Redes neuronales en aprendizaje por refuerzo

Luis Ballado

CINVESTAV - UNIDAD TAMAULIPAS

luis.ballado@cinvestav.mx

February 28, 2023

Contenido

Introducción

Los origines del aprendizaje por refuerzo son variados, pero uno de los origenes se toma de la psicologia de aprendizaje animal.

La idea básica es la de premiar al aprendiz (agente) por las acciones correctas y castigar las acciones incorrectas.

Intuitivamente, el Aprendizaje Reforzado es un proceso de prueba y error, combinado con aprendizaje. El agente decide las acciones a tomar en base al estado del ambiente y atravez de las retroalimentaciones en terminos de las acciones esperadas, decimos que aprende siendo esa acción la mejor asociada con el presente estado.

El agente aprende de la interacción con el ambiente.

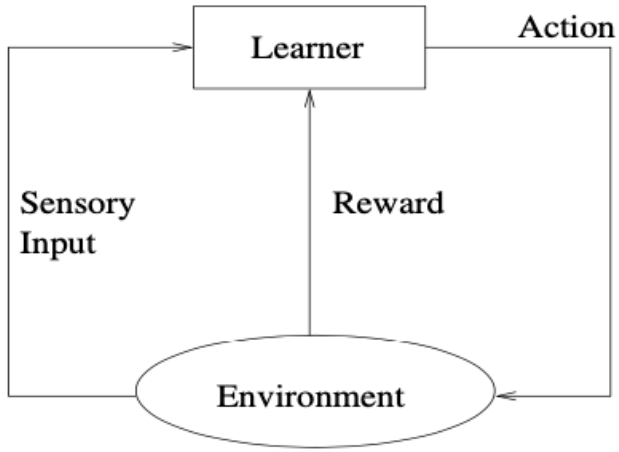
Aprendiendo a través de premios

Definido formalmente, el **aprendizaje por refuerzo** es el aprendizaje de un mapeo de situaciones a acciones con el objetivo principal de **maximizar la recompensa escalar o la señal de refuerzo**. De manera informal, el aprendizaje por refuerzo se define como el aprendizaje por **ensayo y error** a partir de la retroalimentación del desempeño del entorno o de un evaluador externo.

El agente no tiene absolutamente ningún conocimiento previo de qué acción tomar y tiene que descubrir (o explorar) qué acciones producen la mayor recompensa.

El agente recibe entradas sensoriales de su entorno, como una descripción del estado actual del entorno percibido. Se ejecuta una acción, sobre la cual el agente recibe la señal de refuerzo o recompensa. Esta recompensa puede ser una señal positiva o negativa, dependiendo de la corrección de la acción. Una recompensa negativa tiene el efecto de castigar al agente por una mala acción.

Figure: Problema de aprendizaje por refuerzo



La acción puede provocar un cambio en el entorno del agente, afectando así las opciones y acciones futuras del agente. Los efectos de las acciones sobre el medio ambiente y los estados futuros no siempre se pueden predecir. Por lo tanto, es necesario que el agente monitoree frecuentemente su entorno.

El Aprendizaje por refuerzo tiene dos importantes componentes:

1. Una búsqueda de prueba y error para encontrar buenas acciones, que forma el componente de exploración de Aprendizaje por Refuerzo.
2. Un recuerdo de qué acciones funcionaron bien en qué situaciones. Este es el componente de explotación de Aprendizaje por Refuerzo.

Un agente de aprendizaje por refuerzo tiene los siguientes componentes:

1. **Una política**, que es la función de toma de decisiones del agente.
2. **Una función de recompensa**, que define el objetivo del agente.
3. **Una función de valor**, que especifica el objetivo a largo plazo.
4. Opcionalmente, un agente de aprendizaje reforzado también puede tener un **modelo del entorno.**

Se han propuesto varios modelos:

1. **finite-horizon model**, en el que el agente optimiza su recompensa esperada para los siguientes n_t pasos, es decir

$$E\left[\sum_{t=1}^{n_t} r(t)\right]$$

2. **infinite-horizon discounted model**, que toma en consideración toda la recompensa a largo plazo del agente. Sin embargo, cada recompensa recibida en el futuro se descuenta geométricamente de acuerdo con un factor de descuento, $\gamma \in [0, 1)$:

$$E\left[\sum_{t=0}^{\infty} \gamma^t r(t)\right]$$

3. **average reward model**, que prefiere acciones que optimicen la recompensa promedio a largo plazo del agente:

$$\lim_{n_t \to \infty} E\left[\frac{1}{n_t} \sum_{t=0}^{\infty} r(t)\right]$$

Modelo de aprendizaje por refuerzo sin modelo

El objetivo es obtener una póliza óptima sin un modelo del entorno

Aprendizaje de diferencias temporales, aprende la política de valor usando la regla de actualización

$$V(s) = V(s) + \eta(r + \gamma V(s') - V(s))$$

donde η es una tasa de aprendizaje, r es la recompensa inmediata, γ es el factor de descuento, s es la estado actual, y s es un estado futuro. Con base en la ecuación, siempre que un estado, s, se visita, su valor estimado se actualiza para estar más cerca de $r + \eta V(s')$

Q-Learning

la tarea es aprender los valores de refuerzo descontados esperados, $Q(s, a)$, de realizar la acción a en el estado s, y luego continuar eligiendo siempre las acciones de manera óptima. Para relacionar los valores Q con la función de valor, tenga en cuenta que

$$V*(s) = maxQ*(s, a)$$

Donde $V*(s)$ es el valor de s asumiendo que la mejor acción se toma inicialmente

La regla de Q-learning se da como

$$Q(s, a) = Q(s, a) + \eta(r + \gamma maxQ(s', a') - Q(s, a))$$

Luego, el agente realiza la acción con el valor Q más alto.

Redes neuronales y aprendizaje por refuerzo

Las redes neuronales y el aprendizaje por refuerzo se han combinado de varias maneras. Un enfoque para combinar estos modelos es utilizar un NN como aproximador de la función de valor utilizada para predecir la recompensa futur. Otro enfoque utiliza RL para ajustar los pesos

Otros enfoques para usar RL para el entrenamiento de NN incluyen RPROP y el descenso de gradiente en la recompensa esperada. El Q-learning conexionista se utiliza para aproximar la función de valor.

RPROP

La propagación resiliente (RPROP) realiza una adaptación directa del peso paso usando información de gradiente local. Los ajustes de peso se implementan en el forma de recompensa o castigo, como sigue:

Si la derivada parcial, $\frac{\partial E}{\partial v_{ji}}$ (o $\frac{\partial E}{\partial w_{kj}}$) de peso v_{ji} (o w_{kj}) cambia de signo, el valor de actualización del peso, Δ_{ji} (Δ_{kj}), se reduce por el factor, η^-. El motivo de esta penalización es que la última actualización de peso fue demasiado grande, lo que provocó que el algoritmo saltara por encima de un mínimo local. Por otro lado, si la derivada conserva su signo, el valor de actualización se incrementa por el factor η^+ para acelerar la convergencia.

Algorithm 6.1 RPROP Neural Network Training Algorithm

Initialize NN weights to small random values;
Set $\Delta_{ji} = \Delta_{kj} = \Delta_0,\ \forall i = 1, \ldots, I+1,\ \forall j = 1, \ldots, J+1,\ \forall k = 1, \ldots, K$;
Let $t = 0$;
while *stopping condition(s) not true* **do**
  for *each* $w_{kj},\ j = 1, \ldots, J+1,\ k = 1, \ldots, K$ **do**
    if $\frac{\partial E}{\partial w_{kj}}(t-1)\frac{\partial E}{\partial w_{kj}}(t) > 0$ **then**
      $\Delta_{kj}(t) = \min\{\Delta_{kj}(t-1)\eta^+, \Delta_{max}\}$;
      $\Delta w_{kj}(t) = -\text{sign}\left(\frac{\partial E}{\partial w_{kj}}(t)\right)\Delta_{kj}(t)$;
      $w_{kj}(t+1) = w_{kj}(t) + \Delta w_{kj}(t)$;
    else if $\frac{\partial E}{\partial w_{kj}}(t-1)\frac{\partial E}{\partial w_{kj}}(t) < 0$ **then**
      $\Delta_{kj}(t) = \max\{\Delta_{kj}(t-1)\eta^-, \Delta_{min}\}$;
      $w_{kj}(t+1) = w_{kj}(t) - \Delta w_{kj}(t-1)$;
      $\frac{\partial E}{\partial w_{kj}} = 0$;
    else if $\frac{\partial E}{\partial w_{kj}}(t-1)\frac{\partial E}{\partial w_{kj}}(t) = 0$ **then**
      $\Delta w_{kj}(t) = -\text{sign}\left(\frac{\partial E}{\partial w_{kj}}(t)\right)\Delta_{kj}(t)$;
      $w_{kj}(t+1) = w_{kj}(t) + \Delta w_{kj}(t)$;
  end
  Repeat the above for each v_{ji} weight, $j = 1, \ldots, J,\ i = 1, \ldots, I+1$;
end

Aprendizaje por refuerzo de descenso de gradiente

Para problemas en los que solo se maximiza la recompensa inmediata (es decir, no hay una función de valor, solo una función de recompensa), Williams propuso reglas de actualización de peso que realizan un descenso de gradiente en la recompensa esperada. Estas reglas se integran luego con la propagación hacia atrás. Los pesos se actualizan de la siguiente manera:

$$\Delta w_{kj} = \eta_{kj}(r_p - \Theta_k)e_{kj}$$

Donde η_{kj} es un rango de aprendizaje no negativo, r_p es el reforzamiento asociado con el patrón z_p, Θ_k es el valor umbral de refuerzo, y e_{kj} es la elegibilidad del peso w_{kj}, dado como

$$e_{kj} = \frac{\partial}{\partial w_{kj}[ln(g_j]}$$

donde: $g_j = P(o_{k,p} = t_{k,p}|w_k, z_p$ es la función de densidad de probabilidad utilizada para generar acciones aleatoriamente, en función de si el objetivo se predijo correctamente o no.

Q-Learning conexionista

El NN se utiliza para aproximar el mapeo entre estados y acciones, e incluso para generalizar entre estados. La entrada a la NN es el estado actual del entorno y la salida representa la acción a ejecutar. Si hay na acciones, entonces se puede usar un NN con na unidades de salida, o n_a NN, uno para cada una de las acciones, se puede usar

Algorithm 6.3 $Q(\lambda)$ Connectionist Update Algorithm

Reset all eligibilities, $\mathbf{e}(t) = \mathbf{0}$;
$t = 0$;
while *stopping condition(s) not true* **do**
    Select action $a(t)$ as the one with maximum predicted Q-value;
    if $t > 0$ **then**

$$\begin{aligned}\mathbf{w}(t) &= \mathbf{w}(t-1) + \eta([r(t-1) + \gamma \max_{a \in \mathcal{A}} Q(t) - Q(t-1)]\nabla_w Q(t-1) \\ &\quad + [r(t-1) + \gamma \max_{a \in \mathcal{A}} Q(t) - \max_{a \in \mathcal{A}} Q(t-1)]\mathbf{e}(t-1)) \end{aligned} \tag{6.21}$$

    end
    Update eligibilities,

$$\mathbf{e}(t) = \lambda\gamma[\mathbf{e}(t-1) + \lambda_w Q(t-1)] \tag{6.22}$$

    Calculate $\nabla_w Q(t)$ with respect to action $a(t)$;
    Perform action $a(t)$, and receive reward, $r(t)$;
end

Algorithm 6.2 Connectionist Q-Learning Update Algorithm

Reset all eligibilities, $\mathbf{e}(t) = \mathbf{0}$;
$t = 0$;
while *stopping condition(s) not true* **do**
    Select action $a(t)$ as the one with maximum predicted Q-value;
    if $t > 0$ **then**

$$\mathbf{w}(t) = \mathbf{w}(t-1) + \eta(r(t-1) + \gamma Q(t) - Q(t-1))\mathbf{e}(t-1) \tag{6.19}$$

    end
    Calculate $\nabla_w Q(t)$ with respect to action $a(t)$;
    Update eligibilities,

$$\mathbf{e}(t) = \nabla_w Q(t) + \gamma\lambda\mathbf{e}(t-1) \tag{6.20}$$

    Perform action $a(t)$, and receive reward, $r(t)$;
end