Inteligencia Artificial

CC-421

César Lara Avila

Universidad Nacional de Ingeniería

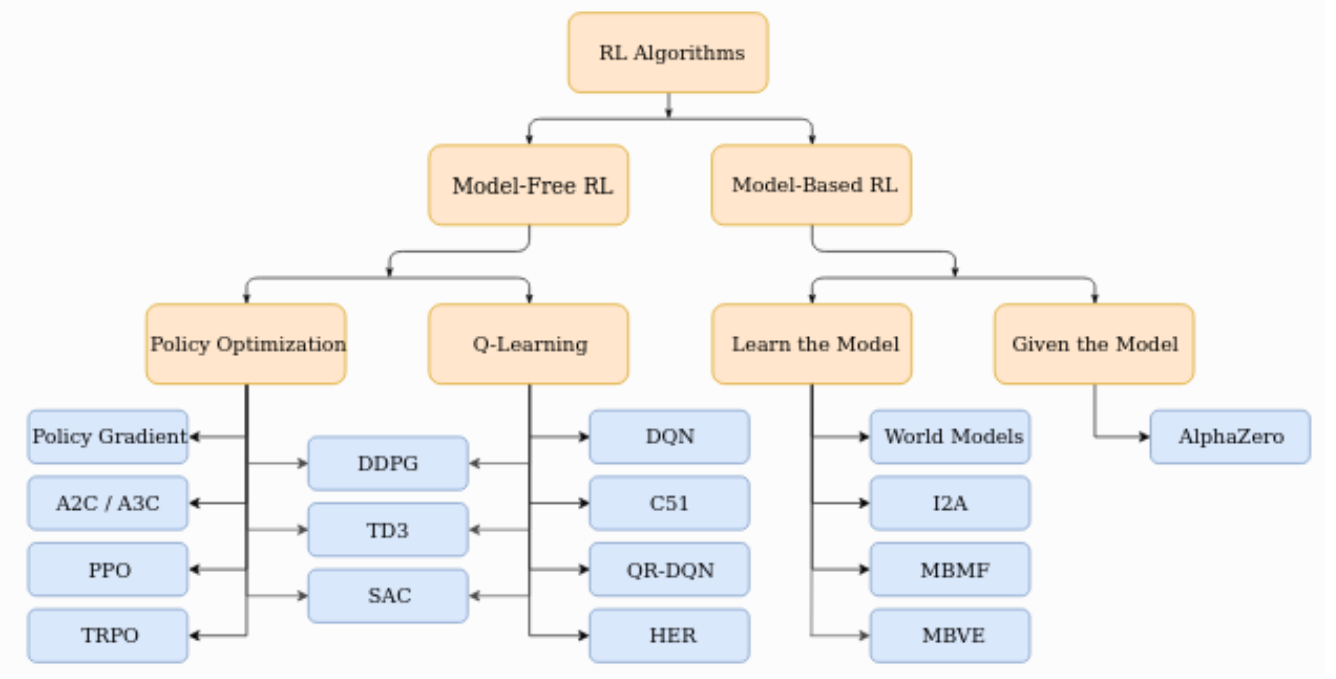
(actualización: 2021-01-25)

Bienvenidos

Aprendizaje por Refuerzo - Parte 2

- Taxonomía de algoritmos RL
- RL sin modelos y basados en modelo
- ¿Qué aprender en RL sin modelos?
- ¿Qué aprender en RL basado en modelos?
- Optimización de políticas
- Otras formas de la políticas de gradiente

Taxonomía de algoritmos RL



RL sin modelos y basados en modelo

Uno de los puntos de ramificación más importantes en un algoritmo RL es la cuestión de si el agente tiene acceso (o aprende) un **modelo del entorno**.

Por modelo de entorno, nos referimos a una función que predice transiciones de estado y recompensas.

La principal ventaja de tener un modelo es que permite al agente planificar pensando en el futuro, viendo lo que sucedería con una variedad de opciones posibles y decidiendo explícitamente entre sus opciones.

Luego, los agentes pueden manejar los resultados de la planificación anticipada en una política aprendida.

El principal inconveniente de este enfoque es que el agente no dispone de un modelo real del entorno. Si un agente quiere usar un modelo en este caso, tiene que aprender el modelo puramente por experiencia, lo que crea varios desafíos.

Los algoritmos que utilizan un modelo se denominan **métodos basados en modelos** y los que no **modelos sin modelos**.

Si bien los **métodos sin modelo** renuncian a las ganancias potenciales en la eficiencia de la muestra al usar un modelo, tienden a ser más fáciles de implementar y ajustar.

Los métodos sin modelos son más populares y se han desarrollado y probado de forma más exhaustiva que los métodos basados en modelos.

¿Qué vamos a aprender?

Otro punto crítico de ramificación en un algoritmo RL es la cuestión de qué aprender. La lista puede incluir:

- Políticas, estocásticas o deterministas
- Funciones de acción-valor (Q-funciones)
- Funciones de valor
- Modelos del entorno.

¿Qué aprender en RL sin modelos?

Hay dos enfoques principales para representar y entrenar a los agentes con RL sin modelos:

Optimización de políticas

Los métodos de esta familia representan una política explícitamente como $\pi_\theta(a|s)$.

Optimizan los parámetros θ directamente mediante el ascenso del gradiente en el rendimiento del objetivo $J(\pi_\theta)$ o indirectamente, maximizando las aproximaciones locales de $J(\pi_\theta)$.

Esta optimización casi siempre se realiza según la política, lo que significa que cada actualización solo utiliza los datos recopilados mientras actúa de acuerdo con la versión más reciente de la política.

La optimización de políticas también suele implicar el aprendizaje de un aproximador $V_\phi(s)$ para la función de valor en la política $V^\pi(s)$, que se utiliza para averiguar cómo actualizar la política.

Un par de ejemplos de métodos de optimización de políticas son:

- **A2C/A3C**, que realiza un ascenso en pendiente para maximizar directamente el rendimiento
- **PPO**, cuyas actualizaciones maximizan indirectamente el rendimiento, maximizando en su lugar una función objetivo sustituta que da una estimación de cuánto $J(\pi_\theta)$ cambiará como resultado de la actualización.

Q-Learning

Los métodos de esta familia aprenden un aproximador $Q_\theta(s, a)$ para la función acción-valor óptima $Q^*(s, a)$.

Por lo general, utilizan una función objetivo basada en la ecuación de Bellman. Esta optimización casi siempre se realiza **fuera de la política**, lo que significa que cada actualización puede utilizar los datos recopilados en cualquier momento durante el entrenamiento, independientemente de cómo el agente eligió explorar el entorno cuando se obtuvieron los datos.

La política correspondiente se obtiene mediante la conexión entre Q^* y π^*: las acciones realizadas por el agente de Q-learning vienen dadas por:

$$a(s) = \arg\max_a Q_\theta(s, a).$$

Ejemplos de métodos de Q-learning incluyen:

- **DQN**, un clásico que lanzó sustancialmente el campo del RL profundo
- **C51**, una variante que aprende una distribución sobre el retorno cuya expectativa es Q^*.

Los métodos de Q-learning pueden reutilizar los datos de manera más eficaz que las técnicas de optimización de políticas por lo pueden ser más eficientes.

Los métodos de optimización de políticas, tienden a ser más estables y confiables. Por el contrario, los métodos de Q-learning solo optimizan indirectamente el rendimiento del agente, al entrenar Q_θ para satisfacer una ecuación de Bellman.

Interpolación entre la optimización de políticas y Q-Learning

La optimización de políticas y el Q-learning no son incompatibles y existe una gama de algoritmos que viven entre los dos extremos. Los algoritmos que viven en este espectro son capaces de equilibrar cuidadosamente las fortalezas y debilidades de cada lado.

Ejemplos incluyen:

- **DDPG**, un algoritmo que aprende simultáneamente una política determinista y una Q-función utilizando cada uno para mejorar el otro,
- **SAC**, una variante que utiliza políticas estocásticas, regularización de entropía y algunos otros trucos para estabilizar el aprendizaje y obtener una puntuación más alta que DDPG en los benchmarks estándars.

¿Qué aprender en RL basado en modelos?

A diferencia de RL sin modelos, no hay una pequeña cantidad de grupos de métodos fáciles de definir para el RL basados en modelos: hay muchas formas ortogonales de usar modelos.

Planificación pura

Utiliza técnicas de planificación puras como el **MPC** (model-predictive control) para seleccionar acciones.

En MPC, cada vez que el agente observa el entorno, calcula un plan que es óptimo con respecto al modelo, donde el plan describe todas las acciones para asumir una ventana fija de tiempo después del presente.

El agente luego ejecuta la primera acción del plan e inmediatamente descarta el resto. Calcula un nuevo plan cada vez que se prepara para interactuar con el entorno, para evitar utilizar una acción de un plan con un horizonte de planificación más corto .

- **MBMF** explora MPC con modelos de entorno aprendidos en algunas tareas de referencia estándar para RL profundo.

Iteración experta

Un seguimiento sencillo de la planificación pura implica el uso y el aprendizaje de una representación explícita de la política, $\pi_\theta(a|s)$.

El agente utiliza un algoritmo de planificación (como **Monte Carlo Tree Search**) en el modelo, generando acciones candidatas para el plan tomando muestras de la política actual.

El algoritmo de planificación produce una acción que es mejor que la que habría producido la política por sí sola, por lo que es un "experto" en relación con la política.

Posteriormente, la política se actualiza para producir una acción más parecida a la salida del algoritmo de planificación.

- El algoritmo **ExIt** utiliza este enfoque para entrenar redes neuronales profundas para jugar Hex. **AlphaZero** es otro ejemplo de este enfoque.

Aumento de datos para métodos sin modelo

Utiliza un algoritmo RL sin modelo para entrenar una política o una Q-función, pero 1) aumenta las experiencias reales con experiencias ficticias al actualizar el agente o 2) usa solo la experiencia ficticia para actualizar el agente.

- Consulta **MBVE** para ver un ejemplo de cómo aumentar experiencias reales con experiencias ficticias.

- Consulta **World Models** para ver un ejemplo del uso de una experiencia puramente ficticia para entrenar al agente, lo que se denomina *entrenamiento en el sueño*.

Incorporación de ciclos de planificación en políticas

Otro enfoque integra el procedimiento de planificación directamente en una política como una subrutina, de modo que los planes completos se conviertan en información secundaria de la política, mientras se entrena el resultado de la política con cualquier algoritmo estándar sin modelo.

El concepto clave es que en este framework, la política puede aprender a elegir cómo y cuándo utilizar los planes.

Esto hace que el sesgo del modelo sea un problema menor, porque si el modelo es malo para la planificación en algunos estados, la política puede simplemente aprender a ignorarlo.

- Consulta **I2A** para ver un ejemplo de agentes dotados de este estilo de imaginación.

Optimización de políticas

Derivando las políticas de gradientes

Consideramos el caso de una política estocástica, parametrizada, π_θ. Nuestro objetivo es maximizar el retorno esperado $J(\pi_\theta) = E_{\tau\sim\pi_\theta}[R(\tau)]$.

Queremos optimizar la política mediante el ascenso de gradiente, por ejemplo:

$$\theta_{k+1} = \theta_k + \alpha\nabla_\theta J(\pi_\theta)|_{\theta_k}$$

El gradiente del rendimiento de la política, $\nabla_\theta J(\pi_\theta)$, se denomina **política de gradiente** y los algoritmos que optimizan la política de esta manera se denominan **algoritmos de políticas de gradientes**.

Comenzaremos por presentar algunos hechos que son útiles para derivar el gradiente analítico.

- **Probabilidad de una trayectoria**. La probabilidad de una trayectoria $\tau = (s_0, a_0, \ldots, s_{T+1})$ dado que las acciones provienen de π_θ es:

$$P(\tau|\theta) = \rho_0(s_0) \prod_{t=0}^{T} P(s_{t+1}|s_t, a_t)\pi_\theta(a_t|s_t).$$

- **El truco de la derivada logarítmica**. El truco de la derivada logarítmica se basa en una regla simple del cálculo: la derivada de $\log x$ con respecto a x es $1/x$. Cuando se reorganiza y se combina con la regla de la cadena, obtenemos:

$$\nabla_\theta P(\tau|\theta) = P(\tau|\theta)\nabla_\theta \log P(\tau|\theta).$$

- **Logaritmo de probabilidad de una trayectoria**. El log-prob de una trayectoria es solo:

$$\log P(\tau|\theta) = \log \rho_0(s_0) + \sum_{t=0}^{T} \Big(\log P(s_{t+1}|s_t, a_t) + \log \pi_\theta(a_t|s_t) \Big).$$

- **Gradientes de funciones de entorno**. El entorno no depende de θ, por lo que los gradientes de $\rho_0(s_0)$, $P(s_{t+1}|s_t, a_t)$ y $R(\tau)$ son cero.

- **Gradiente del log-prob de una trayectoria**. El gradiente del log-prob de una trayectoria es por lo tanto:

$$\nabla_\theta \log P(\tau|\theta) = \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t).$$

Poniendo todo junto, obtenemos el siguiente resultado:

$$
\begin{aligned}
\nabla_\theta J(\pi_\theta) &= \nabla_\theta E_{\tau\sim\pi_\theta} R(\tau) \\
&= \nabla_\theta \int_\tau P(\tau|\theta) R(\tau) \\
&= \int_\tau \nabla_\theta P(\tau|\theta) R(\tau) \\
&= \int_\tau P(\tau|\theta) \nabla_\theta \log P(\tau|\theta) R(\tau) \\
&= E_{\tau\sim\pi_\theta} \nabla_\theta \log P(\tau|\theta) R(\tau) \\
\therefore \nabla_\theta J(\pi_\theta) &= E_{\tau\sim\pi_\theta} \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) R(\tau) \right]
\end{aligned}
$$

Esta es una expectativa, lo que significa que podemos estimarla con una media muestral.

Si recopilamos un conjunto de trayectorias $\mathcal{D} = \{\tau_i\}_{i=1,\ldots,N}$ donde cada trayectoria se obtiene dejando que el agente actúe en el entorno usando la política π_θ, la política de gradiente se puede estimar con:

$$\hat{g} = \frac{1}{|\mathcal{D}|} \sum_{\tau \in \mathcal{D}} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) R(\tau)$$

donde $|\mathcal{D}|$ es el número de trayectorias en $\mathcal{D}$ (aquí N).

Esta última expresión es la versión más simple de la expresión computable que deseamos.

Suponiendo que hemos representado nuestra política de una manera que nos permite calcular $\nabla_\theta \log \pi_\theta(a|s)$ y si podemos ejecutar la política en el entorno para recopilar el conjunto de datos de trayectoria, podemos calcular la política de gradientes y dar un paso de actualización.

Lema EGLP

Supongamos que P_θ es una distribución de probabilidad parametrizada sobre una variable aleatoria, x. Luego:

$$E_{x\sim P_\theta}[\nabla_\theta \log P_\theta(x)] = 0.$$

Realiza una prueba.

Sea nuestra expresión más reciente para la política de gradiente:

$$\nabla_\theta J(\pi_\theta) = E_{\tau \sim \pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) R(\tau)\right]$$

Los agentes solo deberían reforzar las acciones en función de sus consecuencias.

Las recompensas obtenidas antes de realizar una acción no influyen en lo buena que fue esa acción: solo las recompensas que vienen después.

Resulta que esta intuición se muestra en las matemáticas y podemos mostrar que la política de gradiente también se puede expresar mediante:

$$\nabla_\theta J(\pi_\theta) = E_{\tau \sim \pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \sum_{t'=t}^{T} R(s_{t'}, a_{t'}, s_{t'+1})\right]$$

De esta forma, las acciones solo se refuerzan en función de las recompensas obtenidas una vez realizadas.

Llamaremos a esta forma el **política de gradiente de recompensa para llevar**, porque la suma de las recompensas después de un punto en una trayectoria.

$$\hat{R}_t \doteq \sum_{t'=t}^{T} R(s_{t'}, a_{t'}, s_{t'+1}),$$

se denomina **recompensa para llevar** desde ese punto y esta expresión de política de gradientes depende de la recompensa para llevar de los pares estado-acción.

Línea de base de políticas de gradiente

Una consecuencia inmediata del lema EGLP es que para cualquier función b que solo depende del estado,

$$E_{a_t \sim \pi_\theta}[\nabla_\theta \log \pi_\theta(a_t|s_t) b(s_t)] = 0.$$

Esto nos permite sumar o restar cualquier número de términos como este en nuestra expresión para la política de gradiente, sin cambiar la expectativa:

$$\nabla_\theta J(\pi_\theta) = E_{\tau \sim \pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \left(\sum_{t'=t}^{T} R(s_{t'}, a_{t'}, s_{t'+1}) - b(s_t)\right)\right].$$

Cualquier función b utilizada de esta manera se denomina línea base.

La opción más común de línea de base es la función de valor de la política $V^\pi(s_t)$.

Empíricamente, la opción $b(s_t) = V^\pi(s_t)$ tiene el efecto deseable de reducir la varianza en la estimación de la muestra para la política de gradiente. Esto da como resultado un aprendizaje de políticas más rápido y estable.

En la práctica, $V^{\pi}(s_t)$ no se puede calcular exactamente, por lo que debe ser aproximado. Esto generalmente se hace con una red neuronal, $V_{\phi}(s_t)$, que se actualiza al mismo tiempo que la política.

El método más simple para aprender V_{ϕ}, utilizado en la mayoría de las implementaciones de algoritmos de optimización de políticas (incluidos VPG, TRPO, PPO y A2C) es minimizar un objetivo de error cuadrático medio:

$$\phi_k = \arg\min_{\phi} E_{s_t, \hat{R}_t \sim \pi_k} \left[\left(V_{\phi}(s_t) - \hat{R}_t \right)^2 \right]$$

donde π_k es la política en la época k.

Esto se hace con uno o más pasos de descenso de gradiente, a partir de los parámetros de valor anterior ϕ_{k-1}.

Otras formas de la políticas de gradiente

Lo que hemos visto hasta ahora es que el política de gradiente tiene la forma general:

$$\nabla_\theta J(\pi_\theta) = E_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)\Phi_t\right]$$

donde Φ_t podría ser cualquiera de:

$$\Phi_t = R(\tau)$$

$$\Phi_t = \sum_{t'=t}^{T} R(s_{t'}, a_{t'}, s_{t'+1})$$

$$\Phi_t = \sum_{t'=t}^{T} R(s_{t'}, a_{t'}, s_{t'+1}) - b(s_t).$$

Todas estas opciones conducen al mismo valor esperado para la política de gradiente, a pesar de tener diferentes variaciones.

Resulta que hay dos elecciones válidas más de pesos Φ_t, que es importante conocer.

Función de acción-valor en la política.

La elección:

$$\Phi_t = Q^{\pi_\theta}(s_t, a_t)$$

también es válido.

Función de ventaja

La ventaja de una acción, definida por $A^{\pi}(s_t, a_t) = Q^{\pi}(s_t, a_t) - V^{\pi}(s_t)$, describe cuánto mejor o peor es esta acción respecto a otras acciones en promedio (en relación con la política actual).

Esta elección:

$$\Phi_t = A^{\pi_\theta}(s_t, a_t)$$

también es válido.

La prueba es que es equivalente a usar $\Phi_t = Q^{\pi_\theta}(s_t, a_t)$ y luego usar una línea de base de función de valor, lo cual siempre tenemos la libertad de hacer.

La formulación de políticas de gradiente con funciones de ventaja es extremadamente común y hay muchas formas diferentes de estimar la función de ventaja utilizada por diferentes algoritmos.

Fin!