Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Física

Meta aprendizaje de la señal de entrenamiento para entornos con refuerzos dispersos

Tesis de Licenciatura en Ciencias Físicas

Octavio Pappalardo

Julio 2024

Director: Juan Miguel Santos
Codirector: Rodrigo Ramele

Índice general

Resumen

El aprendizaje por refuerzo ha encontrado muchos éxitos en la última década. Sin embargo, aún quedan muchos problemas por resolver para que pueda ser aplicado en más dominios. Algunas de las principales líneas donde se busca mejorarlo son las que corresponden a su pobre eficiencia de datos, su capacidad de generalización y su debilidad en aprender con problemas donde los refuerzos son dispersos; para los cuales generalmente se requiere el diseño humano de un refuerzo denso. El meta aprendizaje ha surgido como una posibilidad para ayudar a lidiar con estas cuestiones al aprender partes del algoritmo de RL para que cumpla con características deseadas. Por otro lado, el uso de refuerzos intrínsecos es ampliamente estudiado en la búsqueda de mejorar las propiedades de exploración de los algoritmos. En este trabajo se estudia como el meta aprendizaje puede mejorar la calidad de la señal de entrenamiento recibida por agentes de RL. En particular, el foco principal está puesto en el meta aprendizaje de refuerzos intrínsecos sin el uso de meta gradientes. Este enfoque es comparado con métodos de RL estándar, con métodos establecidos de meta RL y con el meta aprendizaje de una función de ventaja. Se analizan las ventajas y desventajas que presentan cada uno de ellos. Los algoritmos desarrollados fueron evaluados en distribuciones de tareas de control continuo con variaciones paramétricas y no paramétricas donde en las tareas de evaluación solo hay acceso a recompensas dispersas.

Agradecimientos

Agradezco a los profesores de la Facultad de Ciencias Exactas y Naturales de la Universidad de Buenos Aires por mi formación en la institución. Agradezco a Juan Miguel Santos y a Rodrigo Ramele por supervisar el desarrollo de esta Tesis.

1. Introducción

El campo del aprendizaje automático ha avanzado significativamente en la última década. Un área dentro de este campo es la de aprendizaje por refuerzo (RL) donde un agente debe aprender interactuando con un entorno a través de la toma secuencial de acciones. La adopción de redes neuronales y avances algorítmicos dentro de los métodos de aprendizaje por refuerzo ha permitido la aplicación de ellos a problemas complejos de toma de decisiones. Algunos ejemplos exitosos de esto son [1] que supera el desempeño humano en juegos como Go, ajedrez y *shogi*, [2, 3] que encuentran algoritmos más rápidos para tareas computacionales, [4] que usa RL para el control magnético de plasmas en *tokamaks* y [5] que demostró con éxito el uso de RL para control de sistemas de enfriamiento comercial. Sin embargo, hay muchos retos que todavía deben superarse para que los métodos de aprendizaje por refuerzo puedan ser adoptados más extensivamente. Sobre todo si se busca utilizarlo en problemas donde debe interactuar con el mundo real.

A partir de estos retos han surgido varias sub ramas de investigación: RL basado en modelos [6], RL jerárquico [7], RL fuera de linea [8], RL no supervisado [9, 10], RL multitarea [11] y meta RL [12]. Dos cuestiones principales que abordan son la pobre eficiencia de datos al aprender una tarea, y la limitada capacidad de generalización de las políticas aprendidas al enfrentarse a problemas nuevos.

Otro desafío central al aprendizaje por refuerzo es el balance de exploración y explotación [13]. En RL la distribución de datos es no estática y depende de las decisiones que toma el agente. Por esta razón es importante que el agente encuentre un balance entre *explotar* la información obtenida para ejecutar las acciones que más refuerzo espera recibir, y *explorar* para encontrar secuencias de acciones aún mejores. Este problema es exacerbado en tareas donde los refuerzos son dispersos - donde las señales que guían al agente hacia soluciones útiles son escasas. Una de las principales líneas de investigación que busca lidiar con esto es la del uso de refuerzos intrínsecos [14]. Estos se pueden pensar como un mecanismo de curiosidad incorporado al agente que ayudan a guiarlo hacia un aprendizaje útil (sección 2.3).

En este trabajo nos centramos en técnicas de meta aprendizaje por refuerzo (sección 2.2). En estas se busca aprender partes del algoritmo de aprendizaje a través de la interacción con un conjunto de tareas para que luego pueda ser aplicado a otras con mayor eficiencia. Principalmente estudiamos el modelado de una meta red neuronal que utiliza RL para aprender a proveer señales de aprendizaje a políticas de control. Se estudia cómo esto puede ser favorable con respecto a métodos estándar de aprendizaje por refuerzo y sus diferencias con otros métodos de meta aprendizaje. En particular se analiza las ventajas y desventajas de aprender refuerzos intrínsecos contra otras cantidades y del uso

u omisión de utilizar meta gradientes. Por *cantidades* se hace referencia a todo lo que pueda ser meta aprendido. Se emplea este término genérico debido a la vasta diversidad de entidades que pueden ser objeto de meta-aprendizaje; este punto se trata con mayor detalle en la sección 2.2.

Se llevan a cabo experimentos en un contexto donde los algoritmos tienen acceso a refuerzos moldeados para las tareas de entrenamiento pero solo a refuerzos dispersos para las tareas de evaluación. Los refuerzos moldeados (*shaped rewards*) son refuerzos frecuentes; utilizamos el termino moldeados para destacar que la obtención de estos refuerzos requiere ingeniería humana. Se trabaja con las distribuciones de problemas de control continuo introducidas en Meta World [15] en las cuales a un brazo robótico se le presentan un conjunto de tareas que presentan tanto variaciones paramétricas como no paramétricas (seccion 3.2.1). El código del trabajo esta disponible públicamente[1].

Este manuscrito está organizado de la siguiente manera:

- **Sección 2 (Preliminares) :** Se presenta una introducción a los problemas de aprendizaje por refuerzo y a los métodos basados en políticas. Luego se describe el meta aprendizaje por refuerzo dando una definición, discutiendo las familias principales de métodos, y la bibliografía correspondiente. Finalmente se discute el uso de refuerzos intrínsecos en aprendizaje por refuerzo.
- **Sección 3 (Propuesta y metodología) :** En primer lugar se introducen los métodos que son el foco del trabajo y se discuten trabajos relacionados. Tras esto se presentan los *benchmarks* utilizados, la metodología de evaluación, implementaciones de los algoritmos y otros detalles experimentales del trabajo.
- **Sección 4 (Resultados) :** Se exponen los principales resultados experimentales obtenidos junto a las preguntas que los motivan y sus discusiones.
- **Sección 5 (Conclusiones) :** Se exponen las principales reflexiones y conclusiones a partir de los resultados, y se delinean posibles trabajos futuros.

[1] Código del trabajo

2. Preliminares

Esta sección aborda los fundamentos de las áreas de investigación sobre las que se apoya esta tesis. Su objetivo es dar información necesaria para una mejor comprensión del resto del trabajo, incluyendo su motivación y su posición dentro del actual panorama de investigación. Con una primera aclaración sobre notación, se menciona que en el manuscrito se utiliza tanto $\mathbb{E}_{p(x)}$ como $\mathbb{E}_{x\sim p(x)}$ para la operación de valor esperado sobre la cantidad aleatoria x bajo la distribución $p(x)$.

2.1. Aprendizaje por refuerzo

El aprendizaje por refuerzo(RL) es un paradigma de aprendizaje en el cual un agente debe aprender a comportarse a través de interacciones secuenciales con un entorno. Un problema estándar de aprendizaje por refuerzo está definido por un MDP (Markov decision process) [13]. Éste se puede describir como una tupla $\langle \mathcal{S}, \mathcal{A}, p_T, R, p_0, \gamma \rangle$ donde:

- $\mathcal{S}$ es el conjunto de estados que el agente puede visitar ,
- $\mathcal{A}$ es el conjunto de acciones que puede tomar ,
- $p_T = p_T\left(s_{t+1} \mid s_t, a_t\right)$ es el modelo de transición. $s_t \in \mathcal{S}$ con t indicando el paso temporal, $t \in \{0, 1, 2, \cdots\}$.
- $p_R = p_R\left(r_{t+1} \mid s_t, a_t, s_{t+1}\right)$ es el modelo de refuerzos donde r_{t+1} es el refuerzo cuando el agente realiza una transición de s_t a s_{t+1} dado que llevó a cabo la acción a_t. En vez de utilizar p_R, es común definir un MDP utizando una función de refuerzos $R(s_t, a_t)$ que indica el refuerzo inmediato promedio que se espera recibir del entorno al ejecutar la acción a_t en el estado s_t. Notar que p_R es mas general y se puede utilizar para obtener la función de refuerzos como $R(s, a) = \mathbb{E}_{p_T(s'|s,a)}\left[\mathbb{E}_{p_R(r|s,a,s')}[r]\right]$.
- p_0 es la distribución que determina el estado inicial.
- $\gamma \in [0, 1]$ es un factor de descuento utilizado para controlar el peso de refuerzos futuros .

En general el agente desconoce todos estos parámetros del problema inicialmente (excepto por γ) .

El agente debe interactuar con el entorno para aprender una política $\pi(a \mid s)$. π es una distribución que determina la probabilidad de tomar cada acción a en cada estado s. El aprendizaje es guiado a través de los refuerzos que el agente recibe del entorno al pasar de un estado al siguiente (figura 2.1).

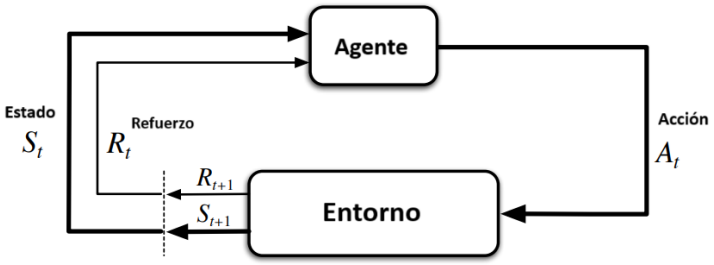

Figura 2.1: Esquema de la interacción de un agente con el entorno en un MDP. A cada paso el agente ejecuta una acción y el entorno da un nuevo estado y un refuerzo en función de esta. Figura adaptada de [13].

La tarea puede ser continua , en cuyo caso el agente interactúa con el entorno potencialmente para siempre, o episódica , en cuyo caso la interacción está separada en episodios. Cada episodio termina cuando el agente llega a un estado terminal tras lo que comienza un nuevo episodio desde un estado inicial $s_0 \sim p_0$. A la trayectoria que toma el agente a lo largo de un episodio se la denota como $\tau = (s_0, a_0, r_1, \ldots, s_T)$ con s_T siendo el estado terminal. Teniendo en cuenta que tanto el modelo de transición como el modelo de refuerzos son markovianos en un MDP, la probabilidad de generar la trayectoria τ bajo la política π se puede factorizar como $p(\boldsymbol{\tau}) = p_0(s_0) \prod_{t=0}^{T-1} \pi(a_t \mid s_t)\, p_T(s_{t+1} \mid s_t, a_t)\, p_R(r_{t+1} \mid s_t, a_t, s_{t+1})$

Si la tarea es episódica, el objetivo del agente es encontrar una política que maximice el valor esperado del retorno. El retorno G_t es la suma descontada de los refuerzos que el agente recibe a futuro partiendo desde el estado presente y siguiendo la política. O sea :

$$G_t = \sum_{k=t+1}^{T} \gamma^{k-t-1} r_k = r_{t+1} + \gamma G_{t+1}. \tag{2.1}$$

Los algoritmos de aprendizaje por refuerzo buscan maximizar la esperanza del retorno a lo largo de una trayectoria:

$$J(\pi) = \mathbb{E}_{\tau \sim p(\tau)} \left[\sum_{t=1}^{T} \gamma^{t-1} r_t \right] = \mathbb{E}_{\tau \sim p(\tau)} \left[G_0 \right] = \mathbb{E}_{\tau \sim p(\tau)} \left[G(\tau) \right]. \tag{2.2}$$

Métodos basados en políticas

Los métodos basados en políticas optimizan los parámetros de una política parametrizada π_θ para maximizar el objetivo 2.2. En particular los métodos de gradiente de política llevan a cabo esta optimización utilizando estimaciones del gradiente de este objetivo. Del teorema de gradiente de política [16] se pueden derivar muchas expresiones para el gradiente. Considerando por simplicidad $\gamma = 1$, estas toman la forma:

$$\nabla_\theta J(\pi_\theta) = \mathop{\mathrm{E}}_{\tau \sim p(\tau)} \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t)\, \Phi_t \right], \tag{2.3}$$

donde Φ_t puede ser una de muchas opciones [17] . Todas las opciones siguientes dan lugar al mismo valor esperado pero tienen distintas varianzas:

- $\Phi_t = \sum_{t'=1}^{T} r_{t'}$, suma de los refuerzos en la trayectoria.
- $\Phi_t = \sum_{t'=t+1}^{T} r_{t'}$, suma de refuerzos tras acción a_t
- $\Phi_t = \sum_{t'=t+1}^{T} r_{t'} - b(s_t)$, donde $b(s_t)$ es una función arbitraria que solo depende del estado.
- $\Phi_t = Q_{\pi_\theta}(s_t, a_t)$, donde $Q_\pi(s,a) = \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a]$ es la función de valor de acción.
- $\Phi_t = A_{\pi_\theta}(s_t, a_t)$, donde $A_\pi(s,a) = Q_\pi(s,a) - V_\pi(s)$ es la función de ventaja.

En particular los métodos actor-crítico utilizan $\Phi_t = A_{\pi_\theta}(s_t, a_t)$ y la estiman teniendo una segunda red neuronal $V_\psi(s)$ de parámetros ψ que estima la función de valor del estado $V_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s]$. Intuitivamente , Φ_t da una medida de cuánto mover los parámetros de la política para favorecer más la acción a_t en el estado s_t.

En la práctica, para estimar el valor esperado en la ecuación 2.3 se suele definir una función de error $\mathcal{L}$ diferenciable tal que su gradiente coincide (a signo opuesto) con el estimador Monte Carlo de 2.3. De esta manera el gradiente puede ser estimado haciendo uso del motor de auto-diferenciación de librerías de aprendizaje profundo [18]. Tras recolectar un conjunto de datos $\mathcal{D}^i$ interactuando con el MDP, un método VPG(*'Vanilla Policy Gradient'*) lleva entonces a cabo la actualización de los parámetros con un paso de gradiente para minimizar esta función de error. En el caso simple donde se usa SGD(*Stochastic Gradient Descent* éste toma la forma $\theta_{i+1} = \theta_i - \alpha \nabla_{\theta_i} \mathcal{L}(\theta_i, \mathcal{D}^i)$.

Hay muchas variantes de métodos de optimización de política. En particular dos métodos populares que se utilizan en esta tesis son TRPO (*Trust Region Policy Optimization*) [19] y PPO (*Proximal Policy Optimization*) [20]. Estos métodos llevan a cabo un proceso de optimización local para cada conjunto de datos colectados haciendo uso de información del gradiente. Esto les permite tener mejor eficiencia de datos y lograr un entrenamiento más estable comparado a VPG.

2.2. Meta aprendizaje por refuerzo

Para aprender una cierta tarea los algoritmos de aprendizaje automático utilizan un conjunto de datos que le proveen información sobre esa tarea. En el caso de aprendizaje por refuerzo estos datos son interacciones con el MDP que corresponde a esa tarea . Vamos a denotar a estos datos como $\mathcal{D}$. Un algoritmo de RL entonces se puede ver como un mapeo de datos $\mathcal{D}$ a una política. Si consideramos políticas parametrizadas π_θ el algoritmo de aprendizaje se puede expresar como una función $\theta = f(\mathcal{D})$.

A su vez f, el algoritmo de aprendizaje, puede tener sus propios parámetros ϕ. Teniéndolos en cuenta podemos expresar esta dependencia como $\theta = f(\mathcal{D}; \phi) = f_\phi(\mathcal{D})$. Es importante notar que mientras que los parámetros θ de la política π suelen hacer referencia a valores numéricos como parámetros en un modelo lineal o una red neuronal, los parámetros ϕ de la función de aprendizaje f pueden hacer referencia a una gran variedad de objetos ; por ejemplo a la arquitectura de la red, al optimizador utilizado, a la función de error utilizada, etc [21] .

Los métodos de meta aprendizaje buscan justamente aprender partes del algoritmo de aprendizaje. Bajo la notación anterior vamos a decir entonces que un algoritmo de

meta aprendizaje M busca aprender un cierto conjunto de parámetros ϕ que influencian el comportamiento de un algoritmo de aprendizaje f_ϕ . Al aprendizaje que hace M lo vamos a llamar como proceso o aprendizaje externo y al que hace f como interno.

En el contexto más común para llevar a cabo el meta aprendizaje, M requiere datos de un conjunto de distintas tareas $\mathcal{D}_{1:J}$. Podemos entonces expresar el proceso de meta aprendizaje como $\phi = M\left(\mathcal{D}_{1:J}\right)$. El objetivo final de esto es que una vez terminado el proceso de meta aprendizaje, al encontrarse con una nueva tarea $j+1$, se puede aprender a mejorar el desempeño en ella aplicando $f(\mathcal{D}_{J+1}; \phi)$. Lo importante es que ahora este proceso interno es más eficiente si se logró encontrar un ϕ adecuado en el meta aprendizaje.

Las principales desventajas del meta aprendizaje son que, mientras que una vez obtenido ϕ el aprendizaje de una tarea puede ser mucho más eficiente, el proceso de meta aprendizaje en sí para obtener ϕ no lo es. En segundo lugar , como ϕ se aprende a partir de datos de un conjunto de tareas $1 : J$, puede sufrir problemas de generalización al aplicarse a la nueva tarea $J + 1$. En general el meta aprendizaje funciona mejor cuanta más estructura en común tienen las tareas a las que se aplica f_ϕ con las tareas que se utilizaron para aprender ϕ.

Para formalizar el objetivo que busca maximizar el meta aprendizaje vamos a denotar a un dado MDP como $\mathcal{M}^i$ y a la distribución de tareas que utiliza para entrenar como $p(\mathcal{M})$; entonces el objetivo más estándar que suelen utilizar los algoritmos de meta aprendizaje se puede expresar como [12]:

$$\mathcal{J}(\phi) = \mathbb{E}_{\mathcal{M}^i \sim p(\mathcal{M})} \left[\mathbb{E}_{\mathcal{D}} \left[\sum_{\tau \in \mathcal{D}} G(\tau) \mid f_\phi, \mathcal{M}^i \right] \right]. \tag{2.4}$$

El objetivo en la ecuación 2.4 puede tener algunas variaciones dependiendo de lo que se busca. Por ejemplo la sumatoria sobre los retornos de todos los episodios en los datos de un MDP se podría restringir a solo ciertos episodios donde se busca buen desempeño (comúnmente los últimos). También se podría considerar un factor de descuento u otra manera de pesar ya sea los episodios o cada paso tomado en el entorno, etc.

Idealmente, se quiere que las tareas sobre las cuales aplica f_ϕ, una vez terminado el meta aprendizaje, provengan de la misma distribución $p(\mathcal{M})$; similar a como ocurre con los datos de entrenamiento y evaluación en aprendizaje supervisado.

A continuación describimos dos familias de métodos de meta aprendizaje por refuerzo. Por más que sus ideas son presentadas independientemente hay métodos que combinan elementos de ambas.

2.2.1. Métodos de contexto largo

Una manera de poder optimizar el aprendizaje de una tarea para que sea más eficiente es directamente considerar una red neuronal que utiliza todos los datos de la tarea que fueron recolectados hasta el paso t para influenciar la acción que se toma en ese paso. Vamos a referirnos a ésta como red recurrente pero puede ser cualquier tipo de red neuronal que utiliza un contexto largo. Al utilizar todos los datos de la interacción con la tarea (las acciones tomadas, los estados visitados, los refuerzos recibidos, etc) durante el meta aprendizaje, la red neuronal puede aprender a usar estos más eficientemente y a reconocer a partir de ellos qué tarea debe resolver y cómo. En esta familia de métodos se puede pensar como que parte del aprendizaje interno - aprendizaje de una dada tarea -

ocurre en las activaciones de la red neuronal a medida que va recibiendo más datos de la interacción.

En la mayoría de los casos se entrena la red recurrente directamente para que su salida sea la distribución de probabilidad sobre qué acción tomar. Esta idea fue por primera vez presentada por [22] y por [23]. Por brevedad vamos a referirnos a este método como RL^2 y lo explicamos a continuación.

RL^2

Considere una red neuronal recurrente con parámetros ϕ que a cada paso toma como valor de entrada $\{s_t, r_t, a_{t-1}, d_t\}$ tal que para cada paso la política computa una distribución $\pi_\phi\left(a_t \mid s_t, r_t, a_{t-1}, d_t, h_{t-1}\right)$ donde $d_t \in \{0, 1\}$ indica si el paso t es el comienzo de un nuevo episodio y h_{t-1} es el estado oculto de la red recurrente que codifica la información de todos los pasos pasados. Esta es la base de RL^2 (fig 2.2).

La red neuronal es entrenada con algún método estándar de aprendizaje por refuerzo sobre una distribución de tareas para maximizar el valor esperado de la suma descontada de refuerzos que obtiene en su interacción con ellas (notar que este objetivo incluye refuerzos a través de muchos episodios mientras que el objetivo estándar de RL, Ec. 2.2, es el retorno esperado de un único episodio) . Bajo la notación anterior se puede considerar que los parámetros θ del aprendizaje interno son las activaciones efímeras de la red recurrente al recibir datos de una tarea específica a cada paso.

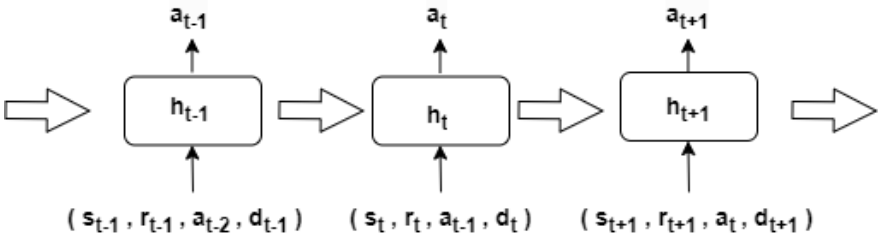

Figura 2.2: Esquema conceptual del método RL^2. Red recurrente que a cada paso utiliza toda la información pasada de su interacción con el MDP para computar una nueva acción.

2.2.2. Métodos de meta gradiente

En los métodos de meta gradiente [24] el proceso interno $f_\phi(\mathcal{D})$ es un algoritmo de aprendizaje por refuerzo. Estos suelen consistir en una serie de actualizaciones a los parámetros de la política , $\theta_{i+1} = g(\mathcal{D}^i, \theta_i, \eta)$ donde $\mathcal{D}^i$ es el subconjunto de interacciones con el entorno utilizadas para la actualización a θ_i. Los algoritmos de meta gradiente toman en cuenta esta estructura interna e intentan aprender algunos de los parámetros de las actualizaciones internas; estos son los parámetros que denotamos como ϕ. Cuando la actualización interna toma la forma de un paso de gradiente, el gradiente para actualizar ϕ en el aprendizaje externo es un gradiente de segundo orden - un meta gradiente.

El método más conocido en esta familia es MAML (*Model Agnostic Meta Learning*) [25]. Lo describimos a continuación.

MAML

MAML meta aprende el valor inicial de la política π_θ, o sea $\phi = \theta_0$. En el caso más simple esta política es actualizada una sola vez al enfrentarse a una nueva tarea y se utiliza VPG con SGD en el aprendizaje interno para hacerlo. Bajo esta situación los parámetros adaptados se pueden expresar como $\theta_1 = \theta_0 - \alpha \nabla_\theta \mathcal{L}\left(\theta_0, \mathcal{D}^{\text{tr}}\right)$ donde $\mathcal{D}^{\text{tr}}$ es el conjunto de datos que se recolectó en la nueva tarea para estimar $\nabla_\theta \mathcal{L}(\theta_0) = -\nabla_\theta J\left(\theta_0\right)$ (ecuación 2.3). MAML busca maximizar el desempeño de los parámetros adaptados θ_1. Éste se mide con un nuevo conjunto de interacciones recolectadas $\mathcal{D}^{\text{ts}}$. El proceso de meta aprendizaje que lleva a cabo MAML puede entonces escribirse como

$$\min_{\theta_0} \sum_{\text{tarea } i} \mathcal{L}\left(\theta_0 - \alpha \nabla_\theta \mathcal{L}\left(\theta_0, \mathcal{D}_i^{\text{tr}}\right), \mathcal{D}_i^{\text{ts}}\right). \tag{2.5}$$

Una variante a primer orden de MAML, también presentada en [25] es FOMAML (*first order MAML*) que trata al jacobiano de los distintos θ_1 con respecto a θ_0 como la identidad. La figura 2.3 muestra esquemáticamente el funcionamiento del método.

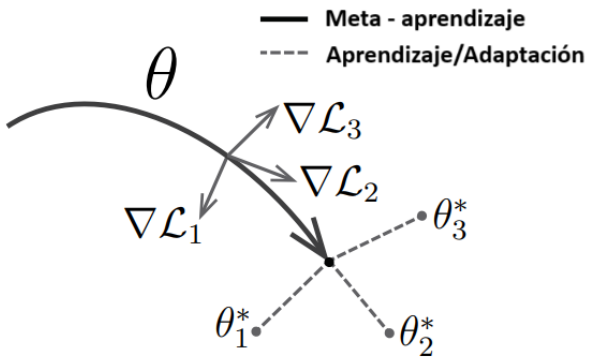

Figura 2.3: Esquema conceptual del método MAML. Éste meta aprende los parámetros iniciales θ de una política que son entrenados tal que al adaptarse a una tarea específica $(1, 2, 3$ en la figura) obtienen buen desempeño en ellas. θ_i^* son los parámetros óptimos en la tarea i. Figura adaptada de [25].

Otros métodos de meta gradiente y aprendizaje de refuerzos intrínsecos

Mientras que MAML busca meta aprender los parámetros iniciales de la política , se ha explorado meta aprender muchas otras componentes de las actualizaciones. La única restricción es que el parámetro meta aprendido debe afectar de manera diferenciable al valor de $\theta_{i+1} \forall i$. Algunas otras opciones que se estudiaron son : meta aprender un subconjunto de los parámetros de la política [26], un distribución sobre políticas iniciales [27], la tasa de aprendizaje [28], matrices de precondicionamiento [29] y componentes de

la función de pérdida $\mathcal{L}$ (Ver [12] para una lista más completa de referencias). Para los propósitos de esta tesis nos vamos a concentrar en esta última categoría. Dentro de ella se puede intentar meta aprender toda la función de pérdida o mantener cierta estructura en ella y aprender solo componentes [30, 31].

Este trabajo tiene un foco importante en el meta aprendizaje de refuerzos intrínsecos. Varios trabajos estudian el aprendizaje de una función de refuerzos usando meta gradientes: [32, 33] lo hacen en el contexto de una tarea única y [34] lo hace en el contexto de una distribución de tareas. Más allá de los métodos de meta gradiente, [35] exploró meta aprender una función de refuerzos con una búsqueda discreta sobre un espacio de programas. Los trabajos mencionados hasta ahora aprenden una función de refuerzos que reemplaza completamente a los refuerzos extrínsecos, por otro lado [36] busca meta aprender a moldear los refuerzos extrínsecos para obtener señales más densas.

2.3. Refuerzos intrínsecos

Como fue discutido en la sección 2.1 , el objetivo de un agente de RL es maximizar el valor esperado de la suma descontada de refuerzos que obtiene. Un problema de muchos algoritmos al intentar directamente maximizar los refuerzos extrínsecos que definen al problema es que esto los puede conducir a llevar a cabo una mala exploración del espacio de estados ; especialmente cuando los refuerzos son dispersos [14]. Una de las ideas principales que se estudia y que se utiliza para mejorar la calidad de la exploración y para obtener un mejor balance entre exploración y explotación es el uso de refuerzos intrínsecos. De manera general se puede considerar refuerzos intrínsecos a cualquier término de refuerzo que complementa o que reemplaza a los refuerzos extrínsecos que definen al problema (aunque algunos trabajos llaman intrínsecos solo a términos aditivos). Es común asociar estos a agregarle 'curiosidad' a los agentes y es común también utilizarlos para pre-entrenar políticas en contextos donde pueden interactuar con el entorno pero no tienen acceso a tareas (sin acceso a refuerzos extrínsecos).

Existe una gran variedad de maneras de diseñar refuerzos intrínsecos [14] .La gran mayoría basado en heurísticas. Algunas de ellas son guiadas por la búsqueda de reducir la incertidumbre que el agente tiene sobre cierta componente de su entorno, otras buscan incentivar la visita de estados nuevos o diferentes de los ya visitados. En este trabajo se estudia un camino distinto, donde se busca *aprender* esta señal de motivación intrínseca [37] .

3. Propuesta y metodología

3.1. Método principal

En la sección 2.2.2 se discutió métodos de meta aprendizaje donde el aprendizaje interno f_ϕ es un algoritmo de RL y ϕ es aprendido con meta gradientes. Hay maneras de meta aprender ϕ que evitan la necesidad de calcular gradientes de segundo orden. El trabajo [38] estudia variantes a primer orden del tipo de actualización que hacen los meta gradientes, [39] utiliza un algoritmo de evolución y [40] utiliza un método de RL basado en la función de valor (en lugar de basado en la política) para el proceso externo.

En esta tesis se explora la opción de evitar los gradientes de segundo orden entrenando ϕ con un algoritmo de aprendizaje por refuerzo pero sin modelar explícitamente la estructura del aprendizaje interno; o sea considerando el proceso de aprendizaje interno dentro de la estocasticidad del problema. El esquema resultante es análogo a aplicar RL^2 sobre un espacio distinto.

Para ser más específicos, describimos el caso donde lo que se busca es meta aprender una función de refuerzos intrínsecos. Modelamos a ésta como un agente $\pi^r_\phi\left(r^i_t \mid \mathcal{D}_{:t}\right)$, donde $\mathcal{D}_{:t}$ codifica toda la interacción con el MDP hasta el paso t . En particular, en este trabajo usamos una red LSTM (*Long Short-Term Memory*) [41] que a cada paso t recibe como entrada la tupla $\{s_t, a_t, \pi_\theta(a_t \mid s_t), r^e_t, r^i_{t-1}, \pi^r_\phi(r^i_{t-1} \mid \mathcal{D}_{:t-1}), d_t\}$ donde π_θ es la política que toma acciones en el entorno, r^e_t es el refuerzo extrínseco que entregó el entorno en el paso t , $d_t \in \{0, 1\}$ indica si el paso t es el comienzo de un nuevo episodio, r^i_{t-1} es el refuerzo intrínseco que entregó π_ϕ en el paso $t-1$ y $\pi^r_\phi(r^i_{t-1} \mid \mathcal{D}_{:t-1})$ es su densidad de probabilidad.

Mientras que los métodos de meta gradiente que aprenden una función de refuerzos intrínsecos (sección 2.2.2) la modelan como una función determinista y obtienen su gradiente usando la regla de la cadena a través de los parámetros θ del agente que toma acciones sobre el entorno, acá la función de refuerzos intrínsecos da una distribución de probabilidad sobre el refuerzo intrínseco a entregar en cada paso y su gradiente se estima respecto al objetivo 2.4 directamente. O sea, se la modela como un agente de RL donde sus acciones son el refuerzo intrínseco a entregar, los estados son los $\mathcal{D}_{:t}$, y el objetivo a maximizar es 2.4. La estocasticidad de la función de transición y de la función de refuerzos en este nuevo espacio están dadas por una combinación de la estocasticidad del MDP subyacente, del proceso de optimización interno f, de los parámetros iniciales de la política π_θ y la estocasticidad intrínseca de la política.

El trabajo que guarda mayor similitud algorítmica al trabajo de esta tesis es [34]. En éste también se estudia el meta aprendizaje de una función de refuerzos intrínsecos para una distribución de tareas con el objetivo 2.4. La principal diferencia algorítmica es que

ellos utilizan meta gradientes para el aprendizaje.

Puede haber posibles ventajas y desventajas de meta aprender los refuerzos intrínsecos (u otras cantidades) como se hace en esta tesis (con un método de caja negra) contra aprenderlos con un método de meta gradiente. La principal potencial desventaja es que como en la actualización de los parámetros los meta gradientes toman en cuenta de manera explícita el rol que toman los refuerzos intrínsecos en el aprendizaje interno, pueden proveer una señal de aprendizaje menos ruidosa. Esto es discutido en [42] en el contexto de meta aprendizaje de políticas. Por otro lado, un método de caja negra es potencialmente más simple ya que está formulado como un problema estándar de RL y no requiere el cálculo de gradientes de segundo orden a través de un proceso de optimización. Además, calcular gradientes de segundo orden es una operación computacionalmente cara comparado a gradientes de primer orden, por lo que el método de caja negra también puede ser favorable en este aspecto. Otra ventaja relacionada a lo discutido es que el método es indistinto a que el proceso interno utilice la cantidad meta aprendida (cualquiera sea esta) de manera no diferenciable para afectar la elección de acciones; mientras que los métodos de meta gradiente no pueden aplicarse en estas situaciones a menos que se les busque variantes. Que la influencia del parámetro meta aprendido (ϕ) sobre la política sea diferenciable o no, depende de la elección de proceso interno y de que parte de este es controlado por ϕ.

Mientras que la mayor parte de la discusión en esta sección siguió el ejemplo de meta aprender refuerzos intrínsecos, se podría haber desarrollado análogamente para el meta aprendizaje de cualquier otra cantidad controlable que de alguna manera afecta a las acciones finales que se toman sobre el entorno. Distintas elecciones pueden significar distintos grados de dificultad en el meta aprendizaje, distintos niveles de generalización entre tareas de las cantidades aprendidas y distintos grados de influencia sobre la política final. Esto indica que la elección correcta de que cantidad meta aprender puede ser un factor importante. Esta discusión se expande en la sección 4.3 donde a modo de ejemplo se meta aprende una función de ventaja para comparar con la función de refuerzos intrínsecos.

3.2. Detalles Experimentales

3.2.1. Entornos de evaluación - Benchmarks

Muchos de los benchmarks utilizados para evaluar métodos de meta aprendizaje contienen un conjunto de tareas donde la variación entre ellas es estrecha. Por ejemplo un conjunto de tareas común es un robot simulado donde las distintas tareas se corresponden con diferentes velocidades de objetivo [25]. En este trabajo utilizamos los benchmarks introducidos en [15]. Estos tienen la ventaja que permiten evaluar los algoritmos tanto en distribuciones estrechas como en distribuciones donde la diferencia entre las tareas es más considerable. Meta-World contiene 50 tipos de problemas de control continuo donde un brazo robótico simulado debe interactuar con un objeto para llevarlo a una configuración deseada. Cada tipo de problema tiene a su vez una infinita diversidad de variaciones paramétricas. Más concretamente una tarea en Meta-World es identificable por la tupla (*clase de problema , posición inicial del objeto de interés , posición objetivo del objeto de interés*). Variaciones en las últimas dos componentes son variaciones paramétricas del problema (del mismo tipo que las distintas velocidades objetivo del ejemplo anterior). Por otro lado, la primera componente indica una variación no paramétrica ; indica que tipo de

problema el brazo robótico debe resolver (ej. abrir una ventana, cerrar un placard). Cada episodio en todas las tareas tiene una duración de 500 pasos sin opción de terminación temprana (incluso si el objetivo es completado).

A partir de todas estas tareas Meta World presenta distintos benchmarks para evaluar los algoritmos. A continuación describimos los que fueron utilizados en este trabajo .

- **ML1:** Los benchmarks ML1 seleccionan una única clase de problema de manipulación. Para esta, contienen 50 variaciones paramétricas en la distribución de tareas de entrenamiento y otras 50 como el conjunto de tareas de evaluación. En este trabajo se utilizaron los benchmarks ML1 para las tareas 'close-door-v2' , 'reach-v2' y 'button-press-v2'.
- **ML10:** El benchmark ML10 contiene 10 clases de problemas en el conjunto de entrenamiento y 5 otros problemas en el conjunto de evaluación . Para cada problema (tanto los de entrenamiento como los de evaluación) se utilizan 50 variaciones paramétricas. Ver apéndice A.1 para detalles de los problemas que contiene.

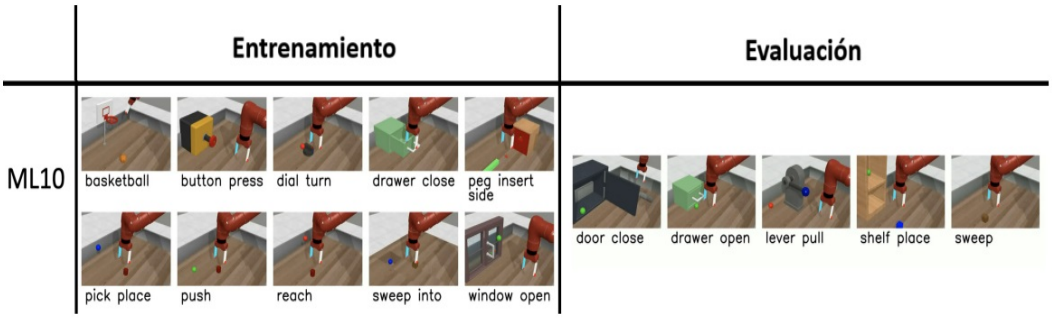

Figura 3.1: Visualización de las distintas tareas involucradas en el benchmark ML10 de Meta World. Este consiste en 10 tipos de tareas de entrenamiento y 5 de evaluación en las cuales un brazo robótico debe alcanzar un cierto objetivo. Figura adaptada de [15].

Espacio de observación y acción

El espacio de acciones del brazo robótico es el mismo para todas las tareas. Este tiene 4 dimensiones. Contiene el cambio en la posición del efector final en el que termina el brazo (3 dimensiones) seguido por el torque (normalizado) aplicado por la pinza de éste (unidimensional). Todas las acciones en este espacio varían entre -1 y 1.

El espacio de observaciones es de dimensión 39. Contiene : La posición 3D del efector final, una medida normalizada de qué tan abierta está la pinza, la posición 3D y el quaternion del primer y segundo objeto de interés, toda esta misma información para el paso anterior y finalmente la posición 3D del objetivo.

Remarcamos que en los benchmarks que utilizamos, la posición 3D del objetivo no se pone a disposición (esta siempre en 0) y notamos que para el caso de ML10 ninguna parte de la observación muestra explícitamente con que clase de problema se está tratando. Esto

fuerza al agente a aprender a reconocer y adaptarse a la tarea a partir de interacción con ella. También aclaramos que para los problemas donde hay un solo objeto de interés, la información que hace referencia al segundo objeto está siempre en 0.

Disponibilidad de refuerzos

Una gran proporción de trabajos en RL lidia con entornos donde el agente obtiene señales de refuerzo frecuentemente. Por otro lado, el desempeño de un agente en una tarea suele ser más fácil de expresar a partir de refuerzos dispersos; por ejemplo un refuerzo positivo al completar un objetivo. La dificultad del aprendizaje incrementa considerablemente a falta de refuerzos frecuentes. Esto lleva a que en muchas ocasiones se busque diseñar refuerzos manualmente para las tareas. Esto último puede resultar muy trabajoso y llevar a problemas inesperados donde el agente aprende políticas que logran obtener muchos de estos refuerzos sin completar el objetivo que realmente interesa [43] (*reward hacking*). El problema de aprendizaje con refuerzos dispersos también llevó a muchas líneas de investigación donde se busca que el agente pueda explorar el entorno de manera más eficiente; entre ellas el uso de refuerzos intrínsecos (sección 2.3) .

Meta World presenta tanto refuerzos dispersos como refuerzos moldeados para sus tareas. En este trabajo se decidió darle acceso al agente a los refuerzos moldeados para las tareas de entrenamiento pero solo acceso a los dispersos a la hora de enfrentarse a las tareas de evaluación. Los esfuerzos dispersos consisten en una señal de éxito cuando el agente logra el objetivo de la tarea - cuando el objeto de interés está a menos de una distancia ϵ de su posición objetivo. Se decidió como señal de refuerzo dispersa utilizar un refuerzo de $1 - 0.7\frac{\text{numero de pasos ejecutados}}{\text{numero máximo de pasos}}$ en el último paso de cada episodio donde el agente logra el objetivo y un refuerzo de -0.2 en el último paso del episodio si el agente nunca logró el objetivo.

Esta configuración en la que el agente tiene acceso a refuerzos moldeados solo durante el entrenamiento no es la más común. Algunos trabajos que la utilizan son [27, 44, 45]. Se eligió por dos razones. 1) Utilizar sólo refuerzos dispersos puede resultar en problemas demasiado difíciles de aprender para los métodos actuales. Dar acceso a refuerzos densos durante el aprendizaje es una manera de ejercer un balance entre dificultad del aprendizaje y aplicabilidad. 2) Por más que esta configuración sigue requiriendo de refuerzos moldeados, lo que no es ideal, sigue siendo interesante de estudiar. Consideremos una empresa que desarrolla robots que deben emplear tareas autónomamente. Es muy difícil o imposible, una vez desplegado, proveer refuerzos moldeados al robot para todas las tareas y situaciones que se puede llegar a encontrar. Por otro lado, durante el entrenamiento, por ejemplo si éste es llevado a cabo en un ambiente controlado, se vuelve mucho más factible proveer le de esta información; ya sea con refuerzos diseñados manualmente o mediante el uso de sensores que no estarían a disposición en los entornos de operación.

3.2.2. Detalles de implementación

En este trabajo se implementaron varios algoritmos. A continuación los mencionamos y damos detalles de su implementación. Los algoritmos implementados fueron: Un algoritmo de PPO estándar (sección 2.1), un meta algoritmo que estima refuerzos intrínsecos y otro que estima ventajas (sección 3.1), RL^2 (sección 2.1) y MAML a primer orden - FOMAML - (sección 2.2.2). Por simplicidad, en partes del trabajo nos referimos a este simplemente como MAML sin aclarar qué es la variante a primer orden.

Las redes neuronales de todos ellos utilizaron inicialización ortogonal y activaciones tanh. En todos los casos se utilizó el método ADAM como optimizador. Para los valores de salida de las redes se utilizaron distribuciones gaussianas de covarianza nula. En el caso de las redes responsables de generar acciones esto implica que su valor de salida se compone de un vector de dimensión 4 para representar el valor medio de esta distribución y otro de dimensión 4 para la desviación estándar de cada componente (siendo 4 la dimensión del espacio de acciones). De manera análoga, las redes neuronales utilizadas para obtener refuerzos intrínsecos y ventajas meta aprendidas tuvieron un valor de salida unidimensional para representar el valor medio de la estimación y otro para su desviación estándar. El valor de las acciones de salida fue cortado de manera de restringirlas dentro del rango válido.

Para el algoritmo PPO estándar se utilizó una red MLP (64,64) tanto para el actor como para el crítico y una tasa de aprendizaje de 3e-4 constante.

Para el agente de refuerzos intrínsecos, de ventajas y RL^2 se usó una red LSTM. Por cada paso en el entorno, la entrada a la red es procesada por dos capas lineales (128,32), donde en cada capa además se vuelve a concatenar los valores de los datos de entrada que son unidimensionales - o sea, todos los elementos menos las observaciones y acciones. Luego, esta información entra al LSTM; para el cual se usó un estado oculto de dimensión 256 en RL^2, y uno de 128 para los refuerzos intrínsecos y ventajas. La salida del LSTM es luego procesada por diferentes redes MLP. Una (128,128) para estimar el valor central de salida de la red -refuerzo intrínseco,ventaja o acción acordemente-, una (128) para dar un valor de desviación estándar de esta cantidad -inicializado en 1 para las ventajas y RL^2 y en 0.2 para los refuerzos intrínsecos- y una (512) para el crítico. El crítico estima el valor de los meta estados, que codifican la informacion $\mathcal{D}_{:t}$, con respecto al objetivo 2.4. Al estimar refuerzos intrínsecos y ventajas el valor central de salida se pasa por una activación arctan. Todos estos agentes externos fueron entrenados con PPO y usando retropropagación a través del tiempo truncada (TBPTT) con una tasa de aprendizaje de 5e-4 para RL^2 y 5e-5 para el aprendizaje de refuerzos intrínsecos y de ventajas. Estos últimos utilizaron a su vez un algoritmo PPO estándar como el descrito anteriormente para el agente interno encargado de ejecutar acciones en los distintos entornos. Cada tarea comienza con un nuevo agente de estos de tal forma que la única transmisión de información entre tareas para estos algoritmos está en los refuerzos intrínsecos y ventajas aprendidas respectivamente.

Para FOMAML se usó una MLP de (128,128) para estimar el valor central de la acción y una de (128) para determinar la desviación estándar. Se utilizó TRPO para optimizarlas con una tasa de aprendizaje de 5e-4 y una desviación estándar inicial en las acciones de 1.0.

El método de refuerzos intrínsecos y el de ventajas utilizan una variante distinta del objetivo 2.4 que la implementación de RL^2 y de FOMAML. Estos últimos utilizan un descuento al nivel de cada paso mientras que los primeros utilizan un descuento al nivel del episodio.

El uso de refuerzos dispersos durante la evaluación en la práctica significa que MAML en el proceso de adaptación debe estimar el objetivo 2.2 con éstos y para los otros algoritmos significa que el refuerzo extrínseco que entra a la red recurrente a cada paso es el disperso. El uso de refuerzos extrínsecos moldeados durante el entrenamiento significa que los algoritmos de meta aprendizaje pueden estimar el objetivo 2.4 con respecto a estos refuerzos densos.

Detalles complementarios de los algoritmos, de sus hiperparámetros y de su búsqueda se encuentra en el apéndice A. Gran parte de esta información también se encuentra en el código del trabajo[1].

3.2.3. Detalles de entrenamiento y evaluación

En la sección 4 presentamos principalmente dos tipos de resultados: Cómo evoluciona la adaptación/aprendizaje de los algoritmos al encontrarse con una nueva tarea y un valor final de su desempeño en el benchmark (tanto para las tareas de entrenamiento como para las de evaluación).

En la evaluación del desempeño final de los algoritmos se le dio a cada uno 4000 pasos en el entorno (8 episodios) para adaptar la política. Luego, el estado de la política se fijó y se evaluó el desempeño sobre nuevos episodios. Notamos que para RL^2, como la adaptación es continua, se puede considerar que se utilizaron hasta 4500 pasos (un episodio más). La métrica con la que se evaluó es el porcentaje de episodios que terminaron en éxito. El proceso de adaptación también se estudió con esta misma métrica.

El entrenamiento de cada algoritmo fue llevado a cabo de tal manera que también se utilizaron solo 4000 pasos para adaptar las políticas a cada tarea. Para cada actualización del agente externo se usaron datos de 30 distintas tareas - 30 procesos internos -. Otros detalles sobre el entrenamiento se pueden encontrar en el apéndice A .

Los valores reportados en la sección de resultados para cada algoritmo son un promedio del desempeño obtenido para diferentes instancias del método (modelo) que fueron entrenadas desde cero; junto con la desviación estándar de cada valor. Cada instancia se entrenó con una semilla distinta. El efecto de la semilla en los benchmarks consiste en seleccionar el conjunto de variantes paramétricas que se utilizan para cada problema (50 de entre un conjunto infinito). Se utilizaron 5 semillas para el agente PPO estándar, entrenado con refuerzos extrínsecos dispersos o moldeados (sección 4.1), y 3 para todos los métodos de meta aprendizaje. No se utilizó un número mayor de semillas debido a la elevada demanda computacional que tienen los algoritmos de meta RL y restricciones en la cantidad de computo.

A su vez, el valor de desempeño que se consideró para cada modelo entrenado desde cero (para cada semilla) se obtuvo a través de un promedio del valor encontrado en distintas corridas de este; como se describe a continuación: Para cada semilla de entrenamiento, se evaluó al modelo resultante sobre cada conjunto de tareas (ej. tareas de entrenamiento del benchmark ML10) corriéndolo en 10 instancias independientes sobre cada tarea del conjunto. Por ejemplo, para el conjunto de evaluación de ML10 que tiene 5 clases de problemas distintos, cada uno con 50 variaciones paramétricas, se consideraron un total de 2500 corridas del modelo para obtener los valores de desempeño de esa instancia del modelo.

Para algunos algoritmos en algunos benchmarks, se observó una mejora en la tasa de éxito en las tareas de entrenamiento al hacer a la política determinista tras la adaptación. En estos casos los valores de desempeño final reportados (tanto para las tareas de entrenamiento como en sus correspondientes tareas de evaluación) fueron los de la política determinista. En las figuras donde se visualiza la adaptación se muestra principalmente el accionar de las políticas es su estado estocástico; solo en algunas se muestra también el desempeño al hacer a la política final determinista (se aclara cuando esto ocurre).

[1] Código del trabajo

Para el meta agente encargado de estimar refuerzos intrínsecos y el encargado de estimar ventajas siempre se utilizó su versión determinista durante la evaluación (o sea, se ejecutó el valor central de la salida de las redes).

4. Resultados y Discusión

En esta sección se presentan los principales resultados experimentales obtenidos en el desarrollo de la tesis. Antes de esto se introducen las preguntas que busca contestar cada conjunto de experimentos. Por cada pregunta hay una sub-sección con sus correspondientes resultados. Las preguntas que se buscaron contestar son:

1. ¿Cómo se compara aprender con refuerzos intrínsecos aprendidos contra aprender con refuerzos estándar de RL? Sección 4.1
2. ¿Cómo se compara meta aprender refuerzos intrínsecos contra otros métodos de meta aprendizaje? Sección 4.2
3. ¿Hay algo especial sobre los refuerzos intrínsecos? ¿Cómo se compara con meta aprender otra parametrización de la pérdida , o más generalmente otra cantidad? Sección 4.3

Se puede encontrar un resumen del desempeño de cada algoritmo sobre los distintos benchmarks en el apéndice B.

4.1. Comparación con métodos estándar de RL

En esta primera parte se buscó estudiar si se puede obtener algún beneficio en hacer que un agente de aprendizaje por refuerzo aprenda utilizando refuerzos meta aprendidos. Para ello se comparó el desempeño con el de un agente que aprende utilizando los refuerzos moldeados que ofrecen las tareas de Meta World y con otro que aprende utilizando solo los refuerzos dispersos que indican si el agente completo la tarea con éxito (ver sección 3.2.1 para su detalle).

En todos los casos el agente que toma las acciones es un agente entrenado con PPO con los mismos hiperparámetros e inicializado con pesos aleatorios. (Referir a la sección 3.2.2 para más detalles sobre la arquitectura e hiperparametros).

Para este conjunto de experimentos se utilizó tres benchmarks ML1 de Meta World:

- **'ML1 reach'** donde el brazo robótico debe mover su efector hasta una cierta posición del espacio. La posición objetivo varía entre las tareas.
- **'ML1 close door'** donde el agente debe cerrar una puerta. La posición de la puerta varía entre tareas.

- **‘ML1 button press’** donde el agente debe apretar un botón. La posicion del boton varía entre tareas.

La elección de estos benchmarks por sobre los de otros problemas de Meta World fue quasi-arbitraria . No hubo ninguna búsqueda sobre el espacio de problemas. La única condición que se aplicó fue que la versión del algoritmo que entrena con refuerzos moldeados mostrará alguna mejora en el intervalo de pasos en el que se trabajó.

La figura 4.1 muestra cómo se compara el desempeño de los diferentes agentes a medida que interactúan durante más episodios con el entorno . El gráfico está hecho con el desempeño sobre las tareas de evaluación.

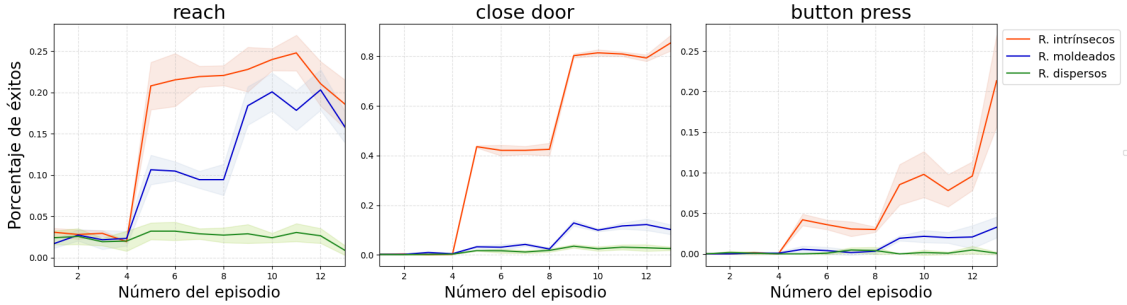

Figura 4.1: Comparación del desempeño promedio de agentes a medida que interactúan con una nueva tarea del conjunto de evaluación. Los valores y sus desviaciones estándar (representadas por la región sombreada) se obtuvieron según fue explicado en la sección 3.2.3. Se muestra el porcentaje de éxitos al entrenar agentes utilizando tres tipos de recompensas: intrínsecas(rojo), moldeadas(azul) y dispersas(verde), en tres benchmarks distintos: ‘ML1-reach’, ‘ML1-close-door’ y ‘ML1-button-press’. El último episodio refleja el desempeño de la política interna al hacerla determinista.

A partir de la figura 4.1 se ve que para estos benchmarks el agente es claramente beneficiado al aprender utilizando los refuerzos intrínsecos meta aprendidos por sobre los refuerzos extrínsecos. También se ve claramente la elección de una política que entrena por lotes y la elección de lotes de 4 episodios(apéndice A.2) ya que los saltos considerables de desempeño ocurren tras la actualización con cada lote. Advertimos también que hacer a la política determinista (reflejado en el último episodio) mejora el desempeño en algunos problemas pero no en todos.

Es importante resaltar que en realidad, en el contexto donde estamos evaluando el algoritmo -donde el agente solo tiene acceso a refuerzos densos durante el entrenamiento- éste no podría utilizar los refuerzos moldeados a la hora de enfrentarse a las tareas de evaluación. Así que la comparación más justa es con el aprendizaje utilizando los refuerzos dispersos. En otras palabras, el agente que entrena utilizando los refuerzos intrínsecos no usa ninguna información de los refuerzos moldeados de las tareas de evaluación. Al tener esto en cuenta, la diferencia es aún mayor. El entrenamiento con refuerzos dispersos muestra mejoras mínimas o nulas en el rango de pasos donde se evaluó. Por otro lado, como fue discutido en la sección 2.2, hay un costo que acompaña a esta mejora. Más específicamente, el costo del meta aprendizaje sobre las tareas de entrenamiento y la necesidad de tener tal conjunto que comparta estructura con las tareas de evaluación. Cosas que los métodos estándar de RL no requieren.

El hecho de que los refuerzos intrínsecos superen también el desempeño de los refuerzos moldeados puede motivar el uso de ellos en el diseño o mejora de los refuerzos diseñados que eligen usar los benchmarks de RL [36].

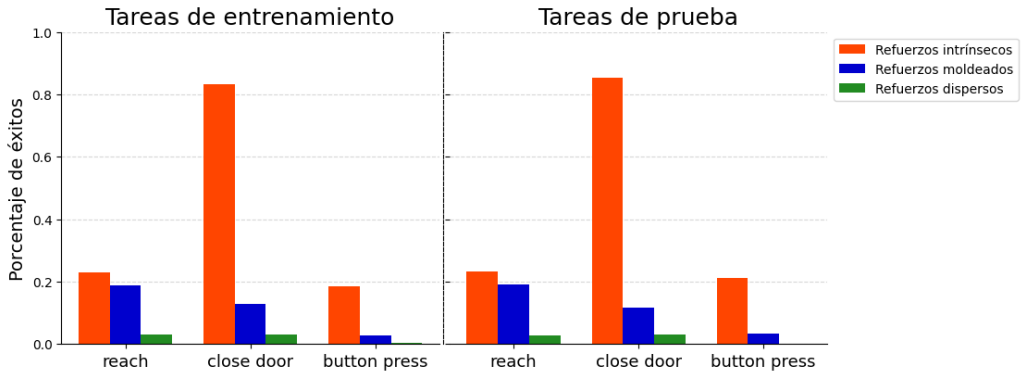

Figura 4.2: Porcentaje de éxito de agentes entrenados con distintos refuerzos sobre diferentes benchmarks de meta aprendizaje, 'ML1-reach', 'ML1-close-door' y 'ML1-button-press', tras un periodo de adaptación de 4000 pasos. La figura compara el desempeño al utilizar tres tipos de refuerzos: intrínsecos(rojo), moldeados(azul) y dispersos(verde). Los valores se obtuvieron según fue explicado en la sección 3.2.3

.

En la figura 4.2 se muestra el desempeño final de cada agente después de tener 4000 pasos de entrenamiento (8 episodios). En primer lugar, notamos que el desempeño de los agentes PPO que entrenan con refuerzos extrínsecos no varía de manera significativa entre las tareas de entrenamiento y las de evaluación. Esto tiene sentido ya que la elección de qué tareas pertenece a cada grupo es arbitraria, la política empieza siempre con parámetros aleatorios, y no hay ninguna información de la interacción con una tarea que estos métodos utilizan al enfrentarse a otra. Un poco más sorprendente quizás es que tampoco hay una pérdida en rendimiento al pasar de las tareas de entrenamiento a las de evaluación al usar los refuerzos intrínsecos. Esto nos dice que para estos benchmarks, donde la variación es paramétrica, el meta agente no tiene problema en generalizar de las tareas de entrenamiento a las de evaluación.

4.2. Comparación con métodos establecidos de meta aprendizaje

Siendo el método desarrollado en este trabajo un método de meta aprendizaje por refuerzo, es natural preguntarse cómo se compara a otros métodos en el área. Este es el objetivo de esta sección. Para ello se implementó RL^2 y una versión de MAML a primer orden (first order MAML [25]) a la cual por simplicidad nos vamos a referir solo como MAML.

En la figura 4.3 se muestra el desempeño que obtienen los diferentes algoritmos de meta aprendizaje a medida que interactúan con el entorno; tanto para las tareas de entrenamiento como para las de evaluación de distintos benchmarks. En esta sección se introduce un nuevo benchmark con respecto a la sección anterior. ML10 es un benchmark de Meta World que contiene 10 tipos de problemas de control continuo para entrenamiento y 5 para evaluación (ver detalles en la sección 3.2.1 y en el apéndice A.1)

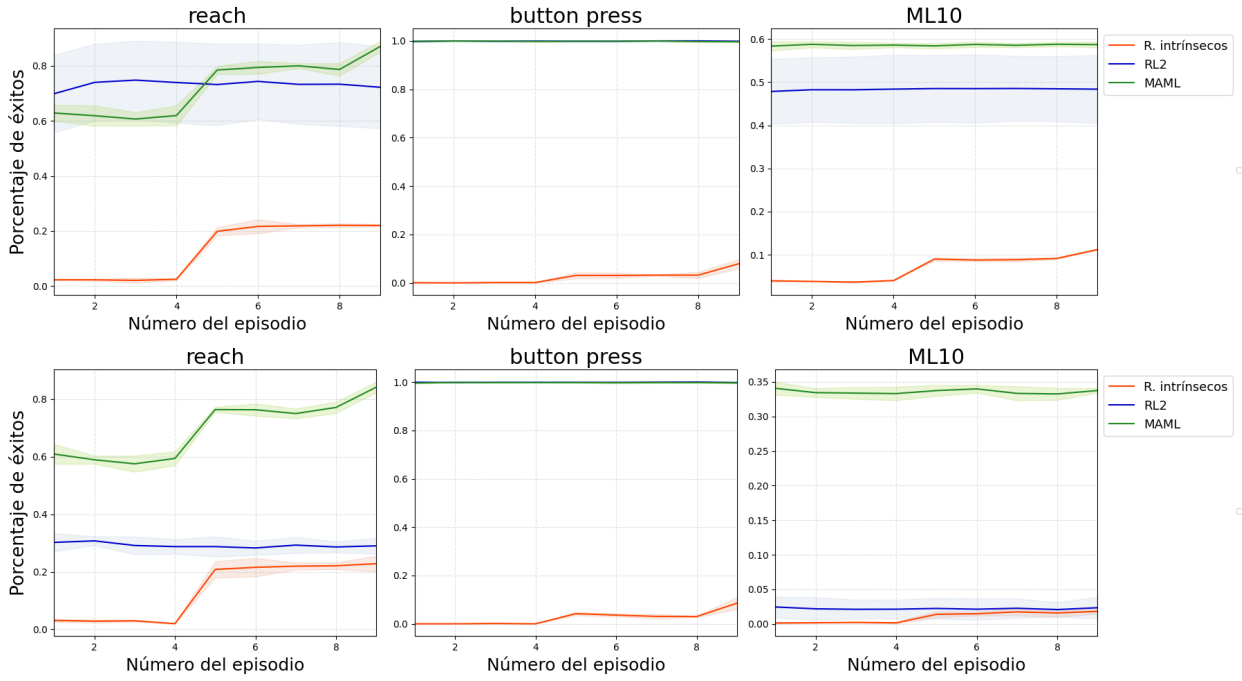

Figura 4.3: Comparación del desempeño promedio de distintos métodos de meta aprendizaje a medida que interactúan con una nueva tarea del conjunto de entrenamiento (fila superior) y del conjunto de evaluación (fila inferior). Los valores y sus desviaciones estándar (representadas por la región sombreada) se obtuvieron según fue explicado en la sección 3.2.3. Se muestra el porcentaje de éxitos para cada uno de ellos: con refuerzos intrínsecos(rojo), RL^2(azul) y MAML(verde), en los benchmarks 'ML1-reach', 'ML1-button-press' y 'ML10'.

Para 'ML1-button-press' se ve que MAML y RL^2 no requieren ningún proceso de adaptación a cada tarea específica. El problema es suficientemente simple tal que la política inicial aprendida puede resolver todas las tareas con éxito; incluso al generalizar a las de evaluación. En ML10 se ve tanto en MAML como en RL^2 que por más que no logran una tasa de éxitos perfecta en todas las tareas, el proceso de adaptación a través de episodios tampoco consigue afectar su desempeño - su probabilidad de éxito en las tareas no mejora al obtener datos de interacción con ella. No logran aprender a utilizar la señal débil que otorgan los refuerzos dispersos de manera eficaz (MAML para las actualizaciones internas y RL^2 como señal de entrada). Lo mismo ocurre con RL^2 en 'ML1-reach'. En cambio, MAML en este benchmark si logra mejorar su desempeño a través de interacciones.

En 'ML1-reach' también notamos que la política base de MAML y el proceso de adaptación generalizan de manera casi perfecta a las tareas de evaluación; obtiene una tasa de éxito muy similar a la de las tareas de entrenamiento (figura 4.4). Por otro lado, la política que aprende RL^2 sufre una caída considerable en desempeño al pasar a las tareas de evaluación en 'ml1-reach'. Algo similar ocurre en el benchmark ML10. En este RL^2 más que logra una tasa de éxito de casi el 50 % en las tareas de entrenamiento obtiene solo un 0,02 % en las de evaluación. MAML en cambio obtiene una tasa de éxito de casi el 60 % en las tareas de entrenamiento y una del 33 % en las de evaluación así que de nuevo la política que aprende MAML generaliza mejor que la de RL^2.

A diferencia de RL^2 y MAML, en el método con refuerzos intrínsecos se ve que para las tareas de entrenamiento de todos los benchmarks el desempeño del agente mejora al obtener más datos. Esto indica que el algoritmo logra utilizar información de la interacción con el entorno para mejorar la política. Notar que esto no nos dice si necesariamente la función de refuerzos en sí transcurre un proceso de adaptación a la tarea que se le presenta o si aprendió a dar refuerzos que son útiles sin distinguir con cual tarea de la distribución lidia. En los dos benchmarks con sólo variaciones paramétricas la tasa de éxitos es prácticamente igual para las tareas de evaluación y las de entrenamiento. Esto nos dice que la función de refuerzos meta aprendida generaliza bien de un conjunto al otro y que su proceso de adaptación también lo hace o es despreciable.

Una conclusión importante de la figura 4.3 y 4.4, aunque quizás no sorprendentemente, de los resultados es que el desempeño general del algoritmo que utiliza los refuerzos intrínsecos es considerablemente inferior al de los otros dos métodos en el intervalo de pasos donde se evaluó. Esto subraya la importancia de no iniciar cada tarea con una política completamente nueva cuando se busca desempeño disponiendo de pocos datos. Incluso si se tuviese los refuerzos óptimos (en el sentido de [37]) hay tareas que requieren más de 2 actualizaciones para poder aprenderse desde cero. El aprendizaje de refuerzos intrínsecos podría ser más útil si 1) se extendiera a aprendizajes más largos y/o 2) se combinará con un método que aprenda una política base con la que se comienzan las tareas. Lograr (1) viene con sus complicaciones. Simplemente alargar indefinidamente la duración de los entrenamientos internos puede llevar a gradientes que explotan o desaparecen y también requiere un aumento de computo. La elección de arquitectura puede ayudar pero no eliminar estos problemas. Por ejemplo, el uso de una LSTM o GRU (*Gated Recurrent Unit*) reduce el problema de gradientes que explotan o desaparecen en comparación con una RNN. Para (2), una política base desde la cual comenzar se podría aprender con métodos de meta learning (como RL^2 y MAML) o con algún método de RL multitarea.

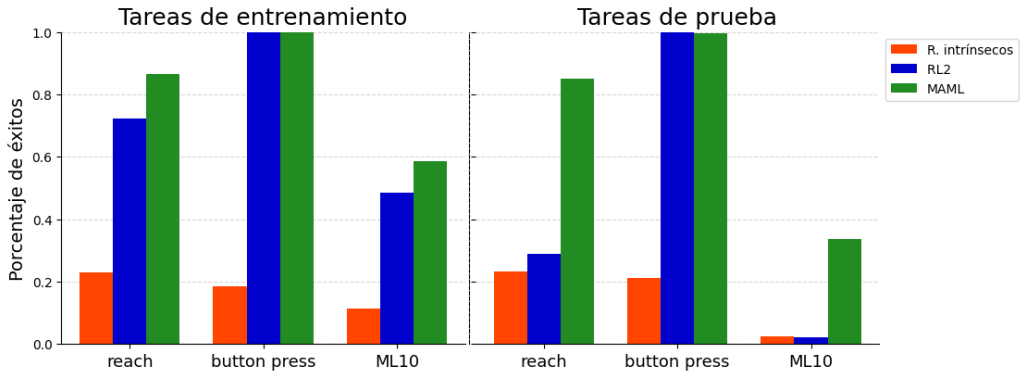

Figura 4.4: Porcentaje de éxitos de métodos de meta aprendizaje sobre distintos benchmarks: 'ML1-reach', 'ML1-button-press' y 'ML10' tras un periodo de adaptación de 4000 pasos. La figura compara el desempeño de los métodos: Refuerzos intrínsecos(rojo), RL^2(azul) y MAML(verde). Los valores se obtuvieron según fue explicado en la sección 3.2.3.

En resumen: Todo los buenos resultados que obtiene RL^2 en esta configuración de refuerzos dispersos lo logra a partir de encontrar una política que generaliza entre tareas y no logra adaptar la política para mejorar su desempeño al interactuar con una tarea durante múltiples episodios. Aparte en 2 de los 3 benchmarks se observa que esta política sufre una caída significante de rendimiento al pasar a las tareas de evaluación. En particular la generalización es muy pobre en ML10, donde las variaciones son no paramétricas. Para MAML se encontró que en el benchmark más simple de los dos donde el desempeño de la política base no es óptimo, logra aprender a adaptar su política a cada tarea encontrada. Además es el método que obtuvo el mejor desempeño en general y generalizó muy bien en los dos benchmarks de variaciones paramétricas. Mientras que para ML10 su política aprendida obtuvo resultados muy superiores a los de los otros dos métodos pero no a tono con su desempeño sobre las tareas de entrenamiento. El método en el que se centra esta tesis se diferencia de RL^2 y MAML en que la cantidad meta aprendida no es una política. Esto significa que al comenzar a interactuar con una tarea, tiene el desempeño de una política aleatoria. Por más que en todos los benchmarks el desempeño aumenta al tener más interacción con el entorno, este no alcanza el nivel de RL^2 y MAML.

Una ventaja de meta aprender refuerzos intrínsecos, por sobre acciones u otras cantidades, es que estos se pueden utilizar para entrenar distintos tipos de agentes internos. Una política meta aprendida se puede naturalmente utilizar para entrenar otras políticas con una red neuronal distinta a través de destilación o imitación. Pero esta transición de conocimiento se vuelve no tan trivial cuando el nuevo agente es por ejemplo un agente que utiliza una función de valor en lugar de una política explícita o cuando la nueva política tiene un espacio de acciones distinto. En cambio los refuerzos meta aprendidos se pueden utilizar para entrenar casi cualquier tipo de agente (que mantiene el mismo espacio de observación) directamente reemplazando los refuerzos extrínsecos por los intrínsecos. Es-

te punto sobre capturar 'que está bien hacer' en vez de 'cómo hacerlo' es discutido más en detalle en [34]. Acá, es importante señalar que en el contexto de este trabajo, la red neuronal de refuerzos intrínsecos tomó las acciones del agente como uno de los valores de entrada, por lo que no es obvio que se pueda aplicar a agentes con un espacio de acciones distinto. Si el refuerzo en un paso está principalmente relacionado al cambio de estado, entonces la red de refuerzos intrínsecos puede no verse severamente afectada por el uso de acciones no vistas durante el entrenamiento; este comportamiento se observa en [34]. También se puede directamente diseñar la red neuronal para que no tome en cuenta las acciones ejecutadas por el agente. Incluso se la puede hacer independiente del espacio de observación; en el estilo de [30].

4.3. Refuerzos intrínsecos contra el meta aprendizaje de otras cantidades

Hasta ahora el foco del trabajo estuvo en el estudio de meta aprender una función de refuerzos intrínsecos. En la sección anterior se discutió ventajas y desventajas de esto contra meta aprender los parámetros iniciales de una política(MAML) y a meta aprender una política que toma en cuenta toda su historia de interacción con el entorno(RL^2). Es lógico preguntarse ¿qué otras cantidades se pueden meta aprender? y ¿cual es la mejor?.

El mismo formalismo con el que se aprendieron refuerzos intrínsecos (descrito en la sección 3.1) se podría aplicar para meta aprender cualquier otra cantidad que tenga influencia sobre las acciones que toma el agente. Sea esta influencia diferenciable o no. La elección de estudiar el meta aprendizaje de refuerzos intrínsecos fue principalmente por 2 razones: 1) estos son un tema muy estudiado en RL y han mostrado ventajas en reiteradas ocasiones (sección 2.3), y 2) Todos los algoritmos estándar de RL asumen que reciben refuerzos; por lo que los refuerzos meta aprendidos se pueden incorporar a cualquiera de ellos directamente.

Como se dijo , cualquier cantidad o decisión puede ser meta aprendida. En particular en la sección 2.2.2 se discutió una familia de meta algoritmos que intentan aprender parámetros de las actualizaciones internas de algoritmos de RL estándar. Tanto meta aprender los parámetros iniciales de la política (MAML) como meta aprender refuerzos intrínsecos se pueden ver como ejemplos de esto. Para seguir el estudio de si hay alguna razón para preferir meta aprender refuerzos intrínsecos por sobre otras influencias, proponemos meta aprender otra cantidad que entra dentro de esta categoría. Se entrenó un agente para que directamente aprenda una función de ventaja.

Como se discutió en la sección 2.1, los métodos de gradiente de política pueden usar las ventajas para estimar que tan buena fue una acción tomada en una transición. Lo que meta aprende el agente es, para ser más precisos, la función Φ_t en la ecuación 2.3. De esta manera el meta agente en vez de aprender a asignar una contribución parcial a la bondad de una transición y de las transiciones pasadas (como es el caso con los refuerzos intrínsecos) , aprende a asignarle a cada transición su bondad independientemente. Al estimar directamente las ventajas se puede prescindir del crítico en el algoritmo PPO interno.

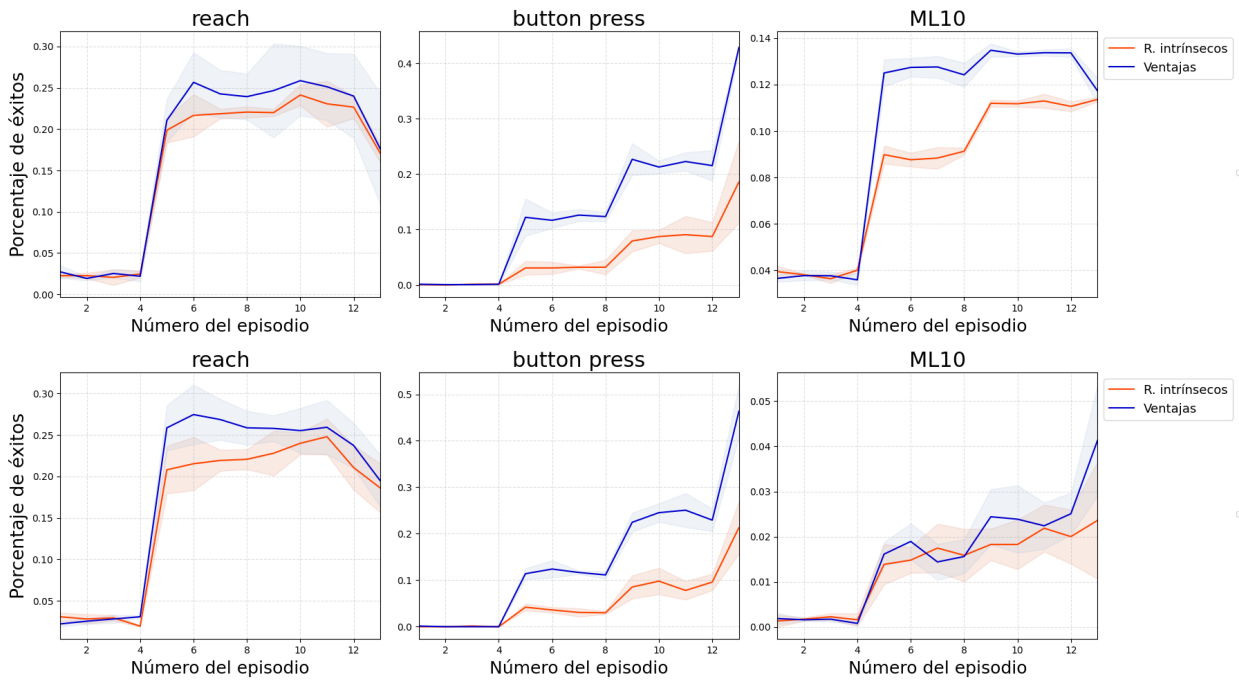

Figura 4.5: Comparación del desempeño promedio de métodos de meta aprendizaje a medida que interactúan con una nueva tarea del conjunto de entrenamiento (fila superior) y del conjunto de evaluación (fila inferior).Los valores y sus desviaciones estándar (representadas por la región sombreada) se obtuvieron según fue explicado en la sección 3.2.3. Se muestra el porcentaje de éxitos para dos métodos que aprenden distintas parametrizaciones de la pérdida: refuerzos intrínsecos(rojo) y ventajas(azul), en los benchmarks 'ML1-reach', 'ML1-button-press' y 'ML10'. El ultimo episodio refleja el desempeño de la política interna al hacerla determinista.

En la figura 4.5 se ve un tipo de comportamiento similar en el aprendizaje de la política cuando ésta hace uso de las ventajas o de los refuerzos intrínsecos. En ambos casos las políticas internas logran utilizar los datos de interacción para mejorar sus tasas de desempeño en todos los benchmarks. En esta figura y en la figura 4.6 se ve también que el uso de las ventajas da incluso una mejoría. La misma solo es significativa para la tarea 'ML1-button-press' y para las tareas de entrenamiento de ML10. De nuevo, hacer a la política final determinista (reflejado en el último episodio de la figura 4.5), sólo muestra mejoras en algunos de los conjuntos de tareas; en otros deteriora el rendimiento. Otro punto en común es que al usar ambas parametrizaciones de la función de pérdida se tiene buena generalización para los benchmarks con variaciones paramétricas. El desempeño para 'ML1-reach' y para 'ML1-button-press' no empeora al pasar a las tareas de evaluación (figura 4.5). En cambio en ML10 ambos métodos alcanzan una tasa de éxito muy baja sobre las tareas de evaluación.

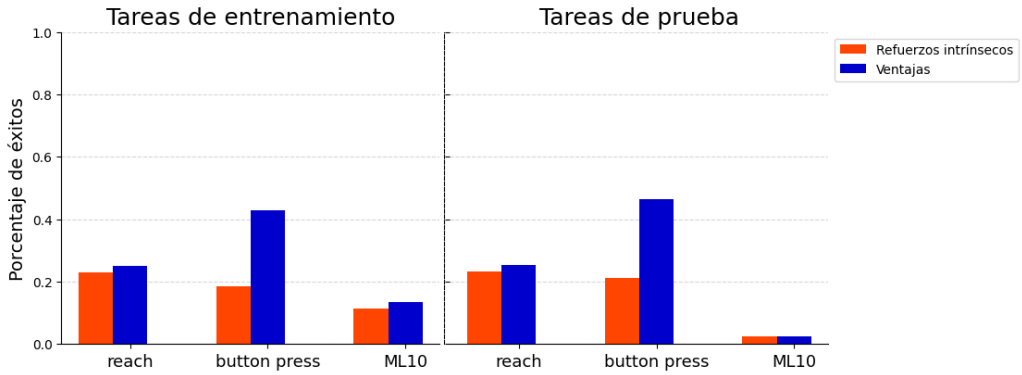

Figura 4.6: Porcentaje de éxitos sobre distintos benchmarks: 'ML1-reach', 'ML1-button-press' y 'ML10' tras un periodo de adaptación de 4000 pasos. La figura compara el desempeño de políticas que utilizan refuerzos intrínsecos meta aprendidos (rojo) y ventajas meta aprendidas (azul). Los valores se obtuvieron según fue explicado en la sección 3.2.3.

Los experimentos realizados respaldan la noción de que no hay necesariamente algo demasiado especial en meta aprender refuerzos intrínsecos. Cuál es la cantidad que conviene meta aprender es un problema abierto y es muy posible que dependa del problema en cuestión. La decisión óptima dependerá de sobre qué espacio es más fácil meta aprender un algoritmo de aprendizaje más eficiente y de con qué espacio el algoritmo meta aprendido generaliza mejor a nuevas tareas. También considerando otras características que se puedan buscar; por ejemplo si se tiene que lidiar con distintas arquitecturas y algoritmos de aprendizaje en los procesos internos. Paralelamente a esto (también discutido en la sección anterior), nada impide meta aprender más de un tipo de influencia sobre la política final. Se ha mostrado que meta aprender imponiendo menos estructura y dándole más libertad al meta agente a explorar el espacio de algoritmos de aprendizaje internos puede ser beneficioso si se cuentan con suficientes datos [30].

5. Conclusiones

Esta tesis se centró en meta aprendizaje en el contexto de aprendizaje por refuerzo. En particular la mayor parte del foco estuvo en el meta aprendizaje de señales de entrenamiento.

Hay muchos trabajos en la literatura que estudian el uso de refuerzos intrínsecos. Mientras que la mayoría de ellos se basa en heurísticas, este trabajo buscó aprenderlos en el marco de meta aprendizaje. La principal diferencia con otros métodos que hacen esto fue en cómo fue llevado a cabo el proceso de optimización. En vez de usar algoritmos evolutivos o meta gradientes, que es la estrategia más popular a la hora de meta aprender parámetros de la actualización interna, en este trabajo se construyó una función de refuerzos estocástica entrenada con RL sin tener en cuenta de manera explícita su influencia sobre la política interna (guardando así similitud con el método RL^2). La omisión de meta gradientes puede traer ventajas en costo computacional, y en flexibilidad a la hora de elegir la cantidad meta aprendida y el proceso interno utilizado; pero sacrificando potencialmente una señal de entrenamiento, para el meta agente, de menos varianza (ya que hay mas fuentes de estocasticidad que influencian su valor; sección 3.1). Otra particularidad del trabajo es el contexto de evaluación empleado. Se trabajó con problemas de control continuo donde el agente no tiene acceso a refuerzos densos al enfrentarse a las tareas de evaluación. Además de esto, se utilizaron variaciones tanto paramétricas como no paramétricas entre las distintas tareas. La mayoría de los trabajos previos utilizan refuerzos moldeados y solo variaciones paramétricas o utilizan refuerzos dispersos pero en entornos discretos sobre grillas.

Con respecto a los resultados experimentales, estos operaron en un periodo de adaptación de hasta tan solo 8 episodios. En este rango, el entrenamiento de un algoritmo estándar de RL con refuerzos dispersos mostró mínima mejora. Resultó ser ampliamente superado por el entrenamiento con los refuerzos meta aprendidos; con los cuales también se superó marcadamente el entrenamiento con refuerzos moldeados. Por otro lado, se encontró el rendimiento del método en este rango de episodios como muy desfavorable en comparación a métodos de meta aprendizaje que aprenden una política. MAML fue el algoritmo que mejor desempeño obtuvo; en particular fue el único que logró una tasa de éxito apreciable al lidiar con variaciones no paramétricas (dónde RL^2 y los refuerzos intrínsecos sufrieron en la generalización a las tareas de evaluación). Se llegó a la conclusión de que el meta aprendizaje de refuerzos intrínsecos podría ser más beneficioso en un contexto donde el periodo de adaptación es más largo [34] (situación donde por otro lado se incrementa la dificultad del proceso de optimización y la demanda de recursos computacionales), o al combinarlo con una política no inicializada desde cero. Más allá de esto, todos los métodos de meta aprendizaje evaluados mostraron espacio para mejora

en los benchmarks.

Mirando no solo el rendimiento final, se encontró que MAML y RL^2, a diferencia de al usar refuerzos intrínsecos, usualmente no mejoraban en desempeño al obtener más datos de interacción con una tarea.

Finalmente se extendió está discusión al marco más general de ¿Qué cantidad conviene meta aprender? Se remarcó que este es un problema abierto y porqué su respuesta puede ser dependiente del conjunto de tareas que se buscan resolver, de la cantidad de datos a la que se tiene acceso y del objetivo. Se discutieron algunas ventajas de los refuerzos intrínsecos: que estos se pueden incorporar a cualquier algoritmo estándar de aprendizaje por refuerzos simplemente reemplazando recompensas extrínsecas por intrínsecas (en particular, se puede transmitir información/entrenar una mayor diversidad de agentes que si solo se tiene acceso a una política), que pueden utilizarse para guiar el diseño de refuerzos moldeados y que ya han mostrado reiteradamente en trabajos previos (estudiado principalmente para refuerzos construidos a partir de heurísticas) que pueden mejorar el desempeño del aprendizaje. También, bajo este mismo análisis, se mostró que con un simple cambio -meta aprender una función de ventaja en lugar de refuerzos intrínsecos- se puede obtener tasas de éxito tan buenas e incluso mejores sobre los benchmarks.

En esta tesis se encuentran muchas motivaciones para trabajo futuro. Más allá de las motivaciones generales del meta aprendizaje y del uso de refuerzos intrínsecos que se presentaron en la sección 2 , hay direcciones que en particular son más cercanas al trabajo de esta tesis. Algunas direcciones que resaltan a lo largo del trabajo como posibles temas de trabajo futuro son: la combinación de refuerzos intrínsecos con el meta aprendizaje de otras cantidades (en particular una política), exploración del espacio de parámetros para meta aprender, estudio cuantitativo entre meta aprender una cantidad con un método de caja negra y con meta gradientes, y técnicas para mejorar el proceso de adaptación y la generalización al lidiar con refuerzos dispersos. Algunas de estas direcciones ya son estudiadas por otros trabajos. Por ejemplo [46, 47] proponen técnicas de meta aprendizaje para entornos con refuerzos dispersos y en la sección 2.2.2 se presentaron varios trabajos que exploran el meta aprendizaje de distintas cantidades.

Aún más específico a este trabajo, hay posibles mejoras por sobre lo presentado en este texto a la hora de entrenar un método que parametriza la función de pérdida. Se mencionan algunas. En la implementación se utilizó una LSTM, por lo cual los refuerzos estimados utilizan sólo información de los pasos previos. Al utilizar un algoritmo que entrena por lotes en el proceso interno (como PPO), se le podría dar al meta agente acceso a todos los pasos futuros que entran en el lote para que haga sus predicciones. Otra opción para mejorar la eficiencia de datos podría ser reutilizar datos haciendo que la red estime refuerzos para datos *off policy*. Por ultimo, como fue mencionado, es importante extender el método a procesos de adaptación más largos.

Bibliografía

[1] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel *et al.*, "Mastering atari, go, chess and shogi by planning with a learned model," *Nature*, vol. 588, no. 7839, pp. 604–609, 2020.

[2] A. Fawzi, M. Balog, A. Huang, T. Hubert, B. Romera-Paredes, M. Barekatain, A. Novikov, F. J. R Ruiz, J. Schrittwieser, G. Swirszcz *et al.*, "Discovering faster matrix multiplication algorithms with reinforcement learning," *Nature*, vol. 610, no. 7930, pp. 47–53, 2022.

[3] D. J. Mankowitz, A. Michi, A. Zhernov, M. Gelmi, M. Selvi, C. Paduraru, E. Leurent, S. Iqbal, J.-B. Lespiau, A. Ahern *et al.*, "Faster sorting algorithms discovered using deep reinforcement learning," *Nature*, vol. 618, no. 7964, pp. 257–263, 2023.

[4] J. Degrave, F. Felici, J. Buchli, M. Neunert, B. Tracey, F. Carpanese, T. Ewalds, R. Hafner, A. Abdolmaleki, D. de Las Casas *et al.*, "Magnetic control of tokamak plasmas through deep reinforcement learning," *Nature*, vol. 602, no. 7897, pp. 414–419, 2022.

[5] J. Luo, C. Paduraru, O. Voicu, Y. Chervonyi, S. Munns, J. Li, C. Qian, P. Dutta, J. Q. Davis, N. Wu, X. Yang, C.-M. Chang, T. Li, R. Rose, M. Fan, H. Nakhost, T. Liu, B. Kirkman, F. Altamura, L. Cline, P. Tonker, J. Gouker, D. Uden, W. B. Bryan, J. Law, D. Fatiha, N. Satra, J. Rothenberg, M. Waraich, M. Carlin, S. Tallapaka, S. Witherspoon, D. Parish, P. Dolan, C. Zhao, and D. J. Mankowitz, "Controlling commercial cooling systems using reinforcement learning," 2022.

[6] T. M. Moerland, J. Broekens, A. Plaat, and C. M. Jonker, "Model-based reinforcement learning: A survey," 2022.

[7] S. Pateria, B. Subagdja, A.-h. Tan, and C. Quek, "Hierarchical reinforcement learning: A comprehensive survey," *ACM Computing Surveys (CSUR)*, vol. 54, no. 5, pp. 1–35, 2021.

[8] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," 2020.

[9] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine, "Diversity is all you need: Learning skills without a reward function," 2018.

[10] M. Laskin, A. Srinivas, and P. Abbeel, "CURL: Contrastive unsupervised representations for reinforcement learning," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, H. D. III and A. Singh, Eds., vol. 119. PMLR, 13–18 Jul 2020, pp. 5639–5650. [Online]. Available: https://proceedings.mlr.press/v119/laskin20a.html

[11] N. Vithayathil Varghese and Q. H. Mahmoud, "A survey of multi-task deep reinforcement learning," *Electronics*, vol. 9, no. 9, 2020. [Online]. Available: https://www.mdpi.com/2079-9292/9/9/1363

[12] J. Beck, R. Vuorio, E. Z. Liu, Z. Xiong, L. Zintgraf, C. Finn, and S. Whiteson, "A survey of meta-reinforcement learning," *arXiv preprint arXiv:2301.08028*, 2023.

[13] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, 2018.

[14] A. Aubret, L. Matignon, and S. Hassas, "A survey on intrinsic motivation in reinforcement learning," 2019.

[15] T. Yu, D. Quillen, Z. He, R. Julian, A. Narayan, H. Shively, A. Bellathur, K. Hausman, C. Finn, and S. Levine, "Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning," 2021.

[16] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in Neural Information Processing Systems*, S. Solla, T. Leen, and K. Müller, Eds. MIT Press.

[17] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[18] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[19] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*. PMLR, 2015, pp. 1889–1897.

[20] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[21] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, "Meta-learning in neural networks: A survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 9, pp. 5149–5169, 2021.

[22] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel, "Rl2: Fast reinforcement learning via slow reinforcement learning," *arXiv preprint arXiv:1611.02779*, 2016.

[23] J. X. Wang, Z. Kurth-Nelson, D. Tirumala, H. Soyer, J. Z. Leibo, R. Munos, C. Blundell, D. Kumaran, and M. Botvinick, "Learning to reinforcement learn," *arXiv preprint arXiv:1611.05763*, 2016.

[24] Z. Xu, H. van Hasselt, and D. Silver, "Meta-gradient reinforcement learning," 2018.

[25] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International conference on machine learning*. PMLR, 2017, pp. 1126–1135.

[26] A. Raghu, M. Raghu, S. Bengio, and O. Vinyals, "Rapid learning or feature reuse? towards understanding the effectiveness of maml," *arXiv preprint arXiv:1909.09157*, 2019.

[27] A. Gupta, R. Mendonca, Y. Liu, P. Abbeel, and S. Levine, "Meta-reinforcement learning of structured exploration strategies," *Advances in neural information processing systems*, vol. 31, 2018.

[28] Z. Li, F. Zhou, F. Chen, and H. Li, "Meta-sgd: Learning to learn quickly for few-shot learning," *arXiv preprint arXiv:1707.09835*, 2017.

[29] E. Park and J. B. Oliva, "Meta-curvature," *Advances in neural information processing systems*, vol. 32, 2019.

[30] J. Oh, M. Hessel, W. M. Czarnecki, Z. Xu, H. van Hasselt, S. Singh, and D. Silver, "Discovering reinforcement learning algorithms," 2021.

[31] L. Kirsch, S. van Steenkiste, and J. Schmidhuber, "Improving generalization in meta reinforcement learning using learned objectives," *arXiv preprint arXiv:1910.04098*, 2019.

[32] Z. Zheng, J. Oh, and S. Singh, "On learning intrinsic rewards for policy gradient methods," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[33] B. Stadie, L. Zhang, and J. Ba, "Learning intrinsic rewards as a bi-level optimization problem," in *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI)*, ser. Proceedings of Machine Learning Research, J. Peters and D. Sontag, Eds., vol. 124. PMLR, 03–06 Aug 2020, pp. 111–120.

[34] Z. Zheng, J. Oh, M. Hessel, Z. Xu, M. Kroiss, H. Van Hasselt, D. Silver, and S. Singh, "What can learned intrinsic rewards capture?" in *International Conference on Machine Learning*. PMLR, 2020, pp. 11 436–11 446.

[35] F. Alet, M. F. Schneider, T. Lozano-Perez, and L. P. Kaelbling, "Meta-learning curiosity algorithms," *arXiv preprint arXiv:2003.05325*, 2020.

[36] H. Zou, T. Ren, D. Yan, H. Su, and J. Zhu, "Reward shaping via meta-learning," 2019.

[37] S. Singh, R. Lewis, and A. Barto, "Where do rewards come from?" 01 2009.

[38] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," 2018.

[39] R. Houthooft, Y. Chen, P. Isola, B. Stadie, F. Wolski, O. Jonathan Ho, and P. Abbeel, "Evolved policy gradients," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[40] F. Sung, L. Zhang, T. Xiang, T. Hospedales, and Y. Yang, "Learning to learn: Meta-critic networks for sample efficient learning," 2017.

[41] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[42] B. C. Stadie, G. Yang, R. Houthooft, X. Chen, Y. Duan, Y. Wu, P. Abbeel, and I. Sutskever, "Some considerations on learning to explore via meta-reinforcement learning," 2019.

[43] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete problems in ai safety," 2016.

[44] K. Rakelly, A. Zhou, C. Finn, S. Levine, and D. Quillen, "Efficient off-policy meta-reinforcement learning via probabilistic context variables," in *International conference on machine learning*. PMLR, 2019, pp. 5331–5340.

[45] T. Z. Zhao, A. Nagabandi, K. Rakelly, C. Finn, and S. Levine, "Meld: Meta-reinforcement learning from images via latent state models," *arXiv preprint arXiv:2010.13957*, 2020.

[46] C. Packer, P. Abbeel, and J. E. Gonzalez, "Hindsight task relabelling: Experience replay for sparse reward meta-rl," *Advances in Neural Information Processing Systems*, vol. 34, pp. 2466–2477, 2021.

[47] L. M. Zintgraf, L. Feng, C. Lu, M. Igl, K. Hartikainen, K. Hofmann, and S. Whiteson, "Exploration in approximate hyper-state space for meta reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2021, pp. 12 991–13 001.

[48] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," 2018.

A. Apéndice de detalles experimentales

Este apéndice da más información sobre los experimentos que se llevaron a cabo en el trabajo.

La sección A.1 describe el detalle de los distintos tipos de problemas involucrados en el benchmark ML10 de Meta World. El resto de las secciones da más información sobre las implementaciones de los distintos algoritmos. En una primera parte A.2 se presentan tablas con los valores de distintos hiperparámetros de cada método. Su nomenclatura coincide con la utilizada en el código del trabajo. Luego en A.3 se da una discusión sobre los distintos hiperparámetros destacando algunos de ellos. En una última sección A.4 se dan otros detalles sobre la implementación y evaluación.

A.1. Detalles tareas de ML10

En la sección 3.2.1 se da una descripción general de los benchmarks de Meta World y en la sección 4.1 una descripción de las tareas en los benchmarks ML1 utilizados. Esta sección del apéndice ofrece una descripción de las tareas involucradas en el benchmark ML10.

Tareas de Entrenamiento

- **insert peg side:** el agente debe insertar una clavija de costado. La posición de la clavija y del objetivo varían entre las tareas.
- **reach:** el agente debe mover su efector hasta una cierta posición del espacio. La posición objetivo varía entre las tareas.
- **open door:** el agente debe abrir una puerta con una bisagra giratoria. La posición de la puerta varía entre las tareas.
- **basketball:** el agente debe encestar el balón de baloncesto. La posición del balón y de la canasta varían entre las tareas.
- **open window:** se debe empujar y abrir una ventana. La posición de la ventana varía entre las tareas.
- **pick & place:** se debe recoger y colocar un disco en un objetivo. La posición del disco y del objetivo varían entre las tareas.
- **press button topdown:** el agente debe presionar un botón desde arriba. La posición del botón varía entre las tareas.

- **push:** el agente debe empujar el disco hacia un objetivo. La posición del disco y del objetivo varían entre las tareas.
- **close drawer:** el agente debe empujar y cerrar un cajón. La posición del cajón varía entre las tareas.
- **sweep:** el agente debe barrer un disco fuera de la mesa. La posición del disco varía entre las tareas..

Tareas de Evaluación

- **close door:** el agente debe cerrar una puerta con una bisagra giratoria. La posición de la puerta varía entre las tareas.
- **place onto shelf:** el agente debe recoger y colocar un disco en un estante. La posición del disco y del estante varían entre las tareas.
- **open drawer:** el agente debe abrir un cajón. La posición del cajón varía entre las tareas.
- **sweep into hole:** el agente debe barrer un disco hacia un agujero. La posición del disco varía entre las tareas.
- **pull lever:** el agente debe tirar de una palanca hacia abajo 90 grados. La posición de la palanca varía entre las tareas.

A.2. Listas de hiperparámetros

Agente PPO standard

Esta lista de hiperparámetros para el agente PPO es la utilizada tanto para el agente PPO por su cuenta como cuando se utiliza dentro de los algoritmos de meta aprendizaje (con el rol de política interna).

Hiperparámetro	**Valor**
total_timesteps	6000
num_steps	2000
learning_rate	3e-4
adam_eps	1e-5
gamma	0.99
gae	True
gae_lambda	0.95
ppo.update_epochs	64
ppo.num_minibatches	16
ppo.normalize_advantage	True
ppo.clip_coef	0.2
ppo.entropy_coef	0.0
ppo.valuef_coef	0.5
ppo.clip_grad_norm	True
ppo.max_grad_norm	0.5
ppo.target_KL	None

Cuadro A.1: Tabla con la configuración de hiperparámetros utilizada para el entrenamiento y evaluación del agente PPO estándar. Válida tanto para su evaluación individual como al ser usado con señales meta aprendidas.

Refuerzos intrinsicos y Ventajas

Hiperparámetro	**Valor**
shaped_rewards_available_at_train_time	True
shaped_rewards_available_at_test_time	False
num_epsiodes_of_validation	4
num_lifetimes_for_validation	60
num_inner_loops_per_update	30
learning_rate	5e-5
adam_eps	1e-5
e_rewards_target_mean	0.0001
meta_gamma	0.9
ae.estimation_method	'bootstrapping skipping uninfluenced future rewards'
ae.bootstrapping_lambda	0.85
ae.starting_n	2200
ae.num_n_step_estimates	6
ae.skip_rate	300
rnn_input_size	32
rnn_type	lstm
rnn_hidden_state_size	128
initial_std	R.intrínsecos=0.2 Ventajas=1.0
ppo.k	400
ppo.update_epochs	12
ppo.num_minibatches	"0" $\approx \left(\frac{\text{num inner loop steps}}{\text{ppo.k}}\right)$
ppo.normalize_advantage	True
ppo.clip_coef	0.2
ppo.entropy_coef	0.0005 R.intr.ML1=0.003
ppo.valuef_coef	0.5
ppo.clip_grad_norm	True
ppo.max_grad_norm	0.5
ppo.target_KL	0.01

Cuadro A.2: Tabla con la configuración de hiperparámetros utilizada para el entrenamiento y evaluación del meta agente encargado de entregar refuerzos intrínsecos y el encargado de entregar ventajas.

RL2

Hiperparámetro	**Valor**
num_epsiodes_of_validation	2
num_lifetimes_for_validation	90
num_inner_loops_per_update	30
num_il_lifetime_steps	4500
learning_rate	5e-4
adam_eps	1e-5
e_rewards_target_mean	0.1
meta_gamma	0.995
ae.estimation_method	'standard GAE'
ae.bootstrapping_lambda	0.95
rnn_input_size	32
rnn_type	lstm
rnn_hidden_state_size	256
initial_std	1.0
ppo.k	400
ppo.update_epochs	10
ppo.num_minibatches	"0" $\approx \left(\frac{\texttt{num\_il\_lifetime\_steps}}{\texttt{ppo.k}}\right)$
ppo.normalize_advantage	True
ppo.clip_coef	0.2
ppo.entropy_coef	ML10=0.0 ML1=0.005
ppo.valuef_coef	0.5
ppo.clip_grad_norm	True
ppo.max_grad_norm	0.5
ppo.target_KL	0.1

Cuadro A.3: Tabla con la configuración de hiperparámetros utilizada para el entrenamiento y evaluación del método RL^2.

MAML

Hiperparámetro	Valor
num_epsiodes_of_validation	4
num_lifetimes_for_validation	120
num_inner_loops_per_update	30
num_adaptation_updates_in_inner_loop	2
num_env_steps_per_adaptation_update	2000
num_env_steps_for_estimating_maml_loss	4000
maml_agent_lr	5e-4
maml_agent_epsilon	1e-5
adaptation_lr	ML10,button_press=1e-3 reach=7e-2
adaptation_gamma	0.995
e_rewards_target_mean	0.1
meta_gamma	0.995
TRPO.cg_damping	1e-2
TRPO.max_kl	0.001
TRPO.cg_iters	10
TRPO.line_search_max_steps	10
TRPO.line_search_backtrack_ratio	0.6

Cuadro A.4: Tabla con la configuración de hiperparámetros utilizada para el entrenamiento y evaluación del método MAML.

A.3. Discusión sobre hiperparámetros y su elección

En la sección anterior se dan valores para la elección de muchos hiperparámetros. Muchos de ellos son hiperparámetros estándar en implementaciones de RL profundo. El significado de varios otros resulta evidente a partir de su nombre. En esta sección se da una breve descripción de aquellos cuya influencia puede ser menos clara. Luego también se destaca la influencia de algunos hiperparámetros específicos.

Dentro de los hiperparámetros del agente estándar de PPO *total_timesteps* es el número total de pasos coleccionados (el largo del proceso interno). *num_steps* es el tamaño del lote de datos con el que lleva a cabo las actualizaciones PPO. Acá aclaramos que en el último paso no lleva a cabo una actualización. Solo hace dos actualizaciones (una en el paso 2000 y otra en el 4000).

num_epsiodes_of_validation y *num_lifetimes_for_validation* hacen referencia a cuántos de los últimos episodios de cada proceso interno y cuantos procesos internos se utilizaron para validar el desempeño de los algoritmos durante la etapa de entrenamiento. Este desempeño de validación fue el utilizado para determinar que versión del modelo guardar. Todos los algoritmos se entrenaron hasta no mostrar ninguna mejora apreciable de desempeño por al menos 200 actualizaciones del agente externo.

A lo largo del entrenamiento externo, los refuerzos extrínsecos obtenidos en cada proceso interno fueron normalizados para que su valor promedio se mantuviera alrededor de *e_rewards_target_mean*. Esta normalización se llevó a cabo al nivel de cada clase de problema. Para lograr esto al entrenar en el benchmark ML10, se mantuvo un promedio independiente de los refuerzos extrínsecos recibidos en cada una de las 10 clases de problema. Con esta normalización, las señales de entrenamiento del meta agente (refuerzos extrínsecos) se mantienen en el mismo rango durante el entrenamiento. También, para el benchmark ML10, la normalización hace que el meta agente le de pesos similares a mejorar en cada una de las distintas clases de problemas.

Los hiperparámetros que comienzan con *ae* y *meta_gamma* controlan qué variante del objetivo 2.4 se utiliza y cómo se estiman las ventajas para las actualizacion de los agentes externos. Entre ellos, los más importantes conceptualmente son *ae.estimation_method* y *meta_gamma*. *ae.estimation_method* controla si se utilizan descuentos al nivel de los episodios o al nivel de cada paso y también controla qué método se utiliza para estimar las ventajas. En particular *'standard GAE'* es GAE [48] con descuentos por paso y *'bootstrapping skipping uninfluenced future rewards'* utiliza descuentos por episodio y estima las ventajas con una combinación exponencial de estimaciones a n-pasos. Al estimar ventajas para un paso, este método ignora todos los refuerzos extrínsecos que el agente recibe entre ese paso y el siguiente paso en el cual el agente interno es actualizado. En el contexto del agente de refuerzos intrínsecos esto significa que para estimar que tan bueno fue un refuerzo intrínseco otorgado se ignora el desempeño de la política interna hasta el paso donde utiliza este refuerzo intrínseco para llevar a cabo una actualización. *meta_gamma* indica el factor de descuento para el aprendizaje externo ; ya sea por paso o episódico.

initial_std controla la desviación estándar inicial de las cantidades estimadas por los meta agentes. Sean éstas acciones, ventajas o refuerzos intrínsecos.

Los hiperparámetros que comienzan con *ppo* para los algoritmos meta tienen influencias similares a los hiperparametros de una actualización PPO estándar pero no idénticas. Las actualizaciones de PPO para los procesos externos utilizan retro-propagación truncada. En particular *ppo.k* controla cada cuántos pasos de interacción se corta la propagación de gradientes. *ppo.num_minibatches* controla cada cuántas secuencias de largo k procesadas se ejecuta un paso de gradiente. El valor *"0"* indicado en las tablas fue el utilizado en la implementación para indicar la ejecución de un paso de gradiente por cada secuencia de largo k. Por 'una secuencia de largo k' nos referimos a una por cada uno de los *num_inner_loops_per_update* procesos internos involucrados en ese paso de actualización del proceso externo.

Para la búsqueda de hiperparámetros en una primera etapa se llevó a cabo una búsqueda aleatoria sobre un subconjunto amplio de hiperparámetros utilizando rangos de valores alrededor de valores encontrados en la bibliografía. Luego, con esta información en una segunda etapa se hizo una búsqueda manual principalmente sobre los hiperparámetros que mostraron mayor efecto sobre el desempeño de los algoritmos: *initial_std* , *entropy_coef* , *e_rewards_target_mean*, *num_inner_loops_per_update* y las tasas de aprendizaje.

Más detalle de los hiperparámetros mencionados y sobre los no mencionados se puede encontrar en el código del proyecto. Destacamos también que en las tablas no se encuentran los detalles completos de las arquitecturas de las redes. Estas se encuentran en 3.2.2.

A.4. Otros detalles de implementación

Todas las implementaciones de los algoritmos se hicieron en Pytorch. Los hiperparámetros cuyo valor no fue mencionado tomaron el valor por defecto que les impone Pytorch. Se utilizó Ray para correr las distintas tareas correspondientes a una actualización paralelamente en distintos procesos. Los benchmarks de Meta World utilizados son los de la version 2.

Aunque no forma parte del trabajo presentado en esta tesis, el código del proyecto también incluye la posibilidad de entrenar los algoritmos considerando refuerzos extrínsecos moldeados también en la evaluación (esto se ve reflejado en la lista de hiperparámetros). Se probó esto para RL^2 en ML10 y los experimentos mostraron un desempeño equivalente al presentado en [15]. Esto no se probó para MAML, pero se encontró que los resultados presentados en este trabajo (con refuerzos dispersos) son igual o incluso ligeramente superiores a los que se obtienen en [15] con refuerzos moldeados.

B. Apéndice de resultados

En esta sección del apéndice se complementan los resultados introducidos en la sección 4. A continuación presentamos una tabla que resume los desempeños finales obtenidos con todos los algoritmos implementados sobre los distintos benchmarks.

Algoritmo	**ML1 close door**	**ML1 reach**	**ML1 button press**	**ML10**
PPO estándar (refuerzos moldeados)	0.127 ± 0.023 0.117 ± 0.018	0.187 ± 0.021 0.192 ± 0.026	0.028 ± 0.009 0.033 ± 0.013	*
PPO estándar (refuerzos dispersos)	0.031 ± 0.008 0.030 ± 0.011	0.029 ± 0.007 0.027 ± 0.009	0.003 ± 0.003 0.002 ± 0.003	*
Refuerzos intrínsecos	0.833 ± 0.022 0.854 ± 0.031	0.230 ± 0.019 0.232 ± 0.027	0.186 ± 0.074 0.213 ± 0.056	0.114 ± 0.001 0.024 ± 0.013
Ventajas estimadas	*	0.249 ± 0.049 0.253 ± 0.028	0.429 ± 0.007 0.464 ± 0.056	0.134 ± 0.001 0.024 ± 0.005
RL^2	*	0.722 ± 0.148 0.290 ± 0.028	0.999 ± 0.002 0.998 ± 0.002	0.484 ± 0.078 0.023 ± 0.016
MAML (aprox. a primer orden)	*	0.866 ± 0.016 0.850 ± 0.021	0.999 ± 0.002 0.998 ± 0.002	0.587 ± 0.006 0.338 ± 0.006

Cuadro B.1: Rendimiento de algoritmos en diferentes benchmarks. Cada celda contiene la tasa de éxito promedio en tareas de entrenamiento (arriba) y tareas de evaluación (abajo) con sus respectivas desviaciones estándar. Todos los resultados son tras un periodo de adaptación de 8 episodios. Los valores y sus desviaciones estándar se obtuvieron según fue explicado en la sección 3.2.3

.