Introducción al aprendizaje reforzado

M. Sc. Saúl Calderón Ramírez
Instituto Tecnológico de Costa Rica,
Escuela de Computación.

27 de mayo de 2019

El presente documento introduce el aprendizaje reforzado. Basado en el libro *Reinforcement Learning: An Introduction de Richard S. Sutton* y los tutoriales `https://bit.ly/2SXtSgY` y `https://towardsdatascience.com/cartpole-introduction-to-reinforcement-learning-ed0eb5b58288`

1. Procesos de decisión de Markov

En el aprendizaje reforzado, el agente tiene por objetivo **maximizar una señal de recompensa, y no se le *enseña* explícitamente qué acciones tomar**, en vez de ello, el agente debe aprender **cuáles acciones le reditúan una mayor recompensa**. Existen dos características básicas de este modelo de aprendizaje:

1. Búsqueda por prueba y error
2. Recompensa retardada.

El concepto de **recompensa retardada** se refiere a recompensa en instantes de tiempo futuros, de acciones que sucedieron en una ventana de tiempo definida. El aprendizaje por refuerzo tiene como diferencia sobre el aprendizaje supervisado, la inexistencia de etiquetas o valores deseados a la salida del modelo, necesarias en el aprendizaje supervisado. En entornos interactivos, es frecuentemente **impráctico obtener ejemplos de la conducta deseada en todas las situaciones en las que el agente debe interactuar**, por lo que en un *terreno no explorado* es más adecuado el aprendizaje del agente de su **propia experiencia anterior**. El aprendizaje por refuerzo también es distinto al aprendizaje no supervisado, puesto que el objetivo de este último enfoque es encontrar la estructura de los datos sin las etiquetas, por lo que las funciones de pérdida a optimizar, se diseñan respecto a tal objetivo, mientras que en el aprendizaje reforzado, el agente tiene por objetivo maximizar una función de recompensa, usualmente modelada a partir de distintos objetivos en la interacción con el ambiente.

Un balance usualmente a realizar en la implementación de los algoritmos de aprendizaje por refuerzo es el de **explotación-exploración,** tal como sucede en los algoritmos de optimización y búsqueda. Un algoritmo que explota mucho su conocimiento, toma decisiones seguras, pero tiende a aprender políticas localmente óptimas en el mejor de los casos, mientras que si el agente favorece por completo la exploración, desprecia su experiencia anterior por explorar nuevas secuencias de acciones, aumentando el riesgo de fallar en lograr su objetivo. Es entonces una heurística común balancear ambos procesos con base a la experiencia, puesto que desarrollos teóricos en el área de la optimización no han logrado dar con respuestas concretas de cómo realizar tan balance.

Como se ilustra en la Figura 1, el aprendizaje por refuerzo define un agente que actúa en un entorno y recibe recompensas como consecuencia de esas acciones. Lo anterior se puede modelar a través de un **proceso de decisión de Markov** (PDM) el cual formalmente consiste en:

- Un **conjunto de estados posibles** $S = \{\overrightarrow{s}_1, \overrightarrow{s}_2, \ldots, \overrightarrow{s}_M\}, \overrightarrow{s}_i \in \mathbb{R}^D$„ donde un estado modela las D variables más importantes del ambiente donde actúa el agente $\overrightarrow{s}_i = [s_{i,1}, \ldots, s_{i,D}]^t$. El agente tiene la posibilidad de sensar u observar las D variables que componen tal estado. La secuencia de estados visitados durante un episodio e_i está dado por el estado visitado por cada instante t de tal episodio: $e_i = [\overrightarrow{s}(1), \overrightarrow{s}(2), \ldots, \overrightarrow{s}(\tau_i)]$ con τ_i instantes en total.

- La **experiencia** de un agente está compuesta por un conjunto de episodios $E = \{e_1, e_2, \ldots, e_K\}$. Cada episodio e_i tiene un largo τ_i, determinado por el logro o fallo total del agente en esa experiencia específica.
- Un **conjunto de acciones posibles** $A = \{a_1, a_2, \ldots, a_N\}$. El agente realiza una acción de tal conjunto para un tiempo t específico, por lo que entonces $a(t) \in A$.
- Un **conjunto de recompensas posibles** $R = \{r_1, r_2, \ldots, r_L\}$. El patrón de los valores de recompensa en el tiempo está dado por la **función de recompensa** $r(t) \in R$. En cada instante de tiempo t, el ambiente envía una señal de recompensa, usualmente un escalar, en función del estado actual y la acción realizada en ese momento por el agente, por lo que entonces $r(t) = f_r(\overrightarrow{s}(t), a(t))$. La función de recompensa genera un estímulo a corto plazo en el agente, comparable con el sentimiento de dolor y placer en el corto plazo para el ser humano. El objetivo del agente es maximizar la recompensa total en su experiencia o *episodio* e_i, de forma que $\rho_i = \sum_t^\tau r(t)$ corresponde a la recompensa del episodio, y $\rho = \sum_i^K \rho_i$ la recompensa total para el agente.
  - La **función de valor** $f_v(\overrightarrow{s}(t), a(t)) \in \mathbb{R}$ estima la recompensa total a **largo plazo** de fijar el estado $\overrightarrow{s}(t)$. Es análogo al juicio de un ser humano de encontrar una experiencia dolorosa o de placer a largo plazo. Tal función de valor necesita estimar constantemente el *valor* de un estado y acción específicos. La efectividad de una técnica específica de aprendizaje por refuerzo depende entonces de la calidad de la estimación por la función de valor.
- Una **política** p_i define el comportamiento del agente en un instante del tiempo t. La política se puede definir como la asociación de un estado en un tiempo t con una acción en el mismo instante, de forma que $\langle \overrightarrow{s}(t), a(t) \rangle$, lo cual se puede modelar en el funcional

  $$a(t) = p_i(\overrightarrow{s}(t)).$$

  En psicología la política corresponde con asociaciones estímulo-respuesta. La política puede ser tan sencilla como una tabla de búsqueda, o compleja involucrando tareas de búsqueda, métodos estocásticos, aprendizaje profundo o mezclas de tales enfoques.
- Un **conjunto de probabilidad de transiciones** $T = \{s_{1,1}, s_{1,2}, \ldots, s_{1,M}, s_{2,1}, s_{2,2}, \ldots, s_{2,M}, \ldots, s_{M,M}\}$. Tales probabilides se pueden construir a partir de las acciones seguidas según una política del agente para un conjunto de episodios.
- Finalmente, algunas técnicas de aprendizaje reforzado implementan un **modelo del ambiente,** el cual facilita la estimación de recompensas y estados futuros, y el consecuente planeamiento explícito basado en tal modelo. Las técnicas de aprendizaje por refuerzo que modelan el ambiente se les llama **técnicas basadas en modelos,** mientras que las técnicas que prescinden de él se refieren como **técnicas libres de modelos,** usualmente basadas de forma exclusiva en el aprendizaje de prueba-y-error.

2. Ejemplo: El péndulo invertido

La Figura 2 muestra un carro con lo que se conoce como un péndulo invertido. El **agente controla la fuerza aplicada sobre el carro**, con lo cual puede moverlo en **dos direcciones, derecha o izquierda**. El **objetivo del agente es mantener el péndulo arriba (sin que caiga) el máximo de tiempo posible**, o *sobrevivir* el mayor tiempo posible. Para este ejemplo se utilizará la librería *OpenAI Gym* disponible de forma libre.

Para modelar este problema, se define un estado $\overrightarrow{s}(t) \in \mathbb{R}^4$ como un vector de **cuatro observaciones correspondiente a la posición del carro, su velocidad, el ánguno del péndulo y su velocidad en la punta**, como detalla la Tabla 1. La combinación de los valores válidos generan el conjunto de estados posibles S (observe que se ignora la relación física de las variables, prescindiendo de un modelo determinístico del mismo).

Las acciones posibles son dos, empujar el carro a la **izquierda o la derecha**, por lo que el conjunto de acciones posibles está dado por $A = \{a_1 = 0, a_2 = 1\}$, respectivamente.

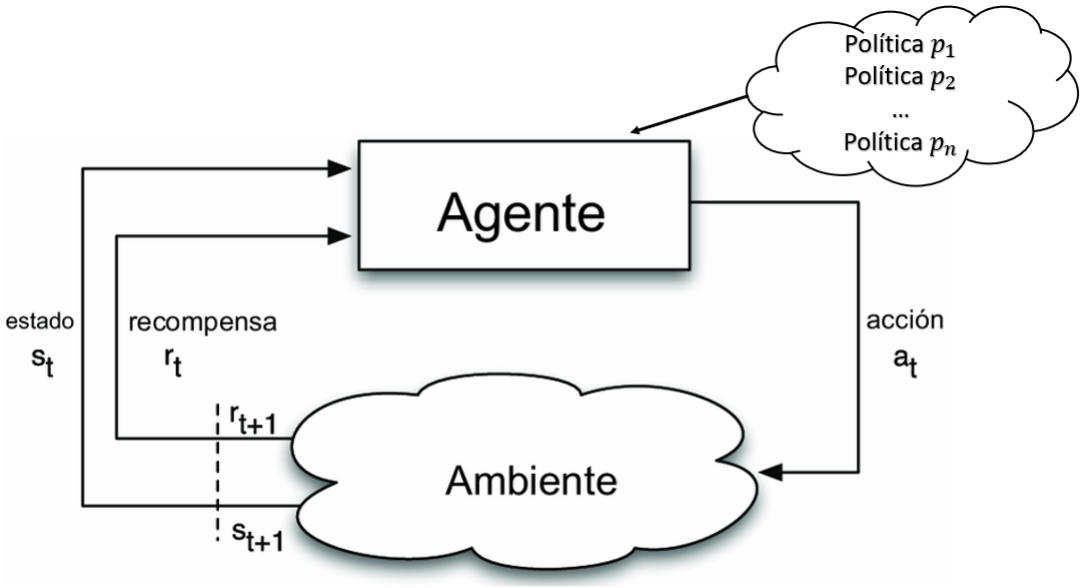

Figura 1: Modelado de Markov de agente, tomado de *Estudios sobre sistemas adaptativos con aplicaciones en la robótica autónoma y los agentes inteligentes*.

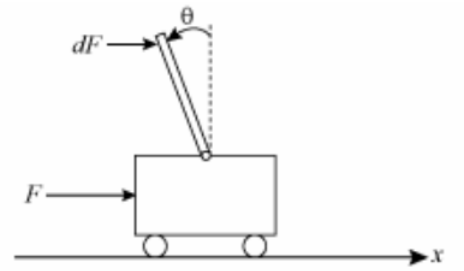

Figura 2: Carrito con el péndulo invertido, video de carrito con péndulo invertido real `https://youtu.be/XiigTGKZfks`.

Componente	Observación	Valor min.	Valor max.
$s_1(t)$	Posición del carro	-2.4 mts	2.4 mts
$s_2(t)$	Velocidad del carro	$-\infty$	$+\infty$
$s_3(t)$	Angulo del péndulo	$-41{,}8^{o}$	$+41{,}8^{o}$
$s_4(t)$	Velocidad del péndulo en la punta	$-\infty$	$+\infty$

Cuadro 1: Estados y rango de valores por cada estado.

Estado/Accion	$a_1 = 0$	$a_2 = 0$
$\overrightarrow{s}_1(t)$		
$\overrightarrow{s}_2(t)$		
$\vdots$		
$\overrightarrow{s}_M(t)$		

Cuadro 2: Tabla Q para el problema del carrito con péndulo invertido.

A continuación se presentan dos enfoques distintos para implementar la política del agente, una basada en el **aprendizaje profundo**, y otra, en el **aprendizaje estadístico**, específicamente la **maximización de la verosimilitud**.

La recompensa es de 1 por cada movimiento donde el péndulo no ha caído y -1 cuando falla, por lo que $R = \{r_1 = 1, r_2 = -1\}$. El episodio finaliza cuando el péndulo tiene más de 15 grados de la vertical o el carro se mueve más de 2.4 unidades desde el centro.

2.1. Aprendizaje de calidad o *Q-learning*

El aprendizaje de calidad define la **función de valor con el nombre de *función de calidad* Q**. De esta forma, la función recibe tanto el estado actual como la acción realizada en el tiempo t

$$Q\left(\overrightarrow{s}\left(t\right), a\left(t\right)\right) \in \mathbb{R}$$

y da como salida una **estimación de la calidad de la combinación de esa acción con ese estado**. La calidad corresponde entonces a un indicador de que *tan buena* es una acción dado un estado, y se almacena en una *tabla Q*, como la ilustrada para el presente problema. La tabla define **una fila por estado posible $\overrightarrow{s}\left(t\right)$ y una columna por cada acción** $a\left(t\right)$, con todas las entradas inicializadas de forma aleatoria o siguiendo alguna heurística, de forma que cada entrada corresponde a la recompensa futura esperada al realizar tal acción en un estado específico, como se ilustra en la Tabla 2 para el problema actual. Observe que la cantidad de estados posibles es muy alta, pues es la combinación posible de todos los valores de cada observación que compone al estado. Para actualizar cada entrada de la tabla, el algoritmo implementa la siguiente ecuación, utilizando uno o más episodios $e_i = [\overrightarrow{s}\left(1\right), \overrightarrow{s}\left(2\right), \ldots, \overrightarrow{s}\left(\tau_i\right)]$ de experiencia:

$$Q^{\text{nuevo}}\left(\overrightarrow{s}\left(t\right), a\left(t\right)\right) = \left(1-\alpha\right) Q\left(\overrightarrow{s}\left(t\right), a\left(t\right)\right) + \alpha\left(r\left(t\right) + \gamma \max_{a}\left\{Q\left(\overrightarrow{s}\left(t+1\right), a\left(t\right)\right)\right\}\right) \tag{1}$$

donde:

- El **coeficiente de aprendizaje** $0 \leq \alpha \leq 1$ define la ***inercia* o peso de la anterior estimación del valor de calidad** para un estado y acción específico, favoreciendo la **explotación cuando α es menor, y la exploración** cuando es mayor. Al inicio del entrenamiento se fija un valor alto, lo cual se conoce como una **estrategia voraz**.
- El término $\max_a \left\{Q\left(\overrightarrow{s}\left(t+1\right), a\left(t\right)\right)\right\}$ busca la acción que maximice la calidad dado el estado siguiente que se visitó en la serie de estados de uno o varios episodios $e_i = [\overrightarrow{s}\left(1\right), \overrightarrow{s}\left(2\right), \ldots, \overrightarrow{s}\left(\tau_i\right)]$. Para simplificar la notación, se define el vector de calidades para todas las acciones $\overrightarrow{q}\left(\overrightarrow{s}\left(t+1\right)\right) = Q\left(\overrightarrow{s}\left(t+1\right)\right)$
- El **factor de descuento** $0 \leq \gamma \leq 1$ da un mayor peso a la estimación de la calidad futura, **valorando más recompensas a futuro**.

Sin embargo, observe que **actualizar las entradas** en una tabla con muchísimos estados para el problema del carrito con el péndulo invertido se vuelve **inviable**, pues la tabla tiene una naturaleza *rala* al dificultarse pasar por la mayoría de los estados posibles en múltiples episodios; un modelo completamente discreto es **poco factible de implementar**. En vez de ello, se puede optar por construir una **red neuronal que dado un estado y una acción, estime el valor Q, entrenándola con los valores Q estimados por los estados que se hayan visitado en una serie de episodios anteriores**. Nos referiremos a tal red de forma general como el **modelo Q**.

Por ello, para reemplazar la tabla Q, se define una red neuronal con:

- D dimensiones a la entrada, de forma que la red para un estado $\overrightarrow{s}(t)$ estimará los valores de calidad (o valores Q).
- N dimensiones a la salida, para estimar por cada acción posible $a(t) \in A$, su valor de calidad Q.
- El modelo está dado por $\overrightarrow{q}(t) = Q(\overrightarrow{s}(t+1))$, prediciendo para la acción siguiente en ese episodio la calidad de todas las acciones

Para decidir cuál acción tomar, al inicio, cuando la tabla Q está vacía, y por tanto, se puede confiar menos en ella, por lo que se establece el **coeficiente de exploración** $\epsilon(1) = 1$ incialmente, **favoreciendo la elección de una acción aleatoria** (correspondiente a la exploración de acciones desconocidas). **Conforme se construye una mejor tabla Q** o modelo Q de estimación de la calidad, se **decrementa progresivamente** $\epsilon(t)$, usando por ejemplo un decaimiento lineal con coeficiente $\beta = 0{,}995$:

$$\epsilon(t+1) = \beta\,\epsilon(t).$$

De esta forma, la siguiente ecuación expresa el algoritmo para elegir la acción a realizar:

$$a(t) = \begin{cases} a_r & p(X < \epsilon(t)) \\ \text{máx}_a \{\overrightarrow{q}(t) = Q(\overrightarrow{s}(t+1))\} & p(X \geq \epsilon(t)) \end{cases} \tag{2}$$

con a_r una acción escogida aleatoriamente y p una función de densidad aleatoria uniforme.

Tomando en cuenta los conceptos anteriores, los pasos básicos del algoritmo de aprendizaje de calidad son los siguientes:

1. Inicializar aleatoriamente la tabla Q, o el modelo Q.
2. Para una cantidad de episodios K realizar lo siguiente en cada episodio e_i:
   a) Por cada unidad de tiempo t hasta terminar el episodio e_i y llegar al máximo de unidades de tiempo τ_i (en el caso del carro con péndulo, hasta que el péndulo se caiga o el agente falle):
      1) **Estimar la acción a realizar** $a(t)$ usando la ecuación 2 y **sensar del ambiente el siguiente estado** $s(t+1)$ **además de la recompensa en esa unidad de tiempo** $r(t)$.
      2) **Guardar el cuarteto** $c(t) = \langle s(t), a(t), r(t), s(t+1)\rangle$ en una lista la cual llamaremos experiencia global $E_{\text{global}} = [c(1), \ldots, c(t)]$ la cual guarda todos los cuartetos generados en todos los episodios vividos por el agente.
      3) Para un tamaño de lote de cuartetos B tomados de la experiencia global E_{global} construída hasta el momento t y episodio e_i, realizar lo siguiente por cada cuarteto $c(t)$:
         a′ Utilizar el modelo Q para estimar los valores de calidad para todas las acciones, haciendo $\overrightarrow{q}(t) = Q(\overrightarrow{s}(t+1))$, donde en este caso el estado siguiente $s(t+1)$ está definido en el cuarteto.
         b′ Al utilizarse un modelo Q, la política de actualización y *qué tanto peso* se le otorga al valor estimado por la ecuación de actualización de la calidad 1, expresado en el coeficiente de aprendizaje α, lo define internamente el modelo Q, en este caso la red neuronal. Es por ello que el de calidad $q'_{a_{\text{mejor}}}(t)$ viene dado simplemente por:

$$q'_{a_{\text{mejor}}}(t) = r(t) + \gamma \max_a \{Q(\overrightarrow{s}(t+1), a(t))\}$$

         c′ Se **actualiza el vector de calidad** para las acciones tomando en cuenta tal estimación de forma que $q_{a_{\text{mejor}}}(t) = q'_{a_{\text{mejor}}}(t)$, creando un nuevo vector $\overrightarrow{q}'(t)$.
         d′ Se **entrena o ajusta el modelo con la nueva muestra** $\overrightarrow{q}'(t)$, usando entonces un enfoque de entrenamiento en línea del modelo.

3. Aprendizaje por máxima verosimilitud

Para el aprendizaje por máxima verosimilitud, modelamos la probabilidad de realizar un movimiento a la izquierda $a(t) = 0$ y la derecha $a(t) = 1$ con una función de densidad condicional de Bernoulli o binomial, de realizar una acción, dado un estado $\overrightarrow{s}(t)$:

$$p_{\text{accion}}\left(a(t)\,|\,\overrightarrow{s}(t)\right) = p\left(\overrightarrow{s}(t)\,|\,\overrightarrow{\omega}\right)^{a(t)}\left(1 - p\left(\overrightarrow{s}(t)\,|\,\overrightarrow{\omega}\right)\right)^{1-a(t)}$$

donde observe que si por ejemplo $a(t) = 1$, $p_{\text{accion}}\left(a(t)\,|\,\overrightarrow{s}(t)\right) = p\left(\overrightarrow{s}(t)\,|\,\overrightarrow{\omega}\right)^{a(t)}$, y para el movimiento a la izquierda, tal probabilidad es el complemento. Esta probabilidad vendría a constituir la **política del agente, como una probabilidad condicional**:

$$a(t) = p_i\left(\overrightarrow{s}(t)\right).$$

El vector $\overrightarrow{\omega}$ define los **parámetros del modelo** a ajustar, y tiene la misma dimensionalidad del estado del sistema, por lo que $\overrightarrow{\omega} \in \mathbb{R}^D$. Tales parámetros controlan el comportamiento de la función de densidad $p\left(\overrightarrow{s}(t)\,|\,\overrightarrow{\omega}\right)$ y constituyen la **política del agente**. Dado que tal función de densidad debe tener valores entre 0 y 1, escogemos la función sigmoidal como modelo:

$$p\left(\overrightarrow{s}(t)\,|\,\overrightarrow{\omega}\right) = \text{sig}\left(\overrightarrow{\omega}, \overrightarrow{s}(t)\right) = \frac{1}{1 + e^{\left(-\omega \cdot \overrightarrow{s}(t)\right)}}$$

y recordemos que su gradiente respecto a $\overrightarrow{\omega}$ está dado por:

$$\nabla_{\overrightarrow{\omega}}\text{sig}\left(\overrightarrow{\omega}, \overrightarrow{s}(t)\right) = \text{sig}\left(\overrightarrow{\omega}, \overrightarrow{s}(t)\right)\left(1 - \text{sig}\left(\overrightarrow{\omega}, \overrightarrow{s}(t)\right)\right)\overrightarrow{s}(t)$$

El objetivo más simple es encontrar los parámetros $\overrightarrow{\omega}$ que maximicen la verosimilitud dados los episodios de aprendizaje del agente $E = \{e_1, e_2, \ldots, e_B\}$ de un lote de B episodios (equivalentes a algo así como el conjunto de muestras de entrenamiento), con una cantidad de unidades de tiempo total $T = \sum_i^K \tau_i$,

$$L\left(\overrightarrow{\omega}|E\right) = \prod_{t=1}^{T} p\left(\overrightarrow{s}(t)\,|\,\overrightarrow{\omega}\right)^{a(t)}\left(1 - p\left(\overrightarrow{s}(t)\,|\,\overrightarrow{\omega}\right)\right)^{1-a(t)}$$

Recordemos que la **maximización de la verosimilitud equivale a la minimización del error del modelo**, y que además el tomar el logaritmo natural de la función de verosimilitud facilita su análisis y cómputo, lo que define nuestra función de costo o pérdida $l\left(\overrightarrow{\omega}, E\right)$:

$$l\left(\overrightarrow{\omega}, E\right) = -\ln\left(L\left(\overrightarrow{\omega}|E\right)\right) = -\frac{1}{B}\sum_{t}^{T} a(t)\ln\left(p\left(\overrightarrow{s}(t)\,|\,\overrightarrow{\omega}\right)\right) + \left(1 - a(t)\right)\ln\left(1 - p\left(\overrightarrow{s}(t)\,|\,\overrightarrow{\omega}\right)\right).$$

Con el objetivo de normalizar la magnitud del error respecto a la cantidad de episodios del lote, se divide la sumatoria por B. Para tomar en cuenta las recompensas, la función de verosimilitud agrega un **coeficiente de pesado** $w(t)$ de la **recompensa actual**, de forma que los episodios cortos tendrán valores de recompensa mas bajos, dando un menor peso a tales experiencias, implementable con por ejemplo un decaimiento lineal respecto al largo del episodio, con el peso mas alto a las primeras acciones. La

$$l\left(\overrightarrow{\omega}, E\right) = -\ln\left(L\left(\overrightarrow{\omega}|E\right)\right) = -\frac{1}{B}\sum_{t}^{T} w(t)\left(a(t)\ln\left(p\left(\overrightarrow{s}(t)\,|\,\overrightarrow{\omega}\right)\right) + \left(1 - a(t)\right)\ln\left(1 - p\left(\overrightarrow{s}(t)\,|\,\overrightarrow{\omega}\right)\right)\right).$$

Para implementar una solución generalizada, se usa el algoritmo del descenso de gradiente, en busca del mínimo de la función de pérdida $l\left(\overrightarrow{\omega}, E\right)$, el cual se calcula cada lote de B episodios:

$$\overrightarrow{\omega}(k+1) = \overrightarrow{\omega}(k) - \alpha\nabla_{\overrightarrow{\omega}} l\left(\overrightarrow{\omega}, E\right)$$

por cada iteración k. El vector gradiente $\nabla_{\overrightarrow{\omega}} l\left(\overrightarrow{\omega}, E\right)$ está dado entonces por:

$$\nabla_{\overrightarrow{\omega}}\left(-\frac{1}{B}\sum_{t}^{T} w\left(t\right)\left(a\left(t\right)\ln\left(\text{sig}\left(\overrightarrow{\omega}, \overrightarrow{s}\left(t\right)\right)\right)+\left(1-a\left(t\right)\right)\ln\left(1-\text{sig}\left(\overrightarrow{\omega}, \overrightarrow{s}\left(t\right)\right)\right)\right)\right)$$

$$= a\left(t\right)\frac{1}{\cancel{\text{sig}\left(\overrightarrow{\omega}, \overrightarrow{s}\left(t\right)\right)}}\cancel{\text{sig}\left(\overrightarrow{\omega}, \overrightarrow{s}\left(t\right)\right)}\left(1-\text{sig}\left(\overrightarrow{\omega}, \overrightarrow{s}\left(t\right)\right)\right)-\left(1-a\left(t\right)\right)\frac{1}{\cancel{1-\text{sig}\left(\overrightarrow{\omega}, \overrightarrow{s}\left(t\right)\right)}}\text{sig}\left(\overrightarrow{\omega}, \overrightarrow{s}\left(t\right)\right)\cancel{\left(1-\text{sig}\left(\overrightarrow{\omega}, \overrightarrow{s}\left(t\right)\right)\right)}$$

$$= \frac{1}{B}\sum_{t}^{T} w\left(t\right)s\left(t\right)\left(\left(1-a\left(t\right)\right)\text{sig}\left(\overrightarrow{\omega}, \overrightarrow{s}\left(t\right)\right)-a\left(t\right)\left(1-\text{sig}\left(\overrightarrow{\omega}, \overrightarrow{s}\left(t\right)\right)\right)\right)$$

Al ejecutar el algoritmo, es de notar la necesidad de muchos episodios para lograr un comportamiento satisfactorio.