow \theta + \alpha^\theta \gamma^t \nabla \ln \pi(A_t|S_t,\theta)$
  end for
end for

Observamos que, dado que REINFORCE es un método de Montecarlo, el proceso de aprendizaje tanto de los parámetros θ como de los pesos $\mathbf{w}$ se hace siguiendo este método para ambos, y con una velocidad de aprendizaje α distinta para cada uno. Respecto del algoritmo anterior, α^{θ} es equivalente a α; su valor depende del rango de variación de las recompensas y de la parametrización de la política. Para determinar $\alpha^{\mathbf{w}}$ se puede aplicar alguna regla lineal, tal que $\alpha^{\mathbf{w}} = 0.1/\mathbb{E}[||\nabla\hat{v}(S_t,\mathbf{w})||^2_{\mu}]$.

4. Inconvenientes de los gradientes de política

Tal y como hemos visto, los PG, y en concreto el algoritmo de REINFORCE, suelen presentar muy buena convergencia con respecto a los algoritmos DQN de base. Muchos de los problemas asociados con las DQN los pudimos corregir añadiendo técnicas o implementando alternativas a las aproximaciones de las funciones, llegando a algoritmos muy potentes del estado del arte. Los PG permiten obtener una buena convergencia muchas veces directamente, sin tener que incorporar otras técnicas que contribuyan a ello. Sin embargo, no están exentos de dificultades y en según qué entornos o dimensión del espacio de estados pueden conducir a resultados menos óptimos o pueden sufrir de problemas de estabilidad. Añadiendo una línea de base hemos visto que hemos podido reducir un poco el problema de la variabilidad, pero existen otros aspectos o particularidades que es importante destacar:

- **Exploración**. Si bien hemos dicho que gracias a la distribución de probabilidades de las acciones ya estamos considerando el comportamiento aleatorio del agente y que no necesitamos ningún ϵ-*greedy*, existe una alta probabilidad de que el agente converja en alguna política óptima a nivel local y deje de explorar el entorno.

- **Correlación**. Igual que ocurría con las DQN, y en todo problema de aprendizaje por refuerzo profundo, los estados están muy correlacionados entre si. Como explicábamos, en PG no podemos usar ningún *buffer* de repetición de experiencias porque los ejemplos generados por la política anterior (del anterior episodio) no pueden ser utilizados con la nueva política dado que no se corresponden.

- **Variabilidad**. A pesar de que añadiendo una línea de base reducimos la variabilidad, podemos encontrarnos en situaciones donde las diferencias de recompensa entre el primer paso y el último dentro de un mismo episodio sigan siendo muy dispares, influyendo negativamente en el proceso de aprendizaje.

- **Convergencia lenta**. La actualización del gradiente de los PG al final de cada episodio se realiza a partir de una estimación del gradiente obtenida de los sucesivos ajustes de parámetros acumulados durante ese episodio (métodos de Montecarlo). No poder actualizar la política hasta el final de cada episodio hace que el aprendizaje sea mucho más lento que con las DQN, las cuales siguen la técnica del aprendizaje TD y la actualización se realiza en cada paso. La solución con PG suele ser más óptima y la conver-

gencia está más garantizada, pero esta es más lenta, puesto que tienden a ir hacia un máximo local en lugar de un máximo global, ralentizando así el entrenamiento.

En el siguiente módulo veremos otro tipo de métodos de DRL, los actor-crítico (AC), que justamente pretenden dar solución a estos problemas. Estudiaremos los dos algoritmos principales que ayudan a mejorar el tema de la correlación, la variabilidad y de la actualización en cada episodio. Y comentaremos brevemente el amplio espectro de algoritmos del estado del arte que buscan reducir aún más estos inconvenientes de manera cada vez más eficiente, mejorando el rendimiento y la optimización del proceso de aprendizaje.

Resumen

En este módulo hemos estudiado un segundo tipo de métodos de DRL, los gradientes de política, que a diferencia de las DQN, están basados en la política. Una ventaja de estos métodos es que optimizan directamente el comportamiento del agente, indicando lo que debe hacer sin necesidad de calcular el valor para cada acción para poder dar una respuesta. Los parámetros de la política se actualizan con SG en la dirección del gradiente (ascenso del gradiente)

Este tipo de métodos permiten determinar las probabilidades de cada acción posible dado un estado, facilitando la exploración del entorno por el agente. Además, el teorema de los gradientes de política proporciona una expresión exacta y sencilla de implementar para determinar el rendimiento de la política según sus parámetros, evitando operaciones más complejas sobre la distribución de estados. Este teorema es fundamental para todo tipo de métodos de gradientes de política, como los dos algoritmos, REINFORCE y REINFORCE con línea de base, estudiados.

El problema con estos algoritmos es principalmente su alta variabilidad entre los retornos de cada episodio. Ambos siguen un proceso de Montecarlo, por lo que sus políticas no se actualizan hasta el final de cada episodio y tampoco pueden usar experiencias anteriores porque las políticas entre dos episodios pueden ser muy distintas; esto también conduce a un aprendizaje más lento puesto que requieren más interacción con el entorno. Aun así, su convergencia suele ser más óptima que en los algoritmos DQN, ofreciendo mejores resultados en muchos casos. Cuando añadimos una línea de base al método REINFORCE conseguimos reducir y suavizar el problema de la variabilidad, aunque no eliminarlo. En el próximo módulo veremos otro método de DRL que nos permitirá reducir mucho más esta varianza.

Glosario

aprendizaje profundo *m* Conjunto de algoritmos de aprendizaje automático que intenta modelar abstracciones de alto nivel utilizando redes neuronales de múltiples capas o niveles.
sigla **DL**
en deep learning

deep learning *m* Véase **aprendizaje profundo**.

DL *m* Véase **aprendizaje profundo**.

i.i.d. *m* Véase **independientes e idénticamente distribuidos**.

independent and identically distributed *m* Véase **independientes e idénticamente distribuidos**.

independientes e idénticamente distribuidos *m* Conjunto de datos o variables aleatorias que son mutuamente independientes entre sí y además cada variable tiene la misma distribución de probabilidad.
sigla **i.i.d.**
en independent and identically distributed

Bibliografía

Bahumi, R. (2019). *Reinforcement Learning - A mathematical introduction to Policy Gradient* [Fecha de consulta: 14 mayo de 2020].

Lapan, M. (2018). *Deep Reinforcement Learning Hands-On*. Expert Insight.

Li, Y. (2018). *Deep Reinforcement Learning: An Overview*. arXiv:1810.06339.

Silver, D.. *Univerity College London Course on RL*. [Fecha de consulta: 14 de mayo de 2020]. <http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html>.

Sutton, R. S.; Barto, A. G. (2019). *Reinforcement Learning. An introduction*. Cambridge, MA: MIT Press.

Sutton, R. S.; McAllester, D.; Singh, S.; Mansour, Y. (2000). *Policy gradient methods for reinforcement learning with function approximation*. Advances in neural information processing systems.

Thomas, S.. *Deep Reinforcement Learning Course*. [Fecha de consulta: 25 de mayo de 2020]. <https://simoninithomas.github.io/Deep_reinforcement_learning_Course/>.

Weng, L. (2018). *Policy Gradient Algorithms*. [Fecha de consulta: 10 de mayo de 2020]. <https://lilianweng.github.io/lil-log/>.

Williams, R. J. (1992). *Simple statistical gradient-following algorithms for connectionist reinforcement learning*. Boston, MA: Springer.